



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Assignment

Name: Sargam Anand

Branch: BE-CSE

Semester: 6th

Subject Name: Advanced Programming

UID: 22BCS14851

Section/Group: 607/B

Date of Performance: 05/03/25

Subject Code: 22CSP-351

Code 1

```
string longestNiceSubstring(string s) {  
    if(s.size()<2)return "";  
    unordered_set<char>st;  
    for(auto ch:s)  
        st.insert(ch);  
    for(int i=0;i<s.size(); i++){  
        if(st.count(toupper(s[i]))&& st.count(tolower(s[i])))  
            continue;  
        string prev = longestNiceSubstring(s.substr(0,i));  
        string next = longestNiceSubstring(s.substr(i+1));  
        return prev.size()>=next.size()? prev: next;  
    }  
}
```

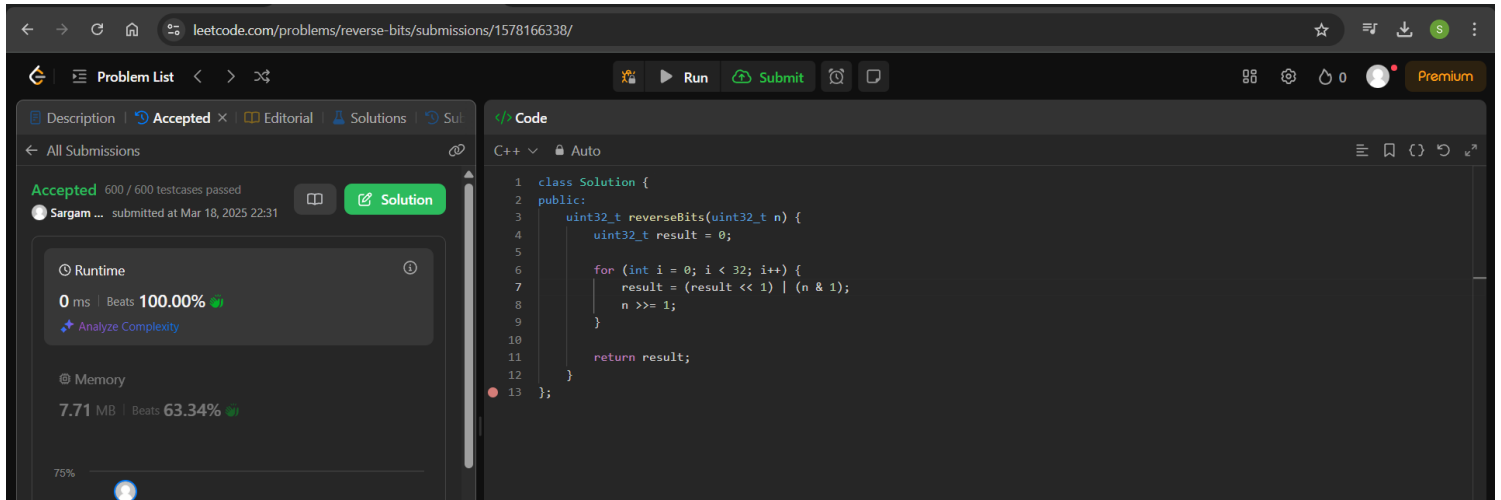
The screenshot shows a web browser with multiple tabs. The active tab is 'Longest Nice Substring - LeetC...'. The browser address bar shows 'leetcode.com/problems/longest-nice-substring/submissions/1578158329/'. The page displays the 'Longest Nice Substring' problem solution page. The submission status is 'Accepted' with 73/73 testcases passed. The user 'Sargam Anand' submitted the solution on Mar 18, 2025 at 22:25. The runtime is 6 ms, which beats 72.81% of other submissions. The memory usage is 14.34 MB, which beats 30.86% of other submissions. The code is written in C++ and matches the code provided in the assignment. The code is as follows:

```
1 class Solution {  
2 public:  
3     string longestNiceSubstring(string s) {  
4         if(s.size()<2)return "";  
5         unordered_set<char>st;  
6         for(auto ch:s)  
7             st.insert(ch);  
8         for(int i=0;i<s.size(); i++){  
9             if(st.count(toupper(s[i]))&& st.count(tolower(s[i])))  
10                continue;  
11             string prev = longestNiceSubstring(s.substr(0,i));  
12             string next = longestNiceSubstring(s.substr(i+1));  
13             return prev.size()>=next.size()? prev: next;  
14         }  
15     }  
16     return s;  
17 }  
18 };
```

Code 2

```
class Solution {
public:
    uint32_t reverseBits(uint32_t n) {
        uint32_t result = 0;

        for (int i = 0; i < 32; i++) {
            result = (result << 1) | (n & 1);
            n >>= 1;
        }
        return result;
    }
};
```



leetcodes.com/problems/reverse-bits/submissions/1578166338/

Problem List

Description | Accepted x | Editorial | Solutions | Submissions

All Submissions

Accepted 600 / 600 testcases passed

Sargam ... submitted at Mar 18, 2025 22:31

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

7.71 MB | Beats 63.34%

Code

```
1 class Solution {
2 public:
3     uint32_t reverseBits(uint32_t n) {
4         uint32_t result = 0;
5
6         for (int i = 0; i < 32; i++) {
7             result = (result << 1) | (n & 1);
8             n >>= 1;
9         }
10
11         return result;
12     }
13 };
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Code 3

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

The screenshot displays a LeetCode submission for the problem "Number of 1 Bits". The code is written in C++ and is accepted. The runtime is 0 ms, which beats 100.00% of other submissions. The memory usage is 8.30 MB, which beats 47.49% of other submissions. A bar chart shows the runtime distribution, with a single bar at 0 ms reaching 100%.

Runtime Performance:

- Runtime: 0 ms
- Beats: 100.00%
- Memory: 8.30 MB
- Beats: 47.49%

Code:

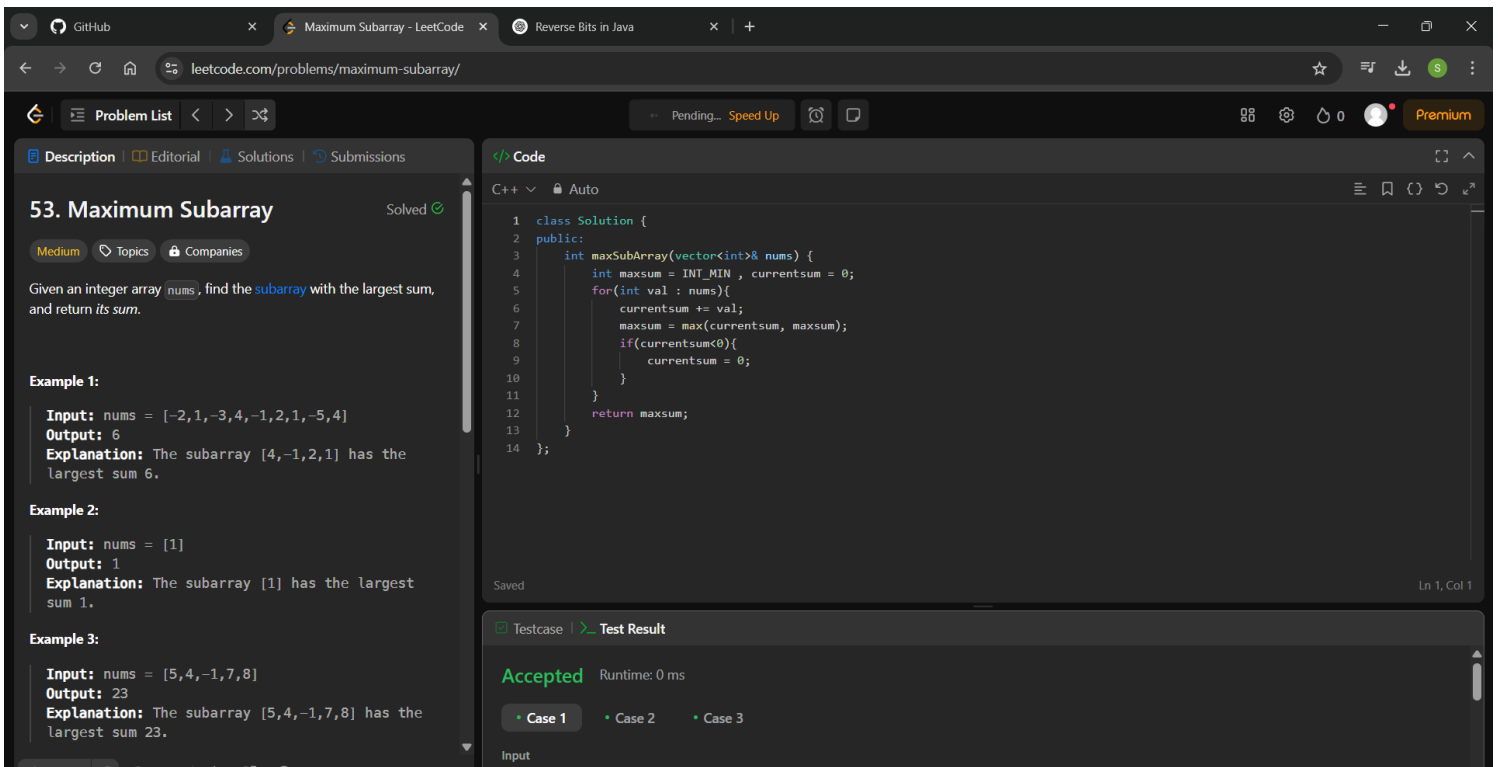
```
1 class Solution {  
2 public:  
3     int hammingWeight(int n) {  
4         int count = 0;  
5  
6         while (n != 0) {  
7             count += (n & 1);  
8             n >>= 1;  
9         }  
10  
11         return count;  
12     }  
13 };
```

Test Result:

Accepted Runtime: 0 ms

Code 4

```
class Solution {  
public:  
    int maxSubArray(vector<int>& nums) {  
        int maxsum = INT_MIN , currentsum = 0;  
        for(int val : nums){  
            currentsum += val;  
            maxsum = max(currentsum, maxsum);  
            if(currentsum<0){  
                currentsum = 0;  
            }  
        }  
        return maxsum;  
    }  
};
```



The screenshot shows a web browser with the URL `leetcode.com/problems/maximum-subarray/`. The page displays the problem description for "53. Maximum Subarray", which is marked as "Solved". The problem asks to find the subarray with the largest sum in a given integer array `nums`. Three examples are provided: Example 1 with `nums = [-2,1,-3,4,-1,2,1,-5,4]` and output 6; Example 2 with `nums = [1]` and output 1; and Example 3 with `nums = [5,4,-1,7,8]` and output 23. The solution is implemented in C++ in the "Code" editor, showing the same code as in the previous block. The "Testcase" tab is selected, showing "Accepted" status with a runtime of 0 ms. Three test cases are listed: Case 1, Case 2, and Case 3.



Code 5

```
int m = matrix.size(), n = matrix[0].size();
int row = 0, col = n - 1;

while (row < m && col >= 0) {
    if (matrix[row][col] == target) return true;
    else if (matrix[row][col] > target) col--;
    else row++;
}

return false;
```

The screenshot displays a web browser window with the LeetCode website. The address bar shows the URL: `leetcode.com/problems/search-a-2d-matrix-ii/submissions/1578175078/`. The page title is "Search a 2D Matrix II - LeetCode". The main content area shows the problem description, a "Solution" button, and a "Runtime" section. The "Runtime" section indicates that the solution is "Accepted" and has a runtime of 49 ms, which beats 74.97% of other submissions. The "Memory" section shows a memory usage of 18.53 MB, which beats 91.53% of other submissions. The code editor on the right shows the C++ solution for the problem, which is a binary search algorithm. The code is as follows:

```
1 class Solution {
2 public:
3     bool searchMatrix(vector<vector<int>>& matrix, int target) {
4         int m = matrix.size(), n = matrix[0].size();
5         int row = 0, col = n - 1;
6
7         while (row < m && col >= 0) {
8             if (matrix[row][col] == target) return true;
9             else if (matrix[row][col] > target) col--;
10            else row++;
11        }
12
13        return false;
14    }
15 }
16 ;
```

The bottom of the page shows a "Testcase" section with a "Test Result" button. The "Test Result" button is highlighted in green, indicating that the solution is accepted. The "Runtime" section also shows "Accepted" and "Runtime: 0 ms".



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Code 6

```
a %= 1337;
int res = 1;
for (int digit : b) {
    int curr = 1;
    for (int i = 0; i < 10; i++) {
        curr = (curr * res) % 1337;
    }
    for (int i = 0; i < digit; i++) {
        curr = (curr * a) % 1337;
    }
    res = curr;
}
return res;
```

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

- Problem List:** Shows the current problem as "Accepted" with a green checkmark.
- Code Editor:** Contains the C++ code for the solution, which is a class `Solution` with a public method `superPow`. The code implements the logic shown in the "Code 6" block.
- Runtime/Memory Analysis:** A panel on the left shows the solution's performance. It indicates a runtime of 0 ms and a memory usage of 15.34 MB, both of which are optimal (100.00% and 15.09% respectively). A bar chart below shows the distribution of runtime across different test cases.
- Test Result:** A section at the bottom right shows the test results for the solution. It indicates that the solution is "Accepted" and provides the runtime for each test case (Case 1, Case 2, Case 3).



Code 7

```
vector<int> res = {1};
while (res.size() < n) {
    vector<int> temp;
    for (int x : res) {
        if (2 * x - 1 <= n) temp.push_back(2 * x - 1);
    }
    for (int x : res) {
        if (2 * x <= n) temp.push_back(2 * x);
    }
    res = temp;
}
return res;
```

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

- Problem List:** A sidebar on the left showing the problem list.
- Description:** A tab showing the problem description.
- Accepted:** A green badge indicating the solution is accepted.
- Runtime:** A section showing the execution time of the solution: 2 ms, Beats 45.73%.
- Memory:** A section showing the memory usage of the solution: 10.19 MB, Beats 40.74%.
- Code:** A section showing the C++ code implementation of the solution.
- Testcase:** A section showing the test results for the solution.

The C++ code is as follows:

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

Code 8

```
vector<vector<int>> res;
multiset<int> heights = {0};
vector<pair<int, int>> points;

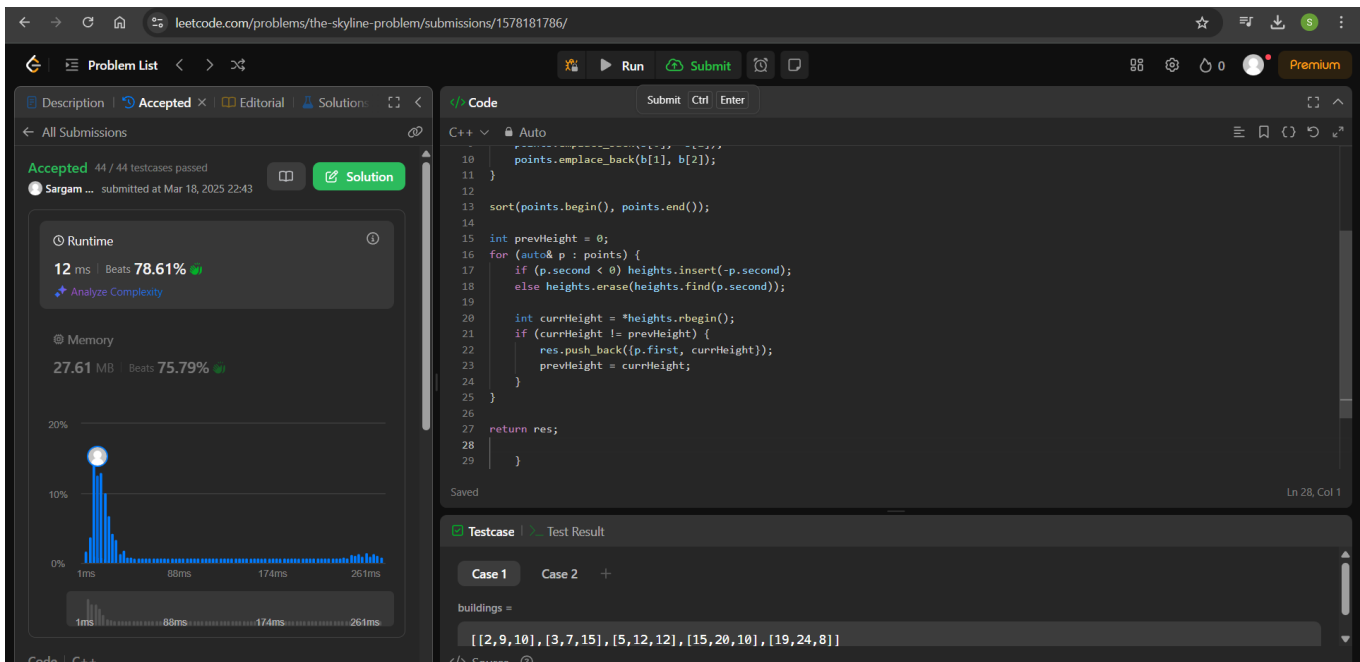
for (auto& b : buildings) {
    points.emplace_back(b[0], -b[2]);
    points.emplace_back(b[1], b[2]);
}

sort(points.begin(), points.end());

int prevHeight = 0;
for (auto& p : points) {
    if (p.second < 0) heights.insert(-p.second);
    else heights.erase(heights.find(p.second));

    int currHeight = *heights.rbegin();
    if (currHeight != prevHeight) {
        res.push_back({p.first, currHeight});
        prevHeight = currHeight;
    }
}

return res;
```



The screenshot shows a LeetCode submission for the problem "The Skyline Problem". The submission is accepted, with 44/44 test cases passed. The runtime is 12 ms, beating 78.61% of submissions. The memory usage is 27.61 MB, beating 75.79% of submissions. The code is written in C++ and implements the solution using a multiset and a vector of points.

Runtime: 12 ms | Beats 78.61%

Memory: 27.61 MB | Beats 75.79%

Code:

```
C++
10 points.emplace_back(b[1], b[2]);
11 }
12
13 sort(points.begin(), points.end());
14
15 int prevHeight = 0;
16 for (auto& p : points) {
17     if (p.second < 0) heights.insert(-p.second);
18     else heights.erase(heights.find(p.second));
19
20     int currHeight = *heights.rbegin();
21     if (currHeight != prevHeight) {
22         res.push_back({p.first, currHeight});
23         prevHeight = currHeight;
24     }
25 }
26
27 return res;
28
29 }
```

Testcase: Case 1

buildings =

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


Code 9

```
int merge(vector<int>& nums, int left, int mid, int right) {
    int count = 0, 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;
}

int mergeSort(vector<int>& nums, int left, int right) {
    if (left >= right) return 0;
    int mid = left + (right - left) / 2;
    int count = mergeSort(nums, left, mid) + mergeSort(nums, mid + 1, right);
    count += merge(nums, left, mid, right);
    return count;
}

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



Accepted 140 / 140 testcases passed
Sargam ... submitted on Mar 18, 2025 22:47

Runtime: 552 ms | Beats 25.67%
Memory: 243.61 MB | Beats 16.15%

Code (C++):

```
20 }  
21  
22 while (i <= mid) temp.push_back(nums[i++]);  
23 while (k <= right) temp.push_back(nums[k++]);  
24  
25 for (int i = left; i <= right; i++) nums[i] = temp[i - left];  
26  
27 return count;  
28 }  
29  
30 int mergeSort(vector<int>& nums, int left, int right) {  
31     if (left >= right) return 0;  
32     int mid = left + (right - left) / 2;  
33     int count = mergeSort(nums, left, mid) + mergeSort(nums, mid + 1, right);  
34     count += merge(nums, left, mid, right);  
35     return count;  
36 }  
37  
38
```

Code 10

```
class Solution {  
public:  
    int reversePairs(vector<int>& nums) {  
        return mergeSort(nums, 0, nums.size() - 1);  
    }  
  
private:  
    int merge(vector<int>& nums, int left, int mid, int right) {  
        int count = 0, 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++]);  
        }  
    }  
};
```



Accepted

140 / 140 testcases passed

Sargam ... submitted at Mar 18, 2025 22:47

Runtime

552 ms | Beats 25.67%

Analyze Complexity

Memory

243.61 MB | Beats 16.15%

6%

4%

2%

0%

3ms158ms313ms468ms

DescriptionAcceptedEditorialSolutions

All Submissions

C++Auto

```
20 }
21
22 while (i <= mid) temp.push_back(nums[i++]);
23 while (k <= right) temp.push_back(nums[k++]);
24
25 for (int i = left; i <= right; i++) nums[i] = temp[i - left];
26
27 return count;
28 }
29
30 int mergeSort(vector<int>& nums, int left, int right) {
31     if (left >= right) return 0;
32     int mid = left + (right - left) / 2;
33     int count = mergeSort(nums, left, mid) + mergeSort(nums, mid + 1, right);
34     count += merge(nums, left, mid, right);
35     return count;
36 }
37 };
38
```

SavedLn 38, Col 1

TestcaseTest Result

Case 1Case 2+