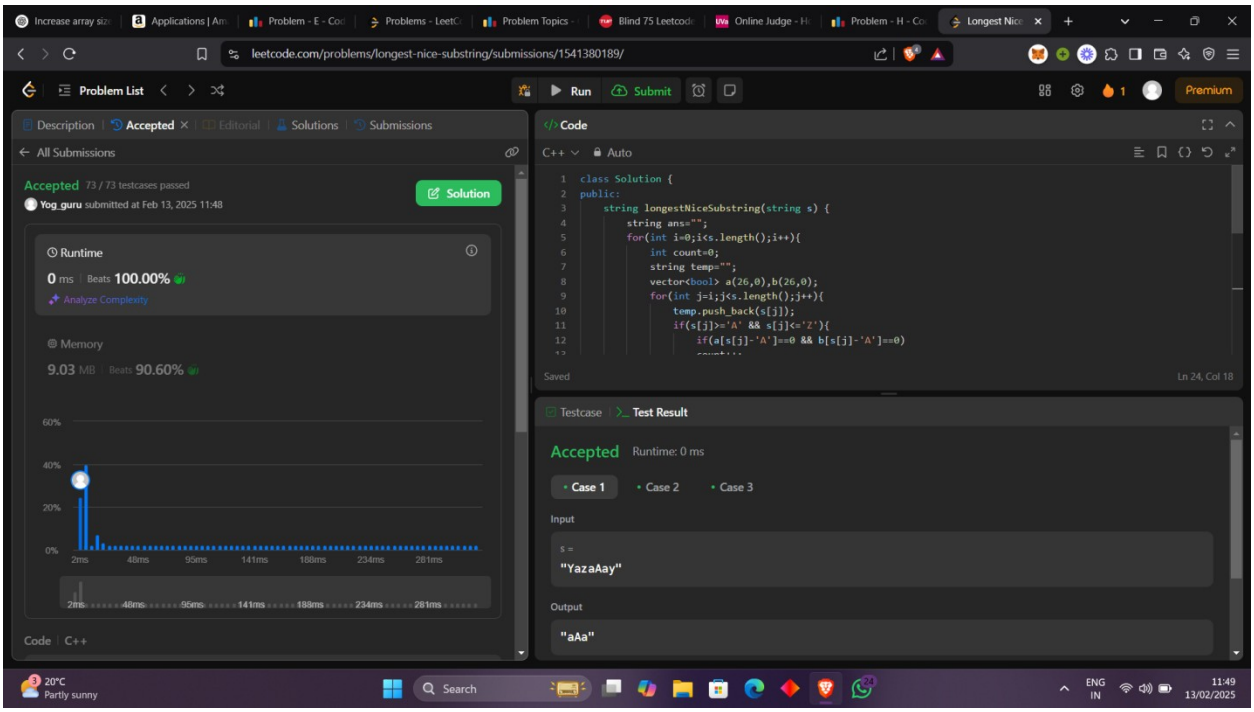
1763.Longest Nice Substring

```
class Solution {
public:
    string longestNiceSubstring(string s) {
        string ans="";
        for(int i=0;i<s.length();i++){
            int count=0;
            string temp="";
            vector<bool> a(26,0),b(26,0);
            for(int j=i;j<s.length();j++){
                temp.push_back(s[j]);
                if(s[j]>='A' && s[j]<='Z'){
                    if(a[s[j]-'A']==0 && b[s[j]-'A']==0)
                    count++;
                    else if(a[s[j]-'A'] && !b[s[j]-'A'])
                    count--;
                    b[s[j]-'A']=1;
                }
                else{
                    if(b[s[j]-'a']==0 && a[s[j]-'a']==0)
                    count++;
                    else if(b[s[j]-'a'] && !a[s[j]-'a'])
                    count--;
                    a[s[j]-'a']=1;
                }
                if(ans.size()<temp.size() && count==0)
                ans=temp;
            }
```

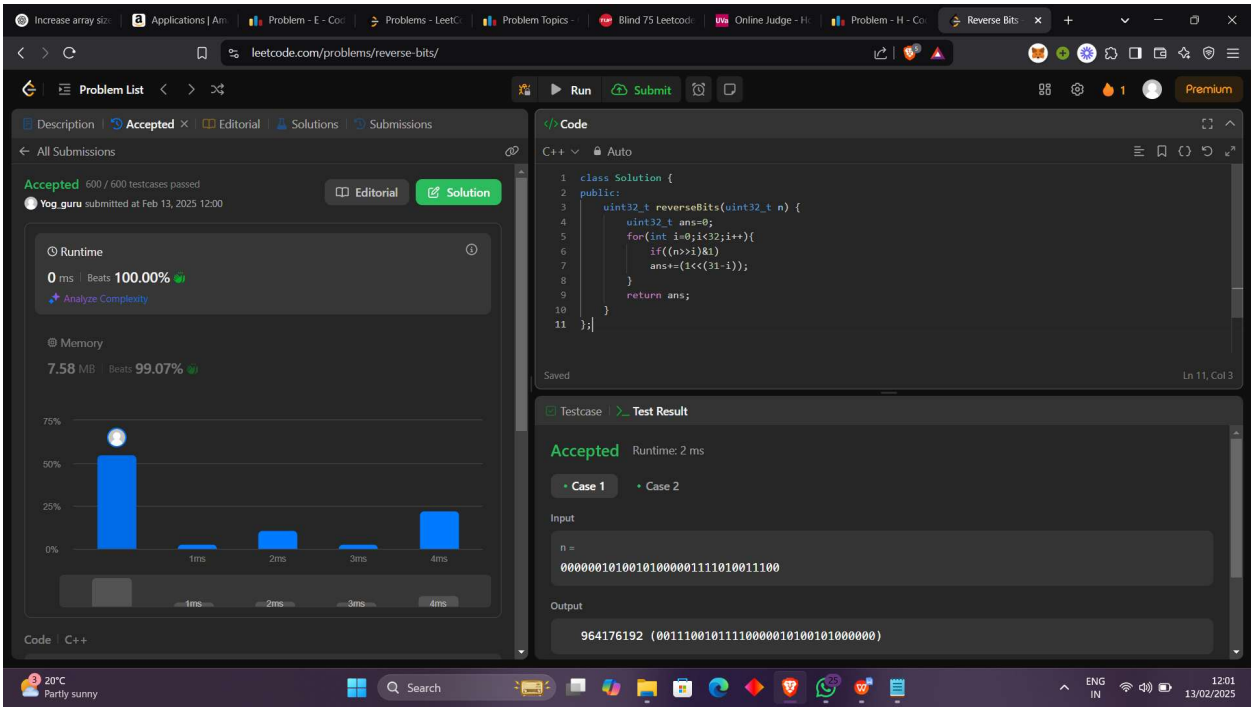
```
        }
        return ans;
    }
};
```



190.Reverse Bits

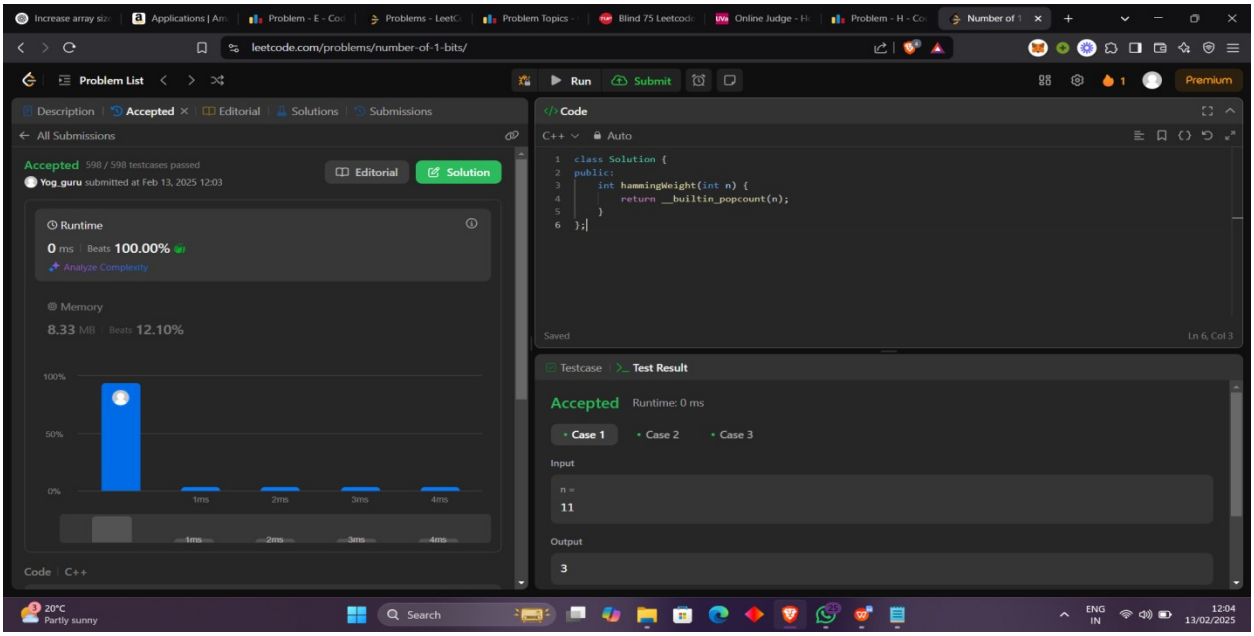
```
class Solution {
public:
    uint32_t reverseBits(uint32_t n) {
        uint32_t ans=0;
        for(int i=0;i<32;i++){
            if((n>>i)&1)
            ans+=(1<<(31-i));
        }
```

```
        return ans;
    }
};
```



191. Number of 1 Bits

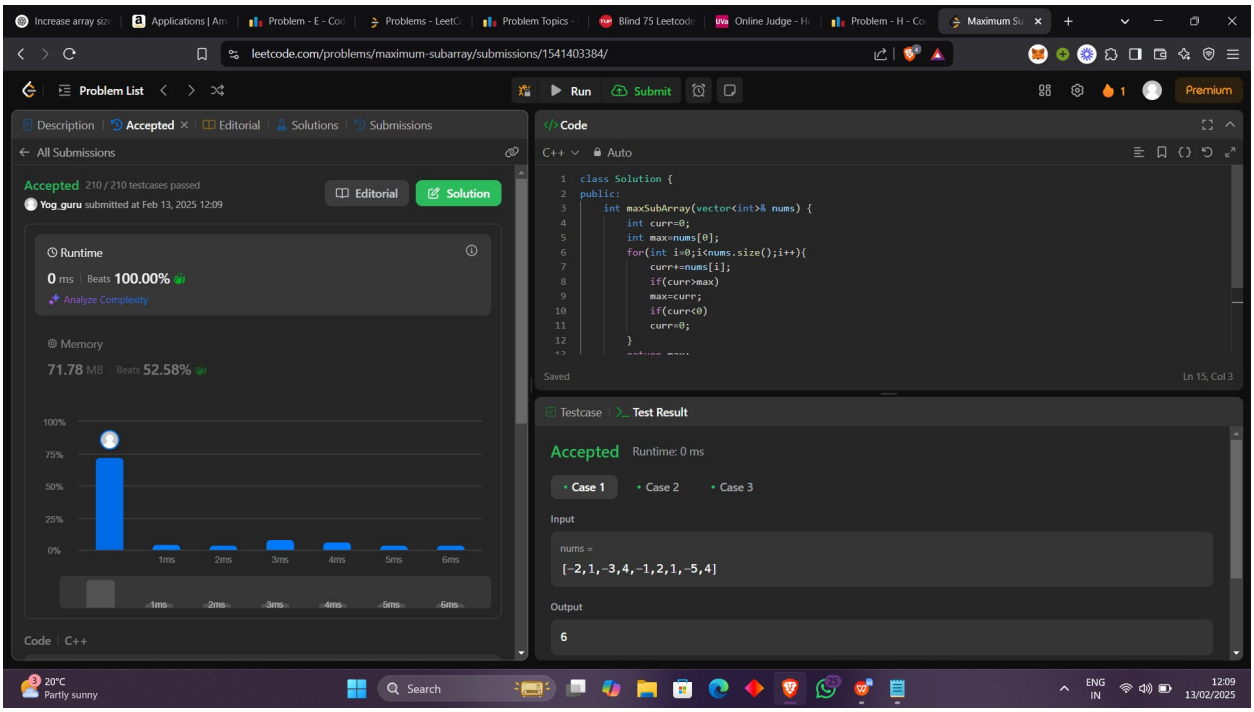
```
class Solution {
public:
    int hammingWeight(int n) {
        return __builtin_popcount(n);
    }
};
```



53. Maximum Subarray

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int curr=0;
        int max=nums[0];
        for(int i=0;i<nums.size();i++){
            curr+=nums[i];
            if(curr>max)
            max=curr;
            if(curr<0)
            curr=0;
        }
```

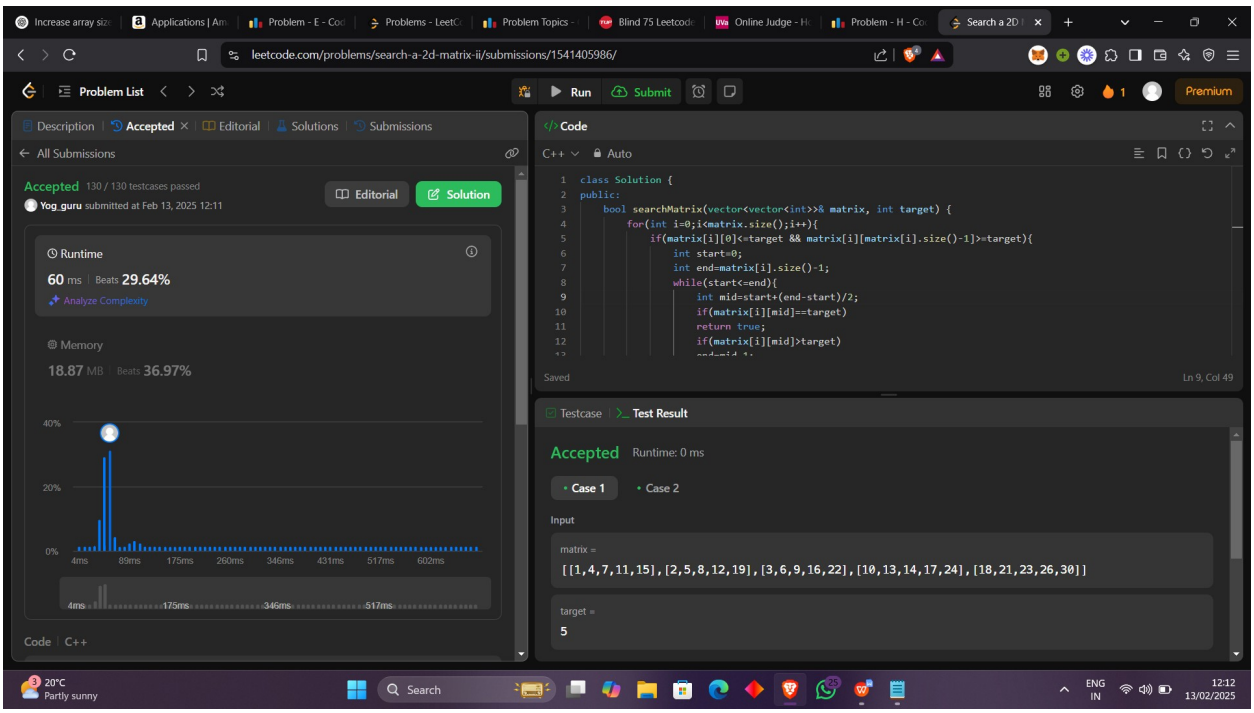
```
        return max;
    }
};
```



240. Search a 2D Matrix II

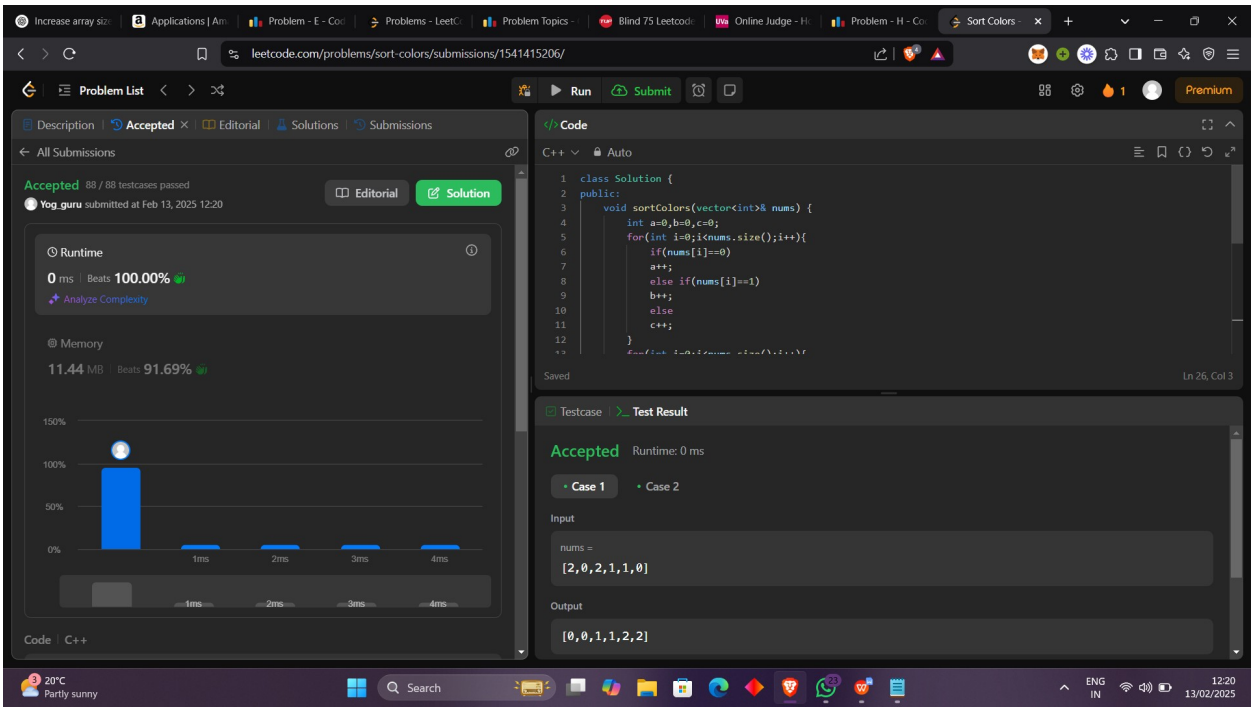
```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        for(int i=0;i<matrix.size();i++){
            if(matrix[i][0]<=target && matrix[i][matrix[i].size()-1]>=target){
                int start=0;
                int end=matrix[i].size()-1;
                while(start<=end){
```

```
            int mid=start+(end-start)/2;
            if(matrix[i][mid]==target)
            return true;
            if(matrix[i][mid]>target)
            end=mid-1;
            else
            start=mid+1;
        }
    }
    return false;
  }
};
```



75. Sort Colours

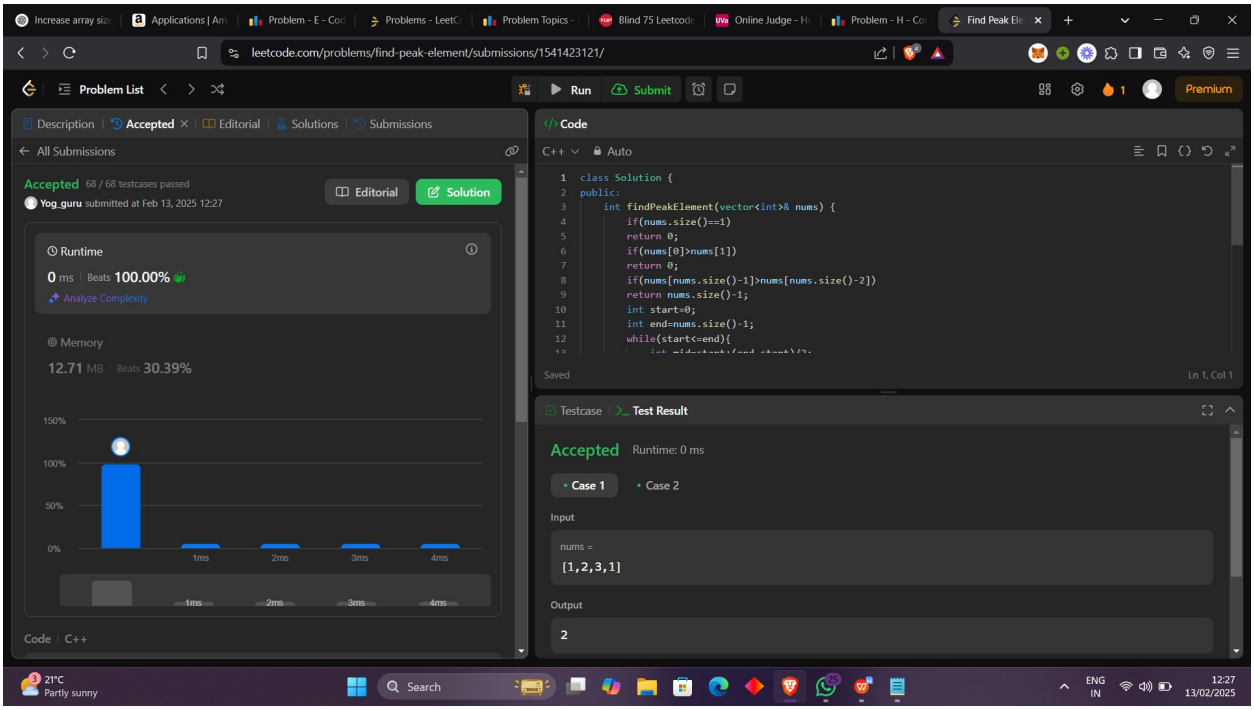
```
class Solution {
public:
    void sortColors(vector<int>& nums) {
        int a=0,b=0,c=0;
        for(int i=0;i<nums.size();i++){
            if(nums[i]==0)
            a++;
            else if(nums[i]==1)
            b++;
            else
            c++;
        }
        for(int i=0;i<nums.size();i++){
            if(a>0){
                nums[i]=0;
                a--;
            }
            else if(b>0){
                nums[i]=1;
                b--;
            }
            else
            nums[i]=2;
        }
    }
};
```



162. Find Peak Element

```
class Solution {
public:
    int findPeakElement(vector<int>& nums) {
        if(nums.size()==1)
        return 0;
        if(nums[0]>nums[1])
        return 0;
        if(nums[nums.size()-1]>nums[nums.size()-2])
        return nums.size()-1;
        int start=0;
        int end=nums.size()-1;
        while(start<=end){
            int mid=start+(end-start)/2;
            if(nums[mid]>nums[mid-1] && nums[mid]>nums[mid+1])
            return mid;
```

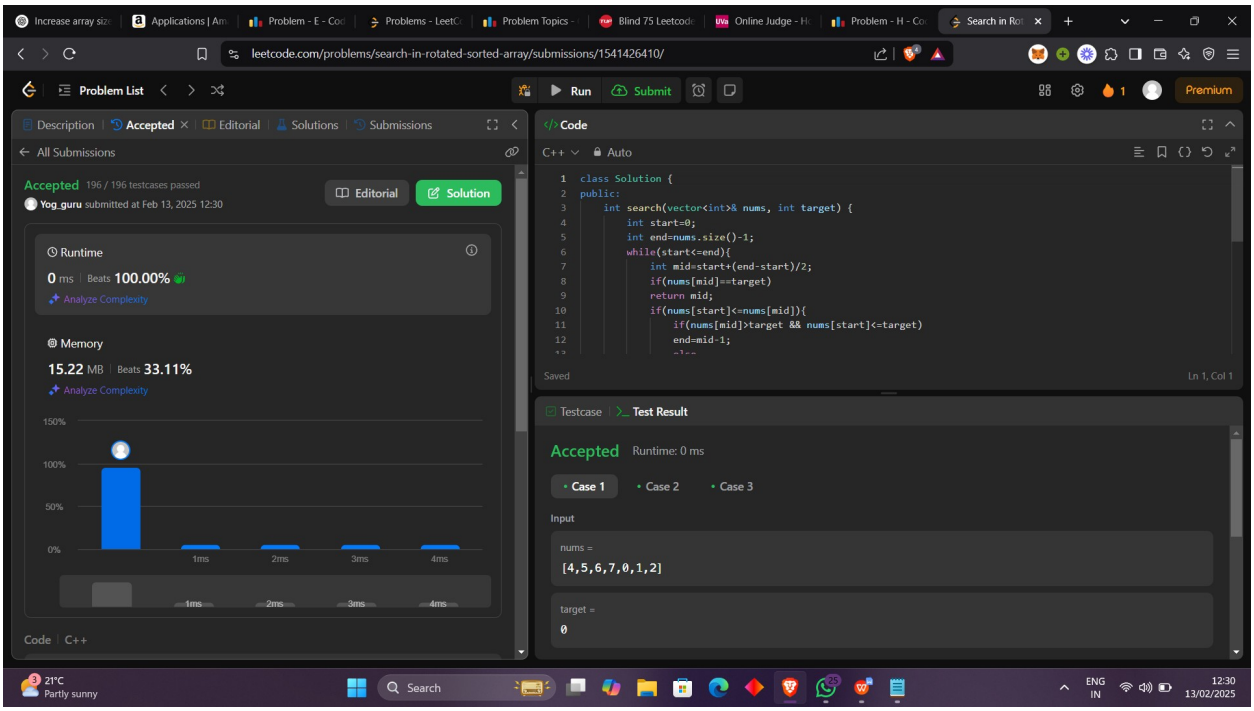
```
            if(start==0)
            start++;
            if(nums[mid]<nums[mid+1])
            start=mid+1;
            else
            end=mid-1;
        }
    return 0;
    }
};
```



33. Search in Rotated Sorted Array

```
class Solution {
public:
    int search(vector<int>& nums, int target) {
```

```
        int start=0;
        int end=nums.size()-1;
        while(start<=end){
            int mid=start+(end-start)/2;
            if(nums[mid]==target)
            return mid;
            if(nums[start]<=nums[mid]){
                if(nums[mid]>target && nums[start]<=target)
                end=mid-1;
                else
                start=mid+1;
            }
            else{
                cout<<mid<<endl;
                if(nums[mid]<target && nums[end]>=target)
                start=mid+1;
                else
                end=mid-1;
            }
            cout<<start<<" "<<end<<endl;
        }
        return -1;
    }
};
```



4. Median of Two Sorted Arrays

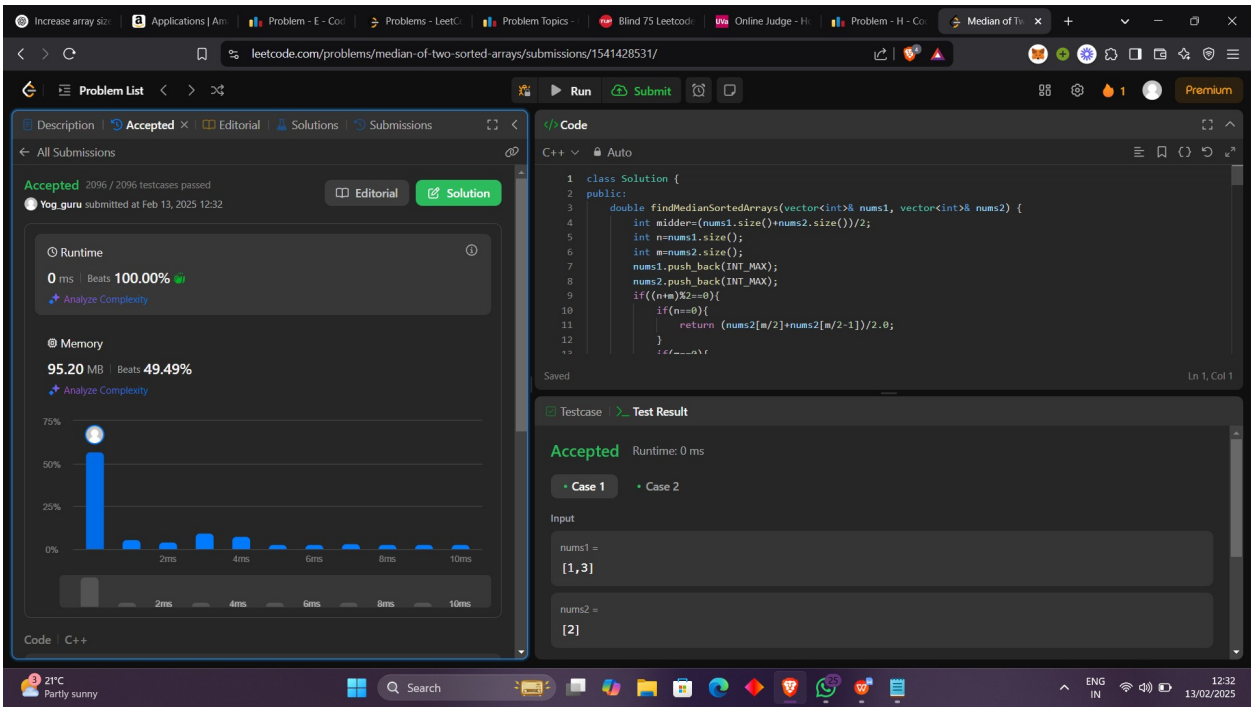
```
class Solution {
public:
    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
        int midder=(nums1.size()+nums2.size())/2;
        int n=nums1.size();
        int m=nums2.size();
        nums1.push_back(INT_MAX);
        nums2.push_back(INT_MAX);
        if((n+m)%2==0){
            if(n==0){
                return (nums2[m/2]+nums2[m/2-1])/2.0;
            }
```

```
if(m==0){
    return (nums1[n/2]+nums1[n/2-1])/2.0;
}
int start=0;
int end=n;
int mid1,mid2;
while(start<=end){
    mid1=start+(end-start)/2;
    mid2=midder-mid1;
    if(mid2<0){
        end=mid1-1;
    }
    else if(m<mid2){
        start=mid1+1;
    }
    else{
        if(mid1==0){
            if(nums2[mid2-1]<=nums1[mid1]){
                return (nums2[mid2-1]+min(nums1[mid1],nums2[mid2]))/2.0;
            }
            else{
                start=mid1+1;
            }
        }
        else if(mid2==0){
            if(nums1[mid1-1]<=nums2[mid2]){
                return (nums1[mid1-1]+min(nums1[mid1],nums2[mid2]))/2.0;
            }
```

```
                else{
                    end=mid1-1;
                }
            }
            else{
                if(nums1[mid1-1]>nums2[mid2]){
                    end=mid1-1;
                }
                else if(nums2[mid2-1]>nums1[mid1]){
                    start=mid1+1;
                }
                else{
                    return (max(nums1[mid1-1],nums2[mid2-
1])+min(nums1[mid1],nums2[mid2]))/2.0;
                }
            }
        }
    }
    else{
        if(n==0){
            return nums2[m/2];
        }
        if(m==0){
            return nums1[n/2];
        }
        midder++;
        int start=0;
        int end=n;
```

```
int mid1,mid2;
while(start<=end){
    mid1=start+(end-start)/2;
    mid2=midder-mid1;
    if(mid2<0){
        end=mid1-1;
    }
    else if(m<mid2){
        start=mid1+1;
    }
    else{
        if(mid1==0){
            if(nums2[mid2-1]<=nums1[mid1]){
                return nums2[mid2-1];
            }
            else{
                start=mid1+1;
            }
        }
        else if(mid2==0){
            if(nums1[mid1-1]<=nums2[mid2]){
                return nums1[mid1-1];
            }
            else{
                end=mid1-1;
            }
        }
```

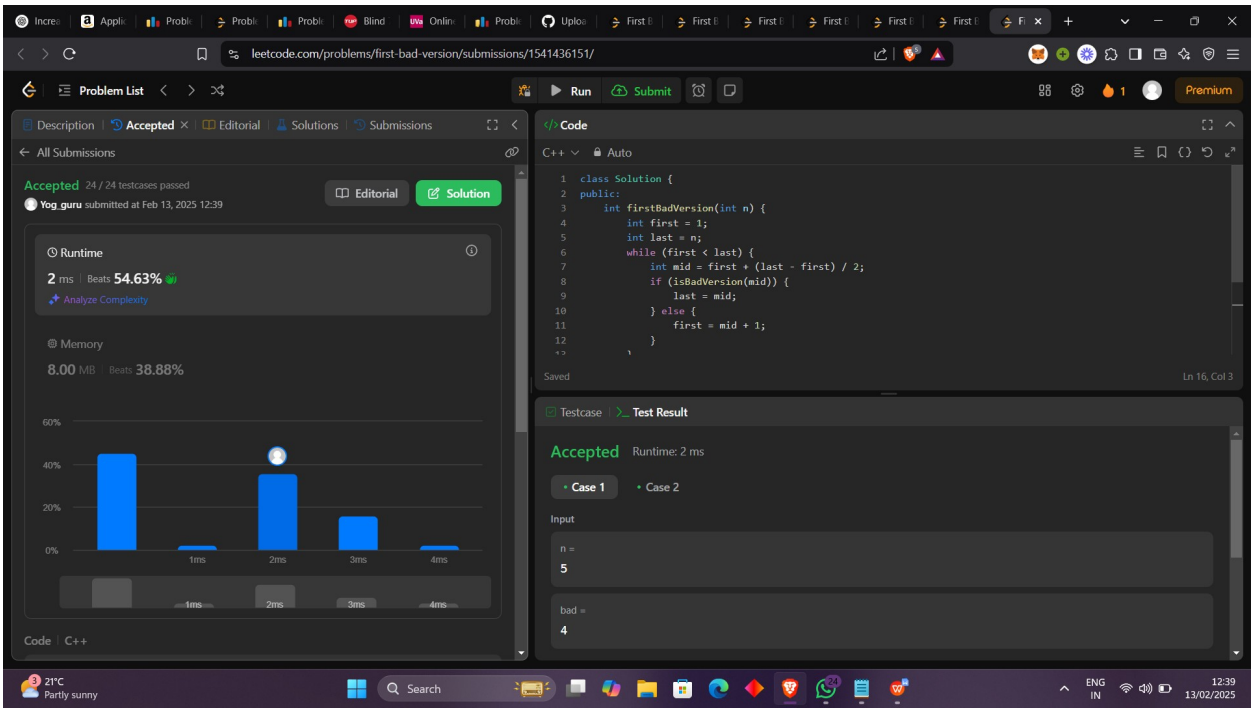
```
                else{
                    if(nums1[mid1-1]>nums2[mid2]){
                        end=mid1-1;
                    }
                    else if(nums2[mid2-1]>nums1[mid1]){
                        start=mid1+1;
                    }
                    else{
                        return max(nums2[mid2-1],nums1[mid1-1]);
                    }
                }
            }
        }
        return 1;
    }
};
```



278. First Bad Version

```
class Solution {
public:
    int firstBadVersion(int n) {
        int first = 1;
        int last = n;
        while (first < last) {
            int mid = first + (last - first) / 2;
            if (isBadVersion(mid)) {
                last = mid;
            } else {
                first = mid + 1;
            }
        }
        return first;
```

```
    }
};
```


leetcode.com/problems/first-bad-version/submissions/1541436151/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 24 / 24 testcases passed
Yog_guru submitted at Feb 13, 2025 12:39
Editorial
Solution
Runtime
2 ms | Beats 54.63%
Analyze Complexity
Memory
8.00 MB | Beats 38.88%
60%
40%
20%
0%
1ms
2ms
3ms
4ms
Code | C++
Code
C++
Auto
class Solution {
public:
int firstBadVersion(int n) {
int first = 1;
int last = n;
while (first < last) {
int mid = first + (last - first) / 2;
if (isBadVersion(mid)) {
last = mid;
} else {
first = mid + 1;
}
Saved
Ln 16, Col 3
Testcase
Test Result
Accepted
Runtime: 2 ms
Case 1
Case 2
Input
n =
5
bad =
4
21°C
Partly sunny
Search
ENG
IN
12:39
13/02/2025