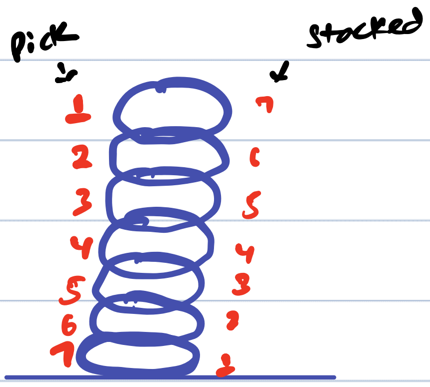
Today's agenda

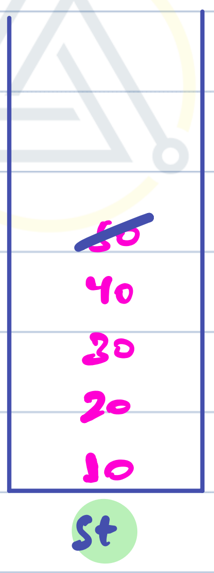
- Stacks
- Linkedlist as stack
- Remove adjacent duplicate
- Balanced Parentheses
- Min stack

Arrays
Hashmap
Linkedlist

//Stacks → Last in first out (LIFO)



→ Stack <Integer> st = new Stack<>();
name



O(1) ← st.push(10) ← add in stack
st.push(20)
st.push(30)
st.push(40)
st.push(50)

O(1) ← st.size() → 5 {no. of elements in stack}

Ex: (i) Piles of Plates/books
(ii) bangles in hand
(iii) undo/redo

O(1) ← st.pop() → 50 {remove and return topmost ele}

O(1) ← st.peek() → 40 {return the topmost element} initialize

Array

Hashmap

Stack

Queue

Priorityqueue

graph

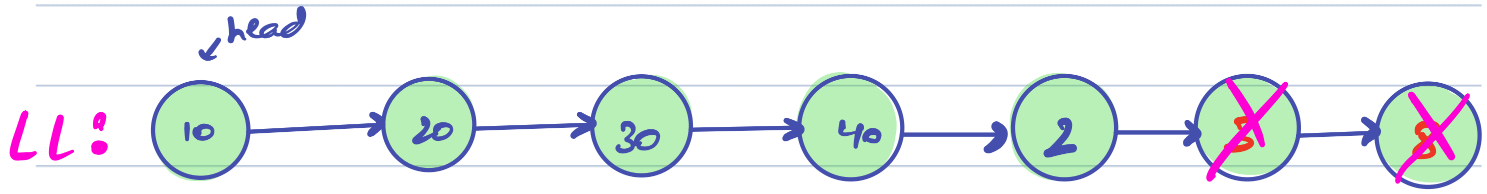
Created linkedlist

Linkedlist $\rightarrow$ Head

Tree

Adapters

//Linkedlist as Stack → Last in first out

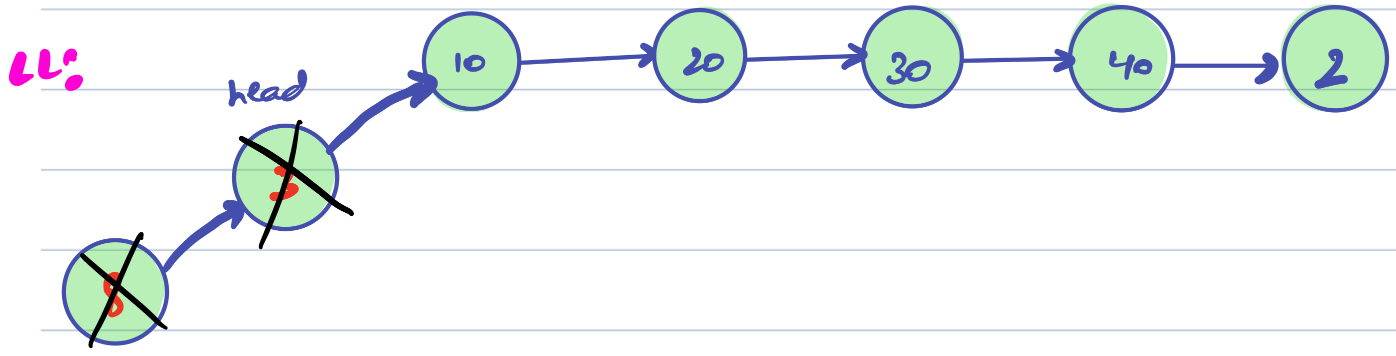


3 8 Pop() Pop()

↓ 8 ↓ 3

(i) addLast → O(N)

(ii) removeLast → O(N)



3 8 Pop() Pop()

↓ 8 ↓ 3

(i) addFirst → O(1)

(ii) remove first → O(1)

Q) Remove adjacent duplicate

↳ Given a String S, Remove equal Pair of adjacent characters. Return the final String.

Ex1: a ~~b b b b~~ d → ad

Ex2: a ~~b c c b~~ d e → ade

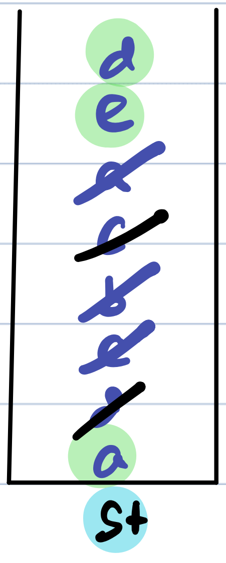
Ex3: a ~~b b~~ b e → abe

Ex4: a d c ~~b b~~ e ~~c a a c~~ d e d
↳ adceded

Ex5: a ~~b b b b~~ d a → ada

i

$$\begin{array}{cccccccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ \text{Ex:} & a & \cancel{d} & \cancel{c} & \cancel{b} & \cancel{b} & \cancel{c} & \cancel{c} & \cancel{a} & \cancel{a} & \cancel{c} & \cancel{d} & e & d \end{array}$$



↳ dea

↳ aed

//Psuedo code

```
String Removeadjacentelement (String s){

        Stack<Character> st = new Stack<>();

        for(int i=0; i<s.length(); i++){

            if(st.size()==0 || st.peek() != s.charAt(i)){
                    st.push(s.charAt(i));
            }
            else{
                    st.pop();
            }

        }

}
```

T.C: O(N)

S.C: O(N)

// H.W → generate ans. from stack.

Break till 10:30 Pm

Q) Valid Parentheses

↳ Given a String with '(' ')' '{' '}' '[' ']', you have to find whether the String is balanced or not.

Note: balanced Strings:

① open brackets must be Closed by the same type of bracket.

S: ( ) { } [ ] (( )) → true

S: ( ) { } [ ) → false

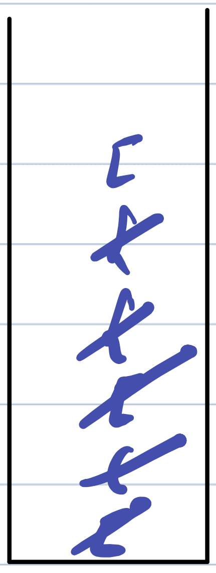
② opening brackets must be Closed in Correct order.

S: [{] } → false

S: ( ) { (}) → false

$$\begin{array}{cccccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ S: & [& (&) & [&] & \{ & \} &] & \{ & \} & [\end{array}$$

i



```
if (st.size() == 0){
    return true;
}
else{
    return false;
}
```

//Psuedo code

T.C: O(N)
S.C: O(N)

```
boolean validParentheses (String s){
    Stack <Character> st = new Stack<>();

    for (int i=0; i<s.length(); i++ ){

        if (st.size()==0|| s[i]== '(' || s[i]== '{' ||s[i]== '[' ){
            st.push (s[i]);
        }
        else {
            if(s[i] == ')'){
                if(st.peek() == '(') { st.pop();}
                else { return false; }
            }
            else if (s[i]= ']'){
                if(st.peek() == '[') { st.pop();}
                else { return false; }
            }
            else {
                if(st.peek() == '{') { st.pop();}
                else { return false; }
            }
        }
    }

    if (st. size()==0) { return true; }
    else { return false;}
}
```

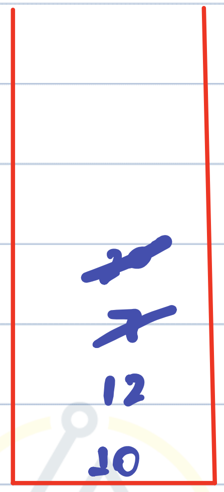
adapter

Q) Min Stack

↳ Normal Stack → PoP(), Push(val), Peek(), size() → O(1)

+ getMin() → min element of Stack

↳ T.C: O(1)



10 12 7 20 getmin() PoP()

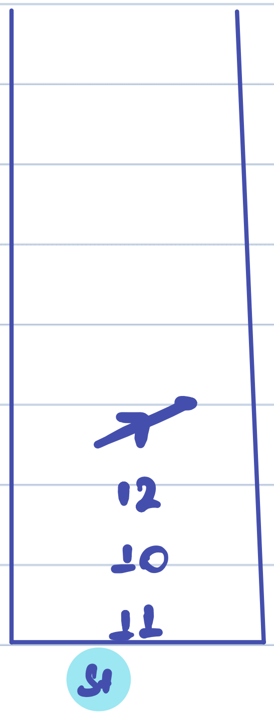
getmin() ↓ 7

PoP() ↓ 20

PoP() ↓ 7

getmin() ↓ 10

// wrong idea



11 10 12 7 getmin()

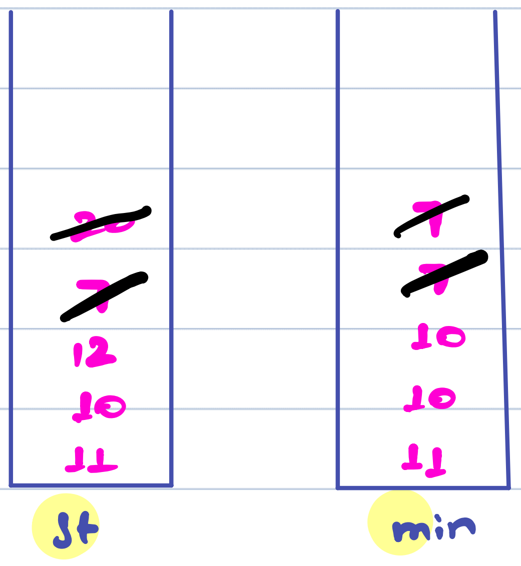
getmin() ↓ 7

PoP() ↓ 7

min: ~~ϕ~~ ~~11~~ ~~10~~ ~~7~~ 10

2nd min: ~~ϕ~~ 10

//Correct idea



11 10 12 7 20 getmin()
$\downarrow$
7

pop() $\downarrow$ 20 pop() $\downarrow$ 7 getmin() $\downarrow$ 10

T.C: O(N) S.C: O(N)
$\downarrow$
possible to reduce to O(1)

	0	1	2	3	4	5
arr[]:	{10	12	7	1	8	15}
min[]:	10	10	7	1	1	1