Stacks and Queues

Two Stacks - Coding Ninjas

Lecture 54: Introduction to Stacks [Theory + Implementation] || C++ Placement Course

In this Video, we are going to learn about Stack Data Structure and its Implementation.

There is a lot to learn, Keep in mind “ Mnn bhot karega k chor yrr apne se nahi hoga ya maza nahi

https://www.youtube.com/watch?v=_6COl6V6mng&list=PLDzeHZWIZsTrhXYYtx4z8-u8zA-DzuVsj&index=1

timestamp - 31:20

```
import java.util.* ;
import java.io.*;
public class TwoStack {
    int[] arr;
  int top1 ;
  int top2 ;
  int size;
  // Initialize TwoStack.
  public TwoStack(int s) {
    // Write your code here
       this.size  =s ;
     top1= -1 ;
     top2 = size ;
     arr= new int[s];

  }

  // Push in stack 1.
  public void push1(int num) {

  // atleast 1 empty space is present in array
  if(top2-top1 >1){
      top1++ ;
    arr[top1] = num ;
  }

  }

  // Push in stack 2.
  public void push2(int num) {
  // atleast 1 empty space is present in array
  if(top2-top1 >1){
      top2-- ;
    arr[top2] = num ;
  }

  }
```

```
  // Pop from stack 1 and return popped element.
  public int pop1() {
  if(top1>=0){
    return(arr[top1--]) ;
  }
  return -1 ;

  }

  // Pop from stack 2 and return popped element.
  public int pop2() {
    if(top2<size){
      return arr[top2++] ;
    }
   return -1 ;
  }

}
```

Reverse a string using stack

```
import java.util.*;

class Practice2 {

    public static void main(String[] args) {
        String str = "smarani";
        Stack<Character> stack = new Stack<>();

// Push in stack -
// Stack returns the input in reverse order - property of stack
        for (int i = 0; i < str.length(); i++) {
            char c = str.charAt(i);
            stack.push(c);
        }
        String ans = "";

        while (!stack.empty()) {
            ans += stack.pop();
        }
        System.out.println(ans);
    }

}
```

Pop the middle of the array using Stack

```
import java.util.* ;
import java.io.*;
public class Solution {
```

```
  public static void deleteMiddle(Stack<Integer> inputStack, int N) {

    int count=0 ;
    solve(inputStack, N, count);
    System.out.print(inputStack)  ;
  }

public static void solve(Stack<Integer> inputStack, int size, int count) {

        if (count == size / 2) {

            inputStack.pop();
            return;
        }

        int last = inputStack.peek();
        inputStack.pop();
        solve(inputStack, size, count + 1);
        inputStack.push(last);

    }

}
```

Postfix expression evaluation - works for only single digit

```
import java.util.*;

class Practice2 {
    public static Stack<Integer> s = new Stack<>();

    public static void main(String[] args) {
        Stack<Integer> stack = new Stack<>();
        String expression = "100200+2/5*7+";

        for (char c : expression.toCharArray()) {
            if (c >= 48 && c <= 57) {
                stack.push(c - '0');
            }
            else {
                int num2 = stack.pop();
                int num1 = stack.pop();

                switch (c) {
                    case '+':
                        stack.push(num1 + num2);
                        break;

                    case '-':
                        stack.push(num1 - num2);
                        break;

                    case '/':
                        stack.push(num1 / num2);
                        break;

                    case '*':
                        stack.push(num1 * num2);
                        break;
```

```
                }
            }

        }
        System.out.println(stack.pop());

    }

}
```

Valid Parentheses

```
Stack<Character> stack = new Stack<Character>();
    for (int i = 0; i < s.length(); i++) {
        // iterate through the string to add(push) into empty stack with "([{" letters only.
        // if characters are not under "([{", then it will be under ")]}"
        if(s.charAt(i) == '(' || s.charAt(i) == '[' || s.charAt(i) == '{') {
            stack.push(s.charAt(i));
        }
        // if chacter is not "([{", then see if the stack is empty.
        // if the stack is not empty, then
        //      remove and get(pop) the bottom element of stack and compare with other pairs of "([{"
        // if popped element is one of those pairs, move on to next element
        // if not, the order is not correct, so return false; ex) ([)] --> false
        else {
            if(s.charAt(i) == ')') {
                if(stack.isEmpty() || stack.pop() != '(')
                return false;
            }
            if(s.charAt(i) == ']') {
                if(stack.isEmpty() || stack.pop() != '[')
                 return false;
            }
            if(s.charAt(i) == '}') {
                if(stack.isEmpty() || stack.pop() != '{')
                return false;
            }
        }
    }
    // if the stack is empty, it means the loop went through without any problem --> True
    // if the stack is not empty, stack will have only one single pair of one of "([{"
    //      ex) string s = "["  --> false;
    return stack.isEmpty();
```

Insert An Element At Its Bottom In A Given Stack

Coding Ninjas Studio

https://www.codingninjas.com/studio/problems/insert-an-element-at-its-bottom-in-a-given-stack_1171166?leftPanelTab=0

```
import java.util.* ;
import java.io.*;
public class Solution
{
  public static Stack<Integer> pushAtBottom(Stack <Integer> myStack, int x)
  {

    solve(myStack,x) ;
    return myStack ;
  }

  public static void solve(Stack<Integer> myStack, int x){
    if(myStack.isEmpty()){
      myStack.push(x) ;
      return ;
    }

    int last = myStack.pop() ;
   pushAtBottom(myStack, x) ;
     myStack.push(last) ;
  }
}
```

Reverse Stack Using Recursion

```
import java.util.Stack;

public class Solution {

  public static void reverseStack(Stack<Integer> stack) {
    if(stack.isEmpty()){
      return ;
    }

    int last = stack.pop() ;
    reverseStack(stack) ;
    pushAtBottom(stack , last ) ;
  }

  public static void pushAtBottom(Stack<Integer> myStack ,int x){

    if(myStack.isEmpty()){
      myStack.push(x) ;
      return ;
    }

    int last = myStack.pop() ;
    pushAtBottom(myStack, x) ;
     myStack.push(last) ;

  }
}
```

Sort Stack

Coding Ninjas Studio

https://www.codingninjas.com/studio/problems/abc_1229513?leftPanelTab=0

```
import java.util.*;

class Practice2 {
    public static Stack<Integer> s = new Stack<>();

    public static void main(String[] args) {
        Stack<Integer> stack = new Stack<>();
        stack.push(-1);
        stack.push(4);
        stack.push(2);
        stack.push(9);
        stack.push(0);

        Stack<Integer> tmpStack = new Stack<Integer>();
// final stack

        while (!stack.isEmpty()) {
            int tmp = stack.pop();
// comparing the top of input stack to the eleemt in temp stack.

            while (!tmpStack.isEmpty() && tmpStack.peek() > tmp) {
// if smaller element found in input stack than the last element of the temp stack, pop that element and push i
t in the the original stack .
                stack.push(tmpStack.pop());
            }
            tmpStack.push(tmp);
        }
        System.out.println(tmpStack);

    }

}
```

OR

Recursive solution - love babbar code but isn't working

```
if(s.isEmpty()){
      return s;
    }

    int last = s.pop() ;
    sortStack(s) ;
     sortedInsert(s,last) ;

  }

  public static void sortedInsert(Stack<Integer> stack, int x){

    if(stack.isEmpty() || (stack.peek()<x)){
      stack.push(x) ;
```

```
      return ;
    }

    int last = stack.pop() ;
    sortedInsert(stack,x) ;
    stack.push(last)  ;
```

Redundant Brackets

Coding Ninjas Studio

https://www.codingninjas.com/studio/problems/redundant-brackets_975473?leftPanelTab=0

```
// No need to check for characters - a,b etc
import java.util.* ;
import java.io.*;
import java.util.ArrayList;
public class Solution
{
    public static boolean findRedundantBrackets(String s)
    {
        Stack<Character> st = new Stack<>();

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

            char ch = s.charAt(i);

            if(ch == '(' || ch == '+' || ch == '-'|| ch == '/'|| ch == '*' ){

                st.push(ch);

            }else if(ch == ')'){

                if(st.peek() != '('){

                while(st.peek() != '('){

                    st.pop();

                }

                st.pop();

                }else{

                    return true;
                }
            }
        }
        return false;
    }
}
```

Minimum Cost To Make String Valid

formula by love babbar = (a+1)/2 + (b+1)/2 ;

Coding Ninjas Studio

https://www.codingninjas.com/studio/problems/minimum-cost-to-make-string-valid_1115770?leftPanelTab=0

Lecture 55: Stack Interview Questions || Placement Series by Love Babbar

In this Video, we are going to solve 8 Stack Interview Questions.

There is a lot to learn, Keep in mind " Mnn bhot karega k chor yrr apne se nahi hoga ya

https://www.youtube.com/watch?v=BmZnJehDzyU&t=2389s

Timestamp - 1:13:49

```
import java.util.* ;
import java.io.*;
public class Solution {
    public static int findMinimumCost(String str) {

// odd length string can never be made valid
      if(str.length()%2==1){
        return -1;
      }

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

  // first if, else  - removing valid paranthesis
    for( int i = 0;i<str.length() ;i++){
         if(str.charAt(i)=='{'){
           stack.push(str.charAt(i)) ;

         }
         else{
           if(!stack.isEmpty() && stack.peek()=='{'){
             stack.pop() ;

           }
  // saving the invalid paranthesis in the stack (after removing the valid paranthesis)
           else{
             stack.push(str.charAt(i)) ;
           }

         }
       }

   int a = 0 ;
   int b = 0 ;
   int size = stack.size();
   // stack is empty means expression is valid
   if(stack.isEmpty()) {
      return 0 ;
   }

// formula by love babbar = (a+1)/2 + (b+1)/2 ;
// where a = no. of open brackets , b = no.of closed brackets
       for( int i =0 ;i<size ;i++){
```

```
            if(stack.pop()=='{'){
              a++ ;
            }
            else{
              b++ ;
            }
        }
   return ((a+1)/2 + (b+1)/2) ;
    }
}
```

Next Smaller Element

Coding Ninjas Studio

https://www.codingninjas.com/studio/problems/next-smaller-element_1112581?leftPanelTab=0

Lecture 56: Largest Rectangular Area in Histogram [Optimised Approach]

In this Video, we are going to solve one of famous Stack Hard Question.

There is a lot to learn, Keep in mind " Mnn bhot karega k chor yrr apne se nahi hoga ya

https://www.youtube.com/watch?v=lJLcqDsmYfg&list=PLDzeHZWIZsTrhXYYtx4z8-u8zA-DzuVsj&index=3

timestamp -5:19

```
import java.util.*;
import java.io.*;

public class Solution{
    static ArrayList<Integer> nextSmallerElement(ArrayList<Integer> arr, int n){
       ArrayList<Integer> list = new ArrayList<>(Collections.nCopies(n, 0));
      Stack<Integer> stack = new Stack<>();
      stack.push(-1) ;

     for(int i = arr.size()-1 ; i>=0 ;i--){
         if(arr.get(i)>stack.peek()){
             list.set(i,stack.peek()) ;
             stack.push(arr.get(i))  ;
         }
         else{
             while(arr.get(i)<=stack.peek()){
                 stack.pop() ;
             }
             list.set(i,stack.peek()) ;
             stack.push(arr.get(i));
         }
     }
```

```
     return list ;
    }
}
```

OR

Nearest Smaller Element on Left & Right side of an Array | Stack | DSA-One Course #42

Hey guys, In this video, We're going to solve an important problem called the Nearest smaller element on the left & right sides of an Array. We'll also solve the Nearest greater element on the left & right sides.

https://www.youtube.com/watch?v=_RtghJnM1Qo

Previous Smaller Element

```
import java.util.*;
import java.util.Collections;
import java.util.Stack;

public class Practice2 {
    public static void main(String[] args) {
        int[] a = { 4, 10, 5, 8, 20, 15, 3, 12 };
        int[] small = new int[a.length];
        Stack<Integer> s = new Stack<>();
        for (int i = 0; i < a.length; i++) {
            while (!s.isEmpty() && s.peek() >= a[i]) {
                s.pop();
            }

            if (s.isEmpty()) {
                small[i] = -1;
            } else {
                small[i] = s.peek();
            }
            s.push(a[i]);
        }

        System.out.println(Arrays.toString(small));
    }
}
```

Previous Greater Element

```
import java.util.*;
import java.util.Collections;
import java.util.Stack;

public class Practice2 {
    public static void main(String[] args) {
        int[] a = { 4, 10, 5, 8, 20, 15, 3, 12 };
        int[] large = new int[a.length];
```

```
        Stack<Integer> s = new Stack<>();
        for (int i = 0; i < a.length; i++) {
            while (!s.isEmpty() && s.peek() <= a[i]) {
                s.pop();
            }

            if (s.isEmpty()) {
                large[i] = -1;
            } else {
                large[i] = s.peek();
            }
            s.push(a[i]);
        }

        System.out.println(Arrays.toString(large));
    }
}
```

Next greater element

```
import java.util.*;
import java.util.Collections;
import java.util.Stack;

public class Practice2 {
    public static void main(String[] args) {
        int[] a = { 4, 10, 5, 8, 20, 15, 3, 12 };
        int[] large = new int[a.length];
        Stack<Integer> s = new Stack<>();
        for (int i = a.length - 1; i >= 0; i--) {
            while (!s.isEmpty() && s.peek() <= a[i]) {
                s.pop();
            }

            if (s.isEmpty()) {
                large[i] = -1;
            } else {
                large[i] = s.peek();
            }
            s.push(a[i]);
        }

        System.out.println(Arrays.toString(large));
    }
}
```

Next Smaller Element

```
import java.util.Collections;
import java.util.Stack;

public class Practice2 {
    public static void main(String[] args) {
```

```
        int[] a = { 4, 10, 5, 8, 20, 15, 3, 12 };
        int[] large = new int[a.length];
        Stack<Integer> s = new Stack<>();
        for (int i = a.length - 1; i >= 0; i--) {
            while (!s.isEmpty() && s.peek() >= a[i]) {
                s.pop();
            }

            if (s.isEmpty()) {
                large[i] = -1;
            } else {
                large[i] = s.peek();
            }
            s.push(a[i]);
        }

        System.out.println(Arrays.toString(large));
    }
}
```

84. Largest Rectangle in Histogram

Using the previous concepts of previous small element and next small element

```
class Solution {
    public int largestRectangleArea(int[] a) {
// previous smaller element
     int[] prevsmall = new int[a.length];
        Stack<Integer> s = new Stack<>();
        for (int i = 0; i < a.length; i++) {
            while (!s.isEmpty() && a[s.peek()] >= a[i]) {
                s.pop();
            }

            if (s.isEmpty()) {
                prevsmall[i] = -1;
            } else {
                prevsmall[i] = s.peek();
            }
            s.push(i);
        }

    // next smaller element
 s = new Stack<>();
 int[] nextsmall = new int[a.length] ;
        for (int i = a.length - 1; i >= 0; i--) {
            while (!s.isEmpty() && a[s.peek()] >= a[i]) {
                s.pop();
            }

            if (s.isEmpty()) {
                nextsmall[i] = a.length;
            } else {
                nextsmall[i] = s.peek();
            }
            s.push(i);
        }

int maxArea = Integer.MIN_VALUE ;
for( int i =0 ;i<a.length ;i++){
    int area = (nextsmall[i]-prevsmall[i]-1)*a[i] ;
```

```
    maxArea = Math.max(maxArea,area) ;
}

return maxArea ;
}
}
```

The Celebrity Problem

The Celebrity Problem | Practice | GeeksforGeeks

A celebrity is a person who is known to all but does not know anyone at a party. If you go to a party of N people, find if there is a celebrity in the party or not. A square NxN matrix M[][] is used to represent people at the party such that if an e

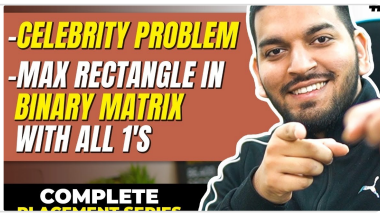
https://practice.geeksforgeeks.org/problems/the-celebrity-problem/1?utm_source=gfg&utm_medium=article&utm_campaign=bottom_sticky_on_article

Lecture 57: Stack - Celebrity Problem && Max Rectangle in Binary Matrix with all 1's

In this Video, we are going to solve some of famous Stack Hard Questions.

There is a lot to learn, Keep in mind " Mnn bhot karega k chor yrr apne se nahi hoga ya maza

https://www.youtube.com/watch?v=9u2BJfmWNEg&list=PLDzeHZWIZsTrhXYYtx4z8-u8zA-DzuVsj&index=4




```
class Solution
{
    //Function to find if there is a celebrity in the party or not.
    int celebrity(int M[][], int n)
    {
      Stack<Integer> stack = new Stack<>() ;

    // push all element in the stack
      for( int i = 0;i<n;i++){
          stack.push(i) ;
      }

      // Compare until only the possible celebrity index remains in the stack
      while(stack.size()!=1){

          int a = stack.pop() ;
          int b = stack.pop() ;

          if(M[a][b]==1){  //if a knows b , discard a , push b
              stack.push(b) ;

          }
          else if(M[b][a]==1){  //if a knows b , discard b, push a
              stack.push(a);
          }
          if(stack.size()==0){
          return -1 ;
      }
      }

      // ind is the potential celeb
      int ind = stack.pop() ;
```

```
      // row check , rows of ind should be 0
//(celebrity does not know anyone)
      for( int i= 0;i<n ;i++){

          if(i==ind ){
              continue;
          }

          if(M[ind][i]==1){
              return -1 ;
          }
      }


      // column check, cols of ind must be 1
 // (everyone knows celebrity)

      for(int i = 0;i<n ;i++){
         if(i==ind){
             continue ;
         }

         if(M[i][ind]==0){
             return -1 ;
         }
      }

      return ind ;
    }
}
```

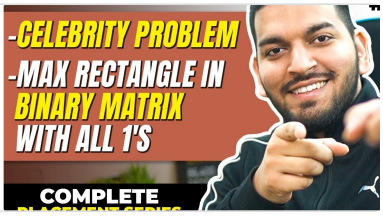
85. Maximal Rectangle

Lecture 57: Stack - Celebrity Problem && Max Rectangle in Binary Matrix with all 1's

In this Video, we are going to solve some of famous Stack Hard Questions.

There is a lot to learn, Keep in mind " Mnn bhot karega k chor yrr apne se nahi hoga ya maza

https://www.youtube.com/watch?v=9u2BJfmWNEg&list=PLDzeHZWIZsTrhXYYtx4z8-u8zA-DzuVsj&index=4



Timestamp - 25:44

Based on **84. Largest Rectangle in Histogram**

```
class Solution {
    public int maximalRectangle(char[][] M) {
       int[][] matrix = new int[M.length][M[0].length];

        for (int i = 0; i < M.length; i++) {
            for (int j = 0; j < M[i].length; j++) {
```

```
                matrix[i][j] = M[i][j] - '0';

            }
        }

        int area = largestRectangleArea(matrix[0]);
        for (int i = 1; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                if (matrix[i][j] != 0) {
                    matrix[i][j] = matrix[i][j] + matrix[i - 1][j];
                }
            }
            System.out.println(Arrays.toString(matrix[i]));
            area = Math.max(area, largestRectangleArea(matrix[i]));
        }
        return area;
    }

public int largestRectangleArea(int[] a) {
// previous smaller element
     int[] prevsmall = new int[a.length];
        Stack<Integer> s = new Stack<>();
        for (int i = 0; i < a.length; i++) {
            while (!s.isEmpty() && a[s.peek()] >= a[i]) {
                s.pop();
            }

            if (s.isEmpty()) {
                prevsmall[i] = -1;
            } else {
                prevsmall[i] = s.peek();
            }
            s.push(i);
        }

    // next smaller element
 s = new Stack<>();
 int[] nextsmall = new int[a.length] ;
        for (int i = a.length - 1; i >= 0; i--) {
            while (!s.isEmpty() && a[s.peek()] >= a[i]) {
                s.pop();
            }

            if (s.isEmpty()) {
                nextsmall[i] = a.length;
            } else {
                nextsmall[i] = s.peek();
            }
            s.push(i);
        }

int maxArea = Integer.MIN_VALUE ;
for( int i =0 ;i<a.length ;i++){
    int area = (nextsmall[i]-prevsmall[i]-1)*a[i] ;
    maxArea = Math.max(maxArea,area) ;
}

return maxArea ;
}


}
```