Stacks

A stack is an ADT where the insertions and deletions of elements follow the LIFO (Last-in-first-out) principle.

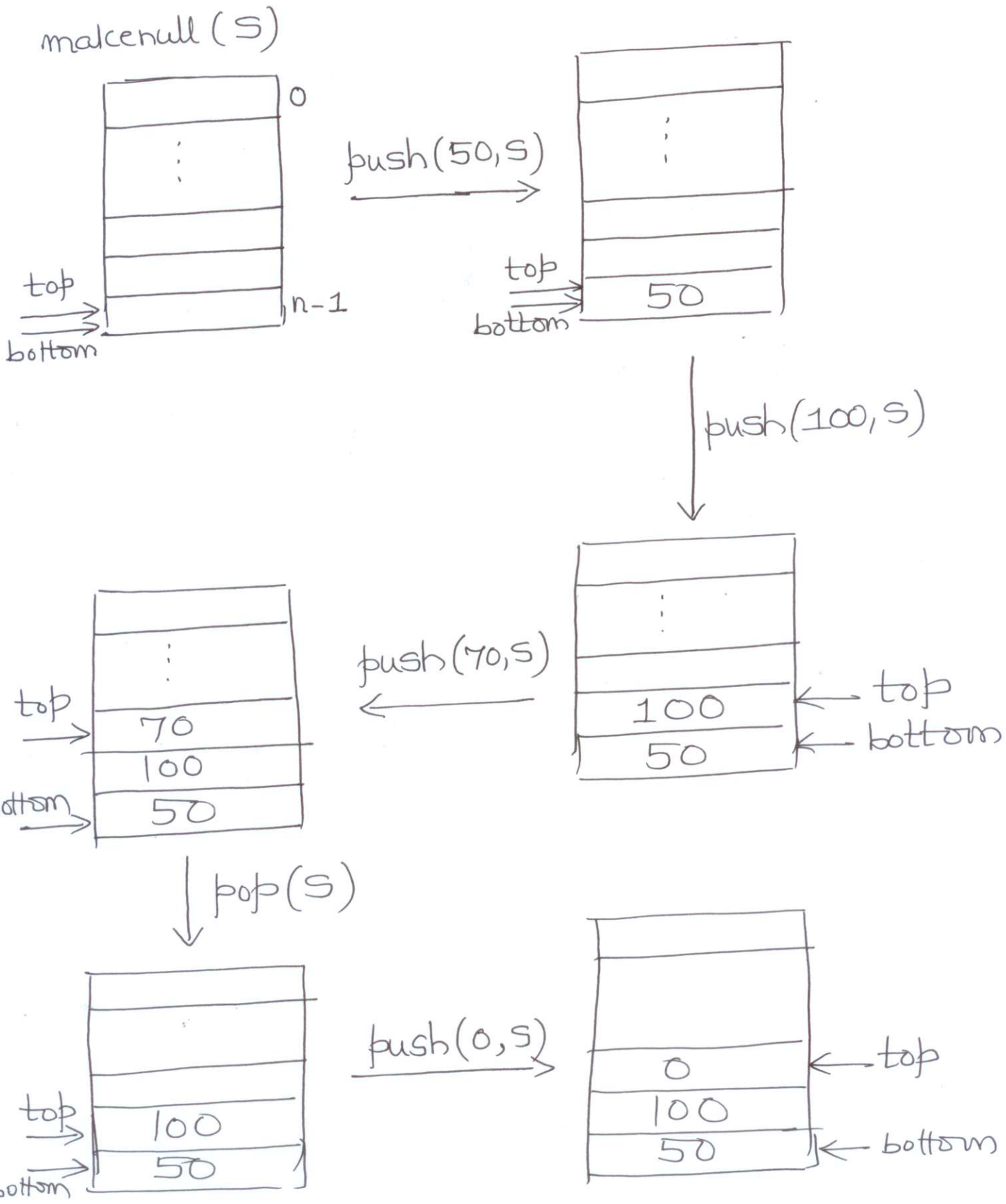
It is a collection of elements with a designated position called "top of the stack" where insertions and deletions occur.

The generic operations on a stack S containing elements x include:

- makenull (S) – create an empty stack
- empty (S) – check if stack S is empty
- top (S) – retrieve element on top of stack
- push (x, S) – insert x on top of the stack
- pop (S) – delete element on top of the stack

Stacks are extremely useful in computer science
- Implementing recursion
- Implementing nested procedure calls
- tree traversals, graph traversals,
- etc.

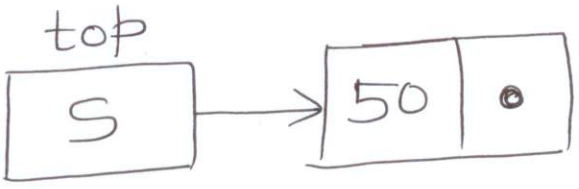
Stack Implementation using Arrays explained through an Example



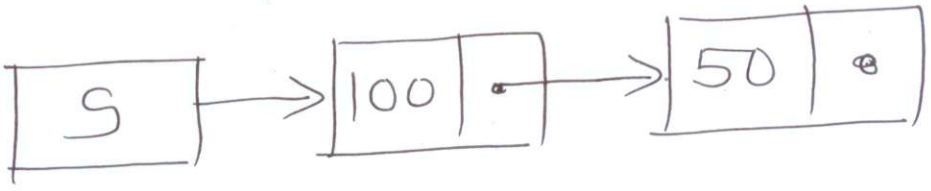
Stack Implementation using Linked Lists Explained through an Example

makenull(S)

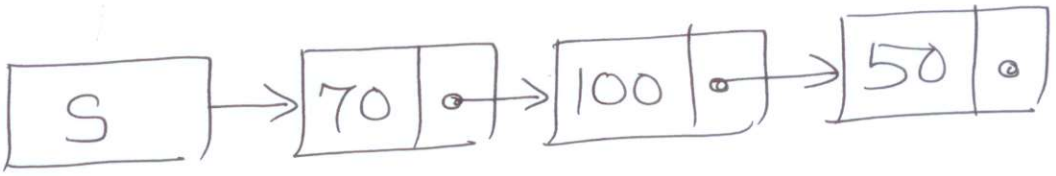
push(50,S)



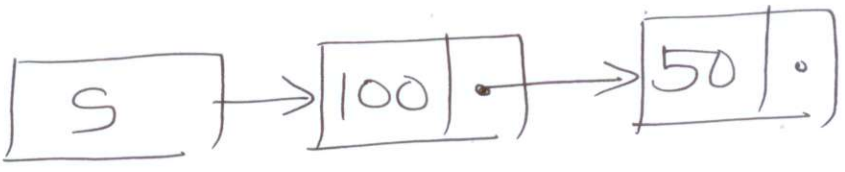
push(100,S)



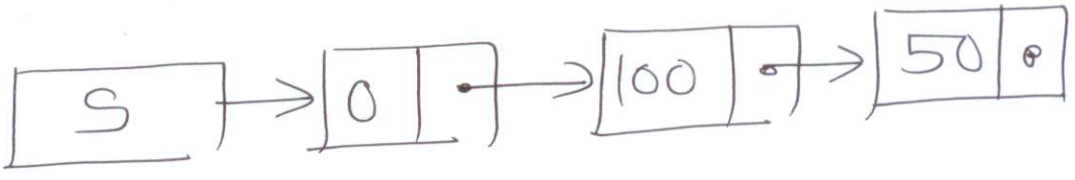
push(70,S)



pop(S)



push(0,S)



The computational complexity of both push and pop operations using both arrays and linked lists is

O(1) (constant time)

Queues

A queue is a collection of elements where the insertions and deletions follow the FIFO (first-in-first-out) principle.

A queue has a head and a tail. Insertions happen at the tail and deletions happen at the head.

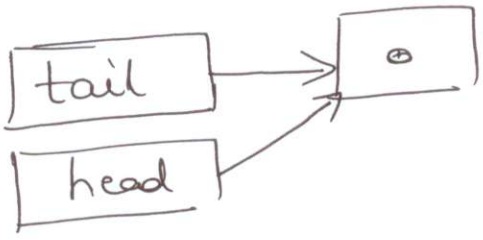
Generic operations on a queue Q containing elements x include:

- makenull(Q) — create an empty queue
- empty(Q) — check if queue Q is empty
- front(Q) — retrieve the element at the head of the queue
- insert(x, Q) — insert x at the tail of the queue
- delete(Q) — delete the element at the head of the queue

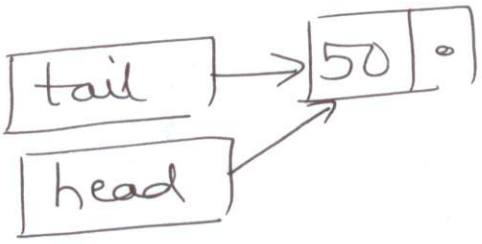
Very useful in simulation, graph algorithms.

Arrays and linked lists can both be used. A linked list implementation is shown below.

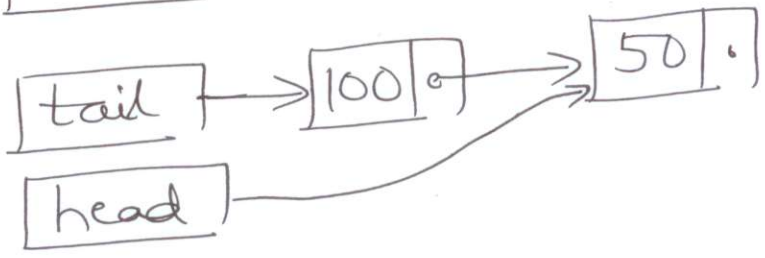
makenull(Q)



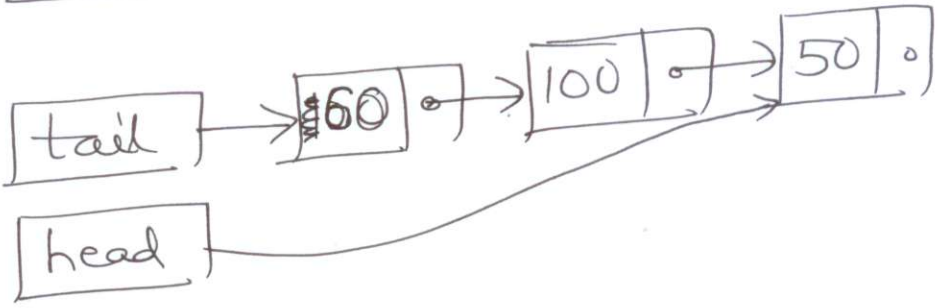
enqueue(50, Q)



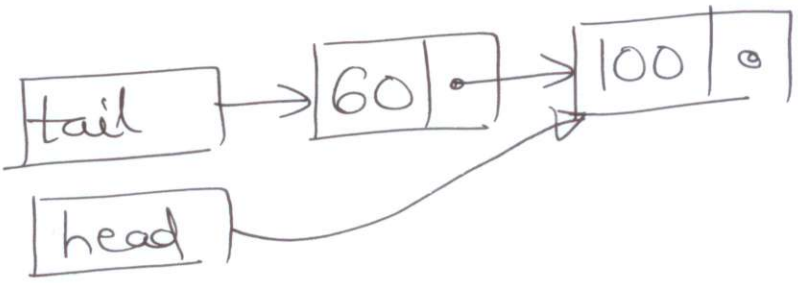
enqueue(100, Q)



enqueue(60, Q)



dequeue(Q)



Circular array, circular linked list are popular for implementing queues. Doubly linked lists can also be used.

Doubly Linked List

This is a linked list where every node has a pointer to its next node as well as to its previous node.

Makes traversals more efficient but at the cost of increased space consumption.

Example: