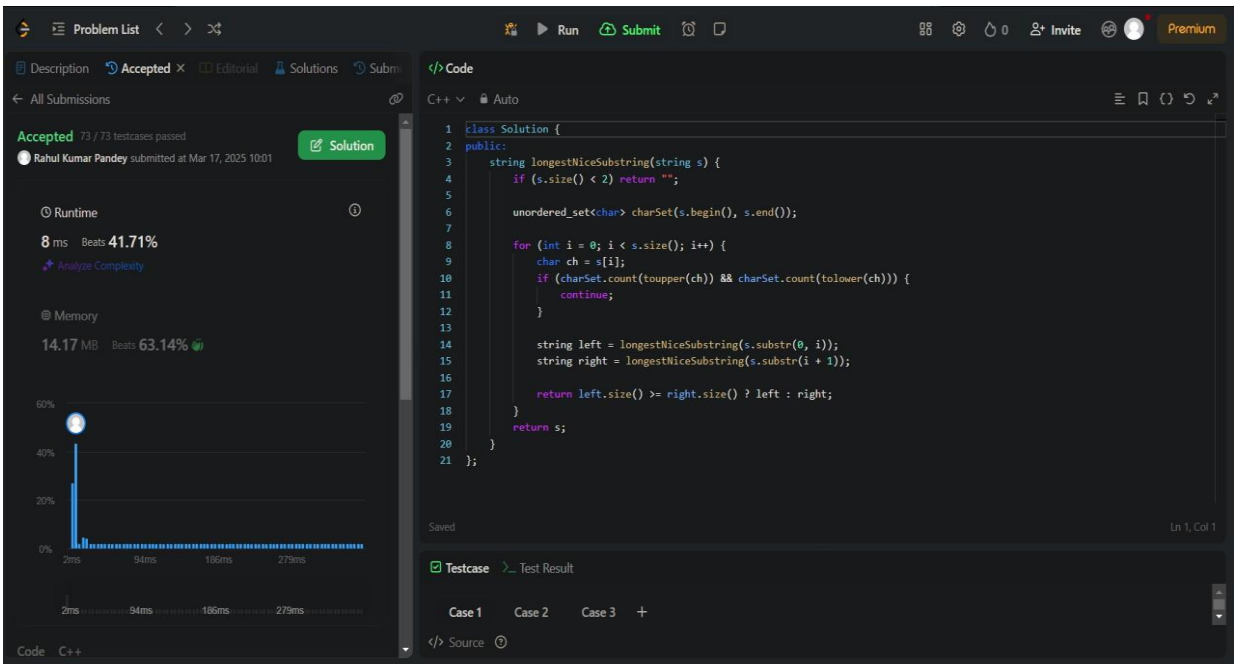
ASSIGNMENT 4

Name- Pranav kumar Singh
UID- 22BCS12894
Section-22BCS_IOT-606-B

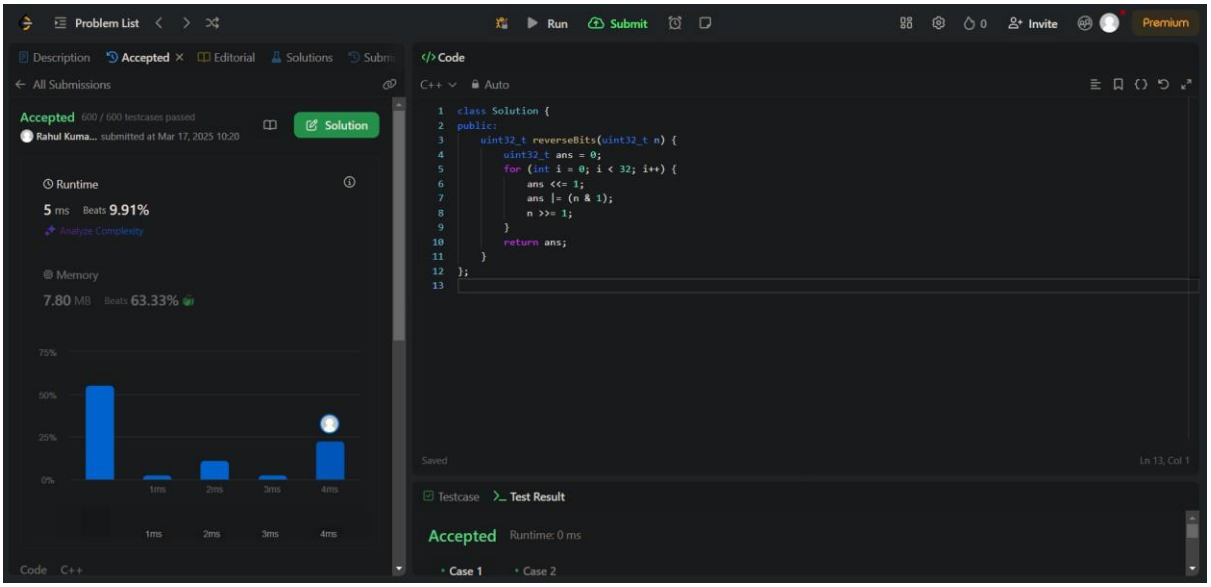
1) Longest Nice Substring

```
class Solution {
public:
  string longestNiceSubstring(string s) {
    if (s.size() < 2) return "";
    unordered_set<char> charSet(s.begin(), s.end());
    for (int i = 0; i < s.size(); i++) {
      if (charSet.count(tolower(s[i])) && charSet.count(toupper(s[i]))) continue;
      string left = longestNiceSubstring(s.substr(0, i));
      string right = longestNiceSubstring(s.substr(i + 1));
      return left.size() >= right.size() ? left : right;
    }
    return s;    }};
```



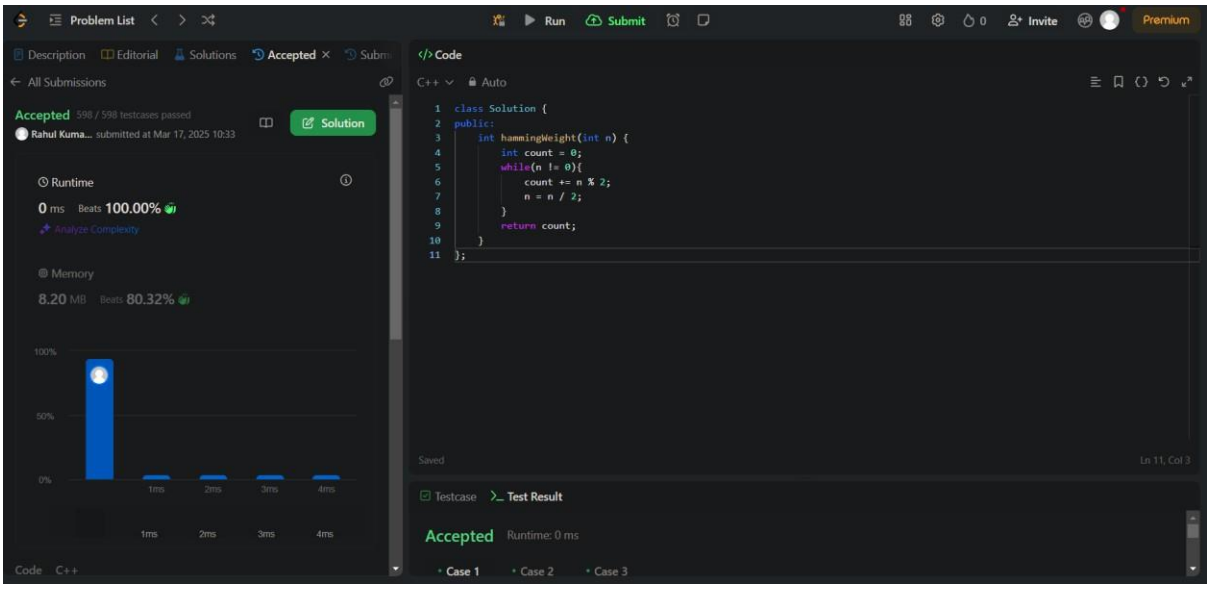
2) **Reverse Bits**

```
class Solution {
public:
    uint32_t reverseBits(uint32_t n) {
        uint32_t res = 0;
        for (int i = 0; i < 32; i++) {
            res = (res << 1) | (n & 1);
            n >>= 1;
        }
        return res;
    }
};
```



3) **Number of 1 Bits**

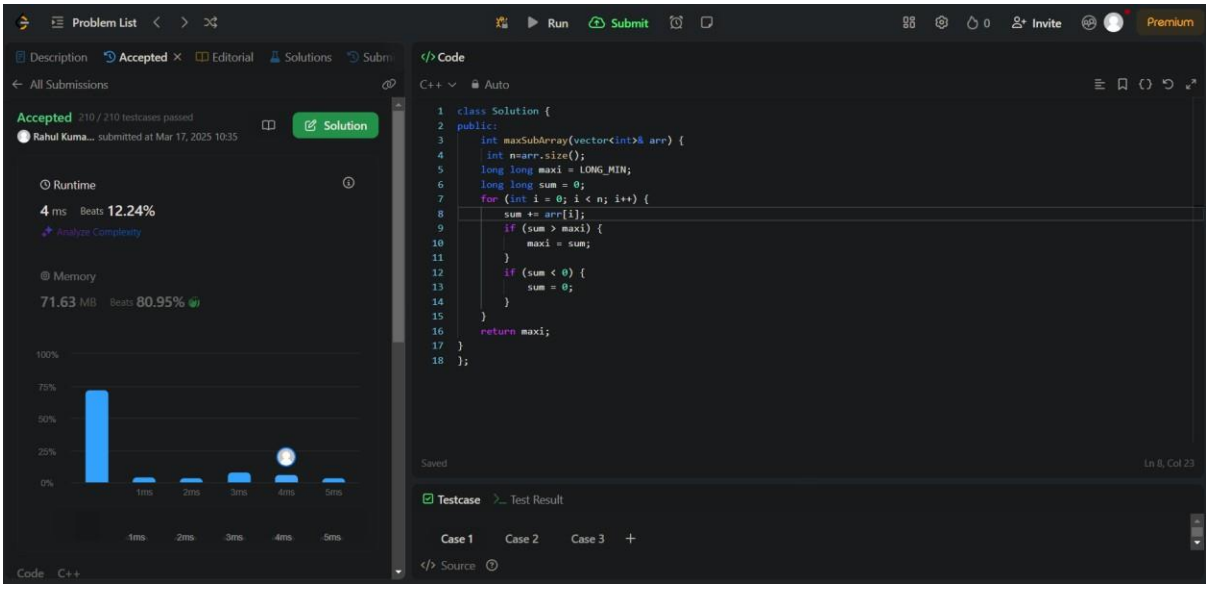
```
class Solution {
public:
    int hammingWeight(uint32_t n) {
        int count = 0;
        while (n) {
            count += n & 1;
            n >>= 1;
        }
        return count;
    }
};
```



4) **Maximum Subarray**

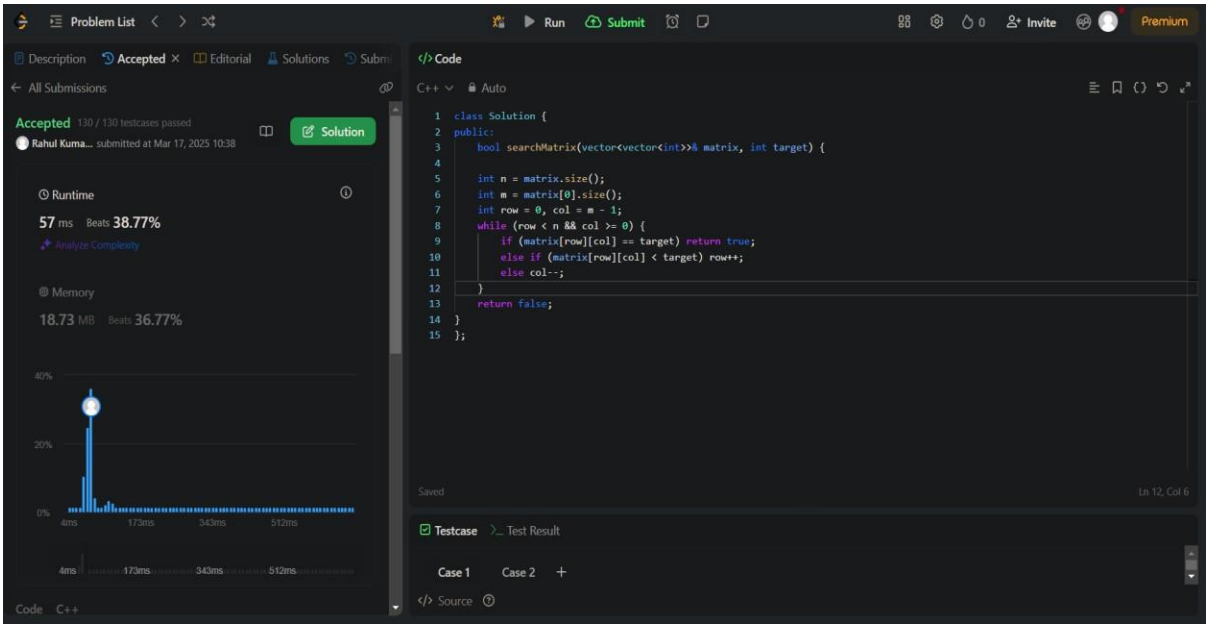
```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int max_sum = nums[0], curr_sum = 0;
        for (int num : nums) {
            if (curr_sum < 0) curr_sum = 0;
            curr_sum += num;
            max_sum = max(max_sum, curr_sum);
        }
        return max_sum;
```

```
    }
};
```


Problem List
Run
Submit
Invite
Premium
Description
Accepted
Editorial
Solutions
All Submissions
Accepted 210 / 210 testcases passed
Rahul Kuma... submitted at Mar 17, 2025 10:35
Solution
Runtime
4 ms Beats 12.24%
Analyze Complexity
Memory
71.63 MB Beats 80.95%
Code
C++
Auto
class Solution {
public:
int maxSubArray(vector<int>& arr) {
int n=arr.size();
long long maxi = LONG_MIN;
long long sum = 0;
for (int i = 0; i < n; i++) {
sum += arr[i];
if (sum > maxi) {
maxi = sum;
}
if (sum < 0) {
sum = 0;
}
}
return maxi;
}
};
Saved
Ln 8, Col 23
Testcase
Test Result
Case 1
Case 2
Case 3
Source

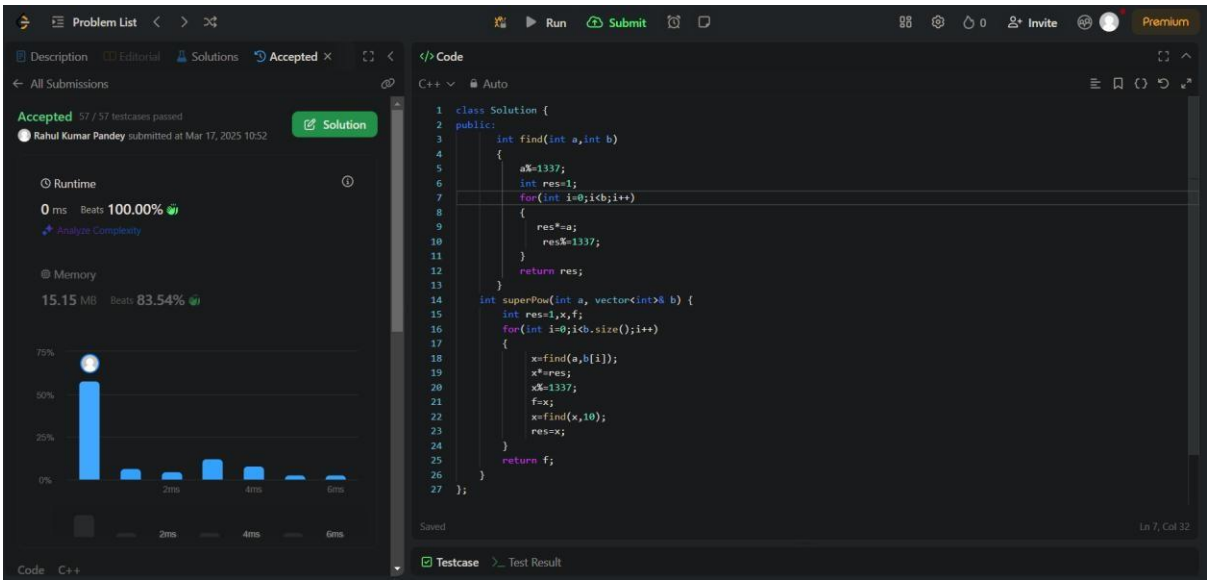
5) Search a 2D Matrix II

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



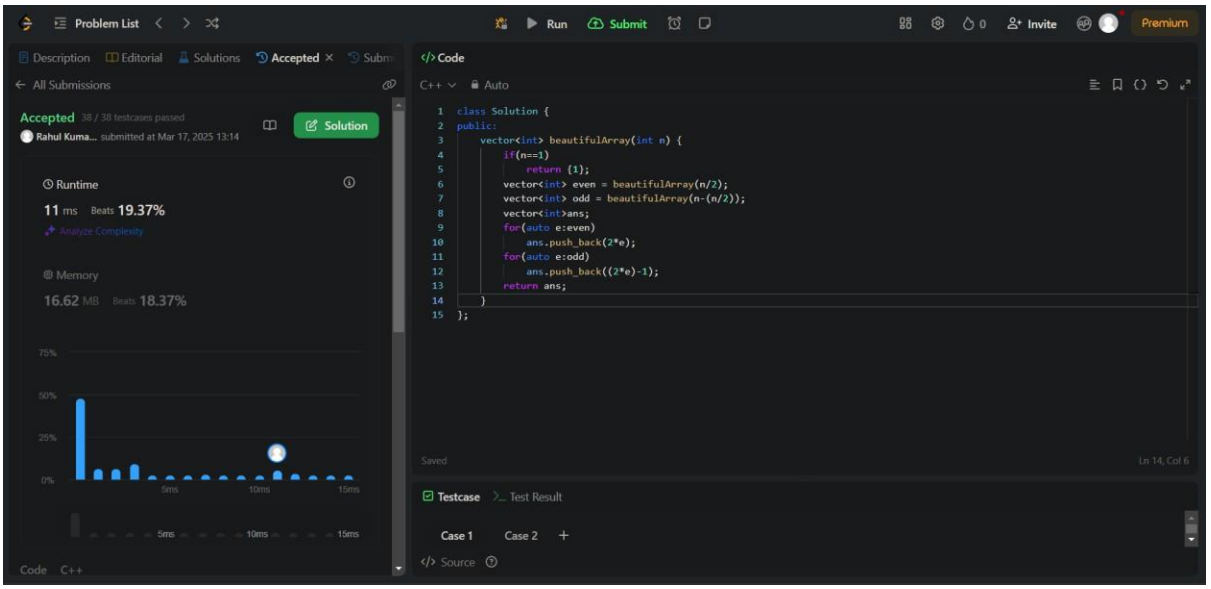
6) Super Pow

```
class Solution {
public:
    const int MOD = 1337;

    int power(int x, int y) {
        int res = 1;
        x %= MOD;
        while (y) {
            if (y % 2) res = (res * x) % MOD;
            x = (x * x) % MOD;
            y /= 2;
        }
        return res;
    }

    int superPow(int a, vector<int>& b) {
        int result = 1;
        for (int digit : b) {
            result = power(result, 10) * power(a, digit) % MOD;
        }
        return result;
    }
};
```



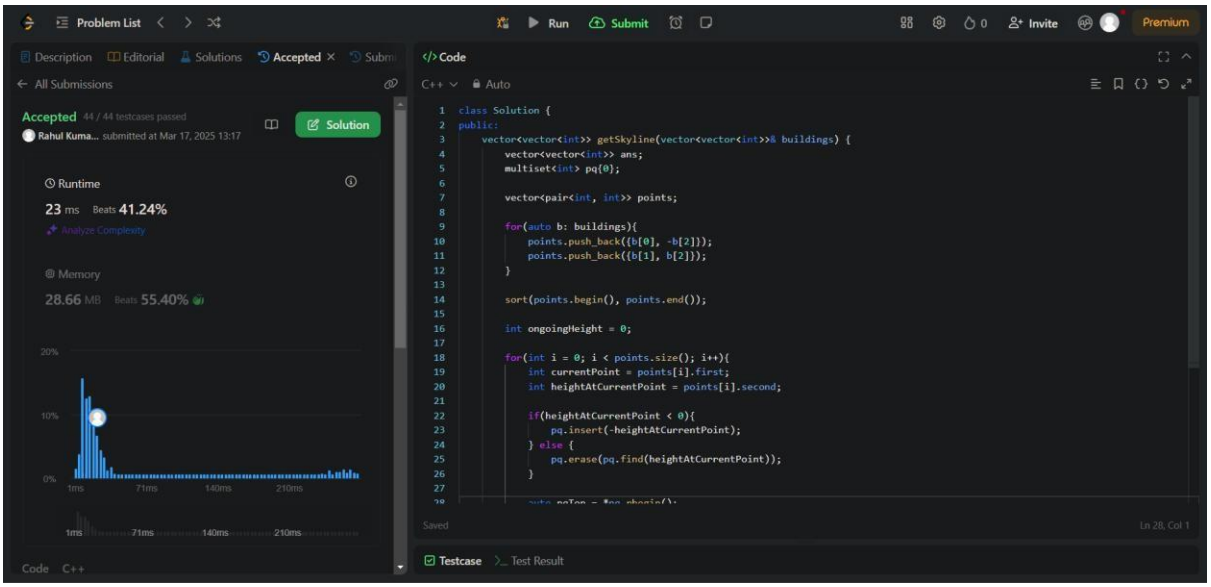
7) Beautiful Array

```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        vector<int> arr = {1};
        while (arr.size() < n) {
            vector<int> temp;
            for (int x : arr) if (x * 2 - 1 <= n) temp.push_back(x * 2 - 1);
            for (int x : arr) if (x * 2 <= n) temp.push_back(x * 2);
            arr = temp;
        }
        return arr;
    }
};
```



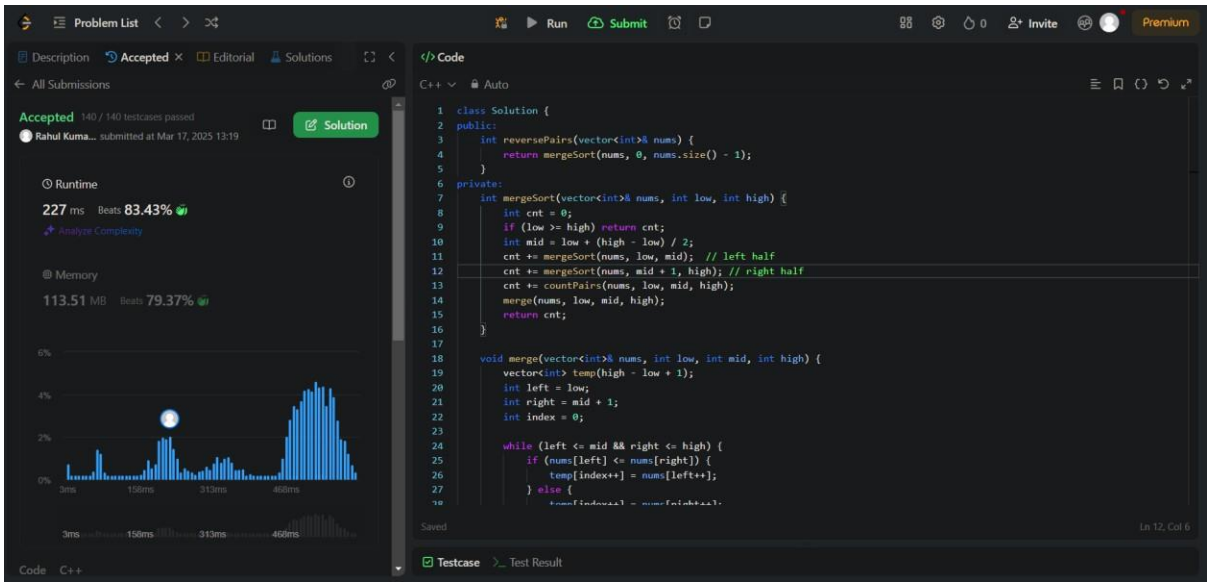
8) The Skyline Problem

```
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<pair<int, int>> events;
        for (auto& b : buildings) {
            events.emplace_back(b[0], -b[2]);
            events.emplace_back(b[1], b[2]);
        }
        sort(events.begin(), events.end());

        multiset<int> heights = {0};
        vector<vector<int>> res;
        int prev = 0;

        for (auto [x, h] : events) {
            if (h < 0) heights.insert(-h);
            else heights.erase(heights.find(h));

            int curr = *heights.rbegin();
            if (curr != prev) {
                res.push_back({x, curr});
                prev = curr;
            }
        }
        return res;
    }
};
```



9)Reverse Pairs

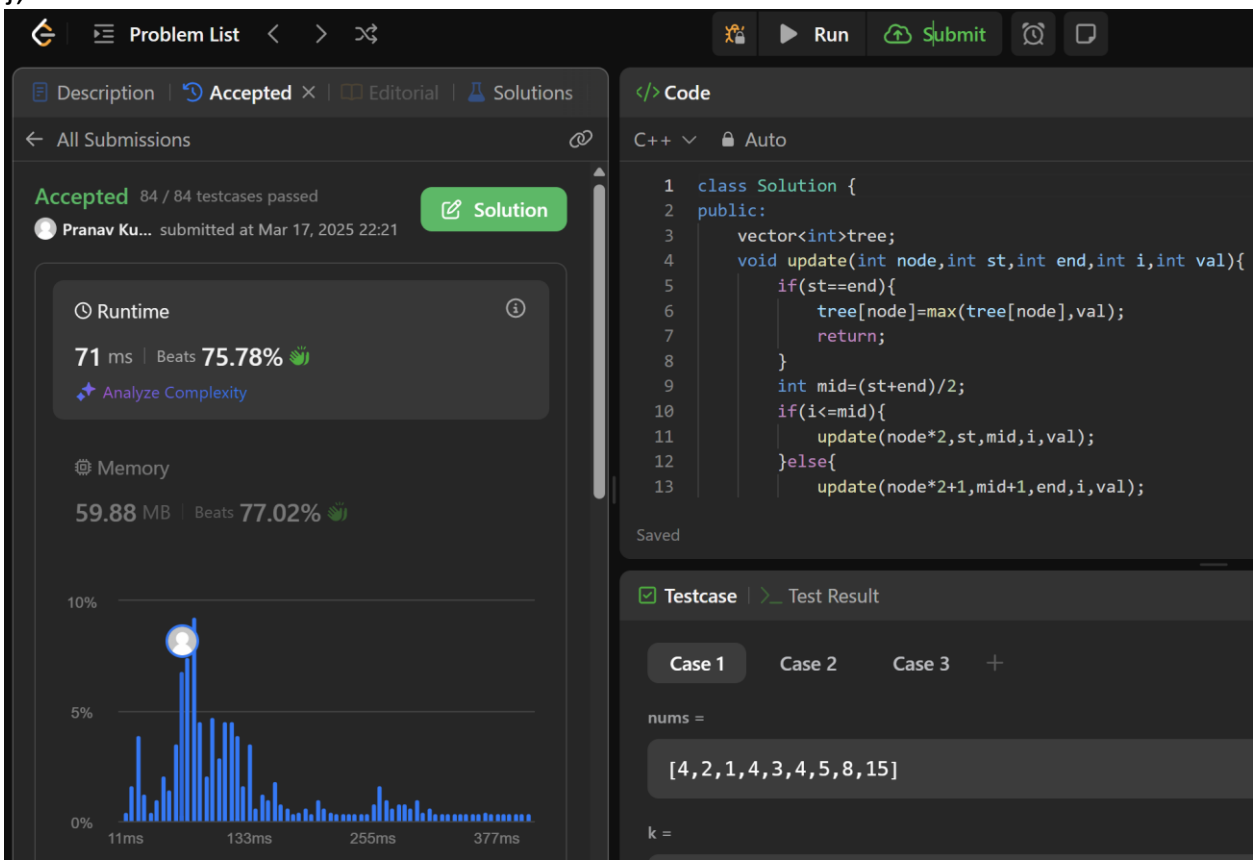
```
class Solution {
public:
    int mergeAndCount(vector<int>& nums, int left, int mid, int right) {
        int count = 0, j = mid + 1;
        for (int i = left; i <= mid; i++) {
            while (j <= right && nums[i] > 2LL * nums[j]) j++;
            count += (j - (mid + 1));
        }
        sort(nums.begin() + left, nums.begin() + right + 1);
        return count;
    }

    int mergeSort(vector<int>& nums, int left, int right) {
        if (left >= right) return 0;
        int mid = (left + right) / 2;
        int count = mergeSort(nums, left, mid) + mergeSort(nums, mid + 1, right);
        count += mergeAndCount(nums, left, mid, right);
        return count;
    }

    int reversePairs(vector<int>& nums) {
        return mergeSort(nums, 0, nums.size() - 1);
    }
};
```



10)Longest Increasing Subsequence II

```
class Solution {
public:
   vector<int>tree;
   void update(int node,int st,int end,int i,int val){
      if(st==end){
         tree[node]=max(tree[node],val);
         return;
      }
      int mid=(st+end)/2;
      if(i<=mid){
         update(node*2,st,mid,i,val);
      }else{
         update(node*2+1,mid+1,end,i,val);
      }
      tree[node]=max(tree[node*2],tree[node*2+1]);
   }
   int query(int node,int st,int end,int x,int y){
      if(x>end || y<st) return -1e9;
      if(st>=x && end<=y){
         return tree[node];
      }
      int mid=(st+end)/2;
      int left=query(2*node,st,mid,x,y);
      int right=query(2*node+1,mid+1,end,x,y);
      return max(left,right);
   }
   int lengthOfLIS(vector<int>& nums, int k) {
      int n=nums.size();
      if(n==1) return 1;
      int m=*max_element(nums.begin(),nums.end());
      tree.clear();
      tree.resize(4*m+10);
      for(int i=n-1;i>=0;i--){
         int l=nums[i]+1,r=min(nums[i]+k,m);
         int x=query(1,0,m,l,r);
         if(x==-1e9) x=0;
         update(1,0,m,nums[i],x+1);
      }
      return tree[1];
```

```
    }
};
```


Problem List
Run
Submit
Description
Accepted
Editorial
Solutions
All Submissions
Accepted 84 / 84 testcases passed
Pranav Ku... submitted at Mar 17, 2025 22:21
Solution
Runtime
71 ms | Beats 75.78%
Analyze Complexity
Memory
59.88 MB | Beats 77.02%
10%
5%
0%
11ms
133ms
255ms
377ms
Code
C++
Auto
class Solution {
public:
vector<int>tree;
void update(int node,int st,int end,int i,int val){
if(st==end){
tree[node]=max(tree[node],val);
return;
}
int mid=(st+end)/2;
if(i<=mid){
update(node*2,st,mid,i,val);
}else{
update(node*2+1,mid+1,end,i,val);
Saved
Testcase
Test Result
Case 1
Case 2
Case 3
nums =
[4,2,1,4,3,4,5,8,15]
k =