AP-Experiment-4

Name :-Rahul Kumar

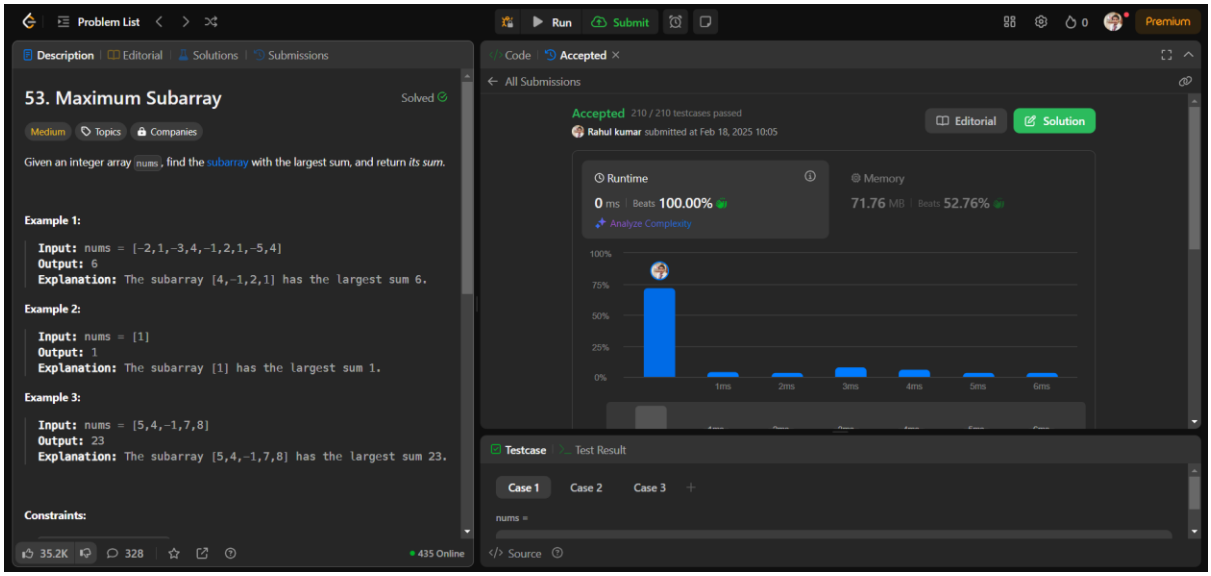
Uid :- 22BCS50181

Sec-FL_IOT-603-A

53. Maximum Subarray

Code:-

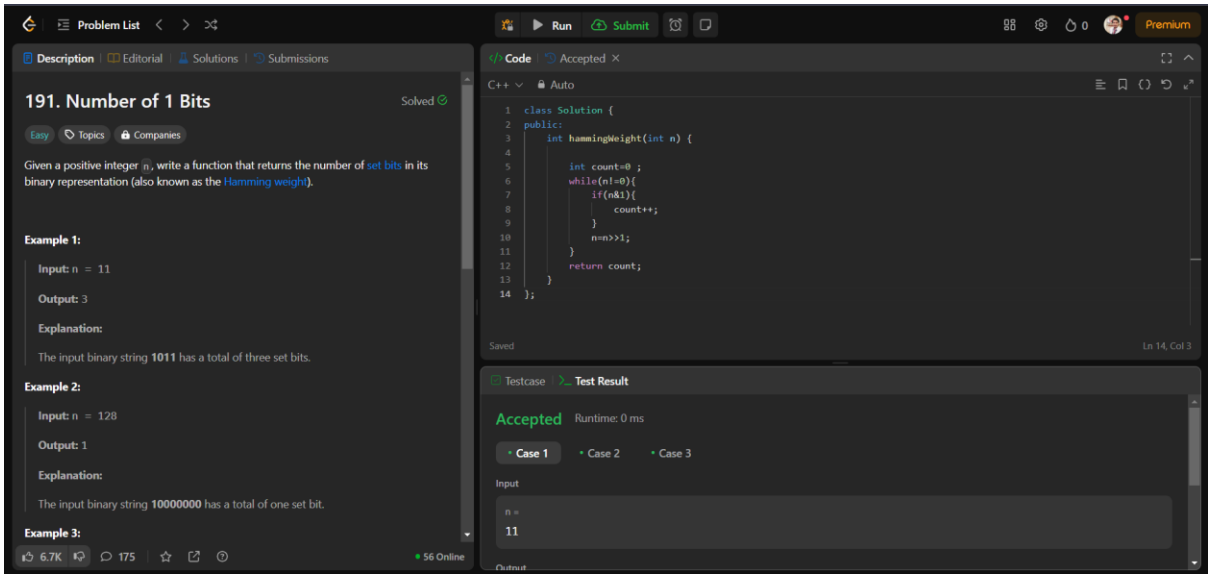
```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int maxi = INT_MIN;
        int currSum = 0;
        for (int i = 0; i < nums.size(); i++) {
            currSum += nums[i];
          if (currSum > maxi) //if current sum is greater than maximum sum then     maxi=currSum
                maxi = currSum;
           if (currSum < 0)//if currentsum become -ve reset it to 0
                currSum = 0;
        }
        return maxi;
    }
};
```



[191. Number of 1 Bits]

Code:-

```
class Solution {
public:
    int hammingWeight(int n) {
        int count=0 ;
        while(n!=0){
            if(n&1){
                count++;
            }
            n=n>>1;
        }
        return count;
    }
};
```

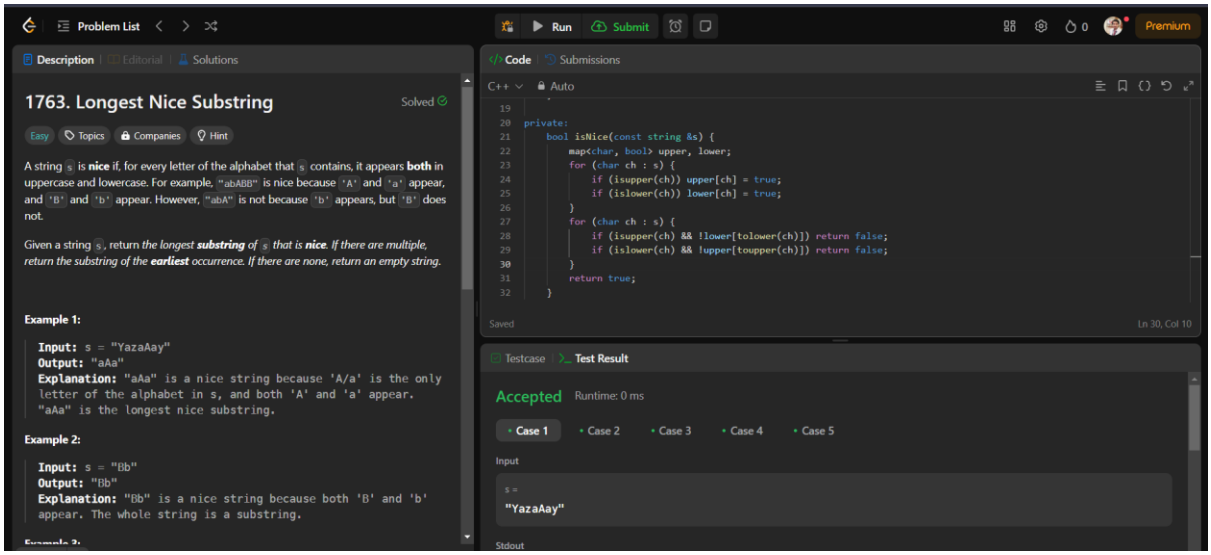


[1763. Longest Nice Substring]

Code:-

```
class Solution {
public:
    string longestNiceSubstring(string s) {
        int n = s.length();
        if (n < 2) return "";
        string result = "";
        for (int i = 0; i < n; ++i) {
            for (int j = i; j < n; ++j) {
                string sub = s.substr(i, j - i + 1);
                if (isNice(sub) && sub.length() > result.length()) {
                    result = sub  }}}
        return result;
    }
private:
```

```
    bool isNice(const string &s) {
        map<char, bool> upper, lower;
        for (char ch : s) {
            if (isupper(ch)) upper[ch] = true;
            if (islower(ch)) lower[ch] = true;
        }
        for (char ch : s) {
            if (isupper(ch) && !lower[tolower(ch)]) return false;
            if (islower(ch) && !upper[toupper(ch)]) return false;
        }
        return true;
    }
};
```

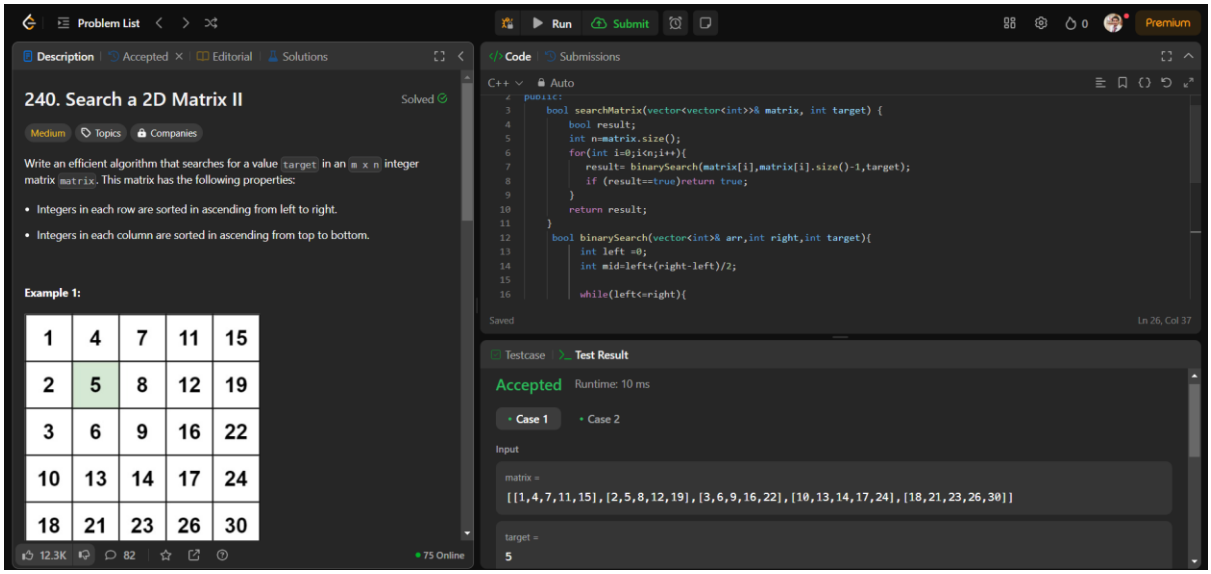


[240. Search a 2D Matrix II]

Code:-

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        bool result;
```

```
        int n=matrix.size();
        for(int i=0;i<n;i++){
          result= binarySearch(matrix[i],matrix[i].size()-1,target);
          if (result==true)return true;
        }
        return result;
    }
     bool binarySearch(vector<int>& arr,int right,int target){
         int left =0;
         int mid=left+(right-left)/2;
         while(left<=right){
          if(arr[mid]==target)return true;
          if(arr[mid]<target){
              left=mid+1;
          }
          else{
              right=mid-1;
          }
          mid=left+(right-left)/2;
        }
        return false;
     }
};
```


Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
240. Search a 2D Matrix II
Solved
Medium
Topics
Companies
Write an efficient algorithm that searches for a value target in an m x n integer matrix matrix. This matrix has the following properties:
Integers in each row are sorted in ascending from left to right.
Integers in each column are sorted in ascending from top to bottom.
Example 1:
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
12.3K
82
75 Online
Code
Submissions
C++
Auto
public:
bool searchMatrix(vector<vector<int>>& matrix, int target) {
bool result;
int n=matrix.size();
for(int i=0;i<n;i++){
result= binarySearch(matrix[i],matrix[i].size()-1,target);
if (result==true)return true;
}
return result;
}
bool binarySearch(vector<int>& arr,int right,int target){
int left =0;
int mid=left+(right-left)/2;
while(left<=right){
Saved
Ln 26, Col 37
Testcase
Test Result
Accepted
Runtime: 10 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