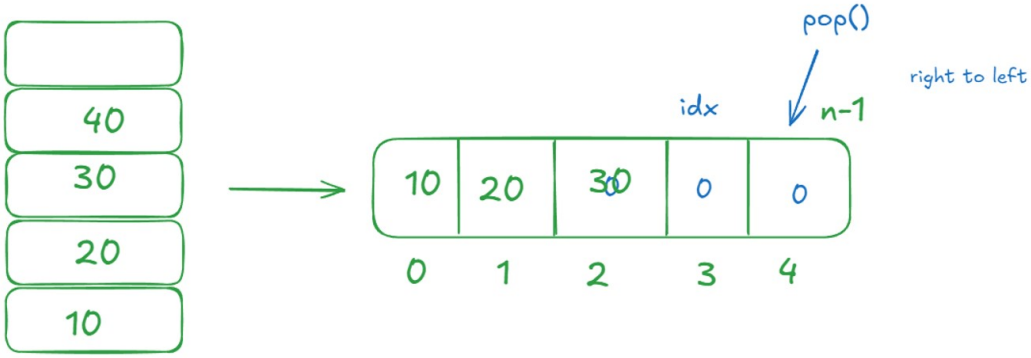
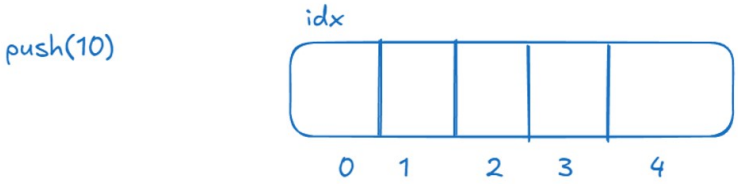
Stack + Queue [Day 16]

Stack Implementation using Array



push() , pop() , peek() , isEmpty() , isFull() , size(), capacity().




```
public class Stack {
    private int[] arr;
    private int idx;
    private int maxSize;

    public Stack(int size){
        this.maxSize = size;
        this.arr = new int[maxSize];
        this.idx = 0;
    }
    public void push(int x){
        if(isFull()){
            System.out.println("Stack is Fulled , Cannot push");
            return;
        }
        arr[idx] = x;
        idx++;
    }
    public int pop(){
        if(isEmpty()){
            System.out.println("Stack is Empty , Cannot pop");
            return -1;
        }
        int top = arr[idx - 1];
        arr[idx - 1] = 0;
        idx--;
```

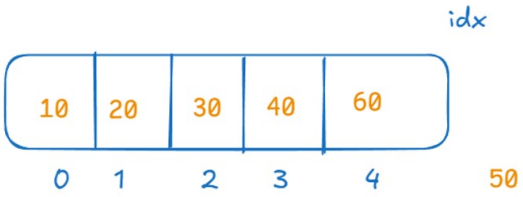
```
    int top = arr[idx - 1];
    arr[idx - 1] = 0;
    idx--;
    return top;
}
```

```
public int peek(){
    if(isEmpty()){
        System.out.println("Stack is Empty");
        return -1;
    }
    return arr[idx - 1];
}
```

```
public boolean isEmpty(){
    return (idx == 0);
}
public boolean isFull(){
    return (idx == maxSize);
}
public int size(){
    return idx;
}
public int capacity(){
    return arr.length;
}
```

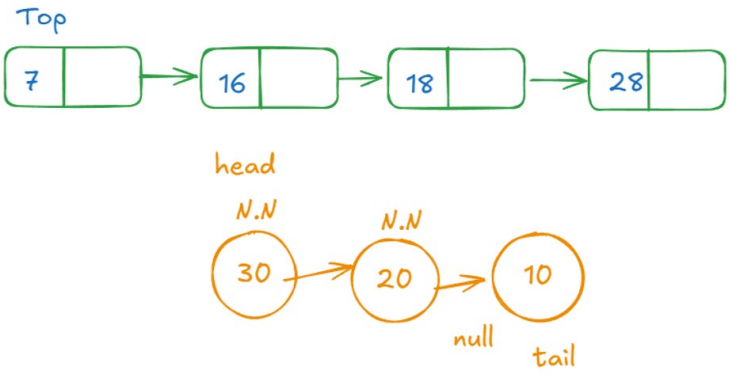
```
public static void main(String[] args) {
    Stack st = new Stack(5);
    st.push(10);
    st.push(20);
    st.push(30);
    st.push(40);
    st.push(50);

    for(int i = 0 ; i < st.size(); i++){
        System.out.print(st.arr[i] + " ");
    }
    System.out.println();
    System.out.println("Peek " + st.peek());
    System.out.println("Pop : " + st.pop());
    for(int i = 0 ; i < st.size(); i++){
        System.out.print(st.arr[i] + " ");
    }
    System.out.println();
    System.out.println("Size : " + st.size());
    st.push(60);
    for(int i = 0 ; i < st.size(); i++){
        System.out.print(st.arr[i] + " ");
    }
    System.out.println();
    System.out.println("Is Full " + st.isFull());
    st.push(70);

}
}
```

}



70 -- isFull() - true , error message

Stack Implementation using LinkedList :




```
class Node{
    int val;
    Node next;

    public Node(int val){
        this.val = val;
        this.next = null;
    }
}
```

```
class myStack{
    private Node head;
    private int size;

    public myStack(){
        this.head = null;
        this.size = 0;
    }
    public void push(int val){
        Node newNode = new Node(val);
        newNode.next = head;
        head = newNode;
        size++;
    }
```

```
    public int pop(){
        if(isEmpty()){
            System.out.println("Stack is Empty , Cannot Pop");
            return -1;
        }
        int top = head.val;
        head = head.next;
        size--;
        return top;
    }

    public int peek(){
        if(isEmpty()){
            System.out.println("Stack is Empty , Nothing to Peek");
            return -1;
        }
        return head.val;
```

```
        System.out.println("Stack is Empty , Nothing to Peek");
        return -1;
    }
    return head.val;
}
```

```
public boolean isEmpty(){
    return head == null;
}

public int size(){
    return size;
}

public void displayStack(){
    if(isEmpty()){
        System.out.println("Stack is Empty ");
        return;
    }
    Node temp = head;
    System.out.println("Stack Element : ");
    while(temp != null){
        System.out.print(temp.val + " ");
        temp = temp.next;
    }
    System.out.println();
}
}

public class StackUsingLL {
    public static void main(String[] args) {
        myStackst = new myStack();
        st.push(10);
        st.push(20);
        st.push(30);
        st.push(40);
        st.push(50);

        st.displayStack();

    }
}
```

20. Valid Parentheses

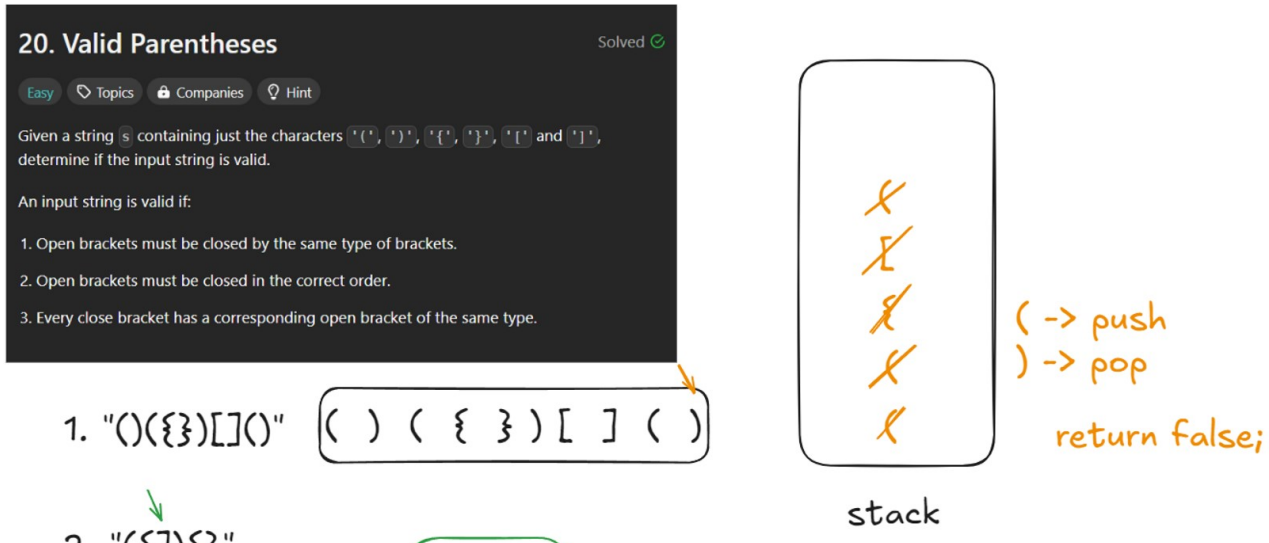
Solved

Easy | Topics | Companies | Hint

Given a string `s` containing just the characters `'('`, `')'`, `'{'`, `'}'`, `'['` and `']'`, determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.
3. Every close bracket has a corresponding open bracket of the same type.



return false;

stack

Hint 1

Use a stack of characters.

Hint 2

When you encounter an opening bracket, push it to the top of the stack.

Hint 3

When you encounter a closing bracket, check if the top of the stack was the opening for it. If yes, pop it from the stack. Otherwise, return false.

st

```java
class Solution {
    public boolean isValid(String s) {
        Stack<Character> st = new Stack<>();
        for(char ch : s.toCharArray()){
            if(ch == '(' || ch == '{' || ch == '['){
                st.push(ch);
            }else if(ch == ')' && !st.isEmpty() && st.peek() == '('){
                st.pop();
            }else if(ch == '}' && !st.isEmpty() && st.peek() == '{'){
                st.pop();
            }else if(ch == ']' && !st.isEmpty() && st.peek() == '['){
                st.pop();
            }else{
                return false;
            }
        }

        return st.isEmpty();
    }
```

1. {}[]())

2. [{}]()({[

stack

921. Minimum Add to Make Parentheses Valid

Solved

Medium Topics Companies

A parentheses string is valid if and only if:

- It is the empty string,
- It can be written as `AB` (`A` concatenated with `B`), where `A` and `B` are valid strings, or
- It can be written as `(A)`, where `A` is a valid string.

You are given a parentheses string `s`. In one move, you can insert a parenthesis at any position of the string.

- For example, if `s = "())"`, you can insert an opening parenthesis to be `"(())"` or a closing parenthesis to be `"())("`.

Return *the minimum number of moves required to make* `s` *valid.*

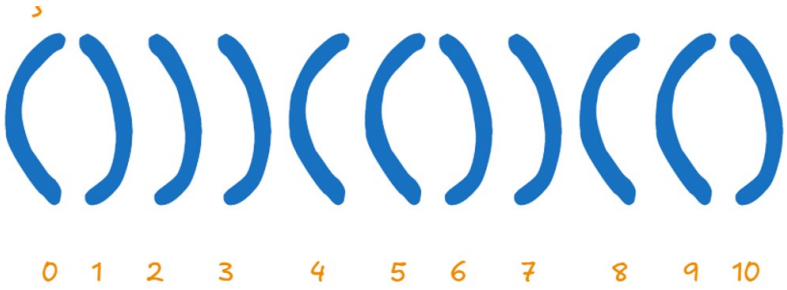
```
if(ch == "("){
    left++;
}else{
    if(left > 0){
        left--;
    } else{
        right++;
    }
}
```

left = 1

right = 2

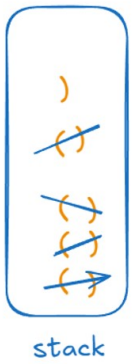
st.size()

stack




```java
public int minAddToMakeValid(String s) {
    int leftBalance = 0;
    int rightBalance = 0;

    for(char ch : s.toCharArray()){
        if(ch == '('){
            leftBalance++;
        }else if(ch == ')'){
            if(leftBalance > 0){
                leftBalance--;
            }else{
                rightBalance++;
            }
        }
    }
    return leftBalance + rightBalance;
}
```

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

for(char ch : s.toCharArray()){
    if(ch == '('){
        st.push(ch);
    }else if(ch == ')'){
        if(!st.isEmpty() && st.peek() == '(' ){
            st.pop();
        }else{
            st.push(ch);
        }
    }
}
return st.size();
```



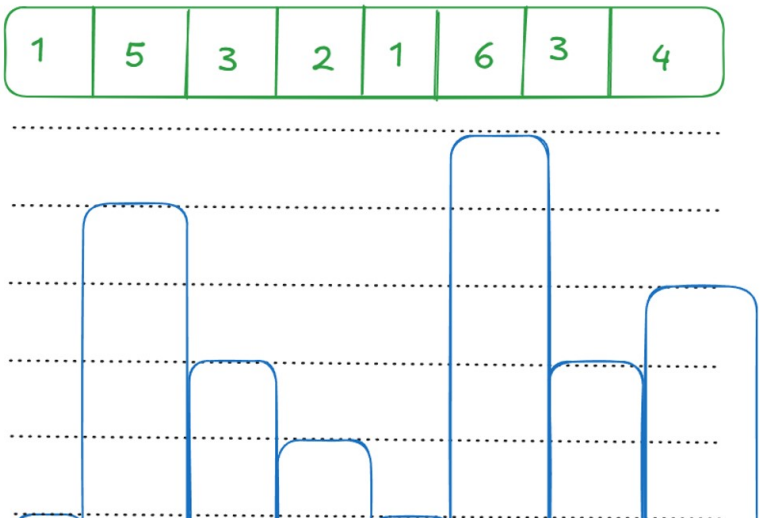
size = 1

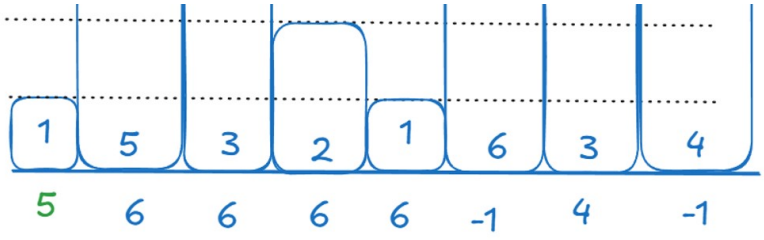
Change Status

Completed

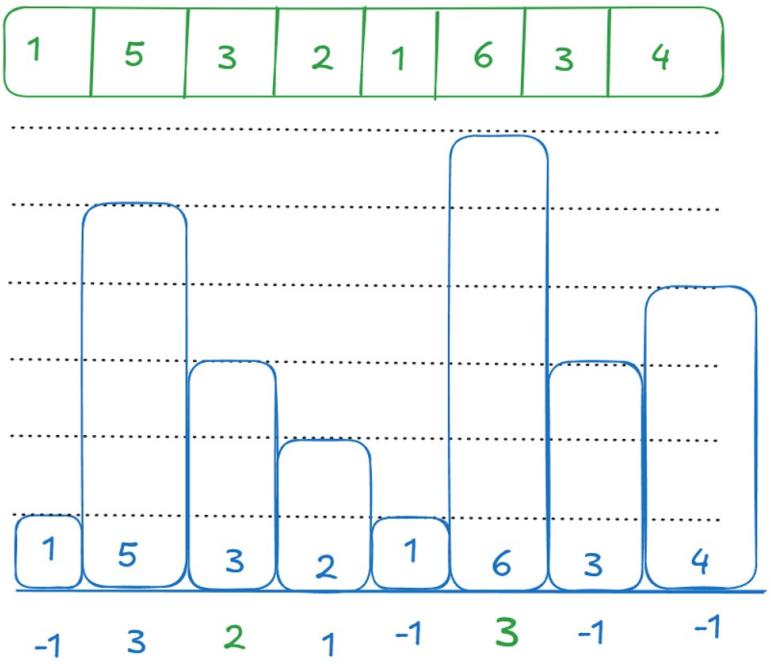
Attendance Code: 4070502D

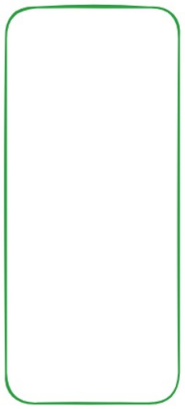
Next Greater Element : [right to left]





Next Smaller Element : [right to left]





Previous Greater Element :[Left to right]