

PROBLEM-1

AIM:-

Longest Nice Substring

CODE:-

```
class Solution {  
    public String longestNiceSubstring(String s) {  
        Set<Character> charSet = new HashSet<>();  
        for (int i = 0; i < s.length(); i++) {  
            charSet.add(s.charAt(i));  
        }  
        for (int i = 0; i < s.length(); i++) {  
            if (charSet.contains(Character.toUpperCase(s.charAt(i))) &&  
                charSet.contains(Character.toLowerCase(s.charAt(i)))) {  
                continue;  
            }  
            String s1 = longestNiceSubstring(s.substring(0, i));  
            String s2 = longestNiceSubstring(s.substring(i+1));  
            return s1.length()>= s2.length() ? s1 : s2;  
        }  
        return s;  
    }  
}
```

OUTPUT:-

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

s =  
"YazaAay"

Output

"aAa"

Expected

"aAa"

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

s =  
"Bb"

Output

"Bb"

Expected

"Bb"

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

s =  
"c"

Output

""

Expected

""

PROBLEM-2

AIM:-

Reverse Bits

CODE:-

```
public class Solution {  
    public int reverseBits(int n) {  
        int ans = 0;  
        for (int i = 0; i < 32; i++) {  
            ans <<= 1;  
            ans |= (n & 1);  
            n >>= 1;  
        }  
        return ans;  
    }  
}
```

OUTPUT:

☒ Testcase ☒ Test Result

Accepted Runtime: 0 ms

• Case 1

• Case 2

Input

n =  
00000010100101000001111010011100

Output

964176192 (00111001011110000010100101000000)

Expected

964176192 (00111001011110000010100101000000)

☒ Testcase | ☒ Test Result

Accepted Runtime: 0 ms

• Case 1

• Case 2

Input

n =  
1111111111111111111111111111101

Output

3221225471 (1011111111111111111111111111111)

Expected

3221225471 (1011111111111111111111111111111)

### PROBLEM-3

**AIM:-**

Number of 1 bits

**CODE:-**

```
class Solution {
    public int hammingWeight(int n) {
        String Binary_no=Integer.toBinaryString(n);
        return count(Binary_no,'1');
    }
    public int count(String str,char value){
        int count=0;
        for (int i=0;i<str.length();i++){
            char a=str.charAt(i);
            if (a==value){
                count++;
            }
        }
        return count;
    }
}
```

**OUTPUT:-**

```
Testcase | >_ Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input
n =
11

Output
3

Expected
3
```

Testcase | > Test Result

**Accepted** Runtime: 0 ms

- Case 1
- **Case 2**
- Case 3

**Input**

n =  
128

**Output**

1

**Expected**

1

Testcase

> Test Result

Accepted

Runtime: 0 ms

• Case 1

• Case 2

• Case 3

Input

n =  
2147483645

Output

30

Expected

30

# PROBLEM-4

AIM:-

Max Subarray

CODE:-

```
class Solution {  
    public int maxSubArray(int[] nums) {  
        int maxSum = nums[0];  
        int currentSum = nums[0];  
  
        for (int i = 1; i < nums.length; i++) {  
            currentSum = Math.max(nums[i], currentSum + nums[i]);  
            maxSum = Math.max(maxSum, currentSum);  
        }  
  
        return maxSum;  
    }  
}
```

OUTPUT:-

Testcase

Test Result

Accepted

Runtime: 0 ms

Case 1

Case 2

Case 3

Input

n =  
11

Output

3

Expected

3

Testcase

Test Result

Accepted

Runtime: 0 ms

Case 1

Case 2

Case 3

Input

n =  
128

Output

1

Expected

1

Testcase

Test Result

Accepted

Runtime: 0 ms

Case 1

Case 2

Case 3

Input

n =  
2147483645

Output

30

Expected

30

## PROBLEM-5

AIM:-

Search 2d matrix 2

CODE:-

```
class Solution {  
  
    public boolean searchMatrix(int[][] matrix, int target) {  
  
        int row = 0;  
  
        int column = matrix[0].length-1;  
  
        while(row<matrix.length&&column>=0){  
  
            if(matrix[row][column]==target){  
  
                return true;  
  
            }else if(matrix[row][column]<target){  
  
                row++;  
  
            }else{  
  
                column--;  
  
            }  
  
        }  
  
        return false;  
  
    }  
  
}
```

OUTPUT:-

Testcase

>\_ Test Result

Accepted

Runtime: 0 ms

• Case 1

• Case 2

Input

matrix =  
[[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]]

target =  
5

Output

true

Expected

true

Testcase

>\_ Test Result

Accepted

Runtime: 0 ms

• Case 1

• Case 2

Input

matrix =  
[[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]]

target =  
20

Output

false

Expected

false

PROBLEM-6

AIM:-

Super Pow

CODE:-

```
class Solution {  
  
    public int superPow(int a, int[] b) {  
  
        int num=0;  
  
        for(int i:b){  
  
            num=(num*10+i)%1140;  
  
        }  
  
        return binexpo(a,num,1337);  
    }  
  
    public int binexpo(int a, int b, int m){  
  
        a%=m;  
  
        int res=1;  
  
        while(b>0){  
  
            if((b&1)==1)  
  
                res=(res*a)%m;  
  
            a=(a*a)%m;  
  
            b>>=1;  
  
        }  
  
        return res;  
    }  
}
```

OUTPUT:-

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

a =  
2

b =  
[3]

Output

8

Expected

8

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

a =  
2

b =  
[1,0]

Output

1024

Expected

1024

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

a =  
1

b =  
[4,3,3,8,5,2]

Output

1

Expected

1

## PROBLEM-7

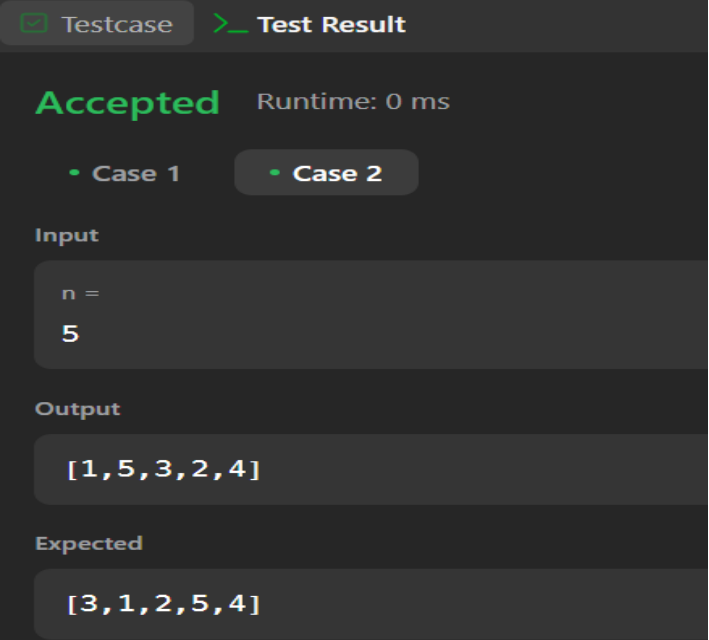
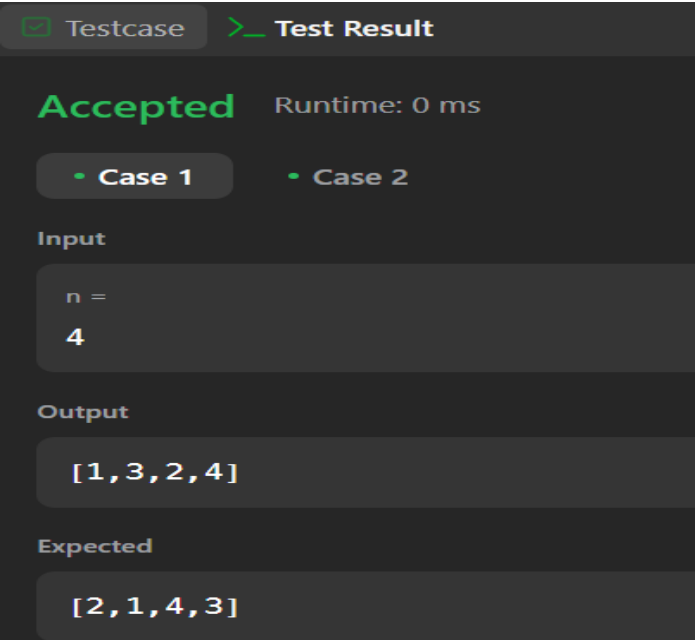
**AIM:-**

Beautiful Array

**CODE:-**

```
class Solution {  
    public int[] beautifulArray(int n) {  
        int[] ans = new int[n];  
        for(int i = 0; i < n; i++){  
            ans[i] = i+1;  
        }  
        recursion(ans, 0, n-1);  
        return ans;  
    }  
    public void recursion(int[] arr, int left, int right){  
        if(left >= right)  
            return;  
        ArrayList<Integer> l = new ArrayList<>();  
        ArrayList<Integer> r = new ArrayList<>();  
        boolean alt = true;  
        for(int i = left; i <= right; i++){  
            if(alt)  
                l.add(arr[i]);  
            else  
                r.add(arr[i]);  
            alt = !alt;  
        }  
        for(int i = left; i <= right; i++){  
            if(!l.isEmpty())  
                arr[i] = l.remove(0);  
            else  
                arr[i] = r.remove(0);  
        }  
        recursion(arr, left, (right+left)/2);  
        recursion(arr, (left+right)/2+1, right);  
    }  
}
```

OUTPUT:-



PROBLEM-8

AIM:-

The Skyline Problem

CODE:-

```
class Solution {
    public class Pair{
        int x,height;
        Pair(int x,int height){
            this.x = x;
            this.height = height;
        }
    }
    class SortbyPoints implements Comparator<Pair>{
        public int compare(Pair a, Pair b){
            if(a.x == b.x) return a.height - b.height;
            return a.x-b.x;
        }
    }
    public List<List<Integer>> getSkyline(int[][] buildings) {
        List<List<Integer>> res = new ArrayList<>();
        List<Pair> list = new ArrayList<>();
        for(int[] arr:buildings){
            list.add(new Pair(arr[0],-arr[2]));
            list.add(new Pair(arr[1],arr[2]));
        }
        Collections.sort(list,new SortbyPoints());
        PriorityQueue<Integer> q = new PriorityQueue<>(Collections.reverseOrder());
        int cur = 0;
        q.add(cur);
```



```

for(int i=0;i<list.size();i++){

    int pos = list.get(i).x,h = list.get(i).height;

    if(h < 0) q.add(-h);

    else q.remove(h);

    if(cur != q.peek()){

        res.add(Arrays.asList(pos,q.peek()));

        cur = q.peek();

    }

}

return res;


}

}

```

**OUTPUT:-**

☒ Testcase
 

 Test Result

Accepted

Runtime: 1 ms

• Case 1

• Case 2

Input

buildings =  
 [ [ 2 , 9 , 10 ] , [ 3 , 7 , 15 ] , [ 5 , 12 , 12 ] , [ 15 , 20 , 10 ] , [ 19 , 24 , 8 ] ]


Output

[ [ 2 , 10 ] , [ 3 , 15 ] , [ 7 , 12 ] , [ 12 , 0 ] , [ 15 , 10 ] , [ 20 , 8 ] , [ 24 , 0 ] ]

Expected

[ [ 2 , 10 ] , [ 3 , 15 ] , [ 7 , 12 ] , [ 12 , 0 ] , [ 15 , 10 ] , [ 20 , 8 ] , [ 24 , 0 ] ]

☒ Testcase
 |
 

 Test Result

Accepted

Runtime: 1 ms

• Case 1

• Case 2

Input

buildings =  
 [ [ 0 , 2 , 3 ] , [ 2 , 5 , 3 ] ]

Output

[ [ 0 , 3 ] , [ 5 , 0 ] ]

Expected

[ [ 0 , 3 ] , [ 5 , 0 ] ]

## PROBLEM-9

**AIM:-**

Reverse Pairs

**CODE:-**

```
class Solution {
    static int count;

    public static void sortMerge(int[] arr){
        int n=arr.length;
        if (n<=1) return ;
        int[] rightSortArray=new int[n/2];
        int[] leftSortArray=new int[n-n/2];
        int x=0;
        for (int i = 0; i < rightSortArray.length; i++) {
            rightSortArray[i]=arr[x];
            x++;
        }
        for (int i = 0; i < leftSortArray.length; i++) {
            leftSortArray[i]=arr[x];
            x++;
        }
        sortMerge(rightSortArray);
        sortMerge(leftSortArray);
        int a=0,b=0;
        while (a<rightSortArray.length && b<leftSortArray.length) {
            if ((long)rightSortArray[a]>2*(long)leftSortArray[b]){
                count+=rightSortArray.length-a;
                b++;
            }else a++;
        }
        int i=0,j=0,k=0;
        while (i<rightSortArray.length && j<leftSortArray.length) {
            if (rightSortArray[i]<leftSortArray[j]) {
                arr[k]=rightSortArray[i];
                i++;
            }else if (rightSortArray[i]>leftSortArray[j]) {
                arr[k]=leftSortArray[j];
                j++;
            }else{
                arr[k]=rightSortArray[i];
                i++;
                k++;
                arr[k]=leftSortArray[j];
            }
        }
    }
}
```

```

        j++;
    }
    k++;
}
if (i==rightSortArray.length) {
    while(j<leftSortArray.length){
        arr[k]=leftSortArray[j];
        j++;
        k++;
    }
}
if (j==leftSortArray.length) {
    while(i<rightSortArray.length){
        arr[k]=rightSortArray[i];
        k++;
        i++;
    }
}

rightSortArray=null;
leftSortArray=null;

}

public int reversePairs(int[] nums) {
    count=0;
    sortMerge(nums);
    return count;
}
}

```

## OUTPUT:-

☒ Testcase
 |
 [Test Result](#)

**Accepted**
Runtime: 0 ms

• Case 1
• Case 2

**Input**

nums =  
 [1,3,2,3,1]

**Output**

2

**Expected**

2

☒ Testcase
 |
 [Test Result](#)

**Accepted**
Runtime: 0 ms

• Case 1
• Case 2

**Input**

nums =  
 [2,4,3,5,1]

**Output**

3

**Expected**

3

PROBLEM-10

AIM:-

Longest increasing subsequence 2

CODE:-

```
class Solution {  
  
    public int lengthOfLIS(int[] nums, int k) {  
  
        int[] temp = new int[nums.length];  
  
        int ans = 1;  
  
        Arrays.fill(temp, 1);  
  
        for(int i = 1; i < nums.length; i++) {  
  
            for(int j = 0; j < i; j++) {  
  
                if(nums[i] > nums[j] && nums[i] - nums[j] <= k) {  
  
                    temp[i] = Math.max(temp[i], temp[j] + 1);  
  
                    ans = Math.max(temp[i], ans);  
  
                }  
  
            }  
  
        }  
  
        return ans;  
  
    }  
  
}
```

OUTPUT:-

Testcase

Test Result

Accepted

Runtime: 0 ms

Case 1

Case 2

Case 3

Input

nums =  
[4,2,1,4,3,4,5,8,15]

k =  
3

Output

5

Expected

5

Testcase

Test Result

Accepted

Runtime: 0 ms

Case 1

Case 2

Case 3

Input

nums =  
[7,4,5,1,8,12,4,7]

k =  
5

Output

4

Expected

4

Testcase

Test Result

Accepted

Runtime: 0 ms

Case 1

Case 2

Case 3

Input

nums =  
[1,5]

k =  
1

Output

1

Expected

1