ASSIGNMENT – 4

Name: Piyush

Section: IOT_608

UID: 22BCS15782

Group: B

Solution 1:

```
class Solution {
public:
string longestNiceSubstring(string s) {
int n = s.length();
string result ="";
for(int i =0 ; i<n; ++i){
for (int j=i;j<n;++j){
string Sub = s.substr(i,j-i+1);
unordered_set<char> st(Sub.begin(),Sub.end());
bool isNice = true;

for(char c : Sub){
if(st.count(tolower(c)) == 0 || st.count(toupper(c))==0){
isNice = false;
break;
}
}
if(isNice && Sub.length()>result.length()){
```

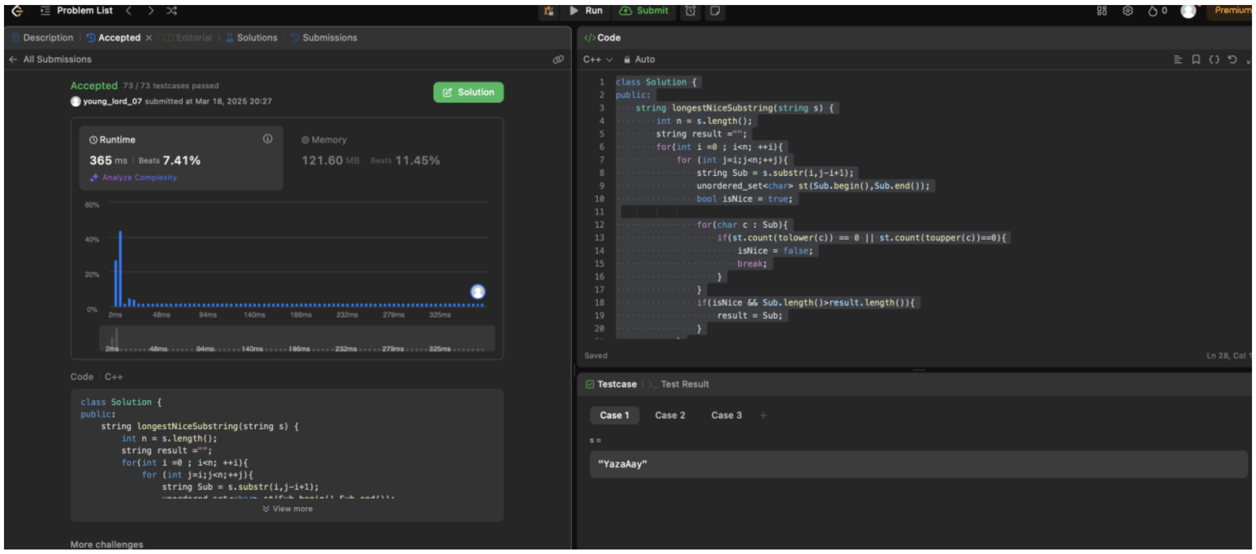
result = Sub;

}

}

} return result;

}};



Accepted 73 / 73 testcases passed
young_lord_07 submitted at Mar 18, 2025 20:27

Runtime 365 ms | Beats 7.41%

Memory 121.60 MB | Beats 11.45%

Code | C++

```
class Solution {
public:
    string longestNiceSubstring(string s) {
        int n = s.length();
        string result ="";
        for(int i =0 ; i<n; ++i){
            for (int j=i;j<n;++j){
                string Sub = s.substr(i,j-i+1);
```

Code

C++ Auto

```
class Solution {
public:
   string longestNiceSubstring(string s) {
        int n = s.length();
        string result ="";
        for(int i =0 ; i<n; ++i){
            for (int j=i;j<n;++j){
                string Sub = s.substr(i,j-i+1);
                unordered_set<char> st(Sub.begin(),Sub.end());
                bool isNice = true;

                for(char c : Sub){
                    if(st.count(tolower(c)) == 0 || st.count(toupper(c))==0){
                        isNice = false;
                        break;
                    }
                }
                if(isNice && Sub.length()>result.length()){
                    result = Sub;
                }
```

s =

"YazaAay"

Solution 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);

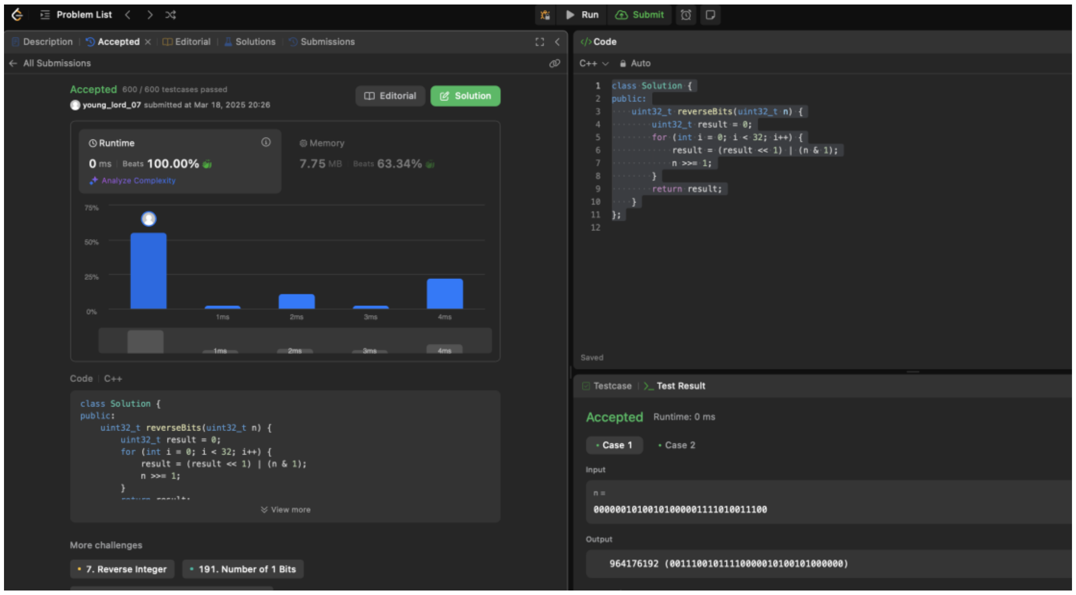
n >>= 1;

}

return result;

}

};



Accepted 600 / 600 testcases passed

young_lord_07 submitted at Mar 18, 2025 20:26

Runtime
0 ms | Beats 100.00%

Memory
7.75 MB | Beats 63.34%

Code | C++

```cpp
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;
    }
};
```

Accepted Runtime: 0 ms

Input

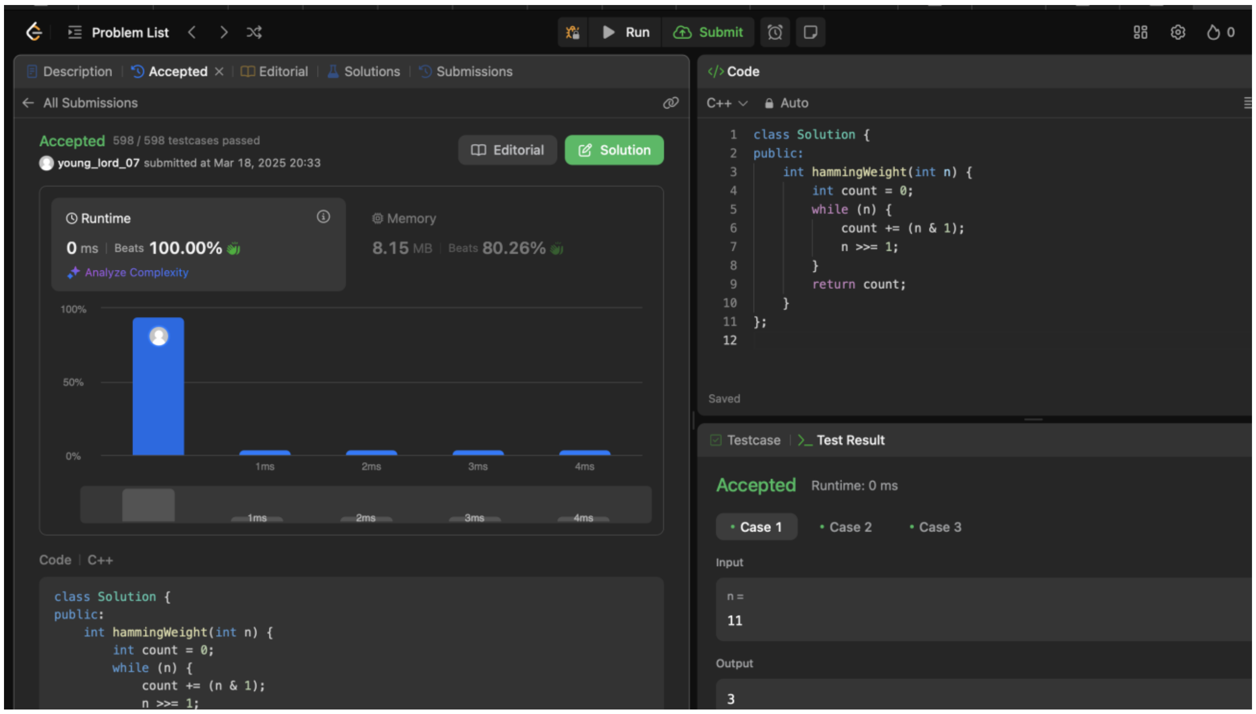
```
n =
00000010100101000001111010011100
```

Output

```
964176192 (00111001011110000010100101000000)
```

Solution 3:

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



Accepted 598 / 598 testcases passed

young_lord_07 submitted at Mar 18, 2025 20:33

Runtime
0 ms | Beats 100.00%

Memory
8.15 MB | Beats 80.26%

Code | C++

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

Accepted Runtime: 0 ms

Input

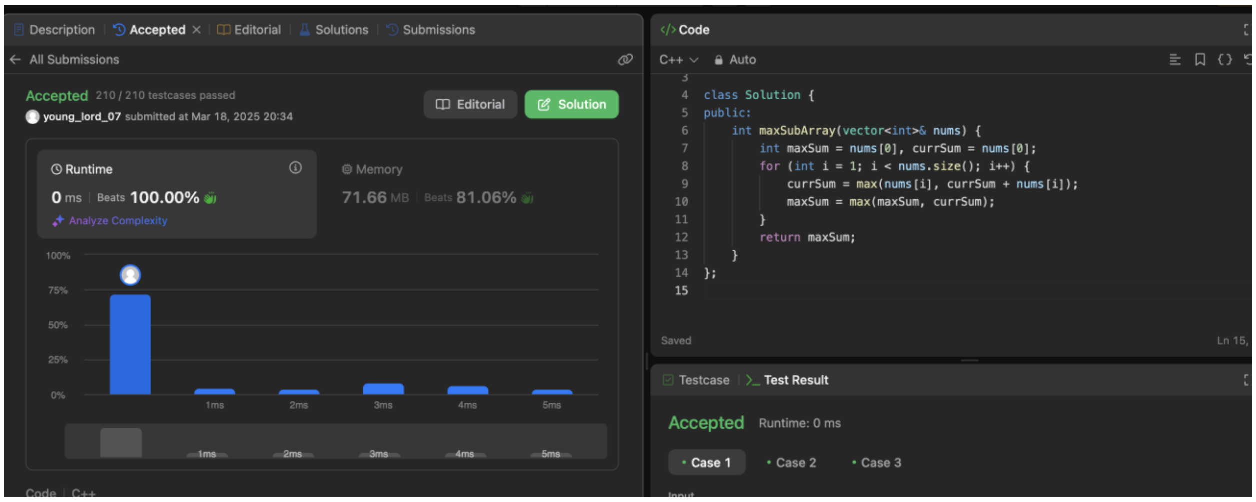
n =
11

Output

3

Solution 4:

```
#include <vector>
using namespace std;

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



Accepted 210 / 210 testcases passed

young_lord_07 submitted at Mar 18, 2025 20:34

Runtime 0 ms | Beats 100.00%

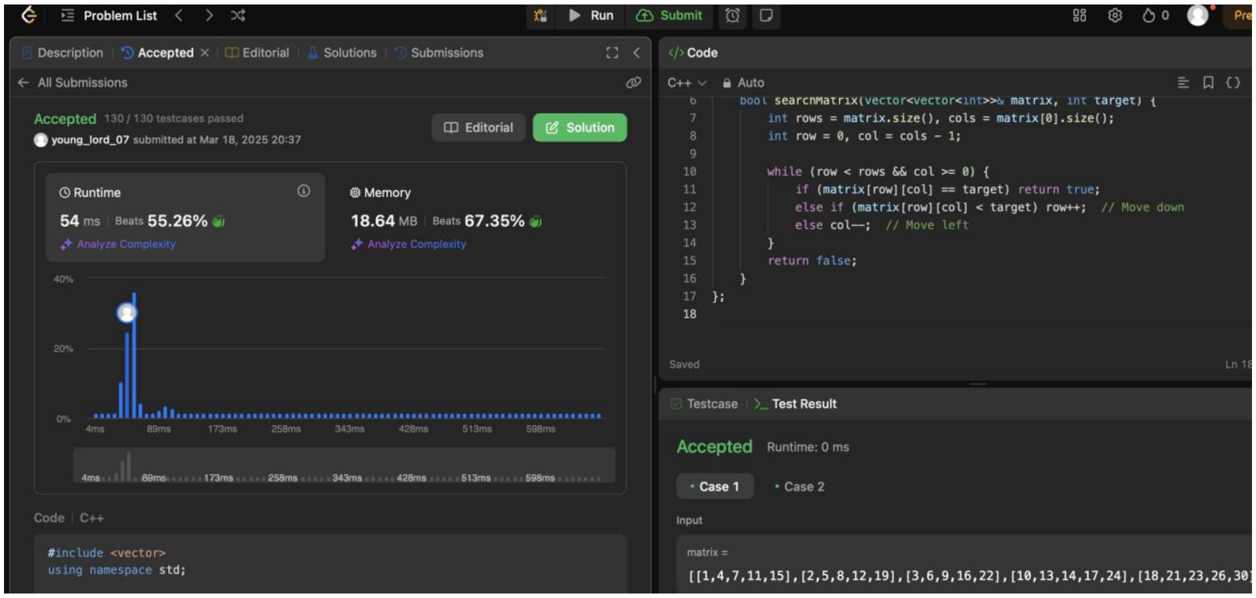
Memory 71.66 MB | Beats 81.06%

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

Accepted Runtime: 0 ms

Solution 5:

```cpp
#include <vector>
using namespace std;

class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        int rows = matrix.size(), cols = matrix[0].size();
        int row = 0, col = cols - 1;

        while (row < rows && col >= 0) {
            if (matrix[row][col] == target) return true;
            else if (matrix[row][col] < target) row++;  // Move down
            else col--;  // Move left
        }
        return false;
    }
};
```



Accepted 130 / 130 testcases passed
young_lord_07 submitted at Mar 18, 2025 20:37

Runtime
54 ms | Beats 55.26%

Memory
18.64 MB | Beats 67.35%

Code | C++

```
#include <vector>
using namespace std;
```

```
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        int rows = matrix.size(), cols = matrix[0].size();
        int row = 0, col = cols - 1;

        while (row < rows && col >= 0) {
            if (matrix[row][col] == target) return true;
            else if (matrix[row][col] < target) row++;  // Move down
            else col--;  // Move left
        }
        return false;
    }
};
```

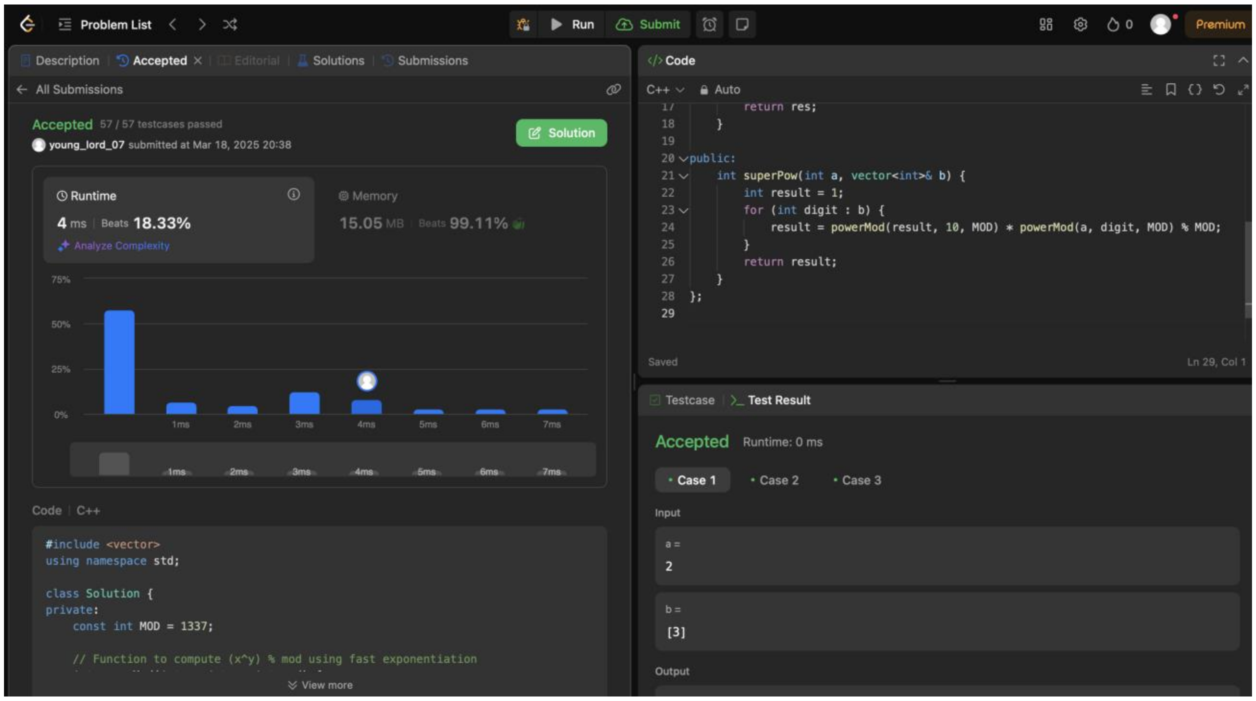
Accepted Runtime: 0 ms

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

Solution 6:

```
#include <vector>
using namespace std;

class Solution {
private:
    const int MOD = 1337;

    // Function to compute (x^y) % mod using fast exponentiation
    int powerMod(int x, int y, int mod) {
        int res = 1;
        x %= mod;
        while (y > 0) {
            if (y % 2 == 1) res = (res * x) % mod;
            x = (x * x) % mod;
            y /= 2;
        }
        return res;
    }

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



Accepted 57 / 57 testcases passed

young_lord_07 submitted at Mar 18, 2025 20:38

Runtime: 4 ms | Beats 18.33%

Memory: 15.05 MB | Beats 99.11%

```
#include <vector>
using namespace std;

class Solution {
private:
    const int MOD = 1337;

    // Function to compute (x^y) % mod using fast exponentiation
```

```
        return res;
    }

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

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

a =
2

b =
[3]

Output

Solution 6:

#include using namespace std;

class Solution {

public: vector beautifulArray(int n) { vector res = {1};

while (res.size() < n) { vector temp;

 for (int num : res)

 if (num * 2 - 1 <= n) temp.push_back(num * 2 - 1);

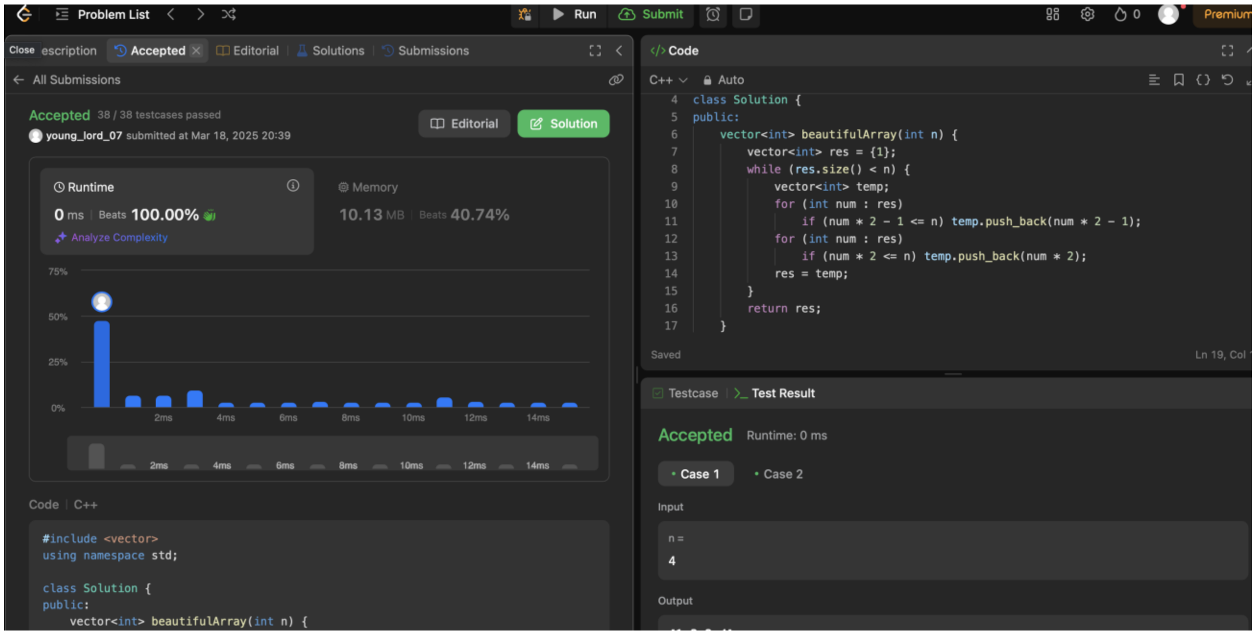
 for (int num : res)

 if (num * 2 <= n) temp.push_back(num * 2);

res = temp;

} return res;

} };



Accepted 38 / 38 testcases passed
young_lord_07 submitted at Mar 18, 2025 20:39

Runtime
0 ms | Beats 100.00%

Memory
10.13 MB | Beats 40.74%

Code | C++

```cpp
#include <vector>
using namespace std;

class Solution {
public:
    vector<int> beautifulArray(int n) {
```

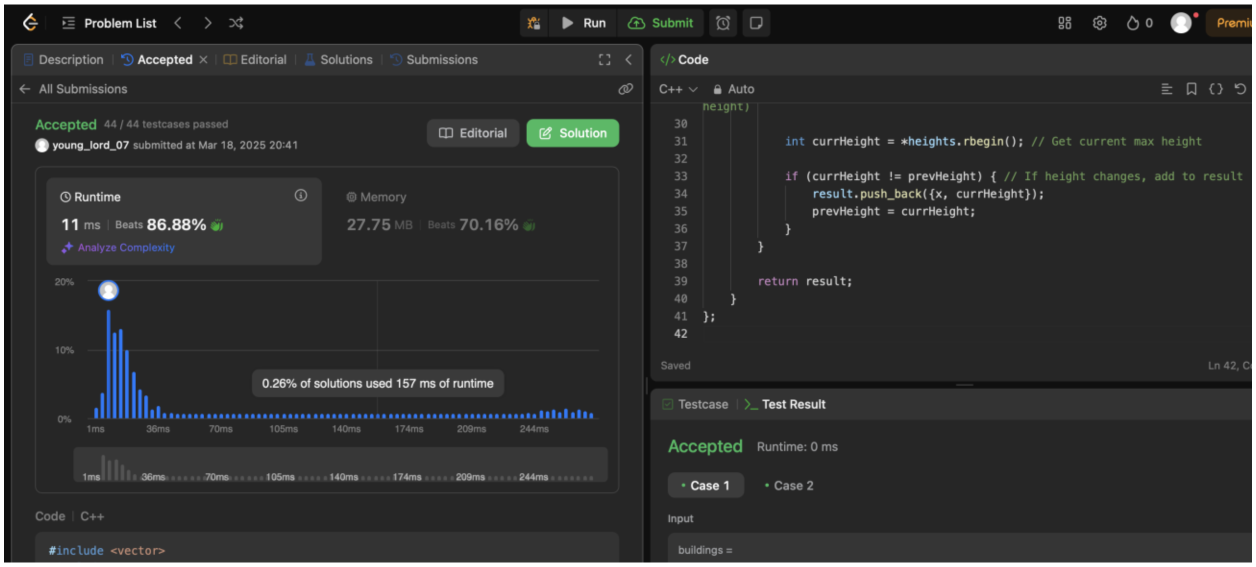
```cpp
class Solution {
public:
    vector<int> beautifulArray(int n) {
        vector<int> res = {1};
        while (res.size() < n) {
            vector<int> temp;
            for (int num : res)
                if (num * 2 - 1 <= n) temp.push_back(num * 2 - 1);
            for (int num : res)
   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

Accepted Runtime: 0 ms

```
n =
4
```

Solution 8:

```cpp
#include <vector>

#include <queue>

#include <set>

using namespace std;


class Solution {

public:

    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {

        vector<pair<int, int>> events;

        vector<vector<int>> result;



        // Step 1: Convert buildings into "events"

        for (auto& b : buildings) {

            events.emplace_back(b[0], -b[2]); // Start of building (negative height)

            events.emplace_back(b[1], b[2]);  // End of building (positive height)
        }

        // Step 2: Sort events
        sort(events.begin(), events.end());

        // Step 3: Process events using max-heap
        multiset<int> heights = {0};
        int prevHeight = 0;

        for (auto& event : events) {
            int x = event.first, h = event.second;

            if (h < 0) heights.insert(-h);  // Start of building (add height)
            else heights.erase(heights.find(h)); // End of building (remove height)

            int currHeight = *heights.rbegin(); // Get current max height

            if (currHeight != prevHeight) { // If height changes, add to result
                result.push_back({x, currHeight});
                prevHeight = currHeight;
            }
        }

        return result;
    }
};
```


Accepted 44 / 44 testcases passed

young_lord_07 submitted at Mar 18, 2025 20:41

Runtime: 11 ms | Beats 86.88%

Memory: 27.75 MB | Beats 70.16%

0.26% of solutions used 157 ms of runtime

Code | C++

```
#include <vector>
```

```
        height)

            int currHeight = *heights.rbegin(); // Get current max height

            if (currHeight != prevHeight) { // If height changes, add to result
                result.push_back({x, currHeight});
                prevHeight = currHeight;
            }
        }

        return result;
    }
};
```

Saved

Ln 42, C

Test Result

Accepted Runtime: 0 ms

Case 1 · Case 2

Input

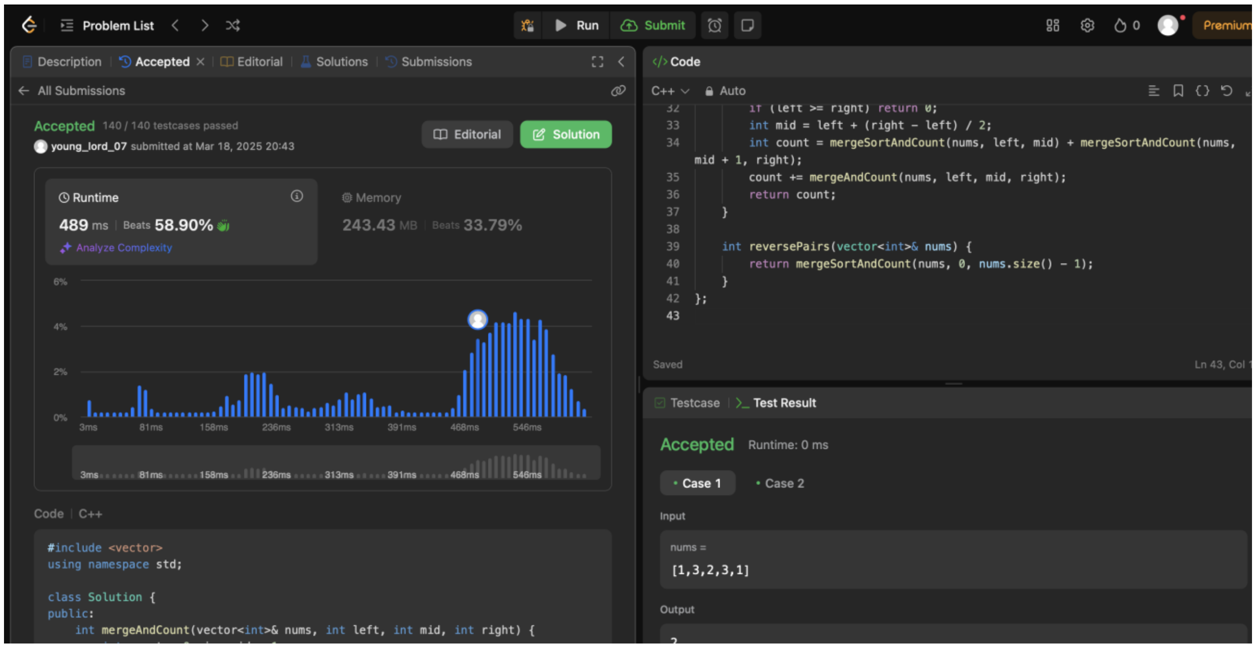
buildings =

Solution 9:

```
#include <vector>
using namespace std;

class Solution {
public:
  int mergeAndCount(vector<int>& nums, int left, int mid, int right) {
    int count = 0, j = mid + 1;

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

    // Merge step
    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++]);


        // Copy sorted array back

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


        return count;

    }


    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);

        count += mergeAndCount(nums, left, mid, right);

        return count;

    }


    int reversePairs(vector<int>& nums) {

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

    }
```

};



Accepted 140 / 140 testcases passed
young_lord_07 submitted at Mar 18, 2025 20:43

Runtime 489 ms | Beats 58.90%
Memory 243.43 MB | Beats 33.79%

Code | C++

```
#include <vector>
using namespace std;

class Solution {
public:
    int mergeAndCount(vector<int>& nums, int left, int mid, int right) {
```

```
        if (left >= right) return 0;
        int mid = left + (right - left) / 2;
        int count = mergeSortAndCount(nums, left, mid) + mergeSortAndCount(nums,
mid + 1, right);
        count += mergeAndCount(nums, left, mid, right);
        return count;
    }

    int reversePairs(vector<int>& nums) {
        return mergeSortAndCount(nums, 0, nums.size() - 1);
    }
};
```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

```
nums =
[1,3,2,3,1]
```

Output

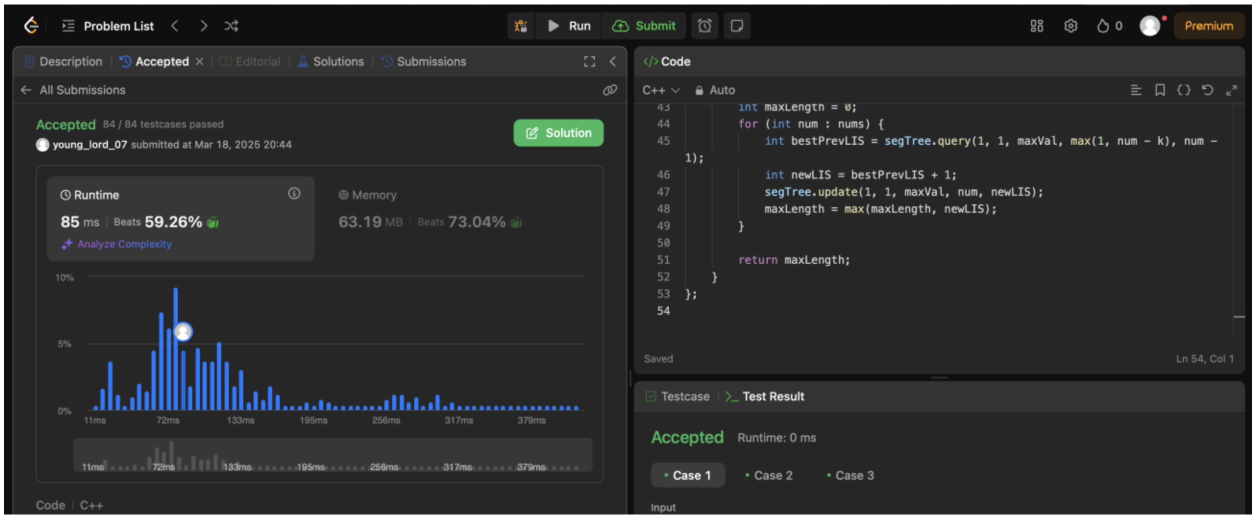
Solution 10:
#include using namespace std;

class SegmentTree { public: vector tree; int size;

```
SegmentTree(int n) {
    size = n;
    tree.resize(4 * n, 0);
}

int query(int node, int start, int end, int L, int R) {
    if (start > R || end < L) return 0;  // Out of range
    if (start >= L && end <= R) return tree[node];  // Inside range

    int mid = (start + end) / 2;
    return max(query(2 * node, start, mid, L, R),
               query(2 * node + 1, mid + 1, end, L, R));
}

void update(int node, int start, int end, int idx, int value) {
    if (start == end) {
        tree[node] = value;
        return;
    }

    int mid = (start + end) / 2;
    if (idx <= mid) update(2 * node, start, mid, idx, value);
    else update(2 * node + 1, mid + 1, end, idx, value);

    tree[node] = max(tree[2 * node], tree[2 * node + 1]);
}


};
```

class Solution { public: int lengthOfLIS(vector& nums, int k) { int maxVal = *max_element(nums.begin(), nums.end()); SegmentTree segTree(maxVal);

```
   int maxLength = 0;
    for (int num : nums) {
        int bestPrevLIS = segTree.query(1, 1, maxVal, max(1, num - k), num - 1);
        int newLIS = bestPrevLIS + 1;
        segTree.update(1, 1, maxVal, num, newLIS);
        maxLength = max(maxLength, newLIS);
    }

    return maxLength;
}


};
```



Accepted 84 / 84 testcases passed

young_lord_07 submitted at Mar 18, 2025 20:44

Runtime

85 ms | Beats 59.26%

Memory

63.19 MB | Beats 73.04%

```cpp
        int maxLength = 0;
        for (int num : nums) {
            int bestPrevLIS = segTree.query(1, 1, maxVal, max(1, num - k), num -
1);
            int newLIS = bestPrevLIS + 1;
            segTree.update(1, 1, maxVal, num, newLIS);
            maxLength = max(maxLength, newLIS);
        }

        return maxLength;
    }
};
```

Accepted Runtime: 0 ms