- Subhesh
- DTV 2021
- 5* on codechef
- teaching 3 year.
- PepCoding got acquired by Scaler

Agenda : Stacks and Queues

Game: win a pizza

A B → whosoever reaches 100 will win the game.

{1, 2, ... 10}

Subha	Prathmesh
1	10
12	20
23	30
34	40
45	50
56	60
67	70
78	80
89	94
100	

Subha	Arjun
1 (1, 10)	2 (2, 11)
12	13
23	24
34	35
45	46
56	61
67	77
78	89
89 (90, 99)	
100	

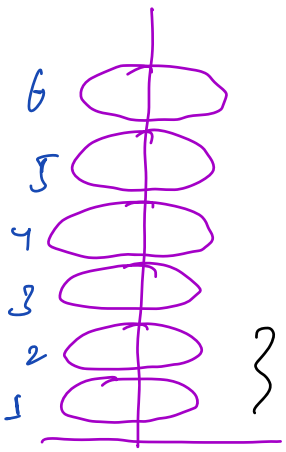
89 90 91 92 93 94 95 96 97 98 99 (me) 100

(1, 10)

Game theory

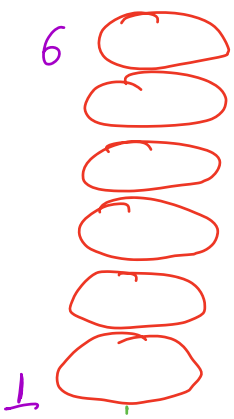
Stacks: An abstract data structure.

↳ Idly cooker?



→ remove at top
→ add at top

↳ Serving Plates in a marriage:



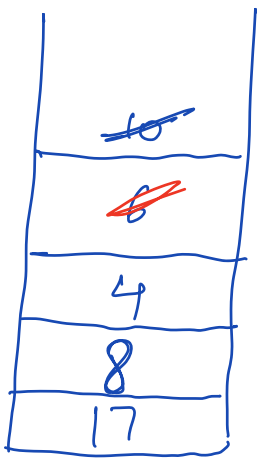
→ Stacks
↳ Came at very first is at the last.
↳ Came at very last is at the top.

↓

Came last → out first [LIFO]
Came first → go last [FILO]

Ex: 17 8 4 6 10 remove() remove()

10 6



↳ data structure that follows LIFO or FILO technique is called Stack.

↳ give me some real life example of Stack:

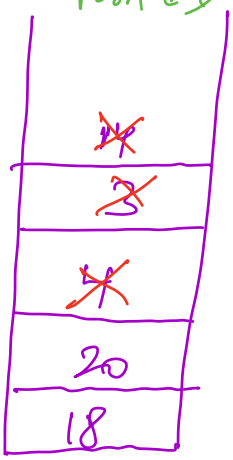
- ↳ Chair Stack
- ↳ recursion
- ↳ webPage history
- ↳ undo - redo
- ↳ ladies bangles

Operations:

- ↳ Push(x): Add x at the top
- ↳ pop(): remove the topmost element
- ↳ top(): return the topmost element

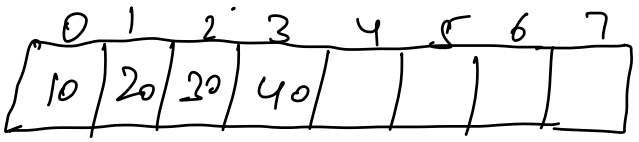
↳ size() : Total element
↳ isEmpty() : Is Stack empty or not?

Ex: Push(18) Push(20) Push(4) PoP() topP() Push(3) PoP()
Push(5) PoP() topP()
20 20



* Implementation: → Arrays
→ Dynamic Array
→ LinkedList

① Stack using arrays:



int[] arr :
int top = -1 :

```
Push (x) {
if (top == arr.length-1){
    //overflow
}
top = top + 1;
arr[top] = x;
}
```

```
PoP() {
if (isEmpty()) {
    //underflow
}
top--;
}
```

```
topP() {
if (isEmpty()) {
    //underflow
}
return arr[top];
}
```

```
size() {
    return top+1;
}
```

```
isEmpty() {
    if (size()==0) {
        return true
    }
    return false
}
```

(11) Stack using list :

↳ quick revision of list :

insert() → insert at the end of list
delete() → remove from the end
size() → length of the list

↳ `List<int> st;`

Push(10)

st.insert(10) → | 10 |

Push(20)

st.insert(20) → | 10 | 20 |

Push(30)

St. insert(30) → 

pop() → Check for underflow

St. remove() 

top()

St[St.size()-1] → 20

size()

St. size() → 2

(iii) Stacks using LinkedList :

addLast() → O(1)

addFirst() → O(1)

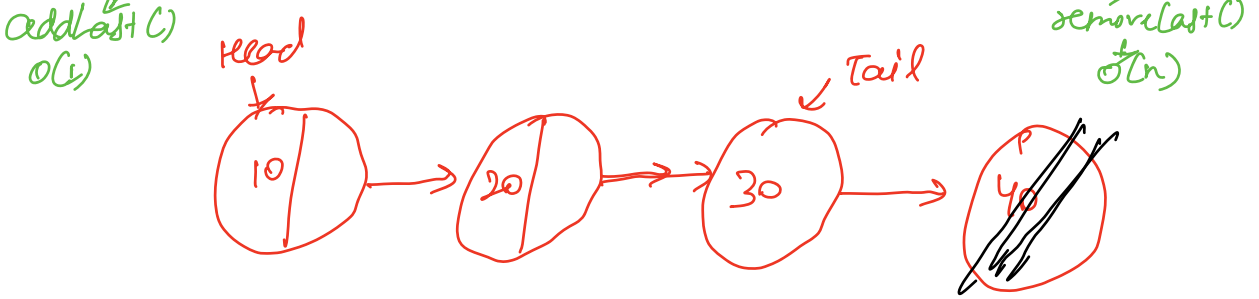
removeLast() → O(n)

removeFirst() → O(1)

Push(10) Push(20) Push(30) Push(40) Pop() Pop()

addLast() O(1)

removeLast() O(n)



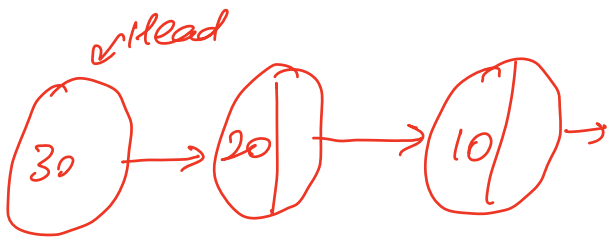
Note: We cannot use addLast() and Remove Last() while adapting LinkedList as a Stack.

↳ addFirst and removeFirst will do.

Push(10) Push(20) Push(30) Push(40) Pop() Pop()

addFirst() → O(1)

removeFirst() O(1)



↳ Every language has inbuilt library of Stack.

↳ Stack<Integer> St = new Stack<>();

datatype name

St.Push(10) St.Pop()

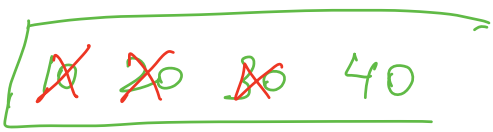
St.Push(20) St.top()

↳ Stack<int> St;

Queues? Covid vaccination centre!

FIFO / First Come First Serve

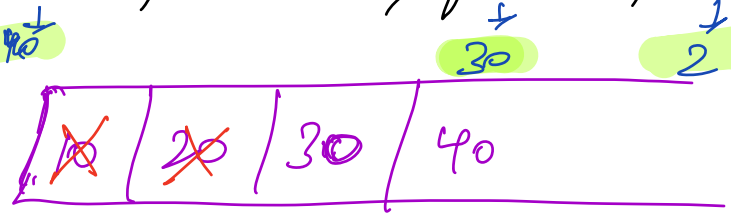
Ex: 10 20 30 40



Real life examples: (i) Scheduling Algorithm
(ii) Chat bot
(iii) Sales website

Operations:
- insert() / Enqueue(): add at last.
- delete() / Dequeue(): remove at first
- rear(): get the last inserted element
- front(): get the first inserted element
- size(): no. of elements

Ex: insert(10), insert(20), insert(30), insert(40), delete()
rear(), delete(), front(), size().



★ implementation ?

→ Array → H.W

↳ Dynamic array

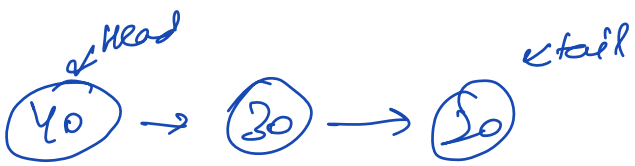
↳ LinkedList

① LinkedList as Stack

addLast()
removeLast()

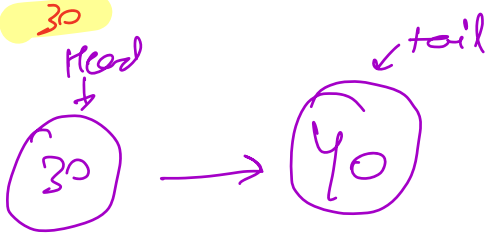
addFirst()
removeFirst()

② non-optimal approach : addFirst() & removeLast() O(n)



③ Optimal approach: addLast() O(1) & removeFirst() O(1)

insert(10), insert(20), insert(30), insert(40), delete()
rear(), delete(), front(), size().
40 30 1



Queue using 2 Stacks