

Assignment 1

Student Name: Anjali Aggarwal

Branch: CSE

Semester: 6

Subject Name: Advanced Programming

UID: 22BCS11246

Section: 22BCS_FL_602-B

Date of Performance: 7-1-25

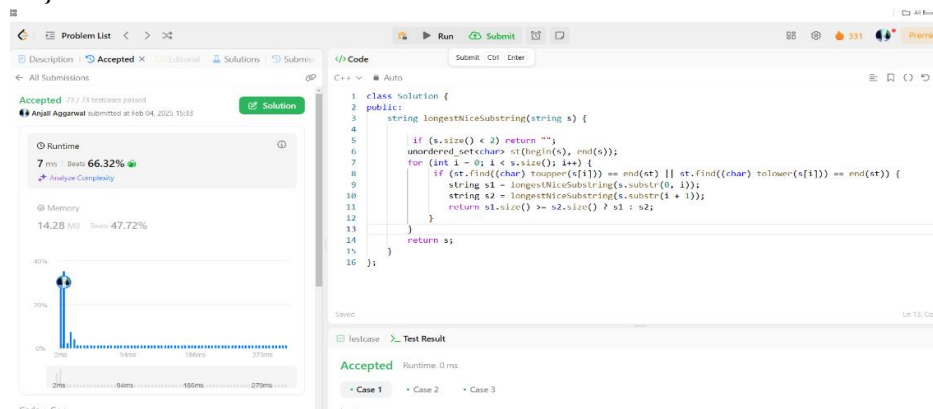
Subject Code: 22CSH-35]

QUESTION

1763. Longest Nice Substring

```
string longestNiceSubstring(string s) {

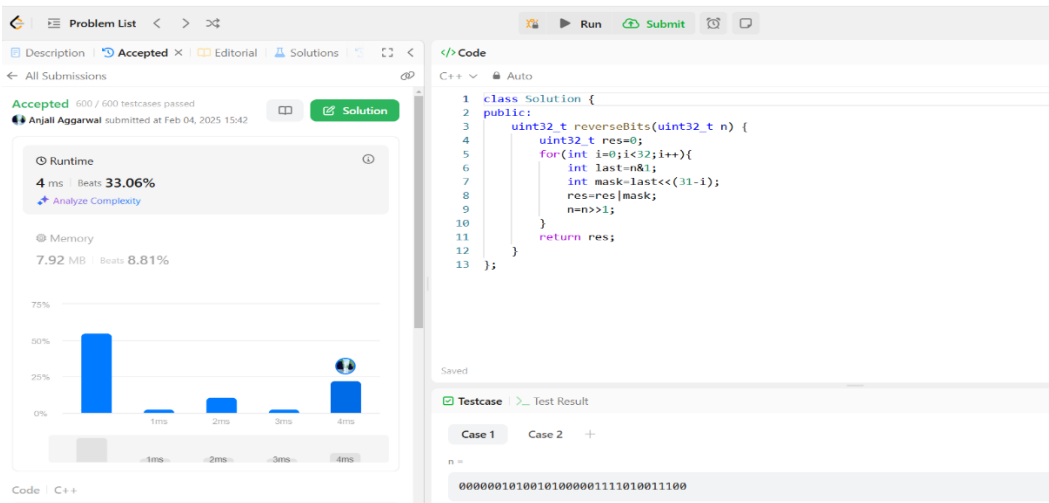
    if (s.size() < 2) return "";
    unordered_set<char> st(begin(s), end(s));
    for (int i = 0; i < s.size(); i++) {
        if (st.find((char) toupper(s[i])) == end(st) || st.find((char) tolower(s[i])) == end(st))
        {
            string s1 = longestNiceSubstring(s.substr(0, i));
            string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



190. Reverse Bits

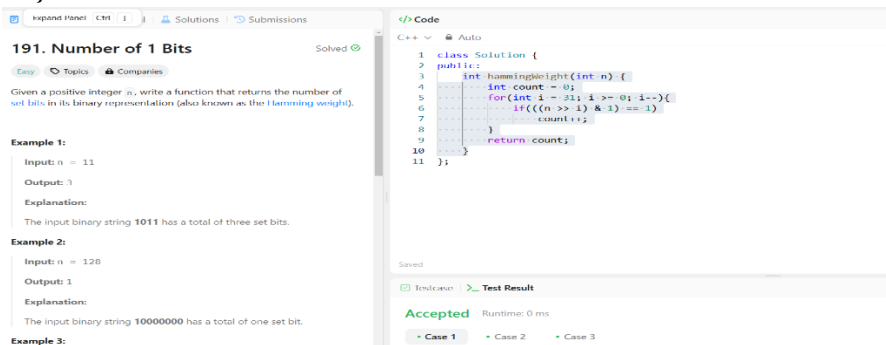
```
uint32_t reverseBits(uint32_t n) {
    uint32_t res=0;
    for(int i=0;i<32;i++){
        int last=n&1;
```

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



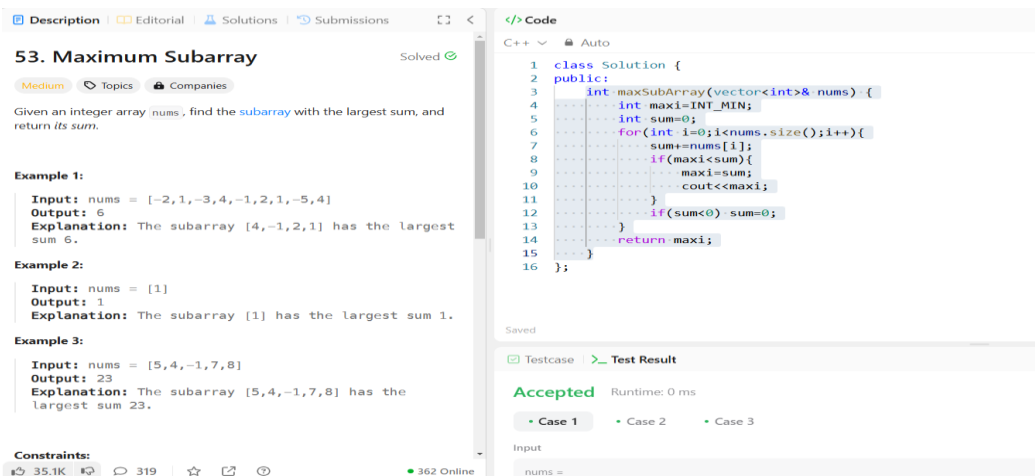
191. Number of 1 Bits

```
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;
```



53. Maximum Subarray Solved

Medium Topics Companies

Given an integer array `nums`, find the `subarray` with the largest sum, and return its sum.

Example 1:
 Input: `nums = [-2,1,-3,4,-1,2,1,-5,4]`
 Output: 6
 Explanation: The subarray `[4,-1,2,1]` has the largest sum 6.

Example 2:
 Input: `nums = [1]`
 Output: 1
 Explanation: The subarray `[1]` has the largest sum 1.

Example 3:
 Input: `nums = [5,4,-1,7,8]`
 Output: 23
 Explanation: The subarray `[5,4,-1,7,8]` has the largest sum 23.

Constraints:
 1 ≤ `nums.length` ≤ 10⁵
 -10⁴ ≤ `nums[i]` ≤ 10⁴

35.1K 319 362 Online

```
1 class Solution {
2 public:
3     int maxSubArray(vector<int>& nums) {
4         int maxi=INT_MIN;
5         int sum=0;
6         for(int i=0;i<nums.size();i++){
7             sum+=nums[i];
8             if(maxi<sum){
9                 maxi=sum;
10                cout<<maxi;
11            }
12            if(sum<0) sum=0;
13        }
14        return maxi;
15    }
16 };
```

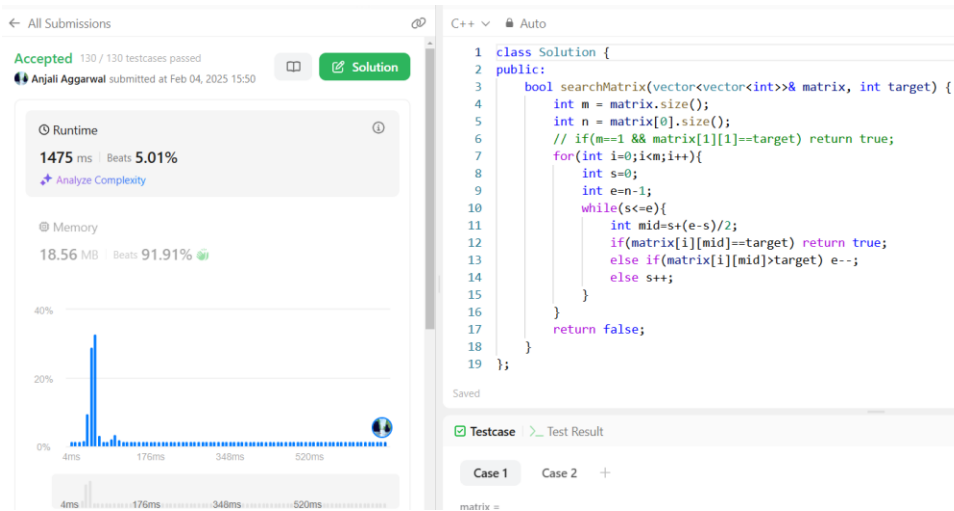
Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

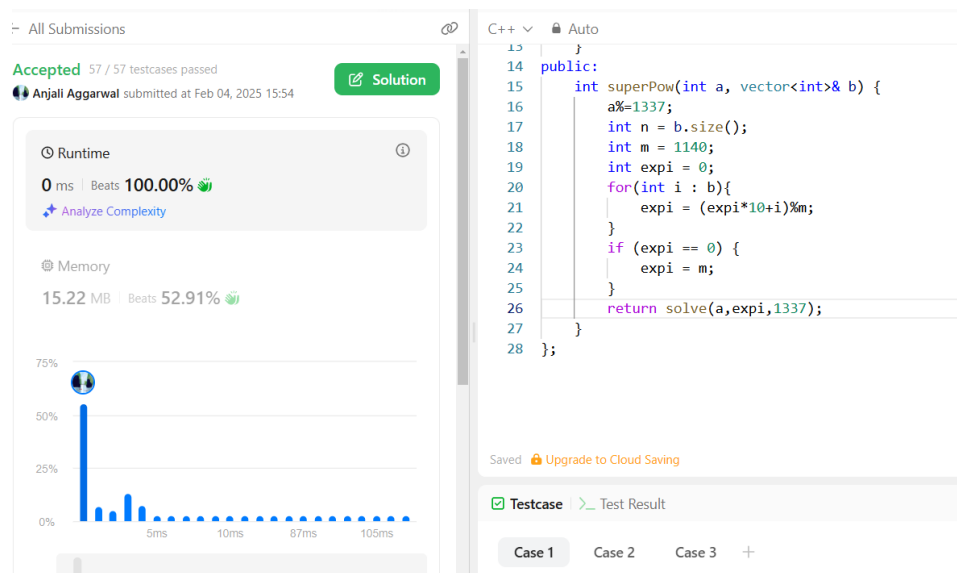
Input
 nums =

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;
}

```

932. Beautiful Array

Medium Topics Companies

An array `nums` of length `n` is **beautiful** if:

- `nums` is a permutation of the integers in the range `[1, n]`.
- For every $0 \leq i < j < n$, there is no index `k` with $i < k < j$ where $2 * \text{nums}[k] == \text{nums}[i] + \text{nums}[j]$.

Given the integer `n`, return *any* **beautiful** array `nums` of length `n`. There will be at least one valid answer for the given `n`.

Example 1:

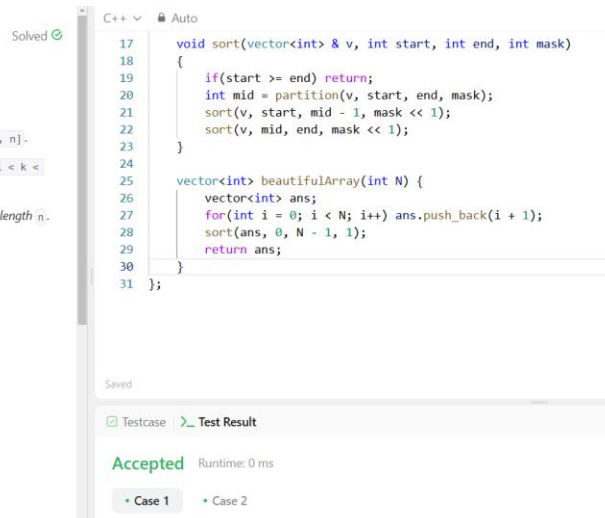
Input: `n = 4`
Output: `[2,1,4,3]`

Example 2:

Input: `n = 5`
Output: `[3,1,2,5,4]`

Constraints:

- $1 \leq n \leq 1000$



218. [The Skyline Problem](#)

```

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

```

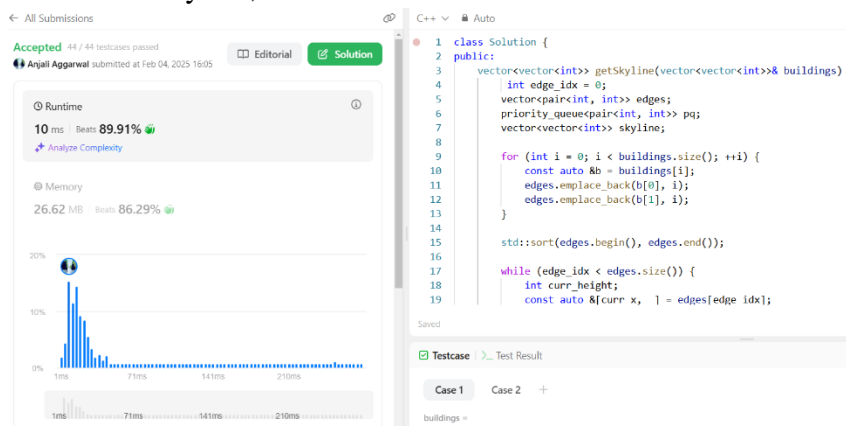
```
priority_queue<pair<int, int>> pq;
vector<vector<int>> skyline;

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;
}

```

Hard
Topics
Companies
Hint

Given an integer array `nums`, return the number of **reverse pairs** in the array.

A **reverse pair** is a pair (i, j) where:

- $0 \leq i < j < \text{nums.length}$ and
- $\text{nums}[i] > 2 * \text{nums}[j]$.

Example 1:

Input: `nums = [1,3,2,3,1]`
Output: 2
Explanation: The reverse pairs are:
 $(1, 4) \rightarrow \text{nums}[1] = 3, \text{nums}[4] = 1, 3 > 2 * 1$
 $(3, 4) \rightarrow \text{nums}[3] = 3, \text{nums}[4] = 1, 3 > 2 * 1$

Example 2:

Input: `nums = [2,4,3,5,1]`
Output: 3
Explanation: The reverse pairs are:
 $(1, 4) \rightarrow \text{nums}[1] = 4, \text{nums}[4] = 1, 4 > 2 * 1$
 $(2, 4) \rightarrow \text{nums}[2] = 3, \text{nums}[4] = 1, 3 > 2 * 1$
 $(3, 4) \rightarrow \text{nums}[3] = 5, \text{nums}[4] = 1, 5 > 2 * 1$

C++
Auto

```

1 class Solution {
2 public:
3     int reversePairs(vector<int>& nums) {
4         int n = nums.size();
5         long long reversePairsCount = 0;
6         for(int i=0; i<n-1; i++){
7             for(int j=i+1; j<n; j++){
8                 if(nums[i] > 2*(long long)nums[j]){
9                     reversePairsCount++;
10                }
11            }
12        }
13        return reversePairsCount;
14    }
15 };

```

Saved

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

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]);
}

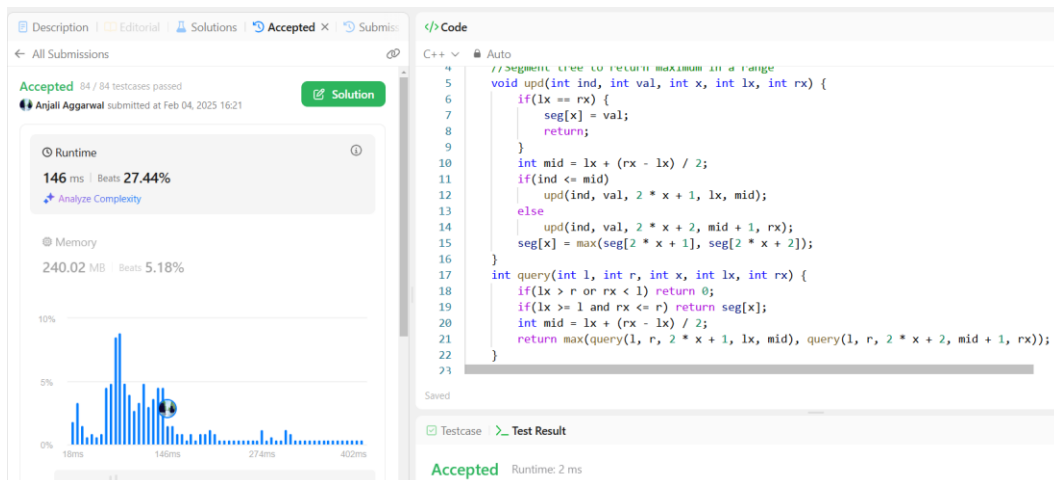
int query(int l, int r, int x, int lx, int rx) {
    if(lx > r or rx < l) return 0;
    if(lx >= l and rx <= r) return seg[x];
}

```

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

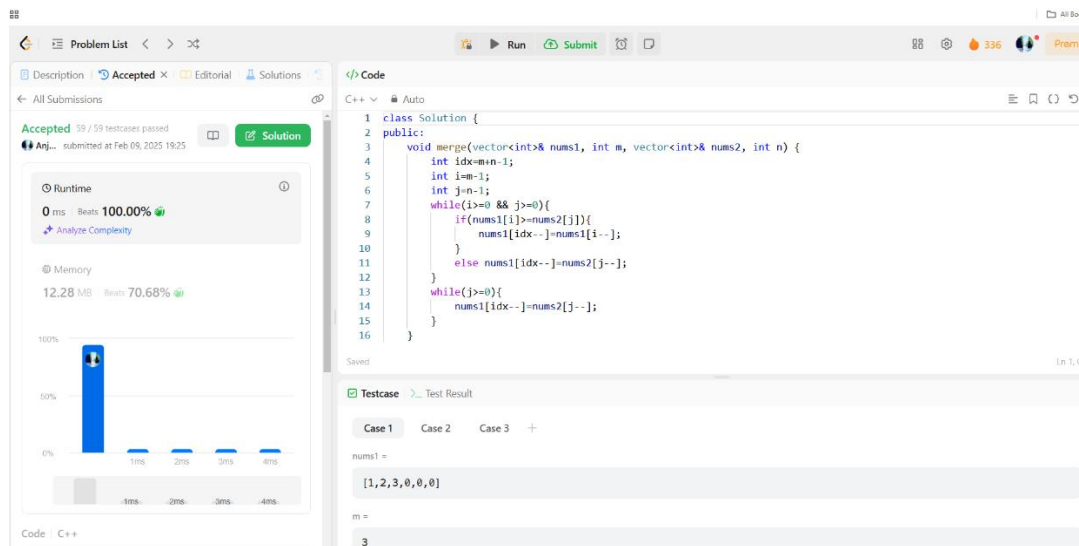
```
void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
    int idx=m+n-1;
    int i=m-1;
    int j=n-1;
    while(i>=0 && j>=0){
        if(nums1[i]>=nums2[j]){
```



```

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

```



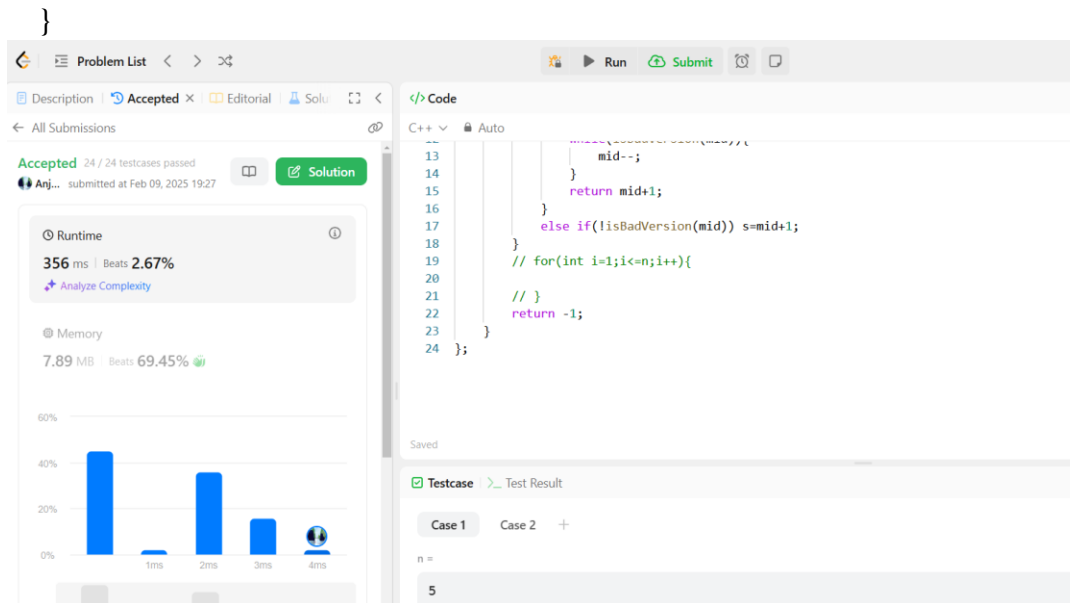
278. First Bad Version

```

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;
}

```

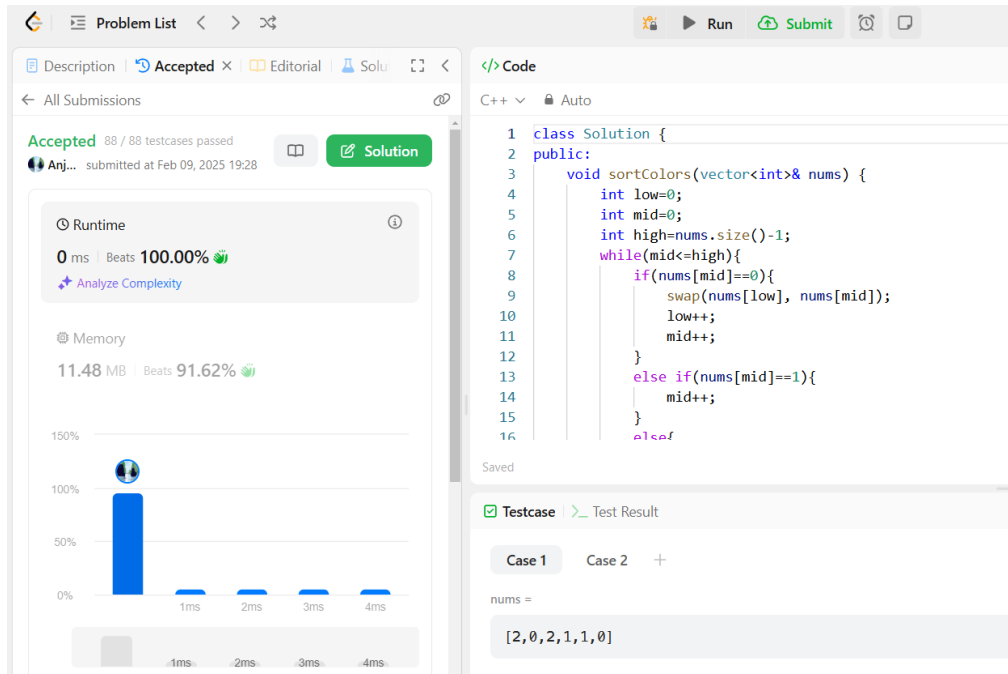


75. Sort Colors

```

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--;
        }
    }
}

```



Accepted 88 / 88 testcases passed
Anj... submitted at Feb 09, 2025 19:28

Runtime
0 ms | Beats 100.00%

Memory
11.48 MB | Beats 91.62%

Code
C++

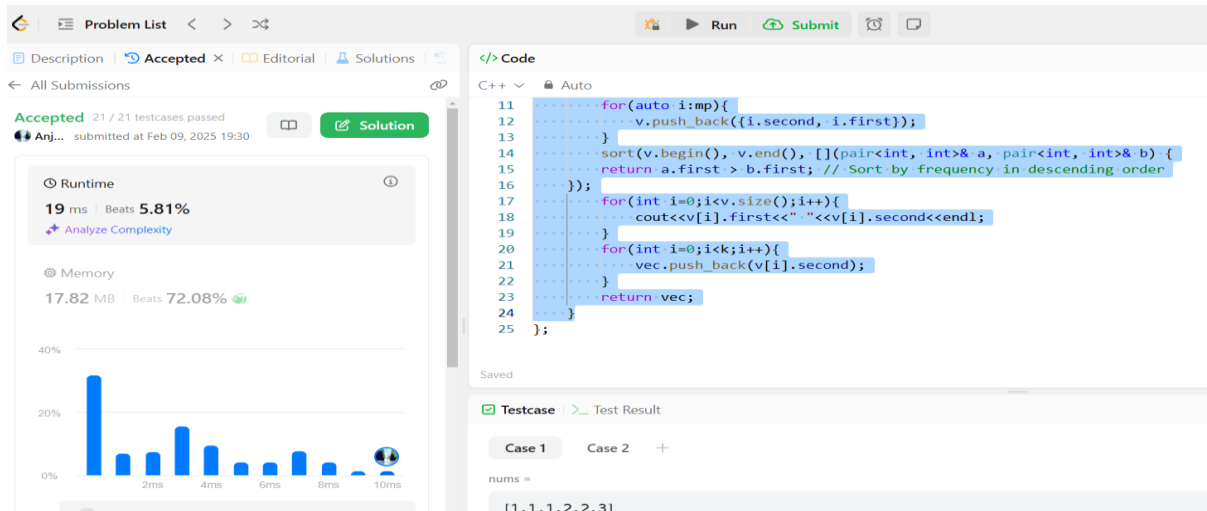
```
1 class Solution {
2 public:
3     void sortColors(vector<int>& nums) {
4         int low=0;
5         int mid=0;
6         int high=nums.size()-1;
7         while(mid<=high){
8             if(nums[mid]==0){
9                 swap(nums[low], nums[mid]);
10                low++;
11                mid++;
12            }
13            else if(nums[mid]==1){
14                mid++;
15            }
16            else {
17                swap(nums[mid], nums[high]);
18                high--;
19            }
20        }
21    }
22 }
```

Testcase
Case 1 Case 2 +

nums =
[2, 0, 2, 1, 1, 0]

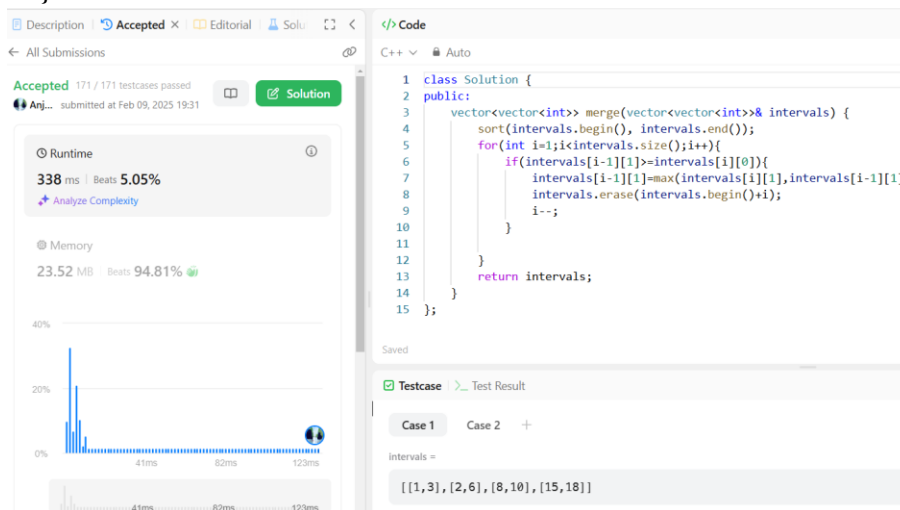
347. Top K Frequent Elements

```
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;
}
```

Description | Editorial | Solutions | Submissions

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

Code

C++ Auto

```

1 class Solution {
2 public:
3     bool searchMatrix(vector<vector<int>>& matrix, int target) {
4         int m = matrix.size();
5         int n = matrix[0].size();
6         // if(m==1 && matrix[1][1]==target) return true;
7         for(int i=0;i<m;i++){
8             int s=0;
9             int e=n-1;
10            while(s<=e){
11                int mid=s+(e-s)/2;
12                if(matrix[i][mid]==target) return true;
13                else if(matrix[i][mid]>target) e--;
14                else s++;
15            }
16        }
17    }
18 }
```

Saved

Testcase Test Result

Case 1 Case 2 +

matrix =

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;
}
```

← All Submissions

Accepted 2096 / 2096 testcases passed

Anj... submitted at Feb 09, 2025 19:33

[Solution](#)


Runtime

103 ms | Beats 5.00%

[Analyze Complexity](#)

Memory

95.61 MB | Beats 37.23%



The graph shows a single bar for the first test case reaching approximately 55% on the y-axis. The x-axis is labeled with 5ms and 10ms. A small profile icon is visible at the end of the x-axis.

C++ Auto

```

11 for(int i=0;i<nums1.size();i++){
12     cout<<nums1[i]<<" ";
13 }
14 double ans=0;
15 if(nums1.size()%2!=0){
16     ans= (double)nums1[nums1.size()/2];
17 }
18 else{
19     ans= (double)(nums1[nums1.size()/2]+nums1[(nums1.size()/2)-1])/2;
20 }
21 // if(ans<0) return (double)0;
22 // return ans;
23 return ans;
24 }
25 ;

```

Saved

Testcase Test Result

Case 1 Case 2 +

nums1 =

[1,3]