O’ZBEKISTON RESPUBLIKASI AXBOROT TEXNOLOGIYALARI VA KOMMUNIKATSIYALARINI RIVOJLANTIRISH VAZIRLIGI

MUHAMMAD AL – XORAZMIY NOMIDAGI
TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI

“Sun’iy Intellekt”

FANI

1. Ma’ruza

Kirish

Reja:

1. Intelektual tizimlar haqida umumiy ma’lumot
2. Intelektual tizimlarning rivojlanish bosqichlari
3. Ekspert tizimlari (ET). ETning xarakteristika va maqsadlari.
4. ETning oddiy ma’lumotlarni qayta ishlash dasturlaridan farqlari.

Tayanch iboralar: *Sun’iy,intelekt, ekspert tizim.*

Ta’rif. Sun’iy intelekt(SI) – bu dasturiy muhitning shunday tizimiki, unda inson tafakkurining kompyuter jaraèniga imitatsiyalangan. Sun’iy intelekt atamasi 1956-yilda Stanford universiteti(SSHA) tomonidan taklif qilingan.

Intelekt – bu maqsadga erishishda zarur bo‘ladigan faktlar va metodlar to‘plamidan iborat. Maqsadga erishish – bu faktlarga zaruriy qoidalarni qo‘llashdan iborat.

Misol.Fakt 1. Yonayotgan ploita – issiq.

Qoida 1. AGAR qo‘lni ènaètgan plitaga tekkizilsa, U HOLDA kuyish mumkin.

Sun’iy intelekt tizimi rivojlanishining quyidagi bosqichlarini ko‘rib chiqishimiz mumkin:

1. XX asrning 70 yillari masalalarni echish metodlarini izlash va ularni universal dasturlarni qurishda foydalanish bilan xarakterlanadi.

2. XX asrning 80 yillari axborotlarni tasavvur qilishning umumiy metodlarini izlashga va ularni maxsus dasturlarga qo‘llash usullarini qidirish bilan xarakterlanadi.

3. XX asrning 90 yillari bir qancha fan soxalari bo‘yicha maxsus dasturlarni yaratish uchun katta xajmli yuqori sifatli maxsus bilimlarni qo‘llanishi bilan xarakterlanadi.

XX asrning 90-yillari boshlarida butunlay yangi konsepsiya qabul qilindi. Intelektual dasturni tuzish uchun, izlanish fan soxasining yuqori sifatli maxsus bilimlari bilan ta’minlash kerak. SHuning uchun loyixalashtirilaètgan SI tizimi yuqori bosqisdagi bilimlari bazasiga ega bo‘lishi kerak. Hozirgi vaqtda eng ko‘p tarqalgan konsepsiya bu ekspert tizimlarini (ET) loyixalashtirishdir.

Ta’rif. Ma’lum fan soxasidagi Sun’iy intelekt tizimi ekspert deb ataladi.

ET – bu aniq fan soxasidagi mutaxassislarning bilimlari to‘plovchi va kam ixtisoslashgan foydalanuvchilarga konsultatsiya uchun empirik tajribasini tirajlashtirvchi(nusxasi èzish) murakkab dasturlar kompleksi.

ET uchun bilimlarni qabul qilishda shu fan soxasidagi ekspertlar xizmat qiladi.

Ularning asosiy xususiyatlari:

1. ET masalani echish uchun yuqori sifatli tajriba va bilimni qabul qiladi;
2. ETdagi bilimlar doim to‘planib va yangilanib boriladi;
3. ET oldindan aytib berish qobilyatiga ega bo‘ladi.
4. ET ishchilarga va mutaxassislarga o‘quv qo‘llanmasi sifatida foydalanilishi mumkin.

ETni loyixalashda va ishlab chiqish jaraènida quyidagi qatnashuvchilarni aytib o‘tish mumkin:

1. ETni loyixalash instrumental muxitini ishlab chiquvchilar;
2. ETni yaratishdagi instrumental muxit(IM);
3. ETning o‘zi;
4. Ekspert;
5. Bilimlar injeneri va bilimlar bazasi(BB) administratori;
6. Foydalanuvchi.

Bilimlar injeneri – bu SI tizimini ishlab chiqishdagi ko‘nikmaga ega va ETni qanday tuzishni biladigan odam. U ekspertdan so‘raydi va BBdagi bilimlarni tashkillashtiradi.

Instrumental muxitni loyixalashga ET dasturlash tili va qo‘llab quvvatlovchi muhit(u orqali foydalanuvchi ET bilan o‘zaro ta’sirlashadi)dan iborat.

ET qatnashuvchilarining o‘zaro aloqasi.

ET asosliligini qarab chiqamiz, inson tafakkurini va SI tizimini solishtirish.

Inson tafakkur tizimi

SI tizimi

Kamchiliklari

1. Barqaror emas
2. Qiyin o‘tkazuvchi (ifodalovchi)
3. Qiyin hujjatlashtiriluvchi
4. Oldindan aytib bo‘lmaydigan
5. Qimmatli

Ustunliklari

1. Doimiy
2. Oson ifodalanuvchi
3. Oson hujjatlashtiriluvchi
4. Doim bir xil
5. Maqbul

Ustunliklari

1. Ijod qiluvchi
2. Moslashuvchan
3. Hissiy idrokdan foydalanadi
4. Har tomonlama
5. Keng qamrovli bilimdan foydalanadi

Kamchiliklari

1. Sun’iy oldindan dasturlashtirilgan
2. Aytib turish kerak
3. Belgili idrokdan foydalanadi
4. Tor yo‘nalishli
5. Maxsus bilimdan foydalanadi

Bu tizimlarni afzalliklari va kamchiliklarini tahlil qilib, inson eskpert asosiy afzalliklari, u ko‘p soxada, masalan, ijodkorlikda, topqirlikda,ma’lumot uzatishda va umuman mazmunan SIdan ustunlikka ega.

Ishlab chiqaruvchi

Ekspert IT loyixalashtirish Injener bilimi ET foydalanuvchi qurish
Foydalanyapti So‘rayapti Ishlab chiqish aniqlashtirish
ET terminologiyasi.
ETda ishlatiladigan asosiy atamalarni ko‘rib chiqamiz:

Algorit – bu optimal echim olishni ta’minlaydigan formal protsedura.

Bilimlar bazasi – bu ETning soxa bilimidan iborat qismi.

Dispetcher – bu bilimlar bazasidan qachon va qay tartibda qoidalarni qabul qilishni boshqarib turadigan mexanizm qismi.

Bilim – bu dasturda ishlatiladigan intellektual axborot.

Interpretator – bu soxa bilimini qaysi shaklda qabul qilishni boshqaradigan mexanizm qismi.

Qaror mexanizmi – bu masalalarni echish jaraènlarini umumiy sxemasini o‘zida mujassamlashtirgan ETning qismi.

Ishonchlilik koeffitsienti – bu berilgan faktlar va qoidalarni aniq hisoblash ehtimoli èki ishonchlilik darajasi belgilaydigan son.

Qoida – bu bilimni quyidagi formal shaklda berilishi: AGAR <shart>, U HOLDA <harakat>.

Ekspert tizim – bu oldidan mo‘ljallab qo‘yilgan va boshqa bilimlardan ajratilgan soxa bilimlariga asoslangan dastur.

Evristika – fan soxasidagi echim izlashni cheklaydi èki soddalashtiradigan qoida.

Semantik tarmoq – bu bilimlarni graf shaklida ko‘rsatuvchi metod, unda uch qismlari ob’ektlarni, èylar esa uning xususiyatlarini bildiradi.

Slot – bu ob’ekt xususiyati atributi tavsifi.

ETni tashkillashtirilgan strukturasi.

ET asosida BB ètadi. Bu erda bilim faktlar va qoidalarning o‘zaro bog‘liq shaklda qabul qilinadi, xoh ular to‘g‘ri bo‘lsin, xoh noto‘g‘ri èki qaysidir darajada ishonchlilikka ega bo‘lsin.

ETda ko‘p qoidalar,masalan,empirik èki tajribali qoidalar èki soddalashtirish, evristika bo‘ladi.

ET evristikani ishlatishga majbur, chunki bu erda echiladigan masalalar odatda qiyin bo‘ladi va oxirigacha tushunib bo‘lmasligi, matematik èzuvga tushmasligi mumkin.

Fan soxasi haqida saralab olingan bilimlar bazaviy bilimlar, masalani echishda qaror qabul qilishda ishlatiladiganlari esa umumiy bilimlar
deyiladi. SHunday qilib, ETdagi BB faktlar va qoidalardan iborat bo‘ladi, qaror qabul qilish mexanizmida esa yangi bilimlar uchun qoidani qanday shaklda qabul qilishni aniqlashtiruvchi interpretator va bu qoidalarni qanday tartibda qabul qilishni o‘rnatuvchi dispetcher bo‘ladi.

ETning tashkillashtirilgan sxemasini quyidagi shaklda tasvirlash mumkin:

BB
(fan soxasidagi bilimlar)

faktlar
qoidalar
interpretator
dispetcher

CHiqari mexanizmi
(masala echilishidagi umumiy bilimlar)
Oddiy axborotni qayta ishlovchi dasturlardan ETning farqli tomonlari:

1. An’anaviy komp’yuter dasturlari – ixtièriy qo‘yilgan masalaga ular doim bir xil jaraènlar ketma-ketligida èndashishadi; ET har bir qo‘yilgan masalaga xususiy echim daraxtini quradi.

2. ET ixtièriy simvolli ifodani (masalan, konseptual, makon va zamon munosabatlari) qayta ishlaydi. Agar oddiy dasturlarda maqsad – sonli qiymatlarni hisoblash, o‘zgarmaslarni to‘plash va xotiradan chiqarish bo‘lsa, ET uchun maqsad – ob’ektlar va hodisalar oqimini kuzatishda oldindan asosli ko‘rsatmalar va tavsiyalar berishdan iborat.

3. Agar an’anaviy dasturlar matematik qoidalardan kelib chiqsa, ET ishlashi esa belgilarni qayta ishlashga va evristik mulohazalarga asoslanadi.

ET inson tafakkurini, aniqlangan muammolarning faraz qilingan echimlarini imitatsiya qiladi, keyin ulardan eng mos, to‘g‘ri keladiganini tanlaydi. Bunda eng oldin keraksiz echimlarni tashlab yuboriladi. Bundanda ko‘proq, u egallangan sub’ektiv bilimlarga bog‘liq bo‘lmagan tarkibiy
tuzilishidan foydalanadi, tadqiq qilingan inson tizimini haètiy muammolar echimiga ekspertiza o‘tkazilishini qabul qilinadi. Muammolarni turli tomondan qarashni tizimli tahlili tufayli, u shundaygina to‘g‘ri keladigan emas, balki eng yaxshi echimni beradi. ET butunlay insoniy ekspertizaga bog‘liq.

Takrorlash uchun savollar:

1. Sun’iy intelektni ta’riflang?
2. Ekspert tizimi nima?
3. Bilimlar muhandisining vazifalari?
4. ET loyihalanishining instrumental muhitiga nimalar kiradi?
5. Ekspert tizimlarining oddiy dasturlardan farqlari?

2. Ma’ruza
Fan tarixi. Turli agentlar arxitekturasi

Reja:

1. Sun’iy intellekt haqidagi tasavvur
2. Sun’iy intellektli tizimlar
3. Sun’iy intellekt rivojlanishining yo’nalishlari

Tayanch iboralar: *ta’rix, arxitiktura, intellekt*

Sun’iy intellekt haqidagi tasavvur va bu sohadagi izlanishlar — «aqliy mashinalar» ishlab chiqarishga ilmiy yondoshish birinchi bo’lib Stanford universitetining (AQSH) professori Djon Makkarti tashabbusi asosida 1956 yili tashkil topgan ilmiy tugarakda paydo bo’ldi [24,34].

Bu tugarak tarkibiga Massachuset (AQSH) texnologiya oliygoxi «Elektronika va xisoblash texnikasi» kulliyotining faxriy professori Marvin Minskiy, «masalalarni universal xal qiluvchi» va «mantiqiy nazariyotchi» intellektual (aqliy) programmalar bunyodkori — kibernetik Allen Nbyuell va Karnegi-Mellen dorilfununining (AQSH) mashxur psixologi Gerbert Seyman, xisoblash texnikasining ko'zga ko'ringan mutaxassislari Artur Semuelb, Oliver Selfridj, Manshenon va boshqa¬lar kirar edilar. Aynan shu tugarakda «Sun'iy intellekt» tushunchasi paydo bo'ldi.

Ma'ruzaning asosiy mazmuniga kirishishdan avval «sun'iy intellekt» (SI), umuman «intellekt» haqidagi tushunchani aniqlab olishimiz kerak. Bu tushunchani oddiy qoida asosida tushuntirish mumkindek tuyuladi, lekin biz buni kila olmaymiz. CHunki, hozircha «intellekt» va «SI» haqida biron-bir aniq fikr yo'q. Bu tushunchani turli fan sohalarida ijod qiluvchi olimlarning talkin qilishlari turlicha, fikrlashlarida yakdillik yuk. SHu sababli bu tushunchalarning mazmunini ukuvchiga tushuntirib berishga xarakat kilamiz.

«Intellekt» so'zi lotincha «intellectus» so'zidan kelib chikkan bo'lib, u bilish (aniqlash), tushunish yoki faxmlash (aql) ma'nosini beradi.

«Intellekt» so'zini aniqlovchi, psixologlar tuzgan uchta tushunchani (Katta sovet entsiklopediyasi va Vesterning amerika lugatidan olingan) keltiramiz. Bu tushunchalar «intellekt» tushunchasi mazmunini aniqlash uchun yordam beradi.

Intellekt — fikrlash qobiliyati, ratsional bilish va shunga uxshash. Umumiy xolda esa fikrlash, shaxsni aqliy rivojlanishi sinonimi bo'lib xizmat qiladi.

Intellekt (aql) — uz xulkini sozlash yuli bilan xar qanday (ayniksa yangi) xolatga etarli baho berish qobiliyati.

Intellekt — turmushdagi dalillar o'rtasidagi o'zaro bog'liqlikni tushunish qobiliyati. Bu qobiliyat belgilan-gan maqsadga erishishga olib boruvchi xarakatlarni ishlab chikish uchun kerak bo'ladi.

YUqorida aniqlangan «intellekt» tushunchasidan shunday xulosa chiqarish mumkinki, ya'ni intellekt faqat insonlarga tegishli va odam aqliy qobiliyatining uziga xos o'lchovidir. Psixologlar shunday maxsus usullar yaratdilarki, bu usullar yordamida tajriba orqali odamning intellektual (aqliy) darajasini aniqlash mumkin bo'ldi. Natijada shu narsa aniqlandiki, intellektning individual darajasi o'rtasidan surilishi (og'ishi) odamning fizik imkoniyatlari darajasi kabidir.

Agar o'rtacha aqliy qobiliyat 100 ball deb qabul kilinsa, u xolda uta qobiliyatli insonlarda bu ko'rsatkich 150, 180, xattoki 200 ballga etish mumkin. Amerikalik shaxmatchi, jaxon eks-chempioni Robert Fisherning bu ko'rsatkichi 187 ball bo'lgan, XIX asr yarmida yashagan angliyalik mantiqchi Djon Styuart Mill uch yoshidayok qadimgi yunon tilida gapira olgan va uning ko'rsatkichi 190 ballgacha borgan.

SHuni qayd qilish lozimki, evolyutsiya davrida intellekt birmuncha bir tekis, inkilobiy rivojlanish davridan toki zamonaviy inson intellekti paydo bulgunga kadar bo'lgan davrni bosib o'tgan.

Intellektning evolyutsion rivojlanishi berilgan bosqichdan birmuncha yuqori printsipial, a'lo darajadagi tashkil topgan bosqichga o'tish bilan davom etadi.

SHuning uchun jamiyatning turli rivojlanish bosqichlarida yashagan insonlarning intellektini bir-biriga solishtirib bo'lmaydi.

«Sun'iy intellekt» tushunchasiga turlicha ma'no kiritish mumkin. Turli mantiq va xisoblash masalalarini echuvchi kompyuterdagi intellektni e'tirof etishdan tortib, to insonlar yoki ularning kupchilik kismi orqali echiladigan masalalar majmuasini echadigan intellektual tizimlarga olib boradigan tushunchagacha kiritish mumkin.

«SI» tushunchasi boshidan va shu kunga kadar olimlarning bu. tushunchaga bo'lgan munosabati va ularning «sun'iy» so'ziga nisbatan kelishmovchiligi tufayli karshiliklarga uchramokda. Masalan, USSR FA Kibernetika institutining sobik direktori, marxum akademik V. M. Glushkov «sun'iy idrok» so'zini kushtirnoksiz ishlatgan. Rossiya FA «SI» masalalari buyicha ilmiy YIRILISH raisi akademik G.

S. Pospelov fikricha, «SI» hakida xech qanday so'z bo'lishi mumkin emas, ya'ni hozir xam, yaqin kelajakda xam «uylaydigan mashina» bo'lmaydi. «SI» tushunchasini uzgartirish kech bo'ldi, - deb yozadi u. Bu narsa injener, matematik, kompyuter buyicha mutaxassislar, psixolog, faylasuflarni birlashtiruvchi juda katta axamiyatga ega bo'lgan ilmiy yo'nalish ekanligiga xech kimda shubxa yuk. U odamlarning maqsadi — kompyuterlarning maxsus programmali va apparatli vositalarini yaratish. Kompyuterning qobiliyati ijodiy natijalarni berib turishdan iborat.»

«SI» tushunchasini aniq ta'riflash shuni takozo qiladiki, bu ilmiy yo'nalish oyokka turish va rivojlanish bosqichidadir. Bugungi kunga kelib, shu narsa ma'lum bo'ldiki, «SI» terminiga tabiatdagi jarayon va xodisa-larni o'rganish (tadqiqot qilish) da insondagi ayrim intellektual qobiliyatlarni texnik jixatdan mujassamlashtirgan umumiy tushuncha deb qaramoq lozim.

Sun'iy intellekt borasida oxirgi 30 yil ichida olib borilayotgan tadqiqotlarni shartli ravishda uch bosqichga bo'lish mumkin. Birinchi bosqichda (50-yillarning oxiri) olimlarning xarakati evristik (mutaxassisning tajribasi asosida) izlash nazariyasini yaratishga va faoliyat yoki intellekt darajasiga tegishli bo'lgan «masala echuvchilar»ni yaratish buyicha muammoni x,al qilishga karatilgan. Tadqiqot uchun instrument (asbob) bo'lib EXM xizmat kilgan, har xil o'yinlar, oddiy musika asarlari, matematik masalalar o'ylab topilgan. SHunga uxshash masalalarni tadqiqot uchun tanlash, muammo muxit (bunday mux,itda masalani echish tarmoqlanadi)ning oddiyligi va aniqligini, etarli darajada oson tanlab olish imkoniyatini va «usulga karab» sun'iy konstruktsiyani tuzishni talab qiladi. Bu yo'nalishda bir qancha yutuklarga erishildi. Xususan shaxmat programmalari x,ozir juda yuqori takomilga etkazildi.

Bu programmalar uchun tanlab olish xarakterli bo'lib, odatda teoremalarni isbotlash jarayoni, uyinning ketishi va xrkazolar juda katta sonli mkoniyatlardan tanlanadi. Har bir masalani echish — maqsadga erishishda istikboli bo'lmagan imkoniyatlarni shartta olib tashlash va istikbollilarini ajratib olish evristik usul (algoritm)larning takomillashganiga boglik. Lekin bunday moxiyat asosida A. Nbyuell va G. Saymon tomonidan yaratilgan «universal masalalar echuvchi»ni yaratishga bo'lgan urinish bexuda ketdi, chunki evristik algoritmlar xar bir masalaning xususiyatiga kuchli darajada bog'liq.

Asosiy kiyinchiliklar masalani echish uchun yaratilgan usullarni sun’iy muxitlarda emas, balki xakkoniy muxitda qo’llashga urinish jarayonida sodir bo’ldi. Bu kiyinchiliklar tashki dunyo to’g’risidagi bilimlarni ifodalash muammolari bilan, bu bilimlarni saqlashni tashkil qilish va ularni etarli darajada izlash, EXM xotirasiga yangi bilimlarni kiritish xamda eskirib kolganlarini olib tashlash, bilimlarning to’laligi va bir-biriga zidligini tekshirish va shunga uxshashlar bilan bog’liq. Ko’rsatilgan muammolar bugungi kunda xam tula echilmagan, lekin hozirgi paytga kelib shu narsa ravshan bo’lib koldiki, muammolarni echish — samarali sun’iy intellekt tizimsini yaratishning kaliti ekan.

Ikkinchi bosqichda asosiy e’tibor (60-yillarning oxiridan to 70 yilgacha) intellektual robotlar (real uch o’lchovli muxitda mustakil xolda xarakat qiladigan va yangi masalalarni echadigan robotlar) ko’rishga qaratildi.

Bu borada «intellektual» funktsiyalarning keraqli doirasi: maqsadga yo’naltirilgan xulk (xolat)ni ta’minlash, tashki muxit to’g’risidagi axborotlarni qabul qilish, xarakatlarni tashkil etish, o’qitish, odam va boshqa robotlar bilan mulokotni uyushtirish tadqiq kilindi va amalga oshirildi. Masalan, robotlarda maqsadga yunaltirilgan xulk (xolat)ni ta’minlash uchun ular atrof-muxit haqida bilimlar majmuasiga ega bo’lishi zarur. Bu bilimlar robotga tashki muxit modeli ko’rinishida kiritib quyilishi lozim. Robotning tashki muxit modeli — bu o’zaro boglangan ma’lumotlar yigindisi bo’lib, bu ma’lumotlar moe sinfdagi masalalarni echish uchun kerak. Robotning bilimlar tizim-siga muxitning «fikrdagi» uzgarishini qayta ishlab chiqarish va shu asosda navbatdagi masalani echishga imkon beruvchi algoritmlar xamda bu rejani baja-rilishini va oldindan rejalashtirilgan xarakatlarning kutilayotgan natijalarini nazorat qiluvchi algoritmlar kiritilishi kerak. Demak, intellektual robotlar bilimlar manbaiga ega bo’lishi shart. Bu bilimlar manbaida bilimlar va maxsus blok («reja tuzuvchi») saqlanadi. «Reja tuzuvchi» blokning zimmasiga robotning xarakati programmasini tuzish yuklangan. Bu xarakat programmasi robot tomonidan qabul qilinadi va robotning sensor (kurish vositasi) tizimsi orqali ko’zatiladi. Robotning ish jarayonida «echuvchi blok» bo’lishi kerak. Bu blok robotning xarakati turrisidagi echimni qabul qiladi. Xar ikkala blok bilimlar manbaida saqlanuv-chi bilimlar asosida ishlaydi.

Bu bosqichda ayrim muammolar aniqlandiki, intellektual robotlar yaratishda ularni xal etish zarur. SHunday muammolarga faoliyat kursatadigan muxit haqidagi bilimlarni tasavvur etish, ko’z bilan kurganlarni uzlashtirish, uzgaruvchan muxitda robotlar xulki (xo-lati)ning murakkab rejalarini tuzish va robotlar bilan tabiiy tilda mulokotda bo’lish kiradi.

Uchinchi bosqichda (70-yillarning oxiridan boshlab) tadqiqotchilarning e’tibori amaliy masalalarni echish uchun muljallangan intellektual tizimlarni yaratish muammolariga karatildi.

Xar qanday intellektual tizim, uning qaerda qo’llanishiga borlik bo’lmagan xolda, odam-mashina tizimidir. Mashina sifatida EXM ishlatiladi. Tizimning vazifasi — oxirgi foydalanuvchiga u yoki bu masalani echishda uning kasbi faoliyati doirasida malakali mutaxassis (ekspert) larning yillar davomida orttirgan bilimlaridan foydalanish uchun imkoniyat yaratishdan iborat. Buning uchun EXM

tarkibiga bilimlar manbai va intellektual interfeys kirishi kerak. Bilimlar manbaida xarakterli bo'lgan masalalarni echish usullari haqidagi axborotlar saqlanadi. Intellektual interfeys masalani echish jarayonida oxirgi foydalanuvchi va tizim o'rtasidagi o'zaro munosabatni (xarakatni, ishlashni) ta'minlaydigan sunggi foydalanuvchining xamma vositalarini uz ichiga oladi.

Intellektual interfeysda «echuvchi» va mulokot tizimsini kursatish mumkin. «Echuvchi» bilimlar manbaidan keladigan ma'lumotlar asosida foydalanuvchi uchun keraqli programmalarni avtomatik tarzda birlash-tiradi. Mulokot tizimsi — bu bilimlar manbaida foydalanuvchi tilidan bilimlarni tasavvur qilish tiliga utkazishni xamda teskari jarayonni amalga oshi-radigan translyator («tarjimon»)lar majmuasidir.

Sun'iy intellektli tizimlarga: axborot-qidiruv tizimlari (savol-javob tizimlari), xisob-mantiq tizimlari va ekspert tizimlari kiradi. Intellektual axborot-qidiruv tizimlari EXM bilan mulokot jarayonida foydalanuvchilarning tabiiy tilga yaqin bo'lgan kasb tillarida sunggi foydalanuvchilar (programma tuzmaydiganlar) bilan ma'lumotlar, bilimlar manbalari o'rtasida o'zaro mulokotni ta'minlaydi. Bu tizimlar sun'iy intellekt tizimlarining dastlabkilaridan bo'lib, ular ustida olib borilgan tadqiqotlar xisoblash texnikasi rivoj-lanishi bilan uzviy boglik bo'lgan.

Xisob-mantiq tizimlari, amaliy matematika va programmalashtirish sox,asida mutaxassis bo'lmagan sunggi foydalanuvchilarni, murakkab matematik usullar va shunga mos amaliy programmalardan foydalanib, o'zaro mulokot shaqlida uzlarining masalalarini EXMda echishni ta'minlaydi.

Sun'iy intellekt rivojlanishining yo'nalishlari

Hozirgi vaqtda sanoat sohalari gurkirab rivojlangan mamlakatlarda (bu mamlakatlar uchun «ilm-xajmiy maxsulotlar» katta solishtirma ogirlikka egaligi bilan xarakterlanadi) kompyuterlarini intellektuallashtirish buyicha

yaratishlarning yuqori darajada ekanligi ko'zatilmokda. 80-yillarning boshigacha EXMlarni intellektuallashtirish, asosan tadqiqot, tajriba xarakteriga ega edi. Dunyoda bu tadqiqrtlarni olib borish uchun EXMlarning intellektual imkoniyatlarini kengaytirish buyicha muammo-larni echish yuli belgilandi, bu yuldagi kiyinchiliklar aniqlandi va ularni engib o'tish usullari kursatildi. 1985 yilda jaxon bozorida (Rossiyadan tashqari) intellektual tizimlar 350 million dollarni (ularni yaratish narxini xam kushib xisoblaganda) tashkil etdi. 1990 yilda esa bu xisob 19 milliard dollarga chikishi ko'zatildi, ya'ni misli kurilmagan usishga erishildi. Bunday katta mablag'ni faqat iqtisodning turli sohalari (xujalik ishlab chiqarish, xarbiy)ga intellektual tizimlarni keng qo'llash orqaligina sarflash mumkin.

Intellektual tizimlar (aniqrogi, amaliy sun'iy intellekt tizimlar) ichida ekspert tizimlar muam-mosi ETlarni yaratish texnologiyasini va bilimlar injeneriyasini uzida mujassamlashtirgan aloxida yo'nalish bo'lib tashkil topdi. Gartner Group Inc (AKJ1I) firmasining ma'lumotlariga kura tayyor ETlarning bozor xajmi 1986 yilda 12 million dollarni, ETni yaratishning instrumental vositalariniki 15 million dollarni tashkil etgan, 1990 yilda esa bu ko'rsatkichlar 350—275 million dollarga etdi.

IBM (AKSH) firmasi 1986 yilda xar xil bosqichda yaratilayotgan 70 ta ETga ega edi. Yirik amerika firmalari uzlarining korxonalari (Apolo Computer, Data General Sperry, DEC) da mexnat unumdorligini kutarish uchun ETlarni keng mikyosda yaratib qo'llay boshladilar. DEC firmasi mutaxassislarining ma'lumotiga kura, yaqin orada bu firmada yaratiladigan tizimlarning 30%iga yaqinini sun'iy intellekt tizimlari tashkil qiladi. YApon mutaxassislari taqlif kilgan, 5-avlod EXMlari loyixasiga kura ETlar bu yangi xisoblash texnikasining asosiy qo'llanish sohasiga aylanadi. 1984 yilda Buyuk Britaniyada sun'iy intellekt muammosini xal qilishga yunaltirilgan Tbyuring instituta ishga tushdi. Evropa o'zaro yordam komissiyasi bu muammoni xal qilish «Esprit» loyixasini ishlab chiqayap-ti. Bu loiyxa doirasida uchta yirik kombpyuter firmalari bo'lgan Compagnie Machines Bull (Frantsiya), ICh (Buyuk Britaniya) va Siemens AG (GFR)lar bilimlar bazasiga asoslangan tizimlarni yaratishga yunaltirilgan birlashgan tadqiqot institutini tuzdilar.

Muammoni xal qilishga karatilgan, oxirgi yillarda yaratilgan ETlarning taxlili shuni kursatadiki, yaratuvchilarning asosiy kuch-gayrati, sanoat va konstruktor- texnologik korxonalarda samarali qo'llanuvchi siste-malar yaratishga karatilgan. Bunday qo'llanuvchi ET lar nafaqat an'anaviy tizimlar (I avlod ETlari) masalalarini, balki boshqaruv masalalarini, berilgan axborotni, apparat va maxsulot parametrlarining xisobini echadi.

Shuning uchun ishlab chiqarish-texnologik qo'llanishga " muljallangan ET (II avlod ET)larni loyixalovchilar-ning e'tibori katta bilimlar bazasini, xususan metabi- limlar va ularni qo'llovchi vositalarni, fikr (muloxaza) ning induktiv va xakikatga uxshash sxemalarini amalga oshirish yordamida ekspertdan bilimlarni ajratib olish jarayonini avtomatlashtiruvchi; echiladigan masalaga boglik ravishda strategiyani tanlash jarayonini avtomatlashtiruvchi; an'anaviy ETlar imkoniyatlarini birlashtiruvchi integrallangan ETlarni, ma'lumotlar va bilimlar bazalarini boshqaruvchi tizimlar xamda intellektual amaliy programmalar paketlarini yaratish uchun samarali vositalarni yaratishga karatilgan.

II avlod ETlarida yuqorida sanab utilgan vazifalarni amalga oshirish sanoat ET lari yaratishga omil bo'ladi va ularning qo'llanish sohalarini kengaytiradi.

ET larning keng ommalashuviga sabab, ularning formallashmagan, an'anaviy programmalash uchun kiyin yoki bajarib bo'lmaydigan masalalarni echishda qo'lla-nishidir. Bundan tashqari u (ET) quyidagi xarakterli xususiyatlarga — bilimlarni tuplash, qayta ishlash, umumlashtirish xamda takliflarni kiritish va bu takliflarni tushuntirib berish qobiliyatiga ega.

ETlarning amalda keng qo'llanishiga erishilgan (AKSH, YAponiya va Evropada) bo'lishiga karamay, ularni ommaviy ishlab chiqarish va yoyishga tuskinlik qiluvchi bir kator xal bo'lmagan quyidagi muammolar xam bor: - ET larni yaratish shu paytgacha uzok va kiyin jarayon bo'lib kolayotganligi;

- bilimlar qabul qilish (olish): saralash, strukturalash, tasvirlash, sozlash va bilimlarni ko'zatib borish;
- xayotda kupincha echiladigan masalalar vaqt o'tishi bilan turlicha echilishi takozo etiladi, ko'pgina ETlar asosan uzgarmas masalalar echishga mo'ljallanganligi uchun ularni yuqoridagi kabi masalalarga qo'llab bo'lmaydi;

Demak, ET larni yaratish va ulardan natijalar olish uchun xali kup ishlar kilinishi kerak.

Ma’lumotlar va bilimlar

Har qanday sun’iy intellekt tizimining asosi bilimlar modeli va uning asosida yaratilgan bilimlar bazasidan iborat bo’lib, u xam ma’lumotlar, xam bilimlar bilan ishlashga yunal-tirilgan. SHuning uchun bilimlar nimasi bilan ma’lumotlardan farq qilishini tushunib olishimiz kerak.

Ma’lumotlar — bu xabarlar bo’lib, ular aniq masalani echayotganda xulosa chiqarish va shu masalani echish usulini aniqlash uchun kerak. Ma’lumotlar bilan bilimlar orasida aniq bir chegara bor deb bo’lmaydi, chunki ma’lumotlarda xam ma’lum bir bilimlar bo’lishi mumkin va aksincha.

Ma’lumotlar maxsus dasturlar yordamida ishlanuvchi matematik modellarning rakamli parametrlarini aks ettirishi yoki biron bir sanoat tarmori sohasidagi korxonalar rejalari bajarilishining hozirgi xolatini aks ettirishi mumkin. Bu ma’lumotlar qayta ishlangandan sunggina kurilayotgan tarmoq buyicha reja bajarilishining umumlashgan sonli xarakteristikasini berish, muxim joylarini aniqlash, kurilayotgan tarmoq kelajagini oldindan aytish mumkin. Bir so’z bilan aytganda, yangi bilimga ega bulinadi. Ta’kidlash kerakki, ma’lumotlar ishlab chiqarish jarayonlariga bevosita ta’sir kursatmaganligi uchun ularni «sust», shu ma’lumotlardan foydalanuvchi dasturlarni esa «faol» (aktiv) deyish mumkin.

Bilim — xayotda sinalgan xakikatni bilish maxsuli, uning inson ongida to’g’ri aks ettirilishi. Ilmiy bilimlar moxiyati uning utmishdagi, hozirgi va kelajakdagi xakikatni tushunishidadir, dalillarni to’g’ri asoslay bilib, umumlashtirishidadir. Odamning fikrlashi xar doim bilmaslikdan bilishga, yuzakilik-dan borgan sari chukurrok va xar tomonlama bilishga tomon xarakat qiladi.

Sun’iy intellektli tizimlarda kurilayotgan soha to’g’risidagi bilimlar bilimlar manbaida tuziladi. Bu manba ma’lumotlari bilimlarni va kurilayotgan sohani uzida aks ettiradi. SHuning uchun xam ma’lumotlar bilan bilimlar o’rtasida kat’iy tafovut yuk. SHunga karamay bilimlarni ma’lumotlardan farklaydigan maxsus alomatlar bor. Quyida biz shu alomatlar-ning ayrimlarini ko’rib chiqamiz.

1. Interpretatsiya. Bu so’z lotincha «interpretatio» so’zidan kelib chikkan bo’lib, sharxlash, tushuntirish, oydinlashtirish singari ma’nolarni anglatadi. kompyuterda joylashtirilgan ma’lumotlar faqat moe dastur orqali mazmunli talkin kilinishi mumkin. Programmalarsiz ma’lumotlar xech qanday mazmunga ega emas. Bilimlar shu bilan farkdanadiki, bunda mazmunli izoxlash imkoniyati xar doim bo’ladi.

2. Strukturalanganlik yoki munosabatlar sinflarining mav-judligi. Ma’lumotlarni saqlash usullarining xar xilligiga karamasdan, ularning bittasi xam ma’lumotlar orasidagi aloqalarni ixcham yozish imkoniyatini ta’minlamaydi. Masalan, ma’lumotlar bilan ishlayotganda umuman elementlar va tuplamlar uchun umumiy bir xil xabarlarni kup marta ifodalashga (yozishga) to’g’ri keladi. Bilimlarga utilganda, bilimlarning ayrim birliklari o’rtasida shunday munosabat urnatish mumkin: «element-tuplam», «tip-tip bulagi», «kism- butun», «sinf-sinf bulagi». Bu tuplamning barcha elementlari uchun bir xil bo’lgan ma’lumotni

aloxida yozib va saqlab quyishga imkon yaratadi. Bu ma’lumotni, agar kerak bulsa, tuplamning xoxlagan elementini ifodalash uchun keraqli joyga avtomatik ravishda berish mumkin. Bunday uzatish jarayonini ma’lumotlarning «vorislik qilish» jarayoni deyiladi.

3. Xolat aloqalarining mavjudligi. Bu aloqalar xotirada saqlanadigan yoki kiritiladigan ayrim xodisa yoki dalillarning bir-biriga (xolat) mosligini xamda o’zaro munosabatini aniqlaydi.

4. Aktivlik (faollik). Bilish aktivligi inson uchun xosdir, ya’ni insonning bilimlari faoldir. Bu esa bilimni ma’lumotlardan umuman farklaydi. Masalan, bilimlarda karama-karshilikni paykash - ularni engib o’tishga va yangi bilimlarni paydo bo’lishiga sabab bo’ladi. Aktivlikning rag’batlantiruvchi omillaridan biri bilimlarning tulik bo’lmasligidir. Bu rag’batlantiruvchi omil bilimlarni tuldirish zarurligi bilan ifodalanadi. Kompyuterdan foydalanilganda dastlabki yangi bilimlar bo’lib dasturlar xisoblanadi, ma’lumotlar esa kompyuter xotirasida sust ravishda (xarakatsiz) saqlanadi.

Ma’lumotlar va ma’lumotlar tuzilishi predmet sohalarining xususiyatlarini to’la o’lchamda ifodalamaydi. YUqorida biz xar doim ma’lumotlar bilan bilimlar o’rtasida aniq chegara quyish mumkin emasligini ta’kidlab o’tgan bo’lsak xam, lekin bular o’rtasida farklar bor. Bu farklar bilimlarni xarakterlaydigan xamma to’rt belgini biror darajada ifodalovchi, kompyuterdagi bilimlarni modellar ko’rinishida tasvirlovchi rasmiyatchilikning paydo bo’lishiga olib keldi.

Bilimlarni taqdim etishning modellari

Bizni o’rab turgan olam to’g’risidagi bilimlar dekla-rativ va protsedurali bilimlarga bulinadi. Deklarativ bilimlar bu biror bir tizimda o’zaro borlangan dalillardir. Xakikatan xam ruy bergan biror bir xodisa, vokea dalilga misol bo’la oladi [24,34].

Protsedurali bilimlar — dalillar ustida bajarilgan amallarni (algoritmlar, dasturlar, analitik uzgartirishlar, empirik qoidalar va shu kabilarni) amalga oshirish natijasida hosil bo’ladigan bilimlardir. Bilimlarning bunday bulinishi shartli xarakterga ega, chunki bilimlarni ifodalash (tasvirlash) ning aniq modellari xar xil maqsadda tasvirlashning deklarativ va protsedurali shaqllarini ishlatadi.

Kompyuterning boshlanrich uchta avlodida protsedurali tasvirlash yagona, u xam masalalarni echishda qo’llaniladi. Kompyuterlar uchun dasturlar bu bilimlarning saqlovchilari bo’ladi, deklarativ bilimlar xar doim tobe bilimlardir. Intellektual tizimlar buyicha muta-xassislarni xar ikki bilim turi bir xilda kiziktiradi.

Ekspert tizimlar sohasidagi tadqiqotlar shuni kursatadiki, bilimlarni tasvirlash uchun kupincha semantiq tarmoqlar, freymlar va maxsulot qoidalarining modellari ishlatiladi. SHuning uchun bu modellarni tularok ko’rib chiqamiz.

1. Semantiq tarmoqlar. Semantiq tarmoqlar apparati yordamida bilimlarni tasvirlash biror bir muxitni tashkil etuvchi ob’ektlar va ular orasidagi aloqalar majmuasidir.

Xar xil avtorlar semantiq tarmoqlarning turli xil turlari tuzilishini taqlif kilmokdalar. Bu turlarning umumiy, asosiy funktsional elementi bo’lib, ikki kiem («tugunlar» va «yoylar»)dan iborat bo’lgan struktura xizmat qiladi. Xar bir tugun

biror bir tushunchani, ey esa ixtiyoriy ikkita tushuncha orasidagi munosabatni bildiradi. Munosabatlarning xar bir jufti oddiy dalilni bildiradi. Tugunlar moe munosabatning nomi bilan belgilanadi, yoy yo'nalishiga ega bo'ladi. Bunga kura aniq dalil tushunchalari orasidagi «sub'ekt yoki ob'ekt» munosabatini tasvirlaydi. Masalan «Rustamov institutda ishlaydi». Bu erda «Rustamov» sub'ekt, «institut» esa ob'ekt sifatida tasvirlanadi, ular («ob'ekt» va «sub'ekt»lar) «ishlaydi» munosabati bilan borlangan. U xolda «Rustamov institutda ishlaydi» dalilini aks ettiradigan semantiq tarmoqning funktsional elementi quyidagi ko'rinishga ega bo'ladi:

Rustamov institutda->-ishlaydi. Bu tarmoqda sub'ekt va ob'ektni boglovchi faqat binar aloqa (munosabat) ishlatilgan. Semantiq tarmoqlarni tuzishda tugunlar orasidagi munosabatlar sonini cheklab bo'lmaydi, ya'ni biror bir tugun boshqa ixtiyoriy tugunlar bilan munosabatda bo'lishi mumkin. Bu ixtiyoriylik natijasida dalillar tarmogini tuzish ta'minlanadi. Masalan, tarmoq, quyidagi tekstni tasvirlaydi:

«Rustamov institutda ishlaydi. U institut direktori. Rustamov texnika fanlari doktori ilmiy darajaga ega, ilmiy unvoni — akademik. U institut ilmiy kengashining raisi. Bugun soat 9da Rustamov institut metodik kengashida, soat 16 da esa institut ilmiy kengashida ma'ruza qiladi». Bu tarmoqda vaqtli boglanishlar yoylar, fe'llarga mos boglanishlar esa tugunlar yordamida tasvirlangan.

Semantiq tarmoqlar ko'rinishidagi bilimlar tasvirlanishining yaxshi tomoni shu bilan xarakterlanadiki, bunday tarmoqlar bilan kompyuterda ishlash oson kechadi. CHunki bunday tarmoqlarda ob'ektlar orasidagi aloqalar aniq kursatiladi, dasturlar tuzish engillashadi.

Masalan, tarmoq buyicha Rustamov qaerda, kim bo'lib ishlashini va aniq vaqtlarda qaerda bo'lishi va nima qilishini bilish mumkin. SHuningdek, boshqa murakkabrok savollarga xam javob topish mumkin. Masalan, «Bugun institut ilmiy kengashi bo'ladimi va soat nechada?»

Semantiq tarmoqlar va ularning modullari bilimlar buyicha muxandis tomonidan yaratiladi, boshqacha so'z bilan aytganda xisob-mantiq tizimlarning yaratuvchilari tomonidan tuziladi. SHundan sung tizim sunggi foydalanuvchilarga xavola etiladi. Semantiq tarmoqlar kurilishiga bunday yondoshish foydalanuvchilarni, masa-lan, texnologik jarayonlarni loyixalash va boshqarish sohasida ishlovchilarni kanoatlantirmaydi. Amaliy dastur tuzuvchi o'zaro munosabat (aloqa) boski-chida texnologik jarayonning xar bir ko'rinishi uchun aloxida bu tizimning semantiq tarmorini tuzadi. Sunggi foydalanuvchi tomonidan texnologik jarayon uzgartirilsa, bilimlar muxandisiga semantiq tarmoqni uzgartirishga to'g'ri keladi.

Takrorlash uchun savollar:

1. Ma'lumotlar va bilimlar haqida nima bilasiz?
2. Bilimlarni taqdim etishning modellarini tushintiring?
3. Protsedurali bilimlarga ta'rif bering?

3- Ma'ruza

Ifodalar mantiqlari

Reja:

1. Mantiq tarifi
2. Tafakkur xususiyatlari
3. Mantiq termini
4. Mantiq qonunlari

Tayanch iboralar: *mantiq, tafakkur, logika, ob'ekt*

Mantiq tarifi

Mantiq arabcha so'zdan olingan bo'lib so'z, fikr, aql ma'nolarini bildiradi (yoki Logika,yunoncha – ***logike,*** *deb ham ataladi*). U ikki hil ma'noda qo'llaniladi: agar so'z narsalarning tarkibi, bog'lanishi ustida borsa - ob'ektiv fikirlar bog'lanishi haqida boradigan bo'lsa – su'bektiv maniq bo'ladi.

Mantiq tafakkur qonunlari, narsalar mantig'i yoki obektiv mantiq inkorisidir.

Taffakur xususiyatlari

Qator predmetlar hodisalarga xos bo'lgan umumiy xususiyatlarni,ular o'rtasidagi aloqalarni aks ettirar ekan, tafakur bir xodisa xaqidagi bilimlardan kelib chiqiarish imkoniatiga egadir. Masalan ishlab chiqarishga texnika yutuqlarining qanday qo'llanilishiga qarab mexnat unumdorligining darajasi xaqida fikir yuritish mumkin. Ttafakkurning yana bir xusisiyatlaridan biri uning til orqali ifoda qilinishidir. Tafakkur o'z mazmuni va formasiga (shakliga) ega. Tafakkurformasi fikir mazmuninitashkil qiluvchi elementlarning bog'lanish usilidan iborat. Tafakkur uch xil shakilda: tushuncha, hukum, xulosa chiqarish formalarida mavjud.

<<Mantiq>> termini

Mantiq termeni bir biridam mustaqil uchta fanni
formal mantiq,
dialektik mantiq,
matematik mantiqni ifoda qiladi.

Dialektik mantiq tafakkurning vujudga kelishi va taraqiy etishining eng ummumiy qonunlari va ular asosida yaratiladigan metadologik prinsiplarni o'rganadi.

Formal mantiq tafakkurning aniq mazmunidan chetlanib etiborini uning tuzilishini tekshirishga qaratadi.Shu jumladan tafakkur formalari o'rtasidagi bog'lanishlarni ham ularning tuzilishi nuqtai nazaridan o'rganadi.

Matematik ***(simvolik) mantiq*** XIX asirning ikkinchi yarimida vujudga kelgan bo'lib matematika fanining muxim yo'nalishlaridan birini tashkil qladi. Matematik mantiqga matematik metodlardan foydalanuvchi mantiq deb tarif berish mumkin.

Mantiq qonunlari

Qonun predmet va hodisalare o'rtasidagi obektiv, muxim, zaruriy, takrorlanuvchi, nisbatan o'zgarmas bog'lanishlardir. Hamma tabiy va ihjtimoiy

hodisalar va boshqa hodisalar malum qonunlar asosida mavjud bo'ladi va o'zgaradi.

Mantiqiy tafakur qonunlari, tafakur formalari kabi umuminsoniydir. Tafakkur qonunlari ob'ektiv voqealarni inson miyasida uzoq vaqt davomida aks etirishi natijasida vujudga kelgan xanda shakillanga. Tafakkur qonunlari amal qilish to'g'iri, tushunarli, aniq, izchil, ziddiatsiz asoslangan fikir yuritish demakdir.

Mantiq qonunlari quyidagi turlarga bo'lnadi

- ***Ayniyat qonuni.***
- ***Ziddiyat qonuni.***
- ***Mustasno qonuni.***
- ***Ziddiyat qonuni.***

Ayniyat qonuni. Ob'ekt voqeadagi pretmed va hodisaning doimo o'zgarib turishiga qaramay, ularda nisbiy barqarorlik mavjud, u o'z ifodasini ayniyat qonunida topadi. Masalan bizga tanish insonni biroz vaqt o'tgandan so'ng ko'rsag ham uni tanib olamiz. Ayniyat qonuni fikirlash jarayonida fikirning aniqligi, muayyan ekanligini ifodalaydi.

Ziddiyat qonuni. Kishilar o'z faoliyatida predmet va hodisalar bir vaqitda, bir sharoitda biror hususiyatga ega bo'lishi, ham ega bo'lmasligi mumkin emasligini bilganlar.Bu hodisa bilimlarimizda ziddiyat qonuni sifatida shakillanib qolgan. Ziddiyat qonunini quyidagicha ifodalash mumkun: ayni bir predmet yoki hodisa haqida aytaylik ikki zid fikir bir vaqitda bir nisbatda chin bo'lishi mumkin emas. Bu qonun <<***A*** ham ***V,*** ham ***V*** esa bo'la olmaydi >> formulasi orqali beriladi.

Mustasno qonuni. Ziddiyat qonuni bilan uzviy bog'liq bo'lib u ikki o'zaro zid fikirning munosabatini bildiradi. Bilish jarayonida biz, fikrimizda obektiv olamdagi predmet va xodisaning ayni bir vaqtda mavjud emasligini, ularga biror xususiyatga xos emasligini aks ettiramiz.

Bu qonun quyidagicha ifodalanadi- ayni bir predmet yoki xodisa haqida bir – birini inkor etuvchi ikki zid fikr ayni bir muxokama doirasida ayni bir vaqtda, ayni bir nisbatda xato bolishi mumkin emas, ularning biri albatta xato boladi, uchinchi xolatning bolishi mumkin emas. Uchinchisi mustasino qonuni << ***AV*** yoki ***V*** emasdr >> formulasi orqali beriladi.

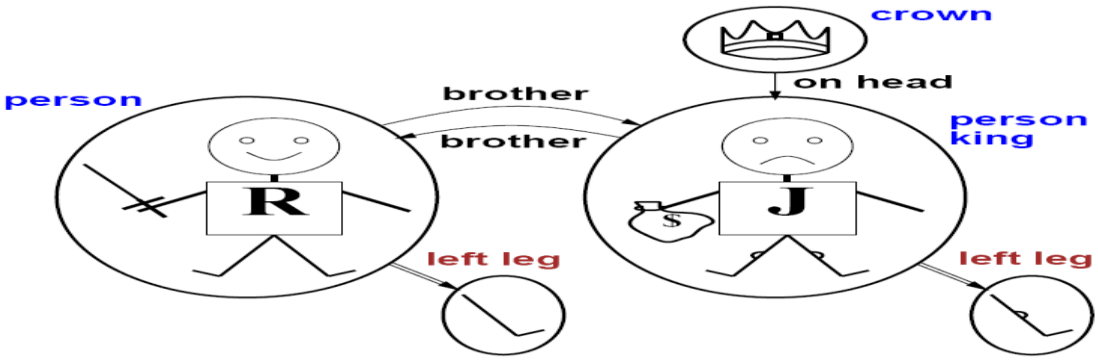
Etarli asos qonuni. Tabiat va jamiyatdagi predmet va xodisalar bir- biri bilan bog'liq xolda rivojlanadi. Ular o'rtasidagi salbir bog'lanishlar eng muxim bog'lanishlardir. Predmet va xodisalardan birining mavjudligi taqazo etadi

Xar bir predmet va xodisaning real asosi bolgani kabi, ularni inkrosi bolgan fikrimiz xam asoslanga bolishi kerak. Bu o'z navbatida etarli asos qonunining talabiga binoan har qanday predmet va hodisa haqida aytilgan fikr asoslangan bolishi kerak. Etarli asos qonuni << Agar ***V*** mavjud bo'lsa uning asosi sifatida ***A*** ham mavjud >> formulasi orqali beriladi..

Hukm model va talqin nisbatan to'g'ridirModel> = 1 ob'ektini (domen elementlar) va ular orasida munosabatlarni o'z ichiga oladi. Og'zaki bo'lgani uchun doimiy belgilar -> ob'ektlar

predicate ramzlar -> munosabatlari funktsiya ramzlar -> funktsional munosabatlarni bildiradi.

Atom hukm predicate (term1, ..., termn) ob'ektlar IFF haqdir term1 bilan ataladi, ..., termn nisbatan ataladi



Bosh yilda Logics

til	**ontologik Majburiyat**	**epistemologik Majburiyat**
taklif etish Mantiq	faktlar	noma'lum / to'g'ri / noto'g'ri
Birinchi tartibi Mantiq	Fact, ob'ektlar, munosabatlar	noma'lum / to'g'ri / noto'g'ri
vaqtinchalik Mantiq	Faktlar, ob'ektlar, munosabatlar, marta	noma'lum / to'g'ri / noto'g'ri
Ehtimollar nazariyasi	faktlar	e'tiqod darajasi Î [0,1]
Fuzzy mantiq	Haqiqat darajasi Î [0,1]	Ma'lum interval qiymati

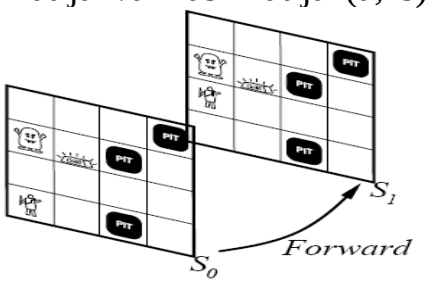
Taklif etish Mantiqi

Taklif etish mantiqi deklarativ bo'ladi: sintaksisga mos taklif etish mantiqiy (qisman / bo'ladigan / rad ma'lumot olish uchun imkon beradi ma'lumotlar tuzilmalari JB dan farqli o'laroq taklif etish mantiq tuzilish bo'ladi: B11 ^ P12 ma'nosi B11 va P12 ma'nosi taklif etish mantiq ma'nosi kontekst mustaqil: (farqli o'laroq tabiy til ma'nosi doirasida)

- Taklif etish mantiq juda cheklangan ifodali qodirdir: (tabiiy tildan farqli o'laroq)

O'zgarish Track tutilishi

Vaziyatlar Natija vazifasi natija (a, s) tomonidan ulangan s bir qilishdan natijalari vaziyat



Takrorlash uchun savollar:

1. Kanday ifodalar mantigini bilasiz?
2. Birinchi tartibi Mantiq nima?
3. Mantiq qonunlari qandday turlarga bo'lnadi?

4. Ma'ruza
Birinchi tartibli predikatlar mantig'i

Reja:

1. Ish muhitlarining xususiyatlari
2. Agent strukturasi
3. Agent dasturlar

Tayanch iboralar: *determestik, stochastic, epizodik, izchillik*

Ish muhitlarining xususiyatlari

Deterministik vs. stochastic. Muhitning keyingi holati to'liqligicha joriy holat bilan aniqlanadi va harakatlar agent tomonidan amalga oshiriladi va bunday holatda muhit aniqlovchidir (deterministic) aksincha holatda bu tasodifiy (stochastic).

Tozalovchi va taksi haydovchilar:

- Bir qancha ko'zga tashlanmaydigan aspektlari tufayli tasodifiy (stochastic)\→ shovqin yoki nomalum.

Epizodik va izchillik

- Epizodik = agentning tushuncha + harakatining yolg'iz juftligi
  - Agent harakatining sifati uning boshqa epizodlariga bog'liq emas
    - Har bir epizod bir biridan mustaqil
  - Epizod muhiti soddaroq
    - Agent oldindan o'ylashga muhtoj emas
- Izchillik
  - Hozirdagi harakat kelajakdagi qarorlarga ta'sir qilishi mumkin

Masalan, Taksi haydovchisi va shahmat.

Turg'un va faol

Faol muhit vaqt o'tgan sari doim o'zgarib turadi

- Masalan., ko'chadagi odamlar soni

Turg'un muhit esa

- masalan., Manzil

Yarimfaol

– Muhit vaqt o'tishi bilan o'zgarmaydi lekin agentning ish bajarishi qayd etiladi

Alohida va Davomiy

– Agar aniq holatlar soni cheklangan bo'lsa persepsiya va harakatlar tushunarli izohlangan bo'lgan holatda muhit alohida

- Misol., Shaxmat o'yini
- Davomiy: Taksi haydash

Yolg'iz agent va ko'p sonli agent

- Krossvord jumbog'ini yechish– yolg'iz agent
- Shaxmat o'ynash– ikkita agentlar
- Raqobatdosh ko'p sonli agent muhiti
  - Sahaxmat o'ynash
- Hamkorlikdagi ko'p sonli agent muhiti
  - Avtomatlashtirilgan taksi haydovchisi
  - To'qnashuvdan saqlanish

Ma'lum va noma'lumlar

Bu aniqlik muhitning o'zini ifodalamaydi lekin agentning muhit haqidagi holati va bilimini ifodalaydi. Ma'lum muhitda hamma harakatning natijalari berilgan (masalan soliter karta o'yini). Agarda muhit noma'lum bo'lsa agent yahshi qaror chiqarish uchun uni qanday ishlashini o'rganishi kerak (masalan yangi video o'yin).

Agent strukturasi

Agent = arxitektura + dastur

- Arxitektura = ma'lum bir hisoblash qurilmalari (sensorlar datchiklar + ijrochi elementlar)
- (Agent) Dasturlar = bir nechta funktsiyalar yaniki agent haritalashni amalga oshiradi = "?"
- Agent Dastur= SI ning vazifasi

Agent dasturlar uchun ma'lumot kiritish

- Faqatgina joriy persepsiya
- Agent funktsiyasi uchun ma'lumot kiritish
- Butun persepsiya ketma -ketligi
- Agent ularning barini eslab qolishi zarur

Agent dasturini quydagicha amalga oshirish. So'rovnoma jadvali (agent funktsiyasi).

Agent dasturning skelet dizayni

```
function TABLE-DRIVEN-AGENT(percept) returns action
  static: percepts, a sequence, initially empty
          table, a table, indexed by percept sequences, initially fully specified

  append percept to the end of percepts
  action ← LOOKUP(percepts, table)
  return action
```

Agent Dasturlar

P = jamlanmasi muvofiq persepsiyalar

T= agentning amal qilish muddati

- Qabul qiladigan persepsiyalarining umumiy soni

Sorovnoma tablitsaning hajmi

Shaxmat o’yini misolida ko’rsak

- $\sum_{t=1}^{T} |P|^t$
- P =10, T=150
- Kamida 10^{150} yozuvli tablitsa so’raladi

Katta hajmga qaramasdan, sorovnoma jadvali istaganimizni amalga oshiradi.

SI ning asosiy topshirig’i

- -ratsional hulq-atvorni ishlab chiqish uchun qanday qilib imkoniyatlari keng bo’lgan dasturlar yozishni aniqlab olish kerak

Jadvaldagi ko’ miqdordagi yozuvlar o’rniga kam miqdordagi kod kiritgan avzal

- Masalan: Nyuton metodining 5 qatorlli dasturi
- V.s. Kvadrat ildizi sinus kosinusli kata tablitsa.

Agent dasturning turlari To’rtta turi mavjud

Oddiy refleks agentlar

Namunaga asoslangan refleks agentlar

Maqsadga asoslangan agentlar

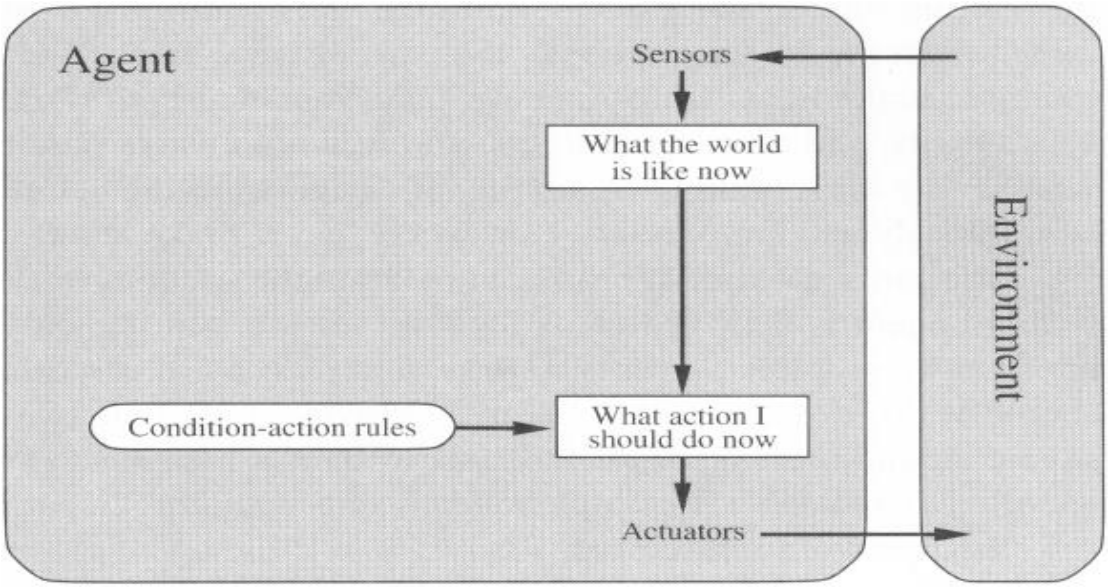
Utilitga asoslangan agentlar

Oddiy reflex agentlar

U faqatgina holat harakat qoidasidan foydalanadi

- qoidalar “agar … keyin …” shakliga o’hshaydi
- Samarali lekin, foydalanish imkoniyati chegaralangan
- Chunki blim ba’zida aniq ifodalanmagan bo’ladi
- Faqat ish
  - Agarda atrof-muhit to’liq ko’zga tashlanadigan bo’lsa

```
function SIMPLE-REFLEX-AGENT(percept) returns action
  static: rules, a set of condition-action rules

  state ← INTERPRET-INPUT(percept)
  rule ← RULE-MATCH(state, rules)
  action ← RULE-ACTION[rule]
  return action
```



Agent programmalashtirilgandan so’ng, u tezda ishlay oladimi?
-yo’q,u haliham o’qitilishi kerak

- SI da,
  - Agent tanlangandan keyin

-misollar jamlanmasi berish orqali uni o’qitish va boshqa misollar jamlanmasi berish orqali tekshirish zarur

- keyin esa bu agent **o’rganuvchi agent** deyilishi mumkin .

Takrorlash uchun savollar:

1. Agentlar tushunchasini tushuntiring?
2. Ish muhiti hususiyatlari qanday.?
3. Oddiy reflex agentlar hususiyati qanday?

5. Ma’ruza
Tanqid cheklovlari

Reja:

1. Qidirish joy muammosi
2. Notugallik va qaror qabul qila olishlik.
3. Qaror qila olishlik va notugallik

Tayanch iboralar: *qidirish, notugallik, qaror qabul qilish, muammolar.*

Qidirish joy muammosi

5.1.rasm.

Avtomatlashtirilganisbotlovchilarbirsekundda 20000 xulosani namoyish eta oladilar. Inson klaresa faqat 1 sekundta 1 xulosaningina namoyish eta oladilar.

- Sabablar:
- Insonlar yuqori darajada ishlaydigan intuitive (sezgir) hisobdan foydalanadilar.

Insonlar g`oya, fikrlar bilan ishlaydilar

Muammolar

Odamlar intuitive ilm haqidagi bilimlarni og`zaki tarzda ifoda eta olmaydilar. Insonlar evristikani tajriba orqali o`rganadilar. Yechim:

- Evristikani mexanizm o`rganish texnikasi, ya`ni Ertel/Shuman/Sutner/Sutner/Ertel kabilar orqali o`rganing.

Misol:qaror

Daliliy yo`nalish moduli turli xil alternativlarni keyingi qadam uchun evriskit tarzda baholaydi.

Alternativlarni eng zo`r, a`lo darajadagi baholash bilan tanlaydi. Mavjud gaplarni baholanishi xatolarraqamiorgali, ijobiyxatolarraqami, vaqt murakkabligi. Ba`zilar mexanizm o`rganish algoritmini muvaffaqiyatli dalillar Ertel/Shuman/Sutner; Sutner/Ertellarnio`rganish uchun qo`llaydi. Muvaffaqiyat luqaror qadamlari ijobiydek saqlanadi. Muvaffaqiyatsiz qaror odimlari salbiydek saqlanadi. Mexanizm o`rganish tizimi gaplarning baholanishi uchun programma yaratadi. Notugallik va qaror qabul qila olishlik. Bu yerda to`gri va to`liq hisob va teorema isbotlovchilari mavjud.

Notugallik va qaror qabul qila olishlik.

Har qanday teorema aniq bir vaqtda isbot qilinishi mumkin. Har qanday teorema aniq bir vaqtda isbot qilinishi mumkin.

- Agar gap yolg`on bo`lsa nima bo`ladi?

Qaror qila olishlik va notugallik

Agar til kuchayib boyib ketsa, til notugallikka yuztutadi. Misol PL1

Exercise ??

Propositional logic is decidable!

Logic of higher (second) level:

"If a predicate p(n) holds for n, then p(n + 1) also holds",

or

$$\forall p\ p(n) \Rightarrow p(n+1)$$

Agar til kuchayib boyib ketsa, til notugallikka yuz tutadi. Misol PL1 Misol: buteoriy apara dosklar (hammaning fikriga zid g`oya)ga yo`l qo`yadi. Misol: buteoriy apara dosklar (hammaning fikriga zid g`oya)ga yo`l qo`yadi. O`zining soqolini olmaydiganlarning soqolini oladigan barbenlar to`plami.

Taajjub: yetarlicha kuchli bo`lgan til qisqartirishlarga olib keladi. Uchadigan pingvin

1. Tweety is a penguin
2. Penguins are birds
3. Birds can y

- Formalized in PL1, the knowledge base WB results:

$$penguin(tweety)$$
$$penguin(x) \Rightarrow bird(x)$$
$$bird(x) \Rightarrow fly(x)$$
$$WB \vdash fly(tweety)$$

5.2.rasm

- New attempt: penguins cannot fly

$$penguin(x)\ ) \Rightarrow \neg fly(x)$$
$$\vdash \neg fly(tweety)$$

But: $\vdash fly(tweety)$

- The knowledge base is inconsistent.
- The logic is monotonic:
  - new knowledge cannot undo old one.

Gavda muammolari Monoton bo`lmagan mantiq: bilimni bilimlar omboridan olib tashlash mumkin. Yanglish mantiq: obyektlar boyligi, boshqa qonunlar amalga kiritilmasa saqlanib qolinadi. Misolda, qushlarning uchaolmaslik qopnuni xato qoidalar bo`lishi mumkin.

- P(qish(X))Y(X))=0:99
- P(YJqush)=0:99 bilan ishlash osonroq
- Noaniq bilimlarining modellashtirilishi

- Noaniq bilimlarning modellashtirilishi uchun turli xil rasmiylikning taqqoslanishidir.

Formalizm	Haqiqiy baho raqamlari	Ifoda etiladigan ehtimolliklar
G’oyaviy mantiq	2	–
Diskret ehtimolli mantiq	∞	–
Davomiy ehtimolli mantiq	n	Ha
	∞	Ha

Takrorlash uchun savollar:

1. Qanday qidiruv algoritmlarini bilasiz?
2. Qidiruvda algoritmlarning ahamiyati nimada?
3. Qaror qabul qilishda qidiruv algoritmlari roli?
4. Notugallik va qaror qabul qilish?

6. Ma’ruza
Qidiruv

Reja:

1. Muammoni hal qilish agenti
2. Tatilni rejalashtirish
3. Vacuum Dunyosi

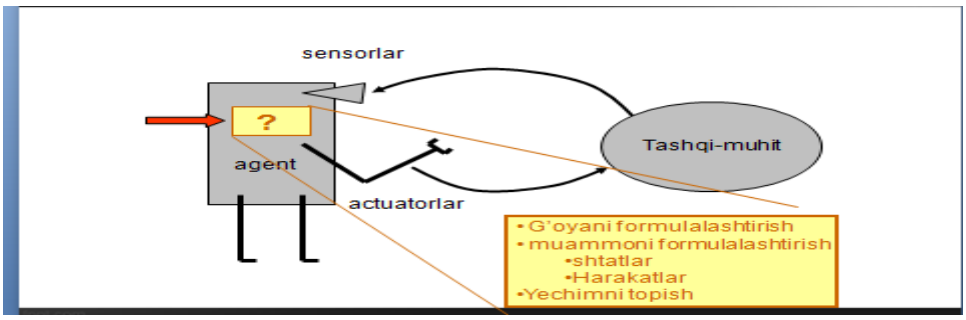
Tayanch iboralar: *maqsad, agent,reja, vacuum*

Tatilni rejalashtirish

Ruminiyadagi tatil; Joylashgan joyi Arad. Buharestdan ertaga parvoz qiladi.Maqsadni formulalashtirish:
Buxarestda bo’lish

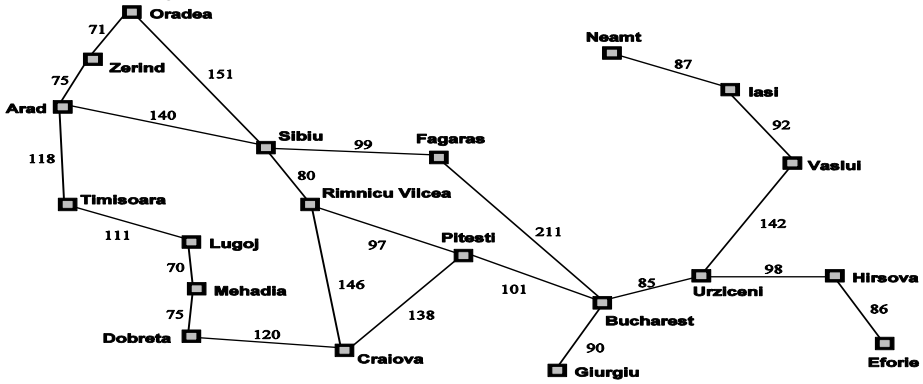
- Muammoni formulalashtirish:
  Shtatlar: Turli shaharlar
  Harakatlar: Shaharlar orasida qatnov
- Yechimni topish:

Shaharlar ketma-ketligi: Arad, Sibiu, Fagaras, Bucharest
Muammoni hal qilish agenti

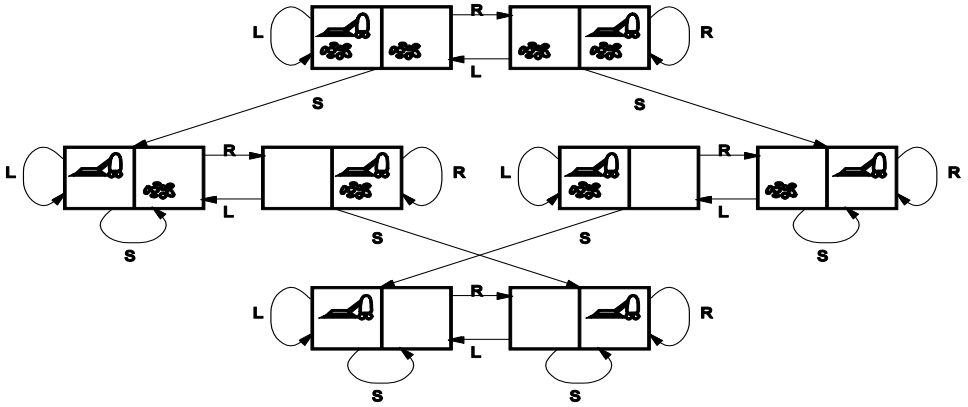


Misol: Marshrut toppish

Muammo yechimi



Vacuum Dunyosi



Muammoni hal qilish agenti.

Muammoni hal qilishning 4 ta asosiy bosqichi bor:

– Maqsadni formulalashtirish
  - Qaysi biri kerakli yo'nalish
– Muammo formulasi
  - Qaysi yo'nalish va shtatlar orqali boriladi

– Qidirish
  - Yo’nalishlarning ehtimoliy ketma-ketliklari ichidan Eng optimal yo’nalishni tanlash.

– Amalga oshirish
  - So’nggi qarorni chiqarish.

function SIMPLE-PROBLEM-SOLVING-AGENT(*percept*) **return** an action

static: *seq*, an action sequence

state, some description of the current world state

goal, a goal

problem, a problem formulation

state ← UPDATE-STATE(*state*, *percept*)

if *seq* is empty **then**

goal ← FORMULATE-GOAL(*state*)

problem ← FORMULATE-PROBLEM(*state*,*goal*)

seq ← SEARCH(*problem*)

action ← FIRST(*seq*)

seq ← REST(*seq*)

return *action*

Qilingan taxminlar

- Muhit static
- Muhit tarifga ega
- Kuzatiladigan muhit
- Harakatlar deterministic

Muammoni formulalashtirish.

- Muammo aniqlashtiriladi:
  – Boshlang’ich nuqta, Arad
  – Vorislik funksiyasi *S(X)*= Shtatlar yo’nalishlari tarmoqlari juftliklari
    - *S(Arad)={<Arad → Zerind, Zerind>,…}*

Boshlanuvchi shtat + vorislik funksiyasi = shtatlar fazosi

– Maqsadni testlash,

– Aniq malumot, *x=‘buharestda bo’lish’*
  - Nazarda tutiladi, *checkmate(x)*

– Yo’l narxi (qo’shimcha)
  - Yo’nalishlarning orasidagi masofa
  - *c(x,a,y)* bosqich narxi, tahminiy >= 0

A solution is a sequence of actions from initial to goal state. Optimal solution has the lowest path cost.

Shtat yo’nalishlarini tanlash

- Real dunyo haddan tashqari murakkab.

Shtatlar fazosidan muammo yechimi uchun abstrakti kerak.

- Shtat = Ko’chmas mulk majmuasi.

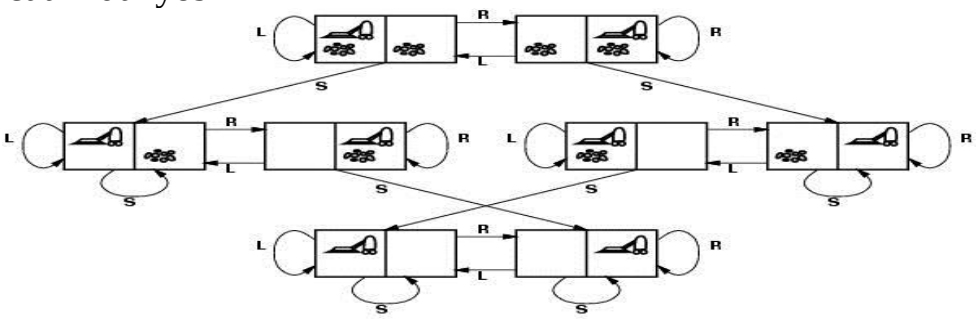
Yo’nalish = real harakatlar kompleks birikmasi.

– e.g. Arad →Zerind mumkin bo'lgan murakkabliklarni ifodalaydi. yo'nalishlar, detours, qolgan to'xtaydi, etc.

– Ikki davlat o'rtasidagi yo’lni real dunyodagi aksi bo'lsa ajralmaslikka amal qiladi.

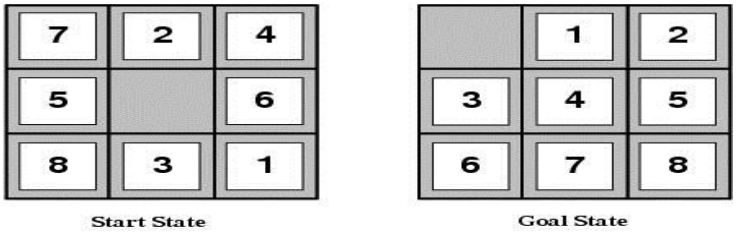
Yechim = real dunyoda yechimning real yo'llari belgilangan. Har bir real muammoga qaraganda mavhum harakat osonroq.

Misol: vacuum dunyosi

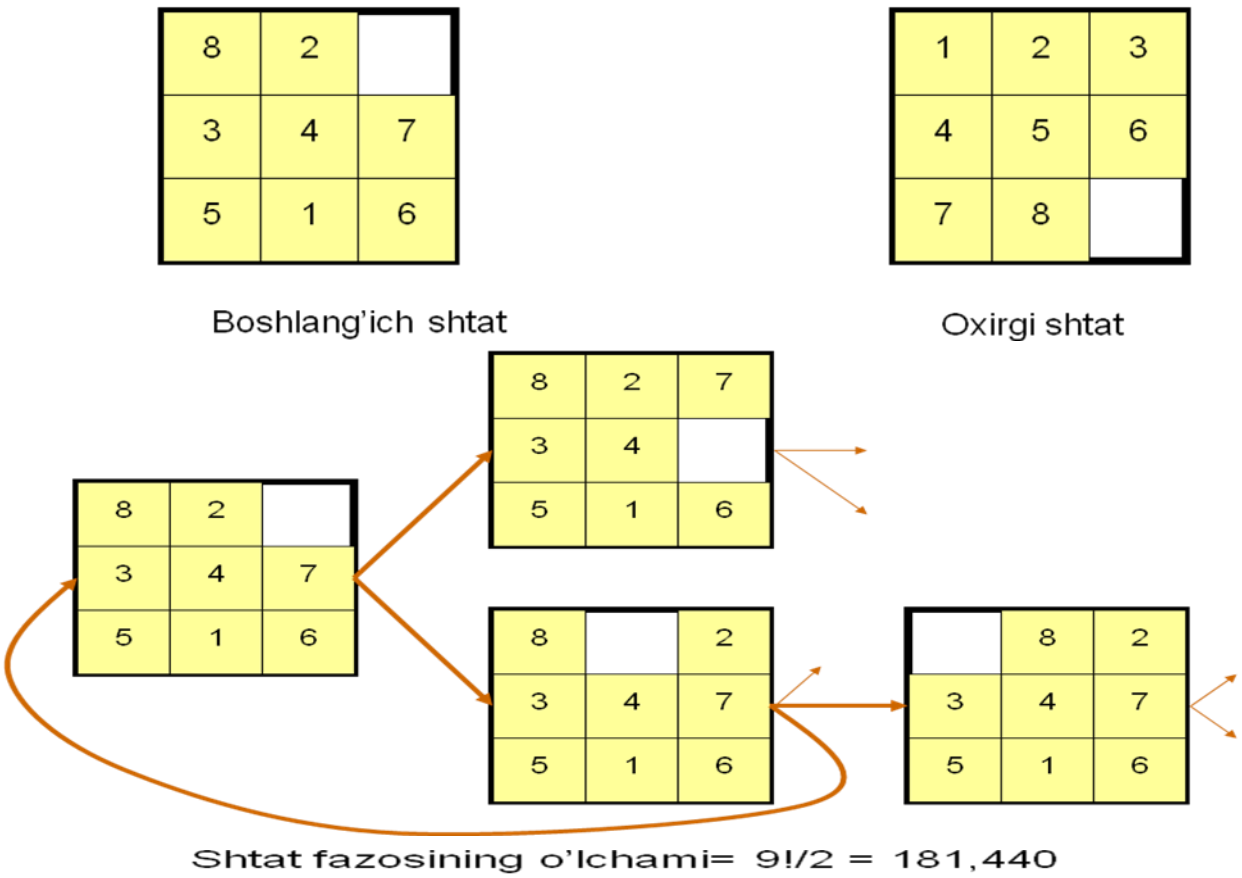


- Shtatlar ??
- Boshlang'ich Shtat ??
- Harakatlari ??
- Maqsad test ??
- Yo'l qiymati ??
- Shtatlar?? Quruqlikdan tashqari ikki yo’naish bor: 2 x 2^2=8 Shtatlar.
- Boshlang’ich shtat?? Biror shtat boshlang’ich deb olinadi
- Harakatlar?? {*chap*, *o’ng*, *to’g’ri*}
- Maqsad test?? maydonlar toza yoki yo'qligini tekshirish.
- Yo’l qiymati?? maqsadga erishish uchun harakatlar soni.

Misol: 8-puzzle



- Shtatlar ??
- Boshlang'ich shtat ??
- Harakatlari ??
- Maqsad test ??
- Yo'l qiymati ??
- Shtatlar?? Har bir g'isht butun son Manzil
- Boshlang’ich shtat?? Har qanday shtat boshlang'ich bo'lishi mumkin
- Harakatlar?? {*chap*, *o’ng*, *tepa*, *past*}
- Maqsad test?? maqsad konfiguratsiyasiga erishilganligini tekshiring
- Yo’l miqdori?? maqsadga erishish uchun harakatlar soni

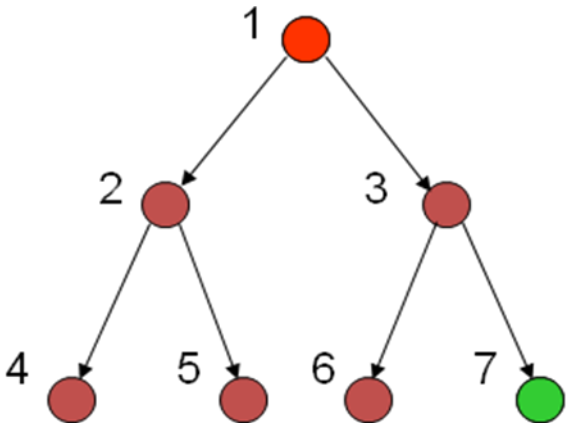


15-jumboq → .65 × 10^{12}

24-jumboq → .5 × 10^{25}

Kengligi-birinchi strategiyasi

- Pastdagi yoyilmagan belgilarga yoyish
- Amalga oshirish: *boshlanishi* FIFO navbati bo'yicha
- Yangi belgilarni navbat oxirida kiritish



Takrorlash uchun savollar:

1. Muammoni hal qilishda masofani aniqlash hususiyatlari?
2. Vacuum dunyosi agenti algoritmi?

3. Muammoni hal qilishda strategiyalar axamiyati?

7. Ma’ruza
Qaror qabul qilishning Markov jarayonlari

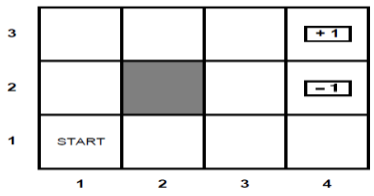
Reja:

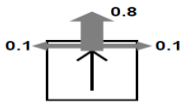
1. Tasodifiy muhitlar uchun umumlashtirilgan qiymatlarga ega atom modeli
2. Taxminiy/Ehtimolli rejalashtirish Markov qaror qabul qilish jarayonlari (MDP)

Tayanch iboralar: *tasodif, taxminiy, ehtimolli, jarayon*

Taxminiy/Ehtimolli rejalashtirish Markov qaror qabul qilish jarayonlari (MDP)

- Atom modeli: chizma qidiruv
- Proportsional modellar: biz muhokama qilgan PDDL rejalashtirish



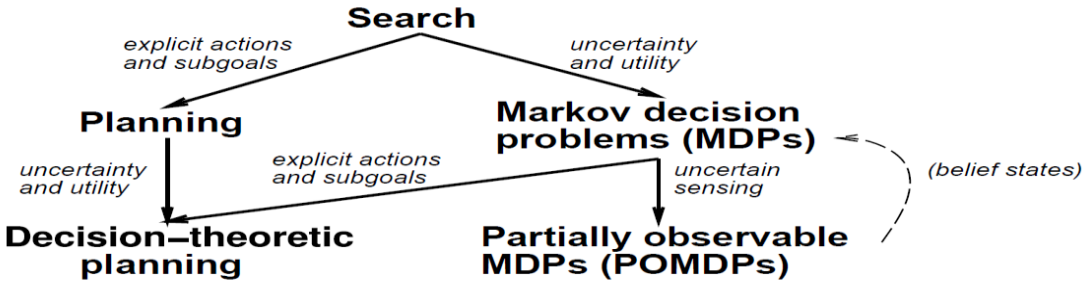


Model $M_{ij}^{a} \equiv P(j|i,a)$ = probability that doing a in i leads to j

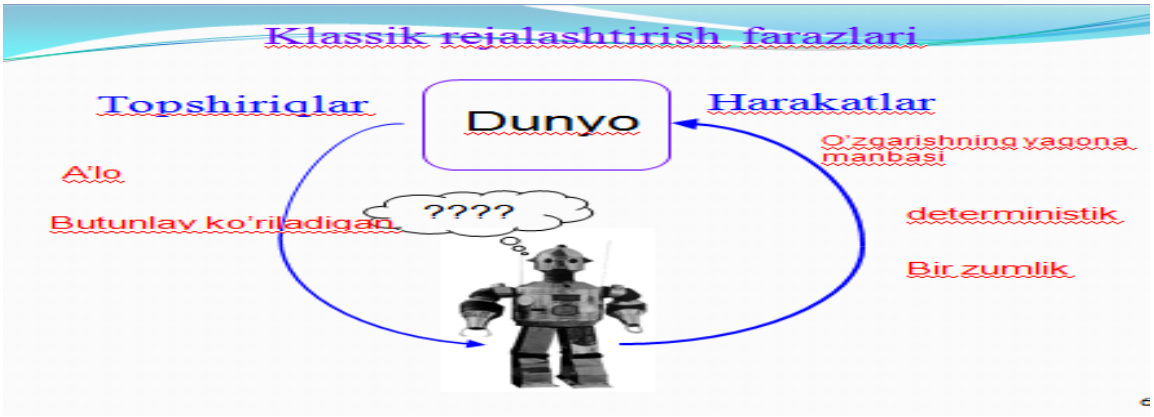
Each state has a *reward* $R(i)$
= -0.04 (small penalty) for nonterminal states
= ± 1 for terminal states

Ma’lum ehtimolliklar bilan harakat sizni ko’plab holatlardan biriga olib borishi mumkin.

- Har bir holatga o’tganda siz qiymatga ega bo’lasiz.
- Maqsad: sizning umumiy (Kulmulativ) qiymatingizni oshirishdir
- Yechim qanday?



Klassik rejalashtirish farazlari



Noaniqlikning turlari.

Dizyunaktiv (Determinestik bo'lamgan rejalashtirishda foydalaniladi)

keyingi holat holatlar majmuyining biri bo'lishi mumkin.

Tahminiy/ Ehtimolli holatlar

keyigi holat holatlar majmuyidan taqsimlangan ehtimollikdan olinadi.

Bu modellar qanday bog'langan?

Markov qaror qabul qilish jarayoni

MQQQ ning to'rtta komponenti bor: S, A, R, T:

- (chekli) S holat majmuyi (|S| = n)
- (chekli) A harakat majmuyi (|A| = m)
- (Markov) o'tish funksiyasi T(s,a,s') = Pr(s' | s,a)
- s holatda a harakatni bajarganda s' holatga o'tish ehtimolligi
- Taqdim etish uchun nechta parametr zarur?

Cheklangan, real-qadrlanadigan (Markov) qiymat funksiyasiR(s)

- s holatda bo'lganlik uchun biz tezkor qiymatni olamiz
- Masala, maqsadga asoslangan domen R(s) maqsad holati uchun 1 qiymatga va qolgan barcha boshqa holatlar uchun 0ga teng bo'lishi mumkin.
- Harakat harajatlarini umumlashtirish mumkin: R(s,a)
- Tasodifiy harakat bo'lishi uchun umumlashtririlishi mumkin.

Sanaladigan va davomli holat va harakat joylariga osongina umumlashtirilishi mumkin (ammo algoritmlar turlicha bo'ladi)

Taxminlar

Birinchi darajali Markov dinamikasi (tarix mustaqilligi)

$$Pr(S_{t+1}|A_t,S_t,A_{t-1},S_{t-1},...,S_0) = Pr(S_{t+1}|A_t,S_t)$$

Keyingi holat faqatgina hozirgi holat va harakatga bog'liq bo'ladi.

Birinchi darajali Markov qiymat jarayoni

$$Pr(R_t|A_t,S_t,A_{t-1},S_{t-1},...,S_0) = Pr(R_t|A_t,S_t)$$

Qiymat faqatgina hozirgi holat va harakatga bog'liq bo'ladi.

Oldinroq tasvirlanganidek, qiymat R(s) determinallashgan funksiya orqali beriladi deb hisoblaymiz.

i.e. $Pr(R_t=R(S_t) \mid A_t,S_t) = 1$

Statsionar dinamikalar va qiymat

$Pr(S_{t+1}|A_t,S_t) = Pr(S_{k+1}|A_k,S_k)$ barcha t, k uchun

Jahon dinamikasi mutlaq vaqtga bog'liq emas

To'liq tahlil

Garchi biz harakatni amalga oshirganimizda qaysii holatga erishishimizni aytolmasakda, u aniqlanganda nimaligini bilamiz.

Siyosatlar (MQQlar uchun "planlar

- Statsionar bo'lmagan siyosat [garchi *statsionar dinamikalar va qiymatimiz bo'lsa ham]*
  - π:S x T → A, bu yerda T manfiy bo'magan butun sonlar
  - π(s,t) bu s holatda t qadam bilan bajarish
  - Nima bo'aldi agar biz buni cheksiz bajarishni

  davom ettirsak?
- Stationary siyosat
  - π:S → A
  - π(s) bu s holatda bajarish (vaqtga bog'liq bo'magan holda)
  - Davomiy reaktiv kontrol qiluvchilarni anglatadi
- Bu quyidagi qiymatlarni tahmin qiladis:
  - To'liq tahlil
  - Tarix mustaqilligi
  - Deterministik harakat tanlovi

Agar sen 20 yoshda bo'lsang va liberal bo'lmasang, sen yuraksizsan. Agar sen 40 yoshda bo'lsang va konservativ bo'lmasang sen aqlsizsan Cherchill.

Nimaga harakatlar ketma-ketligini hisobga olish kerak emas?

Nimaga boshqatdan planlashtrirish kerak emas?

Siyosatli qiymat

- Π siyosati qanchalar yaxshi?
- How do we measure "accumulated" yig'ilgan reward?
- **Value function** V: S →ℝ associates value with each state (or each state and time for non-stationary π)
- $V_\pi(s)$ denotes value of policy at state s
  - Depends on immediate reward, but also what you achieve subsequently by following π
  - An optimal policy is one that is no worse than any other policy at any state
- The goal of MDP planning is to compute an optimal policy (method depends on how we define value).

Takrorlash uchun savollar:

1. Qaror qabul qilish algoritmlari?
2. Markov siyosati mohiyati?
3. Markov qaror qabul qilish jarayoni ahamiyati?
4. Qaror qabul qilishda tahminlar va qiymatlar?

8. Ma’ruza
O’yinlar nazariyasi

Reja:

1. Kirish
2. Qidiruv daraxti
3. Jumboq boshlang'ich

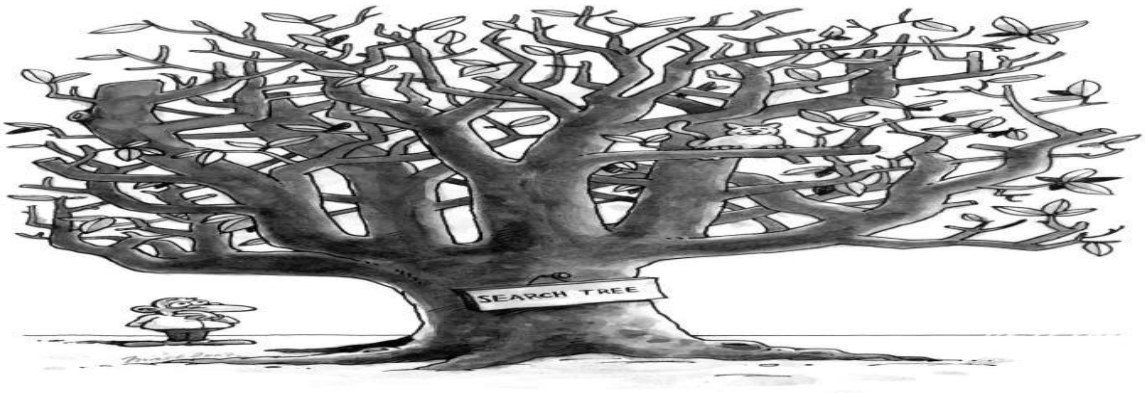
Tayanch iboralar: *qidirish, jumboq, g’oya, nazariya,o’yin.*

Chuqurligi 14 m daraxt ildizini qidirish.
Misol: Shaxmat
gal omili b = 30, chuqurlik d = 50:
qoldiradi. $30^{50} \approx 7.2 \cdot 10^{73}$!
xulosa qadamlar soni = $\sum_{d=0}^{50} 30^d = \frac{1-30^{51}}{1-30} = 7.4 \cdot 10^{73},$
10000 Kompyuter soniyada har bir milliard amallar bajaradi yo'qotishsiz paralell olib borish.
Hisoblash vaqti:

$$\frac{7.4 \cdot 10^{73} \text{ inferences}}{10000 \cdot 10^9 \text{inferences/sec}} = 7.4 \cdot 10^{60} \text{ sec} \approx 2.3 \cdot 10^{53} \text{ years},$$

- 10^{43} times age of the universe

A og'ir kesilgan qidiruv daraxti {yoki: Qaerda, mening mushugim bor? "



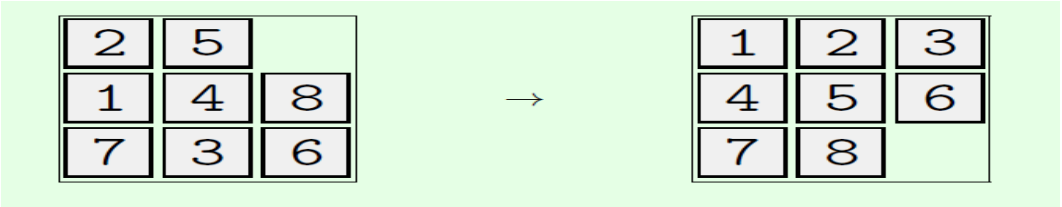
Savollar:

Nima uchun yaxshi shaxmatchilar ham bugungi kunda kelib ompyuterlarday yaxshi shaxmat o’ynolmaydi?

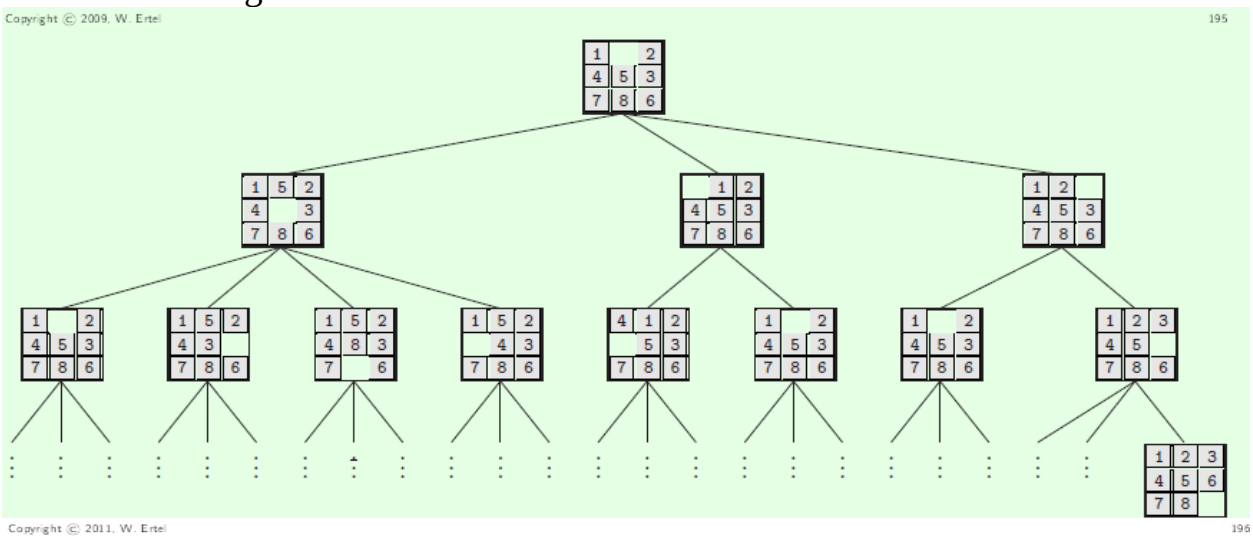
Nima uchun taklif uchun hujjatlar matematik hal qilayotgan bo'sh joy izlashga katta hatto?

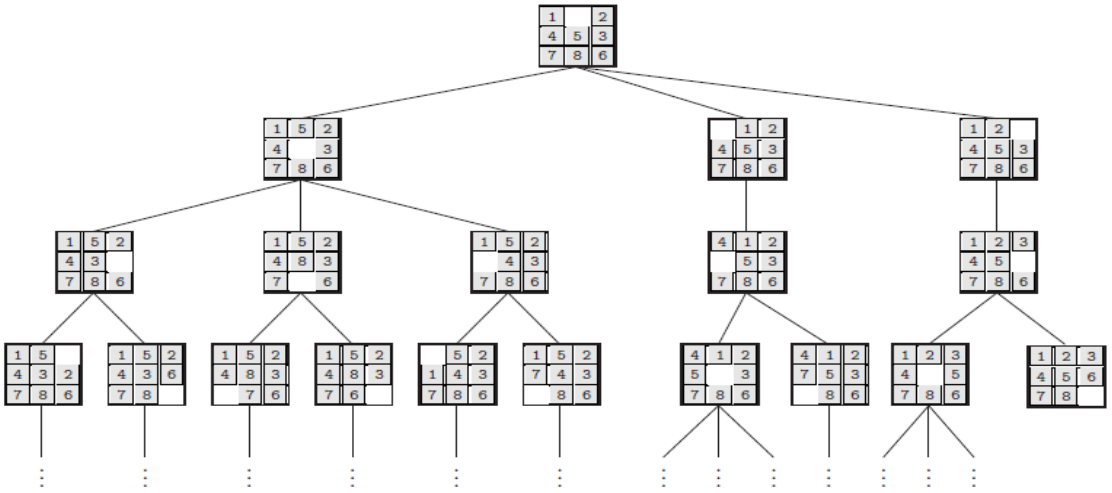
Misol:

8 ta jumboq:



o'rtacha shoxlangan omil= $\sqrt{8} \approx 2.83.$





De tanituvchi 6,1 A qidiruv muammo ham emas?

Quyidagi qadriyatlar bilan Ned Davlat:

dunyo davlatlarning tavsifi qaysi agent NDS o'zi?.

Davlat bo'ylab: a

gent Izlay boshladi bo'lgan boshlang'ich davlat.

Davlatning Maqsadi :

maqsad agent davlatni sotgan bo'lsa, u yakun yasaydi va eziladi bir yechim (agar kerakli).

Actions:

agentlari barcha harakatlarini berdi.

Qarori: maqsadga muvofiq davlat qidiruv daraxt yo'li.

Ahamiyatli vazifasi: har bir harakat, bir iqtisodiy qiymatini soladi. Zarur uchun?

A tugaydigan iqtisodiy optimal yechim.

Davlat maydoni: barcha davlatlar belgilangan. Daraxt qidiruv : Shtatlari barglari, harakatlari tomonlarining. 8 jumboqni qo'llaniladigannini bilib olish Davlat: 3 (Bir marta har bir) qadriyatlar 1,2,3,4,5,6,7,8 va biri bilan 3 mata S muloqot qiladi. Davlatning bo'sh masohati. Davlatni boshlab, A va n o'zboshimchalik davlat qurolmaydi. davlat maqsad : A va n o'zboshimchalik qurilgan davlat, masalan 6.1 rasmda huquqi berilgan davlatlar ko'rsatilgan/ Actions: bo'sh kvadrat zij harakati chapga (agar j 6 = 1) o'ng, (agar up j 6 = 3), (agar pastga i 6 = 1), (agar i 6 = 3). Ahamiyatli funktsiyasi: doimiy vazifasi 1, barcha harakatlar teng narxli, chunki. Davlat maydoni: davlat kosmik o'zaro bo'lgan domen degenerat bo'ladi erishib bo'lmaydigan (Mashq ??). Shunday qilib 8 jumboq muammolar bor. qidiruv algoritmlari tahlil qilish uchun, quyidagi shartlar talab qilinadi:

D tanituvchi 6,2 teorema . davlat lar s vorisi davlatlar soni d omili deb ataladi (Engl. Dal omil) b (lar), yoki b va d omil doimiy bo'lsa. e va n umumiy tugunlari, chuqurligi d daraxt omil d ective D omil ham?

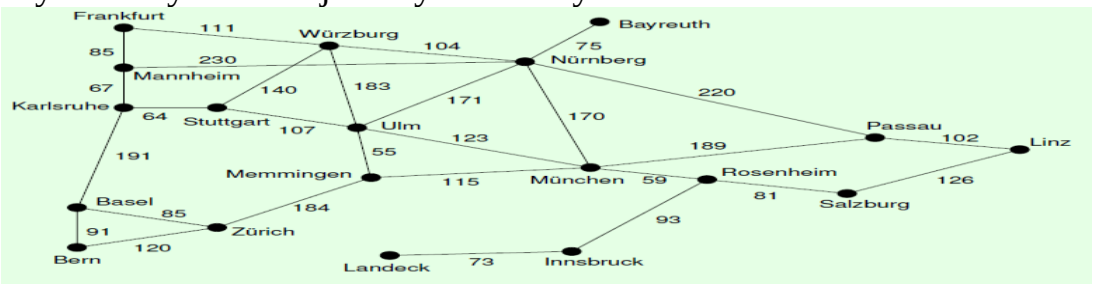
N deb doimiy d omil bilan daraxt, teng chuqurligi va n ga teng n (?? Mashq qarang) bo'ladi A qidiruv algoritm unga, to'liq deb ataladi?

NDS hal bo'ladigan bir yechim muammo to'liq qidiruv algoritmi yechim ga muhtoj holda yasaydigan bo'lsa, keyin muammo Unsolvable hisoblana

- tenglamani yechish = $n = \sum_{i=0}^{d} b^i = \frac{b^{d+1} - 1}{b - 1}$

b ni beradi? omil ective og'ir uchun teoremasi 6,1 keng konstantasi bilan n qidiruv daraxtlar omili, deyarli barcha tugunlari so'nggi darajada bo'ladi.

Isbot: (Mashq ??). Misol: B shahardan A shaharga eng qisqa yo'l iqtisodiy funktsiyasi bilan janubiy Germaniya chizmasi.



Davlat: sayyohi joriy holati bo'lib, bir shahar. Davlat bo'ylab, : A va n o'zboshimchalik shahar. Davlat maqsadi : A va n o'zboshimchalik shahar. Actions: qo'shni shaharga joriy shahardan Travel. Ahamiyati funktsiyasi: shaharlar o'rtasidagi masofa. Davlat maydoni: Barcha shaharlar, grafik, ya'ni tugunlari.

U bir yechim bor, agar bo'lsa ham tanituvchi 6,3 A qidiruv algoritmi, optimal deb ataladi har doim?

NDS eng kam xarajat bilan hal qiladi. 8 jumboq muammo hisoblanadi deterministik: har bir harakat noyob vorisi davlatga bir davlatdan olib keladi. kuzatiladigan: agent har doim u bilan bo'lgan davlatni biladi. Deb atalmish O foydalanish algoritmlarni, optimal echimlar topish mumkin.

Aks holda: Taqvim ta'lim

Eristik qidiruv

Evristika ko'p hollarda bir yechim topish muammo hal qilish strategiyasi tezroq qidiruv. Hech qanday kafolat yo'q! kundalik hayotda, evristik usullari muhim ahamiyatga ega. Cheklangan resurslardan ostida haqiyqi vaqt-qarorlari tez topiladi A yaxshi yechim optimal bo'lgani afzal, lekin olmoq uchun juda qimmat. davlatlar uchun evristik baholash funksiyasi f (bo'ladi)
Node = davlat + evristik baholash

- HeuristicSearch(Start, Goal)
- NodeList = [Start]
- While True
- If NodeList= ; Return(\No solution")
- Node = First(NodeList)
- NodeList = Rest(NodeList)
- If Goal Reached(Node,Goal) Return(\Solution found", Node)
- NodeList = SortIn(Successors(Node),NodeList)

Depth- RST va birinchi qidirish breadth- funktsiyasi maxsus holatlari

- Evristik qidiruv. (Mashq ??)

Takrorlash uchun savollar:

1. O’yinlar nazariyasining mohiyati?
2. Qanday o’yin turlarini bilasiz?
3. O’yinlarni strategiyasi nima?

9. Ma’ruza
Bayes to’rlari

Reja:

1. Uchar Penguin tushuncasi?
2. Probabilistic logika?
3. Shartli ehtimolliklar bilan fikr?

Tayanch iboralar: *to’r, logika, baza, barqaror, probaltik.*

Tweety bir pingvin bo'ladi. Penguins qushlar bor. Barcha qushlar y bo’lishi mumkin. PL1 yilda rasmiy bilimlar bazasi WEB 1 iborat
pingvin (tweety)
pingvin (x)) qush (x)
qush (x)) y (x)
Bu olingan natija bo’lishi mumkin: y (tweety)
Yangi sinov: Penguins bo’lishi mumkin y emas
pingvin (x)): y (x)
Bu olingan natija bo'lishi mumkin: y (tweety)
Lekin: Bundan tashqari, olingan natija bo’lishi mumkin: y (tweety) bilimlar bazasi barqaror emas. mantiq, bir xillik: yangi bilim eski bilim bekor bo’lishi mumkin emas.

Probabilistic logika

- Noaniqlik: barcha qushlarning 99% y bo’lishi mumkin
- To'liq: Agent holati haqida to'liq ma'lumot
- jahon (real vaqt qarorlari)
- evristik qidiruv
- noma'lum yoki to'liq bilim bilan fikr.

Kelinglar faqat orqaga o'tirib nimadir haqida o'ylab ko'raylik!

Misol:

- Agar bemorga o'ng og'riq hissi bo'lsa pastki qorin ko'tardi
- Hujayra (Leykotsitlar) soni, bu appenditsit bo'lishi mumkin shubhali.
- Oshqozon og'riq o'ng pastki ^ Leykotsitlar> 10000! appenditsit
- Oshqozon og'riq o'ng pastki ^ Leykotsitlar> 10000
- Biz appenditsit olmoq uchun modus ponensdan foydalanishimiz mumkin.

1976, Shortlie va Buchanan . Ishonch omillar faktlar va qoidalar aniq ifodalaydi. A! shartli ehtimollik orqali B

misol:

Oshqozonda og'riq o'ng pastki ^ Leykotsitlar> 10000: 0! 6 Appendikulyar qoidalarini omillar ulash uchun formulalar. Analiz noto'g'ri. Zid natijalar olingan bo'lishi mumkin. non-bir xildagi mantiqi Default Lojik Default mantiq Dempster-Shafer nazariyasi: mantiqiy muddatga Loyqa mantiq bir e'tiqod vazifasini Bel (A) soladi () nazorat nazariyasi

Shartli ehtimolliklar bilan fikr

O'rniga ahamiyatga ega bo'lgan shartli ehtimolliklar (material ma'no) sub'ektiv ehtimolliklar, Ehtimollar nazariyasi yaxshi tashkil qilingan noaniq va to'liq bilim bilan fikr. Maksimal entropiya usuli (Maxent)

- Bayes tarmoqlari

Misol: Ehtimollik aforizmlari o'yinlar, ehtimollik

" a olti "oti b 1 = 6 ehtimoli bo'lgan " Toq o’tib 1 = 2 7,1 belgilar yakunlari majmui bo'lsin. har bir ! 2 eksperiment mumkin bo'lgan natija uchun beriladi. Wii 2 istisno bo'lsa bir-biriga, lekin barcha mumkin bo'lgan natijalarini qamrab, ular elementar hodisalar deyiladi. bir marta namuna o’tib = F1; 2; 3; 4; 5; 6g ham soni f2 o’tib; 4; 6g Boshlang'ich voqea emas 5 F1 kichikroq bir qator tashlab; 2;

3; 4g Boshlang'ich voqea emas Sababi: F2; 4; 6g \ F1; 2; 3; 4g = F2; 4g 6 =; ikki tadbirlar A va B, A bilan [B voqea hisoblanadi. ishonch hosil voqeadir bo'sh majmui imkonsiz voqeadir.

Buning o'rniga A \ B, biz, chunki A ^ B yozish x 2 A \ B, 2 A ^ x 2 B x: A, B, etc .: tasodifiy o'zgaruvchilar. Biz n qiymati qator bilan faqat diskret tasodifiy o'zgaruvchilar ko'rib d da, soni qadriyatlar 1,2,3,4,5,6 bilan ajralgan bo'ladi. Ehtimollik, 5 yoki 6 1 = 3 o'tish:

P (2-sonli F5; 6g) = P (soni = 5 _ soni = 6) = 1 = 3:

Belgilar 7,2 Let = f 1; ! 2; :::; ! Ng bo'lsin nite. No boshlang'ich voqeadir afzal, Olişhma chastotasi bilan bog'liq nosimmetrik taxmin anglatadi Barcha elementar hodisalar. Tadbir A ehtimoli P (A), keyin bo'ladi tomonidan rad

P (A) = JAJ jj =

A qulay natijalari soni iloji natijalari soni Misol: bir qolib uchirish, ham qator uchun ehtimollik hisoblanadi P (2-sonli F2; 4; 6g) = jf2; 4; 6gj jf1; 2; 3; 4; 5; 6gj = 3 6 =12

Bayes' Theorem

$$P(A|B) = \frac{P(A \wedge B)}{P(B)} \quad \text{as well as} \quad P(B|A) = \frac{P(A \wedge B)}{P(A)}$$

Bayes' theorem:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{7.2}$$

Appendicitis example:

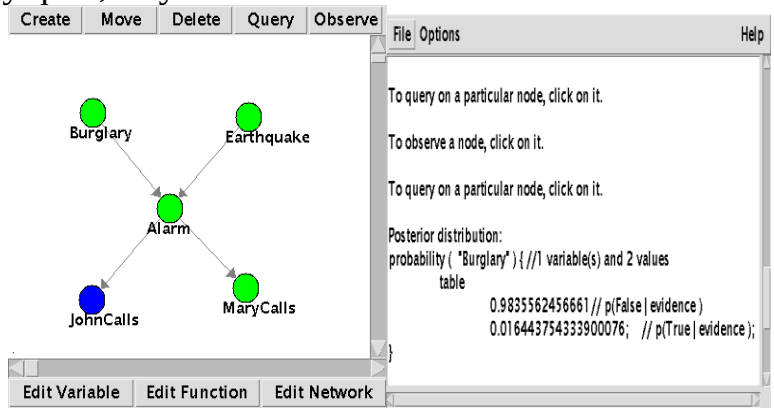
- $P(App|Leuko)$ would be much more interesting for the diagnosis of appendicitis, but is not published!
- Why is $P(Leuko|App)$ published, but $P(App|Leuko)$ not?

$$P(App|Leuko) = \frac{P(Leuko|App) \cdot P(App)}{P(Leuko)} = \frac{0.82 \cdot 0.28}{0.54} = 0.43 \tag{7.3}$$

Juda ishonchli 99% o'g'ri signal ishonchi bilan har qanday o'g'ri bilan suhbat qiladi. Shunday qilib, yuqori ishonch bilan: keyin signal o'g'rilik bo'lsa!

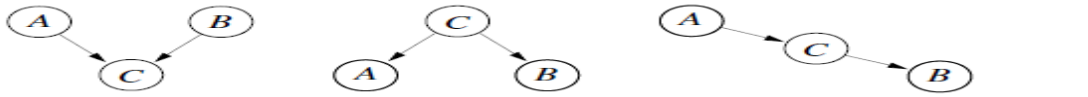
Bayes tarmoqlari va Maxent

- Shartli mustaqillik yoki teng qoidalarini xotima, Maxent tamoyili shu demakdir
- shartli mustaqillik va Bayes tarmoq sababli ham bir xil javob.
- 5 Shunday qilib, Bayes Networks Maxent maxsus ishi bor



Batafsil kuchli va qulay professional vositasi hugin bo'ladi. Uzluksiz o'zgaruvchilar ishlashi mumkin. Bundan tashqari, Bayes tarmoqlarini o'rganishimiz mumkin, ya'ni, to'la avtomatik ravishda tarmoq hosil qilish mumkin. Statistik ma'lumotlarga ishlatiladi

Bayes tarmoqlari semantikasi



There is no edge between A and B if they are independent (left) or conditionally independent (middle, right).

Bayes tarmoq N o'zgaruvchan uni oldin hech qanday o'zgaruvchining ancha past raqami bor yo'qligiga ega: talablar bu tutadi.

P (Xn 1;::; Xn□1) = P (Xj Ota-onalar (Xn)):

Bugungi kunda, Bayes xulosasi juda muhim va Bayes tarmoqlari manzilsiz Bayes tarmoqlari adabiyot cheklovlar ostida sababiyat uzluksiz argumentlarni tashish puxta ishlab chiqilgan bo'lib hisoblanadi.

Takrorlash uchun savollar:

1. Bayes to’rlari tahlili?
2. Bayes tarmoqlari semantikasining mohiyati?
3. Shartli ehtimolliklar?

10. Ma’ruza
Mashinali o’qitishi va ma’lumotlarni inteklektual tahlili

Reja:

1. Curve darslari
2. Approksimatsion metodlar
3. Shartlar

Tayanch iboralar: *curve, model, apraksiya, shart,amal, mantiq.*

Nima uchun mashinani o'rganamiz?

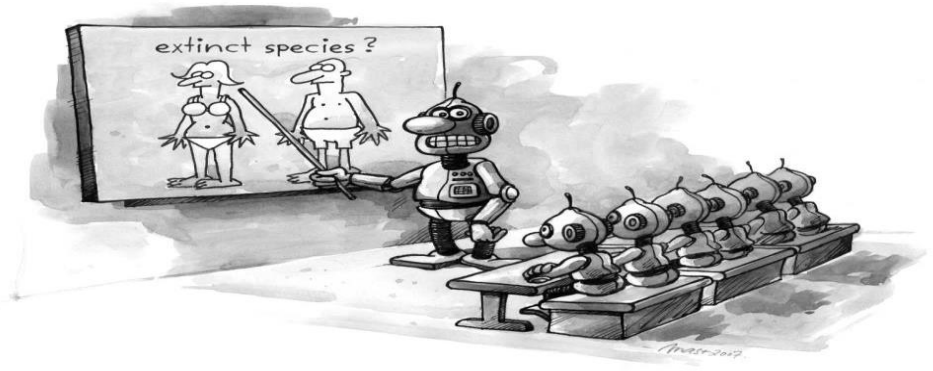
Elaine Rich1:

" Endi moliyaviy razvedka kompyuterlar qilish uchun qanday o'rganish hisoblanadi. Hozir, unday narsadan, odamlar yaxshiroqdir. ", va: Odamlar kompyuterlar ko'ra yaxshiroq o'rganishadi). Mashina ta'lim AI uchun juda muhim

1Rich, dasturiy ta'minot ishlab chiqish murakkabligi avtonom robot o'zini tutish ekspert tizimlari spam filtri O'zim: dasturlashtirilgan va o'rgangan komponentlar bilan gibrid dasturi.

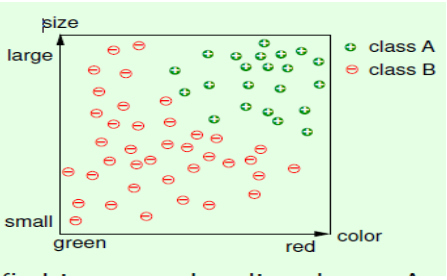
- chet tilida so'z o'rganish?
- bir she'r yodlashga?
- matematik ko'nikmalarini o'zlashtirish?

- tosh o'rganish?

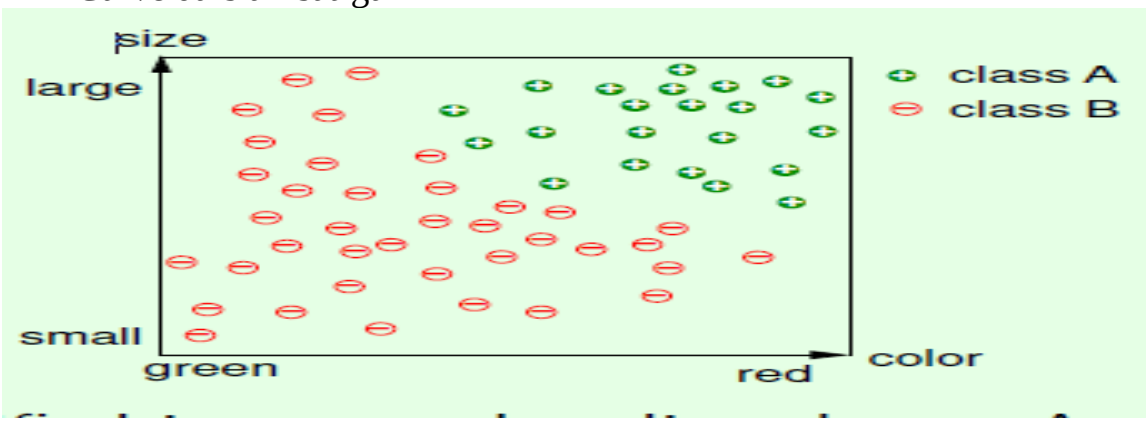


Misol: olma uchun qurilmani tartiblash
Xususiyatlar: Hajmi va rang Tasnifi vazifa: tovarlar sinflar A va B ichiga Divide olma (tasn Hajmi [sm] 8 8 6 3. . . Rang [0: yashil, 1: qizil] 0,1 0,3 0,9 0,8. . .sinf B A A B.iflagichi) olma saralash agenti uchun Training ma'lumotlar.

Olma saralash uskunalari va tovarlar sinflar A tasniflanadi ba'zi olma va xususiyati kosmosda B.



Curve darslari satrga



Amalda: 30 yoki undan ko'p xususiyatlar! n Xususiyatlar: n-o'lchovli fazoda xususiyati esa, bir (n - 1) -dimensional alt düzl emi iloji boricha yaxshi ikki darslari ajratib, qaysi topish uchun qilindi

Shartlari

Siniflanish: a sinf qiymatiga bir xususiyati vektor xaritalar. Shu bilan bir qatorda sobit qoladi.Misol: olma saralash. Yondashish: haqiqiy raqamiga xususiyati vektor xaritalar. Misol: Berilgan xususiyati qadriyatlar chiqib prognozlashtirish ulushi narxlarni tanlaydi

Approksimatsion metodlar

Oliy o'lchovda muammolarni hal qilish bizga model-bepul, taxminan qadriyatlari kerak asab tarmoqlari k-NN imkoniyatlar muammolarni tahlili

$$\hat{f}(x) = \frac{1}{k} \sum_{i=1}^{k} f(x_i)$$

Oddiy yodlash bilan o'rganish. Shuning uchun, juda tez o'rganish. A vektor x sınıflanish hamjihatlikni juda qimmati bo'ladi. N o'quv ma'lumotlar bilan k Yaqin qo'shni topish :? (N). Umumiy hisoblash vaqti :? (N + k). Eng yaqin qo'shni usullari = dangasa ta'lim.

Eng yaqin qo'shni usullariga javob beradi, yuqori qiymatga yaxshi mahalliy taxminiy kerak, lekin muammolarni hal qilish uchun tizimining tezligiga talabni eng yaqin qo'shni usullari mos emas, ma'lumotlar olingan bilim bayoni kerak bo'lsa, qaysi insonlar (ma'lumotlar konchilik) tomonidan tushunarli bo'lishi kerak.

Takrorlash uchun savollar:

1. Mashinali o'qitishda ma'lumitlar o'rni.
2. Mashinali o'qitishda approksimation usullarni qo'llash.
3. Mashinali o'qitishda shartlar va usullar hilma – hilligi.

11 Ma'ruza
Mashinali o'qitish. Sonli yondashuv

Reja:

1. Kirish.
2. Asosiy tamoyillar va idrok etishning yaxlitligi.
3. Belgilarni tanish.

Tayanch iboralar: *tamoillar, idrok,son, mashina.*

Kirish

«Kompyuterga matnni avtomatik kiritadigan» maxsus tizimlarning mavjudligini hatto boshlang'ich foydalanuvchilar ham bilishadi. Ko'rinishidan bu oddiy va mantiqiy. Skaner qilingan tasvirdan tizim xarflarni «taniydigan» qismlarni topadi va bu tasvirlarni haqiqiy harflar bilan, boshqacha aytganda ularni mashina kodi bilan almashtiradi. Matn tasviridan «haqiqiy» matnga o'tish shunday amalga oshiriladi. Bunga qanday erishiladi?

«Bit» kompaniyasi tomonidan «Fontanli almashtirish» deb nomlangan belgilarni tanishning maxsus texnologiyasi ishlab chiqarilgan. Bu texnologiya asosida yuqori bahoga ega bo'lgan savdo maxsuloti ishlab chiqarilgan. Bu Fine Reader optik tanish tizimidir. Hozirgi paytda uning uning uchinchi versiyasi ishlab chiqarilgan bo'lib, nafaqat matnlar bilan, balki shakllar va jadvallar bilan ham ishlash imkoniga ega. Ishlab chiqaruvchilar esa uning to'rtinchi versiyasi nafaqat bosma matnlarni, hatto qo'lyozma matnlarni ham taniy oladigan bo'lishini bashorat qilishmoqda.

Asosiy tamoyillar va idrok etishning yaxlitligi.

«Fontanli almashtirish» asosida yaxlitlilik tamoyili yotadi. Unga muvofiq ixtiyoriy idrok etiladigan ob’ekt yaxlit, ma’lum munosabatlarda o’zaro bog’langan qismlardan iborat holda qaraladi. Masalan bosma sahifa bo’limlardan tashkil topgan, bo’limlar sarlavhalar va kolonkalardan, kolonkalar abzatslardan tashkil topgan. Abzats satrlardan, satrlar so’zlardan, so’zlar haflardan tashkil topgan. Bunda matnning barcha sanab o’tilgan elementlari bir - biri bilan ma’lum fazoviy va til munosabatlar orqali bog’langan. Yaxlitlikni ajratish uchun uning qismlarini aniqlash kerak. Qismlarni o’z navbatida faqat yaxlitlikning tarkibida qarash mumkin. SHuning uchun idrok etishning yaxlit jarayoni faqat idrok etiladigan ob’ekt haqidagi gipotezalar doirasida yaxlit sodir bo’lishi mumkin. Idrok etiladigan ob’ekt haqidagi faraz o’rtaga tashlangandan keyin uning qismlari ajratiladi va interpretatsiya qilinadi. SHundan so’ng boshlang’ich gipotezaning to’g’riligini tekshirish uchun ulardan yaxlitlikni «yig’ish» ga harakat qilinadi. SHubhasiz, idrok etiladigan ob’ekt kattaroq yaxlitlik doirisida interpretatsiya qilinishi mumkin.

Biror gapni o’qib inson harflarni taniydi, so’zlarni idrok etadi, ularni sintaktik konstruktsiyalarga bog’laydi va uning ma’nosini tushunadi.

Texnik tizimlarda matnni tanishdagi ixtiyoriy yechimga birdaniga kirishilmaydi, balki gipotezalarni ketma - ket tekshirish va oldinga surish hamda tadqiq qilinadigan ob’ekt haqidagi bilimlarni ham, umumiy kontekstdagi bilimlarni ham jalb qilish orqali kirishiladi. Idrok etiladigan ob’ekt sinflarining yaxlit tavsifi ikkita shartga javob beradi: birinchidan, berilgan sinfdagi barcha ob’ektlar bu tavsifni qanoatlantiradi, ikkinchidan boshqa sinfdagi hech qanday ob’ekt uni qanoatlantirmaydi. Masalan, «K» harfining tasvirlari sinfi shunday tavsiflanishi kerakki «K» harfining ixtiyoriy tasviri unga tushsin, boshqa barcha xarflarning tasvirlari esa unga tushmasin. Bunday tavsif aks etish xususiyatiga ega bo’ladi, ya’ni tavsiflanadigan ob’ektlarni qayta ishlab chiqishni ta’minlaydi: OCR tizimlari uchun xarfning etaloni xarfni vizual qayta ishlab chiqishga imkon beradi, nutqni tanish uchun so’zlar etaloni so’zlarni talaffuz etish imkonini beradi, sintaktik analizatordagi gapning strukturali tavsifi to’g’ri gapni sintez qilish imkonini beradi. Amaliy nuqtai nazardan aks etish katta rol o’ynaydi, modomiki tavsiflarning sifatini effektiv nazorat qilishga imkon beradi.

Yaxlit tavsiflashning ikki ko’rinishi mavjud: shablonli va strukturali.

Birinchi holda tavsiflash vektorli yoki rastrli ko’rinishdagi tasvirni o’zida aks ettiradi va almashtirishlar sinfi beriladi(masalan, takrorlash, masshtablashtirish va x.k.)

Ikkinchi holda tavsiflash graflar ko’rinishida aks ettiriladi. Grafning tugunlari kiruvchi ob’ektning tashkil etuvchi elementlaridan iborat, yoylari esa ular o’rtasidagi fazoviy munosabatlardan iborat. O’z navbatida elementlar murakkab bo’lishi mumkin(ya’ni o’zining tavsifiga ega bo’lishi mumkin).

Albatta, shablonli tavsiflashni strukturali tavsiflashga qaraganda amalga oshirish ancha oson. Lekin uni yuqori o’zgarish darajasiga ega bo’lgan ob’ektlarni tavsiflash uchun qo’llab bo’lmaydi. SHablonli tavsiflashni masalan, faqat bosma belgilarni tanish uchun, strukturali tavsiflashni esa qo’lyozma matnlarni tanishda ham qo’llash mumkin.

Idrok etishning to’liqligi ikkita muhim arxitekturali yechimlarni taklif qiladi. Birinchidan, barcha bilimlar manbai imkon qadar bir vaqtda ishlashi kerak. Masalan, avval sahifani tanib, so’ngra uni lug’at va kontekst qayta ishlashga berish mumkin emas, modomiki bu holda konteks qayta ishlashdan tanishga qayta aloqani amalga oshirish mumkin bo’lmaydi. Ikkinchidan, tadqiq qilinadigan ob’ekt imkon qadar yaxlit holda aks etishi va qayta ishlanishi kerak.

Idrok etishning birinchi qadami - bu idrok etiladigan ob’ekt haqidagi gipotezani shakllantirishdan iborat. Gipoteza ob’ektning aprior modeli, konteksti va oldingi gopotezalarning natijalarini tekshirish asosida ham(«yuqoridan-quyiga» jarayoni), ob’ektni oldindan analiz qilish asosida ham(«quyidan - yuqoriga») shakllanishi mumkin. Ikkinchi qadam - idrok etishni chuqurlashtirish(gipotezani tekshirish). Bu holda ob’ektni ilgari surilgan gipoteza dorasida qo’shimcha analizi amalga oshiriladi va to’liq kuchni kontekst jalb qiladi.

Idrok etish qulay bo’lishi uchun ob’ektni oldindan qayta ishlashni amalga oshirish zarur. Lekin bu holda ob’ekt haqidagi ma’lumot yo’qolmasligi kerak. Odatda ob’ektni boshlang’ich qayta ishlash kiruvchi ob’ektni keyingi ishlar uchun qulay bo’lgan tasavvurga almashtirishga olib kelinadi(masalan, tasvirni vektorlashtirish) yoki kiruvchi ob’ektning barcha mumkin bo’lgan segmentlash variantlarini olishga olib kelinadi va ularning ichidan gipotezalarni ilgari surish va tekshirish orqali to’g’risi tanlanadi. Gipotezalarni o’rtaga tashlash va tekshirish jarayoni dastur arxitekturasida yaqqol aks etishi lozim. Har bir gopoteza uni baholash yoki boshqasi bilan taqqoslash mumkin bo’lishi uchun ob’ekt bo’lishi kerak. SHuning uchun odatda gipotezalar ketma - ket ravishda o’rtaga tashlanadi, shundan so’ng ro’yxatga birlashtiriladi va oldindan baholash orqali saralanadi. Gipotezani oxirgi tanlashda kontekst va boshqa qo’shimcha bilimlar manbai faol ishtirok etadi.

Hozirgi kunda genetik dasturlash sohasidagi peshqadamlardan biri Stenford universitetida professor Djon Koza rahbarligadi ishlaydigan tadqiqotchilar guruhi hisoblanadi. Genetik dasturlash Djon Makkarti guruhi tomonidan ro’yxatlarni qayta ishlash va funktsional dasturlash uchun mo’ljallangan, allaqachon unutilgan LISP(List Processing) tiliga yangi hayot bag’ishladi. Aynan shu til AQSHda sun’iy ong masalalalari uchun keng tarqalgan dasturlash tillaridan bo’lgan va bo’lib qolmoqda.

Belgilarni tanish.

Hozirgi kunda belgilarni tanishda uchta yondashuv ma’lum - shablonli, strukturali va belgili. Lekin yaxlitlik tamoyiliga faqat birinchi ikkitasi javob beradi.

SHablonli tavsiflashni amalga oshirish uchun oson, ammo, strukturaliga qaraganda u, shakllarning turli - tumanligiga ega bo’lgan murakkab ob’ektlarni tavsiflash imkonini bermaydi. Aynan shuning uchun shablonli tavsiflash faqat bosma belgilarni tanish uchun, ayni vaqtda strukturali tavsiflash ko’proq shakl variantlariga ega qo’lyozma belgilarni tanishda qo’laniladi.

3.1. SHablonli tizimlar. Bunday tizimlar aloxida belgining tasvirini rastrliga almashtiradi, uni bazada mavjud bo’lgan barcha shablonlar bilan taqqoslaydi va kiruvchi tasvirdan eng kam nuqtalar bilan farq qiluvchi shablonni tanlaydi.

SHablonli tizimlar tasvir kamchiliklariga yetarlicha bardoshli va kirituvchi ma’lumotlarni qayta ishlashda yuqori tezlikka ega, ammo shabloni unga ma’lum bo’lgan shriftlarnigina yaxshi taniy oladi. Agar taniladigan shrift etalondan ozgina farq qilsa, shablonli tizimlar hatto yuqori sifatli tasvirlarni qayta ishlashda ham xato qilishi mumkin.

3.2. Strukturali tizimlar. Bunday tizimlarda ob’ekt graf ko’rinishida tavsiflanadi. Grafning tugunlarini kiruvchi ob’ektning elementlari, yoylarini esa ular o’rtasidagi fazoviy munosabatlar tashkil qiladi. Bunday yondashuvni amalga oshiradigan tizim, odatda vektorli tasvirlar bilan ishlaydi. Belgini tashkil etuvchi chiziqlar strukturali elementlar hisoblanadi. Masalan «r» harfi uchun bu vertikal kesma va yoy.

Strukturali tizimlarning kamchiligiga ularning tasvir kamchiliklariga sezuvchanligining yuqoriligini kiritish mumkin. Bundan tashqari bu tizimlar uchun shablonli va belgili tizimlardan farqli ravishda xozirgacha samarali avtomatlashtirilgan o’qitish protseduralari yaratilmagan. SHuning uchun Fine Reader uchun strukturali tavsiflarni qo’lda yaratishga to’g’ri keldi.

3.3. Belgili tizimlar. Bu tizimlarda har bir belgining o’rtacha tasviri n-o’lchovli belgilar fazosidagi ob’ekt sifatida aks ettiriladi. Bu yerda kiruvchi tasvirni tanishda qiymati hisoblanadigan belgilar alifbosi tanlanadi. Hosil qilingan n-o’lchovli vektor etalon bilan taqqoslanadi va tasvir ularning ichidan ko’proq mos keladiganiga tegishli bo’ladi. Belgili tizimlar yaxlitlik tamoyiliga javob bermaydi. Ob’ektlar sinfini tavsiflashni yaxlitligining zarur, ammo yetarli bo’lmagan sharti shundan iboratki, berilgan sinfdagi barcha ob’ektlar tavsifni qanoatlantirishi kerak. Modomiki, belgilarni hisoblashda axborotning ma’lum qismi yo’qolar ekan, faqat berilgan sinfga qarashli ob’ektlarni kiritishga kafolat berish qiyin.

Strukturali - dog’li etalon.

«Fontanli almashtirish» shablonli va strukturali tizimlarning afzalliklarini o’zida birlashtiradi va bizning fikrimizcha, ularning har biriga alohida xos bo’lgan kamchiliklardan qutulishga imkon beradi. Bu texnologiyaning asosida strukturali - dog’li etalonni qo’llash yotadi. U tasvirni belgining strukturasini beradigan bir - biri bilan n-ar munosabatlar orqali bog’langan dog’lar to’plami ko’rinishida tasvirlashga imkon beradi. Bu munosabatlar(ya’ni dog’larning bir - biriga nisbatan joylashishi) belgilarni tashkil etadigan strukturali elementlarni yuzaga keltiradi. Masalan kesma dog’lar orasidagi n-ar munosabatlarning bir turi. Ellips - boshqasi, yoy - uchinchisi. Boshqa munosabatlar belgini tashkil etuvchi elementlarning fazoviy joylashishini beradi.

Etalonda quyidagilar beriladi:

1 Nom;

2 majburiy, taqiqlangan va majburiy bo’lmagan strukturali elementlar;

3 strukturali elementlar orasidagi munosabatlar;

4 strukturali elementlarni belgini tavsiflaydigan to’rtburchak bilan bog’laydigan munosabat;

5 strukturali elementlarni ajratishda ishlatiladigan xususiyatlar;

6 elementlar orasidagi munosabatlarni tekshirishda ishlatiladigan atributlar;

7 elementlar va munosabatlarning sifatini baholashda ishlatiladigan atributlar;
8 elementni ajratish boshlanadigan vaziyat;

Tasvirlar sinfi uchun ajratiladigan strukturali elementlar dastlabki va qo'shma bo'lishi mumkin. Dastlabki strukturali elementlar - bu dog'lar, qo'shmalari - kesma, yoy, xalqa, nuqta. Qo'shma strukturali elemenlar sifatida etalonda tavsiflangan ixtiyoriy ob'ektlar olinishi mumkin. Bundan tashqari ular dastlabki struktulari elementlar orqali ham boshqa qo'shma strukturali elementlar orqali xam tavsiflanishi mumkin.

Masalan koreyscha ierogliflarni tanishda (bo'g'inli xat) bo'g'inni tavsiflash uchun qo'shma element alohida harflarning tavsiflari hisoblanadi. Qo'shma strukturali elementlarni qo'llash taniladigan ob'ektlar sinflarining ierarxik tavsiflarini qurishga imkon beradi.

Munosabatlar sifatida strukturali elementlar orasidagi bog'lanishlar ishlatiladi. Bu bog'lanishlar yoki elementlarning metrik xarakteristikalari(masalan, <uzunligi katta>) orqali aniqlanadi yoki tavsirda ularning o'zaro joylashishiga(masalan, <chaproqda>, <kesishadi>) qarab aniqlanadi.

Strukturali elementlar va munosabatlarni berishda muayyan sinf etalonida bu elementni ishlatganda strukturali element yoki munosabatni aniqlashga imkon beradigan aniqlashtiruvchi parametrlar qo'llaniladi. Strukturali elementlar uchun aniqlashtiruvchilar sifatida masalan kesmaning mumkin bo'lgan yo'nalishini beradigan diapazon parametrlari bo'lishi mumkin. Munosabatlar uchun esa xarakterli nuqtalar orasidagi mumkin bo'lgan chegaraviy masofani beradigan parametr bo'lishi mumkin.

Muayyanlashtiruvchi parametrlar tasvirdagi muayyan strukturali elementining <sifatini> va berilgan munosabatning bajarilish <sifatini> hisoblashda ham ishlatiladi.

Taniladigan ob'ektlar sinflari uchun strukturali-dog'li etalonlarni qo'rish va sinash murakkab va og'ir jarayon. Tavsiflarni sozlash uchun ishlatiladigan tasvirlar bazasi har bir grafema uchun yaxshi va yomon namunalarga ega bo'lishi kerak. Bazadagi tasvirlar o'rgatuvchi va nazorat qiluvchi to'plarga ajratiladi. Tavsifni yaratuvchi strukturali elementlar va ular orasidagi munosabatlarni oldindan beradi. Tasvirlar bazasi asosida o'rgatuvchi tizim avtomatik ravishda elementlar va munosabatlarning parametrlarini hisoblaydi. Hosil qilingan etalon berilgan grafemalarning tasvirlarini nazoratli tanlovi asosida tekshiriladi va to'g'irlanadi. Nazoratli tanlov asosida tanish natijalari tekshiriladi, ya'ni gopotezani tasdiqlash sifati baholanadi.

Strukturali - dog'li etalonni qo'llash orqali tanish quyidagicha sodir bo'ladi. Etalon tasvir ustiga qo'yiladi va tasvirda ajratilgan dog'lar orasidagi munosabatlar etalondagi dog'larning munosabatlari bilan taqqoslanadi. Agar tasvirda ajratilgan dog'lar va ular orasidagi munosabatlar qandaydar belgining etalonini qanoatlantirsa, u holda bu belgi kiruvchi tasvirni tanish haqidagi gipotezalar ro'yxatiga qo'shiladi.

Cognitive Technologies dan mashinali o'qish darslari.

Tizim <bitta tugma> tamoyili asosida ishlaydi. Bu shuni anglatadiki, <(Skanerla va tani)Skaniruy i Raspoznavay> tugmasini bosganda xujjatni qayta ishlash jarayoni ishga tushadi: skanerlash, sahifani matnli va grafik bloklarga ajratish, matnni tanish, orfografiyani tekshirish va chiquvchi faylni shakllantirish. Buning barchasida nima turadi? Ongli algoritm hujjatning foniga bog'liq ravishda skanerning optimal yorug'ligini avtomatik tanlashga(adaptiv skanerlash), illyustratsiyalarni saqlashga(yoki yechiladigan masalaga bog'liq ravishda keraksiz grafik elementlarini o'chirish) imkon beradi.

Cunie Form da bunga o'xshash mosliklarning bir qancha usullari ishlatiladi. Birinchidan, har bir belgining shakli alohida elementar hodisalarga yoyiladi. Masalan kesishishning bir chizig'idan boshqasigacha bo'lgan qism hodisa hisoblanadi. Hodisalar majmui belgining ixcham tavsifini o'zida ifodalaydi.

Boshqa usullar aloxida belgilar elementlari <og'irlik> larining o'zaro nisbatlariga va ularning xarakterli alomatlarini tavsiflashga asoslanadi. Bu tavsiflarning har biriga mos etalonlar topiladigan ma'lumotlar bazasi mavjud. Tasvirning qayta ishlashga beriladigan elementi etalon bilan taqqoslanadi. SHundan so'ng bu taqqoslashga asoslanib hal qiluvchi funktsiya tasvirning aniq belgiga mosligi haqida hukm chiqaradi. Bundan tashqari past sifatli matnlar bilan ishlashga imkon beradigan algoritmlar ham mavjud. Masalan «yopishib qolgan» belgilarni ajratish uchun optimal ajratishni baholash usuli mavjud. Aksincha, «sochilgan» elementlarni birlashtirish uchun ularni birlashtirish mexanizmlari ishlab chiqilgan.

Cunie Form 96 da biz birinchi marta o'z-o'zini o'qitish(yoki adaptiv tanish) algoritmini qo'lladik. Ularning ishlash printsipi quyidagidan iborat. Har bir matnda aniq va noaniq bosilgan belgilar mavjud. Agar tizim matnni tanigandan keyin aniqlik chegaradan pastda ekanligi aniqlansa, yaxshi bosilgan belgilarning tizim generatsiya qilgan shriftiga asoslangan holda matnni qayta tanish amalga oshiriladi. Bu yerda ishlab chiqaruvchilar ikki turdagi tanish tizimining afzalliklarini birlashtirishgan: birinchisi ixtiyoriy shriftni qo'shimcha o'qitmasdan tanish imkonini beradi, ikkinchisi past sifatli matnlarni tanishda chidamli. Cunei Form 96 qo'llash natijalari shuni ko'rsatdiki, o'z - o'zini o'qitadigan algoritmlarning qo'llanilishi past sifatli matnlarni tanish aniqligini 4-5 marta oshirishga imkon beradi. Asosiysi o'z - o'zini o'qitadigan tizimlar tanish aniqligini oshirishda katta potentsialga ega.

Sintaktik va lug'atli tanish usullari muhim rol o'ynayda va mohiyatiga ko'ra geometrik tanishni taminlashda kuchli vosita bo'lib xizmat qiladi. Lekin ularni samarali qo'llash uchun ikkita muhim masalani yechish kerak bo'ladi. Birinchidan katta lug'atga(100000 so'z) tez murojatni amalga oshirish. Natijada so'zlarni saqlash tizimini qurishga erishildi va bunda har bir so'zni saqlash uchun bir baytdan oshmaydi, murojat esa minimal vaqtda amalga oshirildi. Boshqa tomondan hodisalarning alьternativligiga yo'naltirilgan tanish natijalarini to'g'irlaydigan tizimni qurishga talab qilindi. O'z o'zidan tanish natijalarining alternativligi aniq va harflar kollektsiyasining <moslik bahosi> bilan birga saqlanishiga bog'liq. Lug'atli nazorat esa lug'at bazasini qo'llab, bu baholarni o'zgartirishga imkon

berdi. Natijada lug'atni qo'llash belgilarni qayta tanish sxemasini amalga oshirishga imkon berdi.

Hozirgi kunda tanish aniqligini oshirish masalalari bilan birga tanish texnologiyalari bilan arxivli tizimlarni birlashtirish orqali OCR-texnologilarni qo'llanilish sohalarini kengaytirish masalalari oldingi o'ringa chiqmoqda. Boshqacha aytganda, biz matnni kiritishni amalga oshiradigan monoprogrammadan mijozning hujjatni qayta ishlash sohasidagi masalasini yechadigan avtomatlashtirilgan kompleksga o'tyapmiz. Mana yarim yildirki Cunei Form tashkilotlarda ma'lumotlarni birgalikda kiritishga mo'ljallangan Cunei Form OCR Server tanish serveri bilan chiqarilmoqda, tanish modulini o'z ichiga olgan <Evfrat> elektron arxivi esa qisqa vaqt ichida katta shuhrat qozondi.

SHunday mo'ljal bilan umuman tanish tizimlari haqidagi tasavvurlarni tubdan o'zgartirgan Cunei Form96 Professional komplekti yaratildi.

Qo'lyozma matnlarni tanish

Qo'lyozma matnlarni tanish masalasi bosma matnlarni tanishga qaraganda ancha qiyin. Bosma matnlarni tanishda biz shrift tasvirlarining chegaralangan miqdori bilan ishlasak, qo'lyozma matnlarda esa shablonlar soni o'lchab bo'lmas darajada ko'p. Tasvir elemetlari chiziqli o'lchamlarining boshqa munosabatlari qo'shimcha qiyinchiliklarni keltirib chiqaradi.

Lekin baribir qo'lyozma matnlarni tanish texnologiyasini ishlab chiqarishning asosiy bosqichlaridan o'tilganligini tan olishimiz mumkin. Cognitive Technologies zaxirasida barcha asosiy turdagi matnlar: stilizatsiyalangan raqamlar, bosma matnlar va qo'lyozma belgilarni tanish texnologiyalari mavjud. Ammo qo'lyozma matnlarni kiritish texnologiyalarini moslashish(adaptatsiya) bosqichidan o'tishi talab qilinadi. SHundan so'ng xujjatlarni uzluksiz arxivga kiritish uchun vositalar to'liq amalga oshirilganligini e'lon qilish mumkin bo'ladi. Hozirgi kunda birgina tinish tizimiga ega bo'lish yetarli emas. Tanilgan matnli fayllar bilan nimadir qilish kerak: ma'lumotlar bazasiga saqlash, ularni qidirash, lokal tarmoq orqali uzatish va h.k. Bir so'z bilan aytganda xujjat bilan ishlashning arxivli yoki boshqa tizimi bilan birgalikda ishlash talab qilinadi. SHunga ko'ra tanish tizimi xujjatlar bilan ishlashning arxivli yoki boshqa tizimi uchun utilitga aylanadi.

Xujjatlarni skanerlash va tanish tizimlari tarmoq versiyalarining paydo bo'lishi bilan bizning kompaniyamizda turli xil tashkilotlarda bu texnologiyalardan birgalikda foydalanishning ba'zi bir afzalliklarini amalga oshirishga erishildi. SHu sababli, bizning nuqtai - nazarimizga ko'ra, turli xil darajadagi tashkilotlarda xujjatlar bilan ishlashni avtomatlashtirish masalalarini kompaniyalar bilan birga kompleks yechish haqida gapirish muhim bo'lardi. Cognitive Technologies ga kelsak u tomonidan taqdim etilgan <Evfrat> elektron arxivi, yangi utilitlar va katta loyixalarni amalga oshirishda qo'llaniladigan texnologiyalar ma'lumotlarni kiritish tizimlarini qo'llashni kengaytirishga va xujjatlar bilan ishlashni avtomatlashtirish texnologiyalarini ishlab chiqarishga yo'naltirilgan kompaniya yo'nalishini davom ettirmoqda.

«Fontanli almashtirish» asosida yaxlitlilik tamoyili yotadi. Unga muvofiq ixtiyoriy idrok etiladigan ob’ekt yaxlit, ma’lum munosabatlarda o’zaro bog’langan qismlardan iborat holda qaraladi. Masalan bosma sahifa bo’limlardan tashkil topgan, bo’limlar sarlavhalar va kolonkalardan, kolonkalar abzatslardan tashkil topgan. Abzats satrlardan, satrlar so’zlardan, so’zlar haflardan tashkil topgan. Bunda matnning barcha sanab o’tilgan elementlari bir - biri bilan ma’lum fazoviy va til munosabatlar orqali bog’langan. Yaxlitlikni ajratish uchun uning qismlarini aniqlash kerak. Qismlarni o’z navbatida faqat yaxlitlikning tarkibida qarash mumkin. SHuning uchun idrok etishning yaxlit jarayoni faqat idrok etiladigan ob’ekt haqidagi gipotezalar doirasida yaxlit sodir bo’lishi mumkin. Idrok etiladigan ob’ekt haqidagi faraz o’rtaga tashlangandan keyin uning qismlari ajratiladi va interpretatsiya qilinadi. SHundan so’ng boshlang’ich gipotezaning to’g’riligini tekshirish uchun ulardan yaxlitlikni «yig’ish» ga harakat qilinadi. SHubhasiz, idrok etiladigan ob’ekt kattaroq yaxlitlik doirisida interpretatsiya qilinishi mumkin.

Biror gapni o’qib inson harflarni taniydi, so’zlarni idrok etadi, ularni sintaktik konstruktsiyalarga bog’laydi va uning ma’nosini tushunadi.

Texnik tizimlarda matnni tanishdagi ixtiyoriy yechimga birdaniga kirishilmaydi, balki gipotezalarni ketma - ket tekshirish va oldinga surish hamda tadqiq qilinadigan ob’ekt haqidagi bilimlarni ham, umumiy kontekstdagi bilimlarni ham jalb qilish orqali kirishiladi.

Skeletli til– bu maxsus fan bilimlarisiz ochiq ET.

Skeletli tizimlar bilimlarni va chiquvchi tayor mexanizmlarni strukturalashni ta'minlaydi. Lekin ularga umumiylik va egiluvchanlik yetishmaydi. Ular faqat tor doiradagi muammolarga qaratilgan va ET ishlab chiqaruvchi imkoniyatini qattiq chegaralaydi.

Universal tillar turli amaliy doiradagi har har xil tipdagi muammolarga qaratilgan bo'lib, ular ma'lumotlarni qidirish va ularga ruxsat olish imkoniyatiga ega. Lekin ulardan foydalanish “skeletli tillarga” nisbatan murakkab.

ET ni qurishda yordamchi vositalar

Ekspert bilimlariga ega bo'lishda yordam beradigan va ularni taqdim qiladigan dasturlardan, hamda ET loyihalarini qurishga yordam beradigan dastuarlardan tashkil topgan. Bu vositalar biroz kichkina bo'lib, ikki guruhga bo'linadi:

- Loyihalash tizim vositalari
- Bilimlarni egallash vositalari;

AGE, TIMM, EXPERTEASE tizimlari – bu loyihalash tizimlariga misol.

TEIRESIAS ROGET – bu bilimlarni egallash tizimlariga misol.

TEIRESIAS tizimi Bilimlar bazasi ekspertidan bilimlarni egallashga xizmat ko'rsatadi.

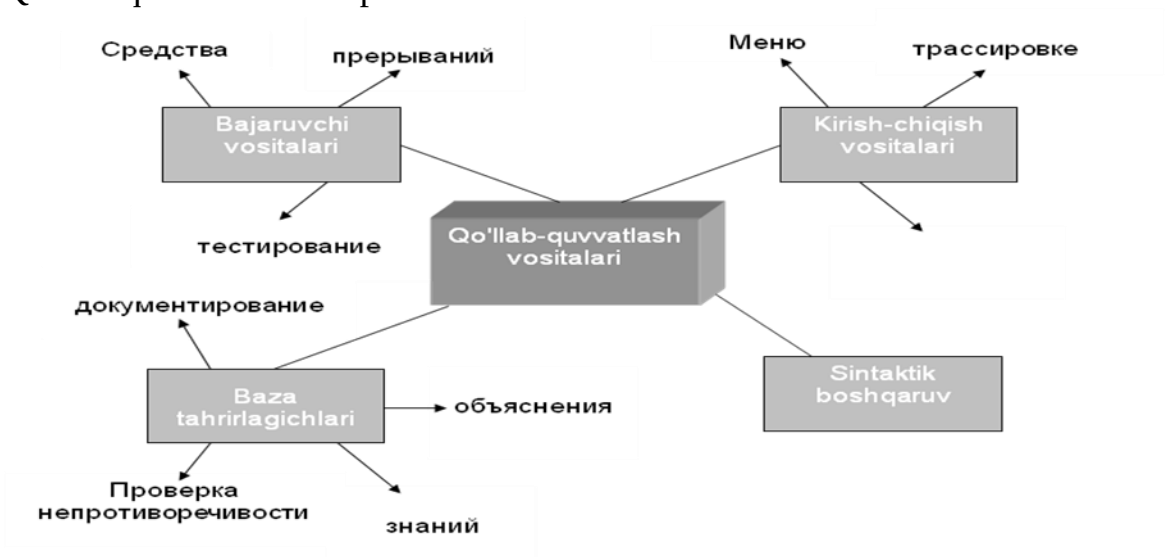
Tizim yangi qoidalarni ekspert bilan muloqot orqali sohadan egallaydi.

AGE tizimi ET qizmlarini shakllantirish uchun yig'iladigan uskunalar naboridan tashkil topgan.

Yordamchi vositalar shuningdek qo'llab- ET ni qurishda uni soddalashtiradigan quvvatlash vositalaridan ham tashkil topgan.

Keyingi sahifadagi rasmda qo'llab-quvvatlash vositalari keltirilgan:

Qo'llab-quvvatlash komponentalari vositalari



Tushintirish vositalari

Ba'zi tizimlar, ENYCIN ga o'xshagan - ichki tushintirish mexanizmlariga ega. Qolganlarida bunday fikr yuritish vositalari yo'q, lekin ulardan foydalanish zarur. Tushintirish mexanizmlarining bir necha turlari mavjud:

- Qadimiy taqsimlash;
- Gipotetik taqsimlash
- Qalbaki taqsimlash

Ishlab chiqish usullari

Ekstper tizimlarini ishlab chiqish jarayonlarida 3 ta usuldan foydalanish mumkin:

1. To'g'ridan-to'g'ri usulda universal dasturlash tillaridan foydalaniladi. Bu usulda ishlab chiilgan tizimlarga misol qilib, DENDRAL, MYCIN va b.larni ko'rsatish mumkin. Bunday tizimlarni yaratish uchun ko'p vaqt sarflanadi.
2. . Bilimlarni namoyish qilish tillari aosida yoki ET qurish tillarida. Bu usul oddiysidan farq qilib, yangi ET qurishga bir necha kun sarflaydi. Odatda ET uchun birinchi navbatda mahsulot modelidan foydalanadi.
3. Bilimlarni namoyish qilish foydalanish klassik usuli. Freym model katta egiluvchanlikka ega va bu bitta freymda ikkala deklarativ va prosedurali bilimlarni yig'ish imkoniga ega.

ET yaratish uchun uskunalar sifatida quidagilar xizmat qilishi mumkin:

- Belgili axborotlarni qayta ishlashga yo'naltirilgan Prosedurali dasturlash tillari(LISP, INTERLISP va b.)
- Istalgan ET yaratishga yo'naltirilgan tillar:(PROLOG, OPS-5, KRL, LOOPS, ПЛЭНЕР va b.);

ET jarayonlarini avtomatizatsiyalash vositalarini loyihalashtirish, foydalanish va modifikatsiyalash (RLL, HEARSAY-III, TEIRESIAS va b.);

Aniq sohadan tashkil topmagan bo'sh bazali ET(EMYCIN, KAS, GURU, ЭКСПЕРТ-МИКРО).

«Uskunalarni» tushuntirish dasturiy vositalarni o'ziga faqat biriktiribgina qolmay, apparat vositalarni ham o'z ichiga oladi. Ko'pgina

Uskunalar va ekspertlar deganda shuningdek ET yaratishda qatnashish ham tushuniladi.
Bular:
1. Soha eksperti;
2. Bilimlar bo'yicha injiner — ET qurish bo'yicha mutaxassis;
3. Dasturchi, uskuna vositalarini integratsiya qiluvchi va o'zgarishlarni amalga oshiruvchi;

Takrorlash uchun savollar:

1. Mashinali o’qitishning usullari?
2. Qo’lyozma matnlarni tanish usullari?
3. Mashinali o’qitishning vositalari?

12. Ma’ruza
Neyron to'rlari

Reja:

1. Neyron to’rlarini tashkil qilish to’g’risida ma’lumot
2. Neyron to’rlarining tarix
3. Neyron to’rlarini kompyuter dasturi sifatida namoyon bo’lishi

Tayanch iboralar: *to’r, daraxt, inson, faoliyat, neyron*

Odam miyasi juda murakkab tuzulishga ega. Uning qanday ishlashini o’rganish maqsadida juda ko’p ilmiy izlanishlar olib borilgan va borilmoqda. Ma’lumki inson miyasi katta xajmdagi axborotni tez qayta ishlay oladi. Bunga sabab millionlab miya nerv xujayralari - neyronlarning parallel ishlashidir [24,34].

Sun’iy neyronlarning g’oyaviy asosi xam biologik neyron xujayralari xisoblanadi. Bugungi kunda miyaning ishlashini o’rganish yo’lida fan erishgan yutuqlardan kelib chiqib biologik neyron quyidagicha ishlashini aytish mumkin. Nerv xujayrasi - neyron bo’lib, u ma’lumotlarni qayta ishlovchi eng kichik birlikdir. O’z o’rnida xar bir neyronda ko’plab o’simtalar bo’ladi. Bu o’simtalarning bittasidan boshqa barchalari akson deb nomlanadi va aksonlar orqali neyronga tashqi signallar keladi. Bitta o’simta dendrid deb nomlanadi va u orqali neyron tashqariga signal beradi. Ko’plab neyronlar bir birlari bilan ma’lum arxitekturada bog’langan bo’ladi. Bir neyronning aksoni boshqa bir neyronning dendridiga bog’langan nuqtalari sinaps deyiladi.

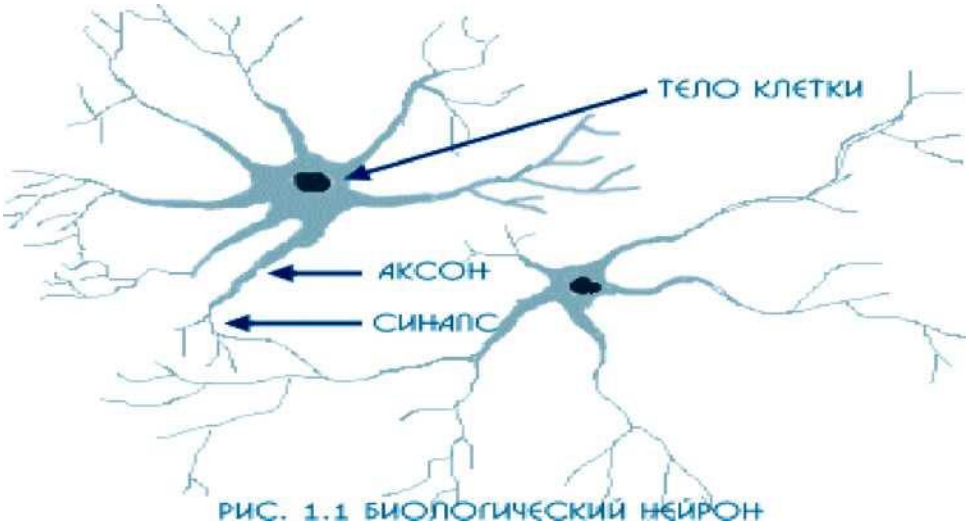


РИС. 1.1 БИОЛОГИЧЕСКИЙ НЕЙРОН

SHu tariqa millionlab neyronlar bir-birlari bilan bog'lanib ma'lum bir arxitekturadagi neyron to'rlarini tashkil qiladi. Bitta oldingi qatlamdagi neyron chiqish o'simtasi - dendrid orqali signalni keyingi qatlamdagi neyronlarga ularning aksonlari orqali beradi. Eng birinchi qatlamdagi neyronlar signallarni ma'lum organlarning retseptorlari orqali oladi. Masalan ko'z, burun, teri va xokazolar. Eng oxirgi qatlamdagi neyronlar esa signallarni ma'lum organlarning muskullariga uzatadi. Masalan qo'l, oyoq, yuz, tovush pardalari va xokazolar.

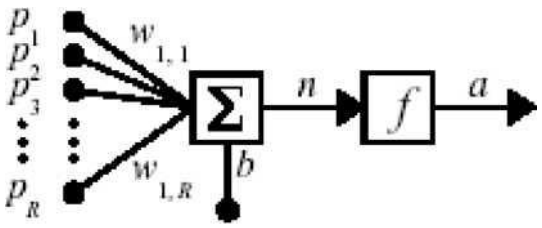
Ana shu kabi miya tuzulishini o'rganishlardan kelib chiqib biologik neyronlarning funktsional analogi sun'iy neyronlarni yaratishga xarakatlar qilinmoqda. Albatta, bugun erishilgan natijalar inson miyasiga nisbatan juda primitiv, lekin shilliqurt, chuvalchang miyasi darajasida deyish mumkin.

Sun'iy neyron tabiiy neyronning funktsiyasini bajara oladigan matematik modelb, apparat yoki kompyuter dasturidir. Bunda signallarning qiymati (ya'ni amplitudasi)gina xisobga olinadi. Tabiiy neyronda esa nafaqat signalning qiymati, balki chastotasi xam xal qiluvchi axamiyatga ega bo'lishi mumkin. Ammo organizmlar miyasini bugungi o'rganilganlik darajasi juda past bo'lib, xozirgacha bu borada ilmiy natijalarga erishilmagan.

Neyron deyilganda sun'iy neyron aniqrog'i, kompyuter dasturini nazarda tutiladi.

Oddiy neyronni ko'rib chiqaylik: кириш нейрон

г—**s***t*——————————**N**



ч ___; _____ _____ ***j***

a fbvp + h)

Bu erda: *p* - kirish vektori (input vector);

R- kirish elementlari soni (number of input elements);

w- og'irliklar vektori (weight vector);

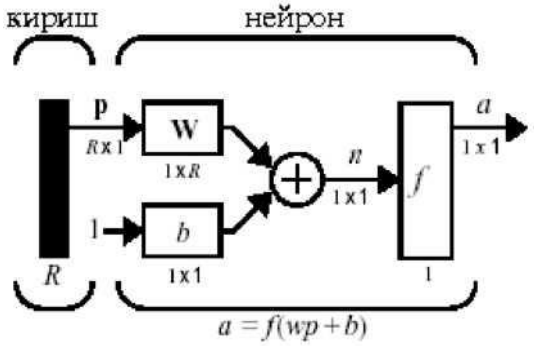
b- surilish (bias);

n- kirishning og'irliklarga ko'paytirilgan va surilgan qiymati (wpqb);

f- transfer funktsiya (transfer function); *a*- chiqish (output).

Neyronga kirish vetori p beriladi. Kirishlarning barchasi bir xil ta'sir kuchiga ega bo'lmaydi. SHuning uchun ma'lum kirishning ta'sir kuchini boshqarish maqsadida og'irlik w tushunchasi kiritilgan. Xar bir kirish qiymati p og'irliklar vetori w ning mos elementiga ko'paytirilib natijalar jamlanadi (ya'ni $wp+p_1w_{11}+p_2w_{12}$ $+...p_Rw_{1,R}$). Summaga surilish qiymati b qo'shiladi. b xam og'irlik w ga juda o'xshash, ammo uning «kirish» qiymati o'zgarmas 1 (bir) konstantadir (ya'ni b kirish qiymati emas). Natijada transfer funktsiyaning kirish qiymati n xosil bo'ladi (ya'ni *n+wp+b).* Bu qiymat transfer funktsiya (uzatish funktsiyasi)ga parametr sifatida berilib neyronning chiqishi a topiladi.

w va b neyronning sozlanadigan parametrlaridir. Ana shu parametrlar o'zgartirilib neyron ma'lum bir funktsiyani bajaradigan xolga keltiriladi. SHu jarayon neyronni o'rgatish deb yuritiladi. Neyron to'rlarning markaziy g'oyasi xam ana shunda: neyronlarning w va b qiymatlarini o'zgartirib, ya'ni o'rgatib ixtiyoriy vazifani bajaradigan xolga keltirish mumkin. Neyronni sxematik ravishda quyidagicha ifodalash mumkin:

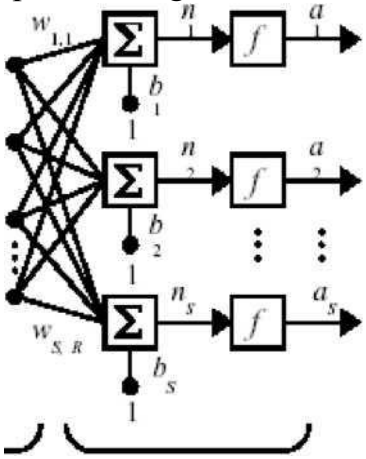


Neyron kirish qiymatlarini og'irliklarga ko'paytmasini jamlabgina qolmasdan ma'lum bir funktsiya - transfer funktsiyada xam qayta ishlaydi. Transfer funktsiya sifatida chiziqli, zinali, logarifmik-sigmoida, tangensoida funktsiyalaridan foydalaniladi. qanday funktsiyadan foydalanish aniq masalaga bog'liq.

Bitta neyronning funktsional quvvati juda past, lekin uning afzalliklaridan biri - ko'plab neyronlar birlashtirilib, quvvati oshirilib ishlatilishi mumkin.

quyida S dona neyrondan tashkil topgan 1 qatlam(layer)li neyron to'r keltirilgan: кириш нейронлар 1 катлами

Tushunish osonroq bo’lishi uchun yuqoridagi detalbniy sxemani quyidagi soddaroq ko’rinishga keltirish mumkin:



Г—\ Г ----------

p2 **I.**

R - kirish elementlari soni;

1. "! *2* ■
a. *2* ■
1 ■ *iv⊥R*
S. *lvS* *'2* ■
1 *U/C R*

S - birinchi qatlamdagi neyronlar soni;

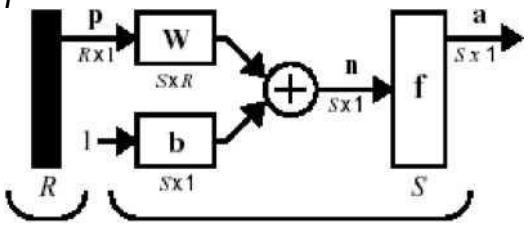
Og’irliklar vektori W matritsasining qatorlari neyronlarning indeksini, ustunlari esa indekslarini ifodalaydi, ya’ni:

kirish a f(Wp bi

w₁i - birinchi neyronning birinchi kirishga og’irligi; ***w₁,₂*** - birinchi neyronning ikkinchi kirishga og’irligi; w_{21} - ikkinchi neyronning birinchi kirishga og’irligi;

кириш нейронларнинг бир катлами

f---------- **ч**

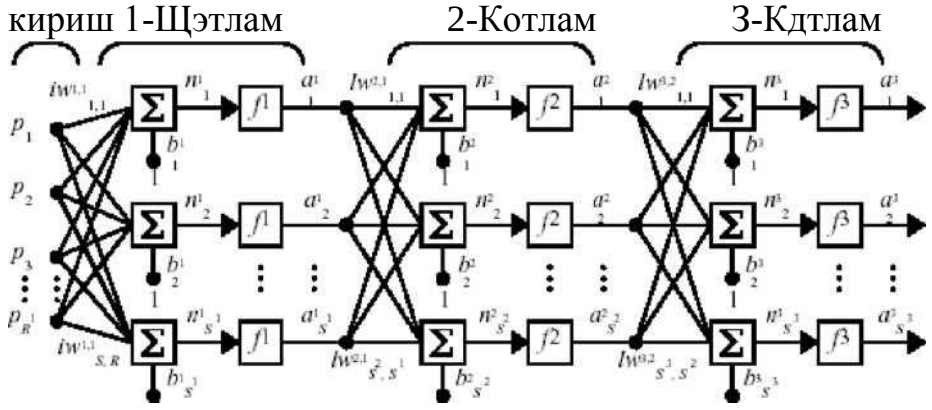


л = f (Wp bi

Neyronlarning bunday tarzda qatlamga biriktirilishi kirish signallarini barcha neyronlarga uzatilishi, neyronlar xar biri o’zi mustaqil ishlashi va xar bir neyronning chiqishini aloxida-aloxida olish imkononi beradi. Bundan tashqari ko’plab sondagi neyronlarni bitta setga birlashtirganda qo’yilagan masalani echish uchun yaroqli arxitekturani xosil qilish mumkin bo’ladi.

Odatda uchraydigan masalalarni echish uchun bir emas ko’p qatlamli neyron to’rlar talab qilinadi. Ko’p qatlamli neyron to’rlarda birinchi qatlam kirish qatlami (input layer), oxirgi qatlam chiqish qatlami (xutput layer) va boshqa barcha ichki qatlamlar berkitilgan qatlamlar (hidden layers) deb nomlanadi.

Quyida ko’p qatlamli neyron to’rga misol tariqasida 3 qatlamli neyron to’r keltirilgan:



I I I

4 *J* _____ *J* \ _____ *J* \ _____ *J*

ai = fl(IWi.ip+bi) я: = P(LW:.iai+b2) |!=П(Ь\¥з:а!4|»)

a; P (LWif^2 (LWnfl (IW'i ip +bi)+b2)‘ b-i|

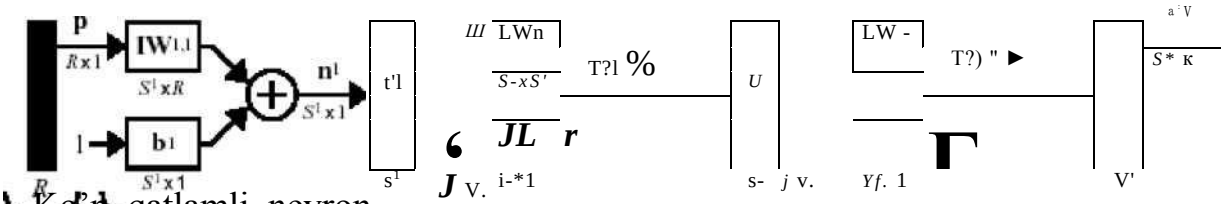
Birinchi qatlamdagi neyronlarning og’irlik matritsasi IW (Input Weights) sifatida belgilangan. Keyingi barcha qatlamlarda esa LW (Layer Weights) tarzida belgilangan.

Sxemadan ko’rish mumkin birinchi qatlamning chiqishi a^1 ikkinchi qatlamga kirish sifatida berilmoqda va mos ravishda ikkinchi qatlamning chiqishi a uchinchi qatlamning kirishiga berilmoqda. Butun setning chiqishi - oxirgi qatlamning chiqishi a^3dir.

Bu sxemani soddalashtirilgan xolda quyidagicha ifodalash mumkin:

кири 1- 2- 3 -

ш кдтлам Л г кдтлам Л г кдтлам Л



Ko’p qatlamli neyron murakkab funksiyalarni approksimatsiyalay oladi. Xususan birinchi qatlami sigmoida va ikkinchi qatlami chiqiqli transfer funktsiya bo’lgan ikki qatlamli neyron to’r ixtiyoriy funktsiyani approksimatsiyalay oladi. Albatta, buning uchun approksimatsiyalanishi kerak bo’lgan funktsiyaning murakkabligiga xarab xar ikkala qatlamdagi neyronlar soni etarli bo’lishi va ko’p, lekin chekli sondagi o’rgatish amalga oshirilishi kerak.

Neyron to’rlarda quyidagi transfer funktsiyalar ishlatiladi:

Zinali transfer funktsiya barcha transfer funktsiyalar ichida eng funktsional kuchsizi, ammo birinchi neyron to’r(perseptron)da aynan mana shu funktsiyadan foydalanilgan. CHiziqli transfer funktsiyaning boshqa transfer funktsiyalardan afzalligi - chiqish doirasi katta, ammo shu bilan birga eng katta kamchiligi ixtiyoriy ko’p qatlamli chiziqli neyron to’rni bir qatlamli chiziqli neyron to’r bilan almashtirish mumkin. YA’ni faqat chiziqli transfer funktsiyalardan foydalanib neyronlarni ko’p qatlamlarga biriktirish ularning funktsional quvvatini oshirmaydi. CHiziqli transfer

funktsiyaning aksini sigmoida transfer funktsiyasida ko'rishiiz mumkin. Sigmoida transfer funktsiyali neyronning chiqishi kirishiga mos ravishda 0 va 1 oralig'ida joylashadi. SHuning uchun xam bunday funktsiyalarni siquvchi funktsiyalar deb xam yuritiladi. Sigmoida transfer funktsiyali neyronlarni ko'p qatlamlarga biriktirish ularning funktsional quvvatini juda oshiradi.

Neyron to'rlarining tarixi

Neyron to'rlarning nazariy asoslari dastlab 1943 yilda U.Makkalox va uning shogirdi U.Pitts olib borgan tadqiqotlarga borib taqaladi. Neyron tushinchasi va og'irlik tushinchasi shu olimlarning ilmiy izlanishlaridan qolgan.

Makkalox modelining asosiy kamchiligi transfer funktsiya (o'tish funktsiyasi) sifatida faqatgina zinali funktsiyadan foydalanilgan. Bu xam aslida Makkaloxning ilmiy qarashlaridan biri edi. Olim transfer funktsiya faqat ikki xolatdagina bo'la olishi kerakligini, neyron xam kirish signallariga qarab ikki xolatning birida - ishlagan yoki ishlamagan xolda o'z natijasini setning keyingi neyronlariga uzatishi lozimligini aytgan.

Ammo keyingi tadqiqotchilarning ilmiy izlanishlari natijasida shu narsa ma'lum bo'ldiki, transfer funktsiya sifatida faqatgina zinali funktsiya emas, balki boshqa funktsiyalardan, masalan chiziqli, logarifmik-sigmoida, tangens-sigmoida kabi funktsiyalardan foydalanish xam yaxshi natijalar beradi (qaysi transfer funktsiyadan foydalanish aniq xolatlarga, muammolarga bog'liq).

Makkaloxning ishlarida ba'zi kamchiliklarga xam yo'l qo'yilgan bo'lishiga qaramasdan neyron to'rlarning nazariyasi negizi xali xam o'shandayligicha qolmoqda.

Neyron to'rlarning rivojlanishiga bo'lgan katta turtkilardan biri neyrofiziolog olim F.Rozenblat taklif qilgan modelb - perseptron bo'ldi. Perseptronning Makkalox modelidan farqi neyronlar orasidagi aloqalardagi og'irliklarning o'zgaruvchanligi edi. O'zgaruvchanlik imkoniyatining mavjudligi neyron to'rlarni turli mua.mmo1a.rni echishga «o'rgana oladigan» qildi.

Keyinchalik Xopfild, Verbos, Koxonen, Fukushima kabi olimlar neyron to'rlar ustida ilmiy izlanishlar olib bordilar va katta natijalarga erishdilar.

Neyron to'rlarni o'rganish natijasida ularning bir qancha xususiyatlari ma'lum bo'ldi. Neyron to'rlardan prognozlashda, jarayonlarni boshqarishda, immitatsiya qilish va taxlil qilishda foydalanish yuqori samara beradi. Neyron to'rlarni boshqa usullarni tadbiq qilish qiyin bo'lgan sharoitlarda - muammoni xal qilish algoritmi mavxum bo'lganda, ma'lumotlar noaniqligida, etishmasligida, juda katta yoki juda kichik xajmdaligida, qarama-qarshiliklar mavjud sharoitlarda tadbiq qilish oson va samarali.

Bunga asosiy sabab boshqa usullardagi kabi kerakli jarayonni qonuniyatlarini aniqlab, matematik tenglamalar tuzib, echish algoritmlari ishlab chiqishning zaruriyati yo'q. Neyron to'rlar arxitekturasi, transfer funktsiyalar va o'rgatish algoritmlari to'g'ri tanlansa neyron to'rni tayyor ma'lumotlarda o'rgatish natijasida, u foydalanishga tayyor bo'ladi.

Neyron to'rlarni o'rgatish deyilganda neyron to'rning o'zi o'z ichki parametrlarini xisoblab topib o'zgartirishi tushiniladi. Buning uchun tarmoqqa tanlangan kirish qiymatlari beriladi va xosil bo'lgan natijalarni xaqiqiy natijalar bilan solishtirib

farqi(xatolik) topiladi. SHu farq neyron to’r uchun parametrlarini to’g’rilashiga asos va ma’lumot bo’ladi.

Neyron to’rlarini ishlab chiqarishning turli sohalariga tadbiqi

Bugun neyron to’rlar o’ta chuqur o’rganilmagan bo’lishiga qaramasdan quyidagi sohalarga qo’llanilib ijobiy natijalarga erishilmoqda:

- biznes - neyron to’rlarning bu sohaga tadbiqi 1984 yilda adaptiv kanal ekvalayzeri yaratilishi bilan boshlandi. Bu qurilma juda sodda bo’lib, bitta neyrondan tashkil topgan. U uzoq masofadagi telefon liniyalarida ovozni stabillashtirib sifatini oshirganligi sababli katta iqtisodiy muvafaqiyat qozongan;
- bank moliya - ko’chmas mulkni baxolashda, kredit berishda risklarni xisoblab mijoz tanlashda, qarzlarni baxolashda, kreditlarning ishlatilishini analiz qilishda, savdo portfeli programmalarida, moliyaviy analiz qilishda, valyuta qiymatini prognozlashda;
- birja - valyuta va aktsiya kurslarini prognozlashda, bozorni prognozlashda, korxonalar kelajagini baxolashda;
- ishlab chiqarish - jarayonlarni boshqarishda, maxsulotlar dizayni va analizida;
- meditsina - o’pka raki xujayralarini analiz qilishda, DNK analizida, protez loyilashda, transplantatsiya vaqtlarini optimizatsiyalashda, shifoxona xarajatlarini kamaytirishda va sifatini oshirishda, shoshilinch yordam xonalarini tekshirishda;
- robototexnika - traektoriya qurishda, xarakatni boshqarishda, manipulyatorlarni boshqarishda, tasvir analizi va ko’rishda, shakllar va figuralarni tanishda, ovoz analizi va sintezida;
- transport - marshrutlarni optimal loyixalashda, vaqt jadvallarini rejalashtirishda, yuk mashinalari tormoz sistemalarining analizida;
- avtomobilb - avtomatik boshqarish tizimlarida, avtomatik xarita tizimlarida, kafolat bilan bog’liq ishlar tekshiruvida;
- kosmos - yuqori samarali avtopilotlar yaratishda, uchish traektoriyasi immitatsiyasi tizimlarida, uchar jismlarni boshqarish tizimlarida, uchar jismlarining kamchilik va buzuqliklarini topish va bartaraf qilishda;
- mudofaa - tovush, radar, infraqizil signallarni taxlil qilishda, axborotlarni umumlashtirishda, avtomatik qurilmalarni boshqarishda;
- telekommunikatsiya - tasvir va ovozni zichlash, shifrlash va boshqacha qayta ishlash jarayonlarida, avtomatlashtirilgan axboratlashtirishda, turli tillarga sinxron tarjima tizimlarida va xokazolarda.

Neyron to’rlarning afzalliklarini va mavjud kompyuter dastur paketlarining qulaylik va samaradorligini xisobga olib uni innovatsiya jarayonlarida qo’llash istiqbolli ekanligini xulosa qilish qiyin emas

Neyron to’rlarini kompyuter dasturi sifatida namoyon bo’lishi

Neyron to’rlarni loyixalash va yaratish borasida ko’plab kompyuter dasturlari ishlab chiqarilgan. Ular orasida MathWorks firmasi tomonidan yaratilgan MatLab kompyuter dasturi paketi ustunliklari bilan aloxida ajralib turadi. CHunki aynan shu dastur matematik yadroga va neyron to’rlar qism paketiga ega. Unda eng sodda

neyron modelidan tortib, ixtiyoriy transfer funktsiyali ixtiyoriy arxitekturadagi murakkab neyron to'rlarni oson va tez yaratish mumkin.

Bundan tashqari paket tarkibiga teskari aloqali chiziqli boshqaruvchi, zavod kelajagini prognozlovchi va baxolovchi, funktsiyalarni approksimatsiyalovchi vositalar xam kiradi. Neyron to'rlarni o'rgatishning bir qancha algoritmlari xam paketda amalga oshirilgan.

MatLab dasturida neyron to'r modeli tuzilgach bu modelb ustida virtual laboratoriya sifatida foydalanib, jarayonni immitatsiya qilish mumkin.

MATLAB dasturi matritsaviy amallarni qo'llashga asoslangan. Bu tizimni nomi MATrix LABoratory matritsaviy laboratoriyada o'z aksini topgan. MATLAB - kengayuvchi tizim, uni xar xil turdagi masalalarni echishga oson moslashtirish mumkin.

Simulink -dinamik tizimlarni modellashtirish, imitatsiya va taxlil qilish uchun interaktiv vositadir. U grafik blok-diagrammalarni qurish, dinamik tizimlarning ishlashini tekshirish va loyixalarni mukammalashtirish imkoniyatlarini beradi. Simulink yuzdan ortiq biriktirilgan bloklarga ega. Bloklar vazifalariga mos xolda guruxlarga bo'lib chiqilgan. Bular: signallar manbalari, qabul qilgichlar, diskret, uzluksiz, chiziqli bo'lmagan, matematik funktsiyalar, signallar va tizimlar. Simulink MATLAB bilan to'la integrallashgan.

Takrorlash uchun sayollar:

1. Neyron to'rlarini kompyuter dasturi sifatida namoyon bo'lishi nimaga bo'gliq?
2. Sun'iy neyronlarning g'oyaviy asosida nimalar yotadi?
3. Neyron to'rlarini ishlab chiqarishning turli sohalariga tadbiqini tushintiring?

13. Maruza.

Tabiiy tilga ishlov berish.

Reja:

1. Tilga ishlov berish mexanizmi.
2. Leksik taxlil.
3. Sintaktik taxlil.
4. Semantik taxlil.

Tayanch iboralar: *til, tahlil, semantik*

Inson va kompyuter mulokoti bu kupgina tadkikotchilar ish olib borayotgan masaladir. Bu ishlardan yakuniy maksad foydalanuvchi va kompyuter uzaro tabiy tilda suxbat kura olishidir misol uchun rus tilida yoki kompyuter ularga shu tilda javobbera olishi.

Tashki kuranishdan bu vazifa engil tuyulishi mumkin, buning sababi biz yoshligimizdan inson mulokotini eshitib kelganligimizda. Kompyuterlarning akli ularni ishlab chikkan insonlarning maxorati bilan ulchanadi, shuning sababidan ular uzguzidan fikrlashga kodir bulmaganliklari uchun ularga uta anik yullanmalarni berish orkali nimani kilish kerakligini tushuntirish mumkin. Inson

tugilganidanok tilni urganishga moyillik bilan tugiladi, lekin kompyuter inson tilini tushunishi uchun tilni avvalo asosiy elementlarga bulish va shu axborotlarni kompyuterga u tushunadigan tarzda kiritish zarur. Inson va kompyuter mulokoti tushunarli bulishi uchun tabiy tilni kayta ishlash tizimini ishlab chikish zarur.

Keling, bu vazifa kanchalik mushkul ekanligini kuramiz. Tasavvur kiling sun'iy tafakkurga ega bulgan robot avtoulovlarni tamirlay oladi.

Unga kuyidagi ikki topshiriklarni berish mumkin.

1. Gildiragi teshilgan uy yonidagi avtoulovni tamirla.
2. Uy yonidagi kizil pardali avtoulovni tamirla.

Birinchi jumlani ikki xil izoxlash mumkinligiga karamay xar bir inson uyning yonida tushirilgan gildirak bulmasligini tushunadi. Inson bu jumladagi noaniklikni darxol sezadi va undan xam muximrogi ongida bu notugri jumlani tugrilaydi chunki malumki teshilgan gildirak avtoulovda uy yonina emas. Robot suzlarni boglashdan va ularni ma'nosini tushunishdan kuprok kila olishi kerak aks xolda u teshilgan gildirakni kidirishiga tugri kelar edi. Ikkala jumla xam bir xil tuzilishga ega bulganligi uchun robot grammatikani va obektlarni ularning manosini takkoslay olishi kerak. Inson tilining koidalari fakatgina inson uchun manogo ega, kompyuter uchun esa gap manosini anglash uchun maxsus koidalar darkor.

Bizning robotimiz ega bulgan sunniy intellekt jumlalar va ularning orasidagi boglikdikni taxlil kila olishi kerak. Misol uchun kuyudagi ikki gapni olamiz.

1. Sarvar sut ichmokda.
2. Sungra u palto kiymokda.

Ikkinchi gapdagi u suzi birinchi gapdagi Savarga taalukli. Birinchi jumlasiz ikkinchi manosiz bular edi. Barcha tabiy tillar kantekstual tillardir. Boshkacha kilib etganda ikkinchi jumlani tushinish uchun birinchi jumlani bilish shart. Birgina jumla orkali izoxlash mumkin bulgan tillar kontekstual mustakil deiladi. Kompyuter insonni tushina olishi uchun tabiiy til taxlilatorini ishlab chikish zarur. Taxlilning asosiy funksiyalari kuyidagicha:

1. Leksik taxlil(suzlar taxlili).
2. Sintaktik taxlil (grammatik koidalar asosida suzlar taxlili).
3. Semantiktaxlil.

Leksik taxlil

Leksik taxlil jumla suzlarning tovush yoki tuxtash belgilari asosida bulish. Bundan tashkari jumlada uzakni va kushimchalarni ajratib olish mumkin. Misol uchun kushimcha suz kuyidagicha bulish mumkin:

Kushimcha (suz) kushish (uzak) cha (kushimcha)

Suzlarni lugatdan olish mumkin lekin ularning umumiy manosini kompyuterga tushintirish kiyin masala.

Sintaktik taxlil

Inson tilini kompbter tushinishi uchun avvalam bor komyuterni suzlarni ajrata olishni urgatish kerak. G rammatika va sintaksiz koidalarini kopmyuter tushunadigan shaklga keltirish kerak.

Odatda jumla (J) otlar gurixi (OG) va fellar gurixi (FG)dan tashkil topgan buladi va ularni kuyidagi kurinishda buladi:

JOG,FG

Ot gurixi kuyidagicha bulinishi mumkin: (atokli ot, olmosh va xkz).

OG->AO.

Grafik tarzda jumlaning sintaksik kurinishi “daraxt” shaklida bulishi mumkin. Misol uchun: “kari utinchi daraxt chopmokda” jumlasi 1-rasm. da kursatilgandek tuzilishga ega. jumla suzlarga bulinadi suzlar esa siniflarga bulinadi. Kari suzi - aniklovchi (A), sifat orkali ifodalangan, utinchi- suzi - ot (O), chopmokda - fe’l (F) va daraxt -ot (O).

Semantik taxlil

Suzni tarkibiy kismlarga bulgandan sung kompyuter uning semantik taxlil kilmokda yani uning manosini tushinmokchi. Sunniy akl tizimida jumlani manosini anglash uchun koidalar umumiyligi ishlatiladi.

AO F O

Kari utinchi daraxt chopmokda

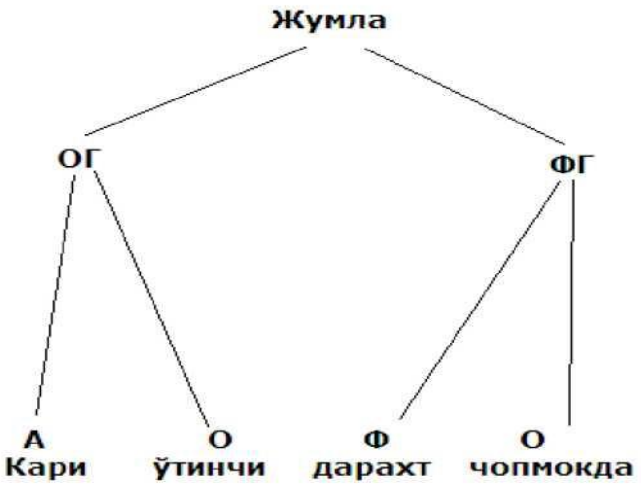
Jumlani izoxlash uchun simantik taxlilchining bilimlar omborida kuyidagi koidalar mavjud bulishi lozim.

1- koida: AGAR aniklovchi birinchi bulib kelsa va undan keyin ot kelsa U XOLDA ot egadir.

2- koida: AGAR egadan keyin fe’l kelsa u xolda fe’l sifatdir va kesimdir.

3- koida: AGAR egadan sung kesim kelsa va undan sung ot u xolda ot tuldiruvchidir.

4- koida: AG AR jumla kuyidagicha ketma-ketlikda bulsa: ega, fe’l, tuldiruvchi u xolda butin jumla egasi tuldiruvchiga nisbatan kesimdir.



5-

6- Rasm.1. Jumlaning sintaksik daraxti.

Rasm 2. Tabiy til protsessori

YUkorida aytilganni misolda tushintiramiz. Faraz kilaylik sunniy akl tizimi kuyidagi masalani echishi kerak: kari utinchi nima kilayotganini va uning faoliyat ob'ektini aniklash. Semantik taxlilchi birinchi koidaga murojat kiladi, uning yordamida u "utinchi" suzi ega ekanligini aniklaydi. 2-koida yordamida "chopmok" kesimligini. Xarakat ob'ekti 3,4-koidalar orkali "darax" suzi ekanligi .

Kuyidagi misol tabiy til protsessorining semantik , leksik va sintaktik koidalar orkali jumlani kanday kilib tushinishini kursatadi. Insonga kampyuter bilan ogzaki mulokat uchun tabiy til protsessori foydalanuvchi va sunniy akl tizimi orasidagi boglovchi zanjir bula oladi. Umuman olganda tabiy tilni kayta ishlash foydalanuvchidan kiyin dasturlash tillarini urganishdan ozod etadi. Agar kompyuter va inson tabiy tilda suzlashishini vujudga keltira oladigan dastur ishlab chikilsa bu xakikiy sun'iy kompyuter buladi.

Takrorlash uchun savollar:

1. Tilga ishlov berishning qanday mexanizmi mavjud?
2. Leksik taxlil mohoyati?
3. Sintaktik taxlilbilan Semantik taxlil ning hususiyatlari?

14 . Ma'ruza

Xulosa. Intelektual tizim va ekcpert tizimlarning istiqbollari.

Reja:

1. IT ni tanlashda hisoblab chiqiladigan faktorlar.
2. IT yaratish qiyinchiligi.
3. IT rivojlanish istiqbollari.
4. ET effektivlik kriteriyalari

Tayanch iboralar: *intellect, ekspert, effect, factor.*

Instrumental vositalarni tanlashda quidagi 6 ta asosiy faktorlarni hisoblab chiqish zarur:

- Joriy vositani taqdim qilingan imkoniyatlari bo'yicha talablarini qondirish;

- Berilgan muddat uchun taqsimlashda dasturchi uchun qo’llab-quvvatlash yetarli bo’lish.
- Xar hil sohalarda to’liq bo’lmagan testlash, eskirgan hujjatlar ma’nosida, tanlagan vositamiz ishonchlimi?
- Instrumental vositalarni qo’llab-quvvatlaydimi?
- Odatda instrumental vositalar foydalanish bo’yicha maslahat beruvchi ekspert-konsultant tomonidan birgalikda kuzatiladi. Shuning uchun shunday o’zini qo’llab-quvvatlamaydigan vositani tanlash kerak.
- Tanlangan instrumental vosita o’ziga hos qo’yilgan masalalarni harakterlaydimi?
- Vosita ilovaning hususiyatiga mos kelishini harakterlaydimi? Ilovalarda mashg’ulot va bashorat qilish imkoniyati bo’lishi.
- Boshqa xususiyati – tizim foydalanuvchi bilan ishlaydimi yoki mustaqil?
- Tizim foydalanuvchi doirasi uchun mo’ljallanganmi yoki keng doiradagi foydalanuvchilargami?

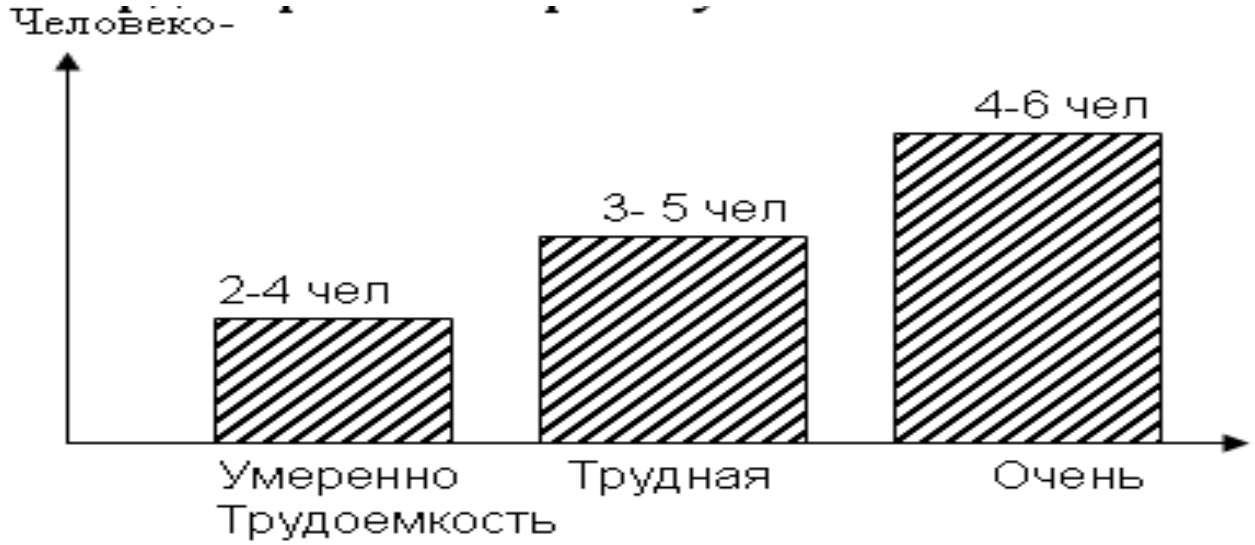
ET qurishdagi qiyinchiliklar

- Ekspert tizim qurishdagi qiyinchliklar ET qurish uchun zarur bo’lgan instrumental vositalarning va ET loyihalashda injinerlarni yetishmasligidan tashkil topadi. Mutaxassis yetishmasligini 2 ta sababi bor:
- II – yangi fan va shuning ucun ko’pgina VT va dasturlash bo’yicha mutaxassislar ET loyihalash masalalari bilan tanish emas.
- ET maydoniga kiruvchi kompaniyalar soni II bo’yicha VUZ larni bitirich chiqayotgan mutaxassislar sonidan ko’p. Bu borada, firmalar tomonidan injinerlar tayyorlash bo’yicha qisqa fursatli kurslar tashkil qilinadi.

ET qurish instrumental vositalarini yetishmasligi quidagi faktorlar bilan bog’liq:

1. IT larning bilim olishga qobiliyatsizligi. Bu ET qurishda ko’p vaqt yo’qotishga olib keladi.
2. IT ni loyihaga ishlov berish va bilimlar bazasini to’g’irlash bo’yicha imkoniyati yetmasligi. Asosan, ma’lumotlarning to’liqligi va doimiyligi muhim hisoblanmaydi. Bugungi kunda hech qanday universal metodlar mavjud emas. Injinerlar qarama-qarshilikdan qutulishi uchun bitta emas, ko’rpoq jalb qilinishi kerak.
3. Instrumental vositalar turli ma’lumotlar tizimi bilan yaxshi ishlashga qodir emas. Masalan, bir xil tilda freymli va mahsulotliga erisihish qiyin.
4. Bundan tashqari intellektual tizimlar ET ishlab chiqish tillari cheklangani sababli, tabiiy tilda erkin suhbatlashishga imkoniyat bermaydi.

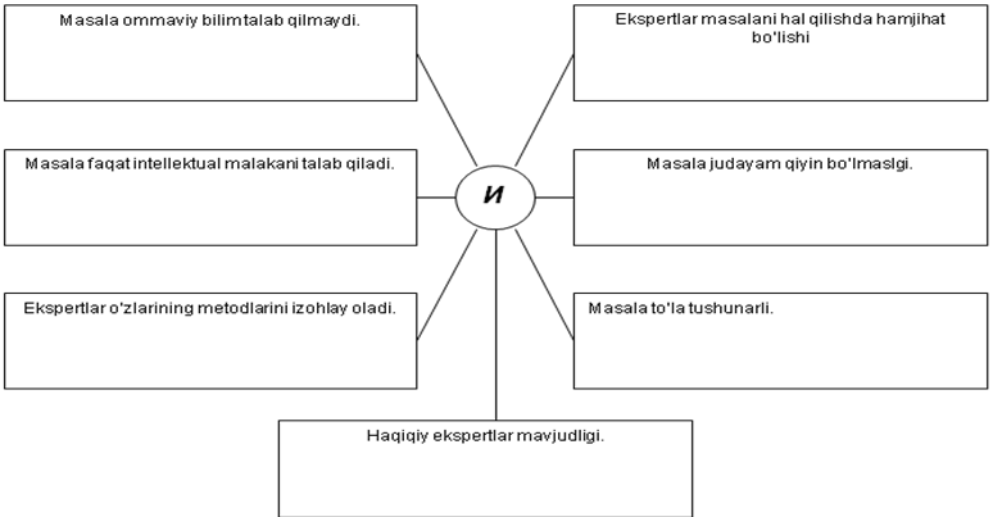
Ekspert tizimlar qurish murakkablik diagrammasi



Qiyin bo'lmagan muammolar doirasi uchun tipik ET loyihasi quidagi xarakteristikalarga ega

Трудоемкость	6 человеко-лет	Занятость
Время	2 года	
Штат	Ст. инженер	0.25
	Мл. инженер	1.0
	Сист.прграммист	1.0
	Эксперт	0.75
	Итого	3.0 специалиста с полным раб. днем

ET yaratish imkoniyati uchun talablar

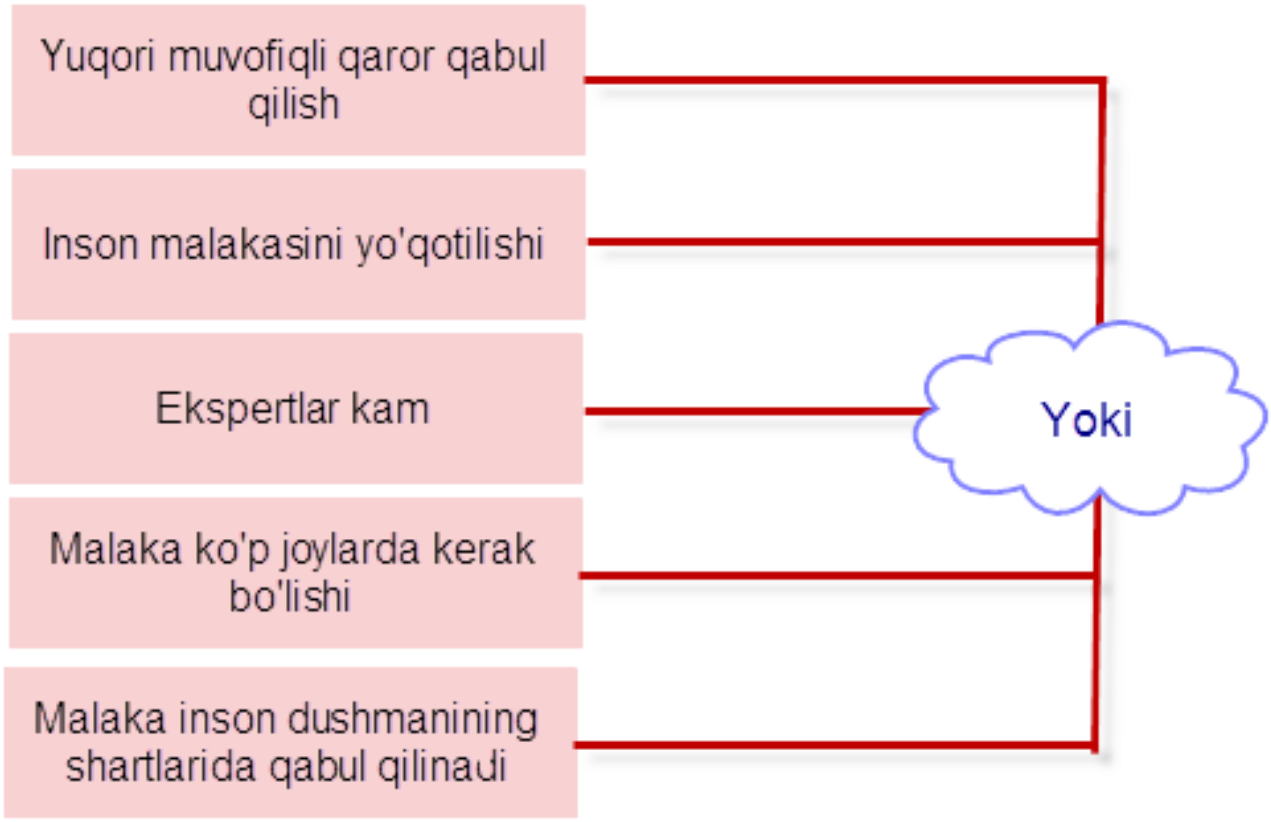


ET effektivligi kriteriyalari
ET effektivligi kriteriyalari.

Taqsimlanadi:

- ET yaratish imkoniyatlari uchun talab.
- Qachonki ET yaratilishi haqli deb topilsa, talab qo’yish
- Qachonki, ET ishlab chiqish mufoviq deb topilsa, talab qo’yiladi.

ET ishlab chiqilishi haqli deb topilishi uchun talablar.



Takrorlash uchiu savollar:

1. IT ni hususiyatli jihatlari va faktorlari?
2. ITlar randay yaratiladi?
3. IT rivojlanish istiqbollari haqida nima bilasiz?

AMALIY MASHG'ULOTLAR VA LABORATORIYA MASHG'ULOTLARI MATERIALLARI

1-Amaliy ish.

Mavzu: Optimallashtirish va nisbiylik

Ishdan maqsad: Masalaning yechimini topishning optimal usulini ishlab chiqishni o'rganish. Massivning eng katta yoki eng kichik elementini topishning optimal usullarini aniqlash.

Uslubiy ko'rsatmalar: Optimizatsiya – bu matematika, informatika va ilmiy izlanishlar masalalarida cheklangan chiziqli yoki chiziqli bo'lmagan tenglama yoki tengsizliklarning chekli vector fazosidagi bir qancha sohalardagi butun funksiyaning ekstremumini topishdir. Optimizaytsiya masalalarini yechish nazariyasini matematik dasturlash o'rganadi.

Loyihalash jarayonida odatda obyekt parametrlari strukturasi va qiyamatlariga qarab eng yaxshi usul aniqlab olinadi. Agar optimizatsiya obyrekt parametrining optimal qiymati bilan bog'lngan bo'lsa, unda bu optimizatsiya parametric optimizatsiya deyiladi. Standart matematik masala quyidagi ko'rinishda shakllantiriladi. f(χ) funksiyaning ko'p sondagi X lar orasida χ elementlar orasida f(χ^*) minimal qiymatini beruvchi χ^* elementni toppish kerak. Masalani yechishda to'g'ri yechimni berish uchun quyidagilarni yechish kerak bo'ladi:

1. Ko'pincha yo'l qo'yilgan X lar

$$\mathbf{X = \{\vec{x} \mid g_i(\vec{x}) \le 0,\ i = 1, \ldots, m\}}$$

2. Butun funksiyani ko'rsatish

$$f : X \to R$$

3. Qidiruv kriteriyasi(max yoki min).

Shunda f(x) → $\frac{min}{\vec{x} \in X}$ masalasini yechilish natijasida quyidagilardan biri hosil bo'ladi:

a) X = ∅.

b) f($\vec{x}$) butun funksiya quyidan chegaralanmagan.

c) $\vec{x}^* \in \mathbf{X}$ **: f(**$\vec{x}^*$**) =** $\frac{min}{\vec{x} \in X}$ f($\vec{x}$).

d) Agar $\nexists \vec{x}^*$ bo'lsa, $\frac{inf}{\vec{x} \in X}$ f($\vec{x}$) topilsin.

Endi optimizatsiyani dasturlash tomonidan ko'rib chiqamiz. Optimizatsiya – bu dasturning effektivligini yaxshilash maqsadida tizimni modifikatsiyalanishi. Bunda tizim bitta kompyuter dasturi, raqamli qurilma, kompyuterlar to'plami yoki hatto internetga o'xshash butun tarmoq bo'lishi mumkin. Hech bo'lmaganda to'liq optimizatsiyalash optimal tizimni olish hisoblanadi. To'g'risini aytadigan bo'lsak opitimizatsiyalash jarayonidagi optimal tizim har doim ham yetarli bo'lib hisoblanmaydi. Optimallangan tizim odatda faqat bitta masala yoki foydalanuvchilar guruhi uchun tegishli bo'ladi. Bu masala yoki foydalanuvchilar guruhidagi optimallashtirishga masalani yechishda, xotira hajmidan yo'qotish

evaziga bo’lsa ham vaqt sarfini kamaytirish; xotira muhim bo’lgan ilovalarda esa xotiraga kam murojat qiluvchi algoritmlardan foydalanish orqali masalani hal qilish kabilar kirishi mumkin.

Ba’zi masalalar ko’pincha samarali bajariladi. Masalan, C tilidagi 1 dan N gacha bo’lgan barcha butun sonlar yig’indisini topuvchi dasturni olaylik:

```
int i, sum = 0;
For (i  = 1; i <= N; i++)
sum+=i;
```

Ko’rinib turganiday bu kodda unchalik ham murakkablik yo’q. Bu kodga quyidagicha o’zgartirish kiritamiz:

```
Int sum = (N*(N+1))/2
```

“Optimizatsiya” odatda anglanilganidek tizimning o’zida funksionallikni saqlashidir. Biroq unumdorlikni sezilarli yaxshilash ortiqcha funksionalliklarni olib tashlash orqali ham amalga oshirish mumkin. Masalan, dastur 100tadan ko’p element kiritishga talab qilmasa dinamik xotiradan emas static xotiradan foydalanish amalga oshiriladi.

Protsessor vaqtini sarflash bo’yicha optimizatsiya odatda matematik hisblashga ega katta solishtirma og’irlikka hisoblovchi protsessor uchun juda muhim. Bu yerda dasturchi dastur kodini yozish paytida bir qancha optimizatsiyalash usullari keltirilgan.

- Ma'lumotlar obyekti initsializatsiyasi. Qaysidir ma'lumotlar obyekti qismining ko'pgina dasturlarida initsializatsiyalash(boshlang'ich qiymatni o'zlashtirish) zarurati mavjud. Bunday boshlang'ich qiymatlarni o'zlashtirish yoki dastur boshida yoki sikl oxirida amalga oshiriladi. Boshlang'ich qiymat berish protsessorning qimmatli vaqtini tejaydi. Masalan, agar gap sikldan foydalanib massivni initsializatsiyalash haqida ketayotgan bo'lsa, bu massivni to'g'ridan to'g'ri o'zlashtirishni e'lon qilishdan ko'ra kamroq samarali bo'ladi.

- Arifmetik operatsiyalarni dasturlash.

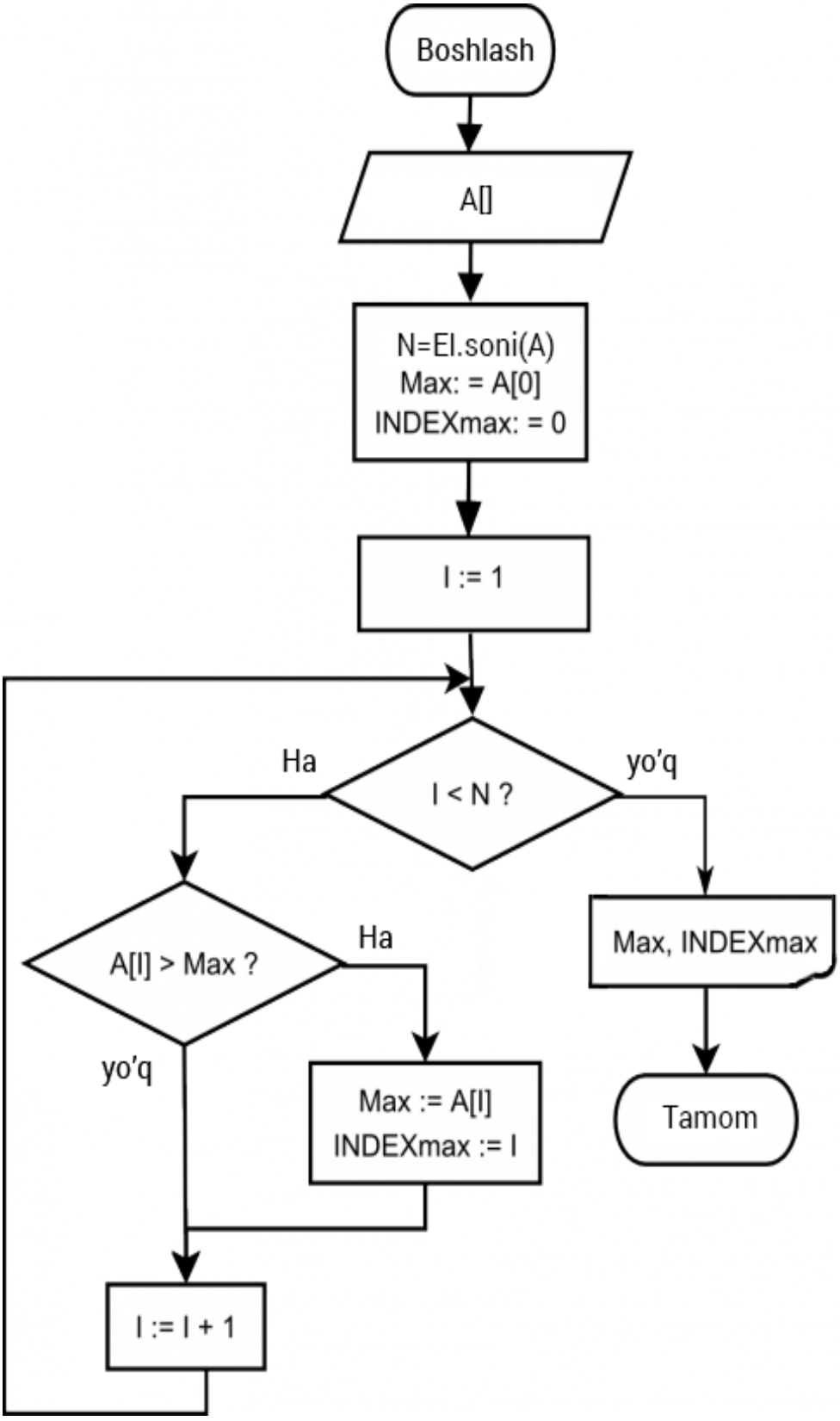
Massivning eng katta yoki eng kichik elementini aniqlashning juda ko’p usullari mavjud bo’lib, ulardan eng oddiy va universal usuli bu solishtiriluvchi elementgacha bo’lgan eng kichik yoki eng katta elementlarni ketma-ket keyingilari bilan solishtirish usulidir.

```
int FindMax(int* mass,int count)
{
            if(count <  1) return null;
            int max = mass[0];
            for (int i = 1; i < count; i++)
              if (mass[i] > max)
            max = mass[i];
   return max;
}
```

Bu usulning qolgan usullardan ustun tomoni shundaki, bu usul ixtiyoriy massiv uchun ishlaydi va har qanday dasturlash tili uchun dastur kodini yozish oson va resurs kam talab etiladi.

Bu usulning yagona kamchiligi shundan iboratki bu bu usul orqali juda katta hajmdagi massivlarda sekin ishlashidir.

Bu usullardan tashqari izlashning binar usuli mavjud bo'lib u juda katta hajmli massivlar uchun ishlatish mumkin. Lekin bu usul uchun juda ko'p kompyuter resursi talab etiladi.



1.1-rasm. Saralash algoritmi uchun blok sxema.

Dastur kodi.

```
#include <iostream>
using namespace std;
int main()
{    int *arr;
   int size;
      cout << "n = ";
   cin >> size;
   if (size <= 0) {
            cerr << "Massiv hajmi 0 dan katta bo'lishi kerak." << endl;
      return 1;    }
   arr = new int[size];
   for (int i = 0; i < size; i++) {
      cout << "arr[" << i << "] = ";
      cin >> arr[i];
   }
   int max = arr[0];
   for (int i = 1; i < size; i++) {
      if (arr[i] > max) {
         max = arr[i];
      }
   }
   cout << "max = " << max << endl;
   delete [] arr;
   return 0;
}
```

Nazorat savollari:

1. Masalalarni yechishda optimallashtirish nimaga kerak?
2. Masalalarni yechishda nisbiylik nimaga kerak?
3. Qidiruv algoritmlarida optimallashtirish va nisbiylik qanday qo'llaniladi?

2-Amaliy ish.

Mavzu: Hisoblash murakkabligi.

Ishdan maqsad: Masalalarni yechishning hisoblash murakkabligini o'rganish. Ushbu jarayonni o'yinlar yaratishda qo'llashni ko'rib chiqish.

Uslubiy ko'rsatmalar: Hisoblash murakkabligi – bir qancha algoritmlarni bajaruvchi ish hajmiga bog'liq funksiylarni belgilydigan informatika va algoritm nazariyasidagi tushuncha. Hisoblash murakkabligini o'rganish bo'limi hisoblash nazariyasi deb nomlanadi. Uning ish hajmi odatda hisoblash resursi deb nomlanuvchi fazo va vaqt abstract tushunchasi bilan o'lchanadi. Unga sarflanuvchi vaqt masalani yechish uchun zarur bo'lgan elementar qadamlar soni bilan aniqlanadi. Bunday holda, "kirish hajmiga bog'liq xotira bandligining hajmini va bajarish vaqtini qanday o'zgartirish kerak" degan algoritmlarni ishlab

chiqish savoliga javob beramiz. Kirish hajmi ostida bitlarda berilgan vazifalarning tavsifi uzunligi, chiqish hajmi ostida esa – masalani yechishning tavsifi uzunligi tushuniladi.

Murakkablik sinflari. Murakkablik sinflari – bu mavjud hisoblash murakkabligi bo'yicha o'xshash algoritmlar yechimini tanishning ko'p sonli vazifasidir. Bunda ikkita muhim sinf mavjud:

P sinfi. P sinfi "tez" hisoblanuvchi barcha yechimlar, muammolarni aralashtirib yuboradi. Bu graf kabilar bilan bo'gliqlikni aniqlovchi massivdan elementlarni qidirish va saralashga olib boradi.

NP sinfi. NP sinfi kiruvchi ma'lumotlar hajmiga bog'liq polynomial qadamlar sonini yechuvchi determinirlanmagan Tyuring mashinasi vazifalarini o'z ichiga oladi. Ularning yechimi polynomial qadamlar soni uchun determinirlangan Tyuring mashinasi yordamida tekshirilishi mumkin. Shuni aytish kerakki, determinirlanmagan Tyuring mashinasi faqat o'sh vaqtdagi chegaralangan xotirali determinirlangan Tyuring mashinasiga mos keluvchi zamonaviy kompyuterlarga o'xshash bo'lgan abstract modeldir. NP sinfi o'z ichiga P sinfini hamda kirish hajmiga eksponensial bog'liq faqat ma'lum bo'lgan algoritmlarni yechishning bir qancha muammolarini o'z ichiga oladi. NP sinfiga komivoyajer masalasi, bul formulalarini bajarish masalasi, faktorizatsiya masalasi va boshqa shu kabi masalalari kiradi.

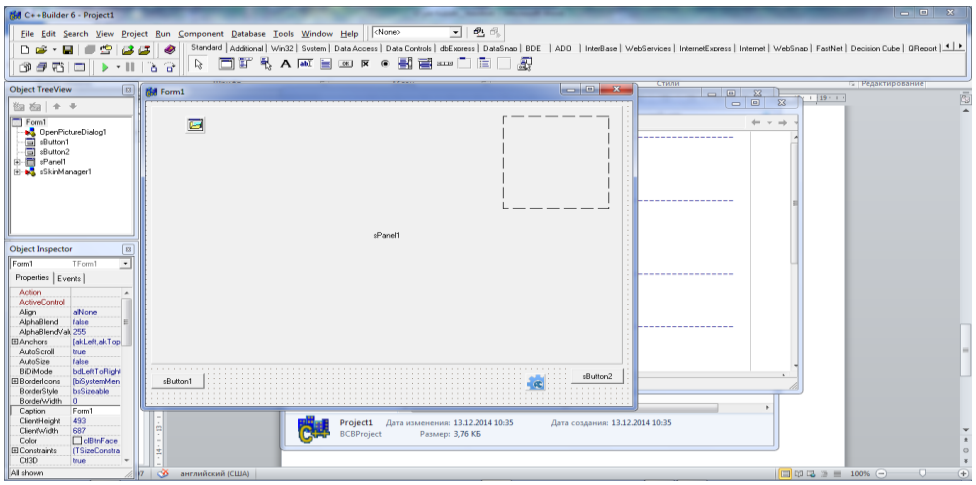
NP va P sinflari tengligi muammosi. Bu ikki sinf tengligi haqidagi savol nazariy informatikadagi eng ochiq muammolardan biri hisoblanadi. Kley matematika instituti bu muammoni ming yillik muammolar ro'yxatiga kiritgan va bu muammo yechimi uchun 1 million AQSH dollari miqdorida mukofot e'lon qilgan.

O'yin dasturining yaratilish jarayoni. Bu laboratoriyada puzzle yani bo'lak rasmlarni yig'ish o'yinini C++ builder muhitida tuzldi. Bu o'yinni tuzishda C++ builderning Additional komponentalar bo'limidagi Image, AlphaLite komponentalar bo'limidagi sButton, sPanel, sSkinManager va Dialogs komponentalari bo'limidagi OpenPictureDialog komponentalaridan foydalanildi.

O'yin algoritmi quyidagicha bo'ladi:

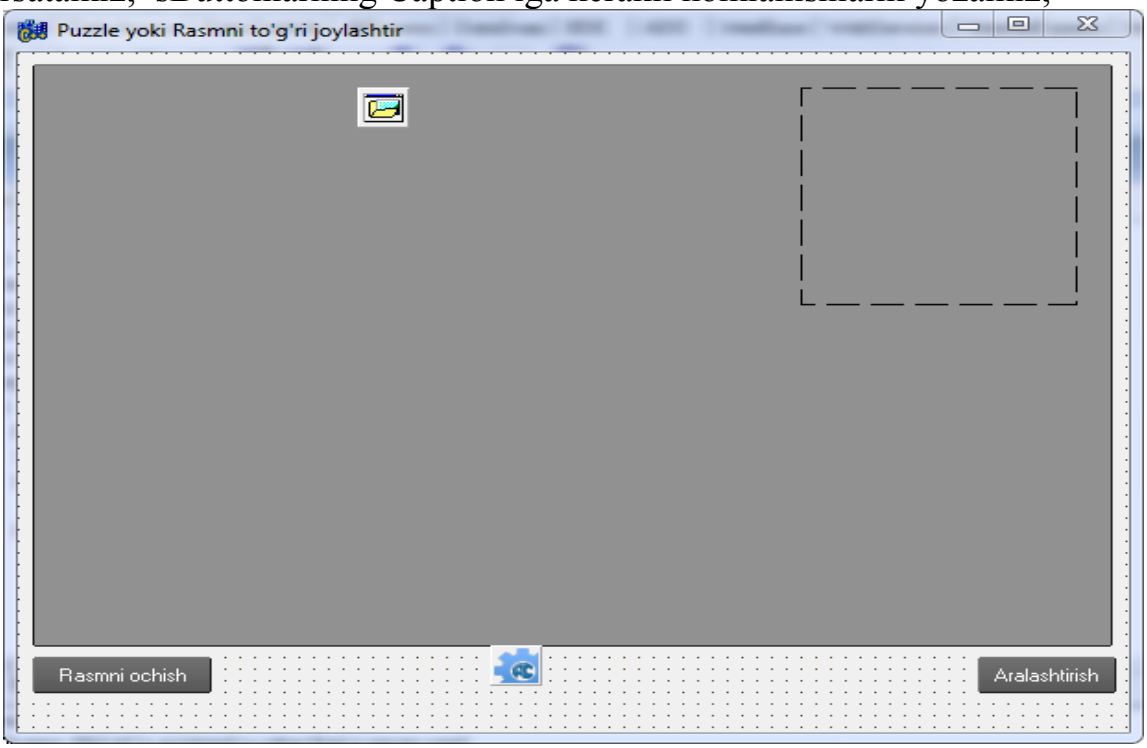
1. Rasm yuklab olinadi
2. Bo'laklarga ajratiladi va dastlabgi holat saqlab qo'yiladi
3. Rasm aralashtiriladi
4. Rasm bo'laklari harakatlantiriladi va ushbu holat dastlabki saqlangan holat bilan solishtiriladi. Agar joriy holat saqlangan holat bilan mos tushsa o'yin tugatiladi, aks holda o'yin davom ettiriladi

Bu o'yinni tuzish uchun avval C++ builderda forma hosil qilamiz va unga yuqoridagi komponentalarni joylashtiramiz.



2.1.rasm. Dasturni tanlash

Keyin joylashtirilgan komponentalarning properties dagi xususiyatlarini sozlab chiqamiz: sSkinManager ga qobiq fayl turgan papkani va qobiq nomini ko'rsatamiz; sButtonlarning Caption iga kerakli nomlanishlarni yozamiz;



2.2.rasm. Rasmni to'g'ri jaoylashtirish

Endi rasmni bo'laklarga bo'lingan holda chiquvchi 30x30 o'lchamli kataklarni hosil qilish uchun

```
TPole *A[30][30];
TPole *B[30][30];
```

maydon kodlarini kiritamiz.

Int tipidagi k va V o'zgaruvchilarini hosil qilamiz.

Rasmni ochish uchun sButton1ga quyidagicha kodlarni yozamiz(Bu kod bmp formatli rasmni OpenPictureDialog1 komponentasi orqali ochadi va biz hosil qilgan kataklarga bo'laklagan holda joylashtiradi. Bundan tashqari rasm ochilgan payt bu rasm Image1 ga yaxlit holda chiqariladi):

```
if(OpenPictureDialog1->Execute()==true)
```

```
{  for(j=0;j<Maydon_olchami_1;j++)
  {   for(i=0;i<Maydon_olchami_1;i++)
   {    delete A[i][j];
    delete B[i][j];
    A[i][j]=NULL;
    B[i][j]=NULL;
    delete PBitmap[j*Maydon_olchami_1+i];
    PBitmap[j*Maydon_olchami_1+i]=NULL;   }  }   Oyin_ishla();
MBitmap1 = new Graphics::TBitmap();
MBitmap1->Height=340;
MBitmap1->Width=340;
 try   {   MBitmap1->LoadFromFile(OpenPictureDialog1->FileName);
  Image1->Picture->LoadFromFile(OpenPictureDialog1->FileName);
  for(j=0;j<Maydon_olchami;j++)   {   for(i=0;i<Maydon_olchami;i++)
   {    if((i==Maydon_olchami-1)&&(j==Maydon_olchami-1))    {
A[i][j]->status=2;
    A[i][j]->SetA(i);
    A[i][j]->SetB(j);
    A[i][j]->X=IntToStr(j*Maydon_olchami+i+1);
    AS=i;
    BS=j;
    A[i][j]->Paint();
    B[i][j]->status=2;
    B[i][j]->SetA(i);
    B[i][j]->SetB(j);
    B[i][j]->X=IntToStr(j*Maydon_olchami+i+1);
    AS=i;
    BS=j;   }   else   {   A[i][j]->status=1;
    Alisher1(MBitmap1,A[i][j]->Bitmap1,i,j);
    A[i][j]->X=IntToStr(j*Maydon_olchami+i+1);
    A[i][j]->SetA(i);
    A[i][j]->SetB(j);
    A[i][j]->Paint();
    B[i][j]->status=1;
   Alisher1(MBitmap1,B[i][j]->Bitmap1,i,j);
    B[i][j]->X=IntToStr(j*Maydon_olchami+i+1);
    B[i][j]->SetA(i);
    B[i][j]->SetB(j);   }   } } } catch (...)   {    MessageBeep(0);  }
  delete MBitmap1;  }
```

Rasmni almashtirishda esa quyidagi kodlar yoziladi(Bu kodda rasm almashtirish xuddi ikki o’lchovli massivlarda bajariladigan amallarkabi bajariladi):

```
RoyxatToplami();
randomize();
for(int j=0;j<Maydon_olchami;j++) {
```

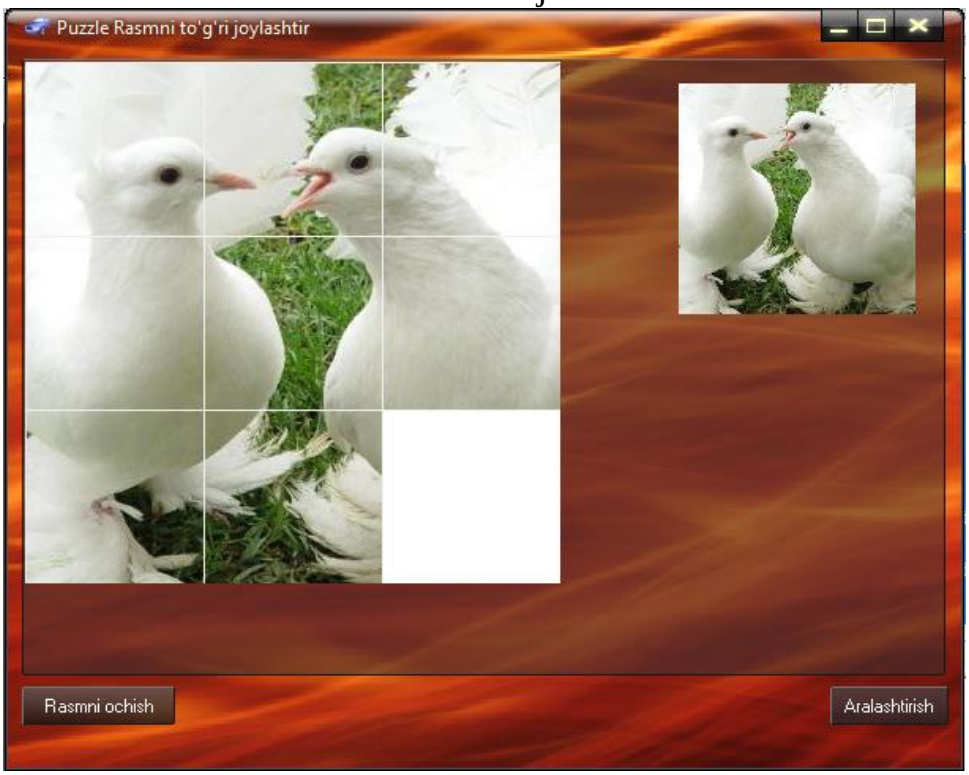
```
        for(int i=0;i<Maydon_olchami;i++)  { {
if((j*Maydon_olchami+i)==(Maydon_olchami*Maydon_olchami-1))
          {    A[i][j]->status=2;
            AS=i;
            BS=j;
            A[i][j]->X=IntToStr(Maydon_olchami*Maydon_olchami);
            A[i][j]->Paint();  }   else    {      A[i][j]->status=1;
              Alisher2=random(MyList->Count);
              AStruct=(PAList)MyList->Items[Alisher2];
              A[i][j]->Bitmap1=B[AStruct->x][AStruct->y]->Bitmap1;
              A[i][j]->X=IntToStr(AStruct->I);
              MyList->Delete(Alisher2);
              A[i][j]->Paint();      }   }  }  }
        MeningRoyxatim();
        int bp=0;
        bp=(Alisher());
        if((bp%2)!=0)  {    TPole *C=new TPole(this);
            Graphics::TBitmap *CBitmap = new Graphics::TBitmap();
            C->Bitmap1=CBitmap;
            C->Bitmap1= A[Maydon_olchami-2][Maydon_olchami-1]->Bitmap1;
            C->X=A[Maydon_olchami-2][Maydon_olchami-1]->X;
            A[Maydon_olchami-2][Maydon_olchami-1]-
>Bitmap1=A[Maydon_olchami-3][Maydon_olchami-1]->Bitmap1;
            A[Maydon_olchami-2][Maydon_olchami-1]-
>Bitmap1=A[Maydon_olchami-3][Maydon_olchami-1]->Bitmap1;
            A[Maydon_olchami-2][Maydon_olchami-1]->X=A[Maydon_olchami-
3][Maydon_olchami-1]->X;
            A[Maydon_olchami-3][Maydon_olchami-1]->Bitmap1=C->Bitmap1;
            A[Maydon_olchami-3][Maydon_olchami-1]->Bitmap1=C->Bitmap1;
            A[Maydon_olchami-3][Maydon_olchami-1]->X=C->X;
            delete CBitmap;
            delete C;
            bp=(Alisher());
            A[Maydon_olchami-3][Maydon_olchami-1]->Paint();
            A[Maydon_olchami-2][Maydon_olchami-1]->Paint(); }
```

Budastur kodi uchun Main.h kutubxonasida ba’zi klass va strukturalar yaratiladi:

```
        #ifndef MainH
        #define MainH
        #include <Classes.hpp>
        #include <Controls.hpp>
        #include <StdCtrls.hpp>
        #include <Forms.hpp>
        #include <Menus.hpp>
        #include <ExtCtrls.hpp>
```

```
#include <Graphics.hpp>
#include <Dialogs.hpp>
#include <ExtDlgs.hpp>
#include "sButton.hpp"
#include "sPanel.hpp"
#include "sSkinProvider.hpp"
#include "sSkinManager.hpp"
#define CM_XXX WM_APP+2
#define CM_XX1 WM_APP+3
typedef struct AList   {  int I;
int x;
int y;  }   TAList;
typedef TAList* PAList;
class TForm1 : public TForm     {   __published:
        TsPanel *sPanel1;
        TImage *Image1;
        TOpenPictureDialog *OpenPictureDialog1;
        TsButton *sButton2;
        TsButton *sButton1;
        TsSkinManager *sSkinManager1;
        void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
        void __fastcall FormKeyDown(TObject *Sender, WORD &Key,
          TShiftState Shift);
        void __fastcall sButton1Click(TObject *Sender);
        void __fastcall sButton2Click(TObject *Sender);
private:
int AS,BS;
int i,j;
        Graphics::TBitmap *MBitmap1;
        Graphics::TBitmap *PBitmap[900];
        void __fastcall Oyin_ishla();
        void __fastcall Alisher1(Graphics::TBitmap
*Asosiy,Graphics::TBitmap *Qoshimcha,int k,int j);
        void __fastcall WndProc(Messages::TMessage &Message);
        void __fastcall RoyxatToplami();
        void __fastcall MeningRoyxatim();
public:                 // foydalanuvchi e'lonlari
int Maydon_olchami,Maydon_olchami_1,Alisher2;
TList *MyList;
PAList AStruct;
PAList A1Struct;
     __fastcall TForm1(TComponent* Owner);
      int Alisher(void);  };
extern PACKAGE TForm1 *Form1;
#endif
```

Dastur natijasi.



2.3.rasm. Rasmni to’gri joylashtirilganligi

Nazorat savollari

1. Hisoblash murakkabligi nima?
2. Murakkablik sinflari nechta va ular qaysilar?
3. NP sinfini tushuntiring.

3-Amaliy ish.
Mavzu. To’g’ri zanjir usuli.

Ishdan maqsad: Qoidalarning to’g’ri zanjiri tushunchasi o’rganish va qoidalarning to’g’ri zanjir usuli asosida hosil yetishtirishni tahlil qiluvchi dastur ishlab chiqish.

Uslubiy ko’rsatmalar: Sun’iy intellekt tizimining an’anaviy dasturiy tizimlaridan asosiy farqi shundan iboratki, uning tuzilishining tarkibiy qismlari bo’lingan holda aniqlanadi va uning istalgan qismini zamonaviylashtirish umumiy tuzilmaga ta’sir qilmaydi.

Insonning miyasi hatto eng oddiy masalani echishga kirishganda ham kerakli harakatlarni tanlash uchun uning ixtiyorida axborotlarning katta hajmi mavjud . Masalan, ishga keta turib, odam uydan chiqadi va ko’chani kesib o’tadi, ammo u o’tish uchun paytni tanlaguniga qadar, uning miyasi ham harakatning jadalligini ham transportning tezligini, hamma qarama-qarshi tomongacha bo’lgan masofani tahlil qiladi. shuning bilan bir vaqtda miya ko’chani kesib o’tishga to’g’ridan to’g’ri tegishli bo’lmagan axborotlarni, ya’ni ob-xavoni, o’tib ketayotgan mashinalarning rangini ishlab chiqadi. Bundan tashqari odam qaerga ketayotganligi, u erga qancha vaqtda etib borishi, kim bilan uchrashishi kerakligi haqida uylaydi. Ammo agar odam ko’chani kesib o’tishdan avval ko’chani kesib

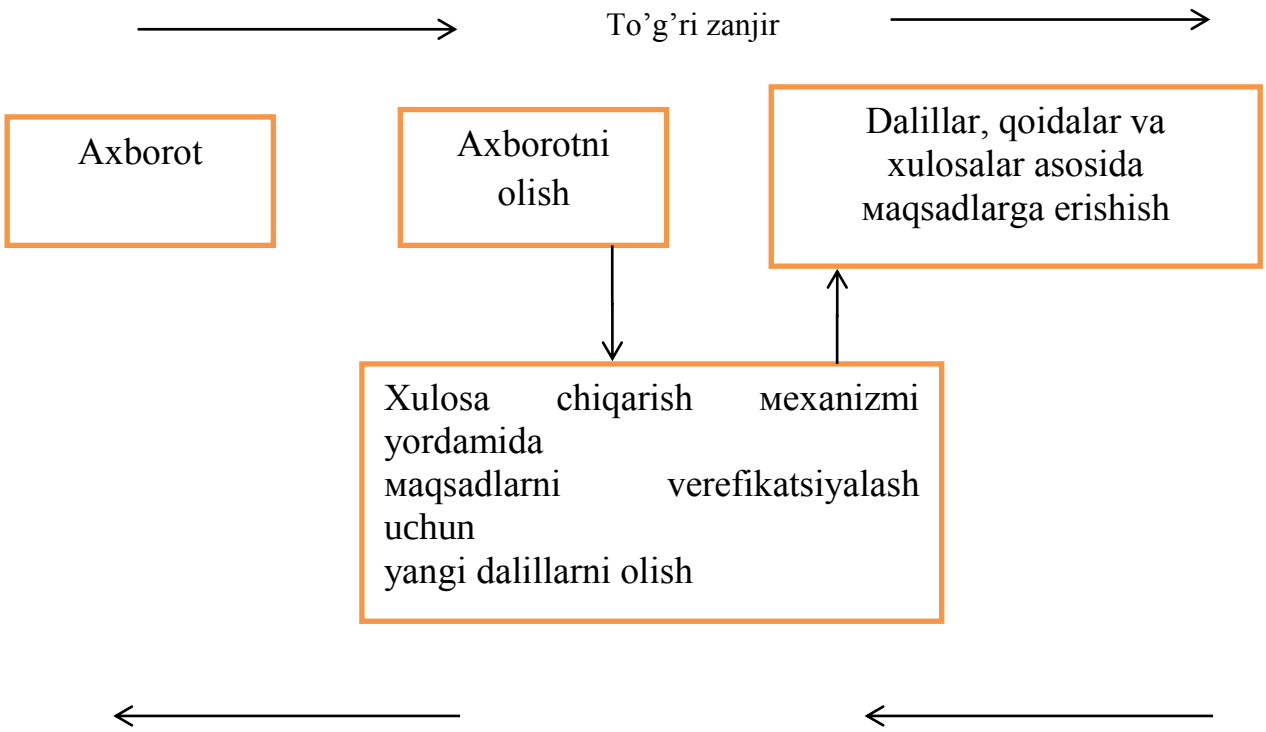
o’tish maqsadiga bilvosita va bevosita aloqador barcha dalillarni tahlil qilganda, u bir necha yil turib qolishi mumkin edi. shunday qilib, inson miyasiga ham aniq vaziyatlarda to’g’ri reaktsiya tanlashga rahbarlik qiluvchi murakkab tizim mavjuddir. Bunday tanlov soddalashtirish deb ataladi. Soddalashtirish mexanizmi ushbu paytda echilaYotgan vazifaga aloqasi bo’lmagan dalillarni to’sib qo’yadi.

Sun’iy intellekt tizimlarini loyihalashtirishda birinchi bosqichda unga erishish uchun mo’ljallangan maqsad aniqlanadi, echilayotgan vazifalarni yirik atamalarda bayon qilishni bilish uchun zarur sinfi belgilaniladi. Dalillar sun’iy intellekt tizimining muhim qismi bo’ladi, ularsiz maqsadga erishish mumkin emas. Har bir maqsadning o’zining dalillari bor. Har bir dalil o’zining salmog’iga ega, ya’ni har bir dalilga nisbatan muhimlik hosdir. Ushbu vazifani echish uchun dalil qanchalik katta ahamiyatga ega bo’lsa, uning salmog’i shunchalik katta. Maqsadlarga erishish uchun zarur bo’lgan umumiy dalillar aniqlangandan keyin, aniq ma’lumotlarni olish kerak. Ma’lumotlarni olish uchun tegishli savollar shakllantiriladi, ularga javoblar tizimini yakuniy qarorga kelishiga yordam beradi. Sun’iy intellect tizimlari uchun dasturni ishlab chiqish quyidagi bosqichlardan iborat.

1. Maqsadlarni aniqlash.
2. Bu maqsadlarga tegishli dalillarni aniqlash.
3. Ushbu vaziyat uchun hos bo’lgan dalillarga mos bo’lgan ma’lumotlarni olish.
4. Xulosa chiqarish qoidalari va mexanizmidan foydalanish bilan ma’lumotlarni olish.

Ma’lum dalillar, qoidalarga ko’ra berilgan vaziyatga nisbatan qo’llaniladi. qoidalar ma’lumotlarni to’g’ri baholash va maqsadga erishishga yordamlashadi.

Yuqoridagi bayon qilingan bosqichlar yordamida maqsadlarga erishish jarayoni mulohazalarning to’g’ridan-to’g’ri zanjiri, ya’ni ma’lumotlardan mantiqiy xulosaga boruvchi zanjir deb ataladi. Maqsadga erishilganda uning to’g’riligini tekshirish, ya’ni ma’lumotlar va qoidalar bilan yana vazifani tahlil qilish zarurdir. Bu jaraYon maqsadlarni verifikatsiyalash deb ataladi. Yana avvaldan to’g’ri deb faraz qilingan xulosani tasdiqlash uchun yangi ma’lumotlarni chiqarish mexanizmlarini tanlash teskari zanjirga misol bo’lishi mumkin. shunday qilib, mulohazalarning teskari zanjiri to’g’ri zanjirga qarama qarshi tomonga, ya’ni xulosadan ma’lumotlarga ketadi. Teskari zanjir faqat maqsadga erishilgandan keyin vujudga keladi. 1-rasmda sun’iy intellekt tizimining konfiguratsiyasi va ishi berilgan.



3.1.rasm. Sun’iy intellekt tizimining konfiguratsiyasi va ishi

To’g’ri zanjir usulida ma’lumotlar qayta ishlanishi jarayonida natijalar ma’lumotlarning o’zaro to’g’ri zanjirli aloqasi asosida chiqariladi. Bunda ma’lumotlarni qayta ishlash bilimlar bazasidagi ma’lumotlar asosida qayta ishlanadi.

Buning uchun hosil yetishishi natijasini tahlil qiluvchi dasturni ko’rib chiqamiz. Bunda dastur ishlashi quyidagicha bo’ladi: dasturda hosil yetishtirishga bog’liq bo’lgan faktorlar beriladi va siz bu faktorlarni keraklisini kiritasiz. Dastur esa o’zidagi bilimlar bazasi asosida berilgan ma’lumotlarni qayta ishlaydi va hosilning qay darajada yetishishini foiz hisobida chiqarib beradi.

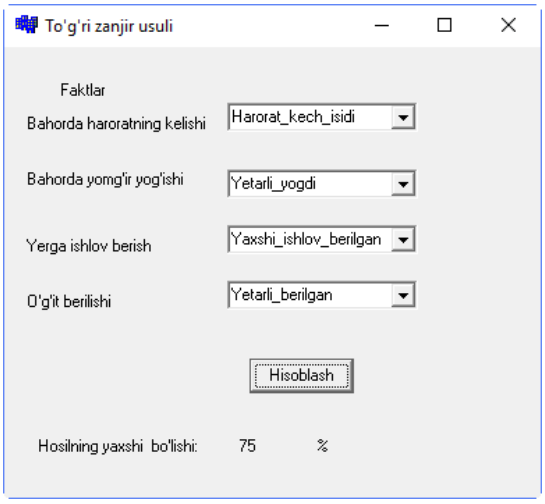
Yaxshi hosil olish uchun to’g’ri zanjir usuli

1. Agar bahorda o'z vaqtida havo ilisa, u holda hosil vaqtida yetilishi mumkin.
2. Agar bahorda yetarli darajada yomg'ir yog'sa, u holda hosil yaxshi bo'lishi mumkin.
3. Agar kuzda yerlar yaxshi haydalib, ishlov berilgan bo'lsa, u holda hosil to'liq bo'lishi mumkin.
4. Agar yerga o'g'it yetarli berilsa, u holda ekin quvvatli bo'lib, holsil to'la bo'lishi mumkin.
5. Agar yuqoridagilarni barchasi bo’lsa, hosil aniq yaxshi bo'ladi.

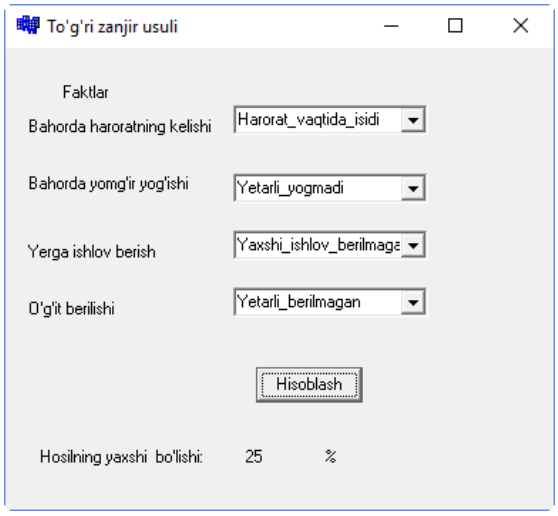
Dastur kodi va natijasi.

```
#include <vcl.h>
    #pragma hdrstop
    #include "Unit1.h"
    #pragma package(smart_init)
```

```
        #pragma resource "*.dfm"
TForm1 *Form1;
void __fastcall TForm1::Button1Click(TObject *Sender)
        {float a = 0, b = 0, c = 0, d = 0, f = 4;
        if(ComboBox1->Text == "Harorat_vaqtida_isidi"&&
        ComboBox2->Text == "Yetarli_yogdi"&&
        ComboBox3->Text == "Yaxshi_ishlov_berilgan"&&
        ComboBox4->Text == "Yetarli_berilgan"
        ){      a = 1; b=1; c = 1; d = 1;
             Label7->Caption = FloatToStr(((a + b + c + d)/f)*100);}
        if(ComboBox1->Text == "Harorat_kech_isidi"&&
        ComboBox2->Text == "Yetarli_yogdi"&&
        ComboBox3->Text == "Yaxshi_ishlov_berilgan"&&
        ComboBox4->Text == "Yetarli_berilgan"
        ){       a = 0; b=1; c = 1; d = 1;
             Label7->Caption = FloatToStr(((a + b + c + d)/f)*100);}
        if( ComboBox1->Text == "Harorat_vaqtida_isidi"&&
        ComboBox2->Text == "Yetarli_yogmadi"&&
        ComboBox3->Text == "Yaxshi_ishlov_berilgan"&&
        ComboBox4->Text == "Yetarli_berilgan"
        ){      a = 1; b=0; c = 1; d = 1;
             Label7->Caption = FloatToStr(((a + b + c + d)/f)*100); }
        if(ComboBox1->Text == "Harorat_vaqtida_isidi"&&
        ComboBox2->Text == "Yetarli_yogdi"&&
        ComboBox3->Text == "Yaxshi_ishlov_berilmagan"&&
        ComboBox4->Text == "Yetarli_berilgan"
        ){      a = 1; b=1; c = 0; d = 1;
             Label7->Caption = FloatToStr(((a + b + c + d)/f)*100);
        }if(ComboBox1->Text == "Harorat_vaqtida_isidi"&&
        ComboBox2->Text == "Yetarli_yogdi"&&
        ComboBox3->Text == "Yaxshi_ishlov_berilgan"&&
        ComboBox4->Text == "Yetarli_berilmagan"
        ){      a = 1; b=1; c = 1; d = 0;
             Label7->Caption = FloatToStr(((a + b + c + d)/f)*100); }
        if(ComboBox1->Text == "Harorat_vaqtida_isidi"&&
        ComboBox2->Text == "Yetarli_yogmadi"&&
        ComboBox3->Text == "Yaxshi_ishlov_berilmagan"&&
        ComboBox4->Text == "Yetarli_berilmagan")
        {       a = 1; b=0; c = 0; d = 0;
             Label7->Caption = FloatToStr(((a + b + c + d)/f)*100); } }
```



3.2.rasm. To’gri zanjir.



3.3.rasm. To’g’ri zanjir natijasi

Nazorat savollari

1. To’g’ri zanjir usulida masalalarni yechishni tushuntiring
2. To’g’ri zanjir usulida masalalardagi ma’lumotlar o’zaro qanday bog’langan?

4-Amaliy ish.
Mavzu: Qoidalar tizimini ishlab chiqish. Poker o’yini.

Ishdan maqsad: O'yinlar yaratishda qoidalar tizimidan foydalanishni o'rganish.

Uslubiy ko’rsatmalar: Odam bilimlarni fikrlash usulini o’zgartirmasdan, ma’lum bo’lgan dalillarni esdan chiqarmasdan jamlashi mumkin. Sun’iy intellekt tizimi xuddi shunday ishlab chiqiladi. Bunda inson xotirasining bloklariga o’xshab dasturlarning ayrim qismlarining yuqori mustaqilligiga erishiladi.

Odamning miyasi kerakli axborotni tanlab turib, faqat ushbu muammoga tegishli bo’lgan dalillarni ulaydi, bunda u kirishi mumkin bo’lgan barcha ma’lumotlardan foydalanmaydi. Inson faoliyatining asosida fikrlash yotadi va ushbu holda fikrlash jarayonining maqsadi yakuniy natija bo’ladi. Bitta maqsadga erishilgandan keyin yangi maqsad qo’yiladi va erishiladi. Maqsadlarni mahalliy va asosiyga bo’lish mumkin. shundan kelib chiqqan holda intellektning ta’rifini berish mumkin.

Intellekt dalillarning majmuasi va ularni belgilangan maqsadga erishish uchun qo’llash usullaridan iborat bo’ladi. Maqsadga erishish esa - bu tegishli omillardan foydalanishning kerakli qoidalarini qo’llanishidir.

Inson har qanday amal yoki jarayonni boshlashdan avval uning algoritmi, qoidalar tizimini ishlab chiqadi. Qoidalar tizimini ishlab chiqish eng muhim amallardan biri hisoblanadi. Shuning uchun ham sun'iy intellekt asosida ishlovchi ko'pgina dastur, qurilma va tizimlarga o’z yo'nalishiga mos keluvchi qoidalar tizimi kiritiladi. Bu jarayonni dasturiy ta’minotni yaratish jarayonidagi qadamlar orqali o’rganamiz.

Dasturiy ta’minotni yaratish jarayoni bir qancha ost jarayonlardan tashkil topadi. Bu ostjarayonlardan bir qanchasi quyida berilgan. Sharshara modelida bu ostjarayonlar o’zaro ketma-ket amalga oshiriladilar, boshqa shunga o’xshash jarayonlarda ularning ketma-ketligi va tarkibi o’zgaradi.

- Talablarni tahlillash.
- Dasturiy ta’minotni loyihalashtirish.
- Dasturiy ta’minotni testlash
- Tizim integratsiyasi.
- Dasturiy ta’minotni ishlatish.
- Dasturiy ta’minotni kuzatib boorish

Talablarni tahlillash – dasturiy ta’minotga bo’lgan talablar to’plamini o’z ichiga oluvchi dasturiy ta’minotni ishlab chiqishni o’z ichiga oluvchi jarayon. Talablarni tahlillash 3ta turni o’z ichiga oladi.

- Talablarni to’plash – foydalanuvchilar bilan suhbatlashib, ularning talab va xohishlarini o’rganish.
- Talablarni tahlillash – yig’ilgan ma’lumotlar ichida keraksizi, bir qiymatlisi, to’liqmaslari kabilar bor yo’qligini tekshirish va ularni hal etish, o’zaro bog’liq talablarni aniqlash.
- Talablarni hujjatlashtirish – talablar turli xil ko’rinishlarda hujjatlashtirilishi mumkin. Oddiy tavsiflar yordamida, foydalanish senariysi ko’rinishida, foydalanuvchi tarixi ko’rinishida yoki jarayonlar spetsifikatsiyasi ko’rinishida.

Dasturiy ta’minotni loyihalash – dasturiy ta’minotning o’zidan kelib chiqqan holda qanday usul, algoritm bilan yaratish mumkinligini aniqlash va dasturiy ta’minot loyihasini yaratish jarayoni. Loyihalash maqsadi tizimning ichki xususiyatlarini aniqlash va uning tashqi xususiyatlarini dastiriy ta’minotga qo’yilgan talab asosida detallashtirish.

Aynan dasturni loyihalash jarayonida dasturda ishlovchi algoritmlarning qoidalar tizimi ishlab chiqiladi. Ya’ni dasturiy ta’minotda qo’llangan algoritmlarning qay biri qanday vaziyatda ishlashi belgilab beriladi. Qoidalar tizimini ishlab chiqish sun’iy intellektni yaratishning asosiy bosqichlaridan biri hisoblanadi. Masalan, shaxmat o’yini dasturini olaylik. Unda foydalanuvchi va kompyuter orasida o’yin o’ynash jarayoni amalga oshirilsin. Bunda dastur kodiga ko’p miqdordagi kombinatsion yurishlar algoritmi kiritilgan bo’ladi. Bu esa dasturning qoidalar tizimiga foydalanuvchi yurgan kombinatsiyaga qarshi kombinatsiyadan foydalanish imkonini beradi. Ya’ni masalan, farzin a:1 dan f:1ga yursa, dastur qoidalar tizimidan kelib chiqqan holda piyoda bilan f:1 ga yurib farzinni oladi. Bu jarayonda hozirda shaxmat o’yinida mavjud bo’lgan barcha yurish kombinatsiyalari dasturning bazasiga kiritib qo’yilgan bo’ladi. O’yin davomida dastur foydalanuvchi tomonidan kiritilgan kombinatsiyaga qarshi o’z kombinatsiyasini bazasidagi mavjud kombinatsiyalarni tahlil qilib, ular orasidan keraklisini tanlab oladi. Qoidalar tizimining ishlab chiqilishi orqali dastur sun’iy intellectga aylantiriladi.

Misolni ko’rib chiqamiz:

Dalil 1. yoqib qo’yilgan plita issik.

Qoida 1. Agar yoqib qo’yilgan plitaga qo’lni qo’ysa, unda kuyib qolish mumkin.

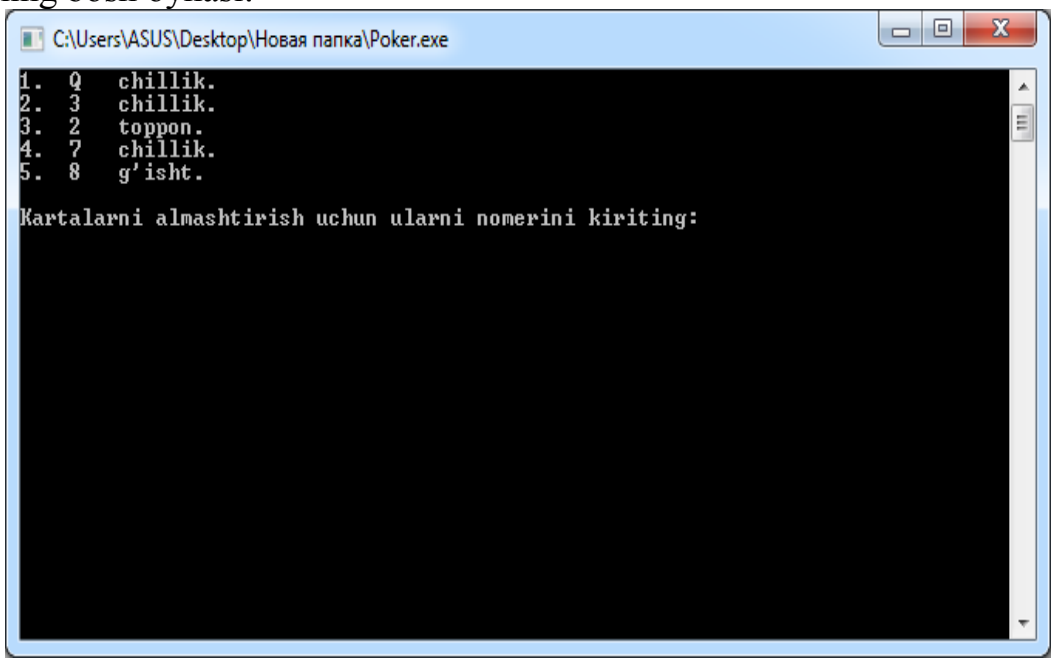
Dalil 2. Tig’iz paytda ko’chada mashinalar ko’p.

Qoida 2. Agar tig’iz paytda ko’chani kesib o’tishga harakat qilinsa, unda mashina ostiga tushib qolish mumkin.

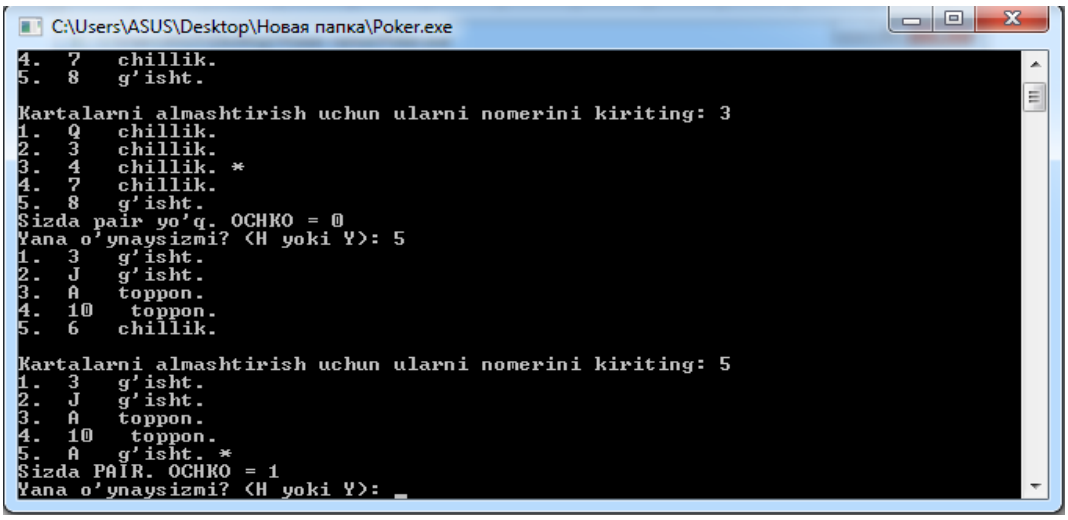
Qoidalar tizimining ishlashini tushunish uchun poker o'yini dasturini ishlashini ko'rib chiqamiz. Dastlab poker o’yini qanday o’yin ekanligini bilib olish kerak. Poker o’yinida ikkitadan kam bo’lmagan o’yinchilar to’rt yoki beshtadan kartalarni ishlatgan holda eng yuqori poker kombinatsiyasini yig’ishga harakat qiladi. Pokerda 32talik, 36talik va 52talik kartalardan foydalanish mumkin. Kartalarning qiymati “tuz”dan boshlab kamayib boradi va shu tariqa karol, dama va hk 2gacha. Pokerning turiga qarab o’yin bir necha fazalardan iborat. Ularning har biri karta tarqatish bilan boshlanadi. Karta tarqatilgandan so’ng o’yinchi xohlasa biror narsa evaziga o’yinni davom ettirishi yoki o’yindan chiqib ketishi mumkin. Eng yaxshi kombinatsiyali karta egasi yoki qolgan o’yinchilarni chiqarib yuborgan o’yinchi g’olib hisoblanadi.

O’yinni tushuntirilishi davomida sezgan bo’lsangiz o’yin dasturini tuzishda uchta asosit algoritm ishlatiladi: O’yinchilarga kartalarni tasodifiy holda tarqatish, o’yinchilarning kartalarini o’sish tartibida saralash va ularni o’zaro solishtirish. Solishtirish jarayonida dastlab eng yuqorida turgan kartalar solishtiriladi. Agar ular teng bo’lsa, solishtirish davom etadi, aks holda kimning kartasi kichik bo’lsa shu o’yinchi chiqarib yuboriladi.

O’yinning bosh oynasi:



4.1.rasm. Nomer kiritish



4.2.rasm. Natijalarni ko'rish

```
#include <string>
#include "card.h"
Card::Card(){}
Card::Card(int rank, int suite)
{    this->rank = rank;
   this->suite = suite;
}void Card::setRank(int rank)
{    this->rank = rank;
}void Card::setSuite(int suite)
{    this->suite = suite;
}int Card::getRank()
{    return this->rank;
}int Card::getSuite()
{    return this->suite;
}
std::string Card::display()
{
   static const std::string aRanks[] = {" 2", " 3", " 4", " 5", " 6",
         " 7", " 8", " 9", " 10", " J", " Q", " K", " A"};
   static const std::string aSuits[] = {"chillik", "g'isht", "toppon", "qarg'a"};
   return aRanks[rank] + "   " + aSuits[suite] + ".";
}
#include <iostream>
#include <ctime>
#include <string>
#include <cstdlib>
#include <algorithm>
#include "desk.h"
#include "card.h"
Deck::Deck()
{    srand(time(0));
   for(int i = 0; i < 52; ++i){        cards[i] = i;
```

```
   }    shuffle();}
void Deck::shuffle()
{    iCard = 0;
   std::random_shuffle(cards, cards + 52);
}
Card Deck::dealACard()
{    if(iCard > 51)
   {        std::cout << std::endl << "Aralashtirilmoqda..." << std::endl;
      shuffle();
   }    int r = cards[iCard] % 13;
   int s = cards[iCard++] / 13;
   return Card(r, s); }
```

Nazorat savollari

1. Qoidalar tizimi qanday ishlab chiqiladi?
2. Qoidalar tizimini ishlab chiqishda nimalardan foydalaniladi?

5-Amaliy ish.

Mavzu: Qoidalar tizimini ishlab chiqish. Oila a`zolari o`rtasidagi ishlab chiqish.

Ishdan maqsad: Qoidalar tizimini ishlab chiqishni o'rganish. Oila a`zolari o`rtasida munosabatlarni ishlab chiqish .

Uslubiy ko'rsatmalar: Insonning miyasi hatto eng oddiy masalani echishga kirishganda ham kerakli harakatlarni tanlash uchun uning ixtiyorida axborotlarning katta hajmi mavjud . Masalan, ishga keta turib, odam uydan chiqadi va ko'chani kesib o'tadi, ammo u o'tish uchun paytni tanlaguniga qadar, uning miyasi ham harakatning jadalligini ham transportning tezligini, hamma qarama-qarshi tomongacha bo'lgan masofani tahlil qiladi. shuning bilan bir vaqtda miya ko'chani kesib o'tishga to'g'ridan to'g'ri tegishli bo'lmagan axborotlarni, ya'ni ob-xavoni, o'tib ketayotgan mashinalarning rangini ishlab chiqadi. Bundan tashqari odam qaerga ketayotganligi, u erga qancha vaqtda etib borishi, kim bilan uchrashishi kerakligi haqida uylaydi. Ammo agar odam ko'chani kesib o'tishdan avval ko'chani kesib o'tish maqsadiga bilvosita va bevosita aloqador barcha dalillarni tahlil qilganda, u bir necha yil turib qolishi mumkin edi. shunday qilib, inson miyasiga ham aniq vaziyatlarda to'g'ri reaktsiya tanlashga rahbarlik qiluvchi murakkab tizim mavjuddir. Bunday tanlov soddalashtirish deb ataladi. Soddalashtirish mexanizmi ushbu paytda echilaYotgan vazifaga aloqasi bo'lmagan dalillarni to'sib qo'yadi.

Maqsadga erisha turib, inson nafaqat oldiga qo'yilgan vazifani echimiga keladi, balki bir vaqtda yangi bilimlarni oladi. Masalan, Jon va Meri – Jeyinning ota-onalari. Maqsad - Jim va Jeyn bir birlariga kim bo'lishlarini aniqlashdir.

Soddalashtirish mexanizmi odamga uning xotirasida saqlanayotgan quyidagi qoidaga murojaat qilishga majbur qiladi: Agar qiz va o'g'il bolada bitta ota-onalar bo'lsa, unda o'g'il va qiz bola aka va singil. Bu maqsadga erishish jarayonida yangi dalil - Jim va Jeyin aka va singil ekanligi aniqlandi.

Intellektning yangi dalillarni chiqarib olishga yordamlashadigan qismi xulosa chiqarish mexanizmi deb ataladi. Xuddi xulosa chiqarish mexanizmi insonga tajribadan o'rganishiga imkon beradi va mavjud bilimlarni yangi vaziyatga qo'llab, mavjudlarda yangi dalillarni generatsiyalash imkonini beradi.

Intellekt dalillarning majmuasi va ularni belgilangan maqsadga erishish uchun qo'llash usullaridan iborat bo'ladi. Maqsadga erishish esa - bu tegishli omillardan foydalanishning kerakli qoidalarini qo'llanishidir.

Misolni ko'rib chiqamiz:

Dalil 1. Devor ostidagi elektr simidan elektr toki oqmoqda va devor nam.

Qoida 1. Agar devorga tegsangiz sizni tok uradi.

Dalil 2. Osmonda qora bulutlar suzib yuribdi

Qoida 2. Agar yomg'irpo'sh olmasangiz, yomg'ir ostida qolib ketasiz.

Dalil 3. Kavsharlash apparati ishlatilishi paytida chiquvchi nur ko'z pardasini keskin zararlaydi.

Qoida 3. Agar ko'zga himoyalovchi ko'zoynak taqmasangiz ko'zingiz zararlanadi.

Sun'iy intellekt tizimlarini loyihalashtirishda birinchi bosqichda unga erishish uchun mo'ljallangan maqsad aniqlanadi, echilayotgan vazifalarni yirik atamalarda bayon qilishni bilish uchun zarur sinfi belgilaniladi. Dalillar sun'iy intellekt tizimining muhim qismi bo'ladi, ularsiz maqsadga erishish mumkin emas. Har bir maqsadning o'zining dalillari bor. Har bir dalil o'zining salmog'iga ega, ya'ni har bir dalilga nisbatan muhimlik hosdir. Ushbu vazifani echish uchun dalil qanchalik katta ahamiyatga ega bo'lsa, uning salmog'i shunchalik katta. Maqsadlarga erishish uchun zarur bo'lgan umumiy dalillar aniqlangandan keyin, aniq ma'lumotlarni olish kerak. Ma'lumotlarni olish uchun tegishli savollar shakllantiriladi, ularga javoblar tizimini yakuniy qarorga kelishiga yordam beradi. Sun'iy intellect tizimlari uchun dasturni ishlab chiqish quyidagi bosqichlardan iborat.

1. Maqsadlarni aniqlash.
2. Bu maqsadlarga tegishli dalillarni aniqlash.
3. Ushbu vaziyat uchun hos bo'lgan dalillarga mos bo'lgan ma'lumotlarni olish.
4. Xulosa chiqarish qoidalari va mexanizmidan foydalanish bilan ma'lumotlarni olish.

Ma'lum dalillar, qoidalarga ko'ra berilgan vaziyatga nisbatan qo'llaniladi. qoidalar ma'lumotlarni to'g'ri baholash va maqsadga erishishga yordamlashadi.

Oila a`zolari o`rtasidagi qoidalar ishlab chiqishda biz turli mantiqiy operattorlardan foydalanamiz . Bu operatorlarga agar , va , yoki , emas kabilar kiradi. Quyidagini misol qilib keltirishimiz mumkin :

Agar (va (ota-onasi (?x) (?y), ota-onasi (?x) (?z)),
U holda (qarindosh (?y) (?z))).

Mana shu qoidalarga asosan biz oila a`zolari o`rtasida qonuniyatlarni ishlab chiqamiz .Masalan

Agar (va (akasi (?x) (?y) , akasi (?y) (?z)),u holda (qarindosh (?x) (?z)).

Yoki quyidagicha

Ota-onasi Nusratillo Oyqiz va Ota-onasi Ubaydullo Gulbahor

Demak quyidagicha bo`ladi :

qarindosh Aziz Mahfuza.

Emas operatoriga esa quyidagicha misol keltirsak bo`ladi :

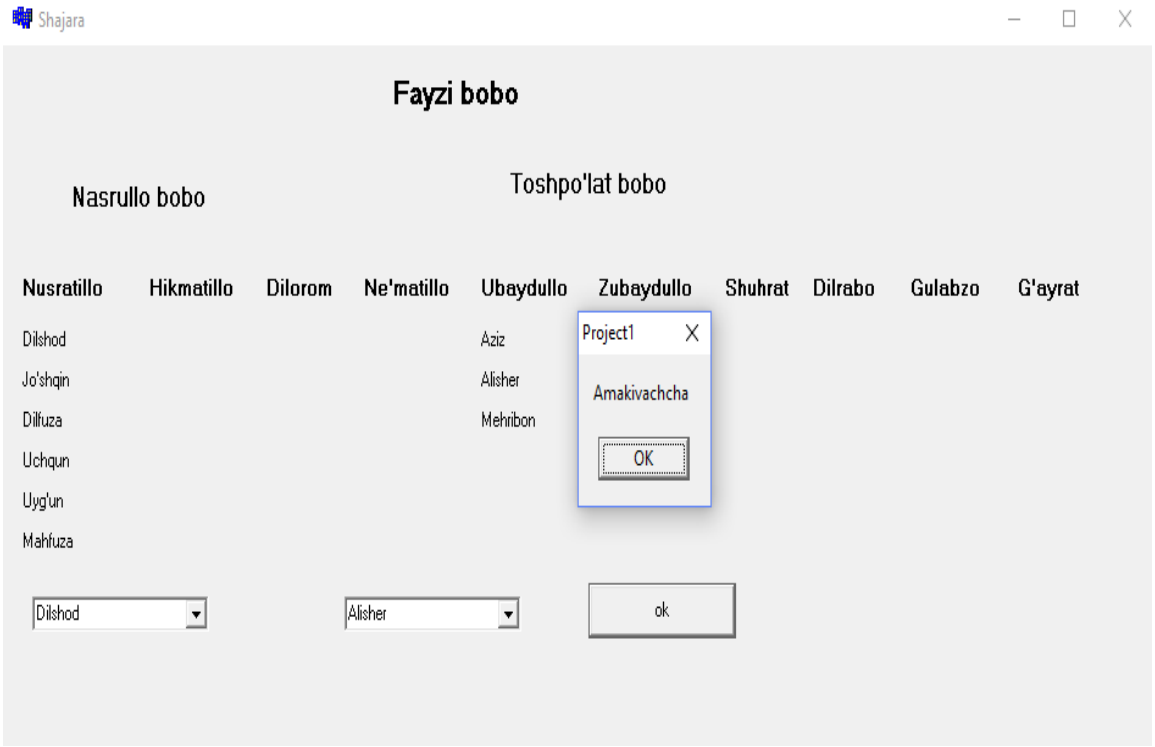
Va ((?x) qush, Emas ((?x) pingvin emas))

Yoki teskarisi:

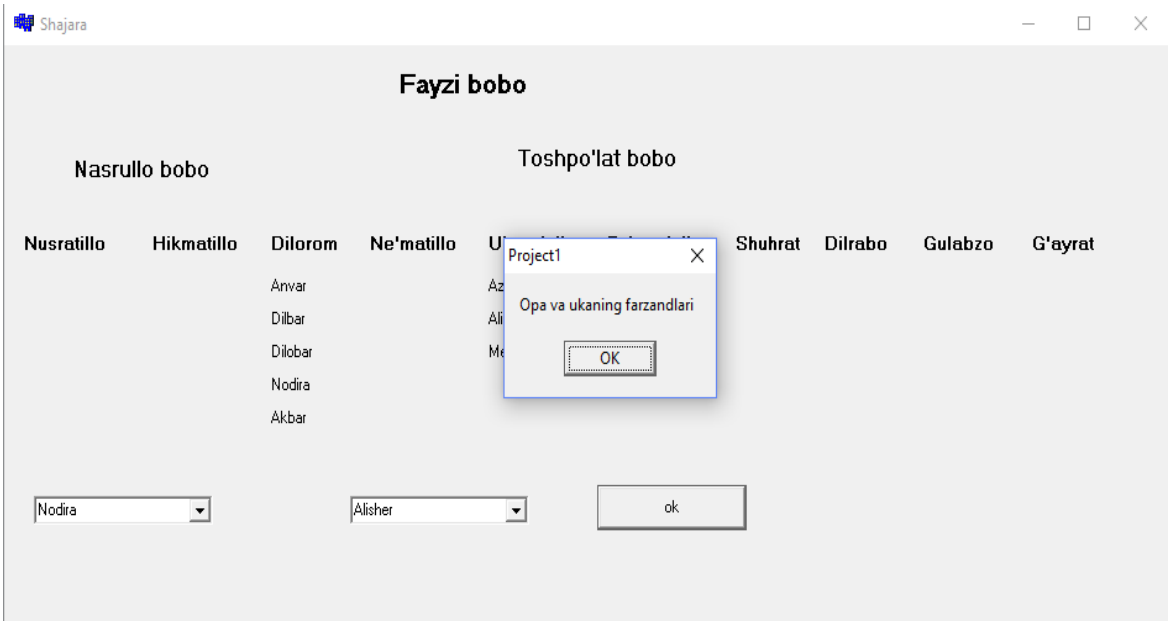
Va (Emas ((?x) pingvin), (?x) qush).

Va quyidagilarni misol qilamiz :

- aka x y: x aka u holda (nima bo`lgan taqdirda ham ularning ota onalari bitta)
- singil x y: x singil u holda (ularning ota –onalari bitta)
- oyi x y: x demak onasi y
- dada x y: x bor dada y
- o`g`il X Y: x demak o`g`li y
- qiz x y: x demak qizi y
- jiyan x y: x va y demak aka(ota - ona x va ota ona y)
- nabira x y: x nabira y



5.1.rasm. Dasturni ochish oynasi



5.2.rasm. Natijani topish

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
void __fastcall TForm1::Label1Click(TObject *Sender)
{
if(Label2->Visible == false){Label2->Visible = true;Label3->Visible = true;}
else
{
Label2->Visible = false;Label3->Visible = false;}}
void __fastcall TForm1::Label2Click(TObject *Sender)
{
if(Label4->Visible == false){
Label4->Visible = true;Label5->Visible = true;Label6->Visible = true;Label7-
>Visible = true;
Label8->Visible = true;Label9->Visible = true;Label10->Visible = true;Label11-
>Visible = true;
Label12->Visible = true;Label13->Visible = true;}
else
{
Label4->Visible = false;Label5->Visible = false;Label6->Visible = false;Label7-
>Visible = false;
Label8->Visible = false;Label9->Visible = false;Label10->Visible =
false;Label11->Visible = false;
Label12->Visible = false;Label13->Visible = false;}}
void __fastcall TForm1::Label4Click(TObject *Sender)
```

```
{
if(Label14->Visible == false){
  Label14->Visible = true;Label15->Visible = true;Label16->Visible =
true;Label17->Visible = true;
Label18->Visible = true;Label19->Visible = true;
}
else
{
Label14->Visible = false;Label15->Visible = false;Label16->Visible =
false;Label17->Visible = false;
Label18->Visible = false;Label19->Visible = false;}}
void __fastcall TForm1::Label5Click(TObject *Sender)
{if(Label20->Visible == false){
  Label20->Visible = true ;  Label21->Visible = true  ;  Label22->Visible = true
;}
else{    Label20->Visible = false ;  Label21->Visible = false  ;  Label22->Visible
= false  ;}}
void __fastcall TForm1::Label6Click(TObject *Sender)
{if(Label23->Visible == false){
   Label23->Visible = true;Label24->Visible = true;Label25->Visible =
true;Label26->Visible = true;
Label27->Visible = true;
}else{
Label23->Visible = false;Label24->Visible = false;Label25->Visible =
false;Label26->Visible = false;
Label27->Visible = false;}}
void __fastcall TForm1::Label7Click(TObject *Sender)
{
if(Label28->Visible == false)
{    Label28->Visible = true;   Label29->Visible = true;
  }  else{    Label28->Visible = false;}   Label29->Visible = false;}
void __fastcall TForm1::Label8Click(TObject *Sender)
{
 if(Label30->Visible == false)
{
   Label30->Visible = true;Label31->Visible = true;Label32->Visible = true;
}else{
Label30->Visible = false;Label31->Visible = false;Label32->Visible = false;}}
void __fastcall TForm1::Label9Click(TObject *Sender)
{ if(Label33->Visible == false){   Label33->Visible = true;}else{
Label33->Visible = false;}       }
void __fastcall TForm1::Label10Click(TObject *Sender)
{ if(Label34->Visible == false)
{   Label34->Visible = true;Label35->Visible = true;Label36->Visible = true;}
```

```
else{Label34->Visible = false;Label35->Visible = false;Label36->Visible = false;}
}void __fastcall TForm1::Label11Click(TObject *Sender)
{
 if(Label37->Visible == false)
{    Label37->Visible = true;Label38->Visible = true;}
else{Label37->Visible = false;Label38->Visible = false;}}
void __fastcall TForm1::Label12Click(TObject *Sender){ if(Label39->Visible ==
false)
{    Label39->Visible = true;Label40->Visible = true;Label41->Visible =
true;Label42->Visible = true;
}else{
Label39->Visible = false;Label40->Visible = false;Label41->Visible =
false;Label42->Visible = false;}}
void __fastcall TForm1::Label13Click(TObject *Sender)
{if(Label43->Visible == false){
   Label43->Visible = true;
Label44->Visible = true;Label45->Visible = true;Label46->Visible =
true;Label47->Visible = true;
}else
{Label43->Visible = false;Label44->Visible = false;Label45->Visible =
false;Label46->Visible = false;
Label47->Visible = false;
}       }void __fastcall TForm1::Button1Click(TObject *Sender)
{if(ComboBox1->Text != ComboBox2->Text){
if((ComboBox1->Text == "Dilshod"||ComboBox1->Text ==
"Jo'shqin"||ComboBox1->Text == "Dilfuza"||
ComboBox1->Text == "Uchqun"||ComboBox1->Text == "Uyg'un"||ComboBox1-
>Text == "Mahfuza" )
&&  (ComboBox2->Text == "Aziz"||ComboBox2->Text ==
"Alisher"||ComboBox2->Text == "Mehribon")
)ShowMessage("Amakivachcha");
if((ComboBox1->Text == "Anvar"||ComboBox1->Text == "Dilbar"||ComboBox1-
>Text == "Dilobar"||
ComboBox1->Text == "Nodira"||ComboBox1->Text == "Akbar")
&&  (ComboBox2->Text == "Aziz"||ComboBox2->Text ==
"Alisher"||ComboBox2->Text == "Mehribon"))
ShowMessage("Opa va ukaning farzandlari");
if((ComboBox1->Text == "Dilshod"||ComboBox1->Text ==
"Jo'shqin"||ComboBox1->Text == "Dilfuza"||
ComboBox1->Text == "Uchqun"||ComboBox1->Text == "Uyg'un"||ComboBox1-
>Text == "Mahfuza" )
&&
  (ComboBox2->Text == "Anvar"||ComboBox2->Text == "Dilbar"||ComboBox2-
>Text == "Dilobar"||
ComboBox2->Text == "Nodira"||ComboBox2->Text == "Akbar")
```

```
)ShowMessage("Amakivachcha");
}
Else
{
  ShowMessage("Siz bir xil kishini tanladingiz!"); } }
```

Nazorat savollari

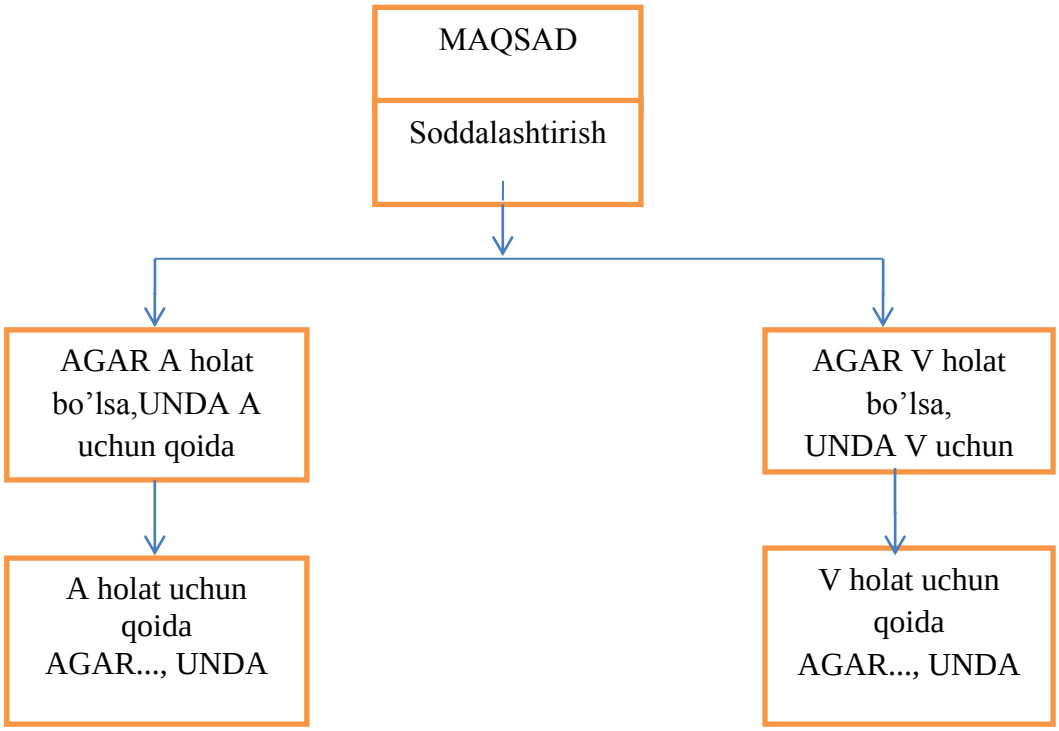
1. Qoidalar tizimi qanday ishlab chiqiladi?
2. Qoidalar tizimini ishlab chiqishda nimalardan foydalaniladi?

6-Amaliy ish.

Mavzu: Teskari zanjir

Ishdan maqsad: Qoidalarning teskari zanjiri tushunchasi va qoidalarning teskari zanjirini ishlab chiqishni o'rganish.

Uslubiy ko'rsatmalar: Sun'iy intellekt haqida gapirish uchun dasturiy tizim inson tomonidan qaror qabul qilish jarayonini tashkil qiluvchi barcha elementlar, maqsadlar, dalillar, qoidalar, mexanizmlar, xulosalar va soddalashtirishga ega bo'lishi kerak. Bunday elementlarning asosiy tarkibiy qismlari 1- rasmda ko'rsatilgan. Sun'iy intellekt tizimining an'anaviy dasturiy tizimlaridan asosiy farqi shundan iboratki, uning tuzilishining tarkibiy qismlari bo'lingan holda aniqlanadi va uning istalgan qismini zamonaviylashtirish umumiy tuzilmaga ta'sir qilmaydi.



6.1.sxema. Sun'iy intellekt tizimining tarkibiy qismlari.

3-amaliy ishda bayon qilingan bosqichlar yordamida maqsadlarga erishish jarayoni mulohazalarning to'g'ridan-to'g'ri zanjiri, ya'ni ma'lumotlardan mantiqiy

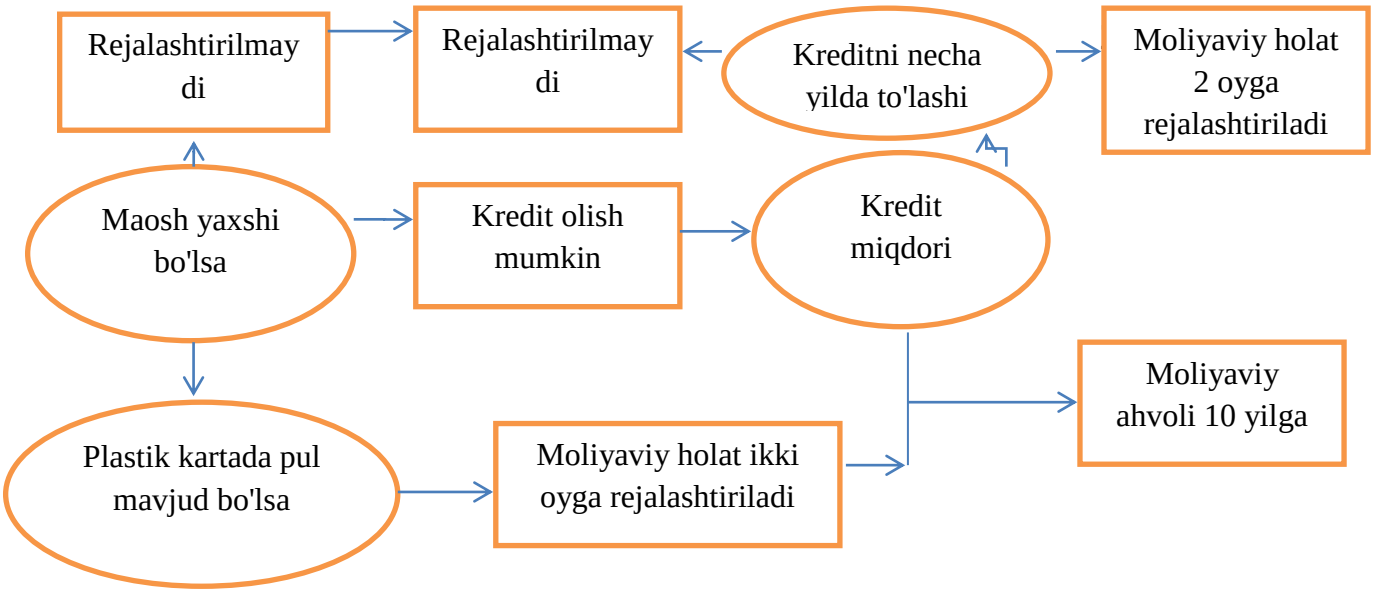
xulosaga boruvchi zanjir deb ataladi. Uni tasdiqlovchi ma’lumotlarni qidirish uchun xulosadan foydalaniladigan jarayon teskari zanjir deb ataladi. Maqsadga erishilganda uning to’g’riligini tekshirish, ya’ni ma’lumotlar va qoidalar bilan yana vazifani tahlil qilish zarurdir. Bu jarayon maqsadlarni verifikatsiyalash deb ataladi. Yana avvaldan to’g’ri deb faraz qilingan xulosani tasdiqlash uchun yangi ma’lumotlarni chiqarish mexanizmlarini tanlash teskari zanjirga misol bo’lishi mumkin. shunday qilib, mulohazalarning teskari zanjiri to’g’ri zanjirga qarama qarshi tomonga, ya’ni xulosadan ma’lumotlarga ketadi. Teskari zanjir faqat maqsadga erishilgandan keyin vujudga keladi.

Inson miyasida soddalashtirish mexanizmi maqsadlarni verifikatsiyalash uchun unga erishishning barcha ehtimol bo’lgan usullari tekshirilguncha qadar qo’shimcha qoidalarni qidirishga rahbarlik qiladi. Sun’iy intellekt tizimining soddalashtirish mexanizmi kompyuterga belgilangan maqsadga erishish uchun bilimlar bazasidan ma’lumotlarning qandaydir qismini ularning muhimligiga ko’ra o’tkazib yuborish va ishlab chiqish imkoniyatini beradi. shunday qilib ham insonning miyasida va ham sun’iy intellekt tizimida soddalashtirish mexanizmi keraksiz va ishga tegishli bo’lmagan mulohazalarni oddiygina e’tiborga olmaydi.

Biror ishni amalga oshirish jarayonini amalga oshirish uchun avval uning qoidalar tizimi ishlab chiqiladi, so'ngra jarayon amalga oshiriladi. Qoidalar tizimi umumiy jarayon tarkibidagi elementar jarayonlar shartlar asosida tekshiriladi va shart natijasi bo'yicha keyingi elementar jarayonga o'tiladi. Qoidalarning teskari zanjirida jarayonlar shartlarga qo'yib tekshiriladi va keyingi etapga o'tishda shu shartga zid keluvchi amalga o'tiladi. Misol uchun biror apparat bor. Unga «apparat uchadimi» degan shart qo'yiladi. Agar «ha» bo'lsa «apparat uchun osmonda parvoz yo'li tashkil etilsin» amali bajariladi, aks holda «apparat uchun yerda yo'l qurilsin» degan amal bajarilsin. Qoidalarning teskari zanjirining asosiy mohiyati shartning amalga oshmaganda sodir bo’luvchi hodisadir Qoidalarning teskari zanjiri ko’pgina sohalarda qo’llaniladi. Misol uchun sun’iy intellektlarni yaratishda. Sun’iy intellektlarning eng ko’p qismi shartlardan iborat bo’ladi.

Qoidalar tizimi :

1. If (maosh(false)) moliyaviy_holat(rejalashtirilmadi).
2. If(maosh(true)) kredit(true).
3. If (maosh(true) && plastic_kartochka(true)) moliyaviy_holat(oy(2));
4. If (kredit(true) && pul()>1000) finance(yil(10))
5. If (kredit(true) && pul()<100 && yil()<2) moliyaviy_holat(yil(10))
6. If (kredit(ha) && pul()<100 && yil()>2) moliyaviy_holat(rejalashtirilmadi);



6.2. sxema. Moliyalashtirish sxemasi

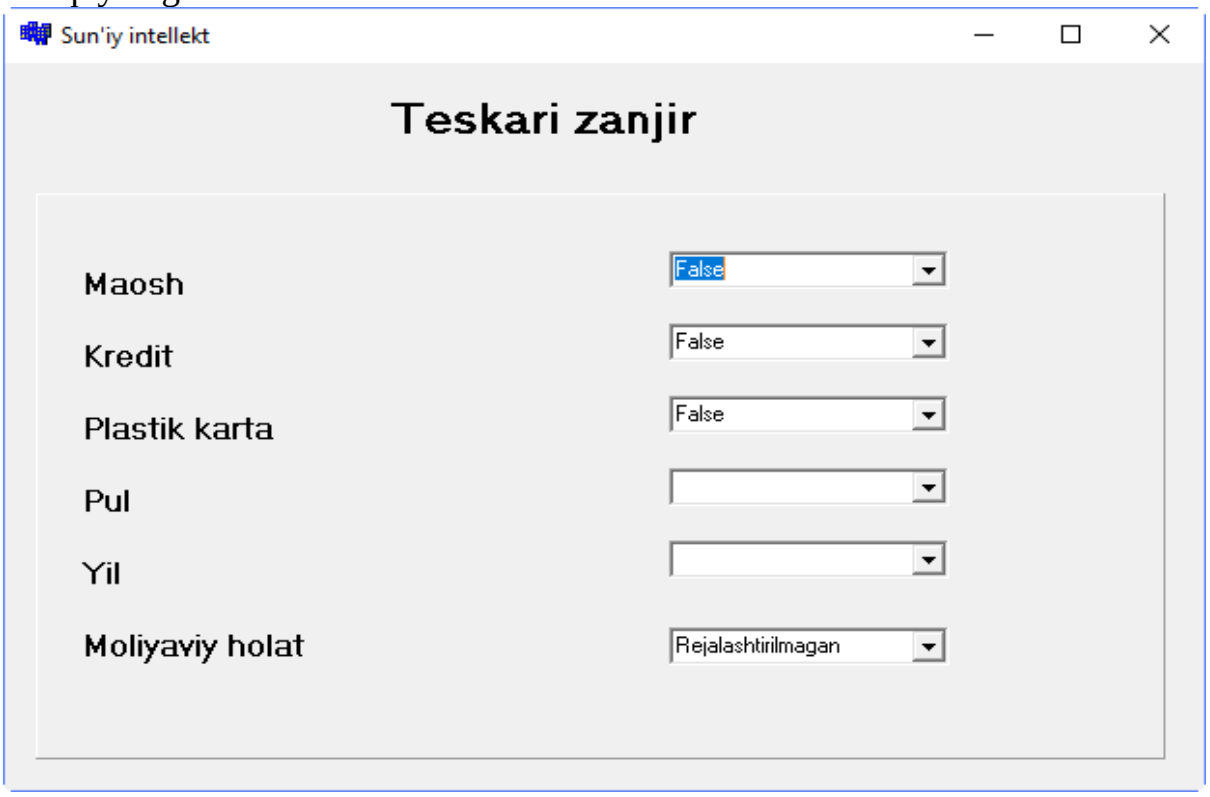
moliyaviy_holat() moliyaviy_holat(),yil(),false<10
moliyav_holat() ,lat()
plastik_karta(), moliyaviy_holat(), Maosh(), kredit(), pul()>1000
true, true, true

Xulosa qilib aytganda Sun’iy intellekt kompyuterni intellektining alomatlari bilan “jihozlashni” ko’zda tutadi. Sun’iy intellekt usullari dasturlarni birlashtirishni soddalashtiradi va tizimga o’z o’zini o’qitish va yangi axborotlarni jamlash qobiliyatini kiritish imkoniyatini beradi. Sun’iy intellekt haqida gapirish uchun dasturiy tizim inson tomonidan qaror qabul qilish jarayonini tashkil qiluvchi barcha elementlar, maqsadlar, dalillar, qoidalar, mexanizmlar, xulosalar va soddalashtirishga ega bo’lishi kerak. Inson miyasida soddalashtirish mexanizmi maqsadlarni verifikatsiyalash uchun unga erishishning barcha ehtimol bo’lgan usullari tekshirilguncha qadar qo’shimcha qoidalarni qidirishga rahbarlik qiladi. Sun’iy intellekt tizimining soddalashtirish mexanizmi kompyuterga belgilangan maqsadga erishish uchun bilimlar bazasidan ma’lumotlarning qandaydir qismini ularning muhimligiga ko’ra o’tkazib yuborish va ishlab chiqish imkoniyatini beradi.

Dasturning asosiy oynasi quyidagi ko`rinishga ega :

6.1.rasm. Teskari zanjir oynasi

Yoki quyidagicha :



6.2.rasm.Tekshirish oynasi

Dastur kodi :

```
/---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
USEFORM("Unit1.cpp", Form1);
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{ try
```

```
        {
                Application->Initialize();
                Application->CreateForm(__classid(TForm1), &Form1);
                Application->Run();
        }
        catch (Exception &exception)
        {
                Application->ShowException(&exception);
        }
        catch (...)
        {
                try
                {
                        throw Exception("");
                }
                catch (Exception &exception)
                {
                        Application->ShowException(&exception);
                }
        }
        return 0;
}
//---------------------------------------------------------------------------v
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "sComboBox"
#pragma link "sPanel"
#pragma link "sSkinManager"
#pragma resource "*.dfm"
TForm1 *Form1;
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}void __fastcall TForm1::sComboBox1Change(TObject *Sender)
{
if ( sComboBox1->ItemIndex ) {
        sComboBox6->ItemIndex = 0;
        sComboBox2->ItemIndex = 1; }
else  {
        sComboBox6->ItemIndex = 1;
        sComboBox2->ItemIndex = 0; }
}
```

```
void __fastcall TForm1::sComboBox4Change(TObject *Sender)
{
if
( sComboBox4->ItemIndex > 2 && sComboBox2->ItemIndex && sComboBox5-
>ItemIndex < 2 ) {
      sComboBox6->ItemIndex = 2; }
else  {
      sComboBox6->ItemIndex = 1;
      sComboBox2->ItemIndex = 0; }
}

void __fastcall TForm1::sComboBox2Change(TObject *Sender)
{
if ( sComboBox2->ItemIndex && sComboBox1->ItemIndex == 0 ) {
      sComboBox6->ItemIndex = 3; }
}
```

Nazotar savollari

1. Teskari zanjir usulida masalalarni yechishni tushuntiring?
2. Teskari zanjir usulida masalalardagi ma’lumotlar o’zaro qanday bog’langan?

7-Amaliy ish.

Mavzu: «Hill Climbing» texnikasi.

Ishdan maqsad: «Hill climbing» texnikasini o'rganish.

Uslubiy ko’rsatmalar: Mantiqiy modellarning asosida rasmiy nazariya tushunchasi yotadi. Mantiqiy modellarda bilimlarning alohida birliklari (dalillar) o’rtasidagi munosabatlar rasmiy nazariyaning sintaktik bilimlari yordamida aks ettiriladi (masalan, predikatlarni hisoblash). Mantiqiydan farqliroq evristik modellar u yoki bu muammoli sohaning o’ziga hos xususiyatlarini uzatuvchi vositalarning turli tuman majmuasiga ega. Buning oqibatida evristik modellar mantiqiylardan ham imkoniyati yoki xuddi shunday aks ettirish, ya’ni muammoli sohani taqdim etish qobiliyati bo’yicha va ham chiqishning foydalanilgan mexanizmining samaradorligi bo’yicha ustivorlik qiladi. Evristik modellar tarmoqli, freymli yoki mahsulotli bo’ladilar. Mantiqiy modellar predikatlarni hisoblash tilidan foydalanadilar. Birinchi predikatga munosabatning nomi mos keladi, dalilning atamasiga esa - ob’ektlar. Predikatlar mantiqida foydalaniladigan barcha mantiqiy iborolar haqiqiy yoki yolg’on ma’noga ega. Masalan “Jon kompyuter bo’yicha mutaxassisdir” iborasini ko’rib chiqamiz. Bu ibora quyidagicha taqdim etilishi mumkin (Jon kompter bo’yicha mutaxassis)dir. Ammo bu ibora quyidagicha interpritatsiyalanishi mumkin: qandaydir X ob’ekti mavjud, u

kompter bo’yicha mutaxassisdir. Bunda yozuvning quyidagi formulasidan foydalaniladi (X, kompter
bo’yicha mutaxassis)dir.

Evristik qidiruvning asosiy g’oyasi shundan iboratki, unda qidiruv jarayonini boshqarish uhcun qo’shimcha axborotlardan foydalaniladi. Evristikada foydalanish masalaning yechimini izlash jarayonida maqsadga tezroq erishishga olib boruvchi ko’rib chiqadigan variantlar sonini qisqartirish imkonini beradi.

Evristik qidiruv algoritmlariida evristik qoidalar asosida shakllantiruvchi cho’qqilar ro’yhati, bir necha baholash funksiyalarining o’sish tartibi bo’yicha tartiblanadi.

Baholash funksiyasi evristik tashkil etuvchisining qiymati qanchalik kichik bo’lsa, ko’rib chiqilayotgan cho’qqi maqsadga chuncha yaqin bo’ladi. Unda “tog’ga ko’tarilish” algoritmini o’z ichiga oladi.

Algoritm maqsadga yo’naltirilgan qidirishni o’z ichiga olib, evristik baholash h’(n) funksiyasining ko’p yo’nalishlarda kamayishini aks ettiradi. h’(n) funksiyaning qiymati qanchalik kam bo’lsa, cho’qqi joylashgan yo’l shunchalik “isdiqbolli” sanaladi.

Funksiyaning maksimumini qidirish eng yuqori cho’qqiga ko’tarilishning optimal yo’lini eslatadi. Shuning uchun ko’rilayotgan algoritm **Hill-climbing** deb ataladi.

Hill climbing dasturlashdagi local qidiruv oilasiga tegishli matematik optimizatsiya texnikasidir. Bu muammoning ixtiyoriy yechimi bilan boshlanuvchi va yechimning yagona elementini qadamba-qadam o’zgartirib yaxshiroq yechim topishga urinib ko’ruvchi takrorlanuvchi algoritmdir.

Masalan, sayohatga chiqish muammosida hill climbing qo’llaniladi. Barcha shaharlarga borishning boshlang’ich yechimni topish oson ammo unda optimal yechim ancha kam. Algoritm yechim bilan boshlanadi va bu yechimga ikki shahardan qaysi biriga borishni tanlash kabi kichik takomillashtirishlar kiritib boriladi. Nihoyat, eng qisqa yo’nalish topilgan hisoblanadi.

Hill climbing local optimum topishga yaxshi ammo barcha barcha yechimlar ichidan eng yaxshisini toppish kafolatlanmagan. Eng ko’zga ko’rinarli muammolarda hill climbing optimal hisoblanadi. Binar qidiruv va chiziqli dasturlashning sodda algoritmlarini o’z ichiga oluvchi ko’zga ko’rinarli muammolarning algoritm namunalarini hal qiladi.

Hill climbing berilgan f(x) funksiyani maksimallashtirishga harakat qiladi. Bu yerda x – to’xtovsiz va/yoki diskret vector. Har bir takrorlanishda hill climbing x dagi yagona elemnti tuzatiladi va f(x)ning qiymatini o’zgartirib aniqlanadi. Hill Climbing bilan f(x)ni yaxshilovchi birorta o’zgartirish qabul qilinadi va jarayon f(x)ni yaxshilovchi birorta o’zgartirish qolmaguncha davom etadi. Shundan so’ng x “local optimal” deb nomlanadi.

Diskret vector fazosida har bir mavjud bo’lishi mumkin bo’lgan x qiymat grafning uchi bo’lib hisoblanadi. Hill climbing x ning local ko’payishi orqali x_m local maksimalga yetgunicha grafning bir uchidan boshqa uchiga harakatlanadi.

Hill climbingning ikki turi mavjud: Stoxastik hill climbing va tasodifiy qayta boshlanuvchi hill climbing. Stoxastik hill climbingda qanday harakatlanishga qaror

qabul qilingunicha barcha qo'shni uchlar tekshirilmaydi. Tasodifiy qayta boshlanuvchi hill climbing hill climbing algoritmining ustida quriluvchi meta-algoritm. Uni yana Shotgun hill climbing nomi bilan ham ma'lum. Bu algoritm har safar o'zlashtirilgan tasodifiy x_0 shart bilan ketma-ketlikda hill climbingni amalga oshiradi. Eng yaxshi x_m saqlab qo'yiladi. Agar yanga qadamda yanayam yaxshi x_m topilsa oldingi saqlangani bilan almashtiriladi.

Algoritmning navbatdagi qadamida h'(n) baholash funksiyasi uchun qoyalar quyidagi qiymatlar n_1, n_2, …n_m va qolgan yo'llar uchun h'(n_i) qiymat belgilanadi. Agar barcha yondosh qoyalar h'(n)>=h(n) qiymatga ega bo'lsa, qidiruv protsedurasi muvaffiqiyatsizlikka uchraydi.

«hill climbing» dasturlashda lokal qidiruv oilasiga kiruvchi matematik optimizatsiyalash texnikasi hisoblash. Bu algoritm muammoni yechishda tasodifiy yechimlarni topish bilan boshlanadi. Keyin esa topilgan yechimlar orasidan optimal yechim qidirib topiladi. Bunda optimal javobning elementi qadamba qadam almashtirib boriladi.

«Hill climbing» metodologiyasi:

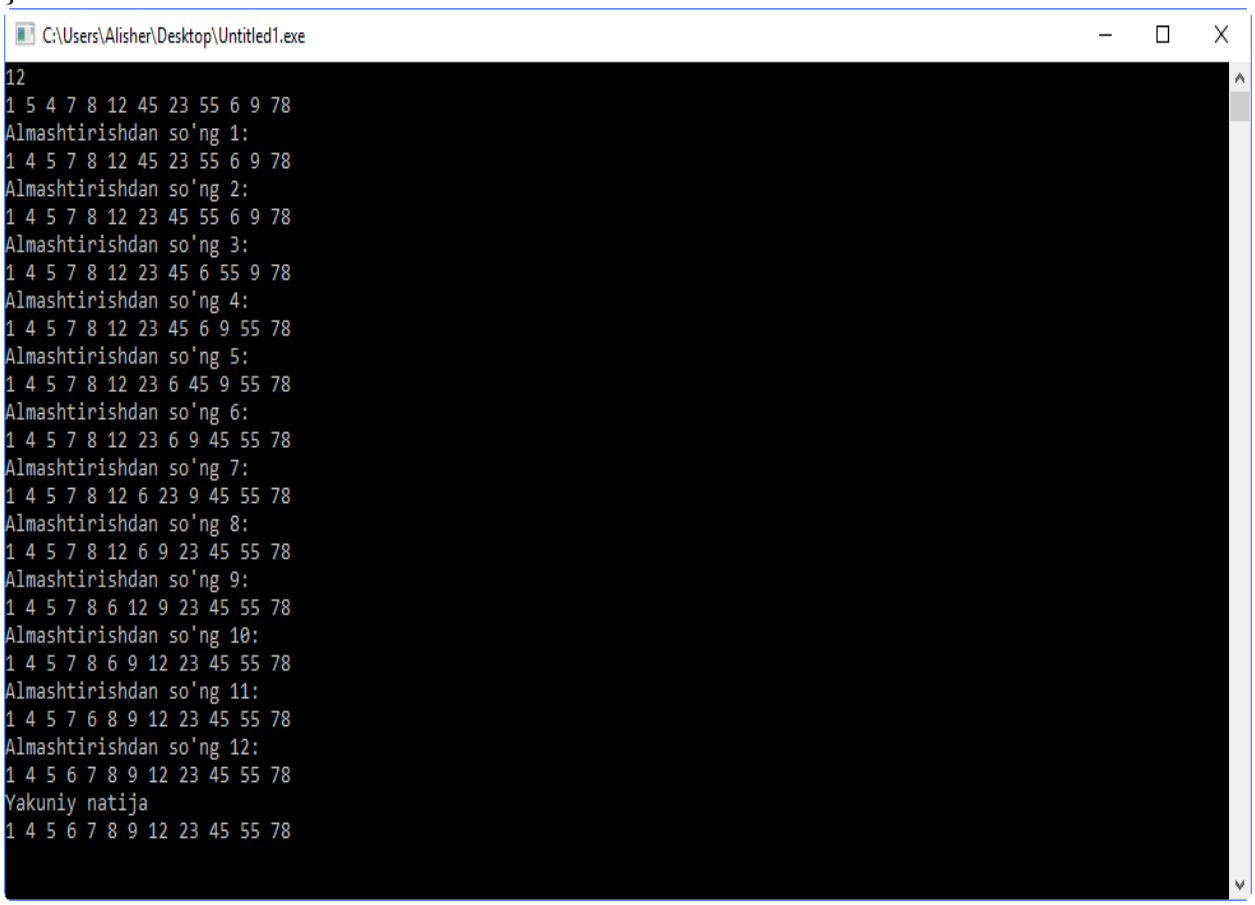
1. Muammoning tuzilmasida uchraydigan ost-optimal yechimlarni qurish.
2. Yechimni olish va uning ustida mukallashtirishni amalga oshirish.
3. Muhim yaxshilanishlar amalga oshirilmaguncha takror takror yechimlarda yaxshilanishlarni amalga oshirish

Hill climbingning eng tanilgan muammolaridan biri tarmoq oqimlari muammosidir. Bundan tashqari hill climbing algoritmi yordamida daraxtlar ustida ham amallar bajarish mumkin.

Dastur:

```
#include<iostream>
#include<conio.h>
#include<cstdio>
using namespace std;
int calcCost(int arr[],int N){
 int c=0;
 for(int i=0;i<N;i++){
  for(int j=i+1;j<N;j++) if(arr[j]<arr[i]) c++; }
 return c;
}void swap(int arr[],int i,int j){
 int tmp=arr[i];
 arr[i]=arr[j];
 arr[j]=tmp;
}
int main(){
 int N;
 scanf("%d",&N);
 int arr[N];
 for(int i=0;i<N;i++) scanf("%d",&arr[i]);
 int bestCost=calcCost(arr,N),newCost,swaps=0;;
```

```
        while(bestCost>0){
         for(int i=0;i<N-1;i++){
          swap(arr,i,i+1);
          newCost=calcCost(arr,N);
          if(bestCost>newCost){
           printf("Almashtirishdan so'ng %d: \n",++swaps);
           for(int i=0;i<N;i++) printf("%d ",arr[i]);
           printf("\n");
           bestCost=newCost; }
          else swap(arr,i,i+1);  } }
        printf("Yakuniy natija\n");
        for(int i=0;i<N;i++) printf("%d ",arr[i]);
        printf("\n");
        getch();
        return 0;
}
```



7.1.rasm. Natija oynasi

Nazorat savollari

1. Evristik qidiruv nima?
2. “Hill Climbing” texnikasini tushuntiring?
3. “Hill climbing” metodologiyasini tushuntiring?

8-Amaliyot ish
Mavzu: Qidiruv Beam search algoritmi

Ishdan maqsad: Bu laboratoriya ishi orqali talabalarda qidiruv Beam search algoritmi haqida ma`lumot hosil qilishdir .

Uslubiy ko'rsatmalar: Qidiruv Beamsearch algoritmini chuqurlik bo`ylab tarqalish deb atashimiz mumkin .Bu atama birinchi marta Raj Reddy tomonidan Carnegi Mllon Universitetida ishlatilgan .Bu qidiruvda har bir usuldan bir birigia o`tish imkoniyatlari mavjud bo`ladi. Beam search eng yaxshi birinchi qidiruv algoritmlari bo`lib xotiradan joy olishi qisqartirib optimallashtiradi . Beam search qidiruv daraxtidan qidirishda foydalaniladi . Uni ko`plab mashina ishini boshqarishda foydalanamiz . Bu algoritmdan foydalanishda biz natijalarni tezda olamiz , chunki natijalar ham o`zaro bir biriga bog`langan bo`ladi. Beam searchni nurli qidiruv deb tarjima qilamiz. Nurli qidiruv optimallashtirilgan "birinchisining eng yaxshisi" algoritmidir. Orginaliga o'xshab u har bir tugunni evristik baholash funksiyasidan foydalanadi. Biroq, faqat har bir o'tishdagi birinchi eng ko'p istiqbolli m tugun baholanadi. Bu yerda m – fiksirlangan son.

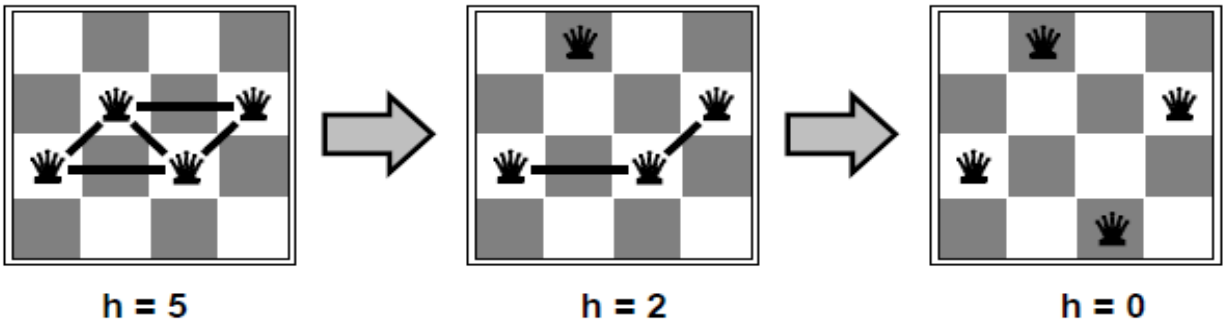
Beam search qidiruv daraxtini quraishda breadth-first search dan foydalanadi. Daraxtning har bir darajasida u holatlarni evristik bahoning o'sish tartibida saralab joriy darajadagi barcha holat davomchilarini ishlab chiqadi. Shunga qaramay, u β – oldindan belgilangan har bir darajadagi eng yaxshi holat nomerini saqlab qo'yadi(nur kengligi deb nomlanadi). Faqat shu holatlar keyingisini kengaytiradi. Eng kata nur kengligi, eng kam holat kesib tashlanadi. "Beam search" atamasini birinchi bo'lib Carnegi Mellon Universitetidan Raj Reddy ishlatgan.

Beam search to'liq qidiruv daraxtini saqlovchi xotiraning nuqsonli yig'iladigan katta tizimlarda itoatkor saqlashda ko'p foydalaniladi. Masalan, u ko'pgina tarjima mashinalarida foydalaniladi. Eng yaxshi tarjimani tanlash uchun har bir qism qo'zg'atiladi va turli xil tarjima yo'llari bilan so'zlar paydo bo'ladi. Ularning gap tuzilishi o'rniga eng yaxshi tarjima qo'shimchasi saqlanadi. Tarjimon keyin berilgan kriteria o'rniga tarjimalarga baho beradi. Beam search birinchi marta 1976-yil Carnegi Mellon Universitetida Harpy nutqni tanish tizimida qo'llanilgan.

Beam search quyidagilarda ishlatiladi:

- Integratsiyalashgan dizayn zanjiri
- Factory-floor layout
- Ishni rejalashtirish
- Tarmoqni optimizatsiyalash
- Transport yo'nalishini aniqlash
- Sayohat uyushtirish muammolarida
- Mashinali tarijama qilishda

Nta qirolichalar masalasi. NxN katakli doskaga Nta qirolichani qator yoki ustun va yoki dioganal bo'yicha 2tadan joylashtirmasdan qo'yamiz.



8.1.rasm. Shaxmat oynasi

O’sish tartibidagi kesishuvchi raqamlar bo’ylab qirolichalarni harakatlantiramiz. Bunda Nta qirolicha masalasi kata N uchun tezda yechiladi.

Beam Search algoritmi

OPEN = {Boshlang’ich holatni berish}

While OPEN bo’sh emas bo’lsa, bajarilsin

OPENdan eng yaxshi tugunni o’chirish

Agar n maqsadli holat bo’lsa, yo’lni nga qaytarish va yo’lni qaytarish

N ning vorislarini yaratish

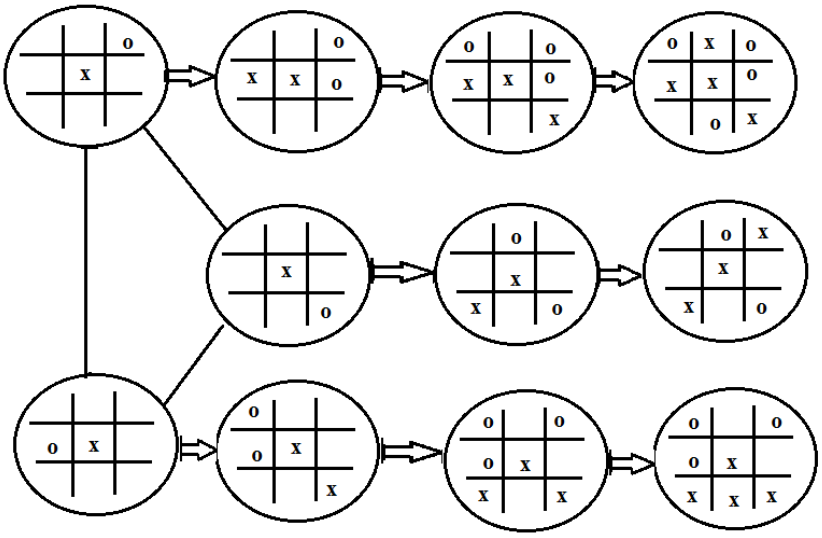
Har bir vorisni baholash, uni OPENga qo’shish, va uning otasini yozib olish

Agar OPEN > ß bo’lsa, eng yaxshi ß tugunlarni olish va qolganlarini OPENdan olib tashlash.

X-o o’yini - 3x3 kvadrat kataklardagi 2ta raqiblar orasidagi mantiqiy o’yin. Bitta o’yinchi x lar bilan ikkinchisi o lar bilan o’ynaydi. An’anaviy xitoy o’yinida qora va oq toshlardan foydalaniladi. Kim o’z belgilari bilan kataklarni gorizontal yoki vertical va yoki x bo’yicha to’ldirsa yutgan hisoblanadi. Har bir tomonda durang qiluvchi yoki raqibini yutishga imkon beruvchi umumiy ma’lum algoritmlar mavjud. Bu o’yinni kompyuterda bajarish uchun mini-maks usuli bilan mos keluvchi o’yin holatlari daraxti quriladi. Bunday daraxtning tugunlari soni 255168taga teng. Bu son barcha mumkin bo’lgan birinchi qadamdagi variantlar soni – 9 taning yig’indisidan olinadi. 2-qadamda 9taning har biri uchun 8 ta variant bo’ladi. 3-qadamda 72ta variantning har biri uchun 7ta variant bo’ladi va hk.

Hamma natijalar ham yaxshi ya`ni kutilganidek bo`lmasligi mumkin . Masalan men tanlab olgan x o o`yinida ham doim g`olib bo`lavaermaydi . Demak doim muvaffaqiyatli natija chiqmasligi mumkin .

“x o ” o`yini uchun kombinatsiyalarning ba`zi birlani ko`rib chiqamiz.



8.2.rasm. Yo’naltiruvchi oyna

Xulosa

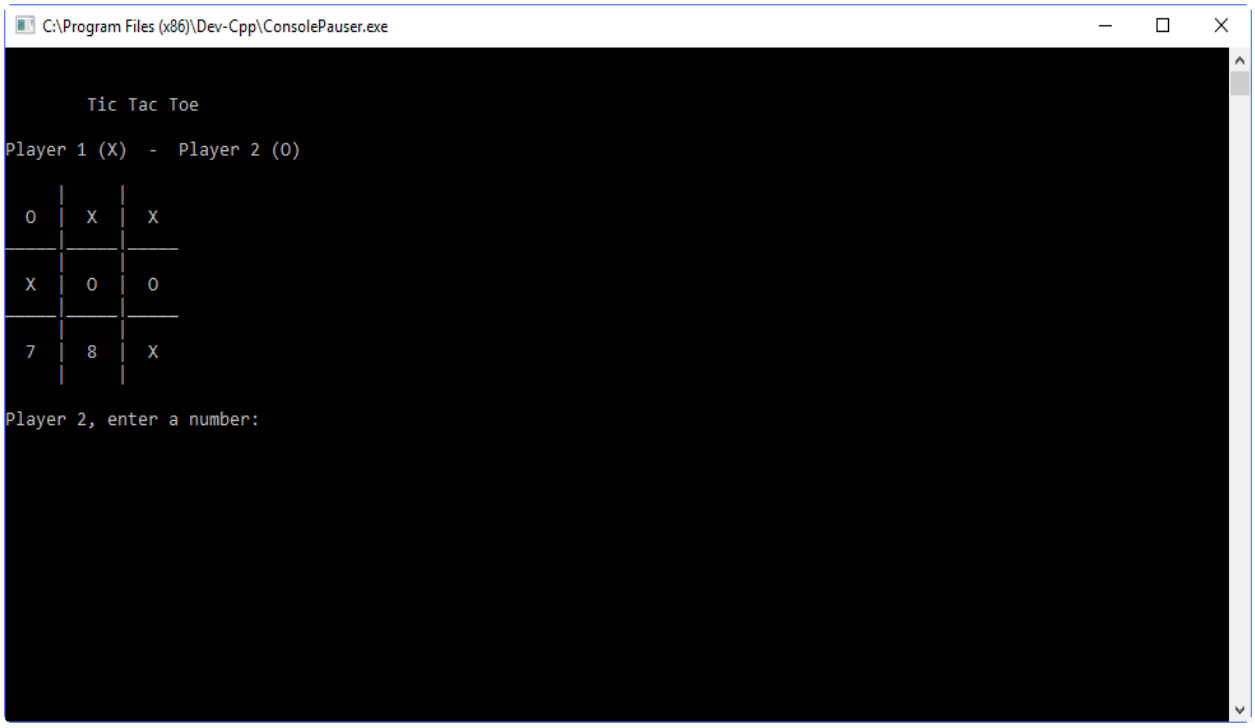
Qidiruv algoritmlarining juda ko`plab turlari mavjud bo`lib , lekin ularning hammasi bitta narsaga natijani olishga qaratilgan . Shu algoritmlaridan biri Beam search hisoblanadi . Xulosa qilib aytadigan bo`lsak bu laboratoriya ishida biz qidiruv algotimining yana bir turi haqida ma`lumotga ega bo`ldik . Bu algoritm orqali natijani tez va qulay usulda olishimiz mumkin.

Dasturi.

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>
char square[10] = {'o','1','2','3','4','5','6','7','8','9'};
int checkwin();
void board();
int main()
{       int player = 1,i,choice;
        char mark;
        system("cls");
        do      {
                board();
                player=(player%2)?1:2;
                std::cout << "Player " << player << ", enter a number:  ";
                std::cin >> choice;
                mark=(player == 1) ? 'X' : 'O';
                if (choice == 1 && square[1] == '1')
                        square[1] = mark;
                else if (choice == 2 && square[2] == '2')
                        square[2] = mark;
                else if (choice == 3 && square[3] == '3')
                        square[3] = mark;
                else if (choice == 4 && square[4] == '4')
```

```
                square[4] = mark;
            else if (choice == 5 && square[5] == '5')
                square[5] = mark;
            else if (choice == 6 && square[6] == '6')
                square[6] = mark;
            else if (choice == 7 && square[7] == '7')
                square[7] = mark;
            else if (choice == 8 && square[8] == '8')
                square[8] = mark;
            else if (choice == 9 && square[9] == '9')
                square[9] = mark;
            else
            {                   std::cout<<"Invalid move ";
                player--;
                getch();            }
            i=checkwin();
            player++;
        }while(i==-1);
        board();
        if(i==1)
            std::cout<<"==>\aPlayer "<<--player<<" win ";
        else
            std::cout<<"==>\aGame draw";
        getch();
        return 0;}
int checkwin()
{       if (square[1] == square[2] && square[2] == square[3])
            return 1;
        else if (square[4] == square[5] && square[5] == square[6])
            return 1;
        else if (square[7] == square[8] && square[8] == square[9])
            return 1;
        else if (square[1] == square[4] && square[4] == square[7])
            return 1;
        else if (square[2] == square[5] && square[5] == square[8])
            return 1;
        else if (square[3] == square[6] && square[6] == square[9])
            return 1;
        else if (square[1] == square[5] && square[5] == square[9])
            return 1;
        else if (square[3] == square[5] && square[5] == square[7])
            return 1;
        else if (square[1] != '1' && square[2] != '2' && square[3] != '3' &&
               square[4] != '4' && square[5] != '5' && square[6] != '6' &&
          square[7] != '7' && square[8] != '8' && square[9] != '9')
```

```
            return 0;
        else
            return -1;
}void board()
{
        system("cls");
        std::cout << "\n\n\tTic Tac Toe\n\n";
        std::cout << "Player 1 (X)  -  Player 2 (O)" << "\n";
        std::cout << "\n";
        std::cout << "     |     |     " << "\n";
        std::cout << "  " << square[1] << "  |  " << square[2] << "  |  " << square[3]
<< "\n";
        std::cout << "_____|_____|_____" << "\n";
        std::cout << "     |     |     " << "\n";
        std::cout << "  " << square[4] << "  |  " << square[5] << "  |  " << square[6]
<< "\n";
        std::cout << "_____|_____|_____" << "\n";
        std::cout << "     |     |     " << "\n";
        std::cout << "  " << square[7] << "  |  " << square[8] << "  |  " << square[9]
<< "\n";
        std::cout << "     |     |     " << "\n" << "\n";
}
```



8.3.rasm. Natija oynasi

Nazorat savollari

1. Beam Search algoritmini tushuntiring?
2. Beam Search algoritmi kim tomonidan yaratilgan?

3. Beam Search algoritmidan qayerlarda foydalaniladi?

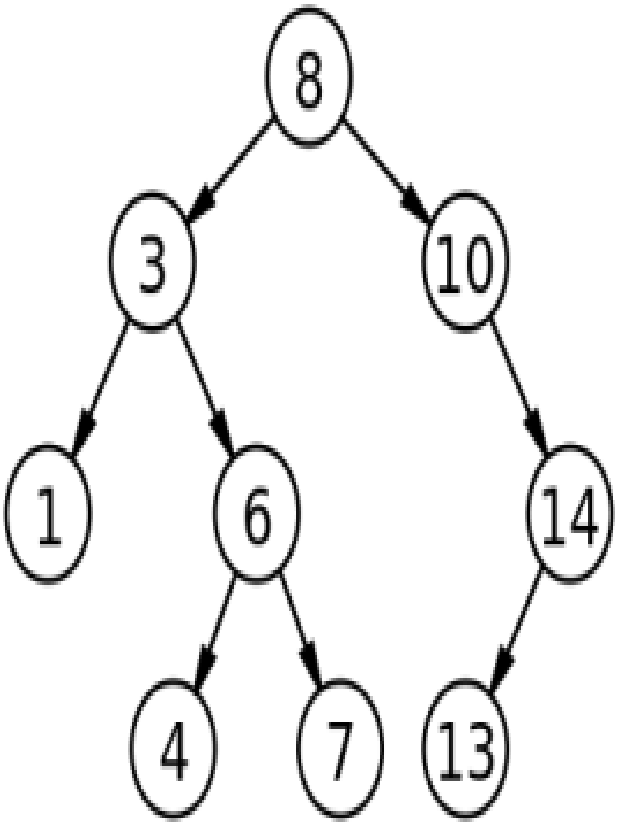
9-Amaliy ish

Mavzu: *Qidiruv "Optimal search"*

Ishdan maqsad: Mantiqiy masalalar uchun Optimal qidiruvni ishlab chiqish.

Uslubiy ko'rsatmalar: Ikkilik qidiruv daraxti yoki "optimal search" bu ikkilik daraxti bo'lib unga quyidagi shartlar qo'yiladi:

- Ikki chap va o'ng ost daraxtlar ikkilik qidiruv daraxti hisoblanadi.
- Barcha X tugundan chiqgan ixtiyoriy chap ichki daraxtlar tugunlari shu tugundagi qiymatlaridan kichik.
- Barcha X tugundan chiqgan ixtiyoriy o'ng ichki daraxtlar tugunlari shu tugundagi qiymatlaridan katta yoki teng.



9.1.rasm. Ikkilik qidiruv daraxti.

Ko'rinib turibdiki har bir tugundagi ma'lumotlar *kichik* solishtirish amali bajariladigan kalitlarga ega bo'lishi kerak.

Har bir tugun taqdim etadigan ma'lumot yagona ma'lumot maydoni emas, balki *yozuv*(yoki class) hisoblanadi. Lekin bu ikkilik qidiruv daraxtining tabiati emas.

Ikkilik qidiruvni amalga oshirish uchun quyidagilarni aniqlash lozim:

- Ikkilik daraxti tugunlardan iborat(qirra) – data, left, right kabi yozuvlarga ega, bu yerda data tugunga bog'langan ma'lumot, left va right tugunning bolalari tugunlarga ko'rsatkich.

- Istalgan X tugun uchun qidiruv daraxti xususiyati amalga oshiriladi: $key\left[left[X]\right] < key[X] \leq key[right[X]]$.

Ikkilik qidiruv daraxtini ikkilik to'plam bilan almashtirib yubormaslik lozim. U boshqa qoidalarga bo'ysungan holda ishlaydi. Ikkilik qidiruv daraxtining boshqa strukturalangan ma'lumotlarni qidirish algoritmlaridan afzal tomonlari shundaki, unda qidiruv va saralash algoritmlari yuqori samaradorlikka ega bo'ladi. Bu algoritm yana to'plamlar, multito'plamlar va asotsiatsiyalangan massivda ham ishlaydi.

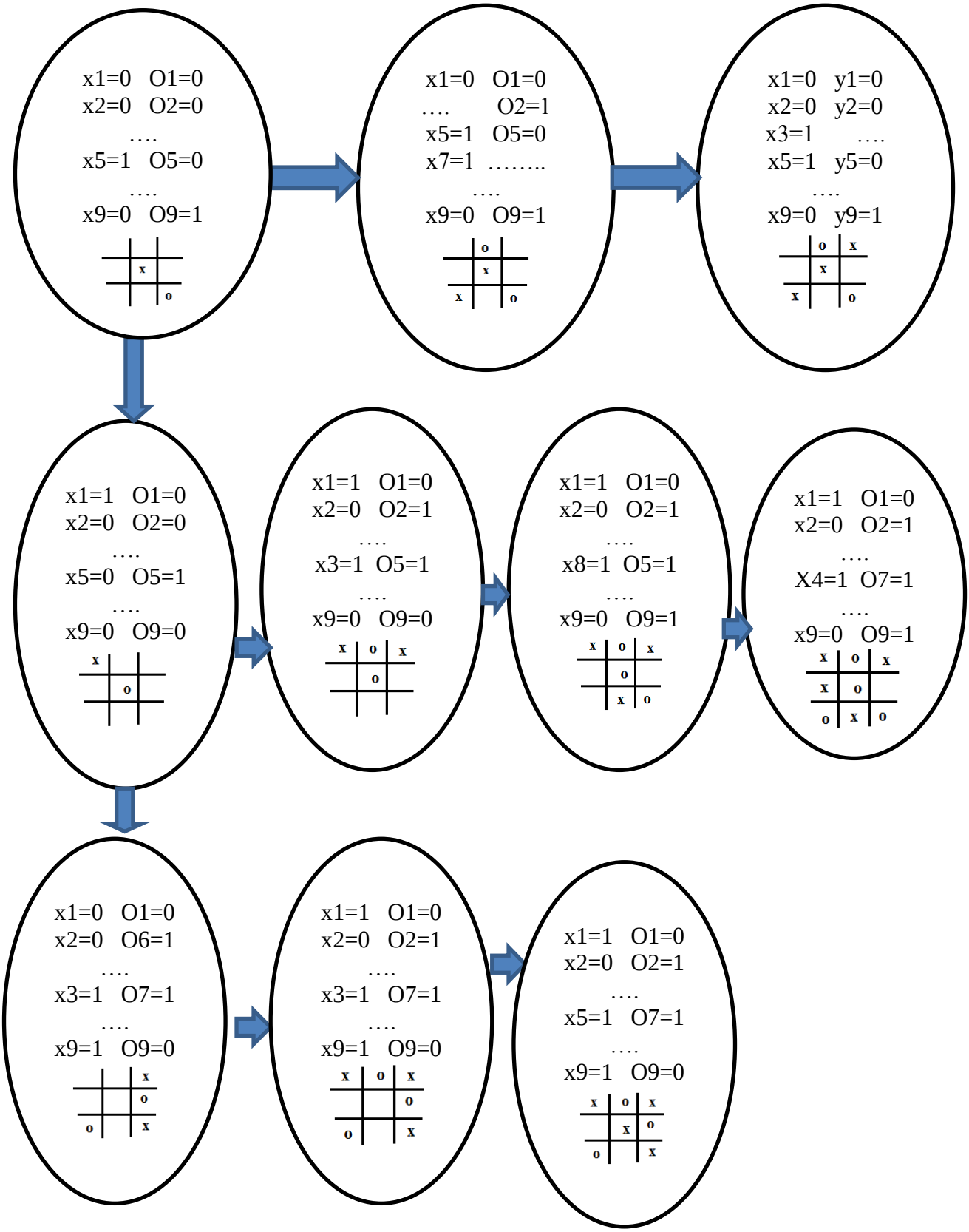
Boshlang'ich holatda ikkilik qidiruv daraxti interfeysi 3 amaldan tashkil topadi:

- FIND(K) – key=K ga teng bo'lgan (key, value) juftliklari saqlanadigan tugunni topish.
- INSERT(K,V) – (key,value)=(K,V) daraxtga juftliklarini qo'shish.
- REMOVE(K)- key=k bo'lgan (key, value) tugunni o'chirish.

X-o o'yini - 3x3 kvadrat kataklardagi 2ta raqiblar orasidagi mantiqiy o'yin. Bitta o'yinchi x lar bilan ikkinchisi o lar bilan o'ynaydi. An'anaviy xitoy o'yinida qora va oq toshlardan foydalaniladi. Kim o'z belgilari bilan kataklarni gorizontal yoki vertical va yoki x bo'yicha to'ldirsa yutgan hisoblanadi. Har bir tomonda durang qiluvchi yoki raqibini yutishga imkon beruvchi umumiy ma'lum algoritmlar mavjud. Bu o'yinni kompyuterda bajarish uchun mini-maks usuli bilan mos keluvchi o'yin holatlari daraxti quriladi. Bunday daraxtning tugunlari soni 255168taga teng. Bu son barcha mumkin bo'lgan birinchi qadamdagi variantlar soni – 9 taning yig'indisidan olinadi. 2-qadamda 9taning har biri uchun 8 ta variant bo'ladi. 3-qadamda 72ta variantning har biri uchun 7ta variant bo'ladi va hk.

Hamma natijalar ham yaxshi ya`ni kutilganidek bo`lmasligi mumkin . Masalan men tanlab olgan x o o`yinida ham doim g`olib bo`lavaermaydi . Demak doim muvaffaqiyatli natija chiqmasligi mumkin .

- "x o " o`yini uchun kombinatsiyalarning ba`zi birlani ko`rib chiqamiz.

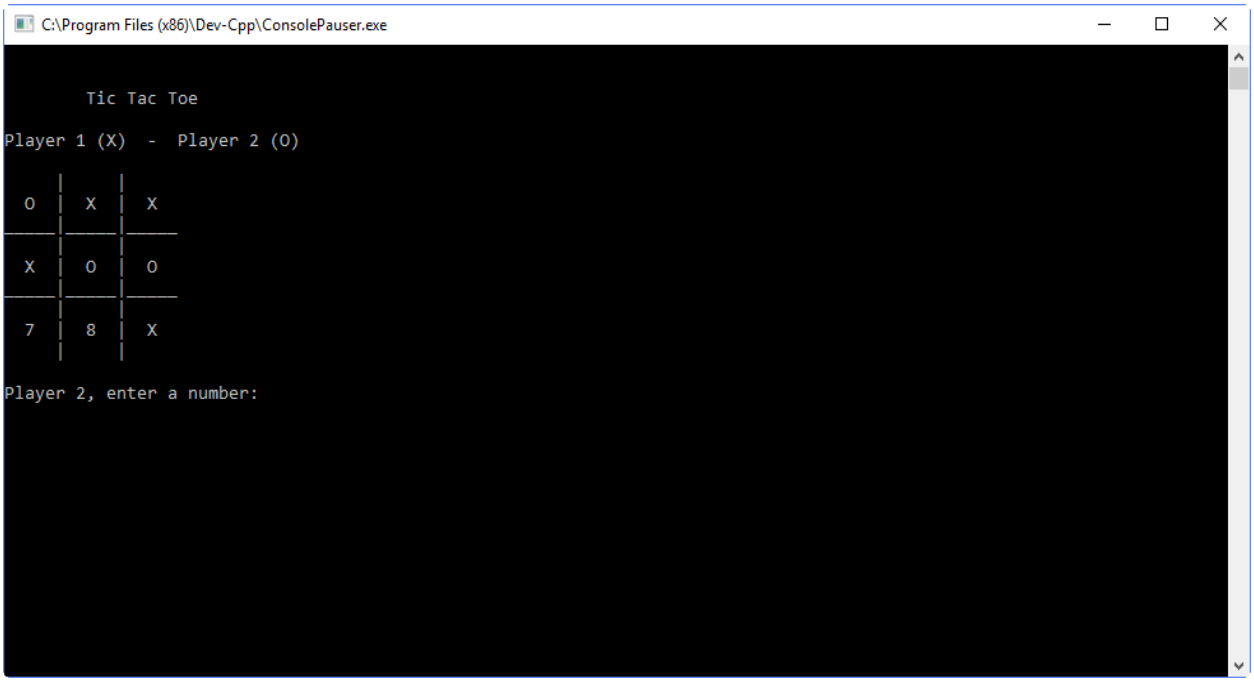


9.2.rasm. O’zgaruvchilarni e’lon qilish

Yechim yo`q
X win Dasturi.
#include<iostream>

```
#include<conio.h>
#include<stdlib.h>
char square[10] = {'o','1','2','3','4','5','6','7','8','9'};
int checkwin();
void board();
int main()
{
	int player = 1,i,choice;
	char mark;
	system("cls");
	do	{
		board();
		player=(player%2)?1:2;
		std::cout << "Player " << player << ", enter a number:  ";
		std::cin >> choice;
		mark=(player == 1) ? 'X' : 'O';
		if (choice == 1 && square[1] == '1')
			square[1] = mark;
		else if (choice == 2 && square[2] == '2')
			square[2] = mark;
		else if (choice == 3 && square[3] == '3')
			square[3] = mark;
		else if (choice == 4 && square[4] == '4')
			square[4] = mark;
		else if (choice == 5 && square[5] == '5')
			square[5] = mark;
		else if (choice == 6 && square[6] == '6')
			square[6] = mark;
		else if (choice == 7 && square[7] == '7')
			square[7] = mark;
		else if (choice == 8 && square[8] == '8')
			square[8] = mark;
		else if (choice == 9 && square[9] == '9')
			square[9] = mark;
		else
		{			std::cout<<"Invalid move ";
			player--;
			getch();		}
		i=checkwin();
		player++;
	}while(i==-1);
	board();
	if(i==1)
		std::cout<<"==>\aPlayer "<<--player<<" win ";
	else
```

```
                std::cout<<"==>\aGame draw";
        getch();
        return 0;}
int checkwin()
{       if (square[1] == square[2] && square[2] == square[3])
                return 1;
        else if (square[4] == square[5] && square[5] == square[6])
                return 1;
        else if (square[7] == square[8] && square[8] == square[9])
                return 1;
        else if (square[1] == square[4] && square[4] == square[7])
                return 1;
        else if (square[2] == square[5] && square[5] == square[8])
                return 1;
        else if (square[3] == square[6] && square[6] == square[9])
                return 1;
        else if (square[1] == square[5] && square[5] == square[9])
                return 1;
        else if (square[3] == square[5] && square[5] == square[7])
                return 1;
        else if (square[1] != '1' && square[2] != '2' && square[3] != '3' &&
                square[4] != '4' && square[5] != '5' && square[6] != '6' &&
          square[7] != '7' && square[8] != '8' && square[9] != '9')
                return 0;
        else
                return -1;
}void board()
{
        system("cls");
        std::cout << "\n\n\tTic Tac Toe\n\n";
        std::cout << "Player 1 (X)  -  Player 2 (O)" << "\n";
        std::cout << "\n";
        std::cout << "     |     |     " << "\n";
        std::cout << "  " << square[1] << "  |  " << square[2] << "  |  " << square[3]
<< "\n";
        std::cout << "_____|_____|_____" << "\n";
        std::cout << "     |     |     " << "\n";
        std::cout << "  " << square[4] << "  |  " << square[5] << "  |  " << square[6]
<< "\n";
        std::cout << "_____|_____|_____" << "\n";
        std::cout << "     |     |     " << "\n";
        std::cout << "  " << square[7] << "  |  " << square[8] << "  |  " << square[9]
<< "\n";
        std::cout << "     |     |     " << "\n" << "\n";
}
```



9.3.rasm. Natijalarni olish

Nazorat savollari

1. Ikkilik daraxt nima?
2. Qidiruv daraxtini tushuntiring?
3. Kichik solishtirishlar amali haqida gapirib bering?

10-Amaliyot ish

Mavzu: O'yinlarda to'rtliklarni birlashtirish (marge sort)

Ishdan maqsad: Marge sort, ya'ni birlashtirish algoritmining ishlash prinsipini ko'rib chiqish va uni o'yinlarning realizatsiyasida qo'llanilishini ko'rib chiqish.

Uslubiy ko'rsatmalar: Merge sort (birlashtirgan holda tartiblash) – oqimdan kelgan ma'lumotlar ro'yhatini tartiblash va aniq bir tartibda tez saralash uchun ishlatiladi. Misol uchun buni "ajrat va boshqar" prinsipida ishlashini aytib o'tishimiz mumkin. Boshlanishda masala bir necha kichik masalalarga bo'linadi. Har bir misolda ajaratilgan bo'lim rekursiv funksiyalar yordamida qayta ishlanadi bu saralash vaqtining qisqarishiga olib keladi

Bunda saralashga ketgan yaxshi vaqt O(n log n) ga teng, yomon vaqt esa O(n log n) odanta O(n). Bu oddiy saralashdan ancha samara hisoblanadi, lekin optimallik darajasi u qadar yuqori emas.

Bu algoritmning boshqacha nomi "ulash orqali" saralash. Bu saralash uch tartibda amalga oshiriladi:

1. Tartiblanayotgan massivi taqriban bir-biroviga teng bo'lgan ikki qismga ajratiladi;
2. Har bir olingan qismlar shu algoritm asosida alohida tartibga solinadi;
3. Ikki tartiblangan massiv bittaga birlashtiriladi.

1.1-2.1 Rekursiv bo’laklash massivning o’lchami birga teng bo’luncha davom etadi.

3.1 Ikki tartiblangan massiv bittaga birlashtiriladi. Massvilarni birlashtirishni quyidagi misolda ko’rib o’tamiz. Tassavur qilaylik o’sish tartibida tartiblangan ikki massivga egamiz. U holda:

3.2 Ikki massivni uchinchi massivga birlashtirish. Har bir qadamda biz ulardan kichigini olib natijaviy massivga yozib qo’yamiz. Natijaviy massivni bittaga oshiramiz.

3.3 Qoldiqni “ulash”.

C++ tilida algoritmning amalda qo’llanilishi.

```
template<typename T>
void MergeSort(T a[], size_t l)
{       size_t BlockSizeIterator;
        size_t BlockIterator;
        size_t LeftBlockIterator;
        size_t RightBlockIterator;
        size_t MergeIterator;
        size_t LeftBorder;
        size_t MidBorder;
        size_t RightBorder;
        for (BlockSizeIterator = 1; BlockSizeIterator < l; BlockSizeIterator *= 2)
        {
                for (BlockIterator = 0; BlockIterator < l - BlockSizeIterator;
BlockIterator += 2 * BlockSizeIterator)
                {
                        LeftBlockIterator = 0;
                        RightBlockIterator = 0;
                        LeftBorder = BlockIterator;
                        MidBorder = BlockIterator + BlockSizeIterator;
                        RightBorder = BlockIterator + 2 * BlockSizeIterator;
                        RightBorder = (RightBorder < l) ? RightBorder : l;
                        int* SortedBlock = new int[RightBorder - LeftBorder];

                        while (LeftBorder + LeftBlockIterator < MidBorder &&
MidBorder + RightBlockIterator < RightBorder)
                        {
                                if (a[LeftBorder + LeftBlockIterator] < a[MidBorder +
RightBlockIterator])
                                {
                                        SortedBlock[LeftBlockIterator +
RightBlockIterator] = a[LeftBorder + LeftBlockIterator];
                                        LeftBlockIterator += 1;
                                }
                                else
                                {
```

```
                                        SortedBlock[LeftBlockIterator +
RightBlockIterator] = a[MidBorder + RightBlockIterator];
                                        RightBlockIterator += 1;
                                }
                        }
                        while (LeftBorder + LeftBlockIterator < MidBorder)
                        {
                                SortedBlock[LeftBlockIterator + RightBlockIterator] =
a[LeftBorder + LeftBlockIterator];
                                LeftBlockIterator += 1;
                        }
                        while (MidBorder + RightBlockIterator < RightBorder)
                        {
                                SortedBlock[LeftBlockIterator + RightBlockIterator] =
a[MidBorder + RightBlockIterator];
                                RightBlockIterator += 1;
                        }

                        for (MergeIterator = 0; MergeIterator < LeftBlockIterator +
RightBlockIterator; MergeIterator++)
                        {
                                a[LeftBorder + MergeIterator] =
SortedBlock[MergeIterator];
                        }
                        delete SortedBlock;
                }
        }
}
```

Algoritmni suniy intellektda ishlatgan holda o’yin yaratish.

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
int i;
void __fastcall TForm1::Button1Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button16Click(TObject *Sender)
{
```

```
if(Button17->Visible==0){
   Button17->Caption=Button16->Caption;
   Button17->Visible=1;    Button16->Visible=0;}
if(Button12->Visible==0){
   Button12->Caption=Button16->Caption;
   Button12->Visible=1;    Button16->Visible=0;}
if(Button15->Visible==0){
   Button15->Caption=Button16->Caption;
   Button15->Visible=1;    Button16->Visible=0;}
}void __fastcall TForm1::Button17Click(TObject *Sender)
{if(Button16->Visible==0){
   Button16->Caption=Button17->Caption;
   Button16->Visible=1;    Button17->Visible=0;}
if(Button13->Visible==0){
   Button13->Caption=Button17->Caption;
   Button13->Visible=1;    Button17->Visible=0;}
if(Button2->Caption=='1' &&   Button3->Caption=='2' && Button4->Caption=='3'
&&   Button5->Caption=='4' &&   Button6->Caption=='5' &&
  Button7->Caption=='6' &&Button8->Caption=='7' && Button9->Caption=='8'
&&   Button10->Caption=='9' && Button11->Caption=="10" &&
  Button12->Caption=="11" &&   Button14->Caption=="13" &&
  Button15->Caption=="14" )
     ShowMessage("Yutting kunti");
}void __fastcall TForm1::Button13Click(TObject *Sender)
{if(Button17->Visible==0){
   Button17->Caption=Button13->Caption;
   Button17->Visible=1;    Button13->Visible=0;}
if(Button12->Visible==0){
   Button12->Caption=Button13->Caption;
   Button12->Visible=1;    Button13->Visible=0;}
if(Button9->Visible==0){
   Button9->Caption=Button13->Caption;
   Button9->Visible=1;    Button13->Visible=0;}
}void __fastcall TForm1::Button15Click(TObject *Sender)
{if(Button16->Visible==0){
   Button16->Caption=Button15->Caption;
   Button16->Visible=1;    Button15->Visible=0;}
if(Button11->Visible==0){
   Button11->Caption=Button15->Caption;
   Button11->Visible=1;    Button15->Visible=0;}
if(Button14->Visible==0){
   Button14->Caption=Button15->Caption;
   Button14->Visible=1;    Button15->Visible=0;}
}void __fastcall TForm1::Button14Click(TObject *Sender)
{if(Button15->Visible==0){
```

```
   Button15->Caption=Button14->Caption;
   Button15->Visible=1;    Button14->Visible=0;}
if(Button10->Visible==0){
   Button10->Caption=Button14->Caption;
   Button10->Visible=1;    Button14->Visible=0;}
}void __fastcall TForm1::Button10Click(TObject *Sender)
{if(Button14->Visible==0){
   Button14->Caption=Button10->Caption;
   Button14->Visible=1;    Button10->Visible=0;}
if(Button11->Visible==0){
   Button11->Caption=Button10->Caption;
   Button11->Visible=1;    Button10->Visible=0;}
if(Button6->Visible==0){    Button6->Caption=Button10->Caption;
   Button6->Visible=1;    Button10->Visible=0;}}
void __fastcall TForm1::Button11Click(TObject *Sender)
{if(Button15->Visible==0){    Button15->Caption=Button11->Caption;
   Button15->Visible=1;    Button11->Visible=0;}
if(Button12->Visible==0){    Button12->Caption=Button11->Caption;
   Button12->Visible=1;    Button11->Visible=0;}
if(Button10->Visible==0){    Button10->Caption=Button11->Caption;
   Button10->Visible=1;    Button11->Visible=0;}
if(Button7->Visible==0){    Button7->Caption=Button11->Caption;
   Button7->Visible=1;    Button11->Visible=0;}
}void __fastcall TForm1::Button12Click(TObject *Sender)
{if(Button8->Visible==0){
   Button8->Caption=Button12->Caption;    Button8->Visible=1;    Button12-
>Visible=0;}
if(Button11->Visible==0){    Button11->Caption=Button12->Caption;
   Button11->Visible=1;    Button12->Visible=0;}
if(Button13->Visible==0){    Button13->Caption=Button12->Caption;
   Button13->Visible=1;    Button12->Visible=0;}
if(Button16->Visible==0){    Button16->Caption=Button12->Caption;
   Button16->Visible=1;    Button12->Visible=0;}
}void __fastcall TForm1::Button6Click(TObject *Sender)
{if(Button10->Visible==0){    Button10->Caption=Button6->Caption;
   Button10->Visible=1;    Button6->Visible=0;}
if(Button2->Visible==0){    Button2->Caption=Button6->Caption;
   Button2->Visible=1;    Button6->Visible=0;}
if(Button7->Visible==0){    Button7->Caption=Button6->Caption;
   Button7->Visible=1;    Button6->Visible=0;}
}void __fastcall TForm1::Button7Click(TObject *Sender)
{if(Button3->Visible==0){    Button3->Caption=Button7->Caption;
   Button3->Visible=1;    Button7->Visible=0;}
if(Button6->Visible==0){    Button6->Caption=Button7->Caption;
   Button6->Visible=1;    Button7->Visible=0;}
```

```
if(Button8->Visible==0){    Button8->Caption=Button7->Caption;
   Button8->Visible=1;    Button7->Visible=0;}
if(Button11->Visible==0){    Button11->Caption=Button7->Caption;
   Button11->Visible=1;    Button7->Visible=0;}
}void __fastcall TForm1::Button8Click(TObject *Sender)
{if(Button4->Visible==0){    Button4->Caption=Button8->Caption;
   Button4->Visible=1;    Button8->Visible=0;}
if(Button7->Visible==0){    Button7->Caption=Button8->Caption;
   Button7->Visible=1;    Button8->Visible=0;}
if(Button9->Visible==0){    Button9->Caption=Button8->Caption;
   Button9->Visible=1;    Button8->Visible=0;}
if(Button12->Visible==0){    Button12->Caption=Button8->Caption;
   Button12->Visible=1;
   Button8->Visible=0;}}void __fastcall TForm1::Button9Click(TObject *Sender)
{if(Button5->Visible==0){
   Button5->Caption=Button9->Caption;
   Button5->Visible=1;
   Button9->Visible=0;}
if(Button8->Visible==0){
   Button8->Caption=Button9->Caption;
   Button8->Visible=1;
   Button9->Visible=0;}
if(Button13->Visible==0){
   Button13->Caption=Button9->Caption;
   Button13->Visible=1;
   Button9->Visible=0;}
}
void __fastcall TForm1::Button18Click(TObject *Sender)
{ AnsiString c;
if(Button17->Visible==0)
switch(rand()%6)
    {case 1: c=Button2->Caption;
           Button2->Caption=Button6->Caption;
           Button6->Caption=c;          c=Button3->Caption;
           Button3->Caption=Button7->Caption;
           Button7->Caption=c;          c=Button4->Caption;
           Button4->Caption=Button8->Caption;
           Button8->Caption=c;          c=Button5->Caption;
           Button5->Caption=Button9->Caption;
           Button9->Caption=c;       break;
     case 2:c=Button6->Caption;
           Button6->Caption=Button10->Caption;
           Button10->Caption=c;          c=Button7->Caption;
           Button7->Caption=Button11->Caption;
           Button11->Caption=c;          c=Button8->Caption;
```

```
        Button8->Caption=Button12->Caption;
        Button12->Caption=c;          c=Button9->Caption;
        Button9->Caption=Button13->Caption;
        Button13->Caption=c;        break;
    case 3:c=Button10->Caption;
        Button10->Caption=Button14->Caption;
        Button14->Caption=c;          c=Button11->Caption;
        Button11->Caption=Button15->Caption;
        Button15->Caption=c;        break;
    case 4:c=Button2->Caption;
        Button2->Caption=Button3->Caption;
        Button3->Caption=c;          c=Button6->Caption;
        Button6->Caption=Button7->Caption;
        Button7->Caption=c;          c=Button10->Caption;
        Button10->Caption=Button11->Caption;
        Button11->Caption=c;          c=Button14->Caption;
        Button14->Caption=Button15->Caption;
        Button15->Caption=c;        break;
    case 5:c=Button3->Caption;
        Button3->Caption=Button4->Caption;
        Button4->Caption=c;          c=Button7->Caption;
        Button7->Caption=Button8->Caption;
        Button8->Caption=c;          c=Button11->Caption;
        Button11->Caption=Button12->Caption;
        Button12->Caption=c;          c=Button15->Caption;
        Button15->Caption=Button16->Caption;
        Button16->Caption=c;        break;
    case 6:c=Button4->Caption;
        Button4->Caption=Button5->Caption;
        Button5->Caption=c;          c=Button8->Caption;
        Button8->Caption=Button9->Caption;
        Button9->Caption=c;        break;    } else ShowMessage("Istimos bo'sh
katakni 4x4 koordinataga ko'chiring!");}
```

10.1.rasm. Tog'ri oyna

10.2.rasm. Teskari oyna.

Nazotar savollari

1. Marge sort algoritmini tushuntiring?
2. Marge sort da amal bajarishga qancha vaqt ketadi?
3. Ushbu algoritm necha bosqichda amalga oshiriladi?

11-Amaliy ish.

Mavzu:O'yinlarda α va β izlash

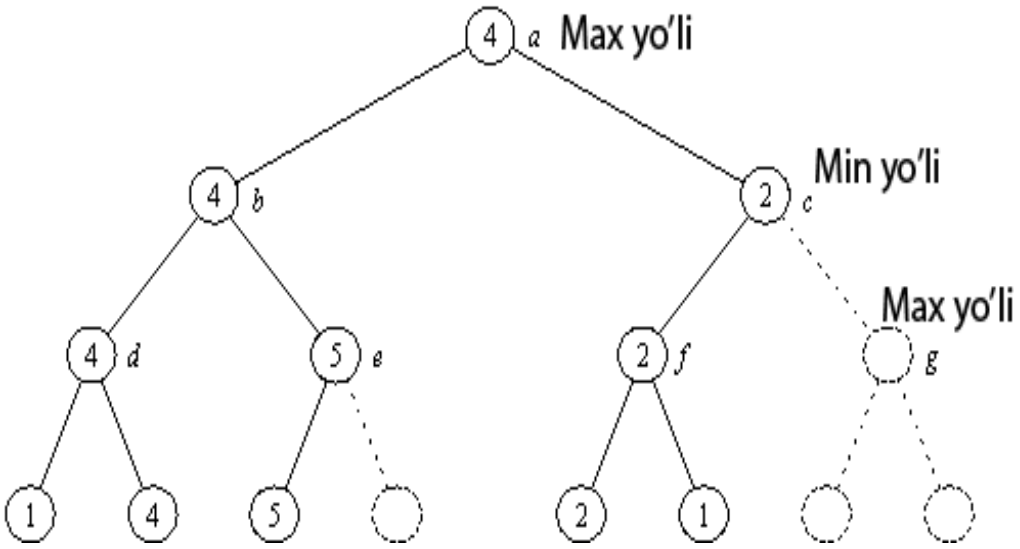
Ishdan maqsad: Ushbu amaliy ishi natijasida o'yinlarning algoritmini tuzishda foydalaniladigan α va β izlash usulini ko'rib chiqish va amaliyotda qo'llay olish.

Uslubiy ko'rsatmalar: α va β izlash usulini – bu o'yin davomida qadamlarning sonini kamaytirish, yani avvaldan qo'yilishi mumkin bo'lgan ammallarning sonini aniqlab berish uchun qo'llaniladi. Bu optimallashtirish ham daraxtlar va graflar bilan bog'liq bo'lib ularning optimallashgan ko'rinishi hisoblanadi.

Alfa-beta kesish algoritmi minimaks algoritmi daraxtida baholangan tugunlarni qisqartirish uchun ishlatiladigan qidiruv algoritmi bo'lib, u antagonik o'yinlar va mashina o'yinlari uchun ishlatiladi. Bu algoritmning asl ma'nosi shundaki, qidiruv daraxtning baholanayotgan tarmog'i baho ffunksiyasi har qanday holda ham oldingi tarmoqdagidan yomon bo'lsa u holda qidiruv to'xtatiladi.

Bu algoritm mustaqil ravishda bir necha olimlar tomonidan bir-biridan behabar holda yaratilgan. Uni ilk marta "Aproksimatsiya" deb nomlashgan. 1956-yilda Makkarti ismli olim Darmurd seminarida bu g'oya haqida aytib o'tgan. 1963-yilda Aleksandr Brudno mustaqil ravishda algoritmni yaratdi va uni ommaga havola etdgan. 1975-yilda esa Donald Kunt va Ronald Mur algoritmga "beta"-kesishishni qo'shib mukammallashtirishdi.

Shaxmat kabi o’yinlarda yo’lni topish uchun kompyuterda juda katta mumkin bo’lgan yo’llarni ko’rib chiqishga to’g’ri keladi. Bu jarayonni ma’lumotlarni yo’qotmasdan tezlashtirish uchun odatda alfa-beta kesish algoritmi ishlatiladi.



11.1.rasm. Yo’lni topish

Misol qilaylik bizda ikkita yo’l varianti bor. Agar biz ulardan biri ikkinchisidan yomonligini bilsak, uni qanchalik yomonroq yo’l ekanini bilishimiz shart emas. Buni qidiruv daraxtida ko’ramiz(11.1-rasm). Qidiruv quyidagicha amalga oshirilaid:

1) a nuqtadan boshlaymiz
2) b nuqtaga o’tamiz
3) d nuqtaga o’tamiz
4) d ning vorislaridan eng katta baholisini olamiz, V(d) = 4
5) b nuqtaga qaytamiz va e ga o’tamiz
6) e pozitsiyaning bahosi 5 ga teng bo’lgan 1-vorisini ko’rib chiqamiz. Bu vaqtda e pozitsiyada bahosi 5 dan kam bo’lmasligi kafolatlangan Maks aniqlanadi. Bu Min e pozitsiyaning bahosini bilmagan holda u uchun b yo’l e da d ga nisbatan yomonroq.

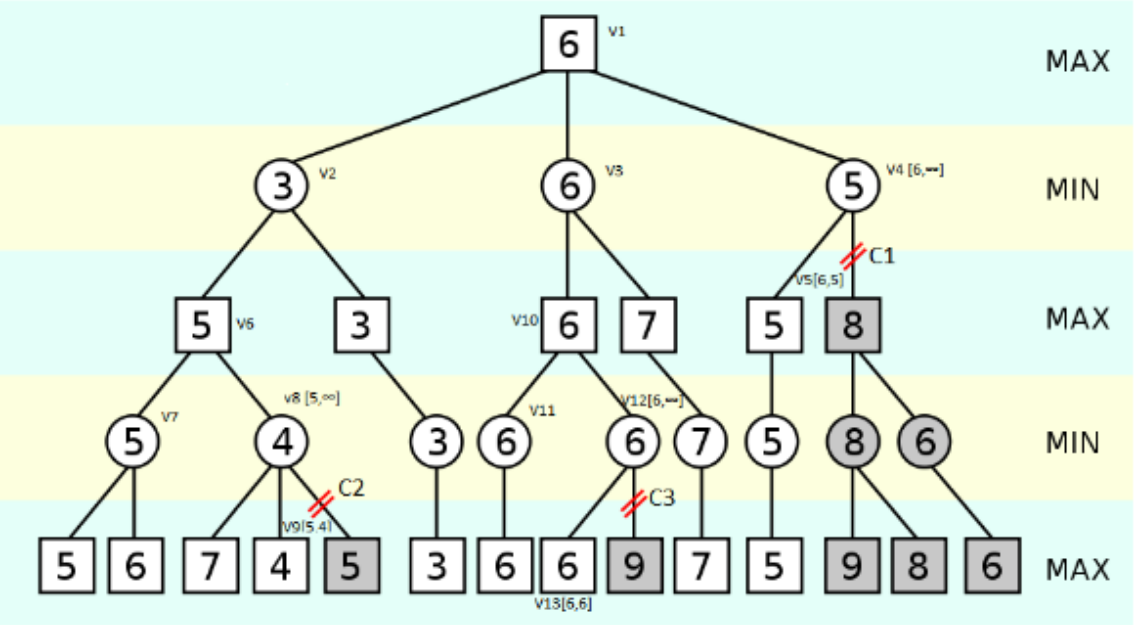
11.1-rasmdagi daraxtda alfa-beta algoritm qo’llangandan keying holat tasvirlangan. Uzuq chiziq chiziq orqali alfa-beta kesish belgilangan.

Usulning asosiy mohiyati shundan iboratki tarmoqlarni to’liq o’rganish natijasida olingan baholarni qolgan taxmin qilinayotgan eng yaxshi natijalar bilan solishtirishdir. Ayrim hollarda ba’zi hisob-kitoblar ortiqchaligini ko’rsatiish mumkin. Minmaks usuldan farqli ravishda bu usulda to’liq birin-ketin barcha usullarni ko’rib chiqilmaydi, balki daraxtni ba’zi qismlari tashlab yuboriladi.

Umuman olganda α va β izlash usulida alfa – maksimumni olgan o’yinchining eng kam olish chegarasi, beta bo’lsa minimumni olgan o’yinchining eng ko’p olish chegarasi deb qabul qilish mumkin.

- *alpha = max (alpha, f(Vi)) maksimallashtirish qismi uchun;*

- *alpha = min (alpha, f(Vi)) minimallashtirish qismi uchun;*



11.2.rasm. Bo’linish oynasi

Bu yerda:

- Vi – yechimning tugunlari, i chi korrelyatsiya qadamidaa yakunlanadi;
- Ci – i chi qadamdagi alpha va betta cheklashuv;
- MIN, MAX – maksimallashtish va minimallashtirish bo’limlarning chegaralari. E’tibor berib qarasangiz qadamlarda ular ketma ket keladi. Bunda maksimmallashtirishdagi qadam faqa shu i chi ketma – ketlikdagi shi bo’limga tegishliligini ola oladi, minimallashtirishdagi esa o’z navbatida min dagisini tanlay oladi.

Dastur kodi.

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
#include<iostream.h>
#include<time.h>
#include<math.h>
#include<stdlib.h>
#include<vector>
#include <algorithm>
#include <fstream.h>
using namespace std;
Graphics::TBitmap* pix = new Graphics::TBitmap();
RGBTRIPLE* t;
bool stop;
bool step;
```

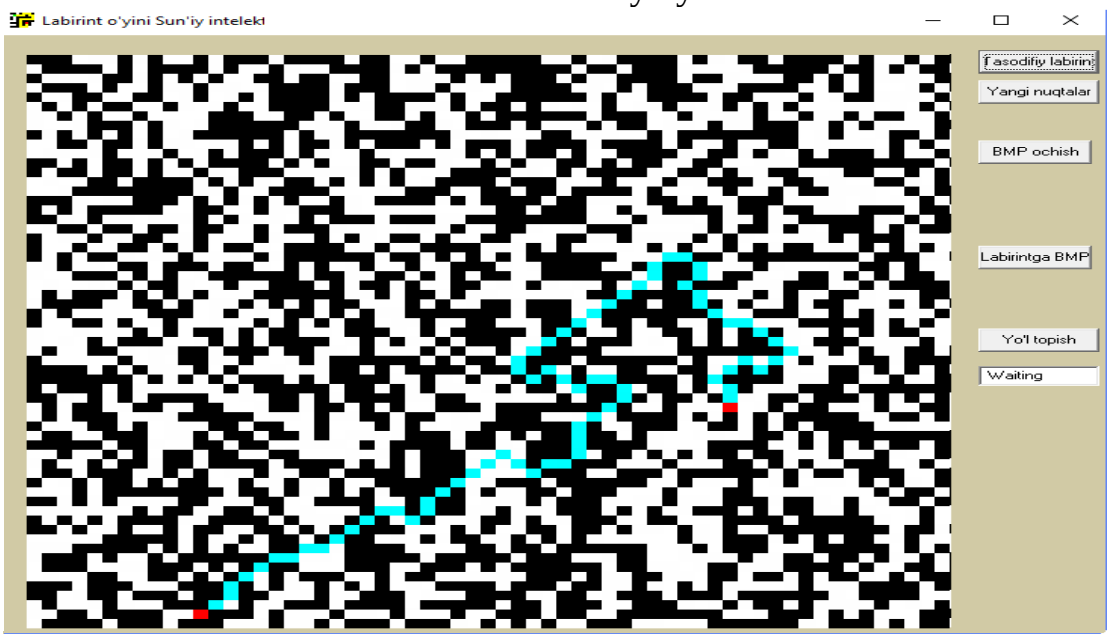
```
bool reduction;
#define GRID_SIZE 61
#define TS 10
int grid[GRID_SIZE][GRID_SIZE];

// Function Prototypes
void renderGrid();
void setSE();
void makeGrid();
void resetGrid();

enum
{       wallTile,
        openTile,
        startTile,
        endTile,
        pathTile,
};
class Astar
{public:      enum
        {
                SEARCHING,
                SUCCEEDED,
                FAILED,
        };      class Node
        {public:
                Node()
                {parent = NULL;  g = 0.0;h = 0.0;
                        f = 0.0;                }float getDistanceEstimate( int endx,
int endy )
                {
        float dx = (float)( (float)x - (float)endx );
                        float dy = (float)( (float)y - (float)endy );

                        return ((dx*dx) + (dy*dy));
                }
                bool isSameState( Node *rhs )
                {                       if( x == rhs -> x && y == rhs -> y )
                                return true;
                        else
                                return false;           }
                Node *parent;
                float g; // cost of this node + its parents
                float h; // heuristic estimate
                float f; // g + h
```

```
        int x, y;
        int type;
};      class HeapComparison
{public:
        bool operator() ( const Node *x, const Node *y ) const
        {                       return x->f > y->f;
        }       };
Astar() {           // Constructor
        clearLists();
}       void clearLists()
{               OPEN.clear();
        CLOSED.clear();
        SUCCESSORS.clear();   }
int getCoords( int xCoord, int yCoord )
{               if( xCoord < 0 || xCoord >= GRID_SIZE ||
                 yCoord < 0 || yCoord >= GRID_SIZE )
                return wallTile;
        return grid[xCoord][yCoord];
}       void setStartEnd( int x, int y, int q, int r)
{               startNode = new Node;
        endNode = new Node;
        //Initialize the start and ending nodes
        startNode -> x = x;
        startNode -> y = y;
        startNode -> type = startTile;
        endNode   -> x = q;
        endNode   -> y = r;
        endNode   -> type = endTile;
        OPEN.push_back( startNode );
        // Sort elements in heap
        push_heap( OPEN.begin(), OPEN.end(), HeapComparison() );
        // Initialise counter for search steps
        stepCounter = 0;
}       unsigned int Search()
{//Check to see if the search succeeded or failed
        if( ( currentState == SUCCEEDED) || ( currentState ==
FAILED ) )              return currentState;
        //The search fails if the OPEN list is empty (no more states to
search)          if( OPEN.empty() )
        {       currentState = FAILED;
        return currentState;
        }
        if( (getCoords( (thisNode -> x)+1, (thisNode -> y) ) != wallTile)
        && !((parentX == (thisNode -> x)+1) && (parentY ==
(thisNode -> y)))  )
```

```
{newNode = new Node;
newNode -> x = (thisNode -> x)+1;
newNode -> y = thisNode -> y;
newNode -> type = getCoords( (thisNode -> x)+1, (thisNode -> y) );
SUCCESSORS.push_back( newNode );
}vector< Node * >::iterator closedlist;
for( closedlist = CLOSED.begin();
        closedlist != CLOSED.end(); closedlist ++ )
{if( (*closedlist) -> isSameState( (*successor) ) )
{       break;
}}void makeGrid()
{       int i;
        int j;
        for( i = 0; i < GRID_SIZE; ++i ) {   // Generate the walls and open
paths
                for( j = 0; j < GRID_SIZE; ++j ) {
                        grid[i][j] = random(2);
                }       }       return;
}void setSE()
{       unsigned int x, y, q, r;
   Form1->EditStatus->Text = " Waiting";
   resetGrid();
        do {            // Set Start point
                x = (unsigned)(random(GRID_SIZE));
                y = (unsigned)(random(GRID_SIZE));
        } while( grid[x][y] == wallTile );
        grid[x][y] = startTile;
        do {    // Set End point
                q = (unsigned)(random(GRID_SIZE));
                r = (unsigned)(random(GRID_SIZE));
        } while( grid[q][r] == startTile || grid[q][r] == wallTile );
        grid[q][r] = endTile;
        aStarSearch.setStartEnd( x, y, q, r );
        return;}
        if ( currentSearchState == Astar::SUCCEEDED ) {
                aStarSearch.writePathToGrid();
                renderGrid();
     Form1->EditStatus->Text = " Success";
        }       else {
                renderGrid();     //Print the grid without writing the path 'X's to
it
                Form1->EditStatus->Text = " Impossible";
        }       aStarSearch.clearLists();}void initialize(){   randomize();
        makeGrid();setSE();       renderGrid();
  Form1->EditStatus->Text = " Waiting";}
```

```
void __fastcall TForm1::FormPaint(TObject *Sender)
{   initialize();
  renderGrid();}
void __fastcall TForm1::ButtonRandomMazeClick(TObject *Sender)
{    initialize();
}void __fastcall TForm1::ButtonFindPathClick(TObject *Sender)
{    findQuickestPath();
}void __fastcall TForm1::ButtonNewSEClick(TObject *Sender)
{    setSE();
   renderGrid();}
```

11.3.rasm. Tasodufiy oyna.



11.3.rasm. Natijalarni ko’rish

Nazorat savollari

1. α va β izlash usuli qayerlarda ishlatiladi?
2. α va β izlashda α va β nima?

12-Amaliy ish

Mavzu: Muammolar cheklovini qondirish

Ishdan maqsad: Sun’iy intellektda muammolar cheklovini qondirishni o’rganish.

Uslubiy ko’rsatmalar: Sun’iy intellektning asosiy vazifalaridan biri cheklovlarni qondirish vazifasidir. Cheklovlarni qondirish nazariyasi sun’iy intellektning kombinator yechimlari va taqdimlari uchun qulay apparat va oddiy formal sxemasidir.

Cheklovlarni qondirishning yechish masalasining maqsadi berilgan cheklovlarni qondiruvchi o’zgaruvchilar qiymatini topish hisoblanadi.

Cheklovlarni qondirish masalasi yechimining mavjudligi muammosi to’liq NP bo’lib hisoblanadi. Kombinator masalalarni effektiv yechish va deklarativ ta’riflash uchundasturlash paradigmasi bo’lib hisoblanadigan cheklovlarda dasturlash cheklovlarni qondirish bilan chambarchas bog’liq. Ferma teoremasi, propozitsional mantiqning bajarilganlik masalasi va graflar nazariyasidagi graflarning izomorflik masalasi kabi ma’lum bo’lgan ko’pgina klassik kombinator masalalar cheklovlarni qondirish masalasi ko’rinishida aniq ifodalanadi. Matematikada avval qo’yilgan bitta masala – xususiy holda kartalarni bo’yash masalasi bo’lib hisoblanuvchi grafni bo’yash masalasiga to’xtalamiz. Cheklovni qondirish masalasi ko’rinishidagi bo’yash masalasini aniq ifodalanishi bo’yaluvchi graf o’zgaruvchisi mos cho’qqiga qo’yiladi. Mumkin bo’lgan rang o’zida o’zgaruvchi domenini aks ettiradi, yonma-yon uchlar orasidagi tengsizlikni cheklash esa vazifalarni cheklash hisoblanadi.

Aytish kerakki, bu yerda barcha aspektlarni, yo’naltirilgan cheklashlarni qondirish nazariyasini va cheklashlarda dasturlashni to’liq tavsiflash imkonsiz. Shuning uchun to’liq ma’lumotni Rassel S, Norvig Plarning tarjima monografiyalaridan olish mumkin.

Cheklovlarni qondirishning optimallashtirish muammosini hal qilish quyidagicha vazifalarni bir ketma-ketlikda hal qilish orqali kamaytirilishi mumkin. Yechim buyechimdan ko’ra obeyektiv funksiyaning qiymati yaxshiroq bo’lsin degan shart qo’yadigan obyektiv funksiyaga tegishli cheklov qo’shilganidan keyin joylashishi mumkin. Shu paytgacha ishlab chiqilgan bu boshlang’ich qiymatning ketma-ket tuzatilishlari masalaga ruxsat berilmagunicha optimal yechimni qidirishga imkon beradi.

1-misol. Cheklovlarni qondirishning eng oddiy masalasi bo’lib tenglamalar sistemasini yechish hisoblanadi. Chekli F maydondagi chiziqli tenglamalar sistemasi hisoblansin. U yechimga egami? Ravshanki, Tenglama o’zgaruvchilari diapason hosil qilishi, ko’pgina tenglama yechimiga mos keluvchi barcha kortejlar cheklash munosabatini hosil qilishi sababli bu misoldagi har bir alohida tenglama cheklash bo’lib hisoblanadi.

2-misol. 3-bajariluvchili propozitsional standart masalasi(3-SAT) konyuksiya, dizyuksiyadan tashkil topgan mantiqiy propozitsional formulalar

vazifalarini aniqlaydi. Har bir dizyunkt 3ta literaldan tashkil topadi va rostlik formulalarini bajaruvchi o’zgaruvchi qiymatlariga egami degan savolga javob beradi.

Cheklovlarni qondirish masalasi SAT masalasini yechish masalasidagi o’zgaritirish bo’lishi mumkin. O’zgaruvchilarni tanlaymiz. O’zgaruvchilar faqat va faqat o’zgaruvchi qiymat o’zlashtirganda “rost” qiymatini qabul qiladi. Har bir o’zgaruvchi uchun bir vaqtda o’zgaruvchiga turli xil qiymatlar o’zlashtirilmasligini kafolatlash uchun shu o’zgaruvchining barcha qiymat juftliklariga klauzlar(dizyunktlar) qo’shiladi. Dizyunkt qo’shilishi o’zgaruvchiga hech bo’lmasa bitta qiymat o’zlashtirilishini kafolatlash uchun qo’shiladi.

3-misol. Cheklovlarni qondirishninf har qanday aniq masalasi mantiqiy mo’rininshda ifodalanishi mumkin. Albatta, predikatlar va munosabatlar orasida standart moslikdan foydalanib, cheklovlarni qondirish masalasini birinchi tartibli formula ko’rinishida qayta yozish mumkin. Savol esa formula bajarilgan bo’lib hisoblanadimi deb qo’yiladi. Bu masala odatda ma’lumotlar bazasi nazariyasida foydalaniladi. Buning uchun yana quyidagi misolni ko’rsatamiz.

4-misol. Relyatsion ma’lumotlar bazasi ko’p sondagi chekli jadvallar kabi ko’rib chiqiladi. Jadval sxemalardan va aniq ma’lumotlardan foydalanadi. Bu yerda sxema chekli sondagi ko’plab atributlar. Har bir atribut o’ziga mos keluvchi mumkin bo’lgan domen deb nomlanuvchi qiymatlarga ega bo’ladi. Relyatsion ma’lumotlar bazasining standart vazifasi konyuktiv so’rov yechimi mavjudligini so’rovchi konyuktiv so’rovlarni baholash vazifasi hisoblanadi. Relyatsion ma’lumotlar bazasi ustidagi konyuktiv so’rovlar aniq cheklovlarni qondirish masalasi misoliga mos keladi. Bunda bazaning terminlari quyidagilarga mos keladi: “atributlar” “o’zgaruvchilar”ga almashadi, “jadvallar” “cheklovlar”ga almashadi, “sxemalar” “diapazonlar”ga, “aniq ma’lumotlar” “cheklov munosabatlariga” va “qatorlar” “kortejlar”ga almashadi. Demak, konyuktiv so’rov cheklovlarni qondirishning aniq vazifasiga ekvivalent bo’lib hisoblanadi. Bunda o’zgaruvchilar – bu so’rov atributlari. So’rovdagi har bir atomar formula uchun cheklov diapazoni cheklov munosabatlari va o’zgaruvchi formulalari ro’yxati kabi cheklovlar topiladi.

«Send More Money» masalasi uchun cheklovlarni qondirish modelini ko'ramiz. Talaba uyiga quyidagi kodlangan telegrammani jo'natyapti:

$$\begin{array}{r} SEND \\ +\ MORE \\ \hline MONEY \end{array}$$

Telegramda foydalanilgan 8 ta kodlangan harfdan 8 ta kodlangan raqamni topish vazifasi berilgan. Aytish kerakki, bu masala butun sonli dasturlash modeli yordamida bajarilishi mumkin.

Shu paytda mantiqiy tahlil uning yechimini topish imkonini beradi. Bu yerda S ha M ham nol bo'lishi mumkin emas. Davom etamiz, M uchun yagona qiymat 1 bo'lishi mumkin, M = 1. Keyin, M = 1 ligidan S = 9 bo'lishi mumkin, Ko'rsatish mumkinki, O belgisi 0 ga mos keladi. Davom etamiz, O = 0 shartda 100 liklarga o'tib, E va N har xil bo'lishini tahmin qilamiz. Bunda N = E + 1ni olamiz. Mos mantiqiy tahlilni davom ettirib masala yechimini topamiz: E = 5, N = 6, D = 7, R =

8, Y = 2. Bu masalaning asosiy murakkabligi «barcha raqamlar har xil» cheklovini qayd etish zaruriyatidir.

Masalaning cheklovini yozamiz:

O'zgaruvchilar: {S, E, N, D, M, O, R, Y}

Domenlar: {0,1,…,9}

Cheklovlar:

C1 : ∀x, y ∈ {S, E, N, D, M, O, R, Y}, x ≠ y

C2 :M = 0 or M = 1

C3 : (1000 x S + 100 x E + 10 x N +D) + (1000 x M + 100 x O + 10 x R +E) = (10000 x M + 1000 x O + 100 x N + 10 x E +Y)

Ikkinchi formulalashtirish

O'zgaruvchilar: {S, E, N, D, M, O, R, Y} + {C1, C2, C3}

Domenlar: ∀x ∈ S, E, N, D, M, O, R, Y, D_x = 0 … 9

∀y ∈ { C1, C2, C3}, D_y = 0, 1

Cheklashlar:

C1 : ∀x, y ∈ {S, E, N, D, M, O, R, Y}, x ≠ y

C2 : M = 0 or M = 1

C3 : D + E = 10 x C1 + Y

C4 : N + R + C1 = 10 x C2 + E

C5 : E + O + C2 = 10 x C3 + N

C6 : S + M + C3 = 10 x M +O

Masalaning Prolog dasturlash tilidagi kodi:

```
sendmore(Digits) :-
Digits = [S,E,N,D,M,O,R,Y], % O'zgaruvchilarni yaratish
Digits :: [0..9], % o'zgaruvchilar domenlari
S #\= 0, % Cheklov: S 0dan farq qilishi kerak
M #\= 0, % Cheklov: M 0dan farq qilishi kerak
alldifferent(Digits), % Barcha o'zgaruvchilar turli xil qiymatlarni qabul qilishi kerak
1000*S + 100*E + 10*N + D % boshqa cheklovlar
+ 1000*M + 100*O + 10*R + E
#= 10000*M + 1000*O + 100*N + 10*E + Y,
labeling(Digits). % qidiruvni boshlash
```

Cheklovli dasturlash cheklovlar formasida ko’rsatilgan o’zgaruvchilar orasidagi munosabatdagi dasturlashning paradigmalari hisoblanadi. Cheklashlar bajarish qadamlari ketma-ketligini emas harakatli yechimlar xususiyatini aniqlovchi umumiy ibtidoiy qat’iy dasturlash tillaridan farq qiladi. Bu deklarativ dasturlashning cheklangan formasini dasturlashtiradi. Cheklovlar dasturlarida foydalaniladigan cheklash turli xil ko’rinishlarda bo’ladi: cheklovlarni qondirish masalalarida ishlatiladiganlari(masalan A yoki B rost), simpleks algoritmlarni yechuvchi(masalan, x ≤ 5) va boshqalar. Odatda, cheklovlar dasturlash tilida yoki mavjud alohida dasturlash kutubxonalarida quriladi .

Cheklovli dasturlash mantiqiy dasturlashdagi cheklovlarni kirituvchi cheklovli mantiqiy dasturlash bilan chambarchas bog’liq.Bunday mantiqiy

dasturlash variantlarining paydo bo’lishi Jaffar va Lassez nomlari bilan bog’liq. Ular 1987-yilda belgilangan cheklovlar sinfini ochishdi. Cheklovlardagi mantiqiy dasturlashning birinchi ishlatilishi CLP(R) va CHIP da bo’lgan.

Nazorat savollari

1. Cheklovlarni qondirish masalasi nima?
2. Cheklovlarni qondirish masalasi qayerlarda ishlatiladi?

13-Amaliy ish

Mavzu: Matnni tanish

Ishdan maqsad: Tasvirlar ustida ishlovchi algoritmlarni o’rganish. Tasvirdagi matnlarni tanishni o’rganish.

Uslubiy ko’rsatmalar: Insonning ko'zi atrof-muhitdagi ko'pgina obyektlarni, ularning rangini ajratan oladi. Hozirda insonning turli xususiyatlariga o'xshash texnologiyalar kompyuterlar yordamida yaratilmoqda va tibbiyot, ta'lim, xavfsizlik va boshqa sohalarga joriy qilinmoqda.

Obrazlarni tanuvchi tizimlarni yaratish, hozirda ham, birmuncha murakkab vazifa bo'lib turibdi. Biroq hozirda harflarni tanuvchi , yuzlarni tanuvchi, shtrix-kodlarni tanuvchi, ovozni, mashina nomerlarini ta'nuvchi ko'pgina tizimlar mavjud.

Matnni tanish tizimlaridagi muhim rol bo'lib matnni tanish hisoblanadi.

Rasmdagi matnni tanish ikki bosqichda amlga oshiriladi:

1. Rasmdan matn bo'lishi mumkin bo'lgan sohalarni aniqlash.
2. Aniqlangan sohalarda matn bor-yo'qligini tekshirish.

Rasmdan matnni tanib olish yangi muammo hisoblanmaydi, asosiy muammo rasmdagi matnni aniqlashning optimal usuli yo'qligidadir.

Matnni tanish bo'yicha olib boriluvchi ilmiy izlanishlar:

Matnni tanish xalqaro mavzu bo'lib, bu bo'yicha dunyoning ko'pgina universitetlari shug'ullanmoqda.

Afina universiteti o'zining elektron kutubxonasida "Text detection in video frames" , "Text Detection in Images and Videos" ishlarini saqlaydi. Rochestr instituti o'z eletron bazasida "Detection of Text in Video" maqolasi mavjud.

Veyvlet o'zgartirish (ing. Wavelet transform) – o'zida signalli veyvlet funksiyalar o'ramini aks ettiruvchi integral o'zgartirishdir. Veyvlet o'zgartirish vaqtga bog’liq berilishdagi signalni chastota-vaqt ko’rinishidagi signalga o’giradi. Ushbu atama(wavelet)ni inglizchadan tarjima qiladigan bo’lsak “kichik to’lqin” degan ma’noni bildiradi. Veyvlet – bu muayyan vaqt va chastota bo’yicha belgilangan formadagi matematik funksiyaning umumiy nomi.

Veyvlet shartlari

Veyvlet o'zgartirishlar mavjud bo’lishi uchun veyvlet funksiyalar quyidagi shartlarni qanoatlantirishi kerak:

Veyvlet chekli energiyaga ega bo’lishi kerak:

$$E = \int_{-\infty}^{\infty} |\Psi(t)|^2 dt < \infty$$

Agar Ψ(t) uchun Ψ(f) - fure o'zgartirish bo'lsa,

$$\Psi(t) = \int_{-\infty}^{\infty} \Psi(t) e^{-i(2\pi f)t} dt$$

Bunda quyidagi shart bajarilishi kerak:

$$C_\Psi = \int_{-\infty}^{\infty} \frac{|\Psi(f)|^2}{f} dt < \infty$$

Bu shart ruxsat etilgan shart deyiladi, va bundan kelib chiqadiki – veyvlet nol chastotali komponentga yaqinlashganda $\Psi(0) = 0$ shartini qanoatlantirishi kerak yoki boshqacha holatda veyvlet $\Psi(t)$ o'rtacha nolga teng bo'lishi kerak.

Rasmlarni qayta ishlash algoritmlari

obyektlarni tanishda rangli rasm bilan birgalikda oq-qora rasmdan ham foydalanish mumkin. Rangli rasmdan foydalanganda biz obyekt haqida ko'proq ma'lumot olishimiz mumkin. Oq-qora rasm bilan ishlaganimizda esa vaqt va tezlikdan yutamiz.

Bu ishimizda biz rangli rasmni qayta ishlashda rasmlarning kulrang ton(greyscale) shkalasi va binarlash amali ishlatiladi.

Greyscale - kulrang tonli rangda aks etuvchi rangli rasm ko'rinishi. bu algoritm rangli ramni kulrang ton shkalasiga o'tkazish imkonini beradi. Algoritm rasmning kengligi va balandligi bo'ylab amalga oshadi.

Quyi boshlanishdan binarlash bir muncha sodda bo'lib, bunda faqat boshlanishning bitta qiymati ishlatiladi:

$$f'(m,n) = \begin{cases} 0, f(m,n) \geq t \\ 1, f(m,n) < t \end{cases}$$

Kriteriya bilan birgalikdagi barcha qiymatlar 1 bo'ladi. Bu holatda t-0(qora) boshlang'ichdan katta barcha piksellarning qiymatlari(ampletudalari) ham 255(oq) bo'ladi. ish algoritmning natijaviy ko'rinishi quyidagi rasmda:

13.1. rasm. Rasmni qayta ishlashning asosiy usullari

Gistogramma "Dokument" ko'rinishidagi rasmning matnini belgilab olishda ishlatiladi. Bunaqangi rasmlarda matnlar qator va ustun bo'yicha tekislangan bo'ladi. Gistogramma tuzishning ikki xil usuli mavjud: qidiruv paytida butun

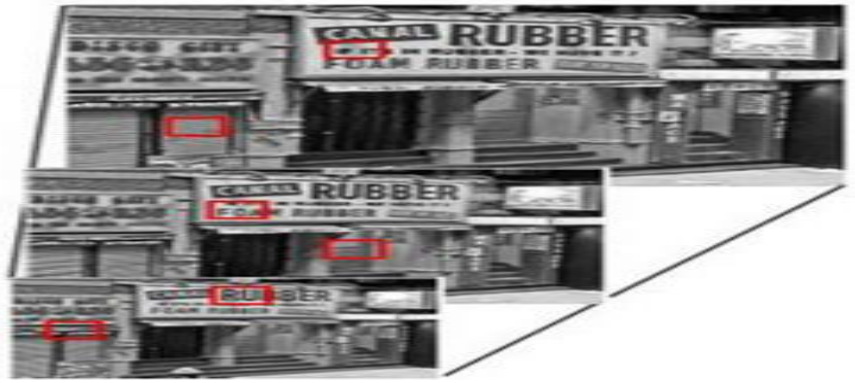
sahifa baholab chiqiladigan yuqoridan pastga usuli va matnli sohalarni aniqlash uchun simvollar belgilanib chiqiluvchi pastdan yuqoriga usuli.

Birinchi imkoniyat sahifani proyeksiyalash usulini o'z ichiga oladi. Bu usulning asosiy g'oyasi belgilangan qatordagi qora rangli piksellarni sanab chiqishni va vertikal yoki gorizontal gistogrammani yaratishni o'z ichiga oladi. Keyin gistogrammaning rangli sohalarini kesish amalga oshiriladi. Vertikal va gorizontal proyeksiyalar bir –birining ustiga qo'yiladi. Agar dokumentda rasm bo'lmasa bu usul yaxshi natija beradi.

«Pastdan yuqoriga» usuli Docstrum algoritmi va Voronov diagrammasidan foydalanuvchi algoritm bo'lib hisoblanadi. Bu algoritmlar uchun gistogrammalar simvollar, so'zlar, qatorlar va bo'limlarga bog'liq komponentlar orasidagi masofaga asosan quriladi.

Dokument tipidagi rasmdan ko'ra ixtiyoriy rasmdan matnlarni ajratib olish uchun ko'proq usullar mavjud. Ularni teksturali foydalanish va belgilangan sohalardan foydalanish usullariga bo'lish mumkin.

Matn teksturasi oddiy rasm teksturasidan ancha farq qiladi.Tekstura belgilarini qurish uchun kattalashtirilgan yoki kichraytirilgan o'lcham bo'yicha rasm «piramida»si quriladi(2-rasm). Keyin barcha rasmlar guruhiga piksel bo'yicha o'tish amalga oshiriladi va kosinus yoki veyvlet o'zgartirish yordamida tekstura belgilari aniqlanadi. Bu usul bir xil shriftdagi bir yo'nalishli tekstlar uchun yaxshi ishlaydi.



13.2.rasm. Belgilab olish.

Xulosa

Xulosa qilib aytganda matnni tanish yo'nalishi tasvirlar bilan ishlashning eng asosiy sohalaridan hisoblanadi. Tasvirlardan matnni tanishda tasvir ustida bir qancha algoritmlar asosida operatsiyalar bajarilganligi sababli ancha murakkab jarayon hisoblanadi. Mutaxasislarning eng asosiy vazifasi jarayonni amalga oshirishning optimal va sodda usulini topish hisoblanadi. Bugungi kunda ana shunday bir qancha usul va algoritmlar ishlab chiqilgan va yana ishlab chiqilmoqda

Ushbu mustaqil ishda tasvirdagi matnni tanishning veyvleto'zgartirish va greyscale algoritmlarini ko'rib chiqdik. Bu usul ancha sodda bo'lgani uchun ishlatish qulay hisoblanadi.

Nazorat savollari

1. Matnni tanish algoritmlari haqida gapiring

2. Veyvlet algoritmini tushuntiring
3. Matnni tanishda veyvlet algoritmi qaysi qismda ishlatiladi?

14-Amaliy ish

Mavzu: Mashina ta’limi

Ishdan maqsad: Mashina ta’limi tushunchasini o’rganish.

Uslubiy ko’rsatmalar: Mashina ta’limi Sun’iy intellektning katta qismini tashkil qiluvchi bo’limi hamda o’rganuvchi algoritmlarni tuzish metodlarni izlaydi. Mashina ta’limi ikki turga bo’linadi:

- Pretsedentlar bo’yicha ta’lim yoki induktiv ta’lim. Berilgan tanlanmalar bo’yicha tanlanma elementlari orasidagi qonuniyatlarni topishga asoslanadi.
- Deduktiv ta’lim. Ekspert bilimlarini formallashtirish hamda ularni bilimlar bazasi ko’rinishida kompyuterga ko’chirish.

Mashina ta’limi matematik statistika, optimallashtirish metodlari va klassik matematik qoidalar bilan bog’liq. Ko’p induktiv ta’limning metodlari klassik statistik yondashuvlarga alternativa sifatida ishlab chiqilgan. Mashina ta’limi ko’p metodlari axborotni olish va ma’lumotlarning intellektual analiziga bog’liq.

Mashina ta’limiga faqat matematik tarafdan emas balki amaliy tarafdan ham qarash lozim. Amaliyotda qo’llanuvchi algoritmlarni yaratish uchun nazatiy qism yetarli emas. Bunday metodlarni “yaxshi” ishlashi uchun qo’shimcha evristikalarni kashf etishga majbur qiladi. Har qanday mashina ta’limi tadqiqotlari amaliy qo’llash eksperimentlarsiz o’tmaydi. Bunday eksperimentlar modelli yoki real ma’lumotlar bilan o’tkaziladi hamda algoritmning amaliy qo’llashni tasdiqlab beradi.

Mashina ta’limining masalaning qo’yishilishiga qarab ikki turga bo’linadi:

Ustoz yordamida ta’lim(supervised learning) – ko’p tarqalgan tur. Har bir pretsedent o’z navbatida “obyekt va javob” juftlikni tashkil qiladi. Berilgan obyektlar xususiyatiga ko’ra javoblar o’rtasidagi bog’liqlik qonuniyatlarini topish algoritmlarini tuzishdan iborat ya’ni kirishga obyektning xususiyatlarni beriladi, chiqishda esa shu belgilarga mos javob hiqadi. Ushbu turga quyidagi masalalar kiradi:

- Klassifikatsiya masalasi (classification);
- Regressiya masalasi (regression);
- Ranjirlash masalasi (learning to rank);
- Bashoratlash masalasi (forecasting).

Ustozsiz ta’lim(unsupervised learning). Bunda javoblar berilmaydi va bu yerda obyektlar orasidagi bog’liqliklarni topish masalasi qo’yiladi. Ushbu turga quyidagi masalalar kiradi:

- Klasterizatsiya masalasi (clustering);
- Bog’liqlik qoidalarini topish masalasi (association rules learning);
- Chiqindilarni filtrlash masalasi (outliers detection);
- Ishonchlilik chagarasini qurish masalasi (quantile estimation);

- O’lchamni qisqartirish masalasi (dimensionality reduction).

Nazorat savollari

1. Sun'iy intellektda mashina ta'limining ishlatilishini tushuntiring
2. Mashina ta'limining keng tarqalgan turi qaysi?
3. Ustozsiz ta'limga kiruvchi masalalar qaysilar?

1-Laboratoriya ishi.

Mavzu: Optimallashtirish va nisbiylik

Ishdan maqsad: Masalaning yechimini topishning optimal usulini ishlab chiqishni o'rganish. Massivning eng katta yoki eng kichik elementini topishning optimal usullarini aniqlash.

Masalaning qo'shilishi: Elementlari sonlardan iborat massiv berilgan. Ushbu massivning eng katta va eng kichik elementlarini topishning optimal usuli topilsin.

Uslubiy ko’rsatmalar: Optimizatsiya – bu matematika, informatika va ilmiy izlanishlar masalalarida cheklangan chiziqli yoki chiziqli bo’lmagan tenglama yoki tengsizliklarning chekli vector fazosidagi bir qancha sohalardagi butun funksiyaning ekstremumini topishdir. Optimizaytsiya masalalarini yechish nazariyasini matematik dasturlash o’rganadi.

Loyihalash jarayonida odatda obyekt parametrlari strukturasi va qiyamatlariga qarab eng yaxshi usul aniqlab olinadi. Agar optimizatsiya obyrekt parametrining optimal qiymati bilan bog’lngan bo’lsa, unda bu optimizatsiya parametric optimizatsiya deyiladi. Standart matematik masala quyidagi ko’rinishda shakllantiriladi. f(χ) funksiyaning ko’p sondagi X lar orasida χ elementlar orasida f(χ^*) minimal qiymatini beruvchi χ^* elementni toppish kerak. Masalani yechishda to’g’ri yechimni berish uchun quyidagilarni yechish kerak bo’ladi:

4. Ko’pincha yo’l qo’yilgan X lar

$$\mathbf{X = \{\vec{x} \mid g_i(\vec{x}) \le 0,\ i = 1, \ldots, m\}}$$

5. Butun funksiyani ko’rsatish

$$f : X \to R$$

6. Qidiruv kriteriyasi(max yoki min).

Shunda $f(x) \to \frac{min}{\vec{x} \in X}$ masalasini yechilish natijasida quyidagilardan biri hosil bo’ladi:

e) $X = \emptyset$.

f) $f(\vec{x})$ butun funksiya quyidan chegaralanmagan.

g) $\vec{x}^* \in \mathbf{X : f(\vec{x}^*) =} \frac{min}{\vec{x} \in X} f(\vec{x})$.

h) Agar $\nexists \vec{x}^*$ bo’lsa, $\frac{inf}{\vec{x} \in X} f(\vec{x})$ topilsin.

Endi optimizatsiyani dasturlash tomonidan ko’rib chiqamiz. Optimizatsiya – bu dasturning effektivligini yaxshilash maqsadida tizimni modifikatsiyalanishi.

Bunda tizim bitta kompyuter dasturi, raqamli qurilma, kompyuterlar to’plami yoki hatto internetga o’xshash butun tarmoq bo’lishi mumkin. Hech bo’lmaganda to’liq optimizatsiyalash optimal tizimni olish hisoblanadi. To’g’risini aytadigan bo’lsak opitimizatsiyalash jarayonidagi optimal tizim har doim ham yetarli bo’lib hisoblanmaydi. Optimallangan tizim odatda faqat bitta masala yoki foydalanuvchilar guruhi uchun tegishli bo’ladi. Bu masala yoki foydalanuvchilar guruhidagi optimallashtirishga masalani yechishda, xotira hajmidan yo’qotish evaziga bo’lsa ham vaqt sarfini kamaytirish; xotira muhim bo’lgan ilovalarda esa xotiraga kam murojat qiluvchi algoritmlardan foydalanish orqali masalani hal qilish kabilar kirishi mumkin.

Ba’zi masalalar ko’pincha samarali bajariladi. Masalan, C tilidagi 1 dan N gacha bo’lgan barcha butun sonlar yig’indisini topuvchi dasturni olaylik:

```
int i, sum = 0;
For (i  = 1; i <= N; i++)
sum+=i;
```

Ko’rinib turganiday bu kodda unchalik ham murakkablik yo’q. Bu kodga quyidagicha o’zgartirish kiritamiz:

```
Int sum = (N*(N+1))/2
```

“Optimizatsiya” odatda anglanilganidek tizimning o’zida funksionallikni saqlashidir. Biroq unumdorlikni sezilarli yaxshilash ortiqcha funksionalliklarni olib tashlash orqali ham amalga oshirish mumkin. Masalan, dastur 100tadan ko’p element kiritishga talab qilmasa dinamik xotiradan emas static xotiradan foydalanish amalga oshiriladi.

Protsessor vaqtini sarflash bo’yicha optimizatsiya odatda matematik hisblashga ega katta solishtirma og’irlikka hisoblovchi protsessor uchun juda muhim. Bu yerda dasturchi dastur kodini yozish paytida bir qancha optimizatsiyalash usullari keltirilgan.

- Ma'lumotlar obyekti initsializatsiyasi. Qaysidir ma'lumotlar obyekti qismining ko'pgina dasturlarida initsializatsiyalash(boshlang'ich qiymatni o'zlashtirish) zarurati mavjud. Bunday boshlang'ich qiymatlarni o'zlashtirish yoki dastur boshida yoki sikl oxirida amalga oshiriladi. Boshlang'ich qiymat berish protsessorning qimmatli vaqtini tejaydi. Masalan, agar gap sikldan foydalanib massivni initsializatsiyalash haqida ketayotgan bo'lsa, bu massivni to'g'ridan to'g'ri o'zlashtirishni e'lon qilishdan ko'ra kamroq samarali bo'ladi.

- Arifmetik operatsiyalarni dasturlash.

Massivning eng katta yoki eng kichik elementini aniqlashning juda ko’p usullari mavjud bo’lib, ulardan eng oddiy va universal usuli bu solishtiriluvchi elementgacha bo’lgan eng kichik yoki eng katta elementlarni ketma-ket keyingilari bilan solishtirish usulidir.

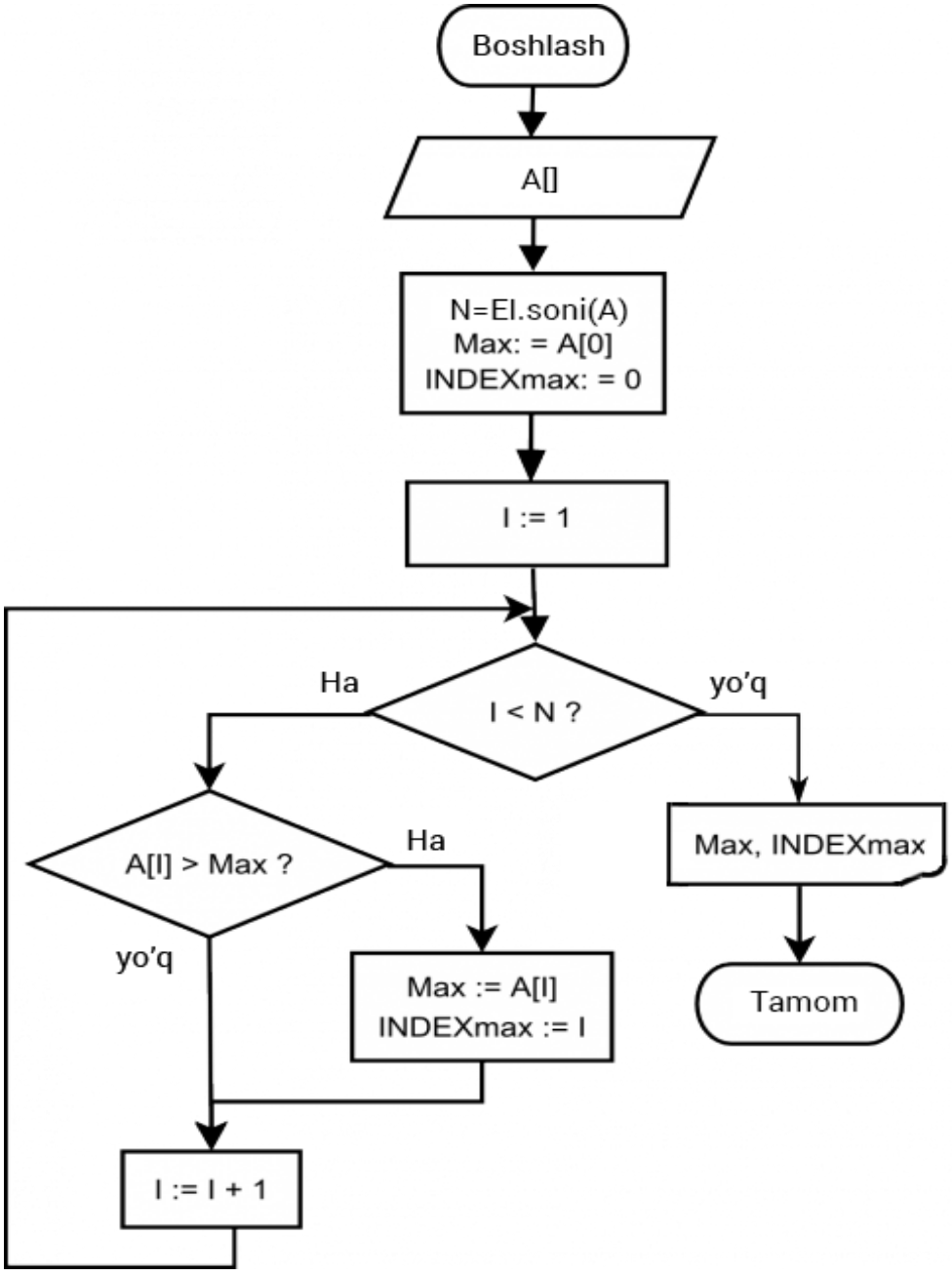
```
int FindMax(int* mass,int count)
{
        if(count <  1) return null;
        int max = mass[0];
        for (int i = 1; i < count; i++)
          if (mass[i] > max)
```

```
            max = mass[i];
    return max;
}
```

Bu usulning qolgan usullardan ustun tomoni shundaki, bu usul ixtiyoriy massiv uchun ishlaydi va har qanday dasturlash tili uchun dastur kodini yozish oson va resurs kam talab etiladi.

Bu usulning yagona kamchiligi shundan iboratki bu bu usul orqali juda katta hajmdagi massivlarda sekin ishlashidir.

Bu usullardan tashqari izlashning binar usuli mavjud bo'lib u juda katta hajmli massivlar uchun ishlatish mumkin. Lekin bu usul uchun juda ko'p kompyuter resursi talab etiladi.



1.1-rasm. Saralash algoritmi uchun blok sxema.

Dastur kodi.

```
#include <iostream>
using namespace std;
int main()
{    int *arr;
   int size;
      cout << "n = ";
   cin >> size;
   if (size <= 0) {
            cerr << "Massiv hajmi 0 dan katta bo'lishi kerak." << endl;
      return 1;    }
   arr = new int[size];
   for (int i = 0; i < size; i++) {
      cout << "arr[" << i << "] = ";
      cin >> arr[i];
   }
   int max = arr[0];
   for (int i = 1; i < size; i++) {
      if (arr[i] > max) {
         max = arr[i];
      }
   }
   cout << "max = " << max << endl;
   delete [] arr;
   return 0;
}
```

Nazorat savollari:

1. Masalalarni yechishda optimallashtirish nimaga kerak?
2. Masalalarni yechishda nisbiylik nimaga kerak?
3. Qidiruv algoritmlarida optimallashtirish va nisbiylik qanday qo'llaniladi?

2 - Laboratoriya ishi.

Mavzu: Hisoblash murakkabligi.

Ishdan maqsad: Masalalarni yechishning hisoblash murakkabligini o'rganish. Ushbu jarayonni o'yinlar yaratishda qo'llashni ko'rib chiqish.

Masalaning qo'yilishi: Puzzle o'yinini ishlab chiqing.

Uslubiy ko'rsatmalar: Hisoblash murakkabligi – bir qancha algoritmlarni bajaruvchi ish hajmiga bog'liq funksiylarni belgilydigan informatika va algoritm nazariyasidagi tushuncha. Hisoblash murakkabligini o'rganish bo'limi hisoblash nazariyasi deb nomlanadi. Uning ish hajmi odatda hisoblash resursi deb nomlanuvchi fazo va vaqt abstract tushunchasi bilan o'lchanadi. Unga

sarflanuvchi vaqt masalani yechish uchun zarur bo’lgan elementar qadamlar soni bilan aniqlanadi. Bunday holda, “kirish hajmiga bog’liq xotira bandligining hajmini va bajarish vaqtini qanday o’zgartirish kerak” degan algoritmlarni ishlab chiqish savoliga javob beramiz. Kirish hajmi ostida bitlarda berilgan vazifalarning tavsifi uzunligi, chiqish hajmi ostida esa – masalani yechishning tavsifi uzunligi tushuniladi.

Murakkablik sinflari. Murakkablik sinflari – bu mavjud hisoblash murakkabligi bo'yicha o'xshash algoritmlar yechimini tanishning ko’p sonli vazifasidir. Bunda ikkita muhim sinf mavjud:

P sinfi. P sinfi “tez” hisoblanuvchi barcha yechimlar, muammolarni aralashtirib yuboradi. Bu graf kabilar bilan bo’gliqlikni aniqlovchi massivdan elementlarni qidirish va saralashga olib boradi.

NP sinfi. NP sinfi kiruvchi ma’lumotlar hajmiga bog’liq polynomial qadamlar sonini yechuvchi determinirlanmagan Tyuring mashinasi vazifalarini o’z ichiga oladi. Ularning yechimi polynomial qadamlar soni uchun determinirlangan Tyuring mashinasi yordamida tekshirilishi mumkin. Shuni aytish kerakki, determinirlanmagan Tyuring mashinasi faqat o’sh vaqtdagi chegaralangan xotirali determinirlangan Tyuring mashinasiga mos keluvchi zamonaviy kompyuterlarga o’xshash bo’lgan abstract modeldir. NP sinfi o’z ichiga P sinfini hamda kirish hajmiga eksponensial bog’liq faqat ma’lum bo’lgan algoritmlarni yechishning bir qancha muammolarini o’z ichiga oladi. NP sinfiga komivoyajer masalasi, bul formulalarini bajarish masalasi, faktorizatsiya masalasi va boshqa shu kabi masalalari kiradi.

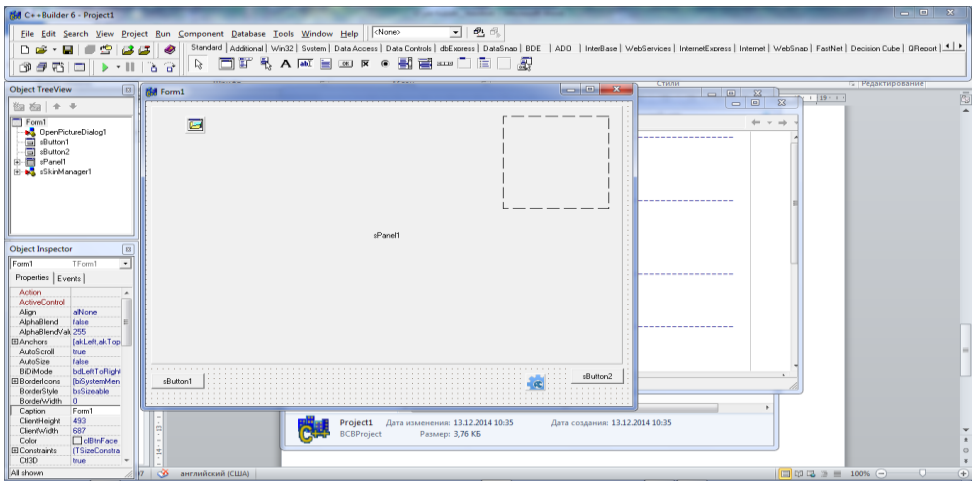
NP va P sinflari tengligi muammosi. Bu ikki sinf tengligi haqidagi savol nazariy informatikadagi eng ochiq muammolardan biri hisoblanadi. Kley matematika instituti bu muammoni ming yillik muammolar ro’yxatiga kiritgan va bu muammo yechimi uchun 1 million AQSH dollari miqdorida mukofot e’lon qilgan.

O’yin dasturining yaratilish jarayoni. Bu laboratoriyada puzzle yani bo’lak rasmlarni yig’ish o’yinini C++ builder muhitida tuzldi. Bu o’yinni tuzishda C++ builderning Additional komponentalar bo’limidagi Image, AlphaLite komponentalar bo’limidagi sButton, sPanel, sSkinManager va Dialogs komponentalari bo’limidagi OpenPictureDialog komponentalaridan foydalanildi.

O’yin algoritmi quyidagicha bo’ladi:

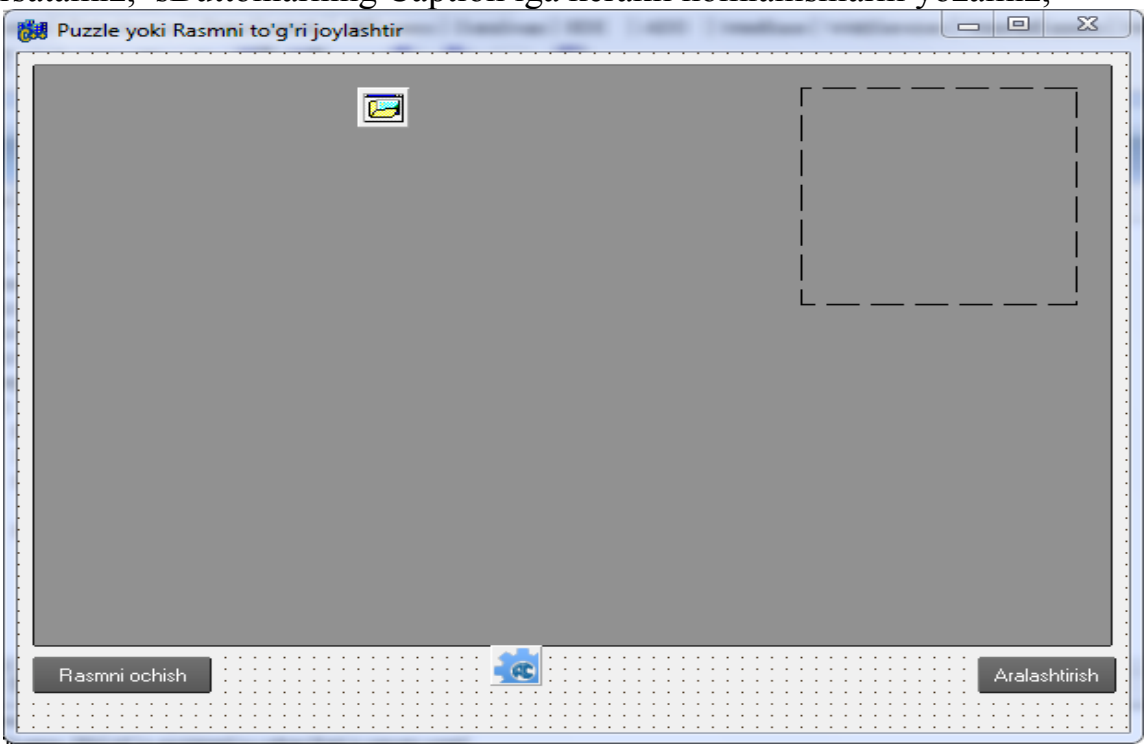
5. Rasm yuklab olinadi
6. Bo’laklarga ajratiladi va dastlabgi holat saqlab qo’yiladi
7. Rasm aralashtiriladi
8. Rasm bo’laklari harakatlantiriladi va ushbu holat dastlabki saqlangan holat bilan solishtiriladi. Agar joriy holat saqlangan holat bilan mos tushsa o’yin tugatiladi, aks holda o’yin davom ettiriladi

Bu o’yinni tuzish uchun avval C++ builderda forma hosil qilamiz va unga yuqoridagi komponentalarni joylashtiramiz.



2.1.rasm. Dasturni tanlash

Keyin joylashtirilgan komponentalarning properties dagi xususiyatlarini sozlab chiqamiz: sSkinManager ga qobiq fayl turgan papkani va qobiq nomini ko'rsatamiz; sButtonlarning Caption iga kerakli nomlanishlarni yozamiz;



2.2.rasm. Rasmni to'g'ri jaoylashtirish

Endi rasmni bo'laklarga bo'lingan holda chiquvchi 30x30 o'lchamli kataklarni hosil qilish uchun

```
TPole *A[30][30];
TPole *B[30][30];
```

maydon kodlarini kiritamiz.

Int tipidagi k va V o'zgaruvchilarini hosil qilamiz.

Rasmni ochish uchun sButton1ga quyidagicha kodlarni yozamiz(Bu kod bmp formatli rasmni OpenPictureDialog1 komponentasi orqali ochadi va biz hosil qilgan kataklarga bo'laklagan holda joylashtiradi. Bundan tashqari rasm ochilgan payt bu rasm Image1 ga yaxlit holda chiqariladi):

```
if(OpenPictureDialog1->Execute()==true)
```

```
{  for(j=0;j<Maydon_olchami_1;j++)
  {   for(i=0;i<Maydon_olchami_1;i++)
   {    delete A[i][j];
    delete B[i][j];
    A[i][j]=NULL;
    B[i][j]=NULL;
    delete PBitmap[j*Maydon_olchami_1+i];
    PBitmap[j*Maydon_olchami_1+i]=NULL;   }  }   Oyin_ishla();
MBitmap1 = new Graphics::TBitmap();
MBitmap1->Height=340;
MBitmap1->Width=340;
 try   {   MBitmap1->LoadFromFile(OpenPictureDialog1->FileName);
  Image1->Picture->LoadFromFile(OpenPictureDialog1->FileName);
  for(j=0;j<Maydon_olchami;j++)   {   for(i=0;i<Maydon_olchami;i++)
   {    if((i==Maydon_olchami-1)&&(j==Maydon_olchami-1))    {
A[i][j]->status=2;
    A[i][j]->SetA(i);
    A[i][j]->SetB(j);
    A[i][j]->X=IntToStr(j*Maydon_olchami+i+1);
    AS=i;
    BS=j;
    A[i][j]->Paint();
    B[i][j]->status=2;
    B[i][j]->SetA(i);
    B[i][j]->SetB(j);
    B[i][j]->X=IntToStr(j*Maydon_olchami+i+1);
    AS=i;
    BS=j;   }   else   {   A[i][j]->status=1;
    Alisher1(MBitmap1,A[i][j]->Bitmap1,i,j);
    A[i][j]->X=IntToStr(j*Maydon_olchami+i+1);
    A[i][j]->SetA(i);
    A[i][j]->SetB(j);
    A[i][j]->Paint();
    B[i][j]->status=1;
   Alisher1(MBitmap1,B[i][j]->Bitmap1,i,j);
    B[i][j]->X=IntToStr(j*Maydon_olchami+i+1);
    B[i][j]->SetA(i);
    B[i][j]->SetB(j);   }   } } } catch (...)   {    MessageBeep(0);  }
  delete MBitmap1;  }
```

Rasmni almashtirishda esa quyidagi kodlar yoziladi(Bu kodda rasm almashtirish xuddi ikki o’lchovli massivlarda bajariladigan amallarkabi bajariladi):

```
RoyxatToplami();
randomize();
for(int j=0;j<Maydon_olchami;j++) {
```

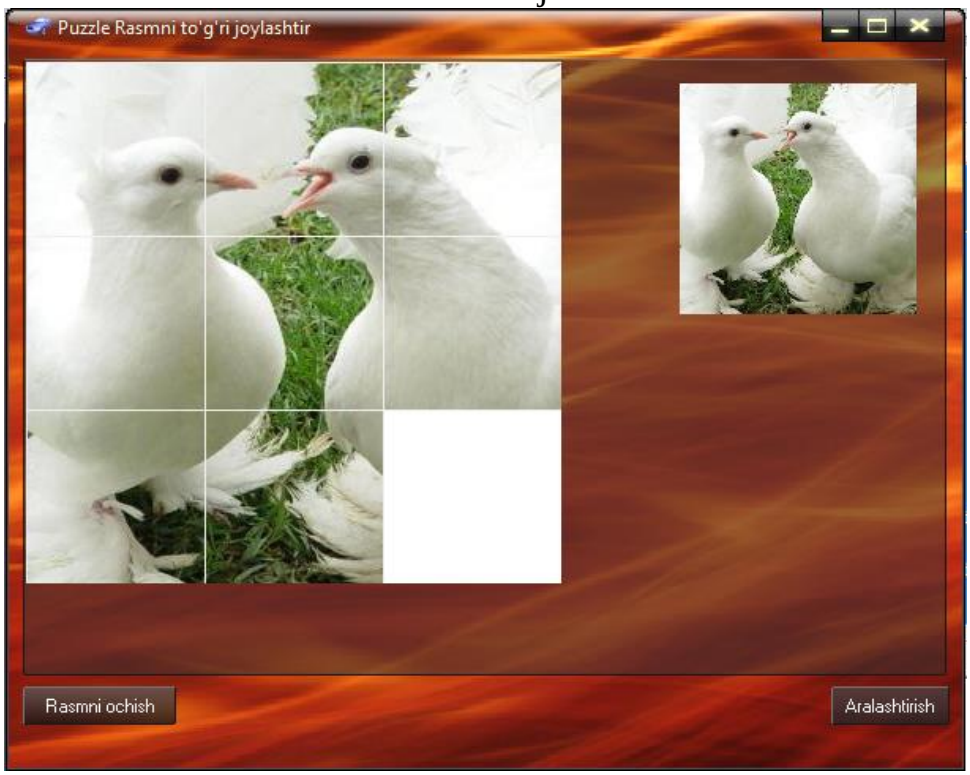
```
        for(int i=0;i<Maydon_olchami;i++)  { {
if((j*Maydon_olchami+i)==(Maydon_olchami*Maydon_olchami-1))
          {    A[i][j]->status=2;
            AS=i;
            BS=j;
            A[i][j]->X=IntToStr(Maydon_olchami*Maydon_olchami);
            A[i][j]->Paint();  }   else    {      A[i][j]->status=1;
              Alisher2=random(MyList->Count);
              AStruct=(PAList)MyList->Items[Alisher2];
              A[i][j]->Bitmap1=B[AStruct->x][AStruct->y]->Bitmap1;
              A[i][j]->X=IntToStr(AStruct->I);
              MyList->Delete(Alisher2);
              A[i][j]->Paint();      }   }  }  }
        MeningRoyxatim();
        int bp=0;
        bp=(Alisher());
        if((bp%2)!=0)  {    TPole *C=new TPole(this);
            Graphics::TBitmap *CBitmap = new Graphics::TBitmap();
            C->Bitmap1=CBitmap;
            C->Bitmap1= A[Maydon_olchami-2][Maydon_olchami-1]->Bitmap1;
            C->X=A[Maydon_olchami-2][Maydon_olchami-1]->X;
            A[Maydon_olchami-2][Maydon_olchami-1]-
>Bitmap1=A[Maydon_olchami-3][Maydon_olchami-1]->Bitmap1;
            A[Maydon_olchami-2][Maydon_olchami-1]-
>Bitmap1=A[Maydon_olchami-3][Maydon_olchami-1]->Bitmap1;
            A[Maydon_olchami-2][Maydon_olchami-1]->X=A[Maydon_olchami-
3][Maydon_olchami-1]->X;
            A[Maydon_olchami-3][Maydon_olchami-1]->Bitmap1=C->Bitmap1;
            A[Maydon_olchami-3][Maydon_olchami-1]->Bitmap1=C->Bitmap1;
            A[Maydon_olchami-3][Maydon_olchami-1]->X=C->X;
            delete CBitmap;
            delete C;
            bp=(Alisher());
            A[Maydon_olchami-3][Maydon_olchami-1]->Paint();
            A[Maydon_olchami-2][Maydon_olchami-1]->Paint(); }
```

Budastur kodi uchun Main.h kutubxonasida ba’zi klass va strukturalar yaratiladi:

```
        #ifndef MainH
        #define MainH
        #include <Classes.hpp>
        #include <Controls.hpp>
        #include <StdCtrls.hpp>
        #include <Forms.hpp>
        #include <Menus.hpp>
        #include <ExtCtrls.hpp>
```

```
#include <Graphics.hpp>
#include <Dialogs.hpp>
#include <ExtDlgs.hpp>
#include "sButton.hpp"
#include "sPanel.hpp"
#include "sSkinProvider.hpp"
#include "sSkinManager.hpp"
#define CM_XXX WM_APP+2
#define CM_XX1 WM_APP+3
typedef struct AList   {  int I;
int x;
int y;  }   TAList;
typedef TAList* PAList;
class TForm1 : public TForm     {   __published:
        TsPanel *sPanel1;
        TImage *Image1;
        TOpenPictureDialog *OpenPictureDialog1;
        TsButton *sButton2;
        TsButton *sButton1;
        TsSkinManager *sSkinManager1;
        void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
        void __fastcall FormKeyDown(TObject *Sender, WORD &Key,
          TShiftState Shift);
        void __fastcall sButton1Click(TObject *Sender);
        void __fastcall sButton2Click(TObject *Sender);
private:
int AS,BS;
int i,j;
        Graphics::TBitmap *MBitmap1;
        Graphics::TBitmap *PBitmap[900];
        void __fastcall Oyin_ishla();
        void __fastcall Alisher1(Graphics::TBitmap
*Asosiy,Graphics::TBitmap *Qoshimcha,int k,int j);
        void __fastcall WndProc(Messages::TMessage &Message);
        void __fastcall RoyxatToplami();
        void __fastcall MeningRoyxatim();
public:                 // foydalanuvchi e'lonlari
int Maydon_olchami,Maydon_olchami_1,Alisher2;
TList *MyList;
PAList AStruct;
PAList A1Struct;
     __fastcall TForm1(TComponent* Owner);
      int Alisher(void);  };
extern PACKAGE TForm1 *Form1;
#endif
```

Dastur natijasi.



2.3.rasm. Rasmni to’gri joylashtirilganligi

Nazorat savollari

1. Hisoblash murakkabligi nima?
2. Murakkablik sinflari nechta va ular qaysilar?
3. NP sinfini tushuntiring.

3 - Laboratoriya ishi.

Mavzu. To’g’ri zanjir usuli.

Ishdan maqsad: Qoidalarning to’g’ri zanjiri tushunchasi o’rganish va qoidalarning to’g’ri zanjir usuli asosida hosil yetishtirishni tahlil qiluvchi dastur ishlab chiqish.

Masalaning qo’yilishi: Hosil yetishishi natijasini faslning kelishiga va hosilga qarashga qarab tahlil qiluvchi dastur tuzilsin.

Uslubiy ko’rsatmalar: Sun’iy intellekt tizimining an’anaviy dasturiy tizimlaridan asosiy farqi shundan iboratki, uning tuzilishining tarkibiy qismlari bo’lingan holda aniqlanadi va uning istalgan qismini zamonaviylashtirish umumiy tuzilmaga ta’sir qilmaydi.

Insonning miyasi hatto eng oddiy masalani echishga kirishganda ham kerakli harakatlarni tanlash uchun uning ixtiyorida axborotlarning katta hajmi mavjud . Masalan, ishga keta turib, odam uydan chiqadi va ko’chani kesib o’tadi, ammo u o’tish uchun paytni tanlaguniga qadar, uning miyasi ham harakatning jadalligini ham transportning tezligini, hamma qarama-qarshi tomongacha bo’lgan masofani

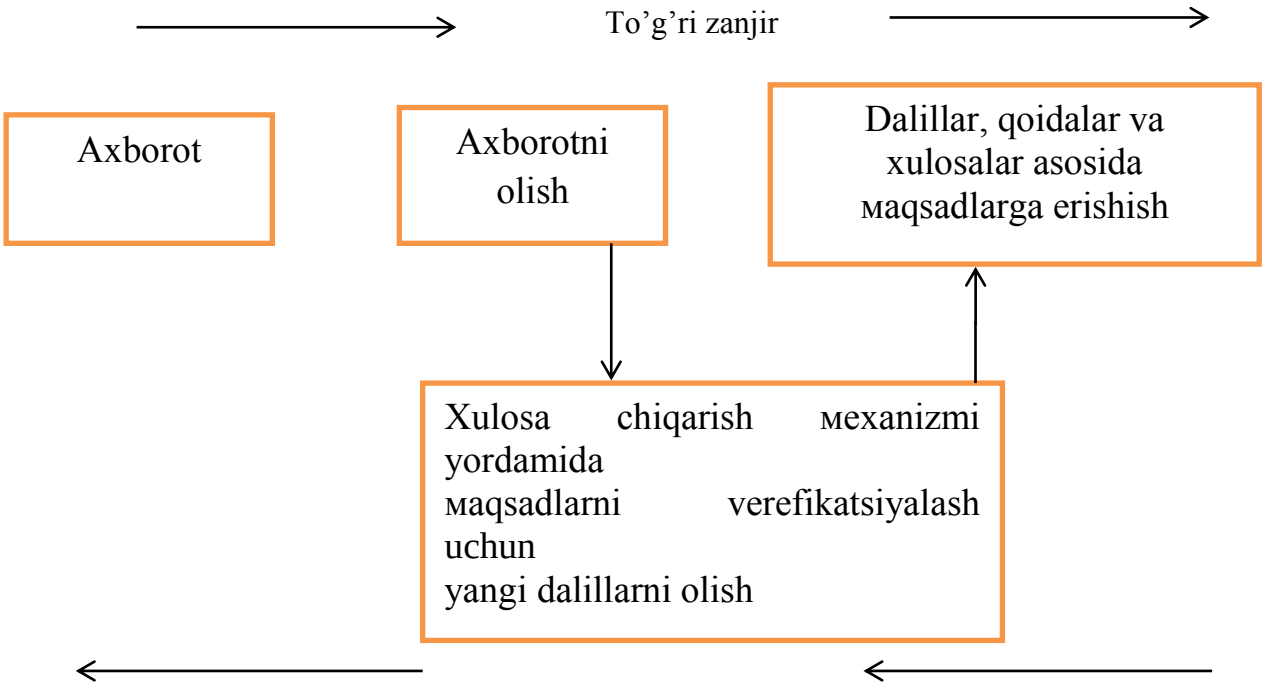
tahlil qiladi. shuning bilan bir vaqtda miya ko’chani kesib o’tishga to’g’ridan to’g’ri tegishli bo’lmagan axborotlarni, ya’ni ob-xavoni, o’tib ketayotgan mashinalarning rangini ishlab chiqadi. Bundan tashqari odam qaerga ketayotganligi, u erga qancha vaqtda etib borishi, kim bilan uchrashishi kerakligi haqida uylaydi. Ammo agar odam ko’chani kesib o’tishdan avval ko’chani kesib o’tish maqsadiga bilvosita va bevosita aloqador barcha dalillarni tahlil qilganda, u bir necha yil turib qolishi mumkin edi. shunday qilib, inson miyasiga ham aniq vaziyatlarda to’g’ri reaktsiya tanlashga rahbarlik qiluvchi murakkab tizim mavjuddir. Bunday tanlov soddalashtirish deb ataladi. Soddalashtirish mexanizmi ushbu paytda echilaYotgan vazifaga aloqasi bo’lmagan dalillarni to’sib qo’yadi.

Sun’iy intellekt tizimlarini loyihalashtirishda birinchi bosqichda unga erishish uchun mo’ljallangan maqsad aniqlanadi, echilayotgan vazifalarni yirik atamalarda bayon qilishni bilish uchun zarur sinfi belgilaniladi. Dalillar sun’iy intellekt tizimining muhim qismi bo’ladi, ularsiz maqsadga erishish mumkin emas. Har bir maqsadning o’zining dalillari bor. Har bir dalil o’zining salmog’iga ega, ya’ni har bir dalilga nisbatan muhimlik hosdir. Ushbu vazifani echish uchun dalil qanchalik katta ahamiyatga ega bo’lsa, uning salmog’i shunchalik katta. Maqsadlarga erishish uchun zarur bo’lgan umumiy dalillar aniqlangandan keyin, aniq ma’lumotlarni olish kerak. Ma’lumotlarni olish uchun tegishli savollar shakllantiriladi, ularga javoblar tizimini yakuniy qarorga kelishiga yordam beradi. Sun’iy intellect tizimlari uchun dasturni ishlab chiqish quyidagi bosqichlardan iborat.

1. Maqsadlarni aniqlash.
2. Bu maqsadlarga tegishli dalillarni aniqlash.
3. Ushbu vaziyat uchun hos bo’lgan dalillarga mos bo’lgan ma’lumotlarni olish.
4. Xulosa chiqarish qoidalari va mexanizmidan foydalanish bilan ma’lumotlarni olish.

Ma’lum dalillar, qoidalarga ko’ra berilgan vaziyatga nisbatan qo’llaniladi. qoidalar ma’lumotlarni to’g’ri baholash va maqsadga erishishga yordamlashadi.

Yuqoridagi bayon qilingan bosqichlar yordamida maqsadlarga erishish jarayoni mulohazalarning to’g’ridan-to’g’ri zanjiri, ya’ni ma’lumotlardan mantiqiy xulosaga boruvchi zanjir deb ataladi. Maqsadga erishilganda uning to’g’riligini tekshirish, ya’ni ma’lumotlar va qoidalar bilan yana vazifani tahlil qilish zarurdir. Bu jaraYon maqsadlarni verifikatsiyalash deb ataladi. Yana avvaldan to’g’ri deb faraz qilingan xulosani tasdiqlash uchun yangi ma’lumotlarni chiqarish mexanizmlarini tanlash teskari zanjirga misol bo’lishi mumkin. shunday qilib, mulohazalarning teskari zanjiri to’g’ri zanjirga qarama qarshi tomonga, ya’ni xulosadan ma’lumotlarga ketadi. Teskari zanjir faqat maqsadga erishilgandan keyin vujudga keladi. 1-rasmda sun’iy intellekt tizimining konfiguratsiyasi va ishi berilgan.



3.1.rasm. Sun’iy intellekt tizimining konfiguratsiyasi va ishi

To’g’ri zanjir usulida ma’lumotlar qayta ishlanishi jarayonida natijalar ma’lumotlarning o’zaro to’g’ri zanjirli aloqasi asosida chiqariladi. Bunda ma’lumotlarni qayta ishlash bilimlar bazasidagi ma’lumotlar asosida qayta ishlanadi.

Buning uchun hosil yetishishi natijasini tahlil qiluvchi dasturni ko’rib chiqamiz. Bunda dastur ishlashi quyidagicha bo’ladi: dasturda hosil yetishtirishga bog’liq bo’lgan faktorlar beriladi va siz bu faktorlarni keraklisini kiritasiz. Dastur esa o’zidagi bilimlar bazasi asosida berilgan ma’lumotlarni qayta ishlaydi va hosilning qay darajada yetishishini foiz hisobida chiqarib beradi.

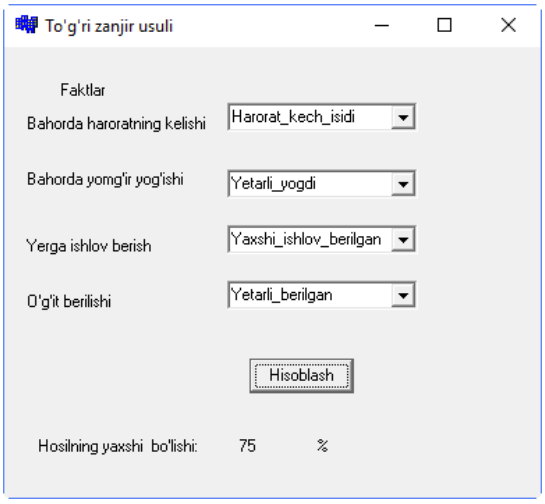
Yaxshi hosil olish uchun to’g’ri zanjir usuli

6. Agar bahorda o'z vaqtida havo ilisa, u holda hosil vaqtida yetilishi mumkin.
7. Agar bahorda yetarli darajada yomg'ir yog'sa, u holda hosil yaxshi bo'lishi mumkin.
8. Agar kuzda yerlar yaxshi haydalib, ishlov berilgan bo'lsa, u holda hosil to'liq bo'lishi mumkin.
9. Agar yerga o'g'it yetarli berilsa, u holda ekin quvvatli bo'lib, holsil to'la bo'lishi mumkin.
10. Agar yuqoridagilarni barchasi bo’lsa, hosil aniq yaxshi bo'ladi.

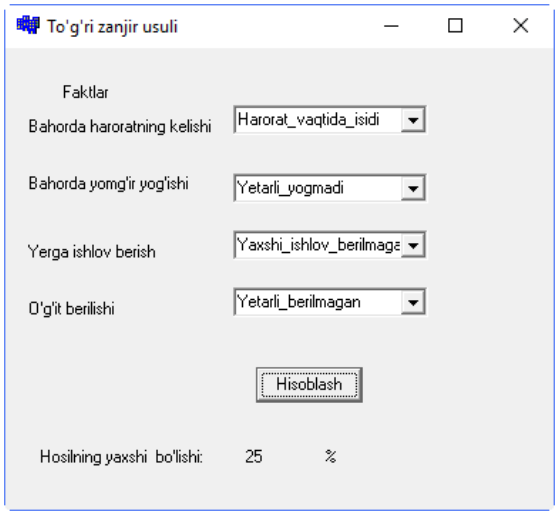
Dastur kodi va natijasi.

```
#include <vcl.h>
    #pragma hdrstop
    #include "Unit1.h"
    #pragma package(smart_init)
```

```
        #pragma resource "*.dfm"
TForm1 *Form1;
void __fastcall TForm1::Button1Click(TObject *Sender)
        {float a = 0, b = 0, c = 0, d = 0, f = 4;
        if(ComboBox1->Text == "Harorat_vaqtida_isidi"&&
        ComboBox2->Text == "Yetarli_yogdi"&&
        ComboBox3->Text == "Yaxshi_ishlov_berilgan"&&
        ComboBox4->Text == "Yetarli_berilgan"
        ){      a = 1; b=1; c = 1; d = 1;
             Label7->Caption = FloatToStr(((a + b + c + d)/f)*100);}
        if(ComboBox1->Text == "Harorat_kech_isidi"&&
        ComboBox2->Text == "Yetarli_yogdi"&&
        ComboBox3->Text == "Yaxshi_ishlov_berilgan"&&
        ComboBox4->Text == "Yetarli_berilgan"
        ){       a = 0; b=1; c = 1; d = 1;
             Label7->Caption = FloatToStr(((a + b + c + d)/f)*100);}
        if( ComboBox1->Text == "Harorat_vaqtida_isidi"&&
        ComboBox2->Text == "Yetarli_yogmadi"&&
        ComboBox3->Text == "Yaxshi_ishlov_berilgan"&&
        ComboBox4->Text == "Yetarli_berilgan"
        ){      a = 1; b=0; c = 1; d = 1;
             Label7->Caption = FloatToStr(((a + b + c + d)/f)*100); }
        if(ComboBox1->Text == "Harorat_vaqtida_isidi"&&
        ComboBox2->Text == "Yetarli_yogdi"&&
        ComboBox3->Text == "Yaxshi_ishlov_berilmagan"&&
        ComboBox4->Text == "Yetarli_berilgan"
        ){      a = 1; b=1; c = 0; d = 1;
             Label7->Caption = FloatToStr(((a + b + c + d)/f)*100);
        }if(ComboBox1->Text == "Harorat_vaqtida_isidi"&&
        ComboBox2->Text == "Yetarli_yogdi"&&
        ComboBox3->Text == "Yaxshi_ishlov_berilgan"&&
        ComboBox4->Text == "Yetarli_berilmagan"
        ){      a = 1; b=1; c = 1; d = 0;
             Label7->Caption = FloatToStr(((a + b + c + d)/f)*100); }
        if(ComboBox1->Text == "Harorat_vaqtida_isidi"&&
        ComboBox2->Text == "Yetarli_yogmadi"&&
        ComboBox3->Text == "Yaxshi_ishlov_berilmagan"&&
        ComboBox4->Text == "Yetarli_berilmagan")
        {       a = 1; b=0; c = 0; d = 0;
             Label7->Caption = FloatToStr(((a + b + c + d)/f)*100); } }
```



3.2.rasm. To’gri zanjir.



3.3.rasm. To’g’ri zanjir natijasi

Nazorat savollari

1. To’g’ri zanjir usulida masalalarni yechishni tushuntiring
2. To’g’ri zanjir usulida masalalardagi ma’lumotlar o’zaro qanday bog’langan?

4 - Laboratoriya ishi.

Mavzu: Qoidalar tizimini ishlab chiqish. Poker o’yini.

Ishdan maqsad: O'yinlar yaratishda qoidalar tizimidan foydalanishni o'rganish.

Masalaning qo’yilishi: Qoidalar tizimi yordamida poker o’yini ishlab chiqilsin.

Uslubiy ko’rsatmalar: Odam bilimlarni fikrlash usulini o’zgartirmasdan, ma’lum bo’lgan dalillarni esdan chiqarmasdan jamlashi mumkin. Sun’iy intellekt tizimi xuddi shunday ishlab chiqiladi. Bunda inson xotirasining bloklariga o’xshab dasturlarning ayrim qismlarining yuqori mustaqilligiga erishiladi.

Odamning miyasi kerakli axborotni tanlab turib, faqat ushbu muammoga tegishli bo’lgan dalillarni ulaydi, bunda u kirishi mumkin bo’lgan barcha ma’lumotlardan foydalanmaydi. Inson faoliyatining asosida fikrlash yotadi va ushbu holda fikrlash jarayonining maqsadi yakuniy natija bo’ladi. Bitta maqsadga erishilgandan keyin yangi maqsad qo’yiladi va erishiladi. Maqsadlarni mahalliy va asosiyga bo’lish mumkin. shundan kelib chiqqan holda intellektning ta’rifini berish mumkin.

Intellekt dalillarning majmuasi va ularni belgilangan maqsadga erishish uchun qo’llash usullaridan iborat bo’ladi. Maqsadga erishish esa - bu tegishli omillardan foydalanishning kerakli qoidalarini qo’llanishidir.

Inson har qanday amal yoki jarayonni boshlashdan avval uning algoritmi, qoidalar tizimini ishlab chiqadi. Qoidalar tizimini ishlab chiqish eng muhim amallardan biri hisoblanadi. Shuning uchun ham sun'iy intellekt asosida ishlovchi

ko'pgina dastur, qurilma va tizimlarga o’z yo'nalishiga mos keluvchi qoidalar tizimi kiritiladi. Bu jarayonni dasturiy ta’minotni yaratish jarayonidagi qadamlar orqali o’rganamiz.

Dasturiy ta’minotni yaratish jarayoni bir qancha ost jarayonlardan tashkil topadi. Bu ostjarayonlardan bir qanchasi quyida berilgan. Sharshara modelida bu ostjarayonlar o’zaro ketma-ket amalga oshiriladilar, boshqa shunga o’xshash jarayonlarda ularning ketma-ketligi va tarkibi o’zgaradi.

- Talablarni tahlillash.
- Dasturiy ta’minotni loyihalashtirish.
- Dasturiy ta’minotni testlash
- Tizim integratsiyasi.
- Dasturiy ta’minotni ishlatish.
- Dasturiy ta’minotni kuzatib boorish

Talablarni tahlillash – dasturiy ta’minotga bo’lgan talablar to’plamini o’z ichiga oluvchi dasturiy ta’minotni ishlab chiqishni o’z ichiga oluvchi jarayon. Talablarni tahlillash 3ta turni o’z ichiga oladi.

- Talablarni to’plash – foydalanuvchilar bilan suhbatlashib, ularning talab va xohishlarini o’rganish.
- Talablarni tahlillash – yig’ilgan ma’lumotlar ichida keraksizi, bir qiymatlisi, to’liqmaslari kabilar bor yo’qligini tekshirish va ularni hal etish, o’zaro bog’liq talablarni aniqlash.
- Talablarni hujjatlashtirish – talablar turli xil ko’rinishlarda hujjatlashtirilishi mumkin. Oddiy tavsiflar yordamida, foydalanish senariysi ko’rinishida, foydalanuvchi tarixi ko’rinishida yoki jarayonlar spetsifikatsiyasi ko’rinishida.

Dasturiy ta’minotni loyihalash – dasturiy ta’minotning o’zidan kelib chiqqan holda qanday usul, algoritm bilan yaratish mumkinligini aniqlash va dasturiy ta’minot loyihasini yaratish jarayoni. Loyihalash maqsadi tizimning ichki xususiyatlarini aniqlash va uning tashqi xususiyatlarini dastiriy ta’minotga qo’yilgan talab asosida detallashtirish.

Aynan dasturni loyihalash jarayonida dasturda ishlovchi algoritmlarning qoidalar tizimi ishlab chiqiladi. Ya’ni dasturiy ta’minotda qo’llangan algoritmlarning qay biri qanday vaziyatda ishlashi belgilab beriladi. Qoidalar tizimini ishlab chiqish sun’iy intellektni yaratishning asosiy bosqichlaridan biri hisoblanadi. Masalan, shaxmat o’yini dasturini olaylik. Unda foydalanuvchi va kompyuter orasida o’yin o’ynash jarayoni amalga oshirilsin. Bunda dastur kodiga ko’p miqdordagi kombinatsion yurishlar algoritmi kiritilgan bo’ladi. Bu esa dasturning qoidalar tizimiga foydalanuvchi yurgan kombinatsiyaga qarshi kombinatsiyadan foydalanish imkonini beradi. Ya’ni masalan, farzin a:1 dan f:1ga yursa, dastur qoidalar tizimidan kelib chiqqan holda piyoda bilan f:1 ga yurib farzinni oladi. Bu jarayonda hozirda shaxmat o’yinida mavjud bo’lgan barcha yurish kombinatsiyalari dasturning bazasiga kiritib qo’yilgan bo’ladi. O’yin davomida dastur foydalanuvchi tomonidan kiritilgan kombinatsiyaga qarshi o’z kombinatsiyasini bazasidagi mavjud kombinatsiyalarni tahlil qilib, ular orasidan

keraklisini tanlab oladi. Qoidalar tizimining ishlab chiqilishi orqali dastur sun’iy intellectga aylantiriladi.

Misolni ko’rib chiqamiz:

Dalil 1. yoqib qo’yilgan plita issik.

Qoida 1. Agar yoqib qo’yilgan plitaga qo’lni qo’ysa, unda kuyib qolish mumkin.

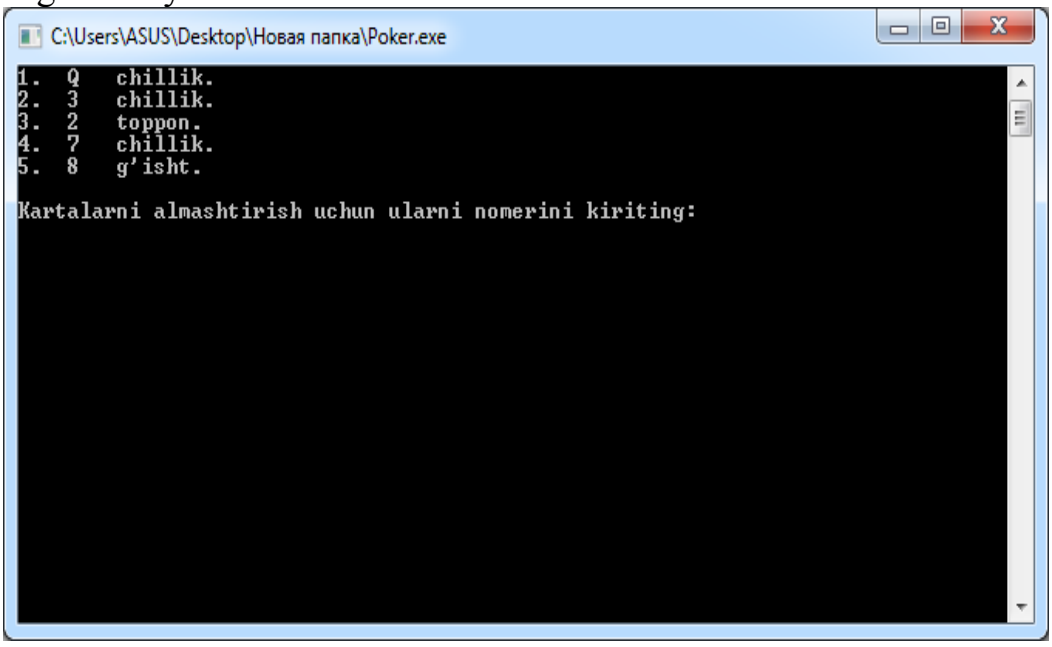
Dalil 2. Tig’iz paytda ko’chada mashinalar ko’p.

Qoida 2. Agar tig’iz paytda ko’chani kesib o’tishga harakat qilinsa, unda mashina ostiga tushib qolish mumkin.

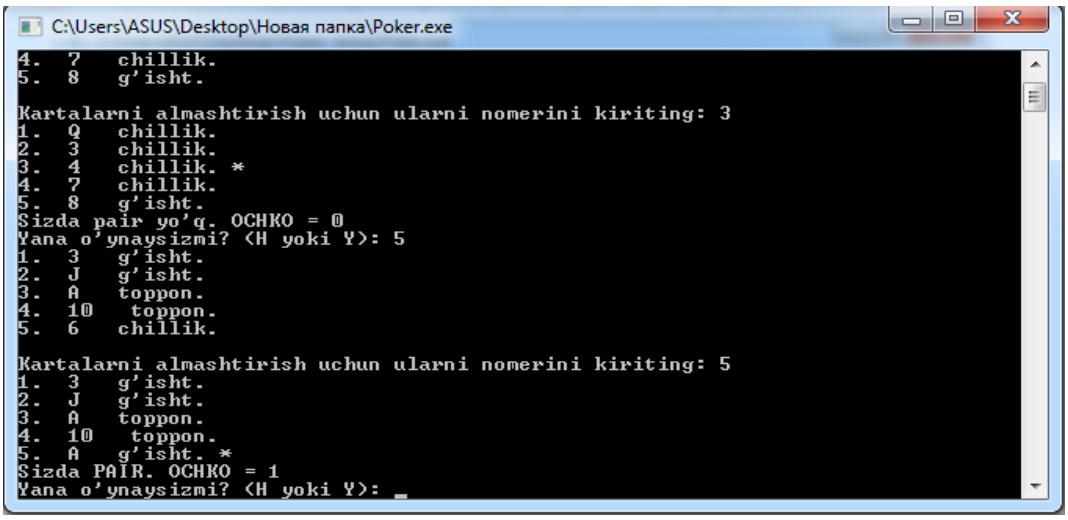
Qoidalar tizimining ishlashini tushunish uchun poker o'yini dasturini ishlashini ko'rib chiqamiz. Dastlab poker o’yini qanday o’yin ekanligini bilib olish kerak. Poker o’yinida ikkitadan kam bo’lmagan o’yinchilar to’rt yoki beshtadan kartalarni ishlatgan holda eng yuqori poker kombinatsiyasini yig’ishga harakat qiladi. Pokerda 32talik, 36talik va 52talik kartalardan foydalanish mumkin. Kartalarning qiymati “tuz”dan boshlab kamayib boradi va shu tariqa karol, dama va hk 2gacha. Pokerning turiga qarab o’yin bir necha fazalardan iborat. Ularning har biri karta tarqatish bilan boshlanadi. Karta tarqatilgandan so’ng o’yinchi xohlasa biror narsa evaziga o’yinni davom ettirishi yoki o’yindan chiqib ketishi mumkin. Eng yaxshi kombinatsiyali karta egasi yoki qolgan o’yinchilarni chiqarib yuborgan o’yinchi g’olib hisoblanadi.

O’yinni tushuntirilishi davomida sezgan bo’lsangiz o’yin dasturini tuzishda uchta asosit algoritm ishlatiladi: O’yinchilarga kartalarni tasodifiy holda tarqatish, o’yinchilarning kartalarini o’sish tartibida saralash va ularni o’zaro solishtirish. Solishtirish jarayonida dastlab eng yuqorida turgan kartalar solishtiriladi. Agar ular teng bo’lsa, solishtirish davom etadi, aks holda kimning kartasi kichik bo’lsa shu o’yinchi chiqarib yuboriladi.

O’yinning bosh oynasi:



4.1.rasm. Nomer kiritish



4.2.rasm. Natijalarni ko'rish

```
#include <string>
#include "card.h"
Card::Card(){}
Card::Card(int rank, int suite)
{    this->rank = rank;
   this->suite = suite;
}void Card::setRank(int rank)
{    this->rank = rank;
}void Card::setSuite(int suite)
{    this->suite = suite;
}int Card::getRank()
{    return this->rank;
}int Card::getSuite()
{    return this->suite;
}
std::string Card::display()
{
   static const std::string aRanks[] = {" 2", " 3", " 4", " 5", " 6",
         " 7", " 8", " 9", " 10", " J", " Q", " K", " A"};
   static const std::string aSuits[] = {"chillik", "g'isht", "toppon", "qarg'a"};
   return aRanks[rank] + "   " + aSuits[suite] + ".";
}
#include <iostream>
#include <ctime>
#include <string>
#include <cstdlib>
#include <algorithm>
#include "desk.h"
#include "card.h"
Deck::Deck()
{    srand(time(0));
   for(int i = 0; i < 52; ++i){        cards[i] = i;
```

```
    }    shuffle();}
void Deck::shuffle()
{    iCard = 0;
    std::random_shuffle(cards, cards + 52);
}
Card Deck::dealACard()
{    if(iCard > 51)
    {        std::cout << std::endl << "Aralashtirilmoqda..." << std::endl;
        shuffle();
    }    int r = cards[iCard] % 13;
    int s = cards[iCard++] / 13;
    return Card(r, s); }
```

Nazorat savollari

1. Qoidalar tizimi qanday ishlab chiqiladi?
2. Qoidalar tizimini ishlab chiqishda nimalardan foydalaniladi?

5 - Laboratoriya ishi.

Mavzu: Qoidalar tizimini ishlab chiqish. Oila a`zolari o`rtasidagi ishlab chiqish.

Ishdan maqsad: Qoidalar tizimini ishlab chiqishni o'rganish. Oila a`zolari o`rtasida munosabatlarni ishlab chiqish .

Masalaning qo'yilishi: Oila a'zolari ma'lumotlaridan foydalanib oila shajarasi dasturi ishlab chiqilsin.

Uslubiy ko'rsatmalar: Insonning miyasi hatto eng oddiy masalani echishga kirishganda ham kerakli harakatlarni tanlash uchun uning ixtiyorida axborotlarning katta hajmi mavjud . Masalan, ishga keta turib, odam uydan chiqadi va ko'chani kesib o'tadi, ammo u o'tish uchun paytni tanlaguniga qadar, uning miyasi ham harakatning jadalligini ham transportning tezligini, hamma qarama-qarshi tomongacha bo'lgan masofani tahlil qiladi. shuning bilan bir vaqtda miya ko'chani kesib o'tishga to'g'ridan to'g'ri tegishli bo'lmagan axborotlarni, ya'ni ob-xavoni, o'tib ketayotgan mashinalarning rangini ishlab chiqadi. Bundan tashqari odam qaerga ketayotganligi, u erga qancha vaqtda etib borishi, kim bilan uchrashishi kerakligi haqida uylaydi. Ammo agar odam ko'chani kesib o'tishdan avval ko'chani kesib o'tish maqsadiga bilvosita va bevosita aloqador barcha dalillarni tahlil qilganda, u bir necha yil turib qolishi mumkin edi. shunday qilib, inson miyasiga ham aniq vaziyatlarda to'g'ri reaktsiya tanlashga rahbarlik qiluvchi murakkab tizim mavjuddir. Bunday tanlov soddalashtirish deb ataladi. Soddalashtirish mexanizmi ushbu paytda echilaYotgan vazifaga aloqasi bo'lmagan dalillarni to'sib qo'yadi.

Maqsadga erisha turib, inson nafaqat oldiga qo'yilgan vazifani echimiga keladi, balki bir vaqtda yangi bilimlarni oladi. Masalan, Jon va Meri – Jeyinning ota-onalari. Maqsad - Jim va Jeyn bir birlariga kim bo'lishlarini aniqlashdir.

Soddalashtirish mexanizmi odamga uning xotirasida saqlanayotgan quyidagi qoidaga murojaat qilishga majbur qiladi: Agar qiz va o'g'il bolada bitta ota-onalar bo'lsa, unda o'g'il va qiz bola aka va singil. Bu maqsadga erishish jarayonida yangi dalil - Jim va Jeyin aka va singil ekanligi aniqlandi.

Intellektning yangi dalillarni chiqarib olishga yordamlashadigan qismi xulosa chiqarish mexanizmi deb ataladi. Xuddi xulosa chiqarish mexanizmi insonga tajribadan o'rganishiga imkon beradi va mavjud bilimlarni yangi vaziyatga qo'llab, mavjudlarda yangi dalillarni generatsiyalash imkonini beradi.

Intellekt dalillarning majmuasi va ularni belgilangan maqsadga erishish uchun qo'llash usullaridan iborat bo'ladi. Maqsadga erishish esa - bu tegishli omillardan foydalanishning kerakli qoidalarini qo'llanishidir.

Misolni ko'rib chiqamiz:

Dalil 1. Devor ostidagi elektr simidan elektr toki oqmoqda va devor nam.

Qoida 1. Agar devorga tegsangiz sizni tok uradi.

Dalil 2. Osmonda qora bulutlar suzib yuribdi

Qoida 2. Agar yomg'irpo'sh olmasangiz, yomg'ir ostida qolib ketasiz.

Dalil 3. Kavsharlash apparati ishlatilishi paytida chiquvchi nur ko'z pardasini keskin zararlaydi.

Qoida 3. Agar ko'zga himoyalovchi ko'zoynak taqmasangiz ko'zingiz zararlanadi.

Sun'iy intellekt tizimlarini loyihalashtirishda birinchi bosqichda unga erishish uchun mo'ljallangan maqsad aniqlanadi, echilayotgan vazifalarni yirik atamalarda bayon qilishni bilish uchun zarur sinfi belgilaniladi. Dalillar sun'iy intellekt tizimining muhim qismi bo'ladi, ularsiz maqsadga erishish mumkin emas. Har bir maqsadning o'zining dalillari bor. Har bir dalil o'zining salmog'iga ega, ya'ni har bir dalilga nisbatan muhimlik hosdir. Ushbu vazifani echish uchun dalil qanchalik katta ahamiyatga ega bo'lsa, uning salmog'i shunchalik katta. Maqsadlarga erishish uchun zarur bo'lgan umumiy dalillar aniqlangandan keyin, aniq ma'lumotlarni olish kerak. Ma'lumotlarni olish uchun tegishli savollar shakllantiriladi, ularga javoblar tizimini yakuniy qarorga kelishiga yordam beradi. Sun'iy intellect tizimlari uchun dasturni ishlab chiqish quyidagi bosqichlardan iborat.

1. Maqsadlarni aniqlash.
2. Bu maqsadlarga tegishli dalillarni aniqlash.
3. Ushbu vaziyat uchun hos bo'lgan dalillarga mos bo'lgan ma'lumotlarni olish.
4. Xulosa chiqarish qoidalari va mexanizmidan foydalanish bilan ma'lumotlarni olish.

Ma'lum dalillar, qoidalarga ko'ra berilgan vaziyatga nisbatan qo'llaniladi. qoidalar ma'lumotlarni to'g'ri baholash va maqsadga erishishga yordamlashadi.

Oila a`zolari o`rtasidagi qoidalar ishlab chiqishda biz turli mantiqiy operattorlardan foydalanamiz . Bu operatorlarga agar , va , yoki , emas kabilar kiradi. Quyidagini misol qilib keltirishimiz mumkin :
Agar (va (ota-onasi (?x) (?y), ota-onasi (?x) (?z)),
U holda (qarindosh (?y) (?z))).

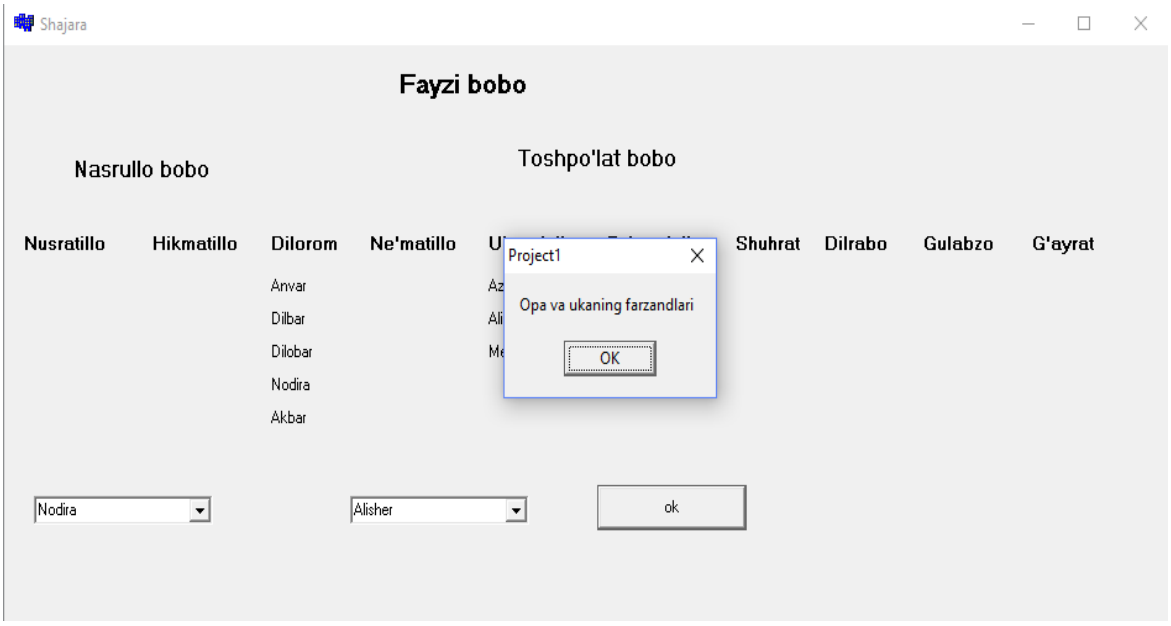
Mana shu qoidalarga asosan biz oila a`zolari o`rtasida qonuniyatlarni ishlab chiqamiz .Masalan

Agar (va (akasi (?x) (?y) , akasi (?y) (?z)),u holda (qarindosh (?x) (?z)).
Yoki quyidagicha
Ota-onasi Nusratillo Oyqiz va Ota-onasi Ubaydullo Gulbahor
Demak quyidagicha bo`ladi :
qarindosh Aziz Mahfuza.

Emas operatoriga esa quyidagicha misol keltirsak bo`ladi :
Va ((?x) qush, Emas ((?x) pingvin emas))
Yoki teskarisi:
Va (Emas ((?x) pingvin), (?x) qush).
Va quyidagilarni misol qilamiz :

- aka x y: x aka u holda (nima bo`lgan taqdirda ham ularning ota onalari bitta)
- singil x y: x singil u holda (ularning ota –onalari bitta)
- oyi x y: x demak onasi y
- dada x y: x bor dada y
- o`g`il X Y: x demak o`g`li y
- qiz x y: x demak qizi y
- jiyan x y: x va y demak aka(ota - ona x va ota ona y)
- nabira x y: x nabira y

5.1.rasm. Dasturni ochish oynasi



5.2.rasm. Natijani topish

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
void __fastcall TForm1::Label1Click(TObject *Sender)
{
if(Label2->Visible == false){Label2->Visible = true;Label3->Visible = true;}
else
{
Label2->Visible = false;Label3->Visible = false;}}
void __fastcall TForm1::Label2Click(TObject *Sender)
{
if(Label4->Visible == false){
Label4->Visible = true;Label5->Visible = true;Label6->Visible = true;Label7-
>Visible = true;
Label8->Visible = true;Label9->Visible = true;Label10->Visible = true;Label11-
>Visible = true;
Label12->Visible = true;Label13->Visible = true;}
else
{
Label4->Visible = false;Label5->Visible = false;Label6->Visible = false;Label7-
>Visible = false;
Label8->Visible = false;Label9->Visible = false;Label10->Visible =
false;Label11->Visible = false;
Label12->Visible = false;Label13->Visible = false;}}
void __fastcall TForm1::Label4Click(TObject *Sender)
```

```
{
if(Label14->Visible == false){
  Label14->Visible = true;Label15->Visible = true;Label16->Visible =
true;Label17->Visible = true;
Label18->Visible = true;Label19->Visible = true;
}
else
{
Label14->Visible = false;Label15->Visible = false;Label16->Visible =
false;Label17->Visible = false;
Label18->Visible = false;Label19->Visible = false;}}
void __fastcall TForm1::Label5Click(TObject *Sender)
{if(Label20->Visible == false){
  Label20->Visible = true ;  Label21->Visible = true  ;  Label22->Visible = true
;}
else{    Label20->Visible = false ;  Label21->Visible = false  ;  Label22->Visible
= false  ;}}
void __fastcall TForm1::Label6Click(TObject *Sender)
{if(Label23->Visible == false){
   Label23->Visible = true;Label24->Visible = true;Label25->Visible =
true;Label26->Visible = true;
Label27->Visible = true;
}else{
Label23->Visible = false;Label24->Visible = false;Label25->Visible =
false;Label26->Visible = false;
Label27->Visible = false;}}
void __fastcall TForm1::Label7Click(TObject *Sender)
{
if(Label28->Visible == false)
{    Label28->Visible = true;   Label29->Visible = true;
  }  else{    Label28->Visible = false;}   Label29->Visible = false;}
void __fastcall TForm1::Label8Click(TObject *Sender)
{
 if(Label30->Visible == false)
{
   Label30->Visible = true;Label31->Visible = true;Label32->Visible = true;
}else{
Label30->Visible = false;Label31->Visible = false;Label32->Visible = false;}}
void __fastcall TForm1::Label9Click(TObject *Sender)
{ if(Label33->Visible == false){   Label33->Visible = true;}else{
Label33->Visible = false;}       }
void __fastcall TForm1::Label10Click(TObject *Sender)
{ if(Label34->Visible == false)
{   Label34->Visible = true;Label35->Visible = true;Label36->Visible = true;}
```

```
else{Label34->Visible = false;Label35->Visible = false;Label36->Visible = false;}
}void __fastcall TForm1::Label11Click(TObject *Sender)
{
 if(Label37->Visible == false)
{    Label37->Visible = true;Label38->Visible = true;}
else{Label37->Visible = false;Label38->Visible = false;}}
void __fastcall TForm1::Label12Click(TObject *Sender){ if(Label39->Visible ==
false)
{    Label39->Visible = true;Label40->Visible = true;Label41->Visible =
true;Label42->Visible = true;
}else{
Label39->Visible = false;Label40->Visible = false;Label41->Visible =
false;Label42->Visible = false;}}
void __fastcall TForm1::Label13Click(TObject *Sender)
{if(Label43->Visible == false){
   Label43->Visible = true;
Label44->Visible = true;Label45->Visible = true;Label46->Visible =
true;Label47->Visible = true;
}else
{Label43->Visible = false;Label44->Visible = false;Label45->Visible =
false;Label46->Visible = false;
Label47->Visible = false;
}       }void __fastcall TForm1::Button1Click(TObject *Sender)
{if(ComboBox1->Text != ComboBox2->Text){
if((ComboBox1->Text == "Dilshod"||ComboBox1->Text ==
"Jo'shqin"||ComboBox1->Text == "Dilfuza"||
ComboBox1->Text == "Uchqun"||ComboBox1->Text == "Uyg'un"||ComboBox1-
>Text == "Mahfuza" )
&&  (ComboBox2->Text == "Aziz"||ComboBox2->Text ==
"Alisher"||ComboBox2->Text == "Mehribon")
)ShowMessage("Amakivachcha");
if((ComboBox1->Text == "Anvar"||ComboBox1->Text == "Dilbar"||ComboBox1-
>Text == "Dilobar"||
ComboBox1->Text == "Nodira"||ComboBox1->Text == "Akbar")
&&  (ComboBox2->Text == "Aziz"||ComboBox2->Text ==
"Alisher"||ComboBox2->Text == "Mehribon"))
ShowMessage("Opa va ukaning farzandlari");
if((ComboBox1->Text == "Dilshod"||ComboBox1->Text ==
"Jo'shqin"||ComboBox1->Text == "Dilfuza"||
ComboBox1->Text == "Uchqun"||ComboBox1->Text == "Uyg'un"||ComboBox1-
>Text == "Mahfuza" )
&&
  (ComboBox2->Text == "Anvar"||ComboBox2->Text == "Dilbar"||ComboBox2-
>Text == "Dilobar"||
ComboBox2->Text == "Nodira"||ComboBox2->Text == "Akbar")
```

```
)ShowMessage("Amakivachcha");
}
Else
{
  ShowMessage("Siz bir xil kishini tanladingiz!"); } }
```

Nazorat savollari

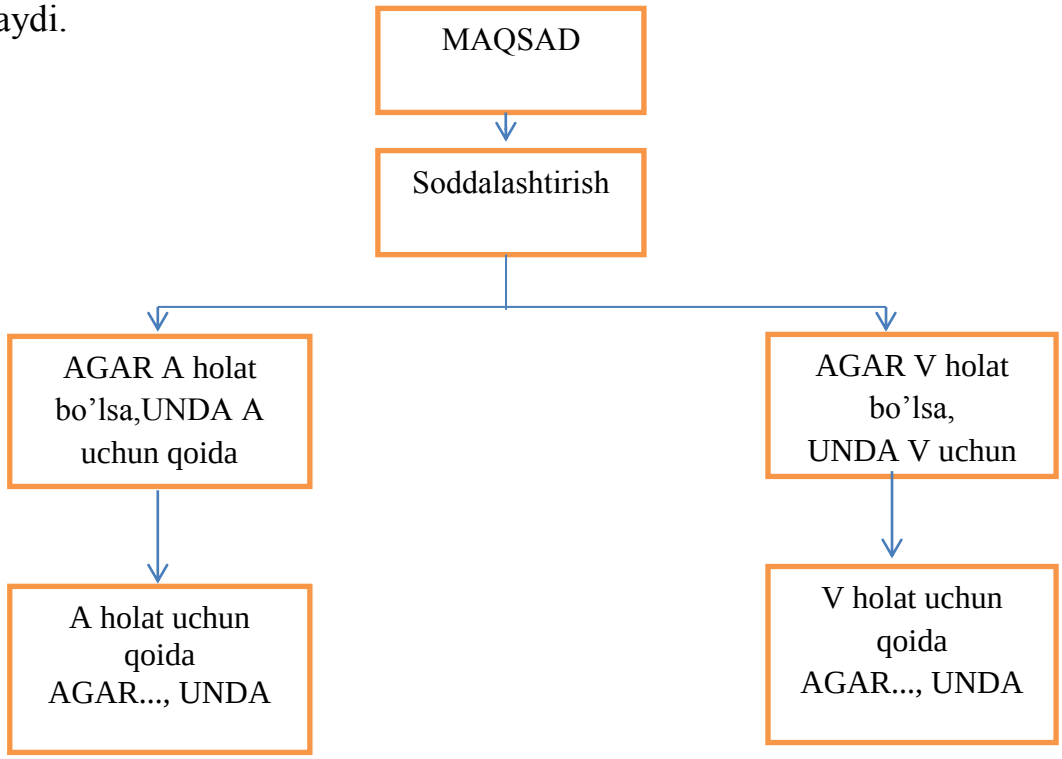
1. Qoidalar tizimi qanday ishlab chiqiladi?
2. Qoidalar tizimini ishlab chiqishda nimalardan foydalaniladi?

6 - Laboratoriya ishi.
Mavzu: Teskari zanjir

Ishdan maqsad: Qoidalarning teskari zanjiri tushunchasi va qoidalarning teskari zanjirini ishlab chiqishni o'rganish.

Masalaning qo'yilishi: Jismoniy shaxsning kelajakdagi iqtisodiy holatini joriy holatiga qarab tahlil qiluvchi dastur tuzilsin.

Uslubiy ko'rsatmalar: Sun'iy intellekt haqida gapirish uchun dasturiy tizim inson tomonidan qaror qabul qilish jarayonini tashkil qiluvchi barcha elementlar, maqsadlar, dalillar, qoidalar, mexanizmlar, xulosalar va soddalashtirishga ega bo'lishi kerak. Bunday elementlarning asosiy tarkibiy qismlari 1- rasmda ko'rsatilgan. Sun'iy intellekt tizimining an'anaviy dasturiy tizimlaridan asosiy farqi shundan iboratki, uning tuzilishining tarkibiy qismlari bo'lingan holda aniqlanadi va uning istalgan qismini zamonaviylashtirish umumiy tuzilmaga ta'sir qilmaydi.



6.1.sxema. Sun'iy intellekt tizimining tarkibiy qismlari.

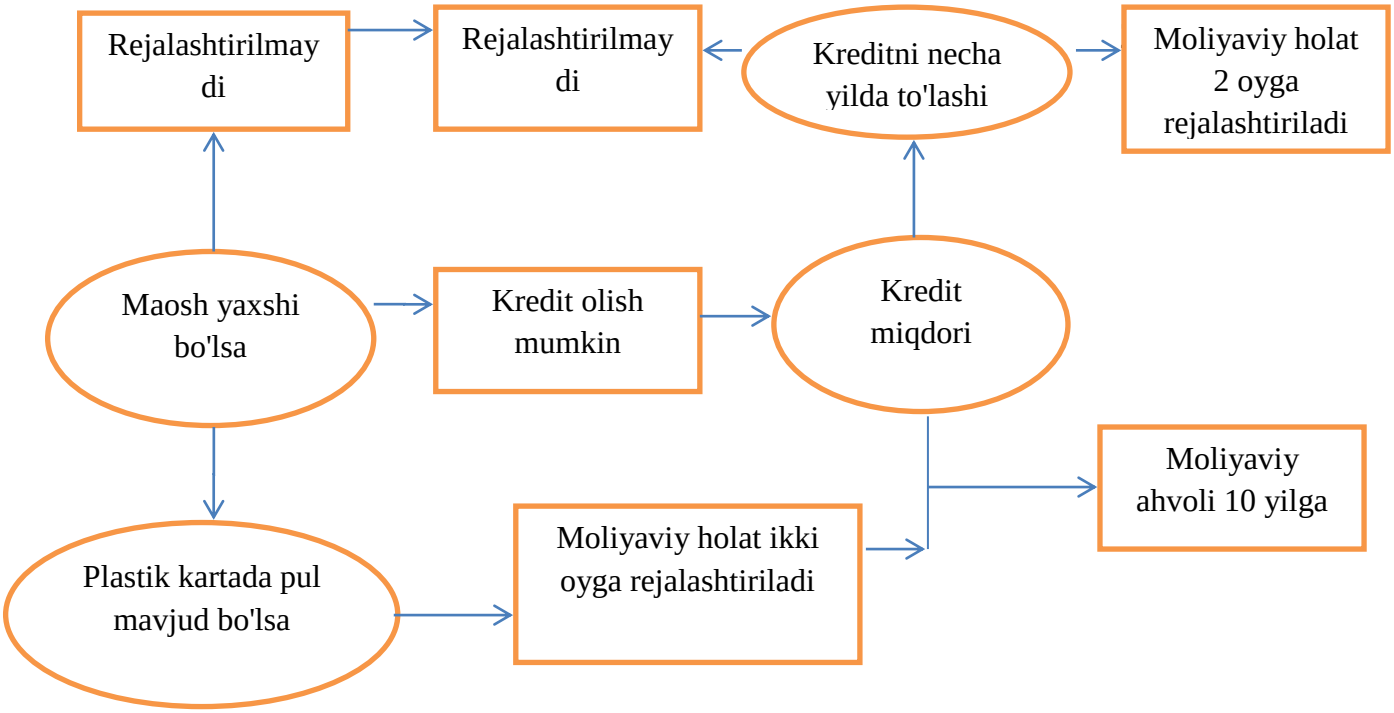
3-Laboratoriya ishida bayon qilingan bosqichlar yordamida maqsadlarga erishish jarayoni mulohazalarning to’g’ridan-to’g’ri zanjiri, ya’ni ma’lumotlardan mantiqiy xulosaga boruvchi zanjir deb ataladi. Uni tasdiqlovchi ma’lumotlarni qidirish uchun xulosadan foydalaniladigan jarayon teskari zanjir deb ataladi. Maqsadga erishilganda uning to’g’riligini tekshirish, ya’ni ma’lumotlar va qoidalar bilan yana vazifani tahlil qilish zarurdir. Bu jarayon maqsadlarni verifikatsiyalash deb ataladi. Yana avvaldan to’g’ri deb faraz qilingan xulosani tasdiqlash uchun yangi ma’lumotlarni chiqarish mexanizmlarini tanlash teskari zanjirga misol bo’lishi mumkin. shunday qilib, mulohazalarning teskari zanjiri to’g’ri zanjirga qarama qarshi tomonga, ya’ni xulosadan ma’lumotlarga ketadi. Teskari zanjir faqat maqsadga erishilgandan keyin vujudga keladi.

Inson miyasida soddalashtirish mexanizmi maqsadlarni verifikatsiyalash uchun unga erishishning barcha ehtimol bo’lgan usullari tekshirilguncha qadar qo’shimcha qoidalarni qidirishga rahbarlik qiladi. Sun’iy intellekt tizimining soddalashtirish mexanizmi kompyuterga belgilangan maqsadga erishish uchun bilimlar bazasidan ma’lumotlarning qandaydir qismini ularning muhimligiga ko’ra o’tkazib yuborish va ishlab chiqish imkoniyatini beradi. shunday qilib ham insonning miyasida va ham sun’iy intellekt tizimida soddalashtirish mexanizmi keraksiz va ishga tegishli bo’lmagan mulohazalarni oddiygina e’tiborga olmaydi.

Biror ishni amalga oshirish jarayonini amalga oshirish uchun avval uning qoidalar tizimi ishlab chiqiladi, so'ngra jarayon amalga oshiriladi. Qoidalar tizimi umumiy jarayon tarkibidagi elementar jarayonlar shartlar asosida tekshiriladi va shart natijasi bo'yicha keyingi elementar jarayonga o'tiladi. Qoidalarning teskari zanjirida jarayonlar shartlarga qo'yib tekshiriladi va keyingi etapga o'tishda shu shartga zid keluvchi amalga o'tiladi. Misol uchun biror apparat bor. Unga «apparat uchadimi» degan shart qo'yiladi. Agar «ha» bo'lsa «apparat uchun osmonda parvoz yo'li tashkil etilsin» amali bajariladi, aks holda «apparat uchun yerda yo'l qurilsin» degan amal bajarilsin. Qoidalarning teskari zanjirining asosiy mohiyati shartning amalga oshmaganda sodir bo’luvchi hodisadir Qoidalarning teskari zanjiri ko’pgina sohalarda qo’llaniladi. Misol uchun sun’iy intellektlarni yaratishda. Sun’iy intellektlarning eng ko’p qismi shartlardan iborat bo’ladi.

Qoidalar tizimi :

7. If (maosh(false)) moliyaviy_holat(rejalashtirilmadi).
8. If(maosh(true)) kredit(true).
9. If (maosh(true) && plastic_kartochka(true)) moliyaviy_holat(oy(2));
10. If (kredit(true) && pul()>1000) finance(yil(10))
11. If (kredit(true) && pul()<100 && yil()<2) moliyaviy_holat(yil(10))
12. If (kredit(ha) && pul()<100 && yil()>2) moliyaviy_holat(rejalashtirilmadi);



6.2. sxema. Moliyalashtirish sxemasi

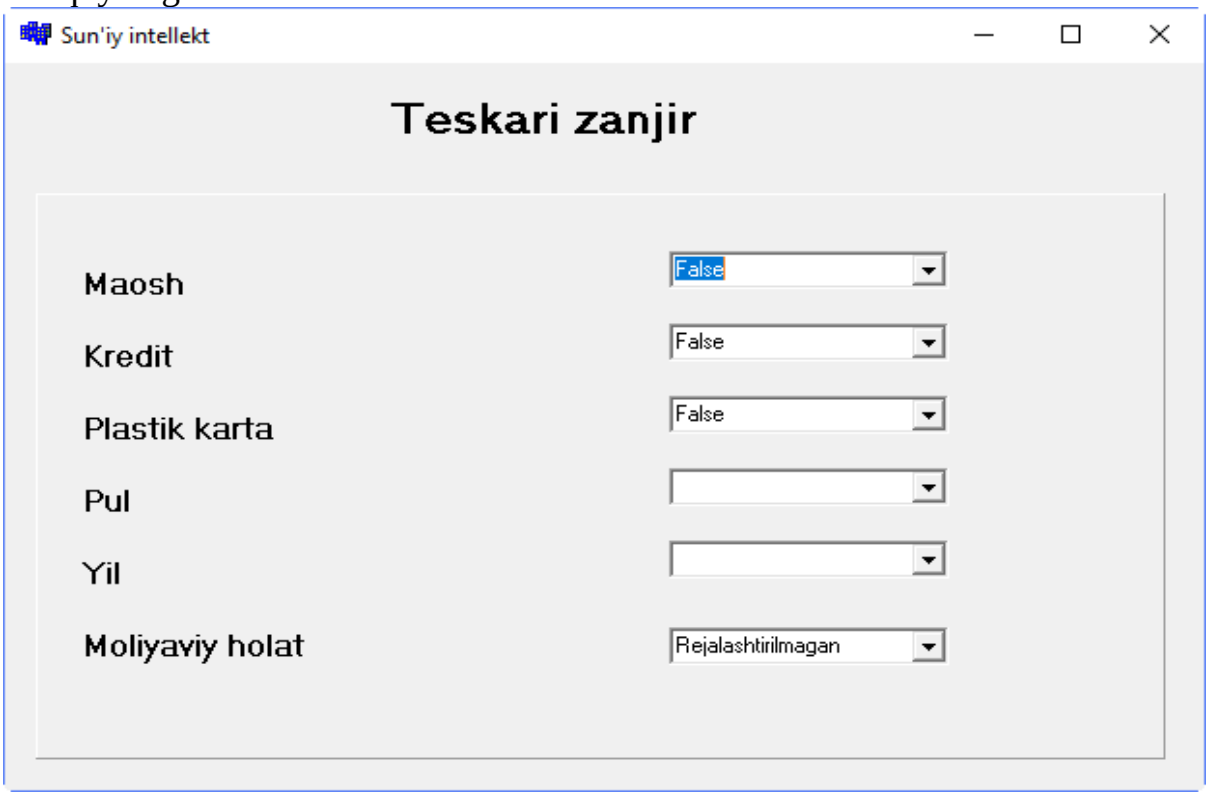
moliyaviy_holat() moliyaviy_holat(),yil(),false<10
moliyav_holat() ,lat()
plastik_karta(), moliyaviy_holat(), Maosh(), kredit(), pul()>1000
true, true, true

Xulosa qilib aytganda Sun'iy intellekt kompyuterni intellektining alomatlari bilan "jihozlashni" ko'zda tutadi. Sun'iy intellekt usullari dasturlarni birlashtirishni soddalashtiradi va tizimga o'z o'zini o'qitish va yangi axborotlarni jamlash qobiliyatini kiritish imkoniyatini beradi. Sun'iy intellekt haqida gapirish uchun dasturiy tizim inson tomonidan qaror qabul qilish jarayonini tashkil qiluvchi barcha elementlar, maqsadlar, dalillar, qoidalar, mexanizmlar, xulosalar va soddalashtirishga ega bo'lishi kerak. Inson miyasida soddalashtirish mexanizmi maqsadlarni verifikatsiyalash uchun unga erishishning barcha ehtimol bo'lgan usullari tekshirilguncha qadar qo'shimcha qoidalarni qidirishga rahbarlik qiladi. Sun'iy intellekt tizimining soddalashtirish mexanizmi kompyuterga belgilangan maqsadga erishish uchun bilimlar bazasidan ma'lumotlarning qandaydir qismini ularning muhimligiga ko'ra o'tkazib yuborish va ishlab chiqish imkoniyatini beradi.

Dasturning asosiy oynasi quyidagi ko`rinishga ega :

6.1.rasm. Teskari zanjir oynasi

Yoki quyidagicha :



6.2.rasm.Tekshirish oynasi

Dastur kodi :

```
/---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
USEFORM("Unit1.cpp", Form1);
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{ try
```

```
        {
                Application->Initialize();
                Application->CreateForm(__classid(TForm1), &Form1);
                Application->Run();
        }
        catch (Exception &exception)
        {
                Application->ShowException(&exception);
        }
        catch (...)
        {
                try
                {
                        throw Exception("");
                }
                catch (Exception &exception)
                {
                        Application->ShowException(&exception);
                }
        }
        return 0;
}
//---------------------------------------------------------------------------v
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "sComboBox"
#pragma link "sPanel"
#pragma link "sSkinManager"
#pragma resource "*.dfm"
TForm1 *Form1;
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}void __fastcall TForm1::sComboBox1Change(TObject *Sender)
{
if ( sComboBox1->ItemIndex ) {
        sComboBox6->ItemIndex = 0;
        sComboBox2->ItemIndex = 1; }
else  {
        sComboBox6->ItemIndex = 1;
        sComboBox2->ItemIndex = 0; }
}
```

```
void __fastcall TForm1::sComboBox4Change(TObject *Sender)
{
if
( sComboBox4->ItemIndex > 2 && sComboBox2->ItemIndex && sComboBox5-
>ItemIndex < 2 ) {
      sComboBox6->ItemIndex = 2; }
else  {
      sComboBox6->ItemIndex = 1;
      sComboBox2->ItemIndex = 0; }
}

void __fastcall TForm1::sComboBox2Change(TObject *Sender)
{
if ( sComboBox2->ItemIndex && sComboBox1->ItemIndex == 0 ) {
      sComboBox6->ItemIndex = 3; }
}
```

Nazotar savollari

1. Teskari zanjir usulida masalalarni yechishni tushuntiring?
2. Teskari zanjir usulida masalalardagi ma’lumotlar o’zaro qanday bog’langan?

7 - Laboratoriya ishi.

Mavzu: «Hill Climbing» texnikasi.

Ishdan maqsad: «Hill climbing» texnikasini o'rganish.

Masalaning qo'yilishi: «Hill climbing» texnikasidan foydalnib dastur tuzing.

Uslubiy ko’rsatmalar: Mantiqiy modellarning asosida rasmiy nazariya tushunchasi yotadi. Mantiqiy modellarda bilimlarning alohida birliklari (dalillar) o’rtasidagi munosabatlar rasmiy nazariyaning sintaktik bilimlari yordamida aks ettiriladi (masalan, predikatlarni hisoblash). Mantiqiydan farqliroq evristik modellar u yoki bu muammoli sohaning o’ziga hos xususiyatlarini uzatuvchi vositalarning turli tuman majmuasiga ega. Buning oqibatida evristik modellar mantiqiylardan ham imkoniyati yoki xuddi shunday aks ettirish, ya’ni muammoli sohani taqdim etish qobiliyati bo’yicha va ham chiqishning foydalanilgan mexanizmining samaradorligi bo’yicha ustivorlik qiladi. Evristik modellar tarmoqli, freymli yoki mahsulotli bo’ladilar. Mantiqiy modellar predikatlarni hisoblash tilidan foydalanadilar. Birinchi predikatga munosabatning nomi mos keladi, dalilning atamasiga esa - ob’ektlar. Predikatlar mantiqida foydalaniladigan barcha mantiqiy iborolar haqiqiy yoki yolg’on ma’noga ega. Masalan “Jon kompyuter bo’yicha mutaxassisdir” iborasini ko’rib chiqamiz. Bu ibora quyidagicha taqdim etilishi mumkin (Jon kompter bo’yicha mutaxassis)dir. Ammo bu ibora quyidagicha interpritatsiyalanishi mumkin: qandaydir X ob’ekti mavjud, u

kompter bo'yicha mutaxassisdir. Bunda yozuvning quyidagi formulasidan foydalaniladi (X, kompter
bo'yicha mutaxassis)dir.

Evristik qidiruvning asosiy g'oyasi shundan iboratki, unda qidiruv jarayonini boshqarish uhcun qo'shimcha axborotlardan foydalaniladi. Evristikada foydalanish masalaning yechimini izlash jarayonida maqsadga tezroq erishishga olib boruvchi ko'rib chiqadigan variantlar sonini qisqartirish imkonini beradi.

Evristik qidiruv algoritmlariida evristik qoidalar asosida shakllantiruvchi cho'qqilar ro'yhati, bir necha baholash funksiyalarining o'sish tartibi bo'yicha tartiblanadi.

Baholash funksiyasi evristik tashkil etuvchisining qiymati qanchalik kichik bo'lsa, ko'rib chiqilayotgan cho'qqi maqsadga chuncha yaqin bo'ladi. Unda "tog'ga ko'tarilish" algoritmini o'z ichiga oladi.

Algoritm maqsadga yo'naltirilgan qidirishni o'z ichiga olib, evristik baholash h'(n) funksiyasining ko'p yo'nalishlarda kamayishini aks ettiradi. h'(n) funksiyaning qiymati qanchalik kam bo'lsa, cho'qqi joylashgan yo'l shunchalik "isdiqbolli" sanaladi.

Funksiyaning maksimumini qidirish eng yuqori cho'qqiga ko'tarilishning optimal yo'lini eslatadi. Shuning uchun ko'rilayotgan algoritm **Hill-climbing** deb ataladi.

Hill climbing dasturlashdagi local qidiruv oilasiga tegishli matematik optimizatsiya texnikasidir. Bu muammoning ixtiyoriy yechimi bilan boshlanuvchi va yechimning yagona elementini qadamba-qadam o'zgartirib yaxshiroq yechim topishga urinib ko'ruvchi takrorlanuvchi algoritmdir.

Masalan, sayohatga chiqish muammosida hill climbing qo'llaniladi. Barcha shaharlarga borishning boshlang'ich yechimni topish oson ammo unda optimal yechim ancha kam. Algoritm yechim bilan boshlanadi va bu yechimga ikki shahardan qaysi biriga borishni tanlash kabi kichik takomillashtirishlar kiritib boriladi. Nihoyat, eng qisqa yo'nalish topilgan hisoblanadi.

Hill climbing local optimum topishga yaxshi ammo barcha barcha yechimlar ichidan eng yaxshisini toppish kafolatlanmagan. Eng ko'zga ko'rinarli muammolarda hill climbing optimal hisoblanadi. Binar qidiruv va chiziqli dasturlashning sodda algoritmlarini o'z ichiga oluvchi ko'zga ko'rinarli muammolarning algoritm namunalarini hal qiladi.

Hill climbing berilgan f(x) funksiyani maksimallashtirishga harakat qiladi. Bu yerda x – to'xtovsiz va/yoki diskret vector. Har bir takrorlanishda hill climbing x dagi yagona elemnti tuzatiladi va f(x)ning qiymatini o'zgartirib aniqlanadi. Hill Climbing bilan f(x)ni yaxshilovchi birorta o'zgartirish qabul qilinadi va jarayon f(x)ni yaxshilovchi birorta o'zgartirish qolmaguncha davom etadi. Shundan so'ng x "local optimal" deb nomlanadi.

Diskret vector fazosida har bir mavjud bo'lishi mumkin bo'lgan x qiymat grafning uchi bo'lib hisoblanadi. Hill climbing x ning local ko'payishi orqali x_m local maksimalga yetgunicha grafning bir uchidan boshqa uchiga harakatlanadi.

Hill climbingning ikki turi mavjud: Stoxastik hill climbing va tasodifiy qayta boshlanuvchi hill climbing. Stoxastik hill climbingda qanday harakatlanishga qaror

qabul qilingunicha barcha qo'shni uchlar tekshirilmaydi. Tasodifiy qayta boshlanuvchi hill climbing hill climbing algoritmining ustida quriluvchi meta-algoritm. Uni yana Shotgun hill climbing nomi bilan ham ma'lum. Bu algoritm har safar o'zlashtirilgan tasodifiy x_0 shart bilan ketma-ketlikda hill climbingni amalga oshiradi. Eng yaxshi x_m saqlab qo'yiladi. Agar yanga qadamda yanayam yaxshi x_m topilsa oldingi saqlangani bilan almashtiriladi.

Algoritmning navbatdagi qadamida h'(n) baholash funksiyasi uchun qoyalar quyidagi qiymatlar n_1, n_2, …n_m va qolgan yo'llar uchun h'(n_i) qiymat belgilanadi. Agar barcha yondosh qoyalar h'(n)>=h(n) qiymatga ega bo'lsa, qidiruv protsedurasi muvaffiqiyatsizlikka uchraydi.

«hill climbing» dasturlashda lokal qidiruv oilasiga kiruvchi matematik optimizatsiyalash texnikasi hisoblash. Bu algoritm muammoni yechishda tasodifiy yechimlarni topish bilan boshlanadi. Keyin esa topilgan yechimlar orasidan optimal yechim qidirib topiladi. Bunda optimal javobning elementi qadamba qadam almashtirib boriladi.

«Hill climbing» metodologiyasi:

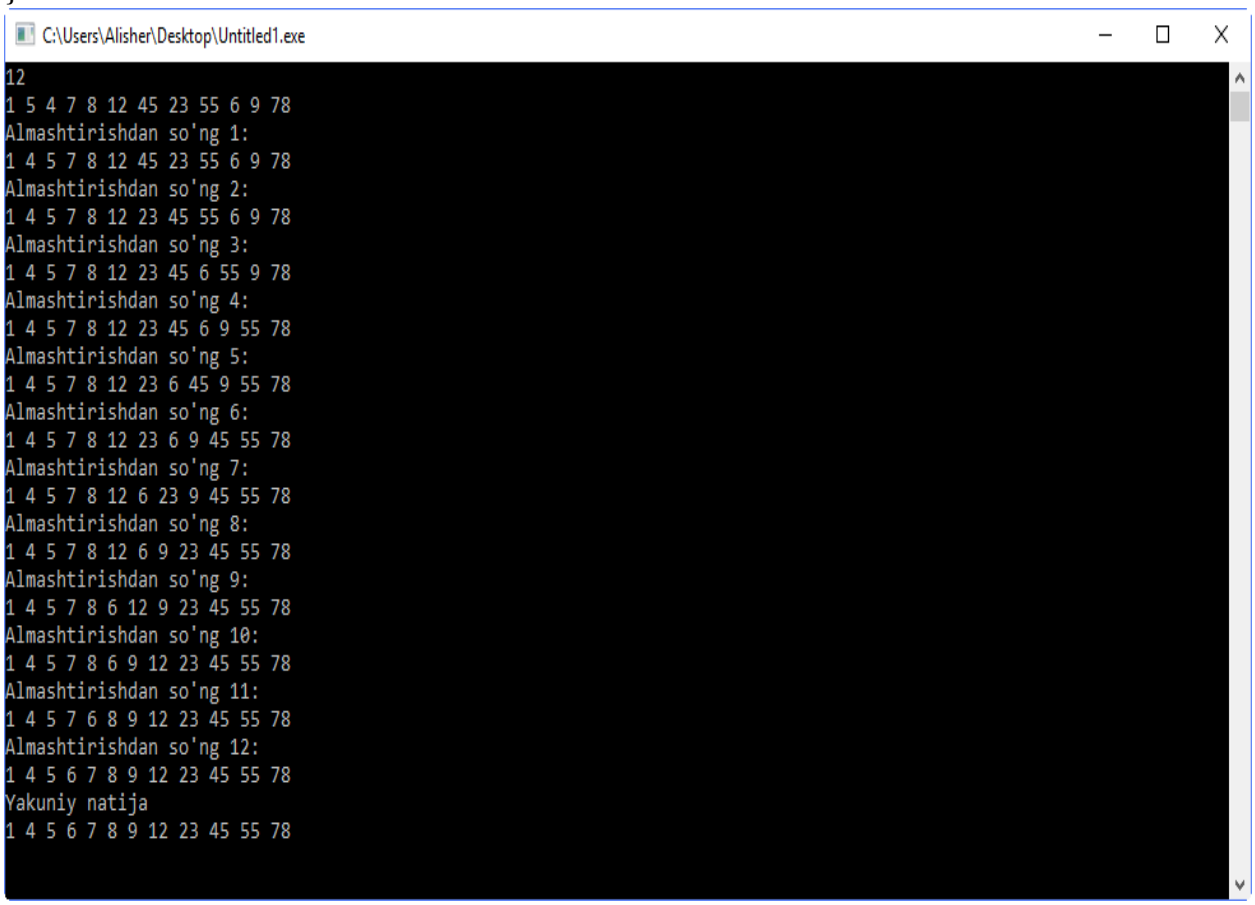
4. Muammoning tuzilmasida uchraydigan ost-optimal yechimlarni qurish.
5. Yechimni olish va uning ustida mukallashtirishni amalga oshirish.
6. Muhim yaxshilanishlar amalga oshirilmaguncha takror takror yechimlarda yaxshilanishlarni amalga oshirish

Hill climbingning eng tanilgan muammolaridan biri tarmoq oqimlari muammosidir. Bundan tashqari hill climbing algoritmi yordamida daraxtlar ustida ham amallar bajarish mumkin.

Dastur:

```
#include<iostream>
#include<conio.h>
#include<cstdio>
using namespace std;
int calcCost(int arr[],int N){
 int c=0;
 for(int i=0;i<N;i++){
  for(int j=i+1;j<N;j++) if(arr[j]<arr[i]) c++; }
 return c;
}void swap(int arr[],int i,int j){
 int tmp=arr[i];
 arr[i]=arr[j];
 arr[j]=tmp;
}
int main(){
 int N;
 scanf("%d",&N);
 int arr[N];
 for(int i=0;i<N;i++) scanf("%d",&arr[i]);
 int bestCost=calcCost(arr,N),newCost,swaps=0;;
```

```
        while(bestCost>0){
         for(int i=0;i<N-1;i++){
          swap(arr,i,i+1);
          newCost=calcCost(arr,N);
          if(bestCost>newCost){
           printf("Almashtirishdan so'ng %d: \n",++swaps);
           for(int i=0;i<N;i++) printf("%d ",arr[i]);
           printf("\n");
           bestCost=newCost; }
          else swap(arr,i,i+1);  } }
        printf("Yakuniy natija\n");
        for(int i=0;i<N;i++) printf("%d ",arr[i]);
        printf("\n");
        getch();
        return 0;
}
```



7.1.rasm. Natija oynasi

Nazorat savollari

1. Evristik qidiruv nima?
2. “Hill Climbing” texnikasini tushuntiring?
3. “Hill climbing” metodologiyasini tushuntiring?

8 – Laboratoriya ishi

Mavzu: Qidiruv Beam search algoritmi.

Ishdan maqsad: Bu laboratoriya ishi orqali talabalarda qidiruv Beam search algoritmi haqida ma`lumot hosil qilishdir .

Masani qo'yilishi: Beam search algoritmi yordamida qidirishni to'gri topish

Uslubiy ko'rsatmalar: Qidiruv Beamsearch algoritmini chuqurlik bo`ylab tarqalish deb atashimiz mumkin .Bu atama birinchi marta Raj Reddy tomonidan Carnegi Mllon Universitetida ishlatilgan .Bu qidiruvda har bir usuldan bir birigia o`tish imkoniyatlari mavjud bo`ladi. Beam search eng yaxshi birinchi qidiruv algoritmlari bo`lib xotiradan joy olishi qisqartirib optimallashtiradi . Beam search qidiruv daraxtidan qidirishda foydalaniladi . Uni ko`plab mashina ishini boshqarishda foydalanamiz . Bu algoritmdan foydalanishda biz natijalarni tezda olamiz , chunki natijalar ham o`zaro bir biriga bog`langan bo`ladi. Beam searchni nurli qidiruv deb tarjima qilamiz. Nurli qidiruv optimallashtirilgan "birinchisining eng yaxshisi" algoritmidir. Orginaliga o'xshab u har bir tugunni evristik baholash funksiyasidan foydalanadi. Biroq, faqat har bir o'tishdagi birinchi eng ko'p istiqbolli m tugun baholanadi. Bu yerda m – fiksirlangan son.

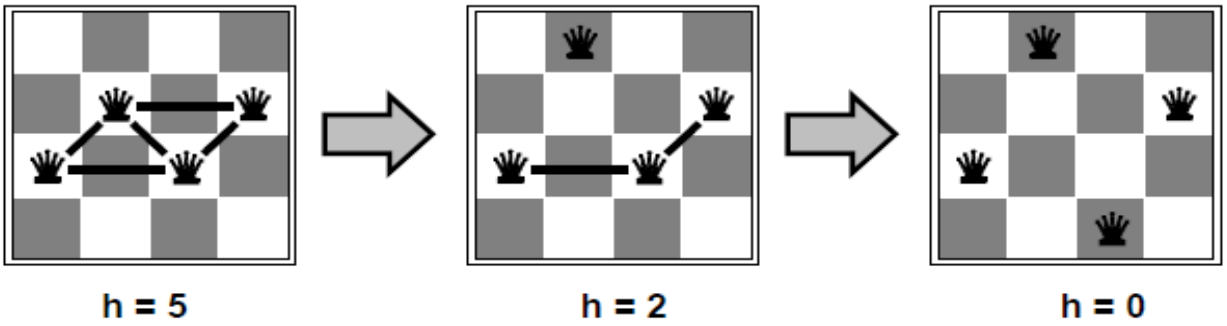
Beam search qidiruv daraxtini quraishda breadth-first search dan foydalanadi. Daraxtning har bir darajasida u holatlarni evristik bahoning o'sish tartibida saralab joriy darajadagi barcha holat davomchilarini ishlab chiqadi. Shunga qaramay, u β – oldindan belgilangan har bir darajadagi eng yaxshi holat nomerini saqlab qo'yadi(nur kengligi deb nomlanadi). Faqat shu holatlar keyingisini kengaytiradi. Eng kata nur kengligi, eng kam holat kesib tashlanadi. "Beam search" atamasini birinchi bo'lib Carnegi Mellon Universitetidan Raj Reddy ishlatgan.

Beam search to'liq qidiruv daraxtini saqlovchi xotiraning nuqsonli yig'iladigan katta tizimlarda itoatkor saqlashda ko'p foydalaniladi. Masalan, u ko'pgina tarjima mashinalarida foydalaniladi. Eng yaxshi tarjimani tanlash uchun har bir qism qo'zg'atiladi va turli xil tarjima yo'llari bilan so'zlar paydo bo'ladi. Ularning gap tuzilishi o'rniga eng yaxshi tarjima qo'shimchasi saqlanadi. Tarjimon keyin berilgan kriteria o'rniga tarjimalarga baho beradi. Beam search birinchi marta 1976-yil Carnegi Mellon Universitetida Harpy nutqni tanish tizimida qo'llanilgan.

Beam search quyidagilarda ishlatiladi:

- Integratsiyalashgan dizayn zanjiri
- Factory-floor layout
- Ishni rejalashtirish
- Tarmoqni optimizatsiyalash
- Transport yo'nalishini aniqlash
- Sayohat uyushtirish muammolarida
- Mashinali tarijama qilishda

Nta qirolichalar masalasi. NxN katakli doskaga Nta qirolichani qator yoki ustun va yoki dioganal bo'yicha 2tadan joylashtirmasdan qo'yamiz.



8.1.rasm. Shaxmat oynasi

O’sish tartibidagi kesishuvchi raqamlar bo’ylab qirolichalarni harakatlantiramiz. Bunda Nta qirolicha masalasi kata N uchun tezda yechiladi.

Beam Search algoritmi

OPEN = {Boshlang’ich holatni berish}

While OPEN bo’sh emas bo’lsa, bajarilsin

OPENdan eng yaxshi tugunni o’chirish

Agar n maqsadli holat bo’lsa, yo’lni nga qaytarish va yo’lni qaytarish

N ning vorislarini yaratish

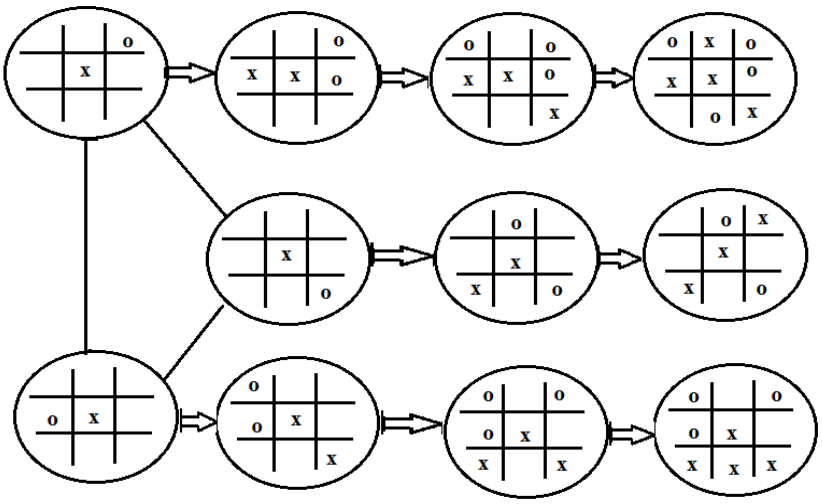
Har bir vorisni baholash, uni OPENga qo’shish, va uning otasini yozib olish

Agar OPEN > ß bo’lsa, eng yaxshi ß tugunlarni olish va qolganlarini OPENdan olib tashlash.

X-o o’yini - 3x3 kvadrat kataklardagi 2ta raqiblar orasidagi mantiqiy o’yin. Bitta o’yinchi x lar bilan ikkinchisi o lar bilan o’ynaydi. An’anaviy xitoy o’yinida qora va oq toshlardan foydalaniladi. Kim o’z belgilari bilan kataklarni gorizontal yoki vertical va yoki x bo’yicha to’ldirsa yutgan hisoblanadi. Har bir tomonda durang qiluvchi yoki raqibini yutishga imkon beruvchi umumiy ma’lum algoritmlar mavjud. Bu o’yinni kompyuterda bajarish uchun mini-maks usuli bilan mos keluvchi o’yin holatlari daraxti quriladi. Bunday daraxtning tugunlari soni 255168taga teng. Bu son barcha mumkin bo’lgan birinchi qadamdagi variantlar soni – 9 taning yig’indisidan olinadi. 2-qadamda 9taning har biri uchun 8 ta variant bo’ladi. 3-qadamda 72ta variantning har biri uchun 7ta variant bo’ladi va hk.

Hamma natijalar ham yaxshi ya`ni kutilganidek bo`lmasligi mumkin . Masalan men tanlab olgan x o o`yinida ham doim g`olib bo`lavaermaydi . Demak doim muvaffaqiyatli natija chiqmasligi mumkin .

“x o ” o`yini uchun kombinatsiyalarning ba`zi birlani ko`rib chiqamiz.



8.2.rasm. Yo’naltiruvchi oyna

Xulosa

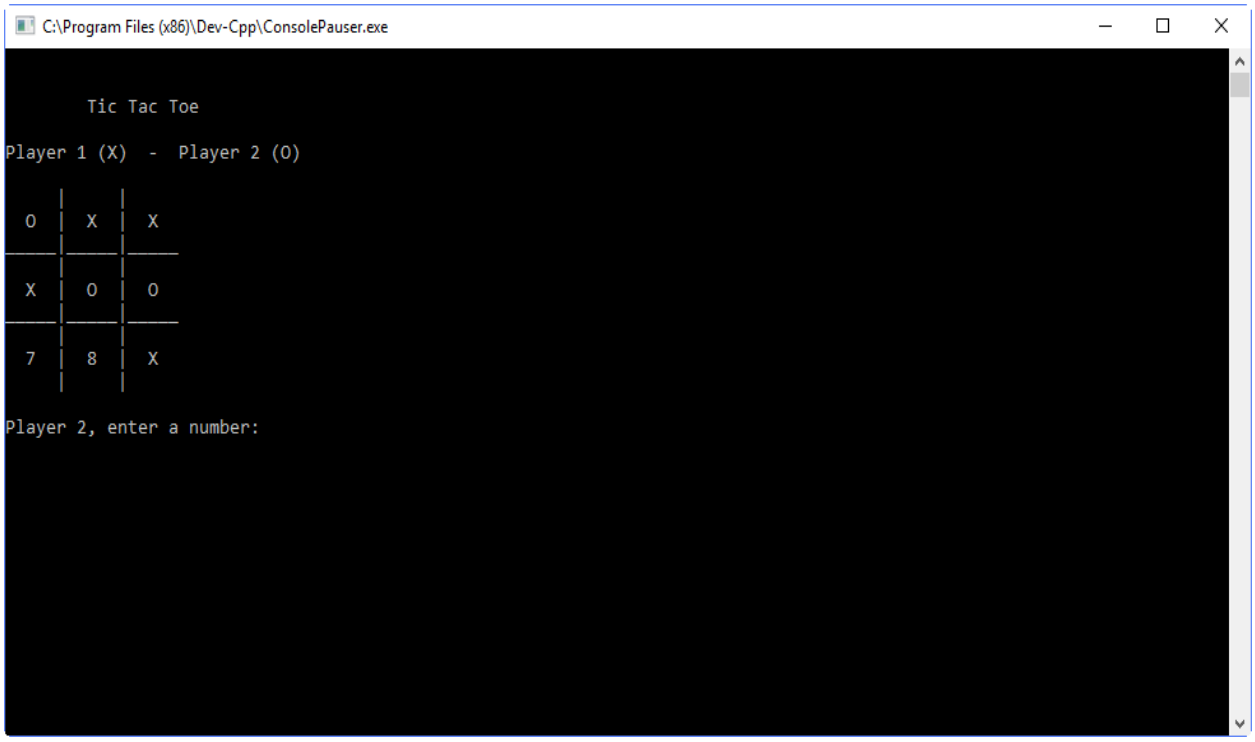
Qidiruv algoritmlarining juda ko`plab turlari mavjud bo`lib , lekin ularning hammasi bitta narsaga natijani olishga qaratilgan . Shu algoritmlaridan biri Beam search hisoblanadi . Xulosa qilib aytadigan bo`lsak bu laboratoriya ishida biz qidiruv algotimining yana bir turi haqida ma`lumotga ega bo`ldik . Bu algoritm orqali natijani tez va qulay usulda olishimiz mumkin.

Dasturi.

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>
char square[10] = {'o','1','2','3','4','5','6','7','8','9'};
int checkwin();
void board();
int main()
{       int player = 1,i,choice;
        char mark;
        system("cls");
        do      {
                board();
                player=(player%2)?1:2;
                std::cout << "Player " << player << ", enter a number:  ";
                std::cin >> choice;
                mark=(player == 1) ? 'X' : 'O';
                if (choice == 1 && square[1] == '1')
                        square[1] = mark;
                else if (choice == 2 && square[2] == '2')
                        square[2] = mark;
                else if (choice == 3 && square[3] == '3')
                        square[3] = mark;
                else if (choice == 4 && square[4] == '4')
                        square[4] = mark;
```

```
                else if (choice == 5 && square[5] == '5')
                        square[5] = mark;
                else if (choice == 6 && square[6] == '6')
                        square[6] = mark;
                else if (choice == 7 && square[7] == '7')
                        square[7] = mark;
                else if (choice == 8 && square[8] == '8')
                        square[8] = mark;
                else if (choice == 9 && square[9] == '9')
                        square[9] = mark;
                else
                {                       std::cout<<"Invalid move ";
                        player--;
                        getch();                }
                i=checkwin();
                player++;
        }while(i==-1);
        board();
        if(i==1)
                std::cout<<"==>\aPlayer "<<--player<<" win ";
        else
                std::cout<<"==>\aGame draw";
        getch();
        return 0;}
int checkwin()
{       if (square[1] == square[2] && square[2] == square[3])
                return 1;
        else if (square[4] == square[5] && square[5] == square[6])
                return 1;
        else if (square[7] == square[8] && square[8] == square[9])
                return 1;
        else if (square[1] == square[4] && square[4] == square[7])
                return 1;
        else if (square[2] == square[5] && square[5] == square[8])
                return 1;
        else if (square[3] == square[6] && square[6] == square[9])
                return 1;
        else if (square[1] == square[5] && square[5] == square[9])
                return 1;
        else if (square[3] == square[5] && square[5] == square[7])
                return 1;
        else if (square[1] != '1' && square[2] != '2' && square[3] != '3' &&
                square[4] != '4' && square[5] != '5' && square[6] != '6' &&
          square[7] != '7' && square[8] != '8' && square[9] != '9')
                return 0;
```

```
        else
                return -1;
}void board()
{
        system("cls");
        std::cout << "\n\n\tTic Tac Toe\n\n";
        std::cout << "Player 1 (X)  -  Player 2 (O)" << "\n";
        std::cout << "\n";
        std::cout << "     |     |     " << "\n";
        std::cout << "  " << square[1] << "  |  " << square[2] << "  |  " << square[3]
<< "\n";
        std::cout << "_____|_____|_____" << "\n";
        std::cout << "     |     |     " << "\n";
        std::cout << "  " << square[4] << "  |  " << square[5] << "  |  " << square[6]
<< "\n";
        std::cout << "_____|_____|_____" << "\n";
        std::cout << "     |     |     " << "\n";
        std::cout << "  " << square[7] << "  |  " << square[8] << "  |  " << square[9]
<< "\n";
        std::cout << "     |     |     " << "\n" << "\n";
}
```



8.3.rasm. Natija oynasi

Nazorat savollari

1. Beam Search algoritmini tushuntiring?

2. Beam Search algoritmi kim tomonidan yaratilgan?
3. Beam Search algoritmidan qayerlarda foydalaniladi?

9-Laboratoriya ishi

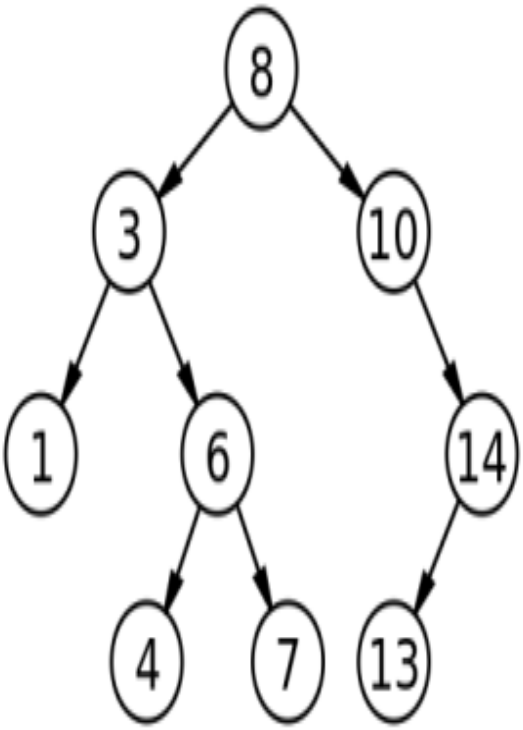
Mavzu: *Qidiruv "Optimal search"*

Ishdan maqsad: Mantiqiy masalalar uchun Optimal qidiruvni ishlab chiqish.

Masalaning qo'yilishi: Optimal search algortimi orqali x o o`yinini yaratish.

Uslubiy ko'rsatmalar: Ikkilik qidiruv daraxti yoki "optimal search" bu ikkilik daraxti bo'lib unga quyidagi shartlar qo'yiladi:

- Ikki chap va o'ng ost daraxtlar ikkilik qidiruv daraxti hisoblanadi.
- Barcha X tugundan chiqgan ixtiyoriy chap ichki daraxtlar tugunlari shu tugundagi qiymatlaridan kichik.
- Barcha X tugundan chiqgan ixtiyoriy o'ng ichki daraxtlar tugunlari shu tugundagi qiymatlaridan katta yoki teng.



9.1.rasm. Ikkilik qidiruv daraxti.

Ko'rinib turibdiki har bir tugundagi ma'lumotlar *kichik* solishtirish amali bajariladigan kalitlarga ega bo'lishi kerak.

Har bir tugun taqdim etadigan ma'lumot yagona ma'lumot maydoni emas, balki *yozuv*(yoki class) hisoblanadi. Lekin bu ikkilik qidiruv daraxtining tabiati emas.

Ikkilik qidiruvni amalga oshirish uchun quyidagilarni aniqlash lozim:

- Ikkilik daraxti tugunlardan iborat(qirra) – data, left, right kabi yozuvlarga ega, bu yerda data tugunga bog'langan ma'lumot, left va right tugunning bolalari tugunlarga ko'rsatkich.
- Istalgan X tugun uchun qidiruv daraxti xususiyati amalga oshiriladi: $key\left[left[X]\right] < key[X] \leq key[right[X]]$.

Ikkilik qidiruv daraxtini ikkilik to'plam bilan almashtirib yubormaslik lozim. U boshqa qoidalarga bo'ysungan holda ishlaydi. Ikkilik qidiruv daraxtining boshqa strukturalangan ma'lumotlarni qidirish algoritmlaridan afzal tomonlari shundaki, unda qidiruv va saralash algoritmlari yuqori samaradorlikka ega bo'ladi. Bu algoritm yana to'plamlar, multito'plamlar va asotsiatsiyalangan massivda ham ishlaydi.

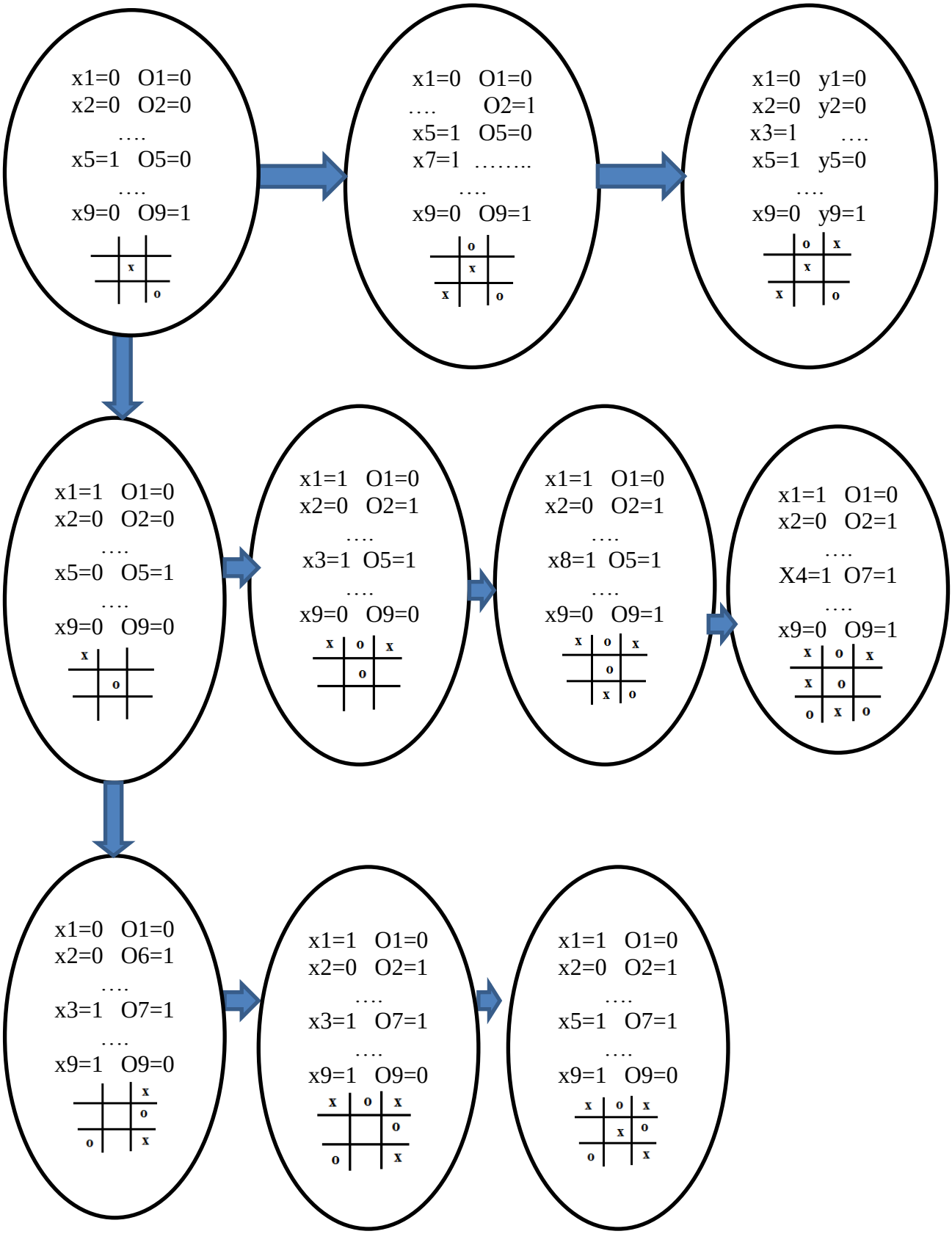
Boshlang'ich holatda ikkilik qidiruv daraxti interfeysi 3 amaldan tashkil topadi:

- FIND(K) – key=K ga teng bo'lgan (key, value) juftliklari saqlanadigan tugunni topish.
- INSERT(K,V) – (key,value)=(K,V) daraxtga juftliklarini qo'shish.
- REMOVE(K)- key=k bo'lgan (key, value) tugunni o'chirish.

X-o o'yini - 3x3 kvadrat kataklardagi 2ta raqiblar orasidagi mantiqiy o'yin. Bitta o'yinchi x lar bilan ikkinchisi o lar bilan o'ynaydi. An'anaviy xitoy o'yinida qora va oq toshlardan foydalaniladi. Kim o'z belgilari bilan kataklarni gorizontal yoki vertical va yoki x bo'yicha to'ldirsa yutgan hisoblanadi. Har bir tomonda durang qiluvchi yoki raqibini yutishga imkon beruvchi umumiy ma'lum algoritmlar mavjud. Bu o'yinni kompyuterda bajarish uchun mini-maks usuli bilan mos keluvchi o'yin holatlari daraxti quriladi. Bunday daraxtning tugunlari soni 255168taga teng. Bu son barcha mumkin bo'lgan birinchi qadamdagi variantlar soni – 9 taning yig'indisidan olinadi. 2-qadamda 9taning har biri uchun 8 ta variant bo'ladi. 3-qadamda 72ta variantning har biri uchun 7ta variant bo'ladi va hk.

Hamma natijalar ham yaxshi ya`ni kutilganidek bo`lmasligi mumkin . Masalan men tanlab olgan x o o`yinida ham doim g`olib bo`lavaermaydi . Demak doim muvaffaqiyatli natija chiqmasligi mumkin .

- "x o " o`yini uchun kombinatsiyalarning ba`zi birlani ko`rib chiqamiz.



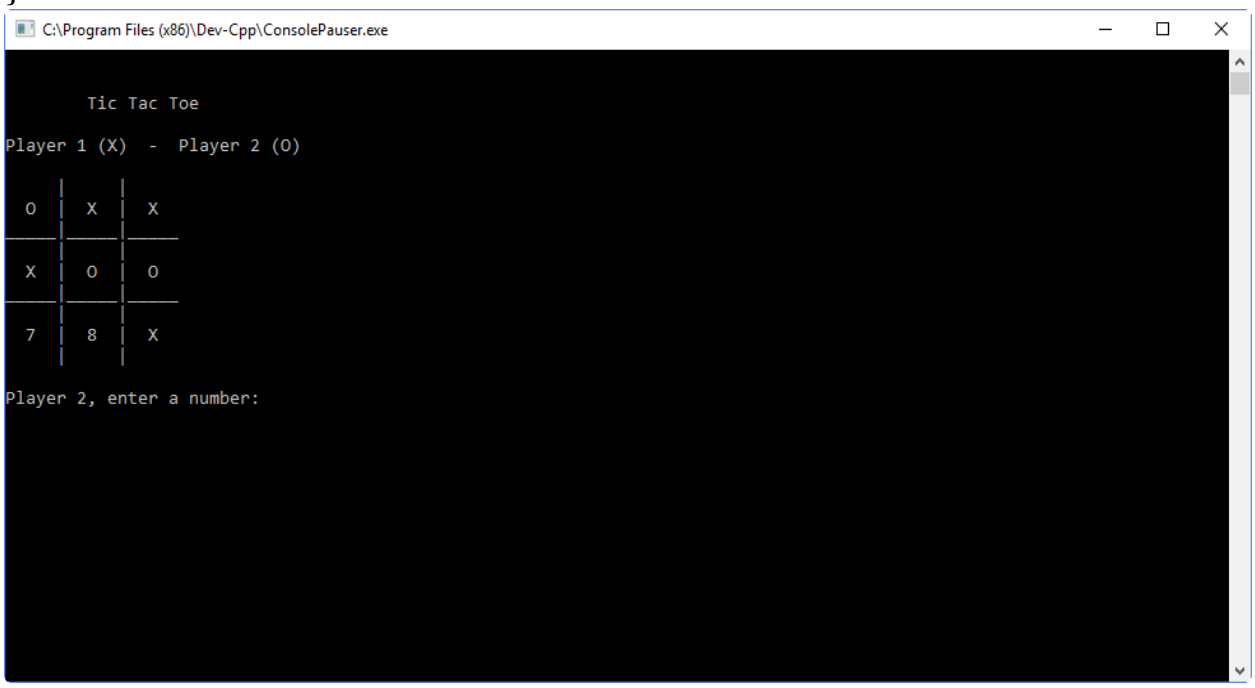
9.2.rasm. O’zgaruvchilarni e’lon qilish

Yechim yo`q
X win Dasturi.

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>
char square[10] = {'o','1','2','3','4','5','6','7','8','9'};
int checkwin();
void board();
int main()
{
	int player = 1,i,choice;
	char mark;
	system("cls");
	do	{
		board();
		player=(player%2)?1:2;
		std::cout << "Player " << player << ", enter a number:  ";
		std::cin >> choice;
		mark=(player == 1) ? 'X' : 'O';
		if (choice == 1 && square[1] == '1')
			square[1] = mark;
		else if (choice == 2 && square[2] == '2')
			square[2] = mark;
		else if (choice == 3 && square[3] == '3')
			square[3] = mark;
		else if (choice == 4 && square[4] == '4')
			square[4] = mark;
		else if (choice == 5 && square[5] == '5')
			square[5] = mark;
		else if (choice == 6 && square[6] == '6')
			square[6] = mark;
		else if (choice == 7 && square[7] == '7')
			square[7] = mark;
		else if (choice == 8 && square[8] == '8')
			square[8] = mark;
		else if (choice == 9 && square[9] == '9')
			square[9] = mark;
		else
		{			std::cout<<"Invalid move ";
			player--;
			getch();		}
		i=checkwin();
		player++;
	}while(i==-1);
	board();
```

```
        if(i==1)
                std::cout<<"==>\aPlayer "<<--player<<" win ";
        else
                std::cout<<"==>\aGame draw";
        getch();
        return 0;}
int checkwin()
{       if (square[1] == square[2] && square[2] == square[3])
                return 1;
        else if (square[4] == square[5] && square[5] == square[6])
                return 1;
        else if (square[7] == square[8] && square[8] == square[9])
                return 1;
        else if (square[1] == square[4] && square[4] == square[7])
                return 1;
        else if (square[2] == square[5] && square[5] == square[8])
                return 1;
        else if (square[3] == square[6] && square[6] == square[9])
                return 1;
        else if (square[1] == square[5] && square[5] == square[9])
                return 1;
        else if (square[3] == square[5] && square[5] == square[7])
                return 1;
        else if (square[1] != '1' && square[2] != '2' && square[3] != '3' &&
                square[4] != '4' && square[5] != '5' && square[6] != '6' &&
          square[7] != '7' && square[8] != '8' && square[9] != '9')
                return 0;
        else
                return -1;
}void board()
{
        system("cls");
        std::cout << "\n\n\tTic Tac Toe\n\n";
        std::cout << "Player 1 (X)  -  Player 2 (O)" << "\n";
        std::cout << "\n";
        std::cout << "     |     |     " << "\n";
        std::cout << "  " << square[1] << "  |  " << square[2] << "  |  " << square[3]
<< "\n";
        std::cout << "_____|_____|_____" << "\n";
        std::cout << "     |     |     " << "\n";
        std::cout << "  " << square[4] << "  |  " << square[5] << "  |  " << square[6]
<< "\n";
        std::cout << "_____|_____|_____" << "\n";
        std::cout << "     |     |     " << "\n";
```

```
        std::cout << "  " << square[7] << "  |  " << square[8] << "  |  " << square[9]
<< "\n";
        std::cout << "     |     |     " << "\n" << "\n";
}
```



9.3.rasm. Natijalarni olish

Nazorat savollari

4. Ikkilik daraxt nima?
5. Qidiruv daraxtini tushuntiring?
6. Kichik solishtirishlar amali haqida gapirib bering?

10 - Laboratoriya ishi

Mavzu: O’yinlarda to’rtliklarni birlashtirish (marge sort)

Ishdan maqsad: Marge sort, ya’ni birlashtirish algoritmining ishlash prinsipini ko’rib chiqish va uni o’yinlarning realizatsiyasida qo’llanilishini ko’rib chiqish.

Masalaning qo'yilishi: To’rtliklarni birlashtirish (merge sort) yordamida 4x4 o'yinini yaratilsin.

Uslubiy ko’rsatmalar: Merge sort (birlashtirgan holda tartiblash) – oqimdan kelgan ma’lumotlar ro’yhatini tartiblash va aniq bir tartibda tez saralash uchun ishlatiladi. Misol uchun buni “ajrat va boshqar” prinsipida ishlashini aytib o’tishimiz mumkin. Boshlanishda masala bir necha kichik masalalarga bo’linadi. Har bir misolda ajaratilgan bo’lim rekursiv funksiyalar yordamida qayta ishlanadi bu saralash vaqtining qisqarishiga olib keladi

Bunda saralashga ketgan yaxshi vaqt O(n log n) ga teng, yomon vaqt esa O(n log n) odanta O(n). Bu oddiy saralashdan ancha samara hisoblanadi, lekin optimallik darajasi u qadar yuqori emas.

Bu algoritmning boshqacha nomi “ulash orqali” saralash. Bu saralash uch tartibda amalga oshiriladi:

1. Tartiblanayotgan massivi taqriban bir-biroviga teng bo’lgan ikki qismga ajratiladi;

2. Har bir olingan qismlar shu algoritm asosida alohida tartibga solinadi;

3. Ikki tartiblangan massiv bittaga birlashtiriladi.

1.1-2.1 Rekursiv bo’laklash massivning o’lchami birga teng bo’luncha davom etadi.

3.1 Ikki tartiblangan massiv bittaga birlashtiriladi. Massvilarni birlashtirishni quyidagi misolda ko’rib o’tamiz. Tassavur qilaylik o’sish tartibida tartiblangan ikki massivga egamiz. U holda:

3.2 Ikki massivni uchinchi massivga birlashtirish. Har bir qadamda biz ulardan kichigini olib natijaviy massivga yozib qo’yamiz. Natijaviy massivni bittaga oshiramiz.

3.3 Qoldiqni “ulash”.

C++ tilida algoritmning amalda qo’llanilishi.

```
template<typename T>
void MergeSort(T a[], size_t l)
{       size_t BlockSizeIterator;
        size_t BlockIterator;
        size_t LeftBlockIterator;
        size_t RightBlockIterator;
        size_t MergeIterator;
        size_t LeftBorder;
        size_t MidBorder;
        size_t RightBorder;
        for (BlockSizeIterator = 1; BlockSizeIterator < l; BlockSizeIterator *= 2)
        {
                for (BlockIterator = 0; BlockIterator < l - BlockSizeIterator;
BlockIterator += 2 * BlockSizeIterator)
                {
                        LeftBlockIterator = 0;
                        RightBlockIterator = 0;
                        LeftBorder = BlockIterator;
                        MidBorder = BlockIterator + BlockSizeIterator;
                        RightBorder = BlockIterator + 2 * BlockSizeIterator;
                        RightBorder = (RightBorder < l) ? RightBorder : l;
                        int* SortedBlock = new int[RightBorder - LeftBorder];

                        while (LeftBorder + LeftBlockIterator < MidBorder &&
MidBorder + RightBlockIterator < RightBorder)
                        {
                                if (a[LeftBorder + LeftBlockIterator] < a[MidBorder +
RightBlockIterator])
                                {
```

```
                                        SortedBlock[LeftBlockIterator +
RightBlockIterator] = a[LeftBorder + LeftBlockIterator];
                                        LeftBlockIterator += 1;
                                }
                                else
                                {
                                        SortedBlock[LeftBlockIterator +
RightBlockIterator] = a[MidBorder + RightBlockIterator];
                                        RightBlockIterator += 1;
                                }
                        }
                        while (LeftBorder + LeftBlockIterator < MidBorder)
                        {
                                SortedBlock[LeftBlockIterator + RightBlockIterator] =
a[LeftBorder + LeftBlockIterator];
                                LeftBlockIterator += 1;
                        }
                        while (MidBorder + RightBlockIterator < RightBorder)
                        {
                                SortedBlock[LeftBlockIterator + RightBlockIterator] =
a[MidBorder + RightBlockIterator];
                                RightBlockIterator += 1;
                        }

                        for (MergeIterator = 0; MergeIterator < LeftBlockIterator +
RightBlockIterator; MergeIterator++)
                        {
                                a[LeftBorder + MergeIterator] =
SortedBlock[MergeIterator];
                        }
                        delete SortedBlock;
                }
        }
}
```

Algoritmni suniy intellektda ishlatgan holda o’yin yaratish.

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
int i;
void __fastcall TForm1::Button1Click(TObject *Sender)
```

```
{
Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button16Click(TObject *Sender)
{
if(Button17->Visible==0){
   Button17->Caption=Button16->Caption;
   Button17->Visible=1;    Button16->Visible=0;}
if(Button12->Visible==0){
   Button12->Caption=Button16->Caption;
   Button12->Visible=1;    Button16->Visible=0;}
if(Button15->Visible==0){
   Button15->Caption=Button16->Caption;
   Button15->Visible=1;    Button16->Visible=0;}
}void __fastcall TForm1::Button17Click(TObject *Sender)
{if(Button16->Visible==0){
   Button16->Caption=Button17->Caption;
   Button16->Visible=1;    Button17->Visible=0;}
if(Button13->Visible==0){
   Button13->Caption=Button17->Caption;
   Button13->Visible=1;    Button17->Visible=0;}
if(Button2->Caption=='1' &&   Button3->Caption=='2' && Button4->Caption=='3'
&&   Button5->Caption=='4' &&   Button6->Caption=='5' &&
  Button7->Caption=='6' &&Button8->Caption=='7' && Button9->Caption=='8'
&&   Button10->Caption=='9' && Button11->Caption=="10" &&
  Button12->Caption=="11" &&   Button14->Caption=="13" &&
  Button15->Caption=="14" )
     ShowMessage("Yutting kunti");
}void __fastcall TForm1::Button13Click(TObject *Sender)
{if(Button17->Visible==0){
   Button17->Caption=Button13->Caption;
   Button17->Visible=1;    Button13->Visible=0;}
if(Button12->Visible==0){
   Button12->Caption=Button13->Caption;
   Button12->Visible=1;    Button13->Visible=0;}
if(Button9->Visible==0){
   Button9->Caption=Button13->Caption;
   Button9->Visible=1;    Button13->Visible=0;}
}void __fastcall TForm1::Button15Click(TObject *Sender)
{if(Button16->Visible==0){
   Button16->Caption=Button15->Caption;
   Button16->Visible=1;    Button15->Visible=0;}
if(Button11->Visible==0){
   Button11->Caption=Button15->Caption;
```

```
   Button11->Visible=1;    Button15->Visible=0;}
if(Button14->Visible==0){
   Button14->Caption=Button15->Caption;
   Button14->Visible=1;    Button15->Visible=0;}
}void __fastcall TForm1::Button14Click(TObject *Sender)
{if(Button15->Visible==0){
   Button15->Caption=Button14->Caption;
   Button15->Visible=1;    Button14->Visible=0;}
if(Button10->Visible==0){
   Button10->Caption=Button14->Caption;
   Button10->Visible=1;    Button14->Visible=0;}
}void __fastcall TForm1::Button10Click(TObject *Sender)
{if(Button14->Visible==0){
   Button14->Caption=Button10->Caption;
   Button14->Visible=1;    Button10->Visible=0;}
if(Button11->Visible==0){
   Button11->Caption=Button10->Caption;
   Button11->Visible=1;    Button10->Visible=0;}
if(Button6->Visible==0){    Button6->Caption=Button10->Caption;
   Button6->Visible=1;    Button10->Visible=0;}}
void __fastcall TForm1::Button11Click(TObject *Sender)
{if(Button15->Visible==0){    Button15->Caption=Button11->Caption;
   Button15->Visible=1;    Button11->Visible=0;}
if(Button12->Visible==0){    Button12->Caption=Button11->Caption;
   Button12->Visible=1;    Button11->Visible=0;}
if(Button10->Visible==0){    Button10->Caption=Button11->Caption;
   Button10->Visible=1;    Button11->Visible=0;}
if(Button7->Visible==0){    Button7->Caption=Button11->Caption;
   Button7->Visible=1;    Button11->Visible=0;}
}void __fastcall TForm1::Button12Click(TObject *Sender)
{if(Button8->Visible==0){
   Button8->Caption=Button12->Caption;    Button8->Visible=1;    Button12-
>Visible=0;}
if(Button11->Visible==0){    Button11->Caption=Button12->Caption;
   Button11->Visible=1;    Button12->Visible=0;}
if(Button13->Visible==0){    Button13->Caption=Button12->Caption;
   Button13->Visible=1;    Button12->Visible=0;}
if(Button16->Visible==0){    Button16->Caption=Button12->Caption;
   Button16->Visible=1;    Button12->Visible=0;}
}void __fastcall TForm1::Button6Click(TObject *Sender)
{if(Button10->Visible==0){    Button10->Caption=Button6->Caption;
   Button10->Visible=1;    Button6->Visible=0;}
if(Button2->Visible==0){    Button2->Caption=Button6->Caption;
   Button2->Visible=1;    Button6->Visible=0;}
if(Button7->Visible==0){    Button7->Caption=Button6->Caption;
```

```
   Button7->Visible=1;    Button6->Visible=0;}
}void __fastcall TForm1::Button7Click(TObject *Sender)
{if(Button3->Visible==0){    Button3->Caption=Button7->Caption;
   Button3->Visible=1;    Button7->Visible=0;}
if(Button6->Visible==0){    Button6->Caption=Button7->Caption;
   Button6->Visible=1;    Button7->Visible=0;}
if(Button8->Visible==0){    Button8->Caption=Button7->Caption;
   Button8->Visible=1;    Button7->Visible=0;}
if(Button11->Visible==0){    Button11->Caption=Button7->Caption;
   Button11->Visible=1;    Button7->Visible=0;}
}void __fastcall TForm1::Button8Click(TObject *Sender)
{if(Button4->Visible==0){    Button4->Caption=Button8->Caption;
   Button4->Visible=1;    Button8->Visible=0;}
if(Button7->Visible==0){    Button7->Caption=Button8->Caption;
   Button7->Visible=1;    Button8->Visible=0;}
if(Button9->Visible==0){    Button9->Caption=Button8->Caption;
   Button9->Visible=1;    Button8->Visible=0;}
if(Button12->Visible==0){    Button12->Caption=Button8->Caption;
   Button12->Visible=1;
   Button8->Visible=0;}}void __fastcall TForm1::Button9Click(TObject *Sender)
{if(Button5->Visible==0){
   Button5->Caption=Button9->Caption;
   Button5->Visible=1;
   Button9->Visible=0;}
if(Button8->Visible==0){
   Button8->Caption=Button9->Caption;
   Button8->Visible=1;
   Button9->Visible=0;}
if(Button13->Visible==0){
   Button13->Caption=Button9->Caption;
   Button13->Visible=1;
   Button9->Visible=0;}
}
void __fastcall TForm1::Button18Click(TObject *Sender)
{ AnsiString c;
if(Button17->Visible==0)
switch(rand()%6)
    {case 1: c=Button2->Caption;
            Button2->Caption=Button6->Caption;
            Button6->Caption=c;          c=Button3->Caption;
            Button3->Caption=Button7->Caption;
            Button7->Caption=c;          c=Button4->Caption;
            Button4->Caption=Button8->Caption;
            Button8->Caption=c;          c=Button5->Caption;
            Button5->Caption=Button9->Caption;
```

```
        Button9->Caption=c;         break;
    case 2:c=Button6->Caption;
        Button6->Caption=Button10->Caption;
        Button10->Caption=c;            c=Button7->Caption;
        Button7->Caption=Button11->Caption;
        Button11->Caption=c;            c=Button8->Caption;
        Button8->Caption=Button12->Caption;
        Button12->Caption=c;            c=Button9->Caption;
        Button9->Caption=Button13->Caption;
        Button13->Caption=c;        break;
    case 3:c=Button10->Caption;
        Button10->Caption=Button14->Caption;
        Button14->Caption=c;            c=Button11->Caption;
        Button11->Caption=Button15->Caption;
        Button15->Caption=c;        break;
    case 4:c=Button2->Caption;
        Button2->Caption=Button3->Caption;
        Button3->Caption=c;             c=Button6->Caption;
        Button6->Caption=Button7->Caption;
        Button7->Caption=c;            c=Button10->Caption;
        Button10->Caption=Button11->Caption;
        Button11->Caption=c;            c=Button14->Caption;
        Button14->Caption=Button15->Caption;
        Button15->Caption=c;        break;
    case 5:c=Button3->Caption;
        Button3->Caption=Button4->Caption;
        Button4->Caption=c;             c=Button7->Caption;
        Button7->Caption=Button8->Caption;
        Button8->Caption=c;             c=Button11->Caption;
        Button11->Caption=Button12->Caption;
        Button12->Caption=c;            c=Button15->Caption;
        Button15->Caption=Button16->Caption;
        Button16->Caption=c;        break;
    case 6:c=Button4->Caption;
        Button4->Caption=Button5->Caption;
        Button5->Caption=c;             c=Button8->Caption;
        Button8->Caption=Button9->Caption;
        Button9->Caption=c;         break;    } else  ShowMessage("Istimos bo'sh
katakni 4x4 koordinataga ko'chiring!");}
```

10.1.rasm. Tog'ri oyna

10.2.rasm. Teskari oyna.

Nazotar savollari

1. Marge sort algoritmini tushuntiring?
2. Marge sort da amal bajarishga qancha vaqt ketadi?
3. Ushbu algoritm necha bosqichda amalga oshiriladi?

11 - Laboratoriya ishi.

Mavzu: O'yinlarda α va β izlash

Ishdan maqsad: Ushbu Laboratoriya ishii natijasida o'yinlarning algoritmini tuzishda foydalaniladigan α va β izlash usulini ko'rib chiqish va amaliyotda qo'llay olish.

Masalaning qo'yilishi. α va β izlashdan foydalanib labirintda ikki nuqta orasidagi yo'lni toppish dasturini tuzing.

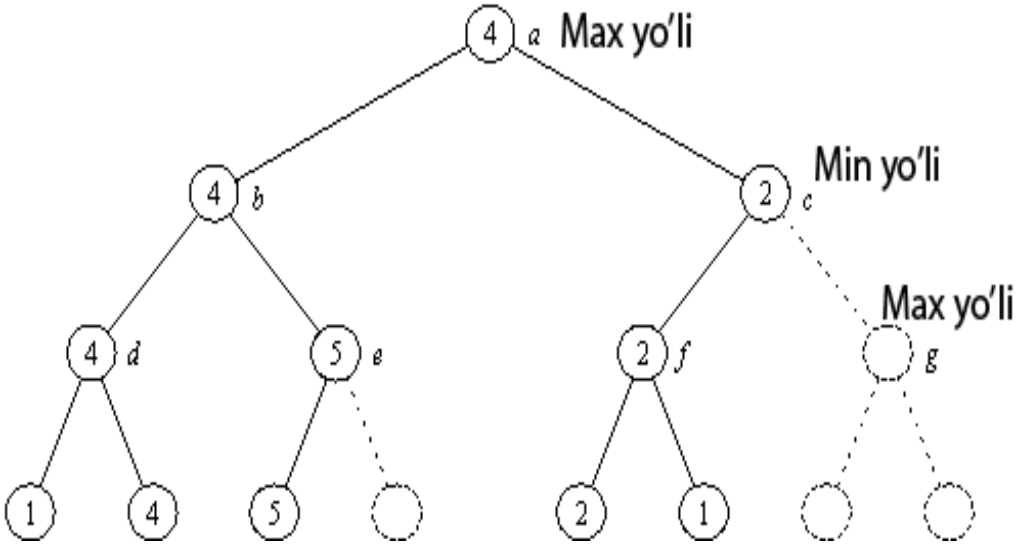
Uslubiy ko'rsatmalar: α va β izlash usulini – bu o'yin davomida qadamlarning sonini kamaytirish, yani avvaldan qo'yilishi mumkin bo'lgan ammallarning sonini aniqlab berish uchun qo'llaniladi. Bu optimallashtirish ham daraxtlar va graflar bilan bog'liq bo'lib ularning optimallashgan ko'rinishi hisoblanadi.

Alfa-beta kesish algoritmi minimaks algoritmi daraxtida baholangan tugunlarni qisqartirish uchun ishlatiladigan qidiruv algoritmi bo'lib, u antagonik o'yinlar va mashina o'yinlari uchun ishlatiladi. Bu algoritmning asl ma'nosi shundaki, qidiruv daraxtning baholanayotgan tarmog'i baho ffunksiyasi har qanday holda ham oldingi tarmoqdagidan yomon bo'lsa u holda qidiruv to'xtatiladi.

Bu algoritm mustaqil ravishda bir necha olimlar tomonidan bir-biridan behabar holda yaratilgan. Uni ilk marta "Aproksimatsiya" deb nomlashgan. 1956-yilda Makkarti ismli olim Darmurd seminarida bu g'oya haqida aytib o'tgan. 1963-yilda Aleksandr Brudno mustaqil ravishda algoritmni yaratdi va uni ommaga

havola etdgan. 1975-yilda esa Donald Kunt va Ronald Mur algoritmga “beta”-kesishishni qo’shib mukammallashtirishdi.

Shaxmat kabi o’yinlarda yo’lni topish uchun kompyuterda juda katta mumkin bo’lgan yo’llarni ko’rib chiqishga to’g’ri keladi. Bu jarayonni ma’lumotlarni yo’qotmasdan tezlashtirish uchun odatda alfa-beta kesish algoritmi ishlatiladi.



11.1.rasm. Yo’lni topish

Misol qilaylik bizda ikkita yo’l varianti bor. Agar biz ulardan biri ikkinchisidan yomonligini bilsak, uni qanchalik yomonroq yo’l ekanini bilishimiz shart emas. Buni qidiruv daraxtida ko’ramiz(11.1-rasm). Qidiruv quyidagicha amalga oshirilaid:

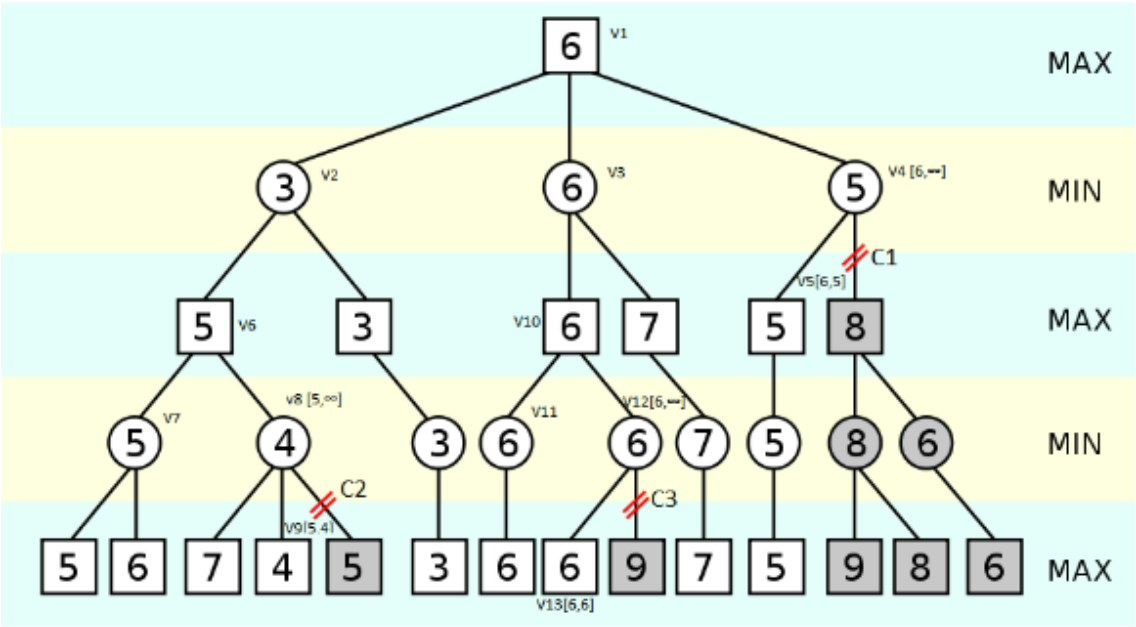
7) a nuqtadan boshlaymiz
8) b nuqtaga o’tamiz
9) d nuqtaga o’tamiz
10) d ning vorislaridan eng katta baholisini olamiz, V(d) = 4
11) b nuqtaga qaytamiz va e ga o’tamiz
12) e pozitsiyaning bahosi 5 ga teng bo’lgan 1-vorisini ko’rib chiqamiz. Bu vaqtda e pozitsiyada bahosi 5 dan kam bo’lmasligi kafolatlangan Maks aniqlanadi. Bu Min e pozitsiyaning bahosini bilmagan holda u uchun b yo’l e da d ga nisbatan yomonroq.

11.1-rasmdagi daraxtda alfa-beta algoritm qo’llangandan keying holat tasvirlangan. Uzuq chiziq chiziq orqali alfa-beta kesish belgilangan.

Usulning asosiy mohiyati shundan iboratki tarmoqlarni to’liq o’rganish natijasida olingan baholarni qolgan taxmin qilinayotgan eng yaxshi natijalar bilan solishtirishdir. Ayrim hollarda ba’zi hisob-kitoblar ortiqchaligini ko’rsatiish mumkin. Minmaks usuldan farqli ravishda bu usulda to’liq birin-ketin barcha usullarni ko’rib chiqilmaydi, balki daraxtni ba’zi qismlari tashlab yuboriladi.

Umuman olganda α va β izlash usulida alfa – maksimumni olgan o'yinchining eng kam olish chegarasi, beta bo'lsa minimumni olgan o'yinchining eng ko'p olish chegarasi deb qabul qilish mumkin.

- *alpha = max (alpha, f(Vi)) maksimallashtirish qismi uchun;*
- *alpha = min (alpha, f(Vi)) minimallashtirish qismi uchun;*



11.2.rasm. Bo'linish oynasi

Bu yerda:

- Vi – yechimning tugunlari, i chi korrelyatsiya qadamidaa yakunlanadi;
- Ci – i chi qadamdagi alpha va betta cheklashuv;
- MIN, MAX – maksimallashtish va minimallashtirish bo'limlarning chegaralari. E'tibor berib qarasangiz qadamlarda ular ketma ket keladi. Bunda maksimmallashtirishdagi qadam faqa shu i chi ketma – ketlikdagi shi bo'limga tegishliligini ola oladi, minimallashtirishdagi esa o'z navbatida min dagisini tanlay oladi.

Dastur kodi.

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
#include<iostream.h>
#include<time.h>
#include<math.h>
#include<stdlib.h>
#include<vector>
#include <algorithm>
#include <fstream.h>
```

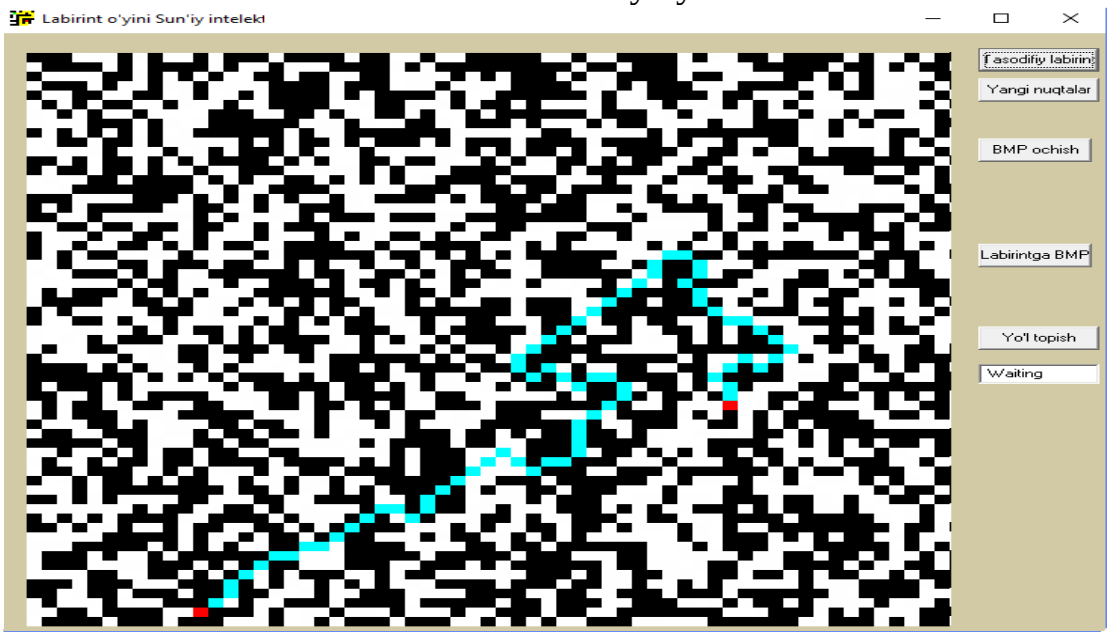
```
using namespace std;
Graphics::TBitmap* pix = new Graphics::TBitmap();
RGBTRIPLE* t;
bool stop;
bool step;
bool reduction;
#define GRID_SIZE 61
#define TS 10
int grid[GRID_SIZE][GRID_SIZE];

// Function Prototypes
void renderGrid();
void setSE();
void makeGrid();
void resetGrid();

enum
{       wallTile,
        openTile,
        startTile,
        endTile,
        pathTile,
};
class Astar
{public:      enum
        {
                SEARCHING,
                SUCCEEDED,
                FAILED,
        };      class Node
        {public:
                Node()
                {parent = NULL;  g = 0.0;h = 0.0;
                        f = 0.0;                }float getDistanceEstimate( int endx,
int endy )
                {
        float dx = (float)( (float)x - (float)endx );
                        float dy = (float)( (float)y - (float)endy );

                        return ((dx*dx) + (dy*dy));
                }
                bool isSameState( Node *rhs )
                {                       if( x == rhs -> x && y == rhs -> y )
                                return true;
                        else
```

```
                        return false;          }
        Node *parent;
        float g; // cost of this node + its parents
        float h; // heuristic estimate
        float f; // g + h
        int x, y;
        int type;
};      class HeapComparison
{public:
        bool operator() ( const Node *x, const Node *y ) const
        {                       return x->f > y->f;
        }       };
Astar() {           // Constructor
        clearLists();
}       void clearLists()
{               OPEN.clear();
        CLOSED.clear();
        SUCCESSORS.clear();   }
int getCoords( int xCoord, int yCoord )
{               if( xCoord < 0 || xCoord >= GRID_SIZE ||
                 yCoord < 0 || yCoord >= GRID_SIZE )
                return wallTile;
        return grid[xCoord][yCoord];
}       void setStartEnd( int x, int y, int q, int r)
{               startNode = new Node;
        endNode = new Node;
        //Initialize the start and ending nodes
        startNode -> x = x;
        startNode -> y = y;
        startNode -> type = startTile;
        endNode   -> x = q;
        endNode   -> y = r;
        endNode   -> type = endTile;
        OPEN.push_back( startNode );
        // Sort elements in heap
        push_heap( OPEN.begin(), OPEN.end(), HeapComparison() );
        // Initialise counter for search steps
        stepCounter = 0;
}       unsigned int Search()
{//Check to see if the search succeeded or failed
        if( ( currentState == SUCCEEDED) || ( currentState ==
FAILED ) )              return currentState;
        //The search fails if the OPEN list is empty (no more states to
search)          if( OPEN.empty() )
        {       currentState = FAILED;
```

```
                    return currentState;
                    }
                    if( (getCoords( (thisNode -> x)+1, (thisNode -> y) ) != wallTile)
                    && !((parentX == (thisNode -> x)+1) && (parentY ==
(thisNode -> y)))  )
        {newNode = new Node;
        newNode -> x = (thisNode -> x)+1;
        newNode -> y = thisNode -> y;
        newNode -> type = getCoords( (thisNode -> x)+1, (thisNode -> y) );
        SUCCESSORS.push_back( newNode );
        }vector< Node * >::iterator closedlist;
        for( closedlist = CLOSED.begin();
                closedlist != CLOSED.end(); closedlist ++ )
        {if( (*closedlist) -> isSameState( (*successor) ) )
        {       break;
        }}void makeGrid()
        {       int i;
                int j;
                for( i = 0; i < GRID_SIZE; ++i ) {   // Generate the walls and open
paths
                        for( j = 0; j < GRID_SIZE; ++j ) {
                                grid[i][j] = random(2);
                        }       }       return;
        }void setSE()
        {       unsigned int x, y, q, r;
           Form1->EditStatus->Text = " Waiting";
           resetGrid();
                do {            // Set Start point
                        x = (unsigned)(random(GRID_SIZE));
                        y = (unsigned)(random(GRID_SIZE));
                } while( grid[x][y] == wallTile );
                grid[x][y] = startTile;
                do {    // Set End point
                        q = (unsigned)(random(GRID_SIZE));
                        r = (unsigned)(random(GRID_SIZE));
                } while( grid[q][r] == startTile || grid[q][r] == wallTile );
                grid[q][r] = endTile;
                aStarSearch.setStartEnd( x, y, q, r );
                return;}
                if ( currentSearchState == Astar::SUCCEEDED ) {
                        aStarSearch.writePathToGrid();
                        renderGrid();
             Form1->EditStatus->Text = " Success";
                }       else {
```

```
                renderGrid();      //Print the grid without writing the path 'X's to
it
                Form1->EditStatus->Text = " Impossible";
            }       aStarSearch.clearLists();}void initialize(){   randomize();
           makeGrid();setSE();       renderGrid();
  Form1->EditStatus->Text = " Waiting";}
void __fastcall TForm1::FormPaint(TObject *Sender)
{   initialize();
  renderGrid();}
void __fastcall TForm1::ButtonRandomMazeClick(TObject *Sender)
{    initialize();
}void __fastcall TForm1::ButtonFindPathClick(TObject *Sender)
{    findQuickestPath();
}void __fastcall TForm1::ButtonNewSEClick(TObject *Sender)
{    setSE();
   renderGrid();}
```

11.3.rasm. Tasodufiy oyna.



11.3.rasm. Natijalarni ko’rish

Nazorat savollari

1. α va β izlash usuli qayerlarda ishlatiladi?
2. α va β izlashda α va β nima?

12 - Laboratoriya ishi

Mavzu: Muammolar cheklovini qondirish

Ishdan maqsad: Sun’iy intellektda muammolar cheklovini qondirishni o’rganish.

Masalani qo’yilishi. Muammolar cheklovini qondirishni toppish.

Uslubiy ko’rsatmalar: Sun’iy intellektning asosiy vazifalaridan biri cheklovlarni qondirish vazifasidir. Cheklovlarni qondirish nazariyasi sun’iy intellektning kombinator yechimlari va taqdimlari uchun qulay apparat va oddiy formal sxemasidir.

Cheklovlarni qondirishning yechish masalasining maqsadi berilgan cheklovlarni qondiruvchi o’zgaruvchilar qiymatini topish hisoblanadi.

Cheklovlarni qondirish masalasi yechimining mavjudligi muammosi to’liq NP bo’lib hisoblanadi. Kombinator masalalarni effektiv yechish va deklarativ ta’riflash uchundasturlash paradigmasi bo’lib hisoblanadigan cheklovlarda dasturlash cheklovlarni qondirish bilan chambarchas bog’liq. Ferma teoremasi, propozitsional mantiqning bajarilganlik masalasi va graflar nazariyasidagi graflarning izomorflik masalasi kabi ma’lum bo’lgan ko’pgina klassik kombinator masalalar cheklovlarni qondirish masalasi ko’rinishida aniq ifodalanadi. Matematikada avval qo’yilgan bitta masala – xususiy holda kartalarni bo’yash masalasi bo’lib hisoblanuvchi grafni bo’yash masalasiga to’xtalamiz. Cheklovni qondirish masalasi ko’rinishidagi bo’yash masalasini aniq ifodalanishi bo’yaluvchi graf o’zgaruvchisi mos cho’qqiga qo’yiladi. Mumkin bo’lgan rang o’zida o’zgaruvchi domenini aks ettiradi, yonma-yon uchlar orasidagi tengsizlikni cheklash esa vazifalarni cheklash hisoblanadi.

Aytish kerakki, bu yerda barcha aspektlarni, yo’naltirilgan cheklashlarni qondirish nazariyasini va cheklashlarda dasturlashni to’liq tavsiflash imkonsiz. Shuning uchun to’liq ma’lumotni Rassel S, Norvig Plarning tarjima monografiyalaridan olish mumkin.

Cheklovlarni qondirishning optimallashtirish muammosini hal qilish quyidagicha vazifalarni bir ketma-ketlikda hal qilish orqali kamaytirilishi mumkin. Yechim buyechimdan ko’ra obeyektiv funksiyaning qiymati yaxshiroq bo’lsin degan shart qo’yadigan obyektiv funksiyaga tegishli cheklov qo’shilganidan keyin joylashishi mumkin. Shu paytgacha ishlab chiqilgan bu boshlang’ich qiymatning ketma-ket tuzatilishlari masalaga ruxsat berilmagunicha optimal yechimni qidirishga imkon beradi.

1-misol. Cheklovlarni qondirishning eng oddiy masalasi bo’lib tenglamalar sistemasini yechish hisoblanadi. Chekli F maydondagi chiziqli tenglamalar sistemasi hisoblansin. U yechimga egami? Ravshanki, Tenglama o’zgaruvchilari diapason hosil qilishi, ko’pgina tenglama yechimiga mos keluvchi barcha kortejlar cheklash munosabatini hosil qilishi sababli bu misoldagi har bir alohida tenglama cheklash bo’lib hisoblanadi.

2-misol. 3-bajariluvchili propozitsional standart masalasi(3-SAT) konyuksiya, dizyuksiyadan tashkil topgan mantiqiy propozitsional formulalar vazifalarini aniqlaydi. Har bir dizyunkt 3ta literaldan tashkil topadi va rostlik formulalarini bajaruvchi o’zgaruvchi qiymatlariga egami degan savolga javob beradi.

Cheklovlarni qondirish masalasi SAT masalasini yechish masalasidagi o’zgaritirish bo’lishi mumkin. O’zgaruvchilarni tanlaymiz. O’zgaruvchilar faqat va faqat o’zgaruvchi qiymat o’zlashtirganda “rost” qiymatini qabul qiladi. Har bir o’zgaruvchi uchun bir vaqtda o’zgaruvchiga turli xil qiymatlar o’zlashtirilmasligini kafolatlash uchun shu o’zgaruvchining barcha qiymat juftliklariga klauzlar(dizyunktlar) qo’shiladi. Dizyunkt qo’shilishi o’zgaruvchiga hech bo’lmasa bitta qiymat o’zlashtirilishini kafolatlash uchun qo’shiladi.

3-misol. Cheklovlarni qondirishninf har qanday aniq masalasi mantiqiy mo’rininshda ifodalanishi mumkin. Albatta, predikatlar va munosabatlar orasida standart moslikdan foydalanib, cheklovlarni qondirish masalasini birinchi tartibli formula ko’rinishida qayta yozish mumkin. Savol esa formula bajarilgan bo’lib hisoblanadimi deb qo’yiladi. Bu masala odatda ma’lumotlar bazasi nazariyasida foydalaniladi. Buning uchun yana quyidagi misolni ko’rsatamiz.

4-misol. Relyatsion ma’lumotlar bazasi ko’p sondagi chekli jadvallar kabi ko’rib chiqiladi. Jadval sxemalardan va aniq ma’lumotlardan foydalanadi. Bu yerda sxema chekli sondagi ko’plab atributlar. Har bir atribut o’ziga mos keluvchi mumkin bo’lgan domen deb nomlanuvchi qiymatlarga ega bo’ladi. Relyatsion ma’lumotlar bazasining standart vazifasi konyuktiv so’rov yechimi mavjudligini so’rovchi konyuktiv so’rovlarni baholash vazifasi hisoblanadi. Relyatsion ma’lumotlar bazasi ustidagi konyuktiv so’rovlar aniq cheklovlarni qondirish masalasi misoliga mos keladi. Bunda bazaning terminlari quyidagilarga mos keladi: “atributlar” “o’zgaruvchilar”ga almashadi, “jadvallar” “cheklovlar”ga almashadi, “sxemalar” “diapazonlar”ga, “aniq ma’lumotlar” “cheklov munosabatlariga” va “qatorlar” “kortejlar”ga almashadi. Demak, konyuktiv so’rov cheklovlarni qondirishning aniq vazifasiga ekvivalent bo’lib hisoblanadi. Bunda o’zgaruvchilar – bu so’rov atributlari. So’rovdagi har bir atomar formula uchun cheklov diapazoni cheklov munosabatlari va o’zgaruvchi formulalari ro’yxati kabi cheklovlar topiladi.

«Send More Money» masalasi uchun cheklovlarni qondirish modelini ko'ramiz. Talaba uyiga quyidagi kodlangan telegrammani jo'natyapti:

$$\begin{array}{r} SEND \\ +\ MORE \\ \hline MONEY \end{array}$$

Telegramda foydalanilgan 8 ta kodlangan harfdan 8 ta kodlangan raqamni topish vazifasi berilgan. Aytish kerakki, bu masala butun sonli dasturlash modeli yordamida bajarilishi mumkin.

Shu paytda mantiqiy tahlil uning yechimini topish imkonini beradi. Bu yerda S ha M ham nol bo'lishi mumkin emas. Davom etamiz, M uchun yagona qiymat 1 bo'lishi mumkin, M = 1. Keyin, M = 1 ligidan S = 9 bo'lishi mumkin, Ko'rsatish mumkinki, O belgisi 0 ga mos keladi. Davom etamiz, O = 0 shartda 100 liklarga o'tib, E va N har xil bo'lishini tahmin qilamiz. Bunda N = E + 1ni olamiz. Mos mantiqiy tahlilni davom ettirib masala yechimini topamiz: E = 5, N = 6, D = 7, R = 8, Y = 2. Bu masalaning asosiy murakkabligi «barcha raqamlar har xil» cheklovini qayd etish zaruriyatidir.

Masalaning cheklovini yozamiz:

O'zgaruvchilar: {S, E, N, D, M, O, R, Y}

Domenlar: {0,1,…,9}

Cheklovlar:

C1 : $\forall x, y \in \{S, E, N, D, M, O, R, Y\}, x \neq y$

C2 :M = 0 or M = 1

C3 : (1000 x S + 100 x E + 10 x N +D) + (1000 x M + 100 x O + 10 x R +E) = (10000 x M + 1000 x O + 100 x N + 10 x E +Y)

Ikkinchi formulalashtirish

O'zgaruvchilar: {S, E, N, D, M, O, R, Y} + {C1, C2, C3}

Domenlar: $\forall x \in$ S, E, N, D, M, O, R, Y, $D_x = 0 \ldots 9$

$\forall y \in$ { C1, C2, C3}, $D_y = 0, 1$

Cheklashlar:

C1 : $\forall x, y \in \{S, E, N, D, M, O, R, Y\}, x \neq y$

C2 : M = 0 or M = 1

C3 : D + E = 10 x C1 + Y

C4 : N + R + C1 = 10 x C2 + E

C5 : E + O + C2 = 10 x C3 + N

C6 : S + M + C3 = 10 x M +O

Masalaning Prolog dasturlash tilidagi kodi:

```
sendmore(Digits) :-
Digits = [S,E,N,D,M,O,R,Y], % O'zgaruvchilarni yaratish
Digits :: [0..9], % o'zgaruvchilar domenlari
S #\= 0, % Cheklov: S 0dan farq qilishi kerak
M #\= 0, % Cheklov: M 0dan farq qilishi kerak
alldifferent(Digits), % Barcha o'zgaruvchilar turli xil qiymatlarni qabul qilishi kerak
1000*S + 100*E + 10*N + D % boshqa cheklovlar
+ 1000*M + 100*O + 10*R + E
#= 10000*M + 1000*O + 100*N + 10*E + Y,
labeling(Digits). % qidiruvni boshlash
```

Cheklovli dasturlash cheklovlar formasida ko’rsatilgan o’zgaruvchilar orasidagi munosabatdagi dasturlashning paradigmalari hisoblanadi. Cheklashlar

bajarish qadamlari ketma-ketligini emas harakatli yechimlar xususiyatini aniqlovchi umumiy ibtidoiy qat'iy dasturlash tillaridan farq qiladi. Bu deklarativ dasturlashning cheklangan formasini dasturlashtiradi. Cheklovlar dasturlarida foydalaniladigan cheklash turli xil ko'rinishlarda bo'ladi: cheklovlarni qondirish masalalarida ishlatiladiganlari(masalan A yoki B rost), simpleks algoritmlarni yechuvchi(masalan, $x \leq 5$) va boshqalar. Odatda, cheklovlar dasturlash tilida yoki mavjud alohida dasturlash kutubxonalarida quriladi .

Cheklovli dasturlash mantiqiy dasturlashdagi cheklovlarni kirituvchi cheklovli mantiqiy dasturlash bilan chambarchas bog'liq.Bunday mantiqiy dasturlash variantlarining paydo bo'lishi Jaffar va Lassez nomlari bilan bog'liq. Ular 1987-yilda belgilangan cheklovlar sinfini ochishdi. Cheklovlardagi mantiqiy dasturlashning birinchi ishlatilishi CLP(R) va CHIP da bo'lgan.

Nazorat savollari

1. Cheklovlarni qondirish masalasi nima?
2. Cheklovlarni qondirish masalasi qayerlarda ishlatiladi?

13 - Laboratoriya ishi

Mavzu: Matnni tanish

Ishdan maqsad: Tasvirlar ustida ishlovchi algoritmlarni o'rganish. Tasvirdagi matnlarni tanishni o'rganish.

Masalaning qo'yilishi: Tasvirdagi matnlarni tanuvchi algoritmlarni ko'rib chiqish.

Uslubiy ko'rsatmalar: Insonning ko'zi atrof-muhitdagi ko'pgina obyektlarni, ularning rangini ajratan oladi. Hozirda insonning turli xususiyatlariga o'xshash texnologiyalar kompyuterlar yordamida yaratilmoqda va tibbiyot, ta'lim, xavfsizlik va boshqa sohalarga joriy qilinmoqda.

Obrazlarni tanuvchi tizimlarni yaratish, hozirda ham, birmuncha murakkab vazifa bo'lib turibdi. Biroq hozirda harflarni tanuvchi , yuzlarni tanuvchi, shtrix-kodlarni tanuvchi, ovozni, mashina nomerlarini ta'nuvchi ko'pgina tizimlar mavjud.

Matnni tanish tizimlaridagi muhim rol bo'lib matnni tanish hisoblanadi.

Rasmdagi matnni tanish ikki bosqichda amlga oshiriladi:

1. Rasmdan matn bo'lishi mumkin bo'lgan sohalarni aniqlash.
2. Aniqlangan sohalarda matn bor-yo'qligini tekshirish.

Rasmdan matnni tanib olish yangi muammo hisoblanmaydi, asosiy muammo rasmdagi matnni aniqlashning optimal usuli yo'qligidadir.

Matnni tanish bo'yicha olib boriluvchi ilmiy izlanishlar:

Matnni tanish xalqaro mavzu bo'lib, bu bo'yicha dunyoning ko'pgina universitetlari shug'ullanmoqda.

Afina universiteti o'zining elektron kutubxonasida "Text detection in video frames" , "Text Detection in Images and Videos" ishlarini saqlaydi. Rochestr instituti o'z eletron bazasida "Detection of Text in Video" maqolasi mavjud.

Veyvlet o'zgartirish (ing. Wavelet transform) – o'zida signalli veyvlet funksiyalar o'ramini aks ettiruvchi integral o'zgartirishdir. Veyvlet o'zgartirish vaqtga bog'liq berilishdagi signalni chastota-vaqt ko'rinishidagi signalga o'giradi. Ushbu atama(wavelet)ni inglizchadan tarjima qiladigan bo'lsak "kichik to'lqin" degan ma'noni bildiradi. Veyvlet – bu muayyan vaqt va chastota bo'yicha belgilangan formadagi matematik funksiyaning umumiy nomi.

Veyvlet shartlari

Veyvlet o'zgartirishlar mavjud bo'lishi uchun veyvlet funksiyalar quyidagi shartlarni qanoatlantirishi kerak:

Veyvlet chekli energiyaga ega bo'lishi kerak:

$$E = \int_{-\infty}^{\infty} |\Psi(t)|^2 dt < \infty$$

Agar Ψ(t) uchun Ψ(f) - fure o'zgartirish bo'lsa,

$$\Psi(t) = \int_{-\infty}^{\infty} \Psi(t) e^{-i(2\pi f)t} dt$$

Bunda quyidagi shart bajarilishi kerak:

$$C_\Psi = \int_{-\infty}^{\infty} \frac{|\Psi(f)|^2}{f} dt < \infty$$

Bu shart ruxsat etilgan shart deyiladi, va bundan kelib chiqadiki – veyvlet nol chastotali komponentga yaqinlashganda $\Psi(0) = 0$ shartini qanoatlantirishi kerak yoki boshqacha holatda veyvlet $\Psi(t)$ o'rtacha nolga teng bo'lishi kerak.

Rasmlarni qayta ishlash algoritmlari

obyektlarni tanishda rangli rasm bilan birgalikda oq-qora rasmdan ham foydalanish mumkin. Rangli rasmdan foydalanganda biz obyekt haqida ko'proq ma'lumot olishimiz mumkin. Oq-qora rasm bilan ishlaganimizda esa vaqt va tezlikdan yutamiz.

Bu ishimizda biz rangli rasmni qayta ishlashda rasmlarning kulrang ton(greyscale) shkalasi va binarlash amali ishlatiladi.

Greyscale - kulrang tonli rangda aks etuvchi rangli rasm ko'rinishi. bu algoritm rangli ramni kulrang ton shkalasiga o'tkazish imkonini beradi. Algoritm rasmning kengligi va balandligi bo'ylab amalga oshadi.

Quyi boshlanishdan binarlash bir muncha sodda bo'lib, bunda faqat boshlanishning bitta qiymati ishlatiladi:

$$f'(m,n) = \begin{cases} 0, f(m,n) \geq t \\ 1, f(m,n) < t \end{cases}$$

Kriteriya bilan birgalikdagi barcha qiymatlar 1 bo'ladi. Bu holatda t-0(qora) boshlang'ichdan katta barcha piksellarning qiymatlari(ampletudalari) ham 255(oq) bo'ladi. ish algoritmning natijaviy ko'rinishi quyidagi rasmda:

13.1. rasm. Rasmni qayta ishlashning asosiy usullari

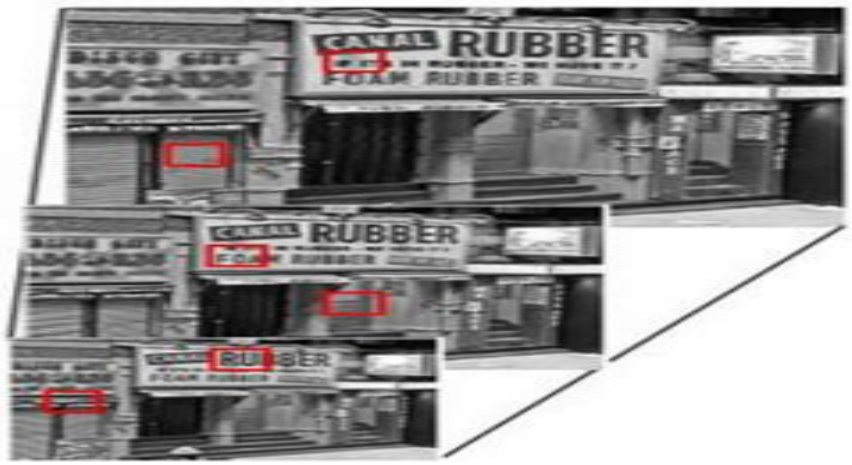
Gistogramma “Dokument” ko’rinishidagi rasmning matnini belgilab olishda ishlatiladi. Bunaqangi rasmlarda matnlar qator va ustun bo'yicha tekislangan bo'ladi. Gistogramma tuzishning ikki xil usuli mavjud: qidiruv paytida butun sahifa baholab chiqiladigan yuqoridan pastga usuli va matnli sohalarni aniqlash uchun simvollar belgilanib chiqiluvchi pastdan yuqoriga usuli.

Birinchi imkoniyat sahifani proyeksiyalash usulini o'z ichiga oladi. Bu usulning asosiy g'oyasi belgilangan qatordagi qora rangli piksellarni sanab chiqishni va vertikal yoki gorizontal gistogrammani yaratishni o'z ichiga oladi. Keyin gistogrammaning rangli sohalarini kesish amalga oshiriladi. Vertikal va gorizontal proyeksiyalar bir –birining ustiga qo'yiladi. Agar dokumentda rasm bo'lmasa bu usul yaxshi natija beradi.

«Pastdan yuqoriga» usuli Docstrum algoritmi va Voronov diagrammasidan foydalanuvchi algoritm bo'lib hisoblanadi. Bu algoritmlar uchun gistogrammalar simvollar, so'zlar, qatorlar va bo'limlarga bog'liq komponentlar orasidagi masofaga asosan quriladi.

Dokument tipidagi rasmdan ko'ra ixtiyoriy rasmdan matnlarni ajratib olish uchun ko'proq usullar mavjud. Ularni teksturali foydalanish va belgilangan sohalardan foydalanish usullariga bo'lish mumkin.

Matn teksturasi oddiy rasm teksturasidan ancha farq qiladi.Tekstura belgilarini qurish uchun kattalashtirilgan yoki kichraytirilgan o'lcham bo'yicha rasm «piramida»si quriladi(2-rasm). Keyin barcha rasmlar guruhiga piksel bo'yicha o'tish amalga oshiriladi va kosinus yoki veyvlet o'zgartirish yordamida tekstura belgilari aniqlanadi. Bu usul bir xil shriftdagi bir yo'nalishli tekstlar uchun yaxshi ishlaydi.



13.2.rasm. Belgilab olish.

Xulosa

Xulosa qilib aytganda matnni tanish yo'nalishi tasvirlar bilan ishlashning eng asosiy sohalaridan hisoblanadi. Tasvirlardan matnni tanishda tasvir ustida bir qancha algoritmlar asosida operatsiyalar bajarilganligi sababli ancha murakkab jarayon hisoblanadi. Mutaxasislarning eng asosiy vazifasi jarayonni amalga oshirishning optimal va sodda usulini topish hisoblanadi. Bugungi kunda ana shunday bir qancha usul va algoritmlar ishlab chiqilgan va yana ishlab chiqilmoqda

Ushbu mustaqil ishda tasvirdagi matnni tanishning veyvleto'zgartirish va greyscale algoritmlarini ko'rib chiqdik. Bu usul ancha sodda bo'lgani uchun ishlatish qulay hisoblanadi.

Nazorat savollari

1. Matnni tanish algoritmlari haqida gapiring
2. Veyvlet algoritmini tushuntiring
3. Matnni tanishda veyvlet algoritmi qaysi qismda ishlatiladi?

14 - Laboratoriya ishi

Mavzu: Mashina ta’limi

Ishdan maqsad: Mashina ta’limi tushunchasini o’rganish.

Masalaning qo’yilishi: Mashina ta’limidagi usul va algoritmlarni o’rganish.

Uslubiy ko’rsatmalar: Mashina ta’limi Sun’iy intellektning katta qismini tashkil qiluvchi bo’limi hamda o’rganuvchi algoritmlarni tuzish metodlarni izlaydi. Mashina ta’limi ikki turga bo’linadi:

- Pretsedentlar bo’yicha ta’lim yoki induktiv ta’lim. Berilgan tanlanmalar bo’yicha tanlanma elementlari orasidagi qonuniyatlarni topishga asoslanadi.
- Deduktiv ta’lim. Ekspert bilimlarini formallashtirish hamda ularni bilimlar bazasi ko’rinishida kompyuterga ko’chirish.

Mashina ta’limi matematik statistika, optimallashtirish metodlari va klassik matematik qoidalar bilan bog’liq. Ko’p induktiv ta’limning metodlari klassik

statistik yondashuvlarga alternativa sifatida ishlab chiqilgan. Mashina ta’limi ko’p metodlari axborotni olish va ma’lumotlarning intellektual analiziga bog’liq.

Mashina ta’limiga faqat matematik tarafdan emas balki amaliy tarafdan ham qarash lozim. Amaliyotda qo’llanuvchi algoritmlarni yaratish uchun nazatiy qism yetarli emas. Bunday metodlarni “yaxshi” ishlashi uchun qo’shimcha evristikalarni kashf etishga majbur qiladi. Har qanday mashina ta’limi tadqiqotlari amaliy qo’llash eksperimentlarsiz o’tmaydi. Bunday eksperimentlar modelli yoki real ma’lumotlar bilan o’tkaziladi hamda algoritmning amaliy qo’llashni tasdiqlab beradi.

Mashina ta’limining masalaning qo’yishilishiga qarab ikki turga bo’linadi:

Ustoz yordamida ta’lim(supervised learning) – ko’p tarqalgan tur. Har bir pretsedent o’z navbatida “obyekt va javob” juftlikni tashkil qiladi. Berilgan obyektlar xususiyatiga ko’ra javoblar o’rtasidagi bog’liqlik qonuniyatlarini topish algoritmlarini tuzishdan iborat ya’ni kirishga obyektning xususiyatlarni beriladi, chiqishda esa shu belgilarga mos javob hiqadi. Ushbu turga quyidagi masalalar kiradi:

- Klassifikatsiya masalasi (classification);
- Regressiya masalasi (regression);
- Ranjirlash masalasi (learning to rank);
- Bashoratlash masalasi (forecasting).

Ustozsiz ta’lim(unsupervised learning). Bunda javoblar berilmaydi va bu yerda obyektlar orasidagi bog’liqliklarni topish masalasi qo’yiladi. Ushbu turga quyidagi masalalar kiradi:

- Klasterizatsiya masalasi (clustering);
- Bog’liqlik qoidalarini topish masalasi (association rules learning);
- Chiqindilarni filtrlash masalasi (outliers detection);
- Ishonchlilik chagarasini qurish masalasi (quantile estimation);
- O’lchamni qisqartirish masalasi (dimensionality reduction).

Nazorat savollari

1. Sun'iy intellektda mashina ta'limining ishlatilishini tushuntiring
2. Mashina ta'limining keng tarqalgan turi qaysi?
3. Ustozsiz ta'limga kiruvchi masalalar qaysilar?