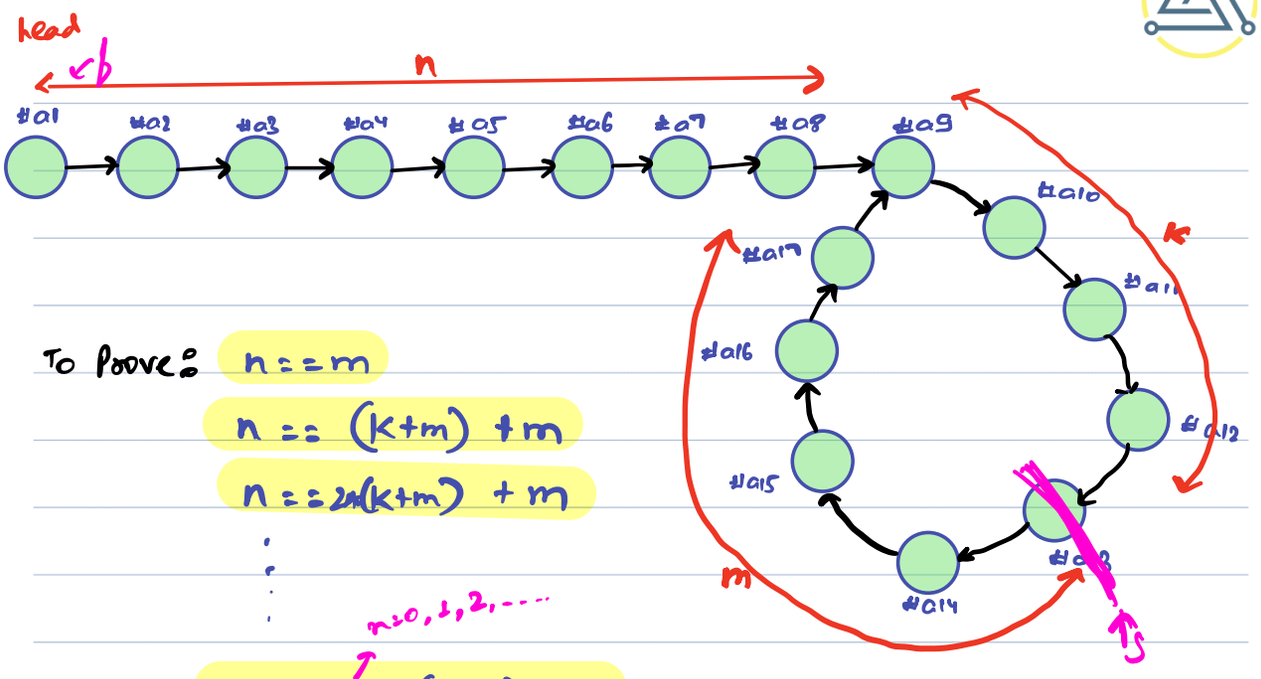


To Proove: $n == m$

$n == (k+m) + m$

$n == 2*(k+m) + m$

.

.

.

$n = n*(k+m) + m$ ($n = 0, 1, 2, \ldots$)

$n = (c-1)*(k+m) + m$

Slow : $n + k$

fast : $n + c*(k+m) + k$

$2 * \text{Slow} = \text{fast}$

$2*(n+k) = n + c*(k+m) + k$

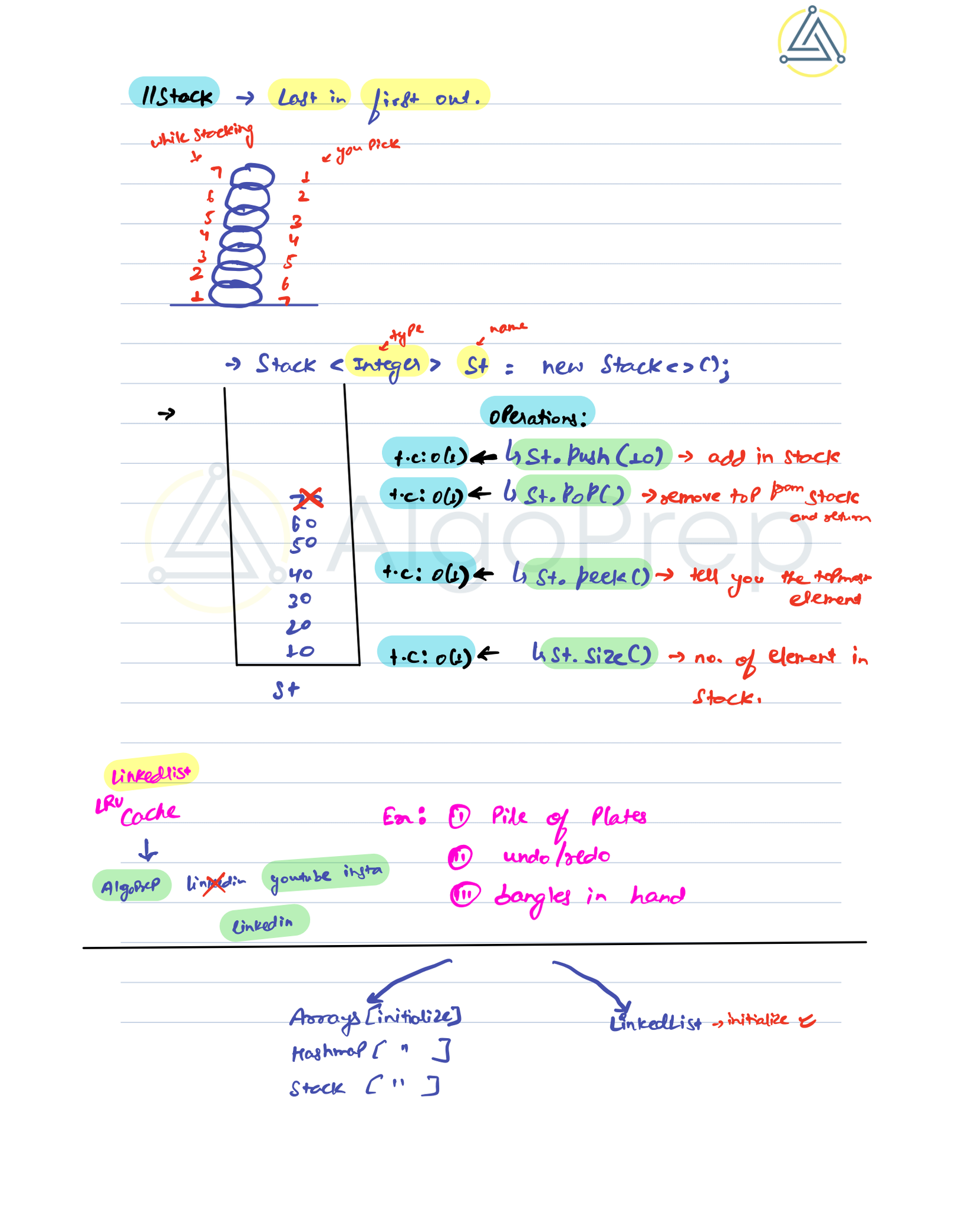
$n + k = c*(k+m)$

$n + k = (c-1)*(k+m) + (k+m)$

$n = (c-1)*(k+m) + m$

Today's agenda

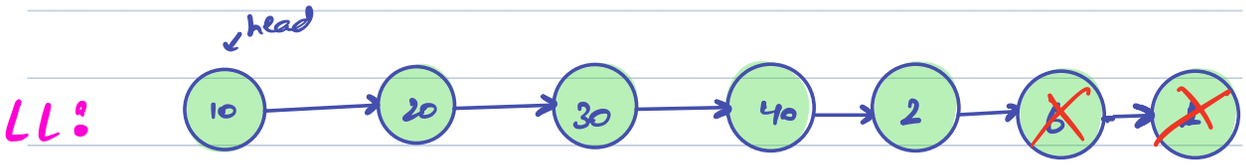
- Stacks
- LinkedList as Stack
- Remove adjacent duplicate
- Balanced Parentheses
- Min Stack


!!Stack → Last in first out.
while stocking
you pick
7 6 5 4 3 2 1
1 2 3 4 5 6 7
→ Stack < Integer > St = new Stack<>();
type
name
Operations:
t.c: O(1) ← St.push(10) → add in stack
t.c: O(1) ← St.pop() → remove top from stack and return
t.c: O(1) ← St.peek() → tell you the topmost element
t.c: O(1) ← St.size() → no. of element in stack.
60
50
40
30
20
10
St
LinkedList
LRU Cache
AlgoPrep
youtube insta
LinkedIn
Ex: (i) Pile of Plates
(ii) undo/redo
(iii) bangles in hand
Arrays [initialize]
Hashmap ["]
Stack ["]
LinkedList → initialize

→adopted

//Linkedlist as Stack

Last in first out

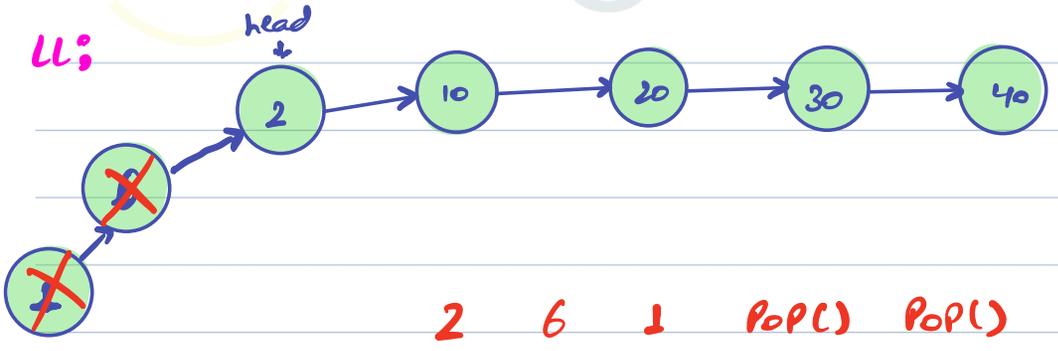


2 6 1 POP() POP()

1 6

(i) addlast → O(N)

(ii) remove last → O(N)



2 6 1 POP() POP()

add first → O(1)

remove first → O(1)

8) Remove adjacent duplicate

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

Ex1: a b b b d → ad

Ex2: a b c c b d e → ade

Ex3: a b b e → abe

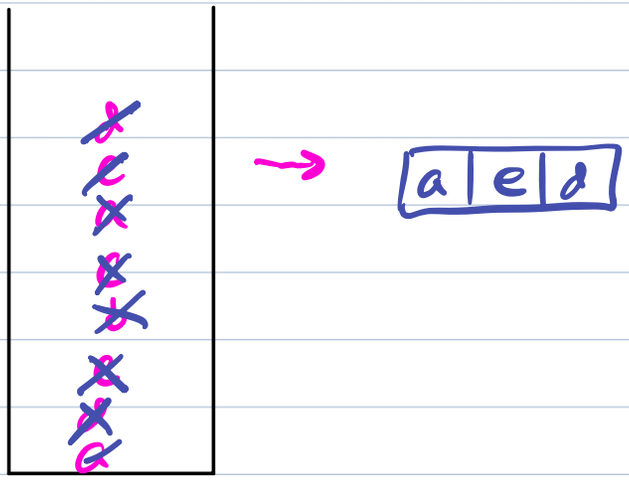
Ex4: a d c b b e c c b d e d

↳ adceded

Ex5: a b b b d a

↳ ada

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



//Pseudo code

```
String RemoveAdjacentDuplicate(String s){

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

    for(int i=0; i<s.length(); i++){
        if(st.size()==0){
            st.push(s.charAt(i));
            continue;
        }

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

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

    → Convert arr to String & return.
}
```

T.C: O(N)

S.C: O(N)

Break till 9:45 PM

Q) Valid Parentheses

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

note: balanced Strings:

(i) open brackets must be closed by the same type of bracket.

s: ( ) { } [ ] (( )) → true

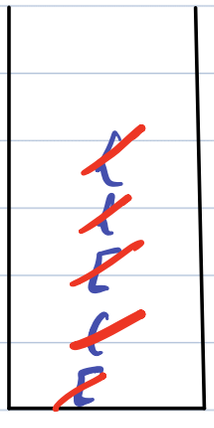
s: ( ) { } [ ) → false

(ii) opening brackets must be closed in correct order.

s: [{] } → false

s: ( ) { (}) → false

s: [( ) [ ] { }] { }



// Pseudo Code

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

    for(int i=0; i<s.length(); i++){
        if(st.size()==0){
            st.push(s[i]);
            continue;
        }

        if(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;}
}
```

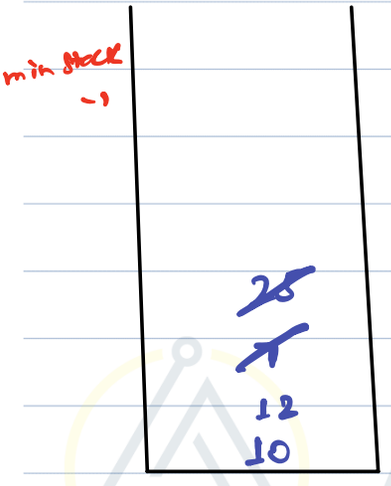
T.C: O(N)

S.C: O(N)

Q) Min Stack

Normal Stack → LIFO → Push(), Peek(), size()
→ Pop() → top element

Min Stack → getMin() → min element of Stack
↳ expected TC: O(1)



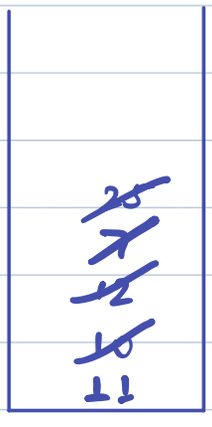
10 12 7 25 getMin() Pop() Pop()
↓
7

getMin()
↓
10

wrong
Idea 1

11 10 12 7 25 getMin() Pop() Pop()
↓ ↓ ↓
7 25 7

Pop() Pop() getMin()
↓ ↓
12 10



min = ~~∞ 11 10 7~~ 10

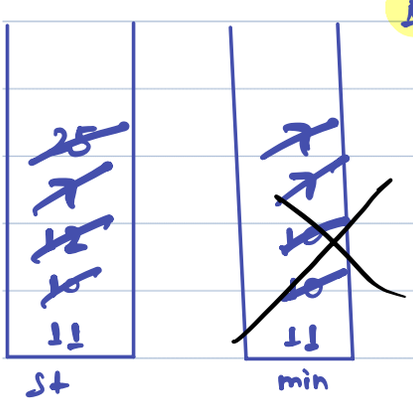
2nd min : ~~∞ 11 10~~

Correct idea !

↳ use 2 stacks

11 10 12 7 25 getmin() PoP() PoP()

↓ 7 ↓ 25 ↓



getmin() PoP() PoP() getmin()

↓ 10 ↓ 12 ↓ 10 ↓ 11

T.C

Push() → O(1)

PoP() → O(1)

getmin() → O(1)

S.C = O(N)

↳ Code → HW

doubt

11 10 12 7 8 25 getmin() PoP() PoP()

↓ 7 ↓ 25 ↓ 8

getmin() PoP() PoP() getmin()

↓ 7 ↓ 7 ↓ 12 ↓ 10