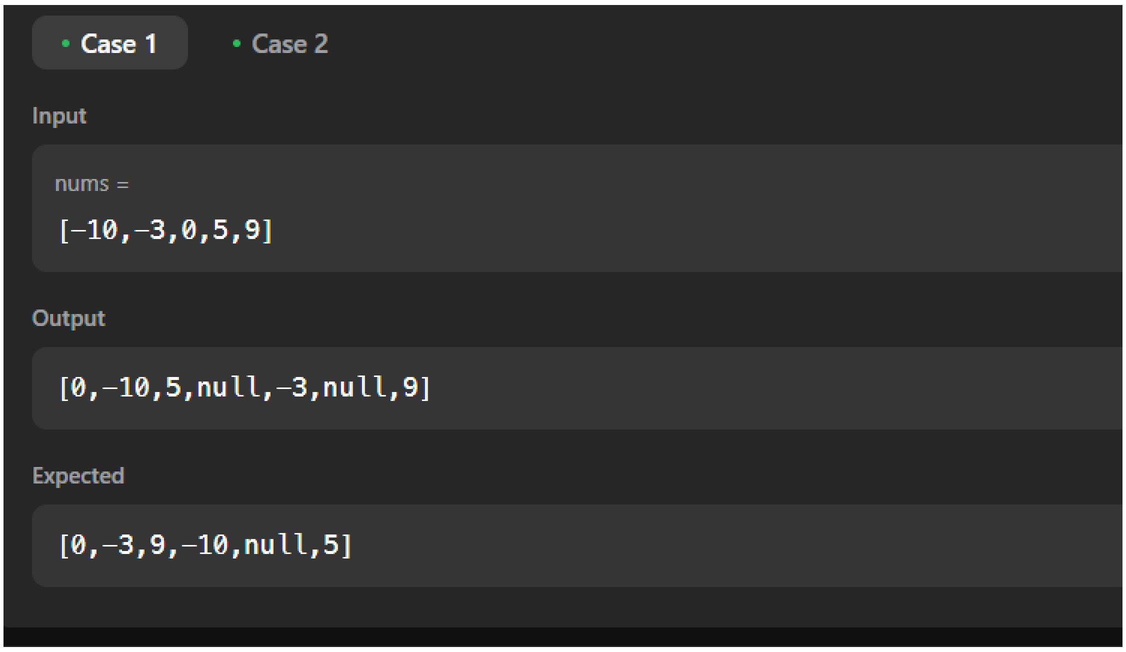
Assignment-6

Name-Akul UID-22BCS10330 Section-605-B

108.Convert Sorted Array to Binary Search Tree

```cpp
class Solution {
public:
    TreeNode* sortedArrayToBST(vector<int>& nums) {
        return helper(nums, 0, nums.size() - 1);
    }
private:
    TreeNode* helper(vector<int>& nums, int left, int right) {
        if (left > right) return nullptr;
        int mid = left + (right - left) / 2;
        TreeNode* root = new TreeNode(nums[mid]);
        root->left = helper(nums, left, mid - 1);
        root->right = helper(nums, mid + 1, right);
        return root;
    }
};
```

Output:



Input

nums =

```
[-10,-3,0,5,9]
```

Output

```
[0,-10,5,null,-3,null,9]
```

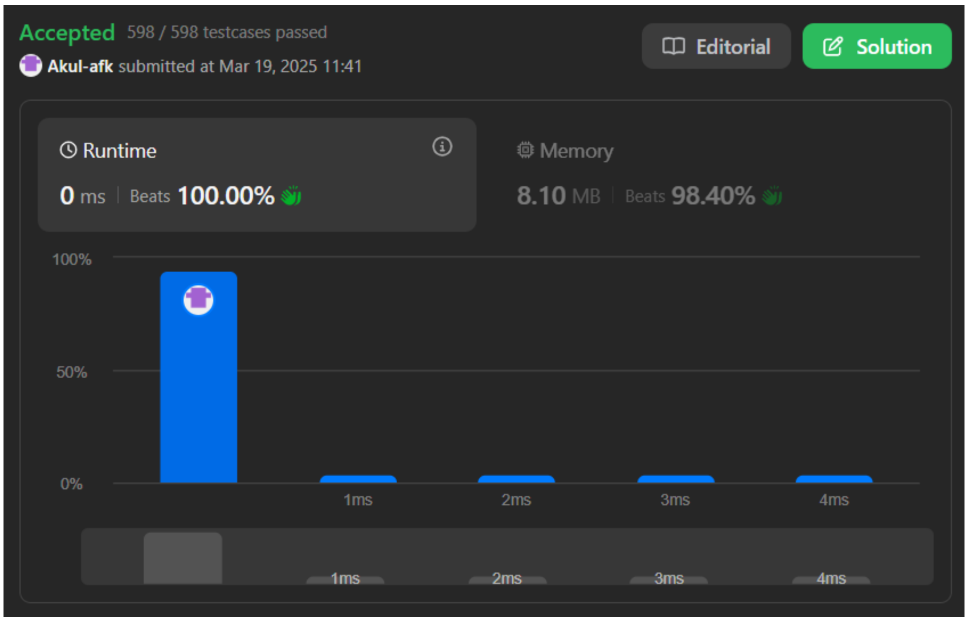
Expected

```
[0,-3,9,-10,null,5]
```

191.Number of 1 Bits

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

Output:

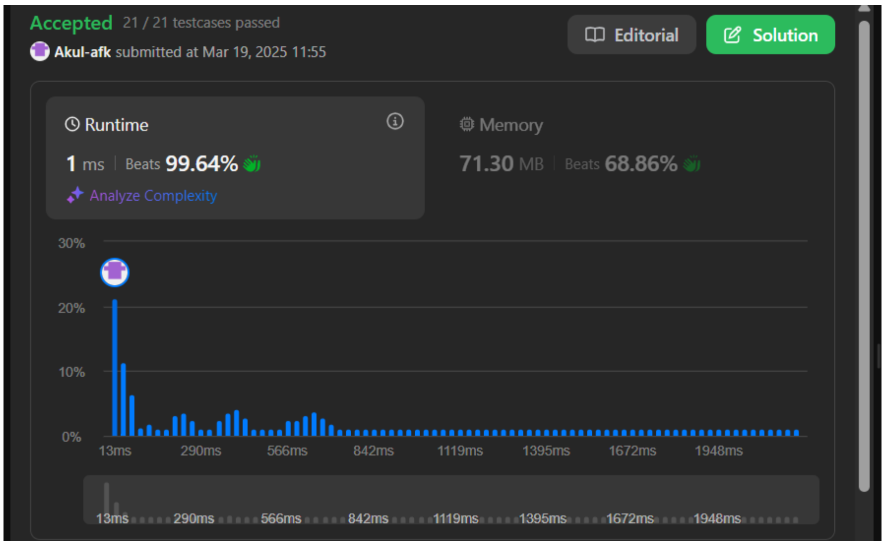

Accepted 598 / 598 testcases passed
Akul-afk submitted at Mar 19, 2025 11:41

Runtime
0 ms | Beats 100.00%
Memory
8.10 MB | Beats 98.40%

912.Sort an Array

```
class Solution {
public:
    vector<int> sortArray(vector<int>& nums) {
        int arr[100001] = {0};
        for (int i = 0; i < nums.size(); i++) {
            arr[nums[i] + 50000]++;
        }
        int index = 0;
        for (int i = 0; i < 100001; i++) {
            for (int j = 0; j < arr[i]; j++) {
                nums[index++] = i - 50000;
            }
        }
        return nums;
    }
};
```

OUTPUT:


Accepted 21 / 21 testcases passed
Akul-afk submitted at Mar 19, 2025 11:55

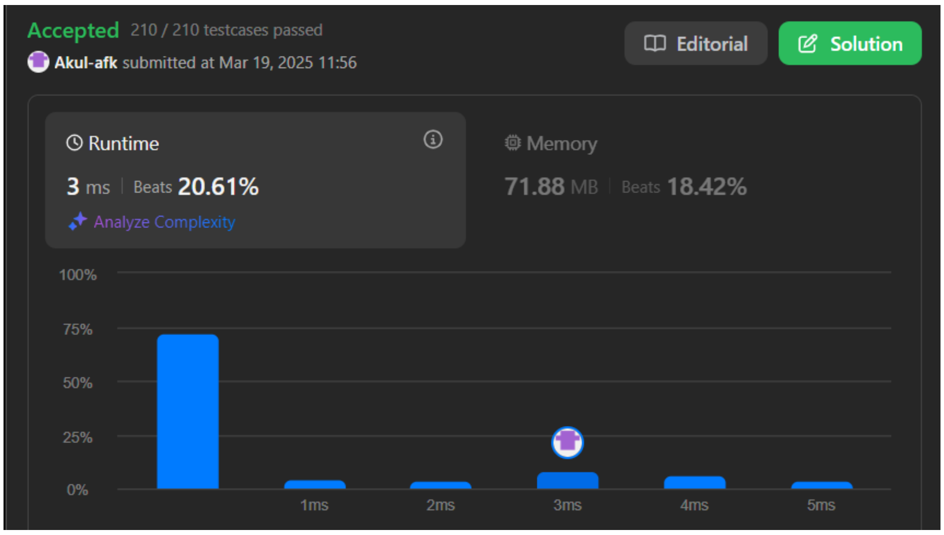
Runtime
1 ms | Beats 99.64%

Memory
71.30 MB | Beats 68.86%

53.Maximum Subarray

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int curMax = 0, maxTillNow = INT_MIN;
        for(auto c : nums)
            curMax = max(c, curMax + c),
            maxTillNow = max(maxTillNow, curMax);
        return maxTillNow;
    }
};
```

OUTPUT:


Accepted 210 / 210 testcases passed
Akul-afk submitted at Mar 19, 2025 11:56

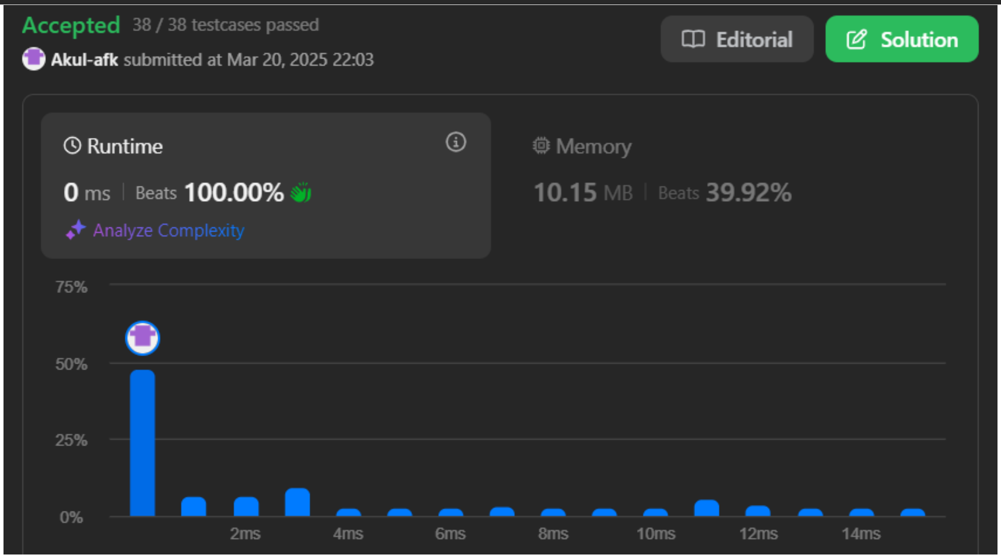
Runtime
3 ms | Beats 20.61%

Memory
71.88 MB | Beats 18.42%

932.Beautiful Array

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

OUTPUT:


Accepted 38 / 38 testcases passed
Akul-afk submitted at Mar 20, 2025 22:03

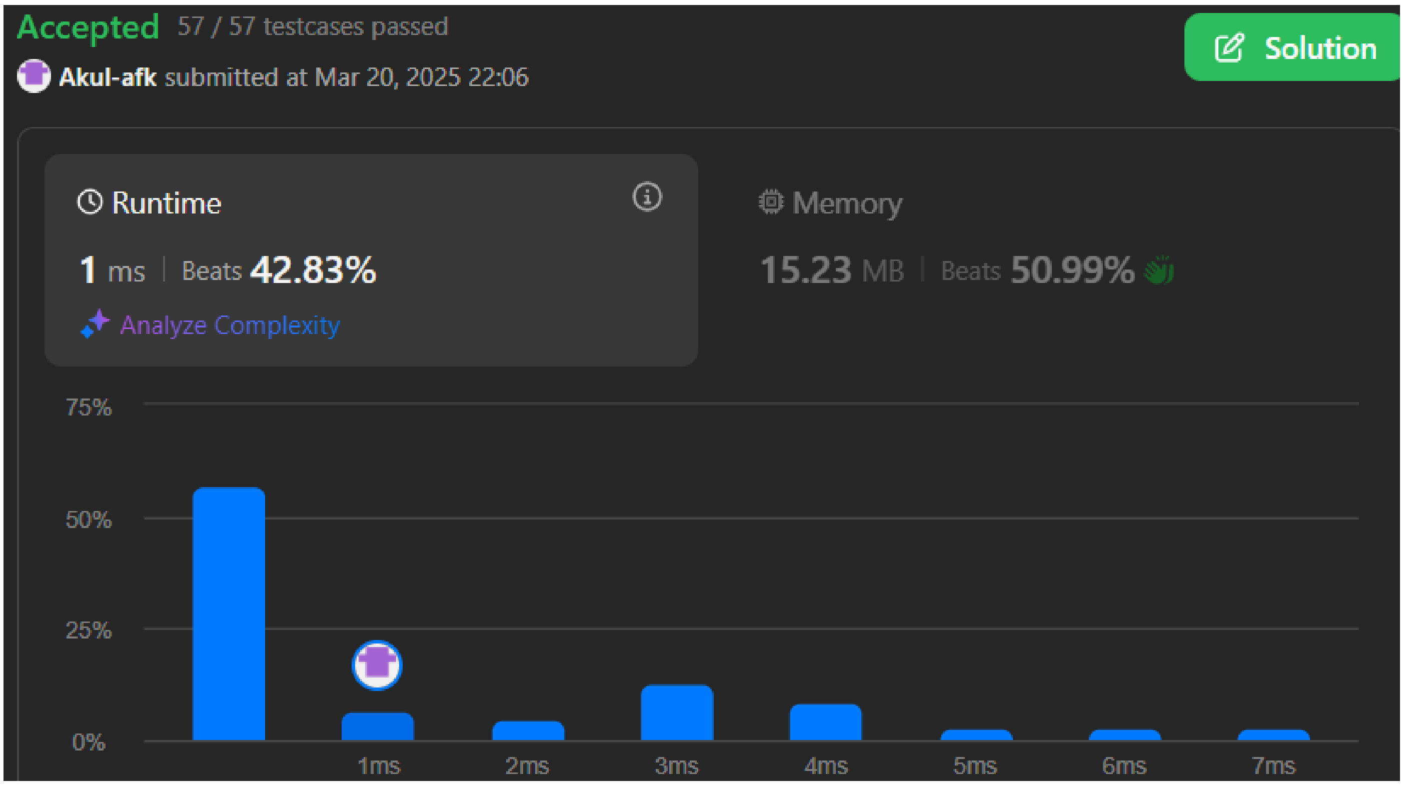
Runtime
0 ms | Beats 100.00%

Memory
10.15 MB | Beats 39.92%

372.Super Pow

```
class Solution {
public:
       int find(int a,int b)
       {
           a%=1337;
           int res=1;
           for(int i=0;i<b;i++)
           {
              res*=a;
               res%=1337;
           }
           return res;
       }
    int superPow(int a, vector<int>& b) {
        int res=1,x,f;
        for(int i=0;i<b.size();i++)
        {
             x=find(a,b[i]);
             x*=res;
             x%=1337;
             f=x;
             x=find(x,10);
             res=x;
        }
        return f;
    }
};
```

OUTPUT:



Accepted 57 / 57 testcases passed
Akul-afk submitted at Mar 20, 2025 22:06

Runtime
1 ms | Beats 42.83%

Memory
15.23 MB | Beats 50.99%

218.The Skyline Problem

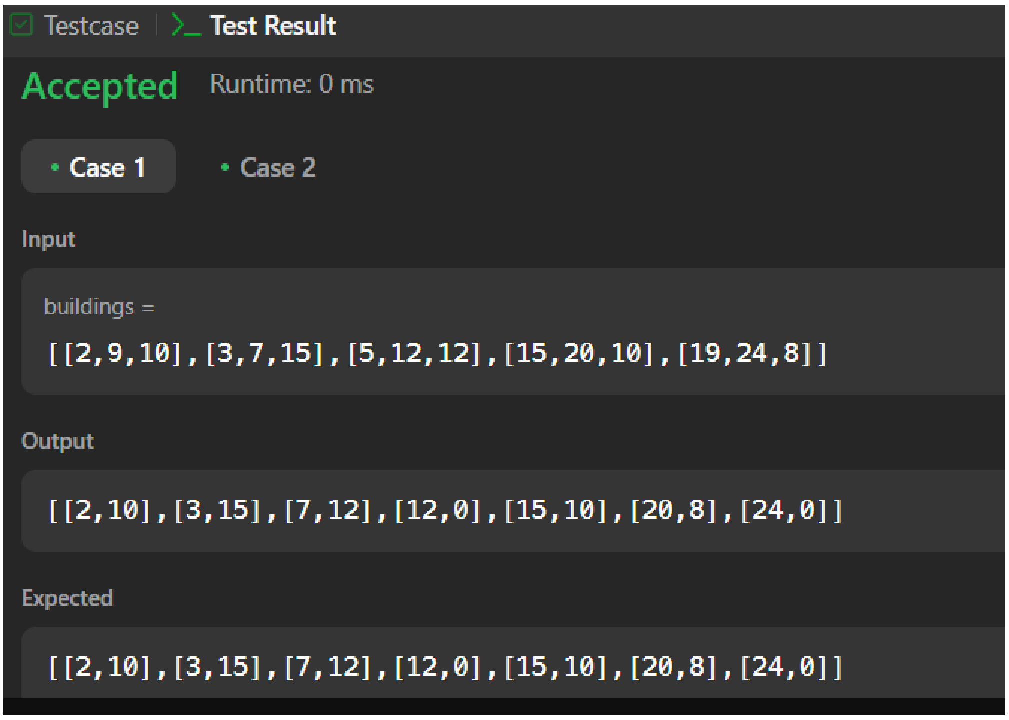
```
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<vector<int>> ans;
        multiset<int> pq{0};

        vector<pair<int, int>> points;
        for(auto b: buildings){
            points.push_back({b[0], -b[2]});
            points.push_back({b[1], b[2]});
        }
        sort(points.begin(), points.end());
        int ongoingHeight = 0;
        for(int i = 0; i < points.size(); i++){
            int currentPoint = points[i].first;
            int heightAtCurrentPoint = points[i].second;

            if(heightAtCurrentPoint < 0){
                pq.insert(-heightAtCurrentPoint);
            } else {
                pq.erase(pq.find(heightAtCurrentPoint));
            }
            auto pqTop = *pq.rbegin();
            if(ongoingHeight != pqTop){
                ongoingHeight = pqTop;
                ans.push_back({currentPoint, ongoingHeight});
            }
        }

        return ans;
    }
};
```

OUTPUT:



Accepted Runtime: 0 ms

Input

buildings =

```
[[2,9,10],[3,7,15],[5,12,12],[15,20,10],[19,24,8]]
```

Output

```
[[2,10],[3,15],[7,12],[12,0],[15,10],[20,8],[24,0]]
```

Expected

```
[[2,10],[3,15],[7,12],[12,0],[15,10],[20,8],[24,0]]
```