

# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

## EXPERIMENT 7

**Student Name:** Praduman Kumar

**UID:** 20BCS9446

**Branch:** CSE

**Section/Group:** 20BCS\_DM\_714-A

**Semester:** 06

**Subject Name:** Competitive Coding

**Subject Code:** 20CSP-351

1. **AIM:** To demonstrate the concept of Divide and Conquer

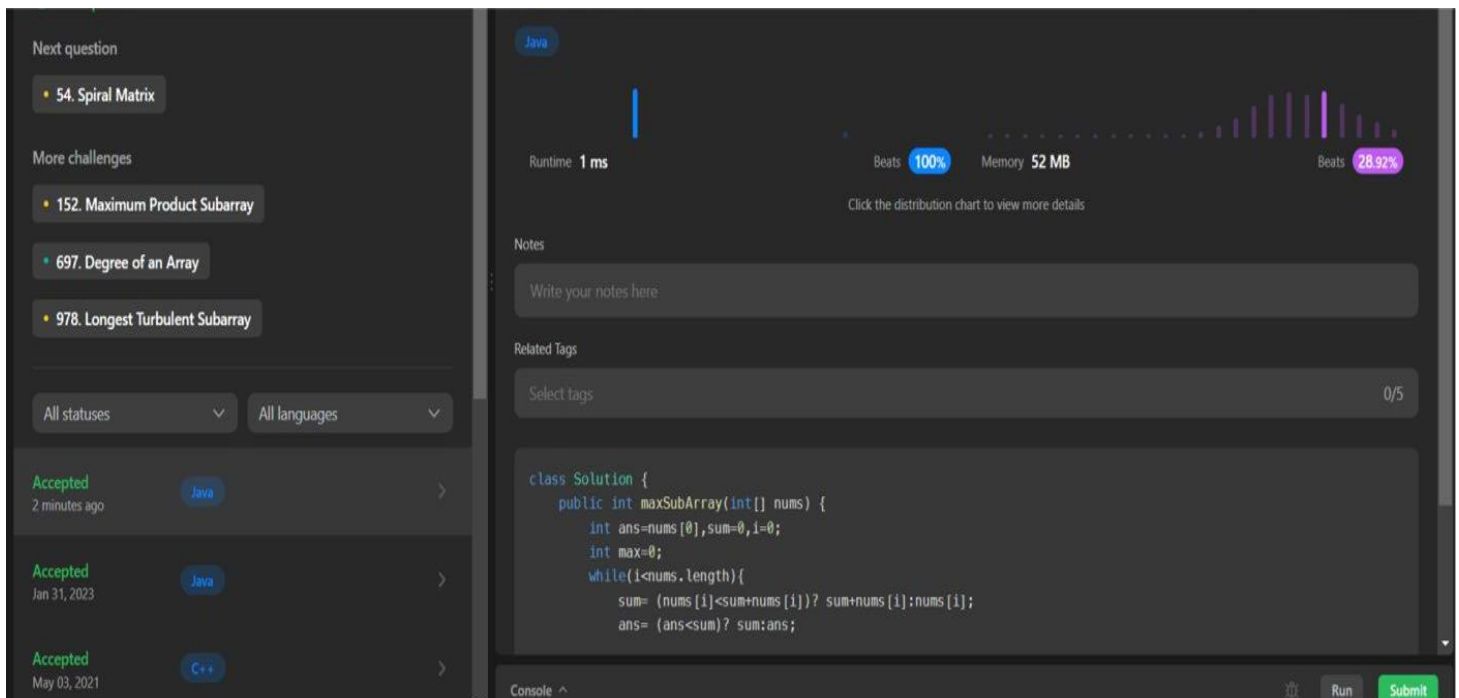
2. **OBJECTIVE 1:** Maximum Subarray

3. **CODE:**

```
class Solution {  
    public int maxSubArray(int[] nums) {  
        int ans=nums[0],sum=0,i=0;  
        int max=0;  
        while(i<nums.length){  
            sum= (nums[i]<sum+nums[i])?  
                sum+nums[i]:nums[i]; ans= (ans<sum)? sum:ans;  
  
            i++;  
        }  
        return ans;  
    }  
}
```

# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

## 4. OUTPUT:



The screenshot shows the LeetCode interface for the 'Maximum Product Subarray' problem (152). The solution is implemented in Java. The runtime is 1 ms, and the memory is 52 MB, both of which are optimal (100% beats). The code defines a class 'Solution' with a method 'maxSubArray' that iterates through the array, maintaining a running sum and updating the maximum product found.

```
class Solution {
    public int maxSubArray(int[] nums) {
        int ans=nums[0],sum=0,i=0;
        int max=0;
        while(i<nums.length){
            sum= (nums[i]<sum+nums[i])? sum+nums[i]:nums[i];
            ans= (ans<sum)? sum:ans;
        }
        return ans;
    }
}
```

## 5. OBJECTIVE 2: Construct Binary Tree From Inorder and PostOrder Traversal

## 6. CODE:

```
class Solution {
    public TreeNode buildTree(int[] inorder, int[] postorder) {
        if(inorder==null || postorder==null || inorder.length!=postorder.length)
            return null;
        HashMap<Integer,Integer> hm=new HashMap<Integer,Integer>();
        for(int i=0;i<inorder.length;i++){
            hm.put(inorder[i],i);
        }
        return buildTreePostIn(inorder,0,inorder.length-1,postorder,0,postorder.length-1,hm);
    }
}
```

# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
    }  
    private TreeNode buildTreePostIn(int[] inorder,int is,int ie,int[]  
postorder,int ps,int pe, HashMap<Integer,Integer>hm){  
        if(ps>pe || is>ie) return null;  
        TreeNode root=new TreeNode(postorder[pe]);  
  
        int inRoot=hm.get(postorder[pe]);  
        int numsLeft=inRoot-is;  
        root.left=buildTreePostIn(inorder,is,inRoot-1,postorder,ps,ps+numsLeft-1,hm);  
        root.right=buildTreePostIn(inorder,inRoot+1,ie,postorder,ps+numsLeft,pe-1,hm);  
        return root;  
    }  
}
```

## 7. OUTPUT

Next question

- 107. Binary Tree Level Order Traversal II

More challenges


- 105. Construct Binary Tree from Preorder and Inorder Traversal

All statuses ▾ All languages ▾

Accepted  
a few seconds ago

Accepted  
Mar 16, 2023

Java



Runtime 2 ms Beats 96.71% Memory 42.6 MB Beats 35.39%

Click the distribution chart to view more details

Notes

Write your notes here

Related Tags

Select tags 0/5

```
/**  
 * Definition for a binary tree node.  
 * public class TreeNode {  
 *     int val;  
 *     TreeNode left;  
 *     TreeNode right;  
 * }
```