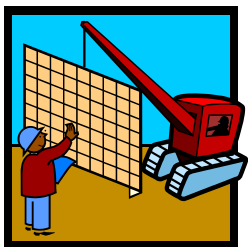
Lecture 3-1. Array Lists

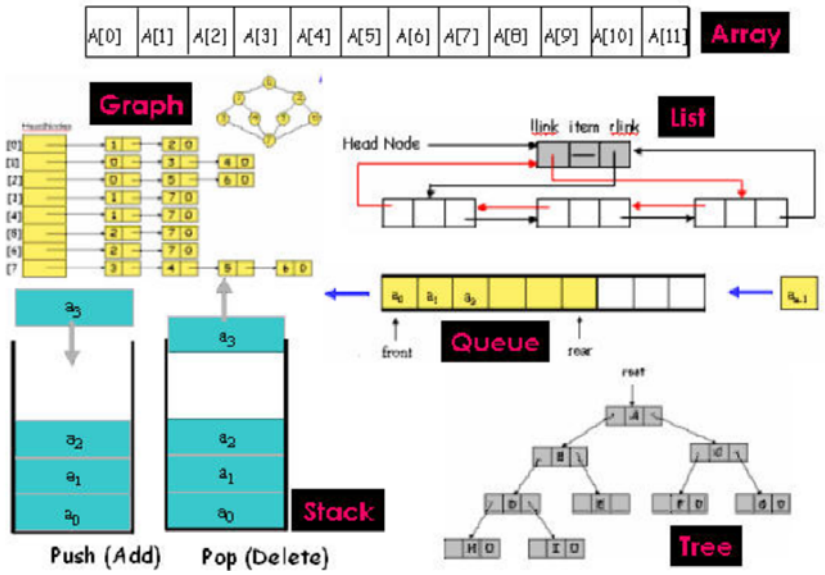


Sunghyun Cho
Professor
Division of Computer Science
chopro@hanyang.ac.kr

HANYANG UNIVERSITY

Keywords

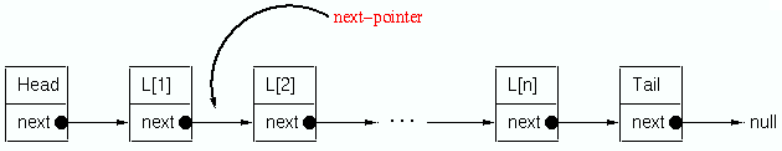
- Big Picture of Data Structures



Keywords

- List
  - ADT that implements an ordered collection of values
  - Is a fundamental data structure and basis of other data structures like stack and queue

- Linked List
  - A representative type of list
  - Node, Pointer (Link), Head, Tail



The Array List ADT

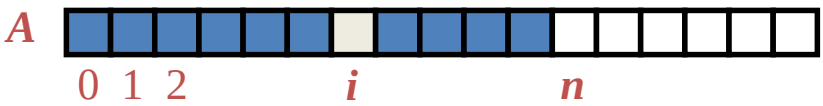
- The Array List ADT extends t he notion of array by storing a sequence of arbitrary objects
- An element can be accessed, inserted or removed by specifying its index (number of elements preceding it)
- An exception is thrown if an incorrect index is given (e.g., a negative index)
- Main methods:
  - get(integer i): returns the element at index i without removing it
  - set(integer i, object o): replace the element at index i with o and return the old element
  - add(integer i, object o): insert a new element o to have index i
  - remove(integer i): removes and returns the element at index i
- Additional methods:
  - size()
  - 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 A of size N
- A variable n keeps track of the size of the array list (number of elements stored)
- Operation *get*(i) is implemented in ( ) time by returning $A[i]$
- Operation *set*(i,o) is implemented in ( ) time by performing $t = A[i]$, $A[i] = o$, and returning t.



Example: Storing Game Entries

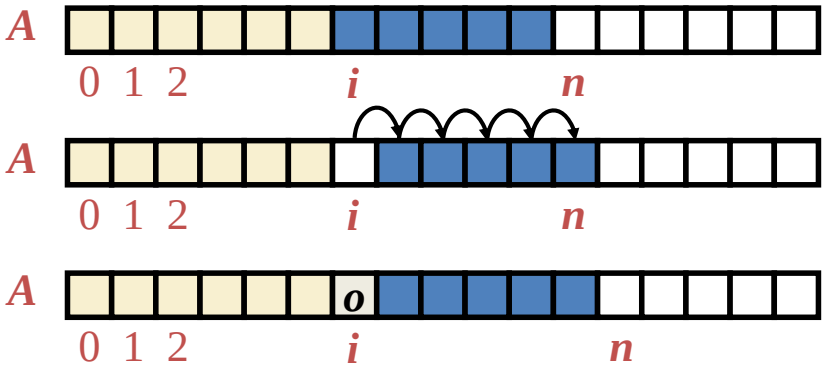
- Array List
  - storing entries in an array, in particular, high score entries for a video game
  - refer to the Code Fragment 3.1, 3.2 (textbook p.94, p.95)
  - two important method
    - Insertion
    - Remove

Insertion

- **add(*e*)**
  - Insert game entry *e* into the collection of high scores. If the collection is full, then *e* is added only if its score is higher than the lowest score in the set, and in this case, *e* replaces the entry with the lowest score.
  - Java code for inserting a GameEntry (Code Fragment 3.3, textbook p.96)

Insertion

- In operation ***add*(*i*, *o*)**, we need to make room for the new element by shifting forward the $n - i$ elements $A[i], \ldots, A[n-1]$
- In the worst case ($i = 0$), this takes ( ) time

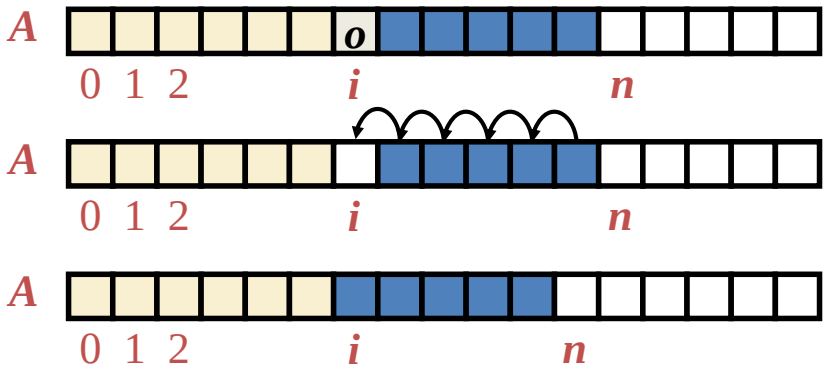


Element Removal

- **remove(*i*)**
  - Remove and return the game entry *e* at index *i* in the entries array. If index *i* is outside the bounds of the entries array, then this method throws an exception; otherwise, the entries array will be updated to remove the object at index *i* and all objects previously stored at indices higher than *i* are "moved over" to fill in for the removed object.
  - Java code for removing a GameEntry (Code Fragment 3.4, textbook p.99)

Element Removal

- In operation ***remove***(i), we need to fill the hole left by the removed element by shifting backward the $n - i - 1$ elements $A[i + 1], \ldots, A[n - 1]$
- In the worst case ($i = 0$), this takes ( ) time



HANYANG UNIVERSITY

Performance

- In the array based implementation of an array list:
  - The space used by the data structure is ( )
  - ***size***, ***isEmpty***, ***get*** and ***set*** run in ( ) time
  - ***add*** and ***remove*** run in ( ) time in worst case
- If we use the array in a circular fashion, operations ***add***$(0, x)$ and ***remove***$(0, x)$ run in ( ) time
- In an ***add*** operation, when the array is full, instead of throwing an exception, we can replace the array with a larger one

HANYANG UNIVERSITY

Q & A

한양대학교 ERICA 캠퍼스