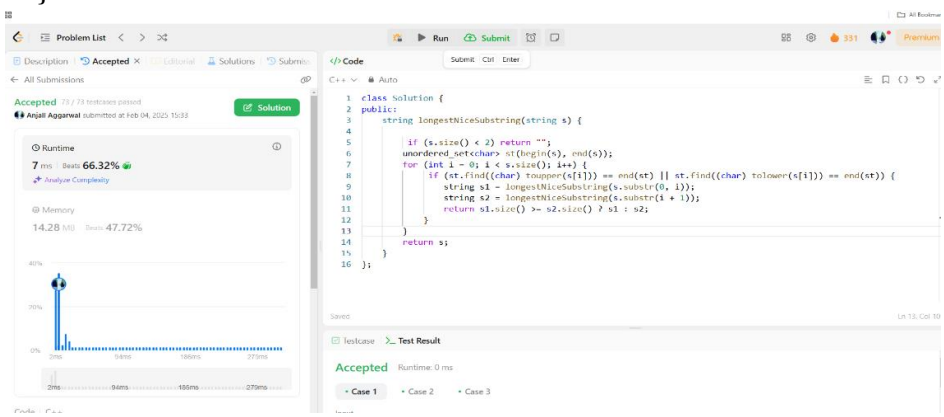
Assignment 1

Student Name: Anjali Aggarwal **UID: 22BCS11246**
Branch: CSE **Section: 22BCS_FL_602-B**
Semester: 6 **Date of Performance:7-1-25**
Subject Name: Advanced Programming **Subject Code:22CSH-35]**

QUESTION

1763. Longest Nice Substring

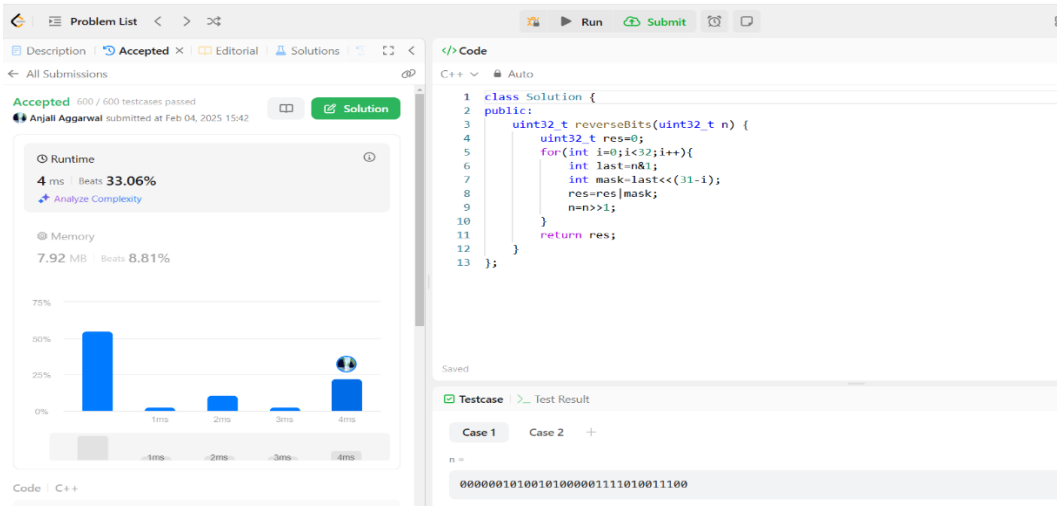
```
string longestNiceSubstring(string s) {

    if (s.size() < 2) return "";
    unordered_set<char> st(begin(s), end(s));
    for (int i = 0; i < s.size(); i++) {
        if (st.find((char) toupper(s[i])) == end(st) || st.find((char) tolower(s[i])) == end(st))
{
            string s1 = longestNiceSubstring(s.substr(0, i));
            string s2 = longestNiceSubstring(s.substr(i + 1));
            return s1.size() >= s2.size() ? s1 : s2;
        }
    }
    return s;
}
```



190.Reverse Bits

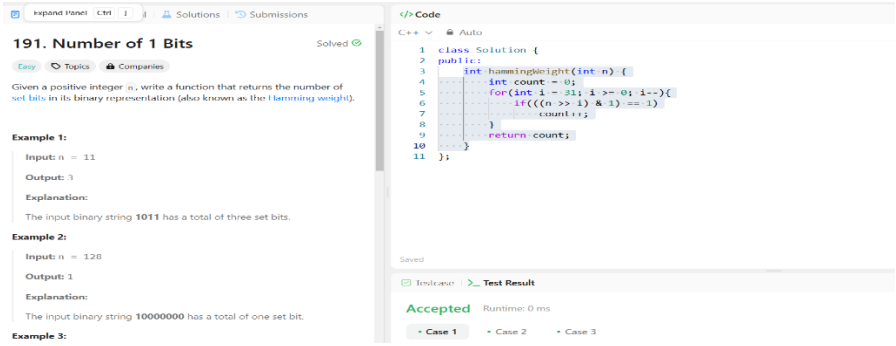
```
uint32_t reverseBits(uint32_t n) {
    uint32_t res=0;
    for(int i=0;i<32;i++){
        int last=n&1;
```

```
        int mask=last<<(31-i);
        res=res|mask;
        n=n>>1;
    }
    return res;
}
```



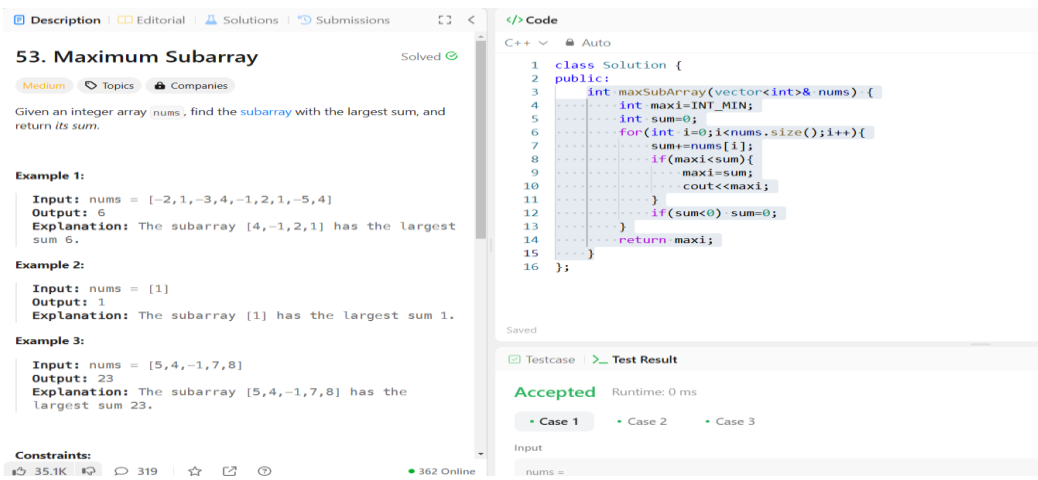
191.[Number of 1 Bits]

```
int hammingWeight(int n) {
    int count = 0;
    for(int i = 31; i >= 0; i--){
        if(((n >> i) & 1) == 1)
            count++;
    }
    return count;
}
```



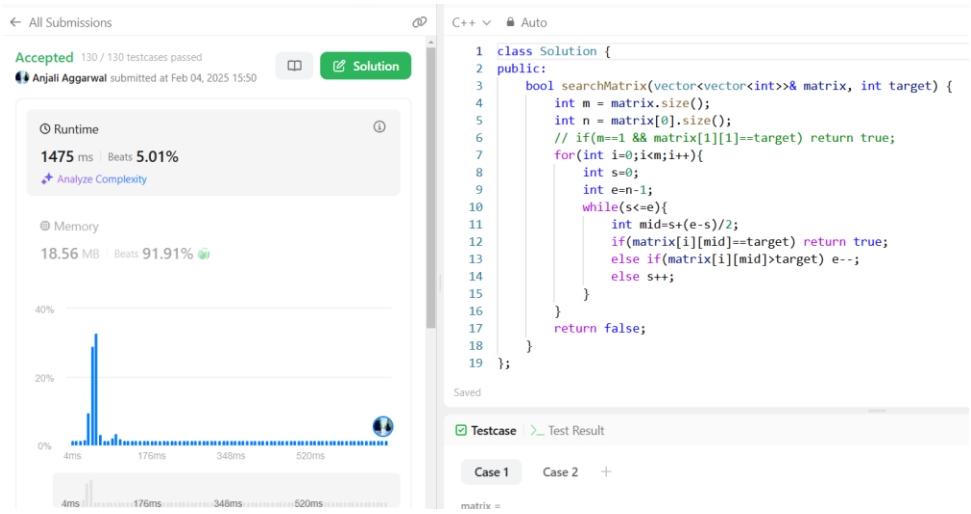
53.[Maximum Subarray]

```
int maxi=INT_MIN;
    int sum=0;
    for(int i=0;i<nums.size();i++){
        sum+=nums[i];
        if(maxi<sum){
            maxi=sum;
            cout<<maxi;
        }
        if(sum<0) sum=0;
    }
    return maxi;
```

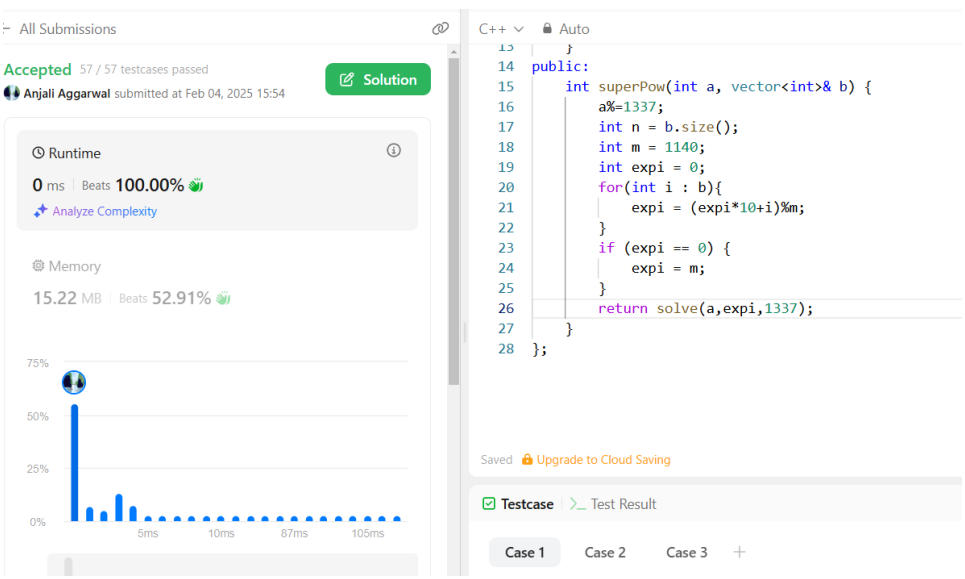


240.[Search a 2D Matrix II]

```
int m = matrix.size();
    int n = matrix[0].size();
    // if(m==1 && matrix[1][1]==target) return true;
    for(int i=0;i<m;i++){
        int s=0;
        int e=n-1;
        while(s<=e){
            int mid=s+(e-s)/2;
            if(matrix[i][mid]==target) return true;
            else if(matrix[i][mid]>target) e--;
            else s++;
        }
    }
    return false;
```



372. Super Pow

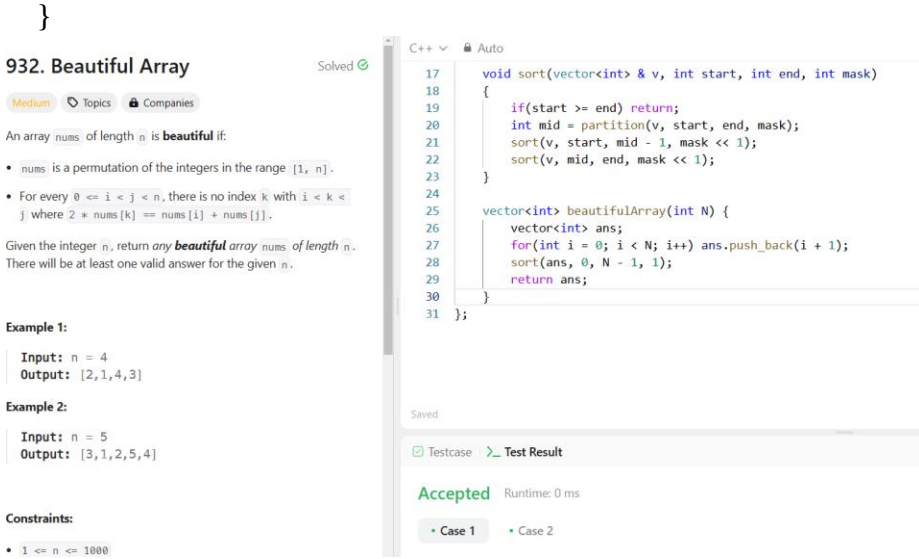



```
int superPow(int a, vector<int>& b) {
    a%=1337;
    int n = b.size();
    int m = 1140;
    int expi = 0;
    for(int i : b){
        expi = (expi*10+i)%m;
    }
    if (expi == 0) {
```

```
        expi = m;
    }
    return solve(a,expi,1337);
}
```

932.Beautiful Array

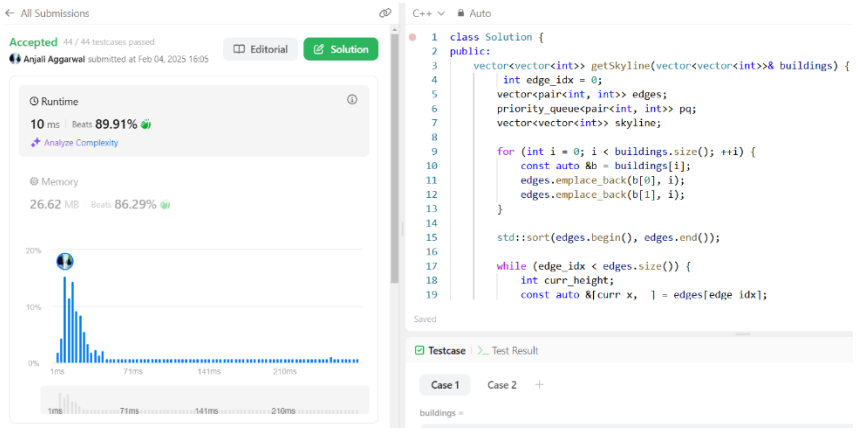
```
int partition(vector<int> &v, int start, int end, int mask)
{
    int j = start;
}
void sort(vector<int> & v, int start, int end, int mask)
{
    if(start >= end) return;
    int mid = partition(v, start, end, mask);
    sort(v, start, mid - 1, mask << 1);
    sort(v, mid, end, mask << 1);
}
vector<int> beautifulArray(int N) {
    vector<int> ans;
    for(int i = 0; i < N; i++) ans.push_back(i + 1);
    sort(ans, 0, N - 1, 1);
    return ans;
}
```



218.The Skyline Problem

```
int edge_idx = 0;
    vector<pair<int, int>> edges;
```

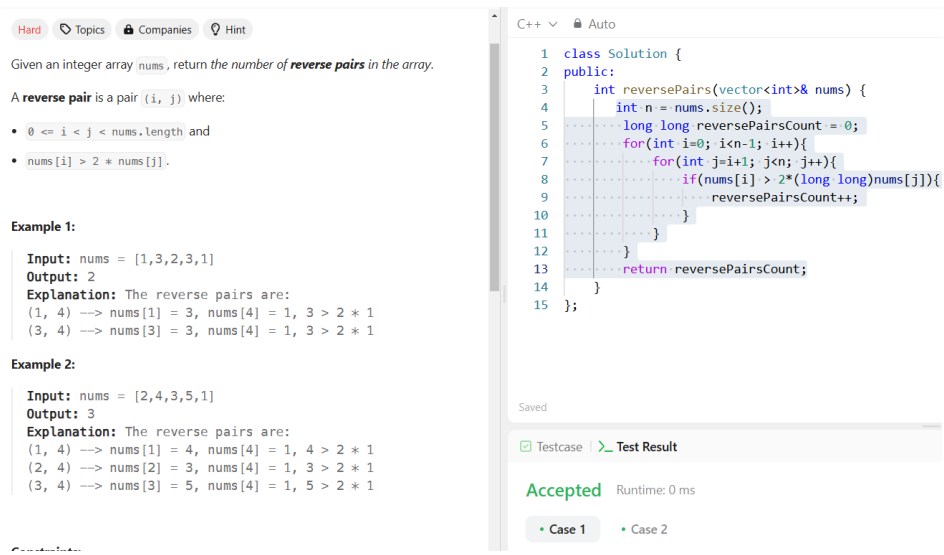
```
    priority_queue<pair<int, int>> pq;
    vector<vector<int>> skyline;

    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) {

        }
        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;
```



493. Reverse Pairs

```
int reversePairs(vector<int>& nums) {
    int n = nums.size();
    long long reversePairsCount = 0;
    for(int i=0; i<n-1; i++){
```

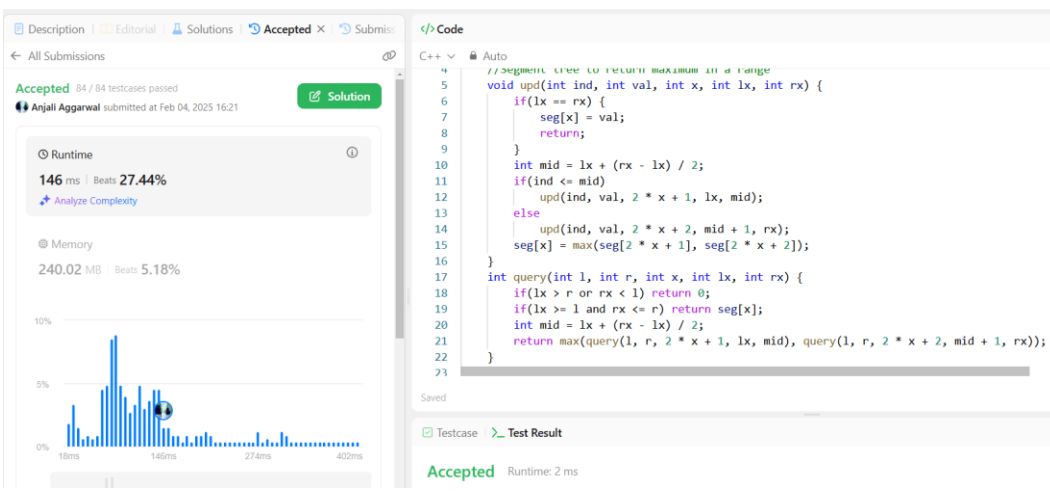
```
        for(int j=i+1; j<n; j++){
            if(nums[i] > 2*(long long)nums[j]){
                reversePairsCount++;
            }
        }
    }
    return reversePairsCount;
}
```



2407.[Longest Increasing Subsequence II]

```
void upd(int ind, int val, int x, int lx, int rx) {
    if(lx == rx) {
        seg[x] = val;
        return;
    }
    int mid = lx + (rx - lx) / 2;
    if(ind <= mid)
        upd(ind, val, 2 * x + 1, lx, mid);
    else
        upd(ind, val, 2 * x + 2, mid + 1, rx);
    seg[x] = max(seg[2 * x + 1], seg[2 * x + 2]);
}
int query(int l, int r, int x, int lx, int rx) {
    if(lx > r or rx < l) return 0;
    if(lx >= l and rx <= r) return seg[x];
```

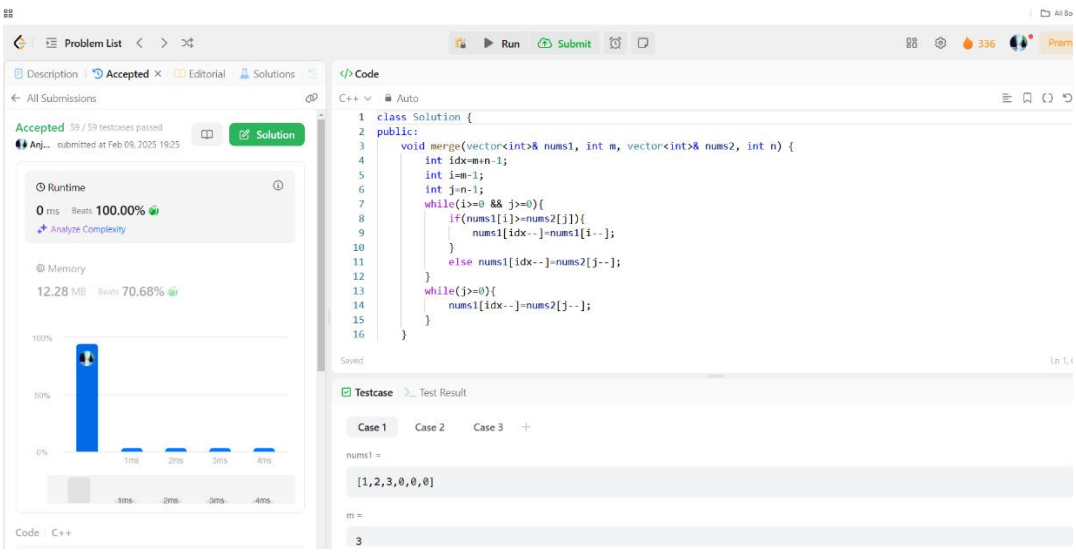
```
        int mid = lx + (rx - lx) / 2;
        return max(query(l, r, 2 * x + 1, lx, mid), query(l, r, 2 * x + 2, mid + 1, rx));
    }

    int lengthOfLIS(vector<int>& nums, int k) {
        int x = 1;
        while(x <= 200000) x *= 2;
        seg.resize(2 * x, 0);

        int res = 1;
        for(int i = 0; i < nums.size(); ++i) {
            int left = max(1, nums[i] - k), right = nums[i] - 1;
            int q = query(left, right, 0, 0, x - 1); // check for the element in the range of [nums[i] - k,
nums[i] - 1] with the maximum value
            res = max(res, q + 1);
            upd(nums[i], q + 1, 0, 0, x - 1); //update current value
        }
        return res;
    }
```



88. Merge Sorted Array

```
void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
    int idx=m+n-1;
    int i=m-1;
    int j=n-1;
    while(i>=0 && j>=0){
        if(nums1[i]>=nums2[j]){
```

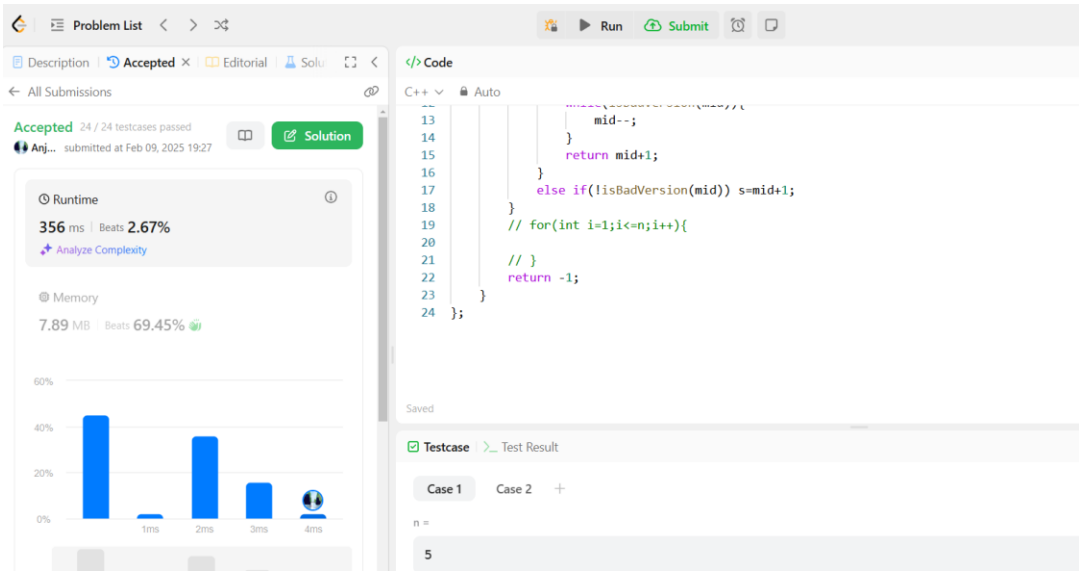
```
            nums1[idx--]=nums1[i--];
        }
        else nums1[idx--]=nums2[j--];
    }
    while(j>=0){
        nums1[idx--]=nums2[j--];
    }
}
```



278. First Bad Version

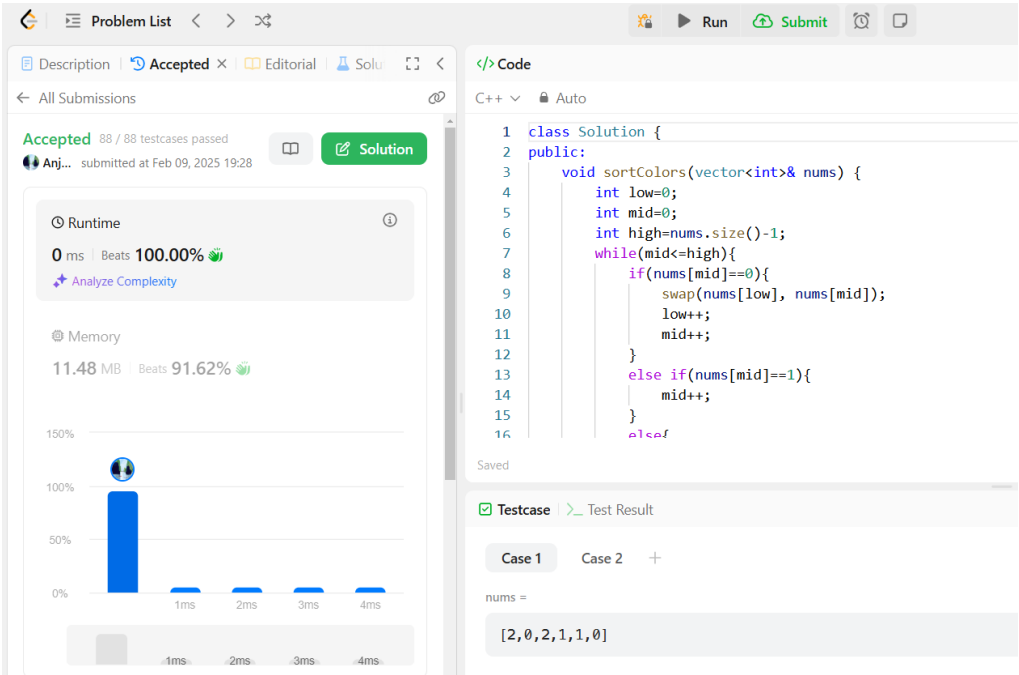
```
int firstBadVersion(int n) {
    int s=0;
    int e=n;
    while(s<=e){
        int mid=s+(e-s)/2;
        if(isBadVersion(mid)) {
            while(isBadVersion(mid)){
                mid--;
            }
            return mid+1;
        }
        else if(!isBadVersion(mid)) s=mid+1;
    }
    // for(int i=1;i<=n;i++){

    // }
    return -1;
```

```
    }
```



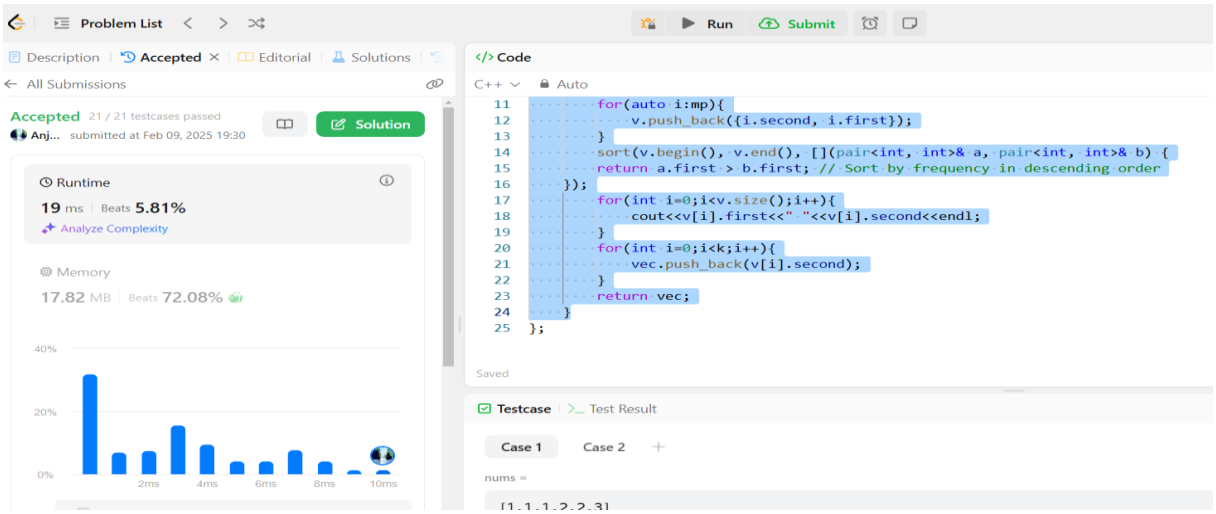
75. Sort Colors

```
void sortColors(vector<int>& nums) {
    int low=0;
    int mid=0;
    int high=nums.size()-1;
    while(mid<=high){
        if(nums[mid]==0){
            swap(nums[low], nums[mid]);
            low++;
            mid++;
        }
        else if(nums[mid]==1){
            mid++;
        }
        else{
            swap(nums[mid], nums[high]);
            high--;
        }
    }
}
```



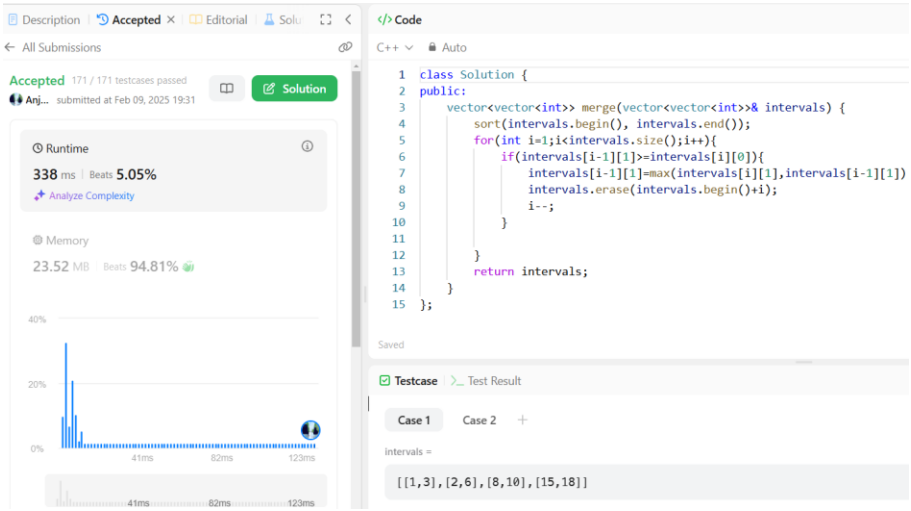
347. Top K Frequent Elements

```
vector<int> topKFrequent(vector<int>& nums, int k) {
    map<int, int>mp;
    vector<int> vec;
    // if(nums.size()==1 && k==1) return nums;
    for(int i=0;i<nums.size();i++){
        mp[nums[i]]++;
    }
    vector<pair<int, int>>v;
    for(auto i:mp){
        v.push_back({i.second, i.first});
    }
    sort(v.begin(), v.end(), [](pair<int, int>& a, pair<int, int>& b) {
    return a.first > b.first; // Sort by frequency in descending order
  });
    for(int i=0;i<v.size();i++){
        cout<<v[i].first<<" "<<v[i].second<<endl;
    }
    for(int i=0;i<k;i++){
        vec.push_back(v[i].second);
    }
    return vec;
  }
```



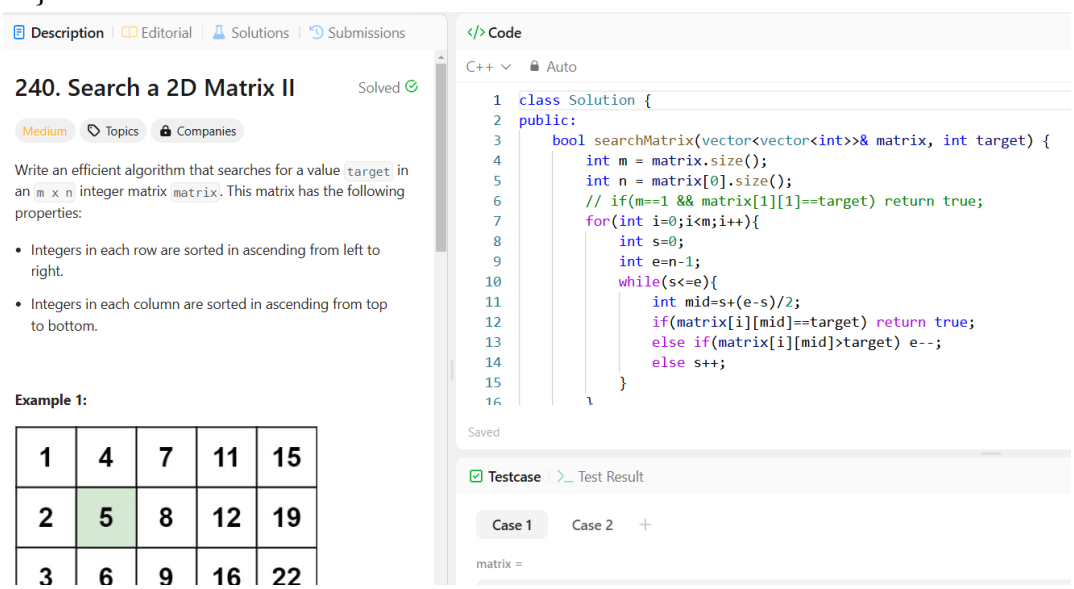
56. Merge Intervals

```
vector<vector<int>> merge(vector<vector<int>>& intervals) {
    sort(intervals.begin(), intervals.end());
    for(int i=1;i<intervals.size();i++){
      if(intervals[i-1][1]>=intervals[i][0]){
        intervals[i-1][1]=max(intervals[i][1],intervals[i-1][1]) ;
        intervals.erase(intervals.begin()+i);
        i--;
      }

    }
    return intervals;
  }
```



240. Search a 2D Matrix II

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



4. Median of Two Sorted Arrays

```
double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
    int n=nums1.size() ,m=nums2.size();
    int j=0;
    for(int i=n;i<m+n;i++){
        nums1.push_back(nums2[j]);
        j++;
    }
```

```
    sort(nums1.begin(), nums1.end());
    for(int i=0;i<nums1.size();i++){
        cout<<nums1[i]<<" ";
    }
    double ans=0;
    if(nums1.size()%2!=0){
        ans= (double)nums1[nums1.size()/2];
    }
    else{
        ans= (double)(nums1[nums1.size()/2]+nums1[(nums1.size()/2)-1])/2;
    }
    // if(ans<0) return (double)0;
    // return ans;
return ans;
}
```