

CS206 Data Structures

Array List or Vector

Sung-eui Yoon (윤성의)
Department of Computer Science
KAIST

<http://sglab.kaist.ac.kr/~sungeui>

Class Objectives (Ch. 7.2)

- Understand the concept of array list (or vector)
 - How to increase its capacity
 - Discuss incremental and doubling strategies

The Array List ADT

- The Array List (=Vector) ADT extends the notion of array by storing a sequence of arbitrary objects
- An element can be accessed, inserted or removed by specifying its rank (number of elements preceding it)
- An exception is thrown if an incorrect rank is specified (e.g., a negative rank)
- Main array list operations:
 - object `elemAtRank(integer r)`: returns the element at rank `r` without removing it
 - object `replaceAtRank(integer r, object o)`: replace the element at rank with `o` and return the old element
 - object `insertAtRank(integer r, object o)`: insert a new element `o` to have rank `r`
 - object `removeAtRank(integer r)`: removes and returns the element at rank `r`
- Additional operations `size()` and `isEmpty()`

Applications of Array Lists

□ Direct applications

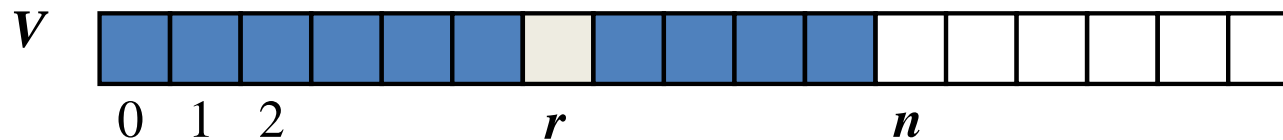
- Sorted collection of objects (elementary database)

□ Indirect applications

- Auxiliary data structure for algorithms
- Component of other data structures

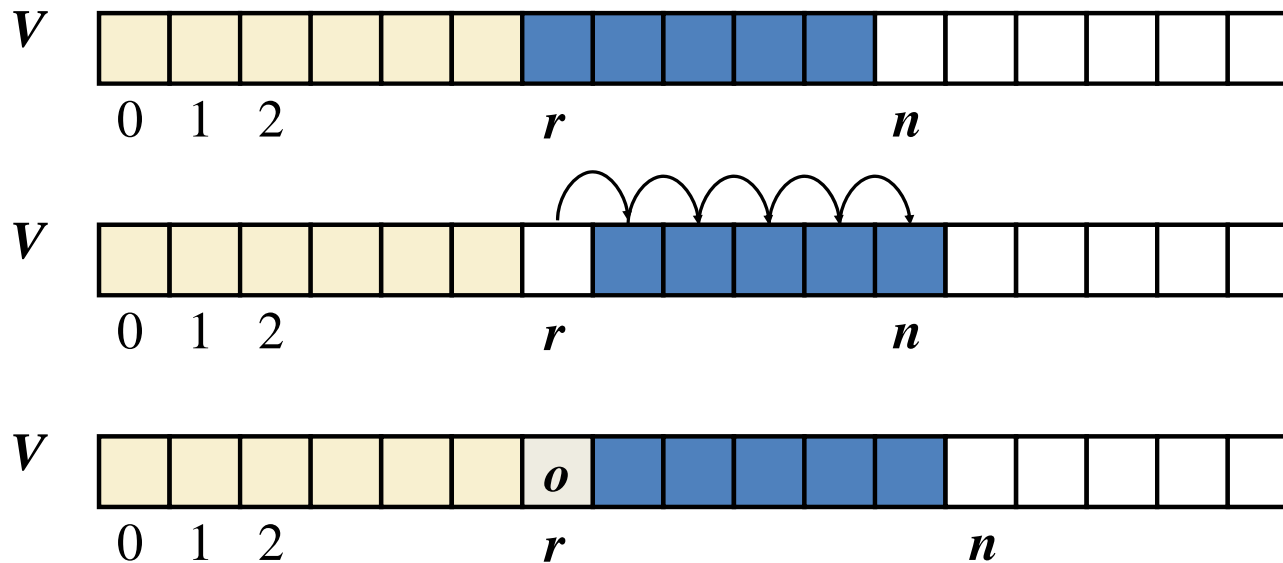
Array-based Implementation

- Use an array V of size N
- A variable n keeps track of the size of the vector (number of elements stored)
- Operation $\text{elemAtRank}(r)$ is implemented in $O(1)$ time by returning $V[r]$



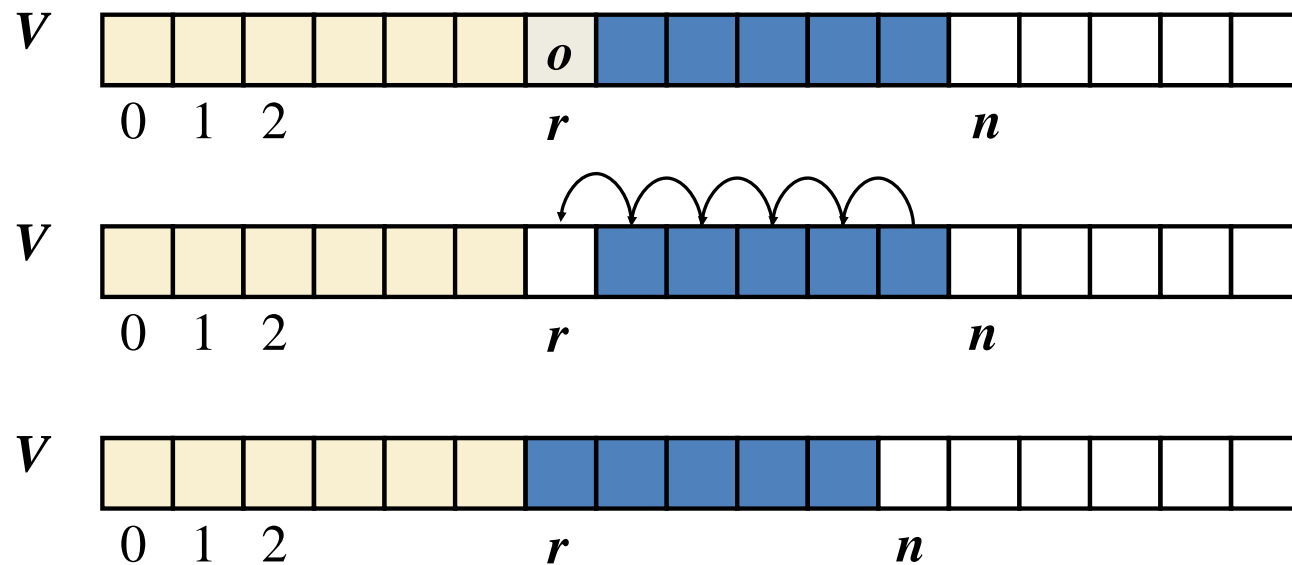
Insertion

- In operation `insertAtRank(r, o)`, we need to make room for the new element by shifting forward the $n - r$ elements $V[r]$, ..., $V[n - 1]$
- In the worst case ($r = 0$), this takes $O(n)$ time



Deletion

- In operation `removeAtRank(r)`, we need to fill the hole left by the removed element by shifting backward the $n - r - 1$ elements $V[r + 1], \dots, V[n - 1]$
- In the worst case ($r = 0$), this takes $O(n)$ time



Performance

- In the array based implementation of a Vector
 - The space used by the data structure is $O(n)$
 - `size`, `isEmpty`, `elemAtRank` and `replaceAtRank` run in $O(1)$ time
 - `insertAtRank` and `removeAtRank` run in $O(n)$ time
- If we use the array in a circular fashion, `insertAtRank(0)` and `removeAtRank(0)` run in $O(1)$ time
- In an `insertAtRank` operation, when the array is full, instead of throwing an exception, we can replace the array with a larger one

Growable Array-based Vector

- In a push operation, when the array is full, instead of throwing an exception, we can replace the array with a larger one
- How large should the new array be?
 - incremental strategy: increase the size by a constant c
 - doubling strategy: double the size

```
Algorithm insertAtRank( $r, o$ )  
if  $r = V.length - 1$  then  
   $A \leftarrow$  new array of  
    size ...  
  for  $i \leftarrow 0$  to  $r$  do  
     $A[i] \leftarrow V[i]$   
   $V \leftarrow A$   
  
// add the element at  $r$  here
```

Comparison of the Strategies

- We compare the incremental strategy and the doubling strategy by analyzing the total time $T(n)$ needed to perform a series of n push operations
- We assume that we start with an empty array list represented by an array of size 1
- We call amortized time of a push operation the average time taken by a push over the series of operations, i.e., $T(n)/n$

Incremental Strategy Analysis

- We replace the array $k = n/c$ times
- The total time $T(n)$ of a series of n push operations is proportional to

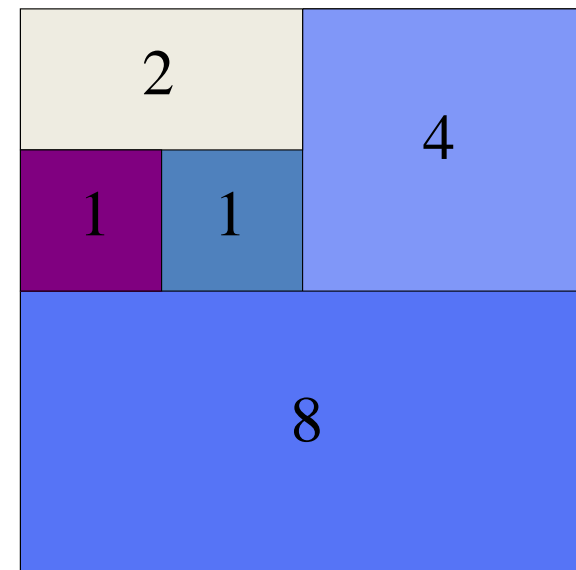
$$\begin{aligned}n + c + 2c + 3c + 4c + \dots + kc &= \\n + c(1 + 2 + 3 + \dots + k) &= \\n + ck(k + 1)/2 &= \end{aligned}$$

- Since c is a constant, $T(n)$ is $O(n + k^2)$, i.e., $O(n^2)$
- The amortized time of a push operation is $O(n)$

Doubling Strategy Analysis

- We replace the array $k = \log_2 n$ times
- The total time $T(n)$ of a series of n push operations is proportional to
$$n + 1 + 2 + 4 + 8 + \dots + 2^k = n + 2^{k+1} - 1 = 3n - 1$$
- $T(n)$ is $O(n)$
- The amortized time of a push operation is $O(1)$

geometric series



Class Objectives were:

- Understand the concept of array list (or vector)
 - How to increase its capacity
 - Discuss incremental and doubling strategies

Next Time

□ Doubly-linked list and iterators

□ HW:

- Go over the next lecture slides before the class
- Just 10 min ~ 20 min should be okay

Any Questions?

- Come up with one question on what we have discussed in the class and submit at the end of the class
 - 1 for typical questions
 - 2 for questions with thoughts or that surprised me

- Write questions at least 4 times

- You can type at KLMS