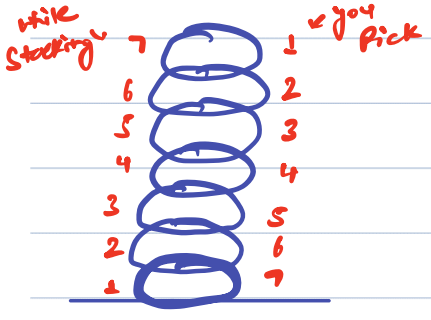
Today's agenda

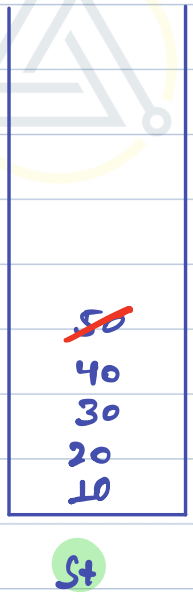
- Stacks
- LinkedList as Stack
- Remove adjacent duplicate
- Balanced Parentheses
- Min Stack

Stacks → Last in first out.



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

(type: Integer, name: st)



Operations:

O(1) ← `st.push(10);` → add in stack.

O(1) ← `st.pop();` → remove topmost no. and return.

O(1) ← `st.peek();` → returns you the topmost element.

O(1) ← `st.size();` → no. of elements in stack.

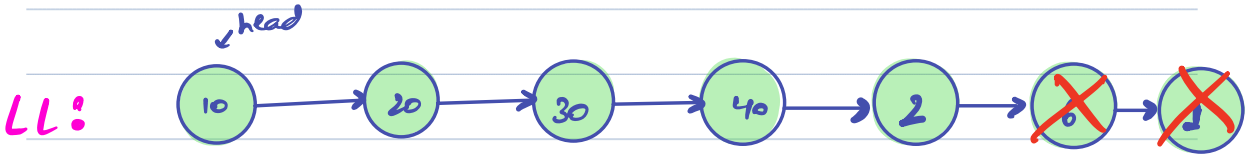
Ex: (i) Pile of Plates
(ii) bangles in hand
(iii) undo / redo

Array
Hashmap
Stack

LinkedList
Tree

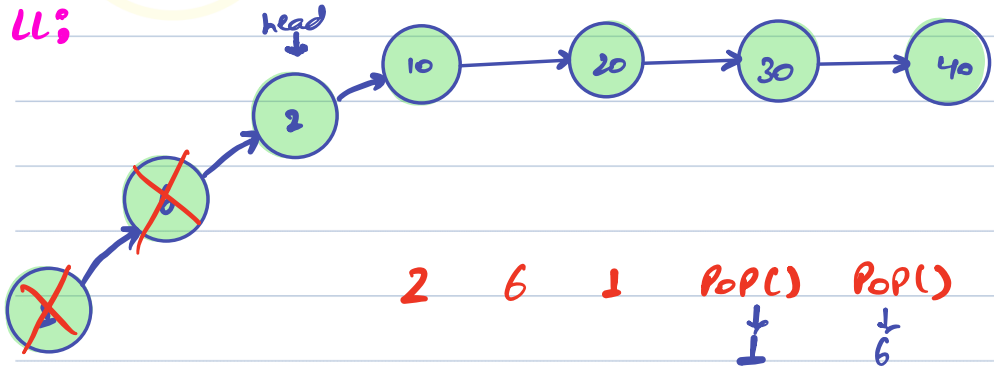
→ adapters

LinkedList as Stack → Last in first out



2 6 1 pop() pop()
↓ ↓
1 6

(i) addLast → O(N)
(ii) removeLast → O(N)



2 6 1 pop() pop()
↓ ↓
1 6

(i) add first → O(1)
(ii) Remove first → O(1)

8) Remove adjacent duplicate

↳ Given a string S, Remove equal pair of adjacent characters. Return the final string.

Ex1: a x x x x d → ad

Ex2: a x x x x d e → ade

Ex3: a x x b e → abe

Ex4: a d c x x e x x x d e d

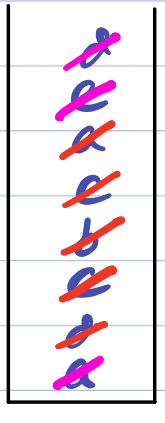
↳ adceded

Ex5: a x x x x d a

↳ ada

Ex: a d c b b c c a a c d e d

0 1 2 3 4 5 6 7 8 9 10 11 12



dea

↳ aed

IIPSuedo Code

```
String RemoveadjacentElement (String s){

    Stack < Characters > 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();
        }

    }

    Char [] arr = new char [st.size()];
    for (int i= arr.length-1; i>=0; i--){
        arr[i] = st.pop();
    }
    -> Char arr to String.

}
```

T.C: O(N)

S.C: O(N)

Break till 9:15Pm

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.

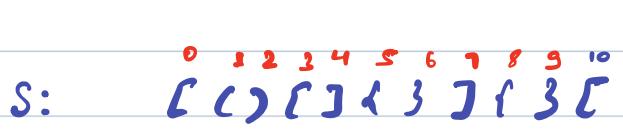
S: ( ) { } [ ] (( )) → true

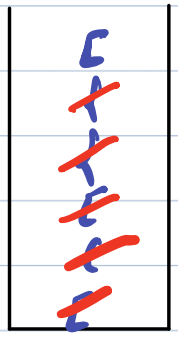
S: ( ) { } [ ) → false

② opening brackets must be closed in correct order.

S: [{] } → false

S: ( ) { (}) → false





//Pseudo code

```
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;}
}
```

T.C: O(N)

S.C: O(N)

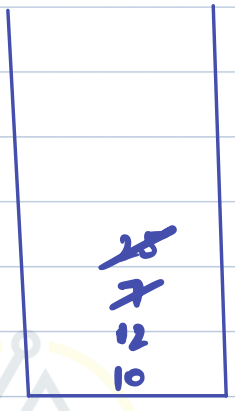
Q) Min Stack

↳ Normal Stack → pop(), push(val), peek(), size()

+

getMin() → min element of Stack

↳ expected T.C: O(1)



10 12 7 25 getMin() pop()

↓ ↓

7 25

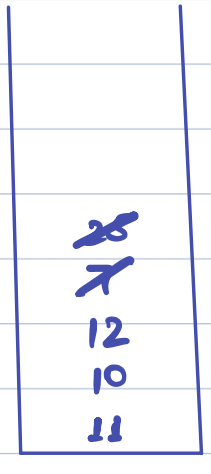
pop() getMin()

↓ ↓

7 10

wrong

// idea!



11 10 12 7 25 getMin()

↓

7

pop() pop() getMin()

↓ ↓

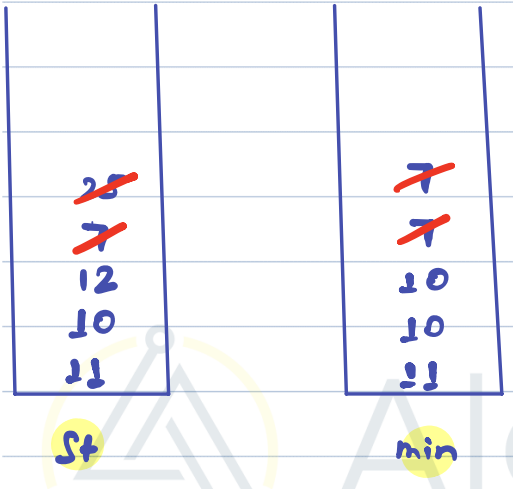
25 7

min: ∞ 11 10 7 10

2ndmin: 10

//Correct idea 1

↳ use 2 Stacks.



11 10 12 7 25 getmin() → 7

pop() → 25 pop() → 7 getmin() → 10

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