Assignment-4

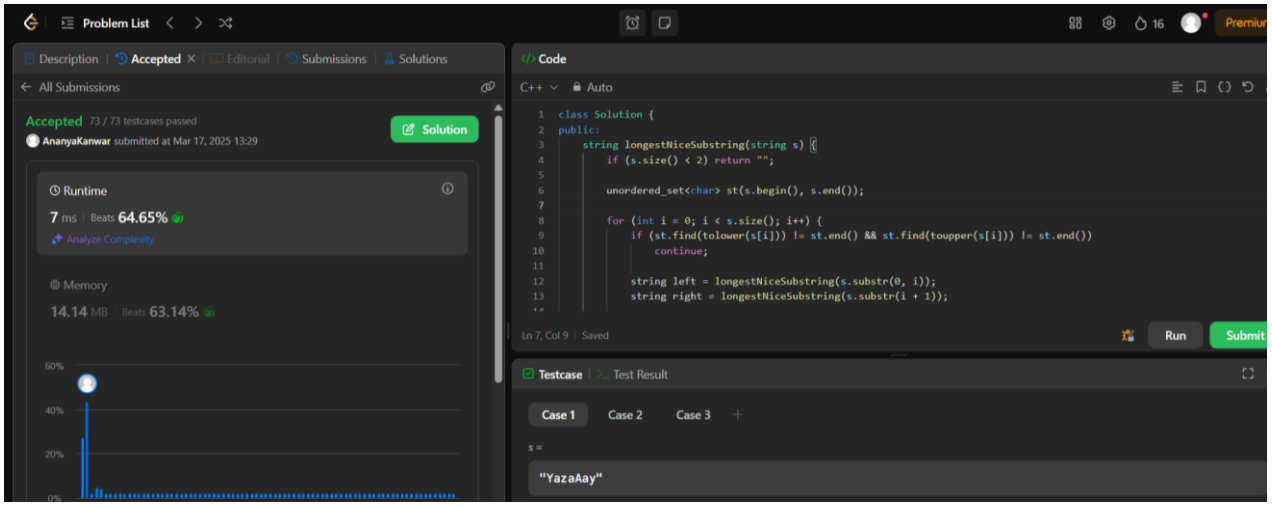
Student Name: Ananya Kanwar **UID: 22BCS10579**
Branch: CSE **Section/Group: IOT-609-B**
Semester: 6th **Date: 17-03-2025**
Subject Name: AP Lab **Subject Code: 22CSP-351**

Ques 1: Longest Nice Substring

Code: -

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

Submission: -

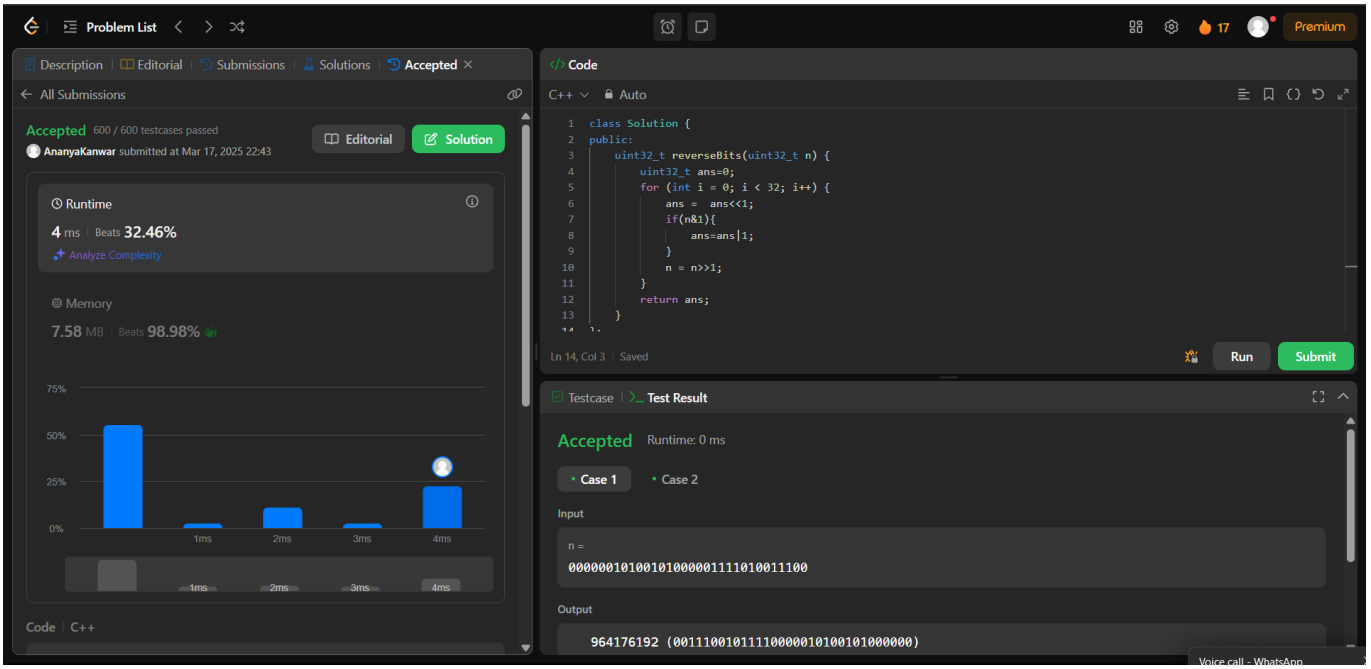


Ques 2: Reverse Bits

Code: -

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

Submission: -

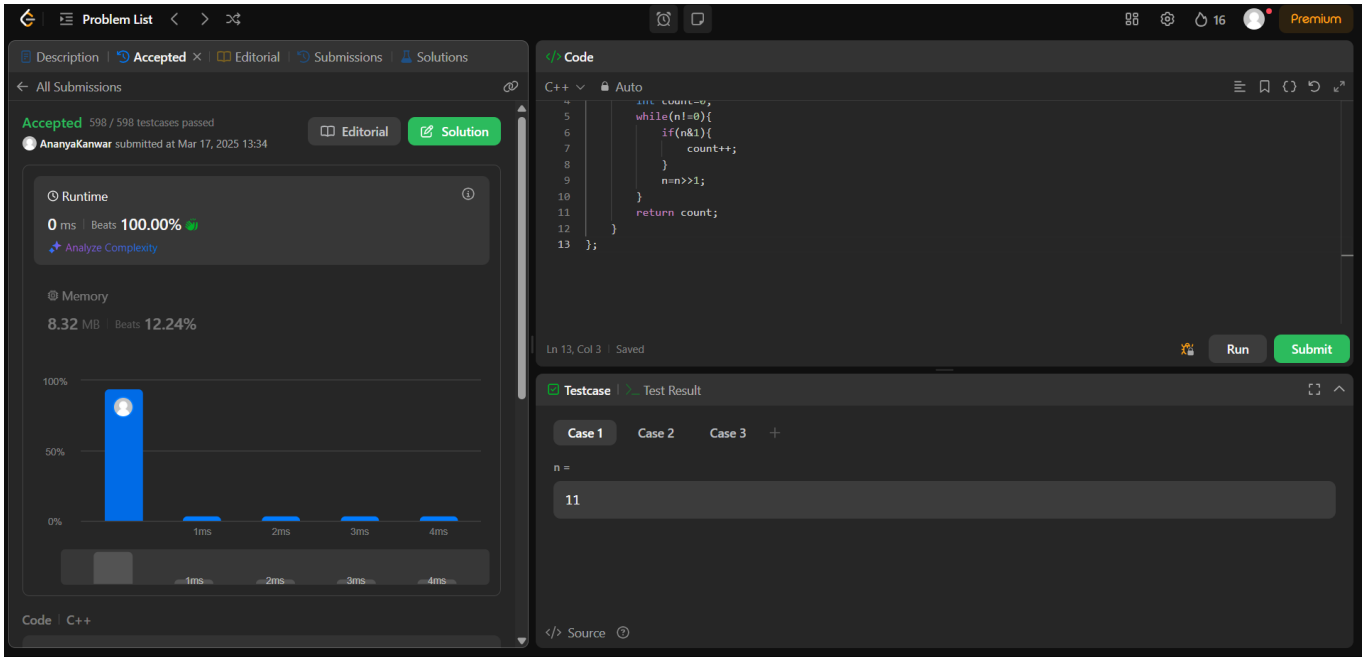


Ques 3: Number of 1 Bits

Code: -

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

Submission: -

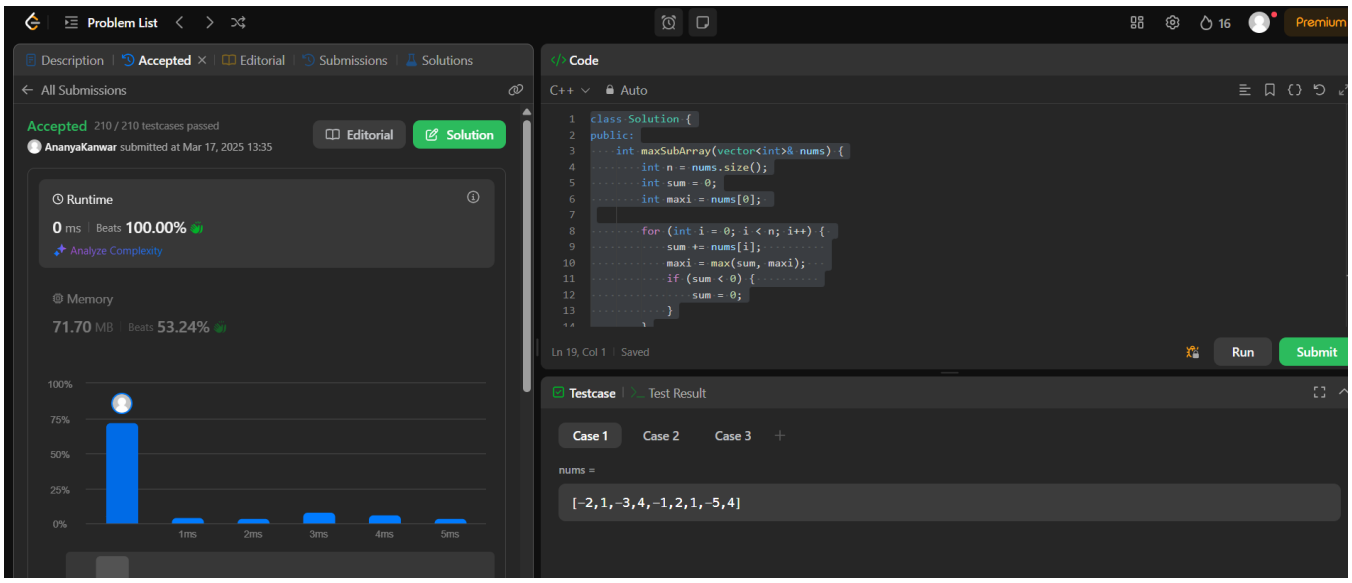


Ques 4: Maximum Subarray

Code: -

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int n = nums.size();
        int sum = 0;
        int maxi = nums[0];

        for (int i = 0; i < n; i++) {
            sum += nums[i];
            maxi = max(sum, maxi);
            if (sum < 0) {
                sum = 0;
            }
        }

        return maxi;
    }
};
```

Submission: -

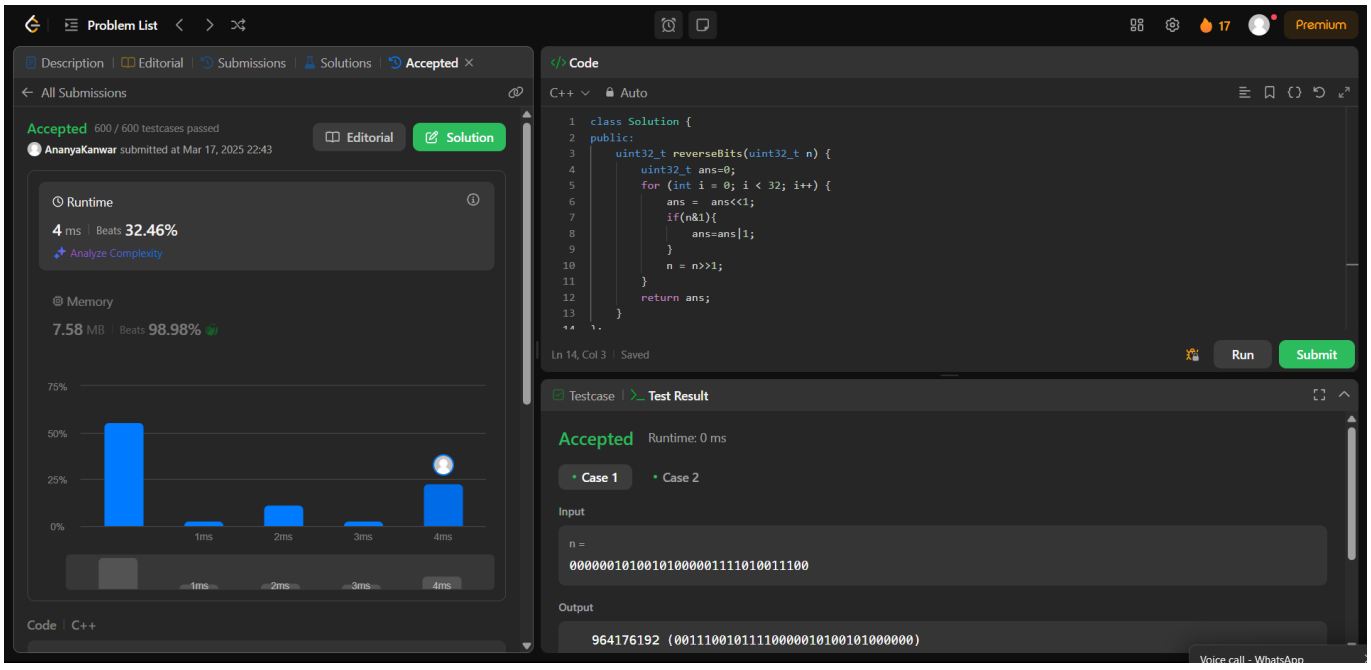


Ques 5: Search a 2D Matrix II

Code: -

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

Submission: -

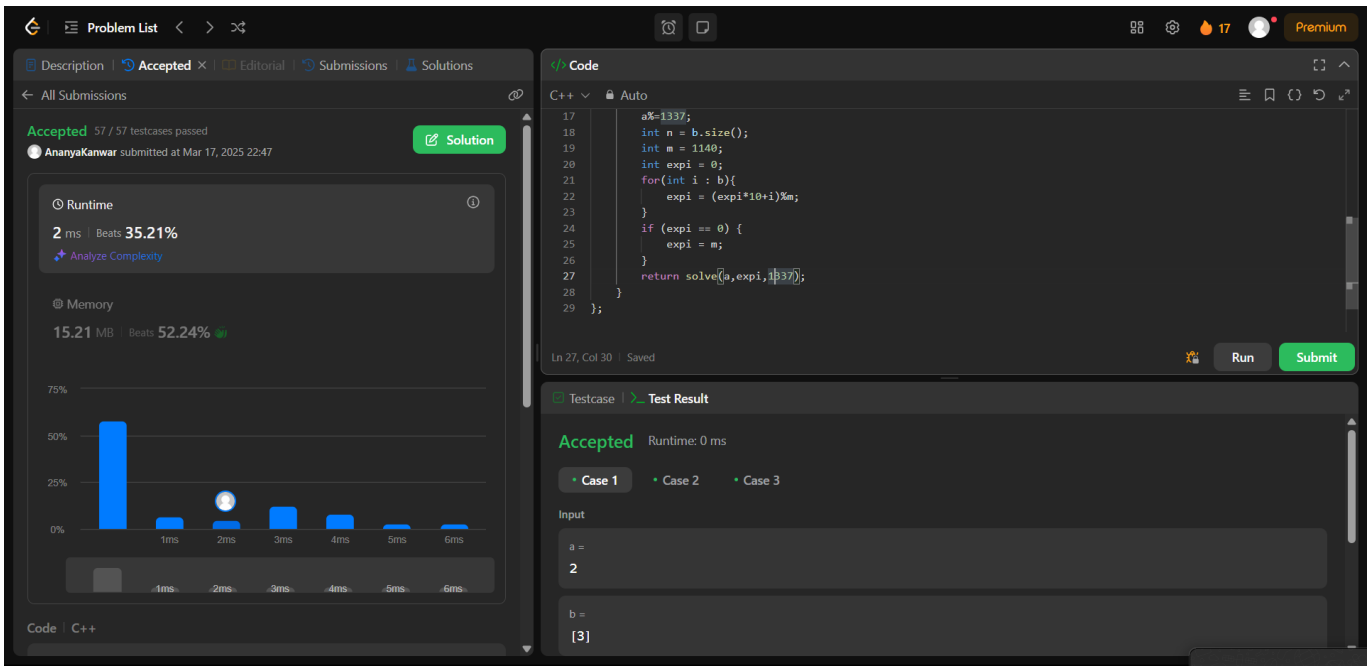


Ques 6: Super Pow

Code: -

```
class Solution {
private:
    int solve(int base, int power, int mod) {
        int ans = 1;
        while (power > 0) {
            if (power & 1) {
                ans = (ans * base) % mod;}
            base = (base * base) % mod;
            power >>= 1;} return ans;}
public:
    int superPow(int a, vector<int>& b) {
        a%=1337;
        int n = b.size();
        int m = 1140;
        int expi = 0;
        for(int i : b){
            expi = (expi*10+i)%m;}
        if (expi == 0) { expi = m; }
        return solve(a,expi,1337); } };
```

Submission: -

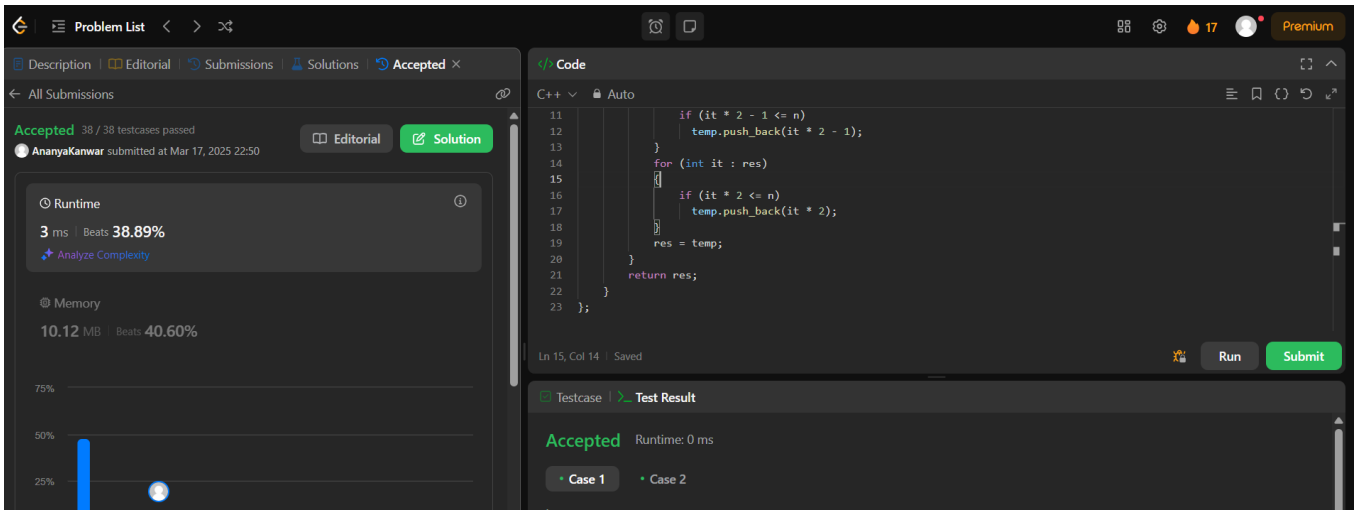


Ques 7: Beautiful Array

Code: -

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

Submission: -

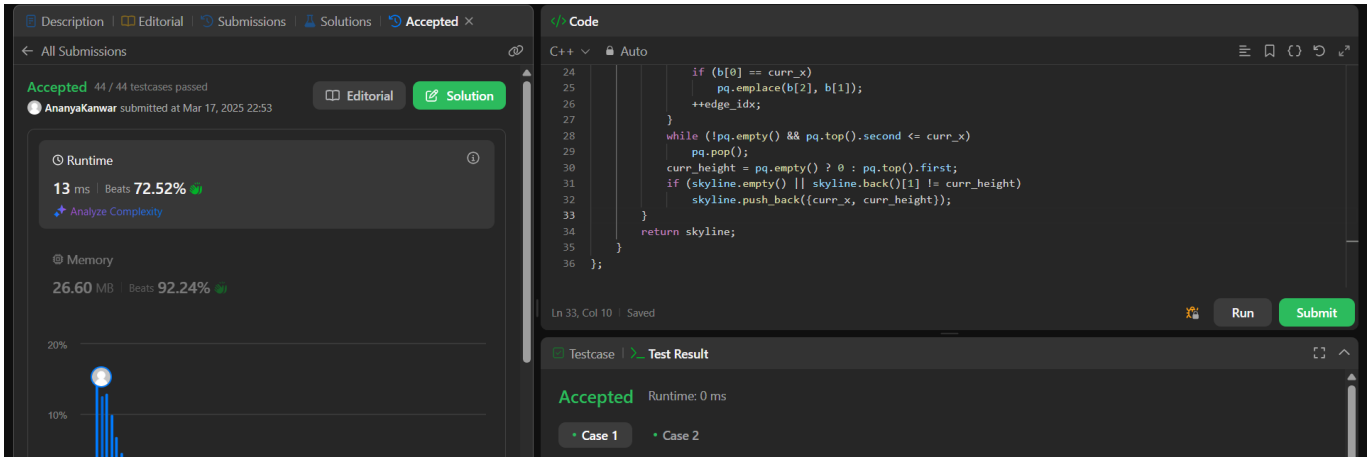


Ques 8: The Skyline Problem

Code: -

```
class Solution {
public:
  vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
    int edge_idx = 0;  vector<pair<int, int>> edges;
    priority_queue<pair<int, int>> pq;
    vector<vector<int>> skyline;
    for (int i = 0; i < buildings.size(); ++i) {
      const auto &b = buildings[i]; edges.emplace_back(b[0], i);
      edges.emplace_back(b[1], i); }
    std::sort(edges.begin(), edges.end());
    while (edge_idx < edges.size()) {
      int curr_height;
      const auto &[curr_x, _] = edges[edge_idx];
      while (edge_idx < edges.size() &&
          curr_x == edges[edge_idx].first) {
        const auto &[_, building_idx] = edges[edge_idx];
        const auto &b = buildings[building_idx];
        if (b[0] == curr_x)
          pq.emplace(b[2], b[1]);
        ++edge_idx; }
      while (!pq.empty() && pq.top().second <= curr_x)
        pq.pop();
      curr_height = pq.empty() ? 0 : pq.top().first;
      if (skyline.empty() || skyline.back()[1] != curr_height)
        skyline.push_back({curr_x, curr_height}); }
    return skyline;
  }
 };
```

Submission: -


Description
Editorial
Submissions
Solutions
Accepted
All Submissions
Accepted 44 / 44 testcases passed
AnanyaKanwar submitted at Mar 17, 2025 22:53
Editorial
Solution
Runtime
13 ms | Beats 72.52%
Analyze Complexity
Memory
26.60 MB | Beats 92.24%
20%
10%
Code
C++
Auto
if (b[0] == curr_x)
pq.emplace(b[2], b[1]);
++edge_idx;
}
while (!pq.empty() && pq.top().second <= curr_x)
pq.pop();
curr_height = pq.empty() ? 0 : pq.top().first;
if (skyline.empty() || skyline.back()[1] != curr_height)
skyline.push_back({curr_x, curr_height});
}
return skyline;
}
};
Ln 33, Col 10
Saved
Run
Submit
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2

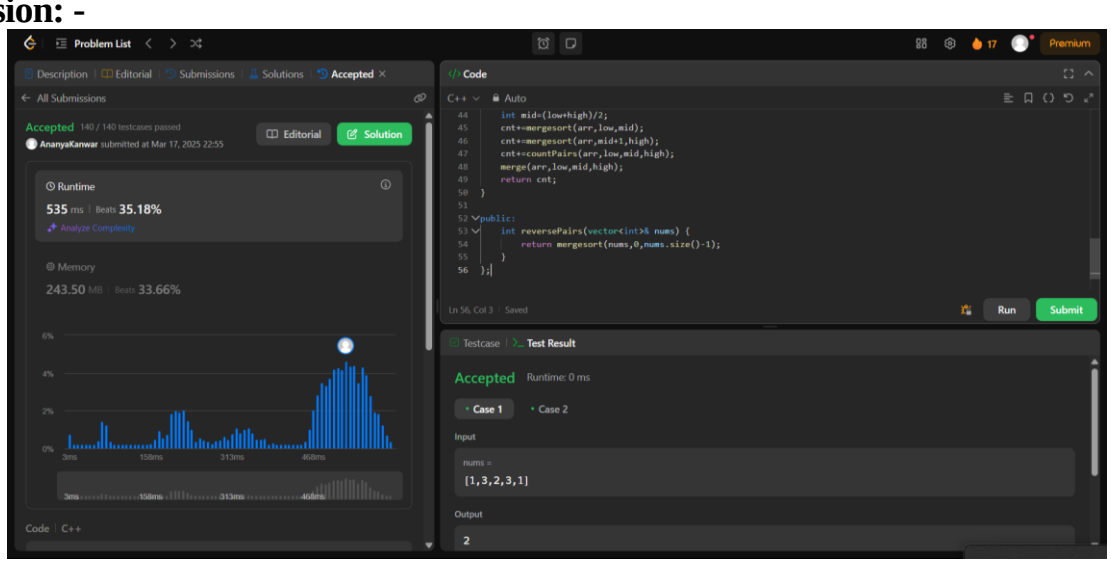
Ques 9: Reverse Pairs

Code: -

```
class Solution {
private:

int countPairs(vector<int>& arr,int low,int mid,int high){
        int cnt=0;
        int right=mid+1;
        for(int i=low;i<=mid;i++){
                while(right<=high && 0.5*arr[i]>arr[right]) right++;
                cnt+=right-(mid+1);
        }
        return cnt;
}

void merge(vector<int>& arr,int low,int mid,int high){
        int left=low;
        int right=mid+1;
        vector<int> temp;
        while(left<=mid && right<=high){
                if(arr[left]<=arr[right]){
                        temp.push_back(arr[left]);
                        left++;
                }
                else{
                        temp.push_back(arr[right]);
                        right++;
                }
        }
        while(left<=mid){
                temp.push_back(arr[left]);
                left++;
        }
        while(right<=high){
                temp.push_back(arr[right]);
```

```
                right++;
            }
            for(int i=low;i<=high;i++){
                arr[i]=temp[i-low];
            }
    }

    int mergesort(vector<int>& arr,int low,int high){
            int cnt=0;
            if(low>=high) return cnt;
            int mid=(low+high)/2;
            cnt+=mergesort(arr,low,mid);
            cnt+=mergesort(arr,mid+1,high);
            cnt+=countPairs(arr,low,mid,high);
            merge(arr,low,mid,high);
            return cnt;
    }

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

Submission: -



Ques 10: Longest Increasing Subsequence II

Code: -

```
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];
    }
};
```

Submission: -