Arrays , Link Lists, Stacks and Queues

Deliverables

- Arrays
- Link Lists
- Stacks
- Queues

Copyright@ gdeepak.com

Arrays

Arrays are stored in contiguous memory locations and contain similar data. Position of any array element can be accessed from its index position with help of array's starting address.

An element can be accessed, inserted or removed by specifying its position (number of elements preceding it)

Vector is abstraction of array data structure, where we can access by rank.

Copyright@ gdeepak.com

Array operations

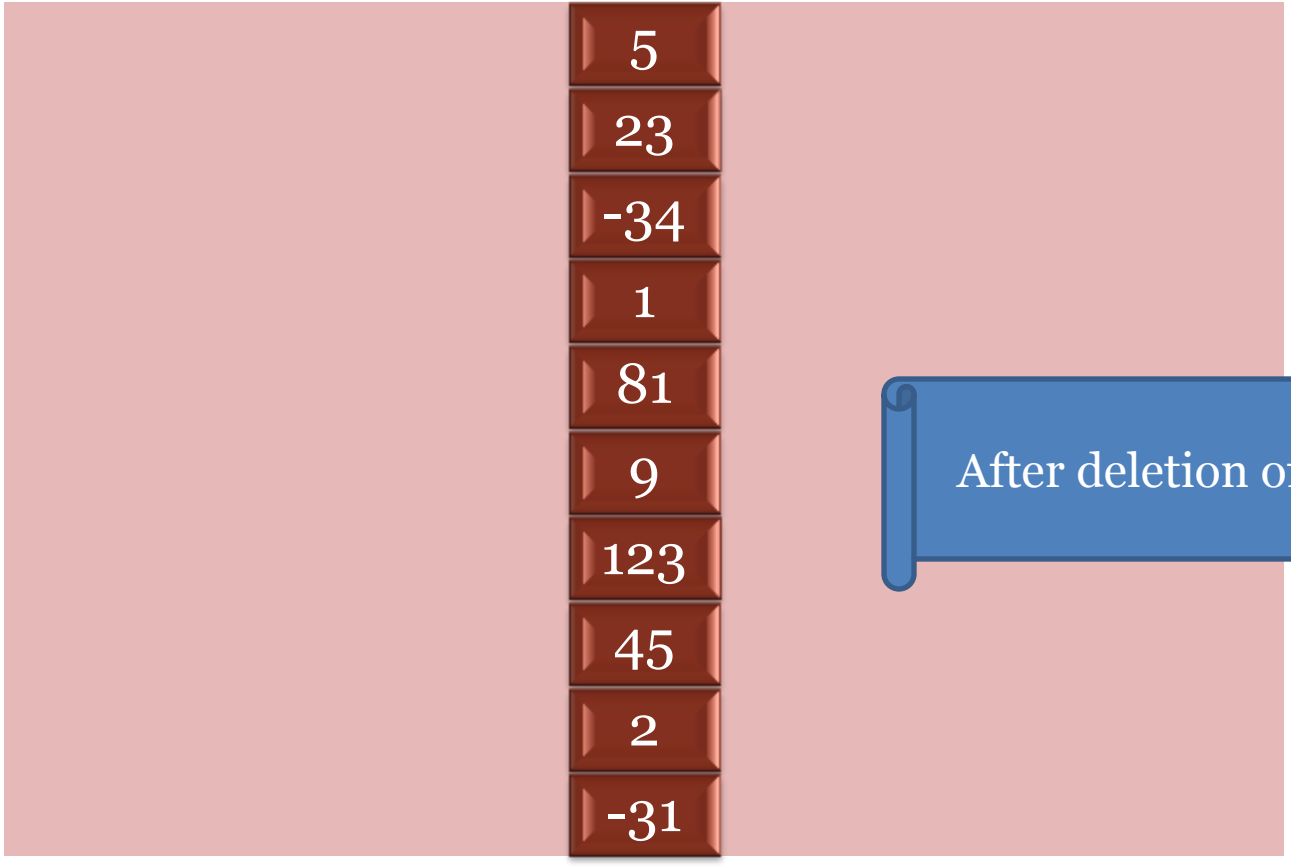
- Deletion: In operation removeAtPosition(r), we need to fill the hole left by the removed element by shifting backward n-r-1 elements A[r + 1], …, A[n - 1]. In the worst case (r = 0), this takes O(n) time. In the average case also it is O(n).
- A[0] A[1] A[2] A[3] A[4] A[5] A[6] A[7] A[8] A[9]

3	10	0	-135	2	21	41	6	87	11

- Is it necessary to shift the array elements after deletion?

Copyright@ gdeepak.com

Deletion of -34



Copyright@ gdeepak.com

Insertion

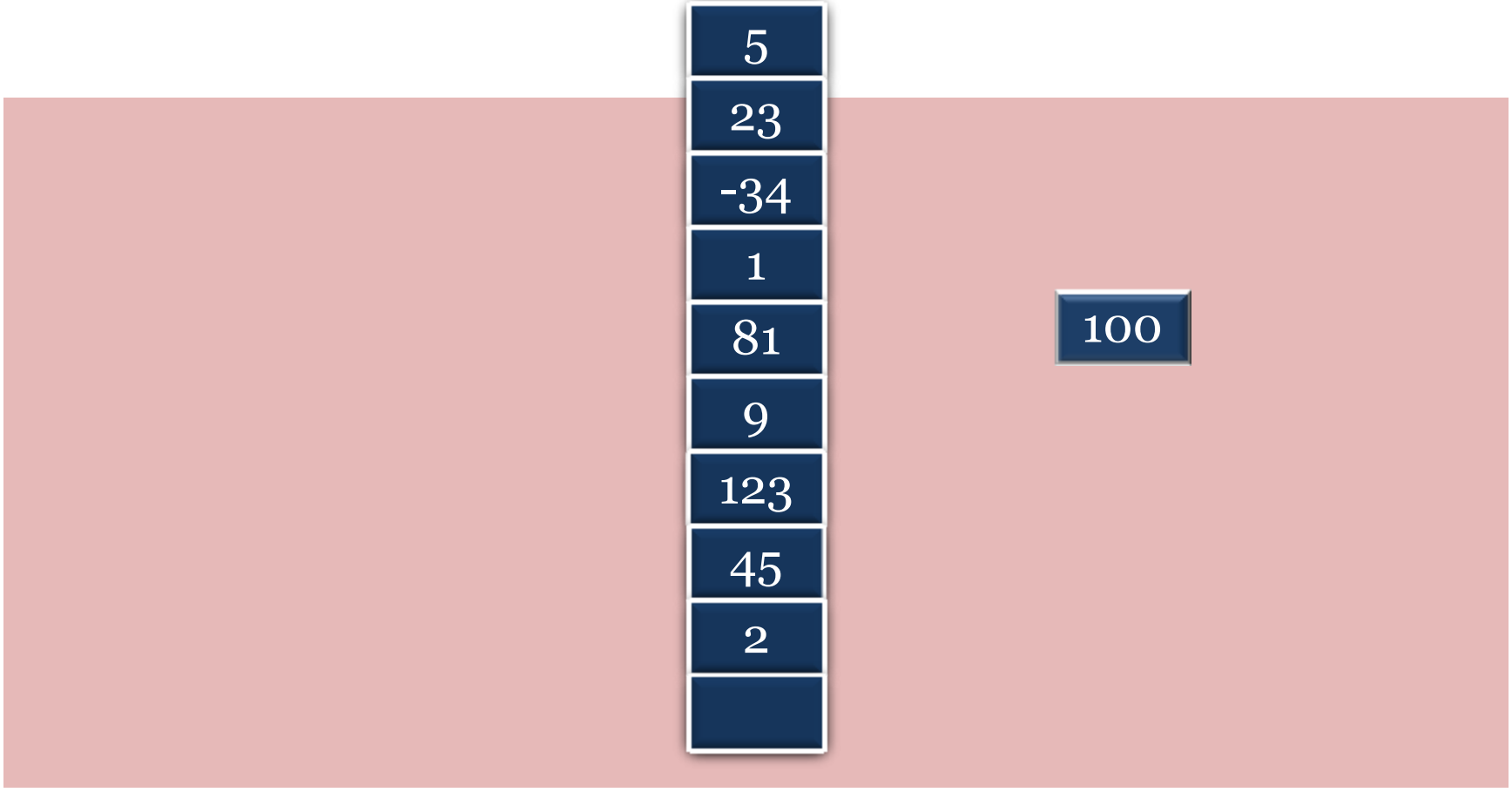
Insertion in the array again requires shifting all the elements one index ahead to make room for insertion.

Need to understand shifting different in deletion and insertion, otherwise data may be lost.

In Insertion shifting is to start from the last element of the array and in deletion shifting is to start from the next array index from where the element is to be deleted.

Copyright@ gdeepak.com

Inserting 100 after -34



Copyright@ gdeepak.com

Array complexity

- Accessing an array: O(1) time
- Copying an array: O(n) time
- Merging two arrays: O(m+n) time
- Splitting an array: O(n) time
- Intersection of two arrays: $O(n^2)$
- Union of two arrays: $O(n^2)$

Copyright@ gdeepak.com

Array incrementing by C or doubling

Let n be the number of elements in the array. The space used is $O(n)$ and each operation runs in time $O(1)$

The maximum size of the array must be defined apriori and cannot be changed.

Incremental strategy: increase the size by a constant C

Doubling strategy: double the size

Copyright@ gdeepak.com

Array incrementing by C or Doubling

If increment by C every time then

Creation copying insertion (iteration wise)

ci c(i-1) c

Total = ci+ci-c+c=2ci

If there are n elements in end then total iterations will be n/c , So it is 2c(1+2+3+.....n/c) =

$$\frac{2c(\frac{n}{c})(\frac{n}{c}+1)}{2} = \frac{n^2}{c} = O(n^2)$$

Copyright@ gdeepak.com

Array incrementing by C or Doubling

If we double the array every time then the creation, copying, insertion cost will be as follows

Creation	copying	insertion (iteration wise)
2^i	2^{i-1}	2^{i-1}

Total cost in every iteration is $2^i+(2^{i-1}+2^{i-1}) =2^i+2^i=2^{i+1}$

if there are n elements to be inserted then there will be lgn iterations , total becomes $(2^1 +2^2+2^3+...+ 2^{lgn+1})=4n-1=O(n)$

Copyright@ gdeepak.com

Link List

Singly linked list is a data structure consisting of a sequence of nodes

Each node stores element and link to the next node

There may be a header link and trailer link

Copyright@ gdeepak.com

Link list operations

Access (): Accessing an element from a list

Worst case: O (n)

Average Case: 1+2+3+...n = n(n-1)/2 divided by n=n-1/2 = O(n)

Delete (): O(1) complexity, but it is preceded by accessing that position, effectively taking O(n) time. Delete first takes O(1) time

Copyright@ gdeepak.com

Link list complexity

Insert : O(1) but inserting before or after an element will require accessing element effectively taking O(n) time. Inserting at first place is O(1).

Merging : O(1) If the trailer node information is not available then it will be equal to the length of the smaller list to reach at the end of the list for merging with the second list

Size: O(n) time

Intersection: $O(n^2)$

Union: $O(n^2)$

Copyright@ gdeepak.com

Doubly link list

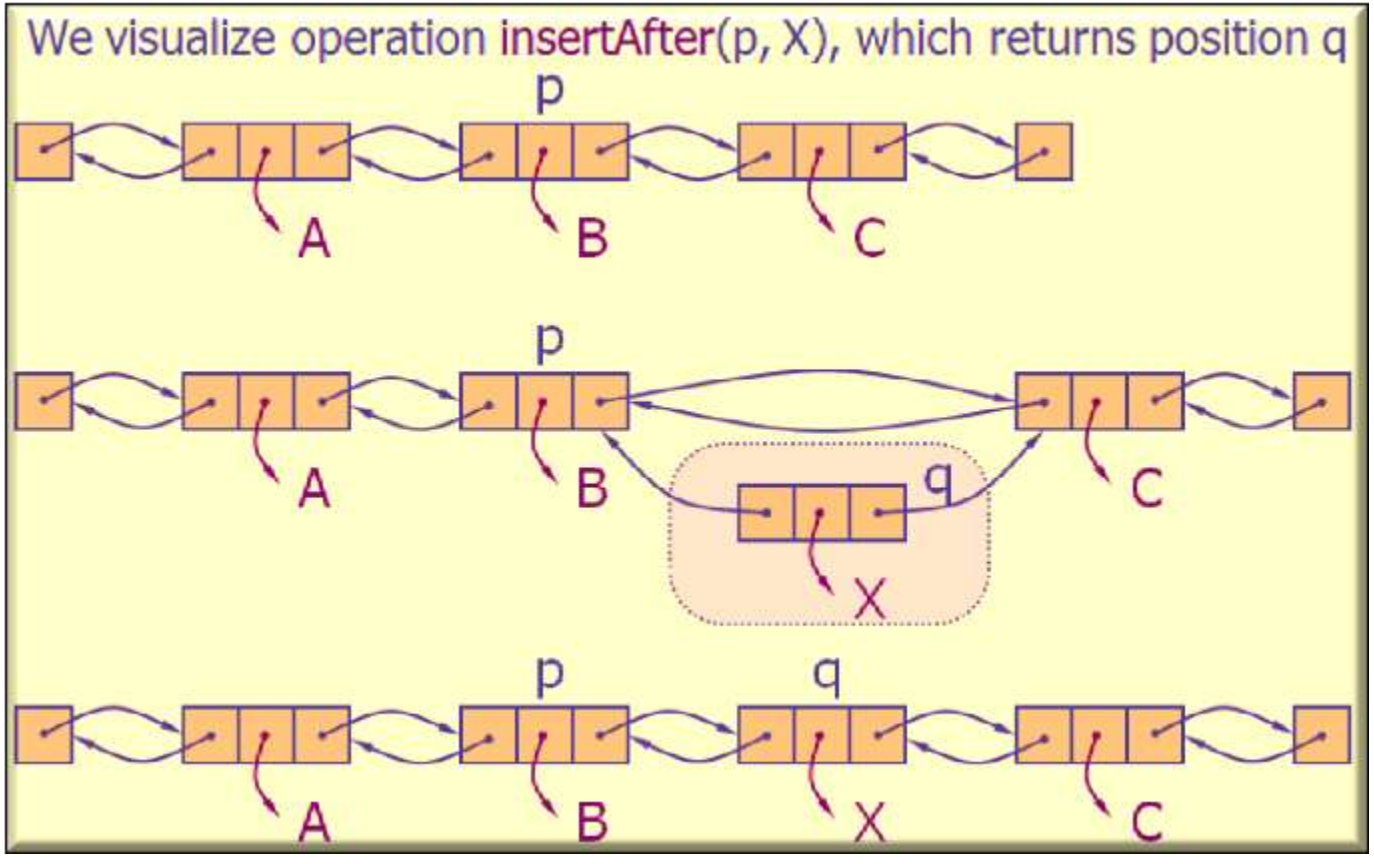
A doubly linked list provides a natural implementation of the List

Nodes implement Position and stores element, link to the previous node and link to the next node

Special trailer and header nodes. Deletion an Insertion becomes easy as there is no need to remember the address of the previous node.

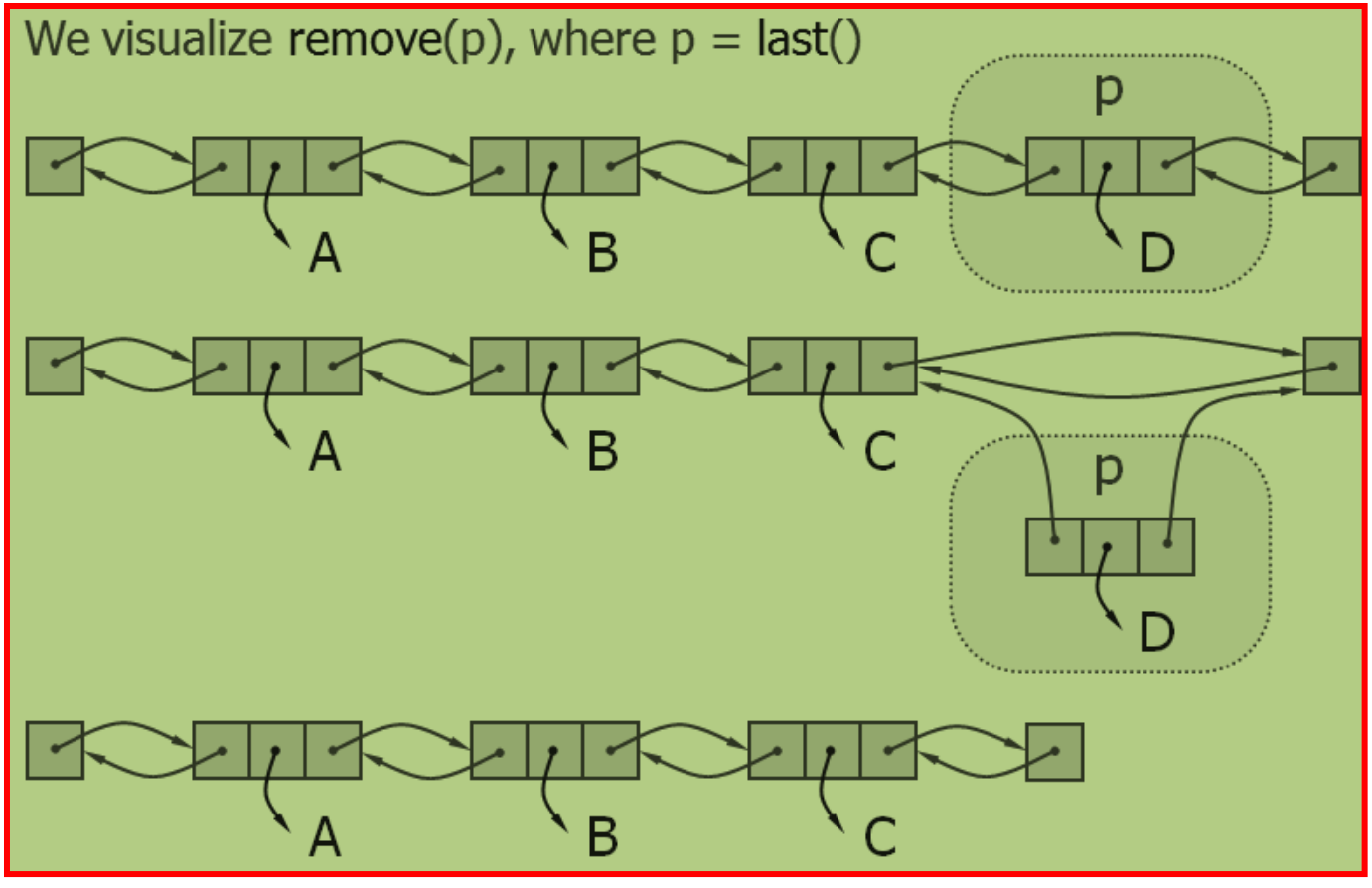
Copyright@ gdeepak.com

Insertion



Copyright@ gdeepak.com

Deletion



Copyright@ gdeepak.com

Circular link list

There will be no difference in the time complexity of various operations

Circular Link List and circular doubly link list

Input output buffers use circular queue for buffering.

Copyright@ gdeepak.com

Comparison link list vs. array

It is easier to delete in the link list

it is easier to insert in the link list

It is easier to access in the array

We have to predefine the array so there can be wastage of memory

Copyright@ gdeepak.com

Comparison link list vs. array

Array Size has to be defined and is limited to the defined size while link list can grow to limit of memory

Continuous allocation is required for array, while this is not with link list

Arrays compute address of data elements by arithmetic operations while in linked list addresses are in structure itself.

Arrays can be accessed backward and forward but we have to use special link lists like doubly linked list for backward access

Copyright@ gdeepak.com

Comparison link list vs. array

Arrays definition is part of the language construct but link list we have to create

Merging two arrays is very difficult while merging link lists is easy

There will be more cache misses in the Link list then in the arrays

With large records, moving pointers is easier and faster than moving the items themselves.

Copyright@ gdeepak.com

Comparison link list vs. array

There can be an array of pointers associated with a link list, in that case address part of the link list is not required.

In link lists two different processors may work on two different parts of link list.

Due to address part there is wastage of memory in link list, it is not much if we look at practical size of record of name, age etc.

Copyright@ gdeepak.com

Stack

- Last in First out, in and out only from one end, other end is closed
- Page-visited history in a Web browser
- Undo sequence in a text editor
- Chain of method calls in the Java Virtual Machine, Function Calls
- Recursion
- In other data Structures

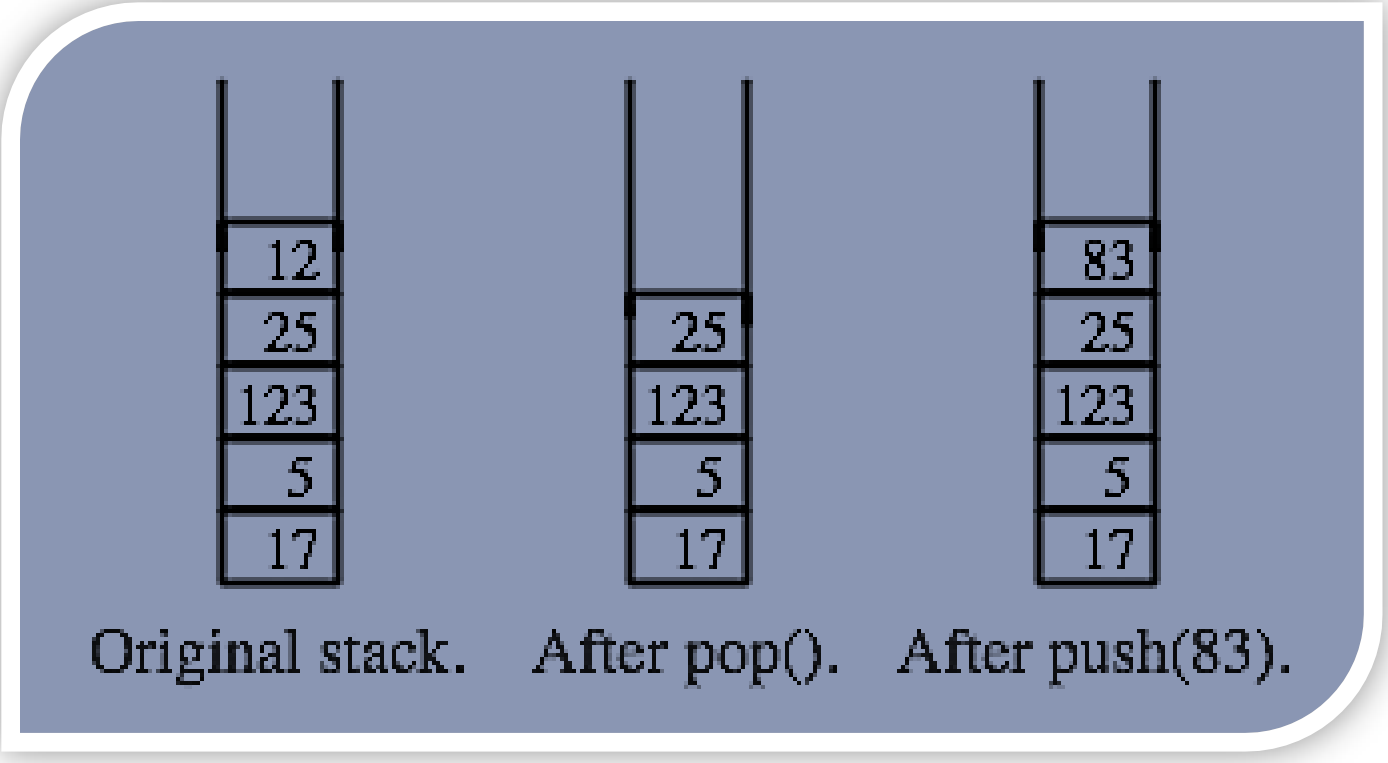
Copyright@ gdeepak.com

Simulating Pop and Push

Apply Operations
1. Pop
2. Pop
3. Push 37

Copyright@ gdeepak.com

Stack Example



Copyright@ gdeepak.com

Stack

- Creation of an stack: O(n)
- push(object): inserts an element O(1)
- pop(): removes and returns the last inserted element. O(1)
- top(): returns last inserted element without removing O(1)
- size(): returns number of elements stored. O(1)
- isEmpty(): indicates whether no elements are stored. O(1)
- isFull(): Indicates whether array limit is over. O(1)

Copyright@ gdeepak.com

Link List Vs. Array Implementation

Pop
Array: return element at “top” and decrement “top”
Linked list: return and remove at front of linked list

Push
Array: increment “top” and insert element. Must check for overflow!
Linked list: insert element at front of linked list

Array: more memory efficient
Linked list: don’t have to worry about “overflow”

Copyright@ gdeepak.com

Axioms

pop(push(S, v))=S

Top(push(S, v))=v

Top(pop(push(push(S,12),15))) = top(push(S,12))

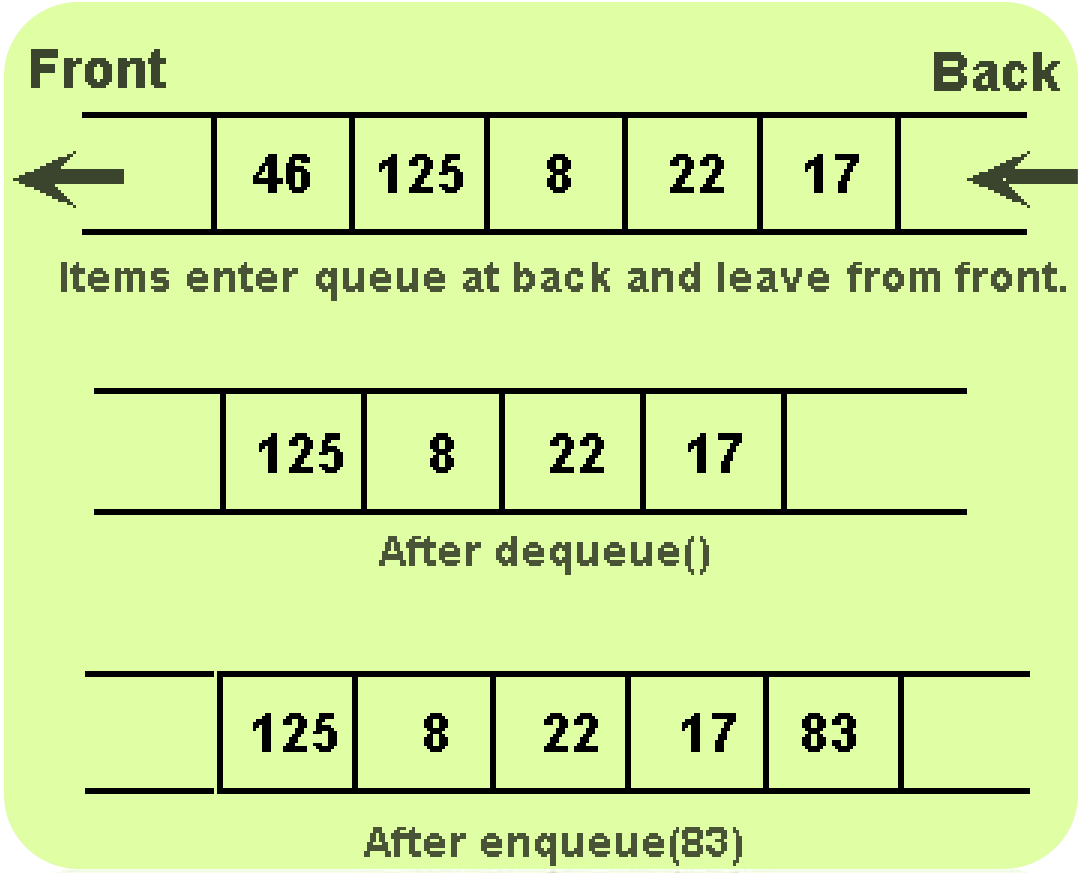
Copyright@ gdeepak.com

Queues

- Insertions and deletions follow the first-in first-out scheme
- Insertions are at the rear of the queue and removals are at the front of the queue
- Two variables keep track of the front and rear. Careful implementation is necessary to avoid overlapping.
- Operations complexity may vary depending upon the implementation, whether front or rear is fixed or not

Copyright@ gdeepak.com

Queue Example



Copyright@ gdeepak.com

Simulating Enqueue and Dequeue

Perform Operations
Enqueue 99
Dequeue
Dequeue

Copyright@ gdeepak.com

Queues

enque(Object o): inserts an element o at end of queue. O(1)

deque(): removes and returns front element of queue. O(1)

front(): returns element at front without removing it. O(1)

size(): returns number of elements stored. O(1)

isEmpty():returns Boolean indicating queue is empty or not O(1)

Copyright@ gdeepak.com

Queue Applications

- Waiting lists, bureaucracy
- Access to shared resources (e.g. printer)
- Multiprogramming
- Tunnel
- In other Data Structures

Copyright@ gdeepak.com

Axioms

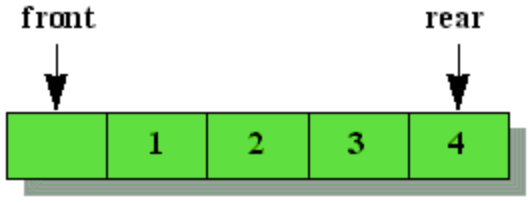
Front(Enqueue(Enqueue(Q, w),v))=front(Enqueue(Q, w))

Deque(Enque(Enque(Q, w),v))=Enque(Deque(Enque(Q, w)),v)

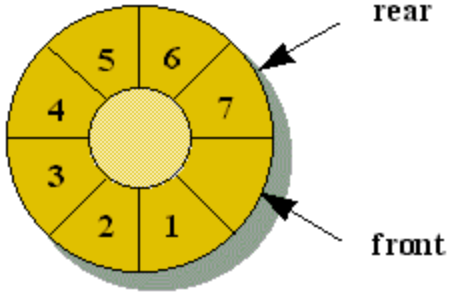
In an Enqueue operation, when the array is full, instead of throwing an exception, we can replace the array with a larger one, Similar to what we did for an array based stack.

Copyright@ gdeepak.com

Circular queue



linear queue



circular queue

Circular Queue is essentially implemented as a linear array in a wrapped around fashion where even rear pointer of the queue can come before the front pointer

Copyright@ gdeepak.com

Questions, Comments and Suggestions

Copyright@ gdeepak.com

Question 1

What is result of Last operation in given sequence :
Push(&);Push($);Push(@);Top();Pop();Push(#);
Pop();Pop();Push(%);Pop();

Copyright@ gdeepak.com

Question 2

What will be the result of last result of given priority queue if number 1 is the highest priority

Enqueue(3), Enqueue(6), Enqueue(2),dequeue, dequeue, Enqueue(5),Enqueue(1),dequeue, dequeue

Copyright@ gdeepak.com

Question 3

In a Singly circular linked list organization, insertion of a record involves modification of

A. One pointer

B. Two pointers

C. Three pointers

D. Four pointer

E. No Pointer

Copyright@ gdeepak.com