

Nandini Nandwani

22BCS11637

## 108. Convert Sorted Array to Binary Search Tree

Accepted 31 / 31 testcases passed  
Nandini Nandwani submitted at Mar 19, 2025 11:35

Runtime: 7 ms | Beats 14.07%  
Memory: 23.07 MB | Beats 23.38%

```
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;
    }
};
```

Accepted Runtime: 0 ms

## 191. Number of 1 Bits

Accepted 598 / 598 testcases passed  
Nandini Nandwani submitted at Mar 19, 2025 11:37

Runtime: 0 ms | Beats 100.00%  
Memory: 8.27 MB | Beats 47.36%

```
class Solution {
public:
    int hammingWeight(uint32_t n) {
        int count = 0;
        while (n) {
            count += (n & 1);
            n >>= 1;
        }
        return count;
    }
};
```

## 912. Sort an Array

Accepted 21 / 21 testcases passed  
Nandini Nandwani submitted at Mar 19, 2025 11:40

Runtime: 253 ms | Beats 49.57%  
Memory: 146.62 MB | Beats 41.66%

```
class Solution {
public:
    void merge(vector<int>&nums, int start, int mid, int end) {
        vector<int> temp(end - start + 1);
        int left = start, right = mid + 1, index = 0;
        while (left <= mid && right <= end) {
            if (nums[left] <= nums[right]) {
                temp[index] = nums[left];
                index++; left++;
            } else {
                temp[index] = nums[right];
                index++; right++;
            }
        }
    }
};
```

### 53. [Maximum Subarray](#)

The screenshot shows the LeetCode submission interface for the "Maximum Subarray" problem. On the left, the "Accepted" status is shown with 210/210 testcases passed. The submission was made by Nandini Nandwani on Mar 19, 2025 at 11:42. The runtime is 0 ms, beating 100.00% of solutions. The memory usage is 71.92 MB, beating 6.68% of solutions. On the right, the C++ code is displayed, showing a class Solution with a public method maxSubArray that uses a single loop to calculate the maximum sum of a contiguous subarray.

```
1 class Solution {
2 public:
3     int maxSubArray(vector<int>& nums) {
4         int maxSum = nums[0];
5         int currentSum = 0;
6
7         for (int n : nums) {
8             currentSum = max(n, currentSum + n);
9             maxSum = max(maxSum, currentSum);
10        }
11
12        return maxSum;
13    }
14 };
15
```

### 932. [Beautiful Array](#)

The screenshot shows the LeetCode submission interface for the "Beautiful Array" problem. On the left, the "Accepted" status is shown with 38/38 testcases passed. The submission was made by Nandini Nandwani on Mar 19, 2025 at 11:45. The runtime is 0 ms, beating 100.00% of solutions. The memory usage is 10.18 MB, beating 40.90% of solutions. On the right, the C++ code is displayed, showing a class Solution with a public method beautifulArray that generates a beautiful array by iteratively adding elements to a vector.

```
1 class Solution {
2 public:
3     vector<int> beautifulArray(int n) {
4         vector<int> answer = {1};
5
6         while (answer.size() < n) {
7             vector<int> temp;
8             for (int num : answer) {
9                 if (num * 2 - 1 <= n) {
10                     temp.push_back(num * 2 - 1);
11                 }
12             }
13             for (int num : answer) {
14                 if (num * 2 <= n) {
15                     temp.push_back(num * 2);
16                 }
17             }
18             answer = temp;
19         }
20         return answer;
21     }
22 };
23
```

### 372. [Super Pow](#)

The screenshot shows the LeetCode submission interface for the "Super Pow" problem. On the left, the "Accepted" status is shown with 57/57 testcases passed. The submission was made by Nandini Nandwani on Mar 19, 2025 at 11:46. The runtime is 0 ms, beating 100.00% of solutions. The memory usage is 15.10 MB, beating 83.16% of solutions. On the right, the C++ code is displayed, showing a class Solution with a public method pow and a method superPow that uses modular arithmetic to calculate the result.

```
1 class Solution {
2 public:
3     const int MOD = 1337;
4
5     int pow(int a, int b) {
6         int result = 1;
7         a %= MOD;
8         for (int i = 0; i < b; i++) {
9             result = (result * a) % MOD;
10        }
11        return result;
12    }
13
14    int superPow(int a, vector<int>& b) {
15        int result = 1;
16        for (int i = b.size() - 1; i >= 0; i--) {
17            result = (result * pow(a, b[i])) % MOD;
18        }
19        return result;
20    }
21 };
22
```

### 218. [The Skyline Problem](#)

The screenshot shows the LeetCode submission interface for the "The Skyline Problem". On the left, the "Accepted" status is shown with 44/44 testcases passed. The submission was made by Nandini Nandwani on Mar 19, 2025 at 11:47. The runtime is 17 ms, beating 57.16% of solutions. The memory usage is 27.86 MB, beating 63.33% of solutions. On the right, the C++ code is displayed, showing a class Solution with a public method getSkyline that uses a priority queue to calculate the skyline from a list of buildings.

```
1 class Solution {
2 public:
3     vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
4         vector<pair<int, int>> events;
5         priority_queue<int> heights;
6         vector<vector<int>> skyline;
7         unordered_map<int, int> heightCount;
8         for (const auto& b : buildings) {
9             events.emplace_back(b[0], b[2]);
10            events.emplace_back(b[1], b[2]);
11        }
12        sort(events.begin(), events.end());
13        heights.push(0);
14        int prevHeight = 0;
15        for (const auto& [x, h] : events) {
16            if (h < 0) {
17                heights.pop();
18            } else {
19                heights.push(h);
20                if (heights.top() > prevHeight) {
21                    skyline.push_back({x, heights.top()});
22                    prevHeight = heights.top();
23                }
24            }
25        }
26        return skyline;
27    }
28 };
29
```