

Name: Akshita Sood

Uid : 22BCS14993

Section: FL_lot 601 'A'

1. Convert Sorted Array to Binary Search Tree

```
class Solution {
```

```
public:
```

```
    TreeNode* sortedArrayToBST(vector<int>& nums) {  
        return create(nums, 0, nums.size() - 1);  
    }
```

```
private:
```

```
    TreeNode* create(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 = create(nums, left, mid - 1);  
        root->right = create(nums, mid + 1, right);  
        return root;  
    } };
```

The screenshot displays a coding platform interface with the following components:

- Problem List:** Shows the current problem and navigation options.
- Accepted:** Indicates that the solution has passed all 31 test cases.
- Runtime:** 0 ms, Beats 100.00%.
- Memory:** 23.10 MB, Beats 12.23%.
- Code Editor:** Contains the C++ code for the solution, including the `sortedArrayToBST` method and the `create` helper function.
- Testcase:** Shows the input `nums = [-10, -3, 0, 5, 9]` and the output.
- Test Result:** Shows the result as `Accepted` with a runtime of 0 ms.

2. Number of 1 Bits

```
class Solution {
```

```

public:
    int hammingWeight(int n) {
        int count = 0;
        for(int i = 31; i >= 0; i--){
            if(((n >> i) & 1) == 1)
                count++;
        }
        return count;
    }
}

```

The screenshot shows the LeetCode submission interface for the 'hammingWeight' problem. The submission is marked as 'Accepted' with 598/598 test cases passed. The performance metrics are: Runtime 0 ms (Beats 100.00%) and Memory 8.16 MB (Beats 80.26%). The code is written in C++ and is shown in the right panel. The test case result for input n=11 is also shown, indicating it was accepted with a runtime of 0 ms.

3. [Sort an Array](#)

```

class Solution {
public:
    vector<int> sortArray(vector<int>& nums) {
        mergeSort(nums, 0, nums.size() - 1);
        return nums;
    }

private:
    void mergeSort(vector<int>& nums, int l, int r) {
        if (l >= r)
            return;
        const int m = (l + r) / 2;
        mergeSort(nums, l, m);
    }
}

```

```

mergeSort(nums, m + 1, r);
merge(nums, l, m, r);
}
void merge(vector<int>& nums, int l, int m, int r) {
    vector<int> sorted(r - l + 1);
    int k = 0;
    int i = l;
    int j = m + 1;
    while (i <= m && j <= r)
        if (nums[i] < nums[j])
            sorted[k++] = nums[i++];
        else
            sorted[k++] = nums[j++];
    while (i <= m)
        sorted[k++] = nums[i++];
    while (j <= r)
        sorted[k++] = nums[j++];
    copy(sorted.begin(), sorted.end(), nums.begin() + l);
}
};

```

The screenshot displays a coding platform interface with the following details:

- Problem List:** Includes tabs for Description, Editorial, Solutions, Accepted, and Submissions.
- Submission Status:** "Accepted" with 21/21 testcases passed. Submitted by `akshita_sood` on Mar 18, 2025 at 10:51.
- Runtime Performance:** 248 ms, Beats 50.29%.
- Memory Performance:** 146.69 MB, Beats 41.83%.
- Code Editor:** Shows a C++ solution with the following code:


```

1 class Solution {
2 public:
3     vector<int> sortArray(vector<int>& nums) {
4         mergeSort(nums, 0, nums.size() - 1);
5         return nums;
6     }
7 }
8 private:

```
- Testcase Results:** Shows "Accepted" with a runtime of 0 ms. Includes tabs for Case 1 and Case 2.
- Input:** `nums = [5, 2, 3, 1]`
- Output:** (Empty field)

4. [Maximum Subarray](#)

```

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

```

The screenshot shows a C++ IDE interface. On the left, the 'Accepted' status is confirmed with '210 / 210 testcases passed'. The runtime is '0 ms' (Beats 100.00%) and memory is '71.63 MB' (Beats 81.06%). A bar chart shows the user's performance relative to others. The main editor displays the following C++ code:

```

4 #include <limits.h> // For INT_MIN
5 #include <vector>
6
7 class Solution {
8 public:
9     int maxSubArray(std::vector<int>& nums) {
10         int maxSum = INT_MIN, currSum = 0;
11         for (int num : nums) {
12             currSum = std::max(num, currSum + num);
13             maxSum = std::max(maxSum, currSum);
14         }
15     }
16 };

```

Below the code, the 'Testcase' section shows 'Accepted' with 'Runtime: 0 ms'. The input for 'Case 1' is:

```

nums =
[-2, 1, -3, 4, -1, 2, 1, -5, 4]

```

5. [Beautiful Array](#)

```

class Solution {
public:
    vector<int> beautifulArray(int n) {
        vector<int> arr(n);
        iota(arr.begin(), arr.end(), 1);
        divide(arr, 0, n - 1, 1);
        return arr;
    }
private:
    void divide(vector<int>& arr, int l, int r, int mask) {
        if (l >= r)
            return;
    }
};

```

```

const int m = partition(arr, l, r, mask);
divide(arr, l, m, mask << 1);
divide(arr, m + 1, r, mask << 1);
}
int partition(vector<int>& arr, int l, int r, int mask) {
    int nextSwapped = l;
    for (int i = l; i <= r; ++i)
        if (arr[i] & mask)
            swap(arr[i], arr[nextSwapped++]);
    return nextSwapped - 1;
}
};

```

The screenshot shows a LeetCode submission interface. On the left, the 'Accepted' status is confirmed with '38 / 38 testcases passed'. The runtime is 2 ms, beating 45.73% of solutions, and the memory usage is 9.65 MB, beating 86.89%. A bar chart at the bottom left shows the distribution of runtime performance. The main area displays the C++ code for a 'beautifulArray' problem, which includes a 'divide' function and a 'beautifulArray' method. The test results section shows 'Accepted' with a runtime of 0 ms for two test cases. The input for the test cases is 'n = 4'.

6. Super Pow

```

class Solution {
public:
    int superPow(int a, vector<int>& b) {
        int ans = 1;
        a %= kMod;
        for (const int i : b)
            ans = modPow(ans, 10) * modPow(a, i) % kMod;
        return ans;
    }
private:

```

```

static constexpr int kMod = 1337;
long modPow(long x, long n) {
    if (n == 0)
        return 1;
    if (n % 2 == 1)
        return x * modPow(x % kMod, (n - 1)) % kMod;
    return modPow(x * x % kMod, (n / 2)) % kMod;
}
};

```

7. The Skyline Problem

```

class Solution {
public:
    vector<vector<int>> getSkyline(const vector<vector<int>>& buildings) {
        const int n = buildings.size();
        if (n == 0)
            return {};
        if (n == 1) {
            const int left = buildings[0][0];
            const int right = buildings[0][1];
            const int height = buildings[0][2];
            return {{left, height}, {right, 0}};
        }
    }
}

```

```

const vector<vector<int>> left =
    getSkyline({buildings.begin(), buildings.begin() + n / 2});
const vector<vector<int>> right =
    getSkyline({buildings.begin() + n / 2, buildings.end()});
return merge(left, right);
}

```

private:

```

vector<vector<int>> merge(const vector<vector<int>>& left,
                        const vector<vector<int>>& right) {

```

```

    vector<vector<int>> ans;

```

```

    int i = 0;

```

```

    int j = 0;

```

```

    int leftY = 0;

```

```

    int rightY = 0;

```

```

    while (i < left.size() && j < right.size())

```

```

        if (left[i][0] < right[j][0]) {

```

```

            leftY = left[i][1];

```

```

            addPoint(ans, left[i][0], max(left[i++][1], rightY));

```

```

        } else {

```

```

            rightY = right[j][1];

```

```

            addPoint(ans, right[j][0], max(right[j++][1], leftY));

```

```

        }

```

```

    while (i < left.size())

```

```

        addPoint(ans, left[i][0], left[i++][1]);

```

```

    while (j < right.size())

```

```

        addPoint(ans, right[j][0], right[j++][1]);

```

```

    return ans;

```

```

}

```

```

void addPoint(vector<vector<int>>& ans, int x, int y) {

```

```

    if (!ans.empty() && ans.back()[0] == x) {

```

```

        ans.back()[1] = y;

```

```

    return;
}
if (!ans.empty() && ans.back()[1] == y)
    return;
ans.push_back({x, y});
}
};

```

The screenshot shows a LeetCode submission page for a C++ solution. The top navigation bar includes 'Problem List', 'Accepted', 'Editorial', 'Solutions', and 'Premium'. The main content area is divided into three sections:

- Submission Status:** Shows 'Accepted' for 44/44 testcases passed, submitted by 'akshit...' on Mar 18, 2025 at 11:13. Buttons for 'Editorial' and 'Solution' are visible.
- Performance Metrics:**
 - Runtime:** 290 ms, Beats 5.24%. A link to 'Analyze Complexity' is provided.
 - Memory:** 147.29 MB, Beats 10.43%. A link to 'Analyze Complexity' is provided.
 - A bar chart shows the distribution of runtime and memory usage across all submissions.
- Code Editor:**
 - Language: C++
 - Code:

```

1 class Solution {
2 public:
3     vector<vector<int>> getSkyline(const vector<vector<int>>& buildings) {
4         const int n = buildings.size();
5         if (n == 0)
6             return {};
7         if (n == 1) {
8             const int left = buildings[0][0];
9             const int right = buildings[0][1];
10            const int height = buildings[0][2];

```
 - Buttons: 'Run' and 'Submit'.
- Testcase Results:**
 - Status: 'Accepted', Runtime: 0 ms.
 - Testcases: Case 1, Case 2.
 - Input:

```

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

```