



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

ASSIGNMENT: 04

Student Name: Tushar Ranjan Jha

Branch: BE-CSE

Semester: 6th

Subject Name: AP LAB-II

UID: 22BCS15552

Section/Group: IOT- 606-B

Date of Submission: 17/03/2025

Subject Code: 22CSP-351

Problems:

1. Longest Nice Substring

2. Code:

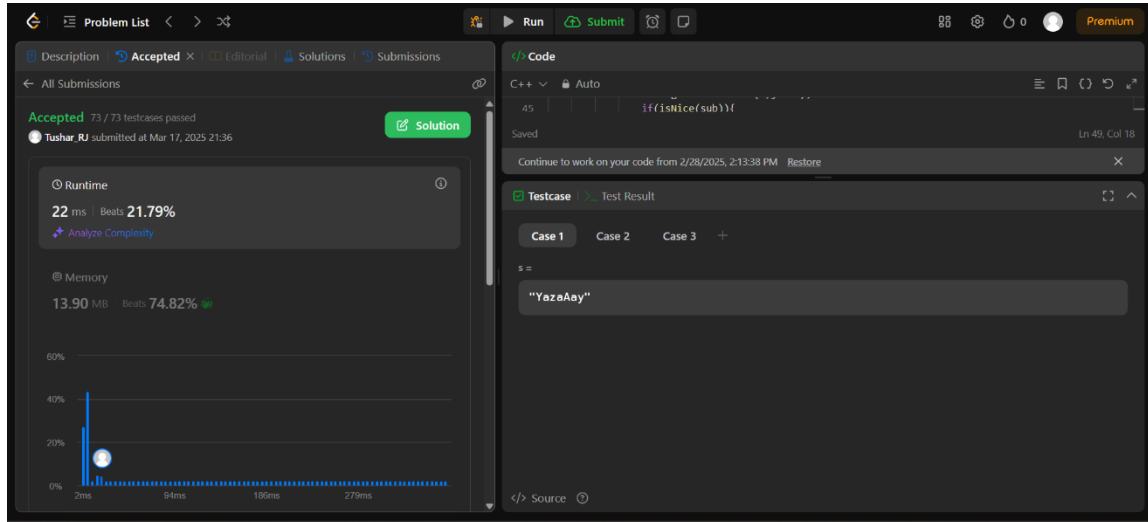
```
class Solution {
private:
    bool isNice(string& str){
        for(char c:str){
            if(islower(c)&&str.find(toupper(c))==string::npos){
                return false;
            }
            if(isupper(c)&&str.find(tolower(c))==string::npos){
                return false;
            }
        }
        return true;
    }
public:
    string longestNiceSubstring(string s) {
        string ans="";
        int n=s.length();
        for(int i=0;i<n;i++){
            for(int j=i;j<n;j++){
                string sub=s.substr(i,j-i+1);
                if(isNice(sub)){
                    if(sub.length()>ans.length()){
                        ans=sub;
                    }
                }
            }
        }
    }
}
```

```

    }
    return ans;
}
};

```

3. Submission Screenshot:



2. Reverse Bits:

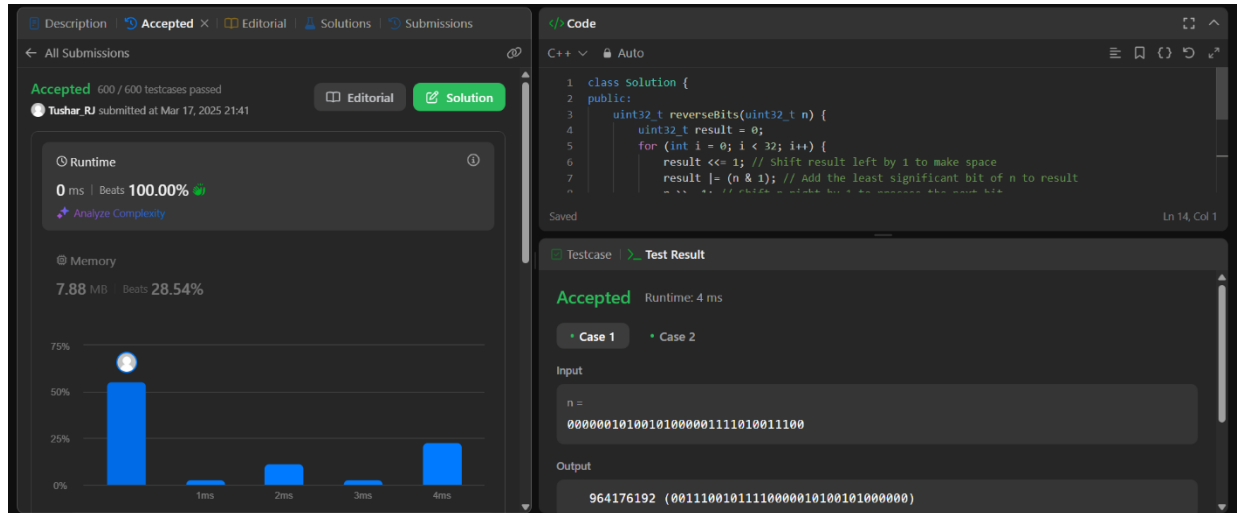
1. Code:

```

class Solution {
public:
    uint32_t reverseBits(uint32_t n) {
        uint32_t result = 0;
        for (int i = 0; i < 32; i++) {
            result <<= 1; // Shift result left by 1 to make space
            result |= (n & 1); // Add the least significant bit of n to result
            n >>= 1; // Shift n right by 1 to process the next bit
        }
        return result;
    }
};

```

2. Submission Screenshot:



3. Number of 1 Bits:

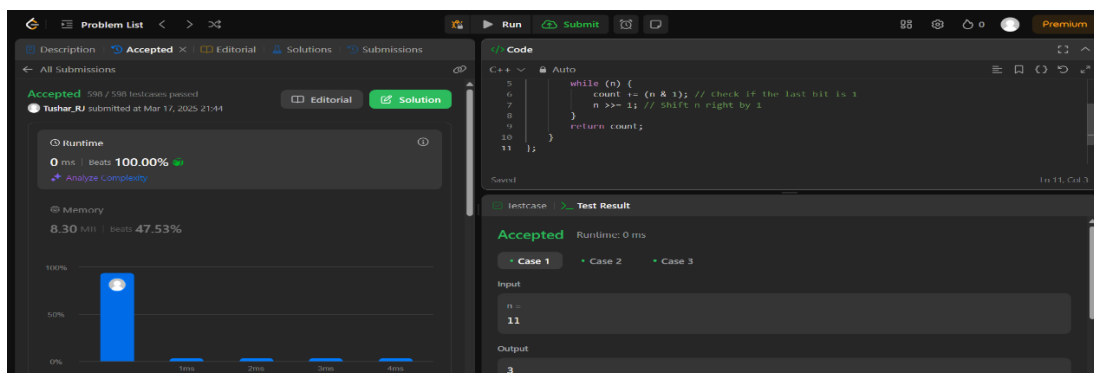
i. Code:

```

class Solution {
public:
    int hammingWeight(uint32_t n) {
        int count = 0;
        while (n) {
            count += (n & 1); // Check if the last bit is 1
            n >>= 1; // Shift n right by 1
        }
        return count;
    }
};

```

ii. Submission Screenshot:

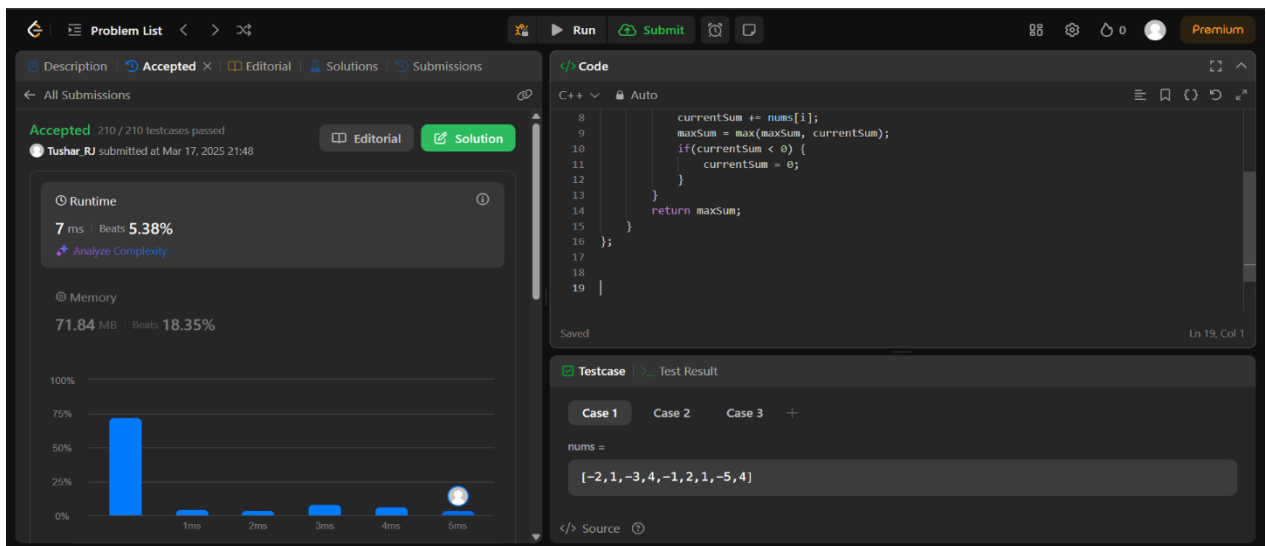


4. Maximum Subarray:

i. Code:

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

ii. Submission Screenshot:



5. Search a 2D Matrix II:

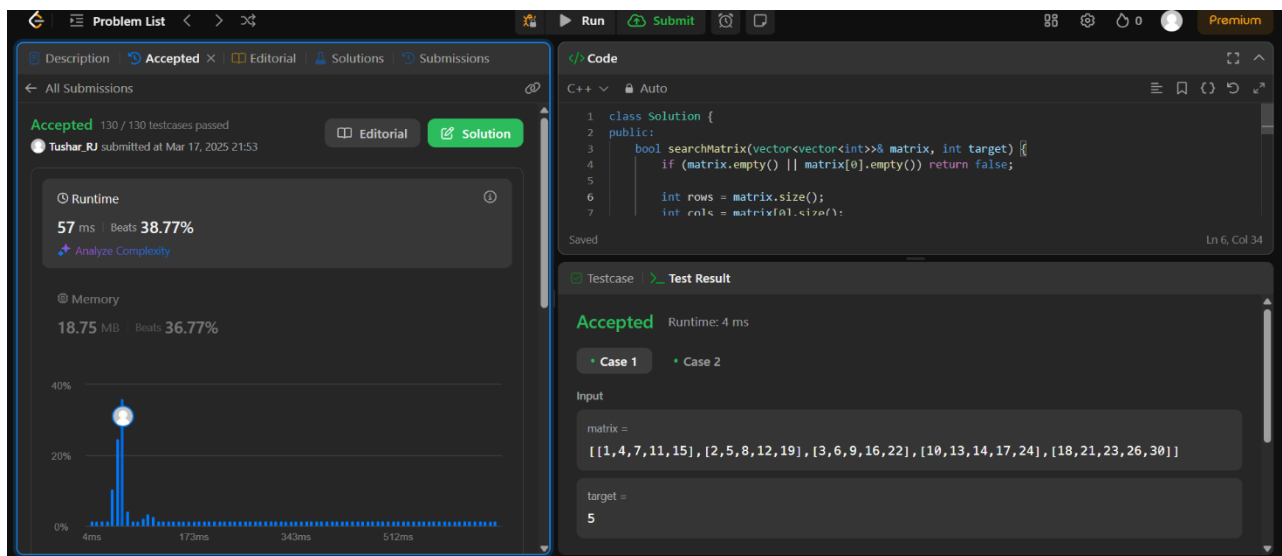
i. Code:

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        if (matrix.empty() || matrix[0].empty()) return false;

        int rows = matrix.size();
        int cols = matrix[0].size();
        int row = 0, col = cols - 1; // Start from the top-right corner

        while (row < rows && col >= 0) {
            if (matrix[row][col] == target) {
                return true; // Target found
            } else if (matrix[row][col] > target) {
                col--; // Move left
            } else {
                row++; // Move down
            }
        }
        return false; // Target not found
    }
};
```

ii. Submission Screenshot:



The screenshot displays a code submission interface. On the left, the 'Submissions' tab is active, showing 'Accepted' status for 130/130 testcases. The submission was made by 'Tushar_RU' on Mar 17, 2025, at 21:53. The 'Runtime' section shows 57 ms and Beats 38.77%. The 'Memory' section shows 18.75 MB and Beats 36.77%. A bar chart shows the runtime distribution. On the right, the 'Code' tab is active, displaying the C++ code for the 'searchMatrix' function. Below the code, the 'Testcase' tab is active, showing 'Accepted' status for 'Case 1' with a runtime of 4 ms. The input for Case 1 is a 4x4 matrix and a target value of 5.

Submissions

Accepted 130 / 130 testcases passed

Tushar_RU submitted at Mar 17, 2025 21:53

Runtime: 57 ms | Beats 38.77%

Memory: 18.75 MB | Beats 36.77%

Code

```
1 class Solution {
2 public:
3     bool searchMatrix(vector<vector<int>>& matrix, int target) {
4         if (matrix.empty() || matrix[0].empty()) return false;
5
6         int rows = matrix.size();
7         int cols = matrix[0].size();
```

Testcase

Accepted Runtime: 4 ms

Case 1

Input

matrix =

```
[[1,4,7,11,15], [2,5,8,12,19], [3,6,9,16,22], [10,13,14,17,24], [18,21,23,26,30]]
```

target =

```
5
```

6. Super Pow:

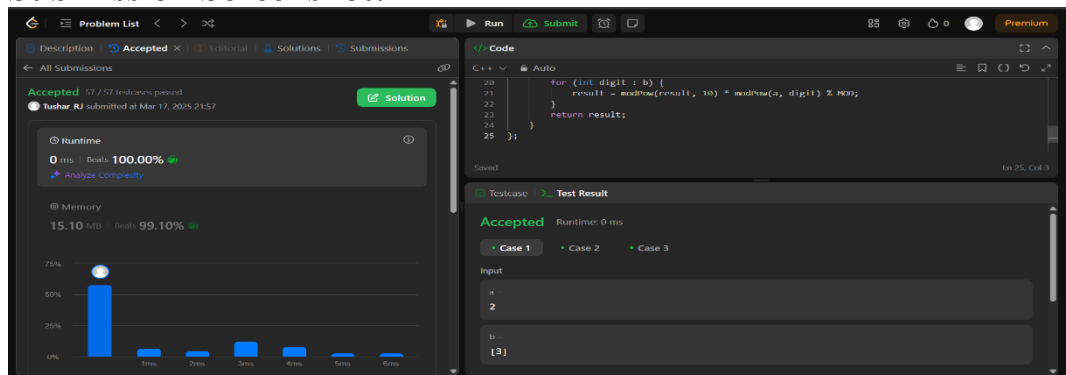
i. Code:

```
class Solution {
public:
    const int MOD = 1337;

    int modPow(int a, int b) {
        int result = 1;
        a %= MOD;
        while (b) {
            if (b % 2 == 1) {
                result = (result * a) % MOD;
            }
            a = (a * a) % MOD;
            b /= 2;
        }
        return result;
    }

    int superPow(int a, vector<int>& b) {
        int result = 1;
        for (int digit : b) {
            result = modPow(result, 10) * modPow(a, digit) % MOD;
        }
        return result;
    }
};
```

ii. Submission Screenshot:

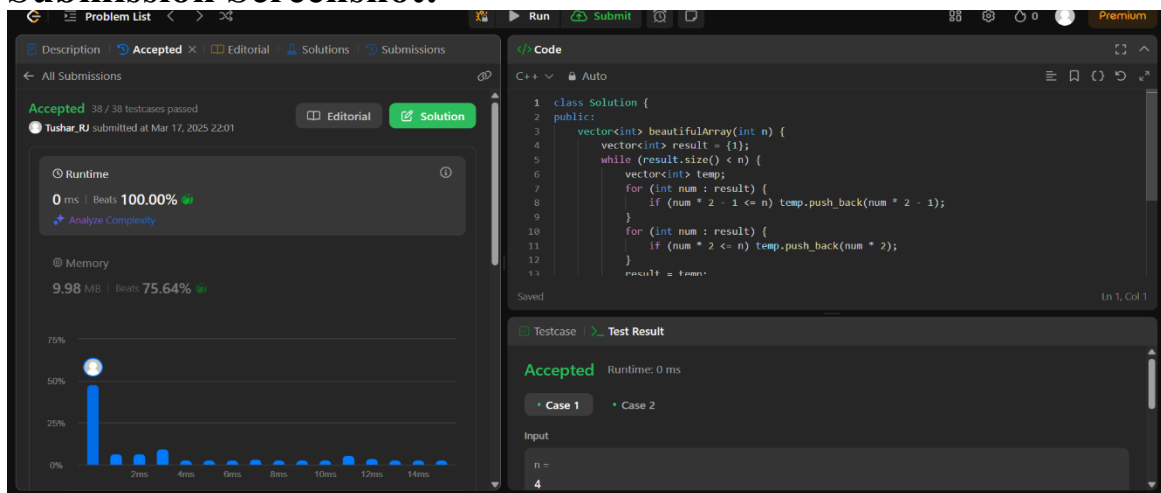


7. Beautiful Array:

Code:

```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        vector<int> result = {1};
        while (result.size() < n) {
            vector<int> temp;
            for (int num : result) {
                if (num * 2 - 1 <= n) temp.push_back(num * 2 - 1);
            }
            for (int num : result) {
                if (num * 2 <= n) temp.push_back(num * 2);
            }
            result = temp;
        }
        return result;
    }
};
```

Submission Screenshot:



8. The Skyline Problem:

Code:

```
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<pair<int, int>> events;
        for (const auto& b : buildings) {
            events.emplace_back(b[0], -b[2]); // Start of a building, store height as
negative
            events.emplace_back(b[1], b[2]); // End of a building, store height as
positive
        }

        sort(events.begin(), events.end()); // Sort by x-coordinate, process starts
before ends

        multiset<int> heights = {0};
        vector<vector<int>> result;
        int prevMaxHeight = 0;

        for (const auto& e : events) {
            int x = e.first, h = e.second;
            if (h < 0) {
                heights.insert(-h); // Insert new height
            } else {
                heights.erase(heights.find(h)); // Remove height
            }

            int currentMaxHeight = *heights.rbegin();
            if (currentMaxHeight != prevMaxHeight) {
                result.push_back({x, currentMaxHeight});
                prevMaxHeight = currentMaxHeight;
            }
        }

        return result;
    }
};
```



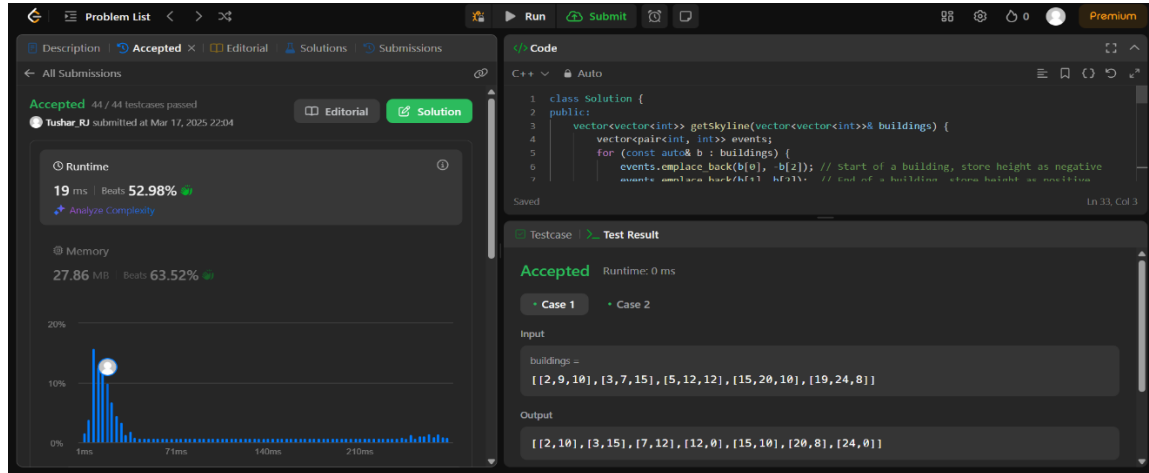
```

    }
    }

    return result;
}
};

```

Submission Screenshot:



9. Reverse Pairs:

Code:

```

class Solution {
public:
    int reversePairs(vector<int>& nums) {
        function<int(vector<int>&, int, int)> mergeSortAndCount = [&](vector<int>& nums, int
left, int right) {
            if (left >= right) return 0;
            int mid = left + (right - left) / 2;
            int count = mergeSortAndCount(nums, left, mid) + mergeSortAndCount(nums, mid + 1,
right);

            int j = mid + 1;
            for (int i = left; i <= mid; i++) {
                while (j <= right && nums[i] > 2LL * nums[j]) {
                    j++;
                }
                count += (j - (mid + 1));
            }

            vector<int> temp;

```

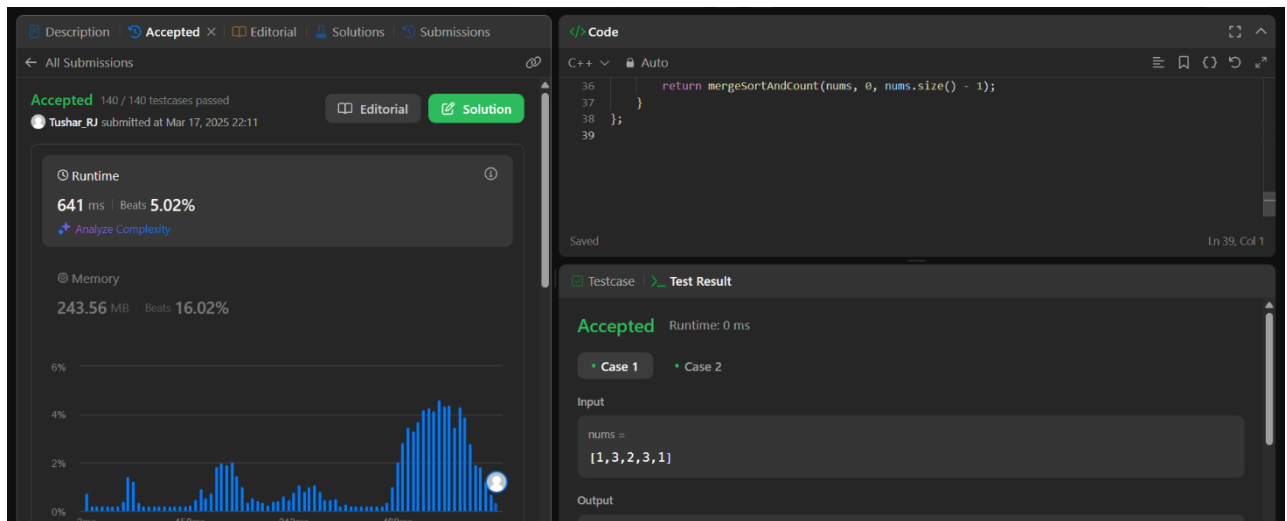
```
int i = left, k = mid + 1;
while (i <= mid && k <= right) {
    if (nums[i] <= nums[k]) {
        temp.push_back(nums[i++]);
    } else {
        temp.push_back(nums[k++]);
    }
}
while (i <= mid) temp.push_back(nums[i++]);
while (k <= right) temp.push_back(nums[k++]);

for (int i = left; i <= right; i++) {
    nums[i] = temp[i - left];
}

return count;
};

return mergeSortAndCount(nums, 0, nums.size() - 1);
}
};
```

Submission Screenshot:



10.Longest Increasing Subsequence II:

Code:

```
class Solution {
public:
    int lengthOfLIS(vector<int>& nums, int k) {
        int n = nums.size();
        int maxVal = 100000;
        vector<int> tree(4 * maxVal);

        function<void(int, int, int, int, int)> update =
            [&](int node, int start, int end, int idx, int val) {
                if (start == end) {
                    tree[node] = val;
                    return;
                }
                int mid = (start + end) / 2;
                if (idx <= mid) {
                    update(2 * node, start, mid, idx, val);
                } else {
                    update(2 * node + 1, mid + 1, end, idx, val);
                }
                tree[node] = max(tree[2 * node], tree[2 * node + 1]);
            };

        function<int(int, int, int, int, int)> query =
            [&](int node, int start, int end, int left, int right) {
                if (right < start || end < left) {
                    return 0;
                }
                if (left <= start && end <= right) {
                    return tree[node];
                }
                int mid = (start + end) / 2;
                return max(query(2 * node, start, mid, left, right),
                    query(2 * node + 1, mid + 1, end, left, right));
            };

        int maxLength = 0;
        for (int num : nums) {
            int left = max(1, num - k);
            int currentLength = query(1, 1, maxVal, left, num - 1) + 1;
            update(1, 1, maxVal, num, currentLength);
            maxLength = max(maxLength, currentLength);
        }
    }
};
```

```
        return maxLength;  
    }  
};
```

Submission screenshot:

The screenshot displays a LeetCode submission page for a C++ solution. The interface is divided into several sections:

- Top Bar:** Includes navigation links (Problem List, Run, Submit, etc.) and a Premium badge.
- Submission Status:** Shows "Accepted" with 84 / 84 testcases passed. The user "Tushar_RJ" submitted on Mar 17, 2025 at 22:22.
- Runtime and Memory:**
 - Runtime: 427 ms, Beats 6.42%.
 - Memory: 195.48 MB, Beats 11.60%.
- Code Editor:** Contains the following C++ code:

```
1 class Solution {  
2 public:  
3     int lengthOfLIS(vector<int>& nums, int k) {  
4         int n = nums.size();  
5         int maxVal = 100000;  
6         vector<int> tree(4 * maxVal);
```
- Testcase Results:** Shows "Accepted" with a runtime of 3 ms. Three test cases are listed: Case 1, Case 2, and Case 3.
- Input:** The input for Case 1 is:

```
nums =  
[4, 2, 1, 4, 3, 4, 5, 8, 15]  
  
k =  
3
```