# NAME- Akshit Boparai| UID- 22BCS14939 | SECTION- 601/A 1 CONVERT SORTED ARRAY TO BINARY SEARCH TREE

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; // Base case

int mid = left + (right - left) / 2; // Find the middle element

TreeNode\* root = new TreeNode(nums[mid]);

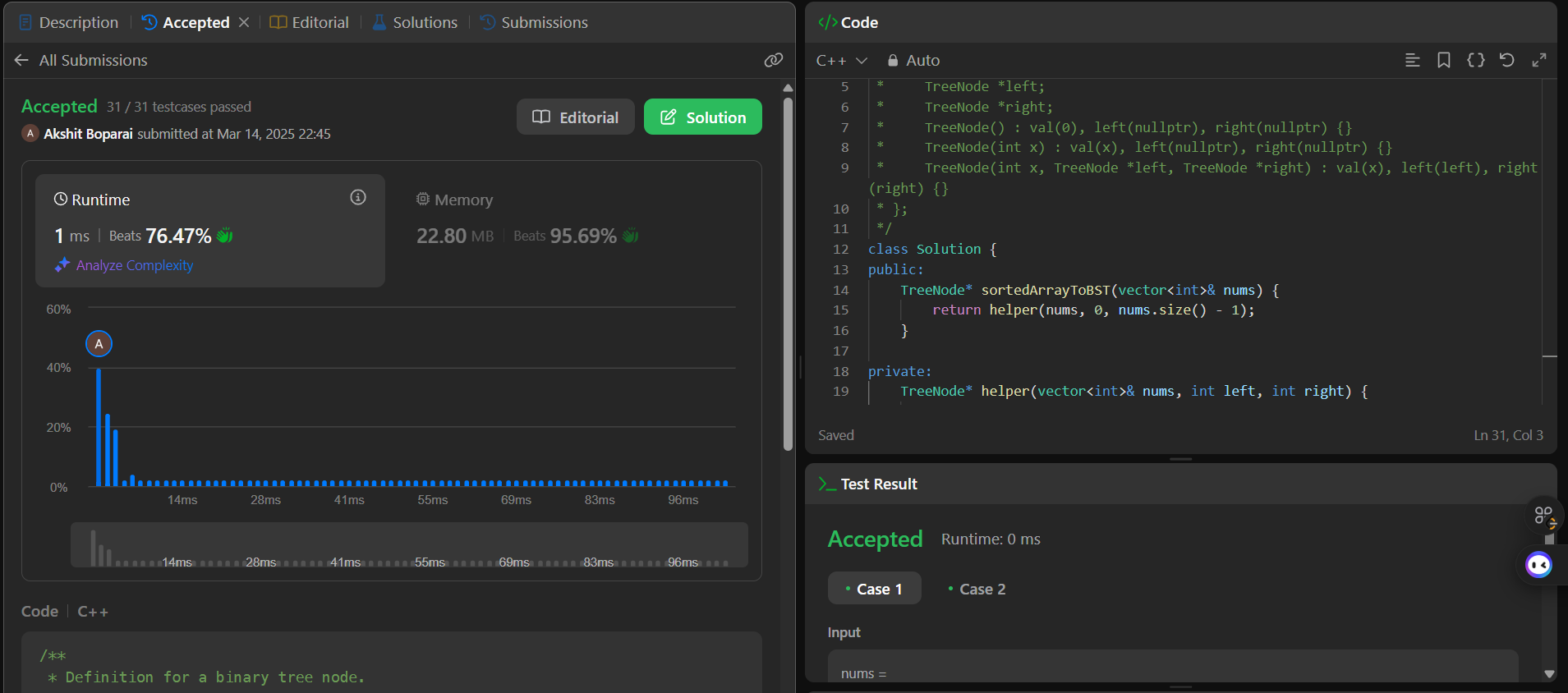
root->left = helper(nums, left, mid - 1);

root->right = helper(nums, mid + 1, right);

return root;

}

};



# NUMBER OF 1 BITS

class Solution {

public:

int hammingWeight(uint32\_t n) {

int count = 0;

while (n) {

n &= (n - 1); // Clear the lowest set bit

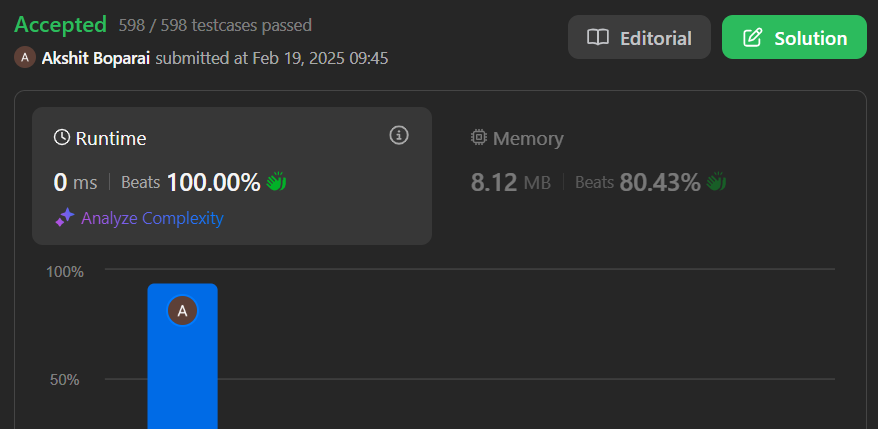
count++;

}

return count;

}

};



# SORT AN ARRAY

class Solution {

public:

vector<int> sortArray(vector<int>& nums) {

vector<int> counting(2 \* 50000 + 1, 0);

for (int num : nums) {

counting[num + 50000]++;

}

int write\_ind = 0;

for (int number\_ind = 0; number\_ind < counting.size(); ++number\_ind) {

int freq = counting[number\_ind];

while (freq != 0) {

nums[write\_ind] = number\_ind - 50000;

write\_ind++;

freq--;

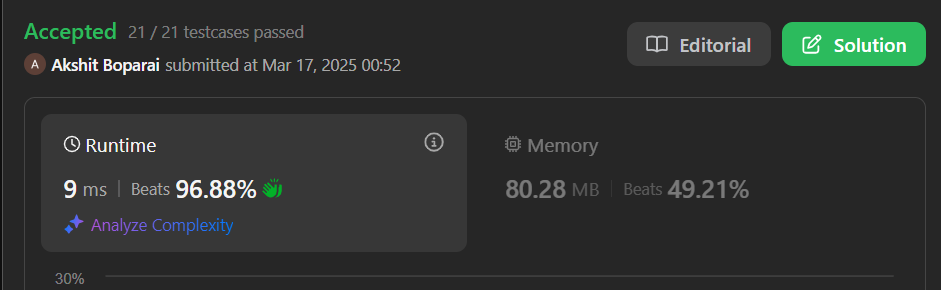
}

}

return nums;

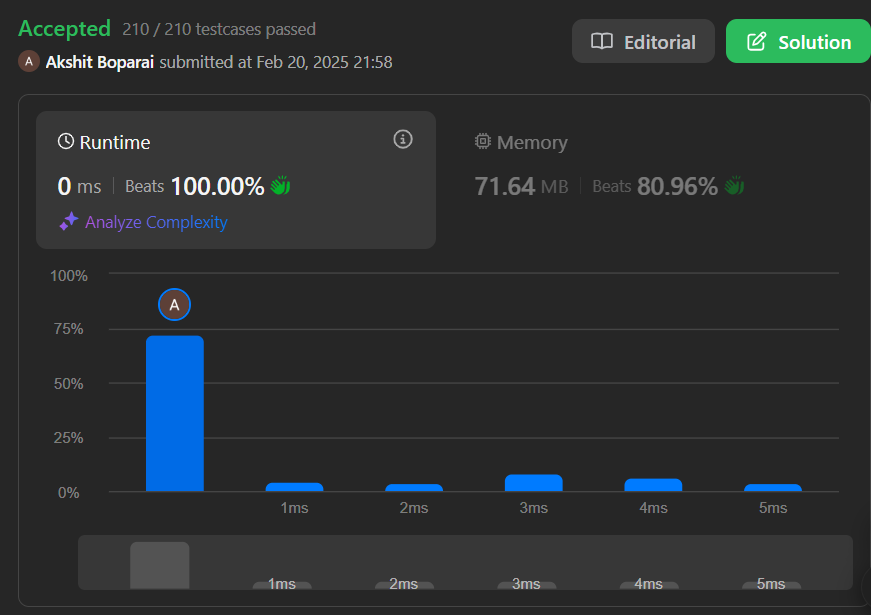
}

};



# MAXIMUM SUBARRAY

class Solution {public: int maxSubArray(vector<int>& nums) { int current\_sum = nums[0]; int max\_sum = nums[0]; for (size\_t i = 1; i < nums.size(); ++i) { current\_sum = max(nums[i], current\_sum + nums[i]); max\_sum = max(max\_sum, current\_sum); } return max\_sum; }};



# BEAUTIFUL ARRAY

class Solution {

public:

vector<int> beautifulArray(int n) {

if (n == 1) {

return {1};

}

// Divide into odd and even parts

vector<int> odd = beautifulArray((n + 1) / 2); // Odd numbers

vector<int> even = beautifulArray(n / 2); // Even numbers

// Combine: odd \* 2 - 1 and even \* 2

vector<int> result;

for (int num : odd) {

result.push\_back(num \* 2 - 1);

}

for (int num : even) {

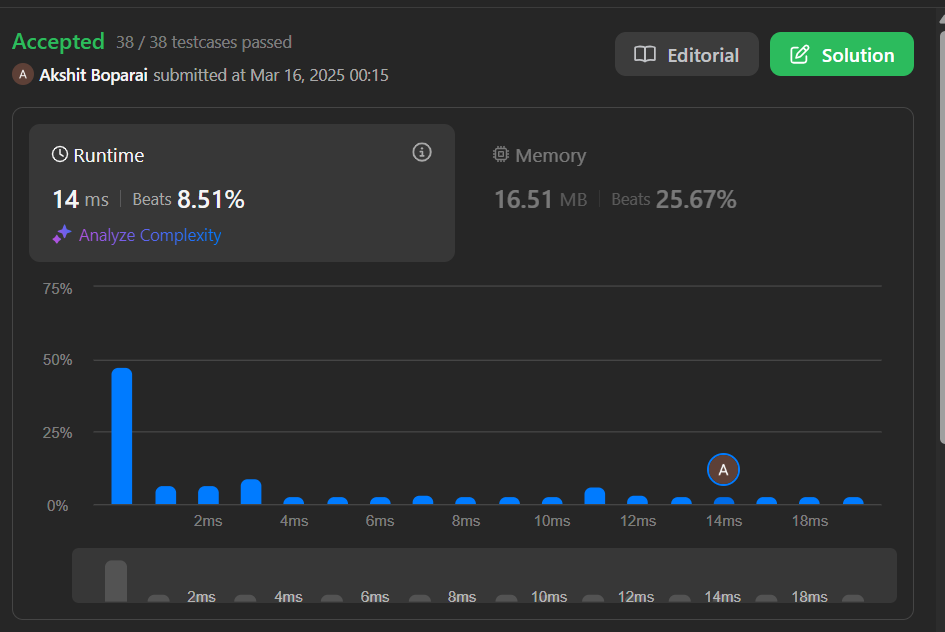
result.push\_back(num \* 2);

}

return result;

}

};



## SUPER POW

class Solution {

public:

    int modPow(int a, int b, int mod) {

    int result = 1;

    a %= mod;

    while (b > 0) {

        if (b % 2 == 1) {

            result = (result \* a) % mod;

        }

        a = (a \* a) % mod;

        b /= 2;

    }

    return result;

    }

    int superPow(int a, vector<int>& b) {

    const int mod = 1337;

    int result = 1;

    for (int digit : b) {

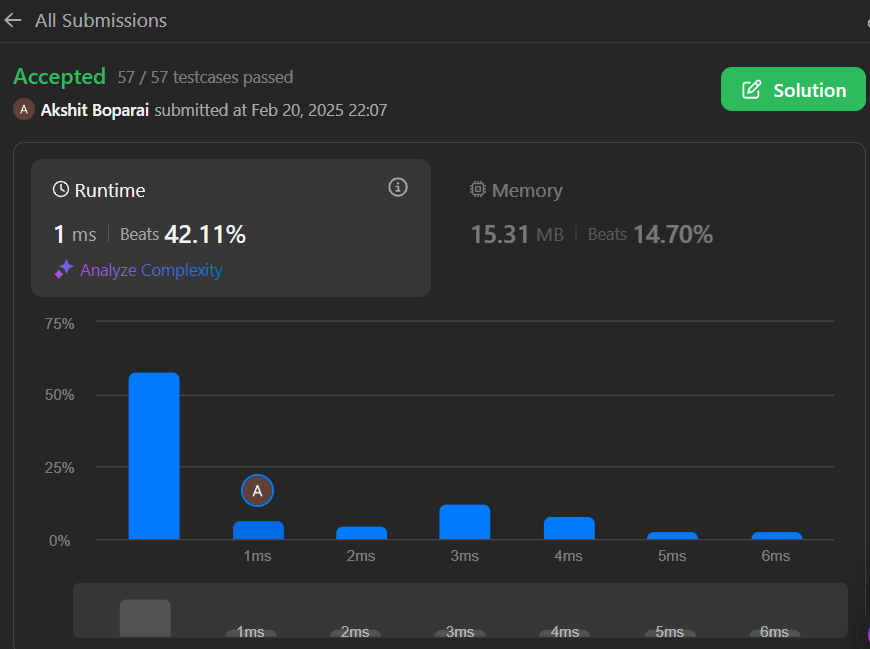
        result = modPow(result, 10, mod) \* modPow(a, digit, mod) % mod;

    }

    return result;

    }

};



## SKYLINE PROBLEM

class Solution {public: vector<vector<int>> getSkyline(vector<vector<int>>& buildings) { vector<vector<int>> result; vector<pair<int, int>> events; for (const auto& building : buildings) { int left = building[0], right = building[1], height = building[2]; events.push\_back({left, -height}); events.push\_back({right, height}); } sort(events.begin(), events.end()); multiset<int> heights = {0}; int prevMax = 0; for (const auto& event : events) { int x = event.first, height = event.second; if (height < 0) { heights.insert(-height); } else { heights.erase(heights.find(height)); } int currMax = \*heights.rbegin(); if (currMax != prevMax) { result.push\_back({x, currMax}); prevMax = currMax; } } return result; }};

