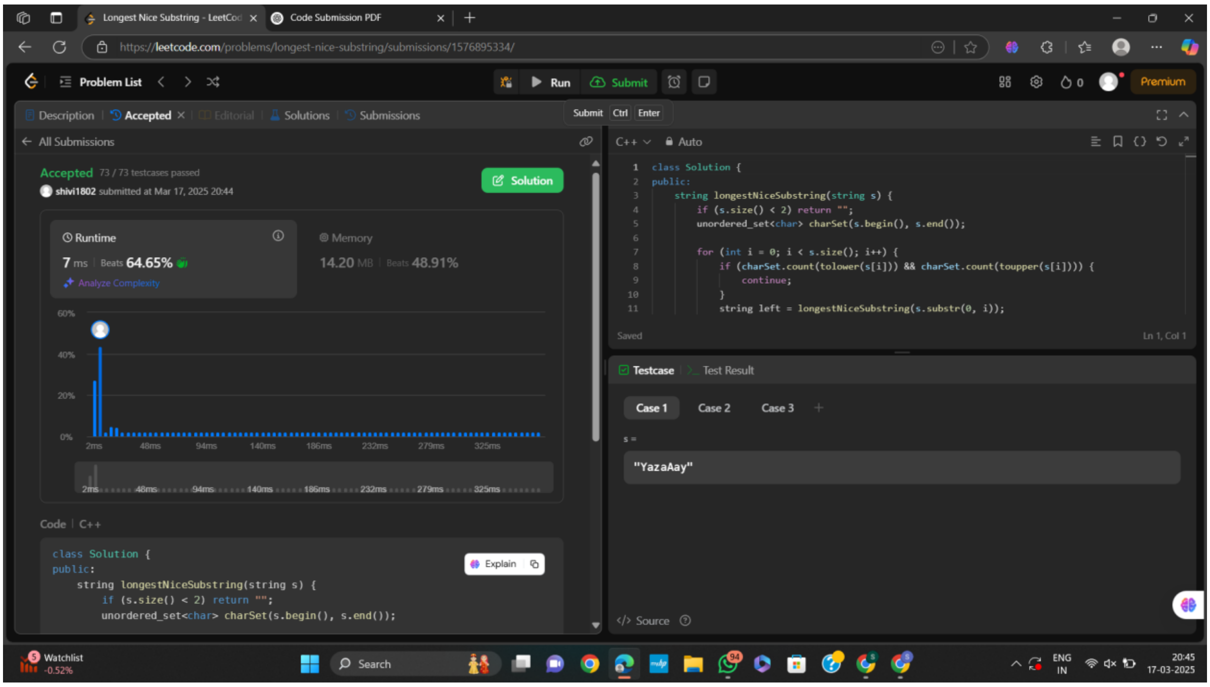
```
Code 1: class Solution {
public:
  string longestNiceSubstring(string s) {
    if (s.size() < 2) return "";
    unordered_set<char> charSet(s.begin(), s.end());

    for (int i = 0; i < s.size(); i++) {
      if (charSet.count(tolower(s[i])) && charSet.count(toupper(s[i]))) {
        continue;
      }
      string left = longestNiceSubstring(s.substr(0, i));
      string right = longestNiceSubstring(s.substr(i + 1));
      return left.size() >= right.size() ? left : right;
    }
    return s;
  }
};
```



Accepted 73 / 73 testcases passed

shivi1802 submitted at Mar 17, 2025 20:44

Runtime: 7 ms | Beats 64.65%

Analyze Complexity

Memory: 14.20 MB | Beats 48.91%

Code | C++

```
class Solution {
public:
    string longestNiceSubstring(string s) {
        if (s.size() < 2) return "";
        unordered_set<char> charSet(s.begin(), s.end());
```

```
class Solution {
public:
    string longestNiceSubstring(string s) {
        if (s.size() < 2) return "";
        unordered_set<char> charSet(s.begin(), s.end());

        for (int i = 0; i < s.size(); i++) {
            if (charSet.count(tolower(s[i])) && charSet.count(toupper(s[i]))) {
                continue;
            }
            string left = longestNiceSubstring(s.substr(0, i));
```

Saved

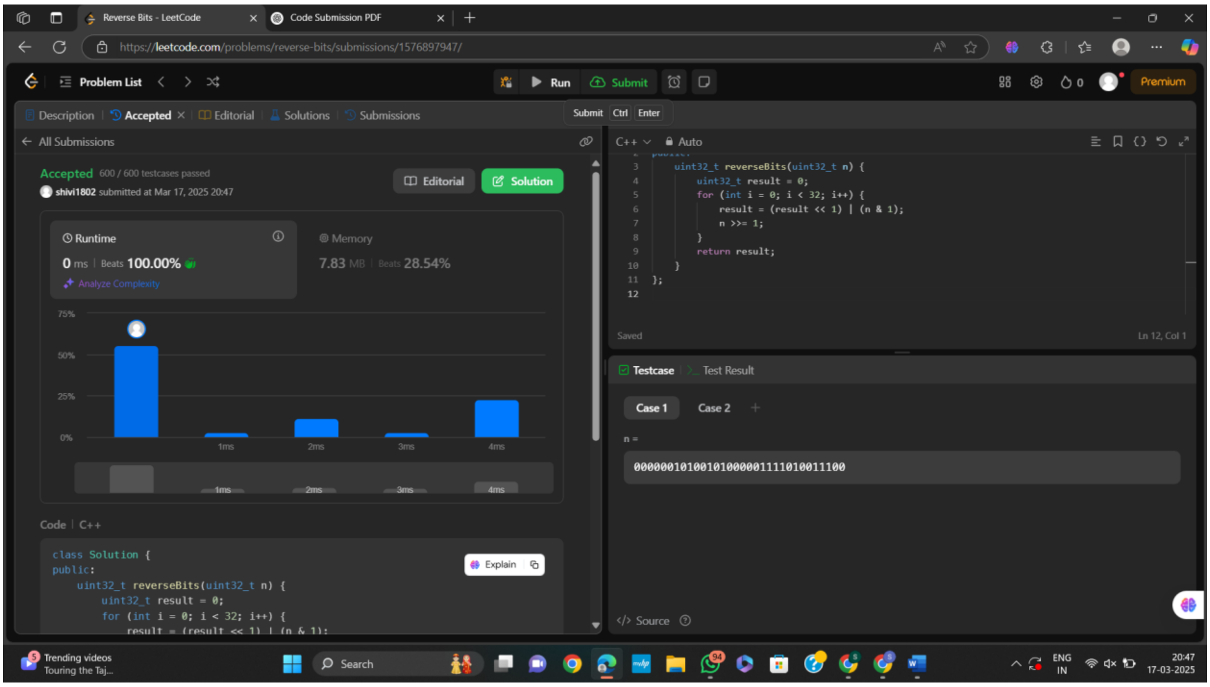
Ln 1, Col 1

s =

"YazaAay"

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;
    }
};
```



Accepted 600 / 600 testcases passed

shivi1802 submitted at Mar 17, 2025 20:47

Runtime
0 ms | Beats 100.00%

Memory
7.83 MB | Beats 28.54%

```
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;
    }
};
```

Saved

Ln 12, Col 1

n =

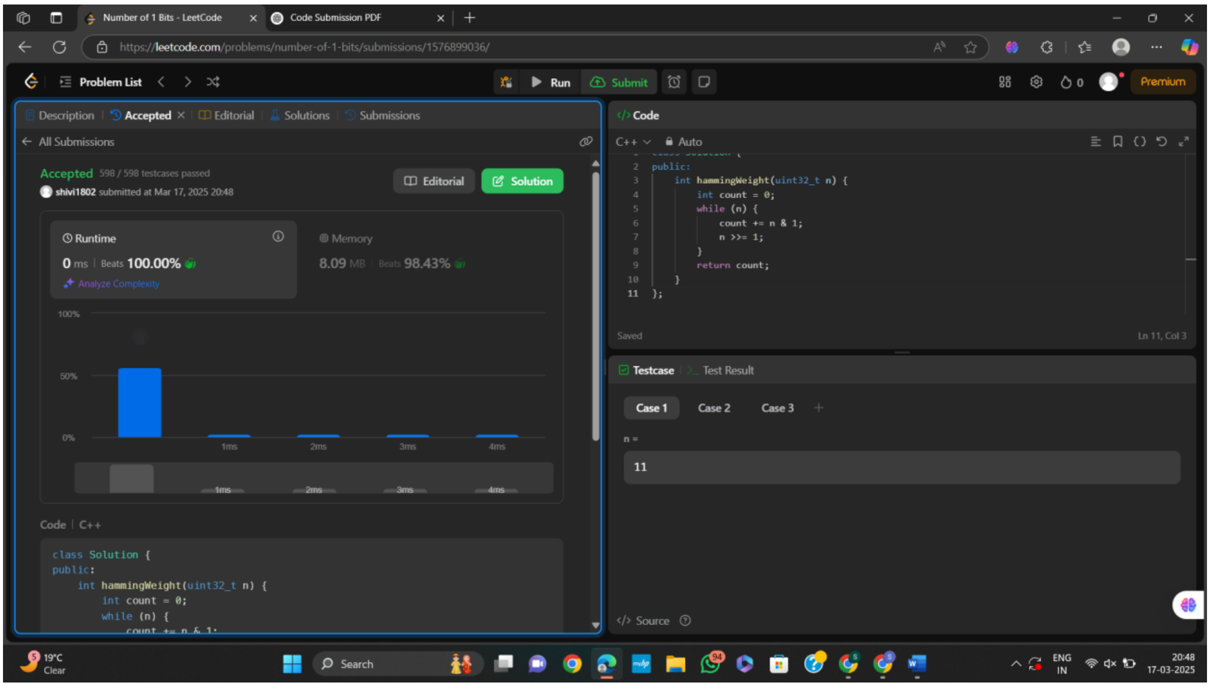
```
00000010100101000001111010011100
```

Code | C++

```
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);
```

Code 3:

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



Accepted 598 / 598 testcases passed

shivi1802 submitted at Mar 17, 2025 20:48

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

8.09 MB | Beats 98.43%

Code | C++

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

Saved

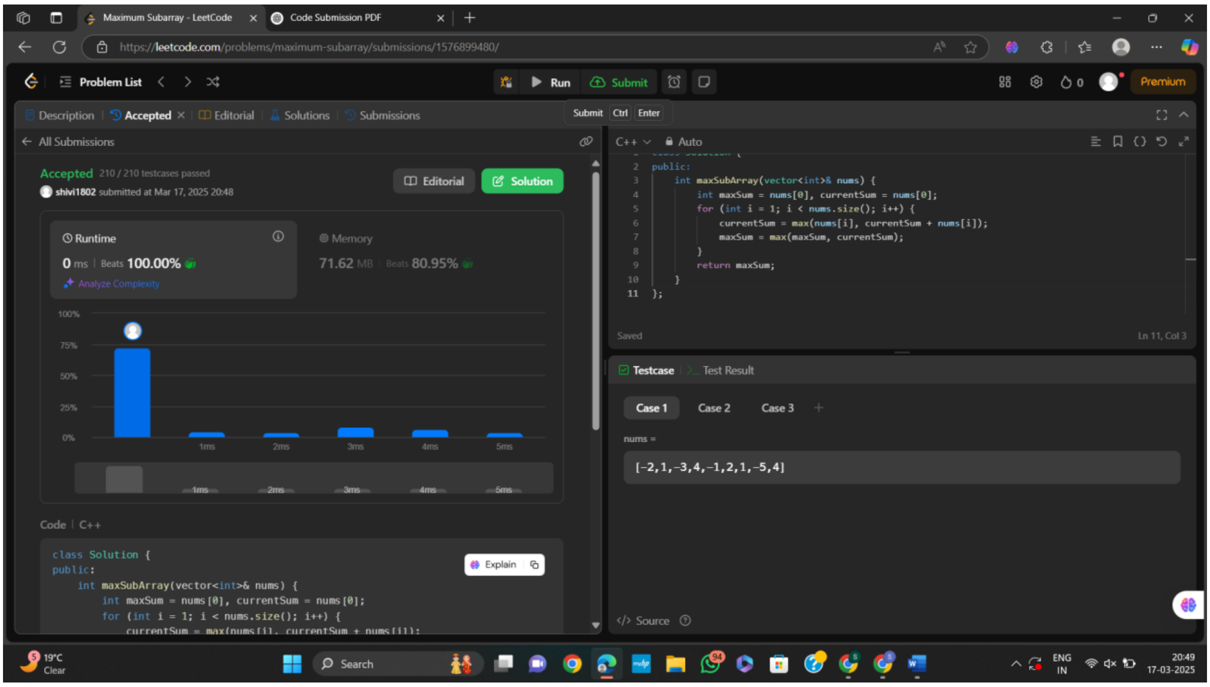
Ln 11, Col 3

n =

11

Code 4:

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



Accepted 210 / 210 testcases passed

shivi1802 submitted at Mar 17, 2025 20:48

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

71.62 MB | Beats 80.95%

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

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

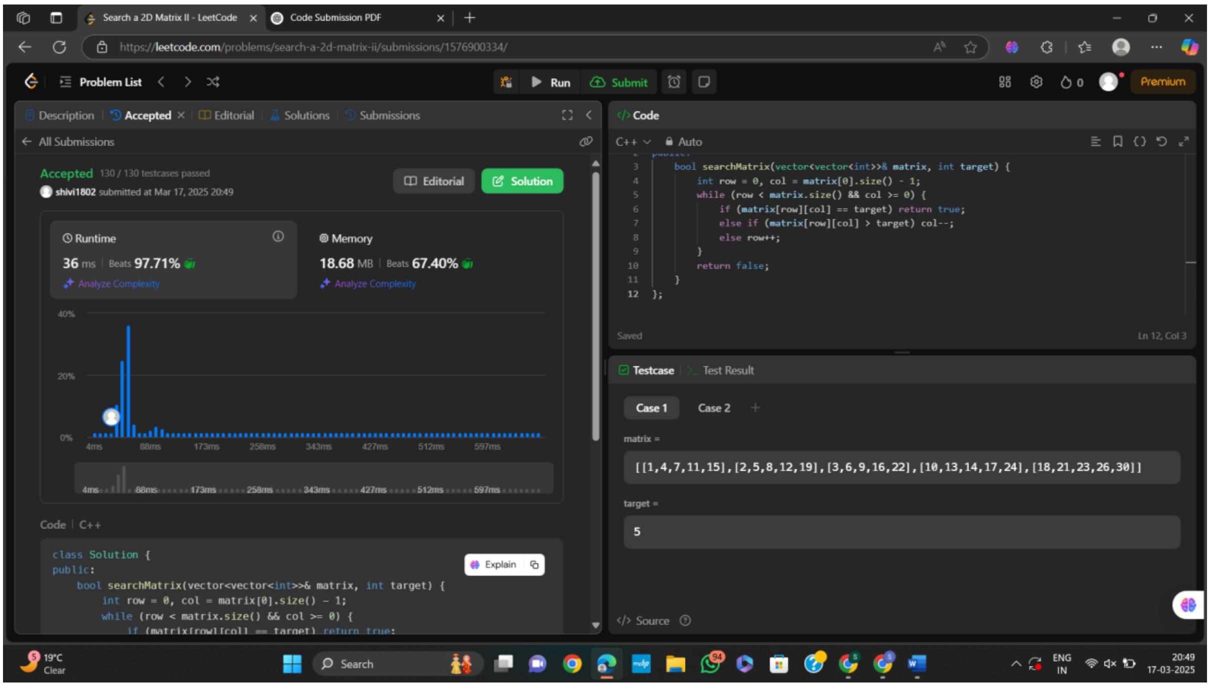
nums =

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

Code 5:

```
class Solution {
public:
  bool searchMatrix(vector<vector<int>>& matrix, int target) {
    int row = 0, col = matrix[0].size() - 1;
    while (row < matrix.size() && col >= 0) {
      if (matrix[row][col] == target) return true;
      else if (matrix[row][col] > target) col--;
      else row++;
    }
    return false;
  }
```

};



Accepted 130 / 130 testcases passed

shivi1802 submitted at Mar 17, 2025 20:49

Runtime: 36 ms | Beats 97.71%

Memory: 18.68 MB | Beats 67.40%

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

```
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        int row = 0, col = matrix[0].size() - 1;
        while (row < matrix.size() && col >= 0) {
            if (matrix[row][col] == target) return true;
            else if (matrix[row][col] > target) col--;
            else row++;
        }
        return false;
    }
};
```

matrix =

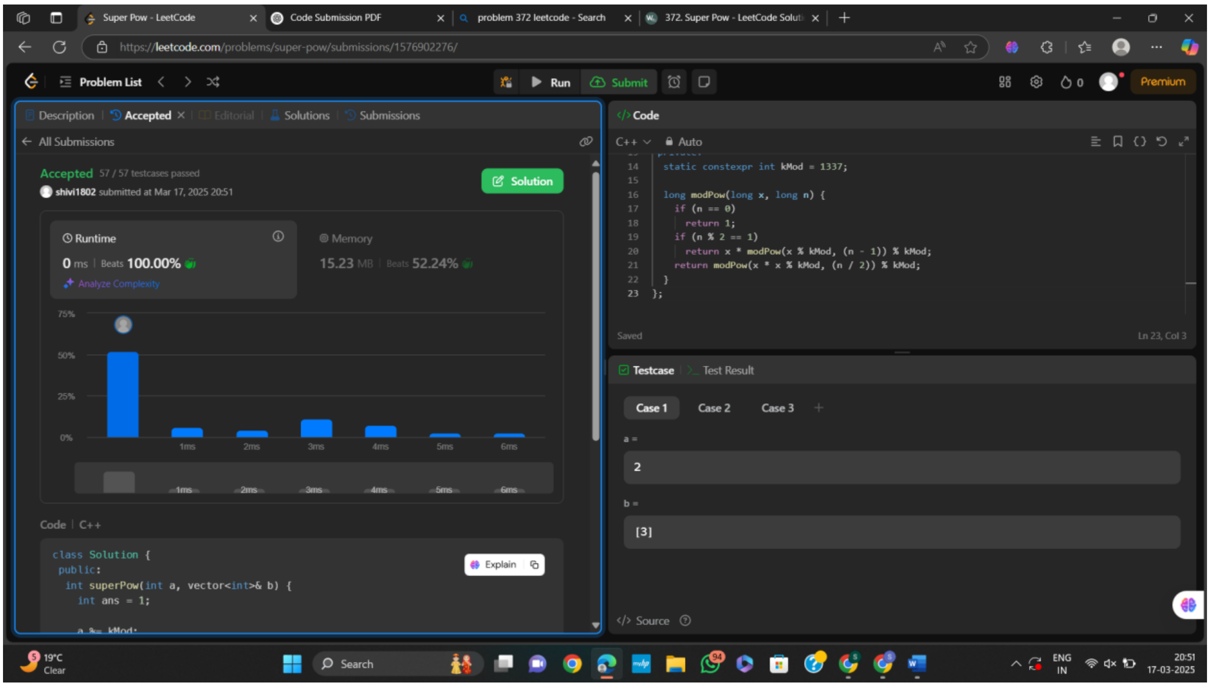
```
[[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]]
```

target =

```
5
```

Code 6:

```
class Solution {
public:
  int modPow(int a, int b, int mod) {
    int result = 1;
    a %= mod;
    while (b > 0) {
      if (b % 2 == 1) result = (result * a) % mod;
      a = (a * a) % mod;
      b /= 2;
    }
    return result;
  }

  int superPow(int a, vector<int>& b) {
    int mod = 1337;
    int power = 0;
    for (int digit : b) {
      power = (power * 10 + digit) % 1140;
    }
    return modPow(a, power, mod);
  }
};
```



Accepted 57 / 57 testcases passed

shivi1802 submitted at Mar 17, 2025 20:51

Runtime 0 ms | Beats 100.00%

Memory 15.23 MB | Beats 52.24%

Code | C++

```
class Solution {
 public:
  int superPow(int a, vector<int>& b) {
    int ans = 1;

    a %= kMod;
```

```
 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;
  }
};
```

a =

2

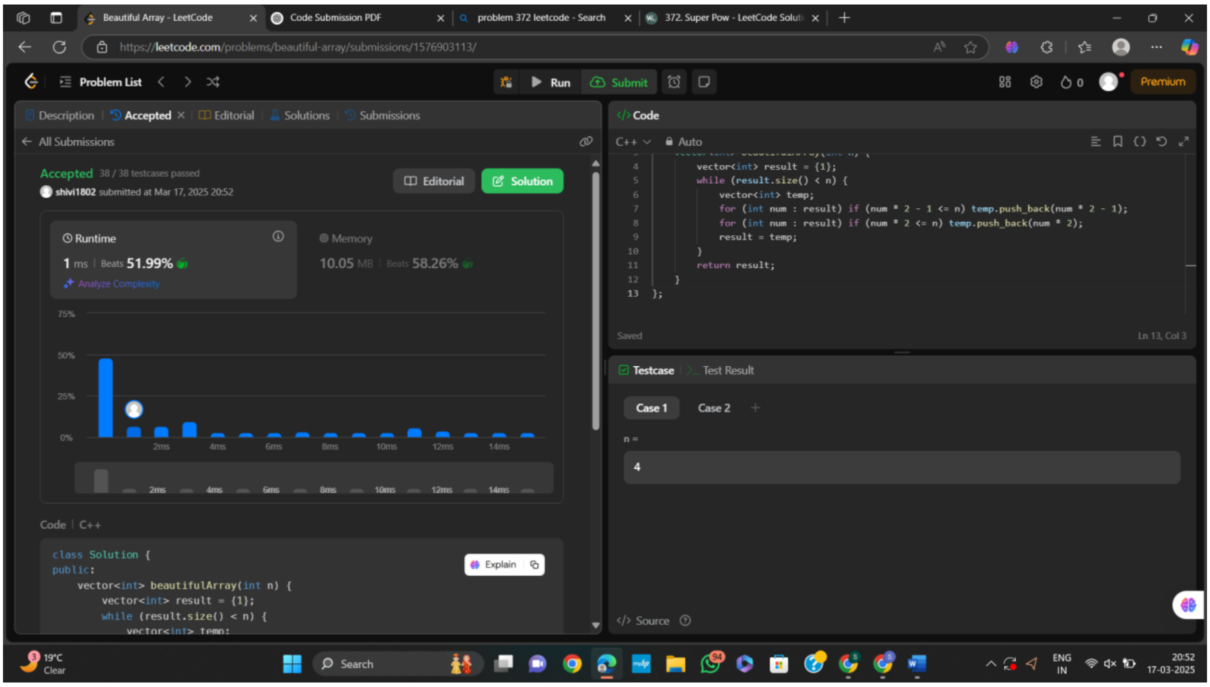
b =

[3]

Code 7:

```
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;
  }
```

};



Accepted 38 / 38 testcases passed

shivi1802 submitted at Mar 17, 2025 20:52

Runtime: 1 ms | Beats 51.99%

Memory: 10.05 MB | Beats 58.26%

Code | C++

```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        vector<int> result = {1};
        while (result.size() < n) {
            vector<int> temp;
```

```
        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;
    }
};
```

Saved

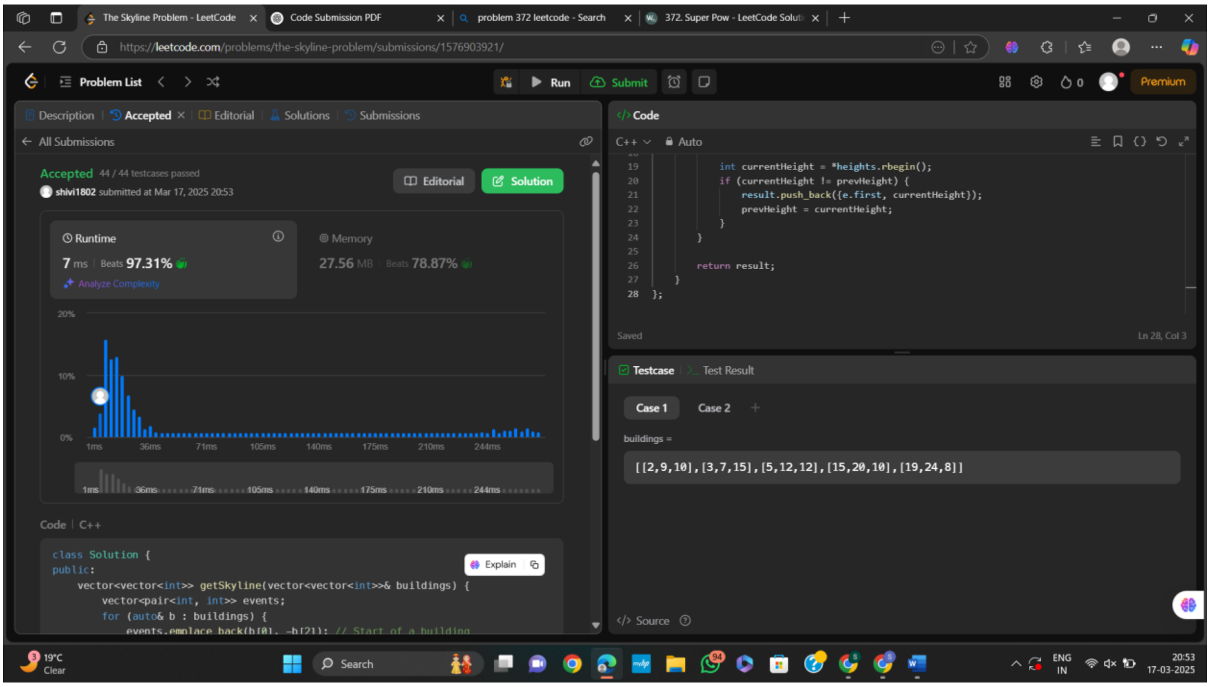
Ln 13, Col 3

n =

4

Code 8:

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

        sort(events.begin(), events.end());
        multiset<int> heights = {0};
        vector<vector<int>> result;
        int prevHeight = 0;

        for (auto& e : events) {
            if (e.second < 0) heights.insert(-e.second);
            else heights.erase(heights.find(e.second));
            int currentHeight = *heights.rbegin();
            if (currentHeight != prevHeight) {
                result.push_back({e.first, currentHeight});
                prevHeight = currentHeight;
            }
        }

        return result;
    }
};
```



Accepted 44 / 44 testcases passed

shivi1802 submitted at Mar 17, 2025 20:53

Runtime 7 ms | Beats 97.31%

Memory 27.56 MB | Beats 78.87%

```
            int currentHeight = *heights.rbegin();
            if (currentHeight != prevHeight) {
                result.push_back({e.first, currentHeight});
                prevHeight = currentHeight;
            }
        }

        return result;
    }
};
```

buildings =

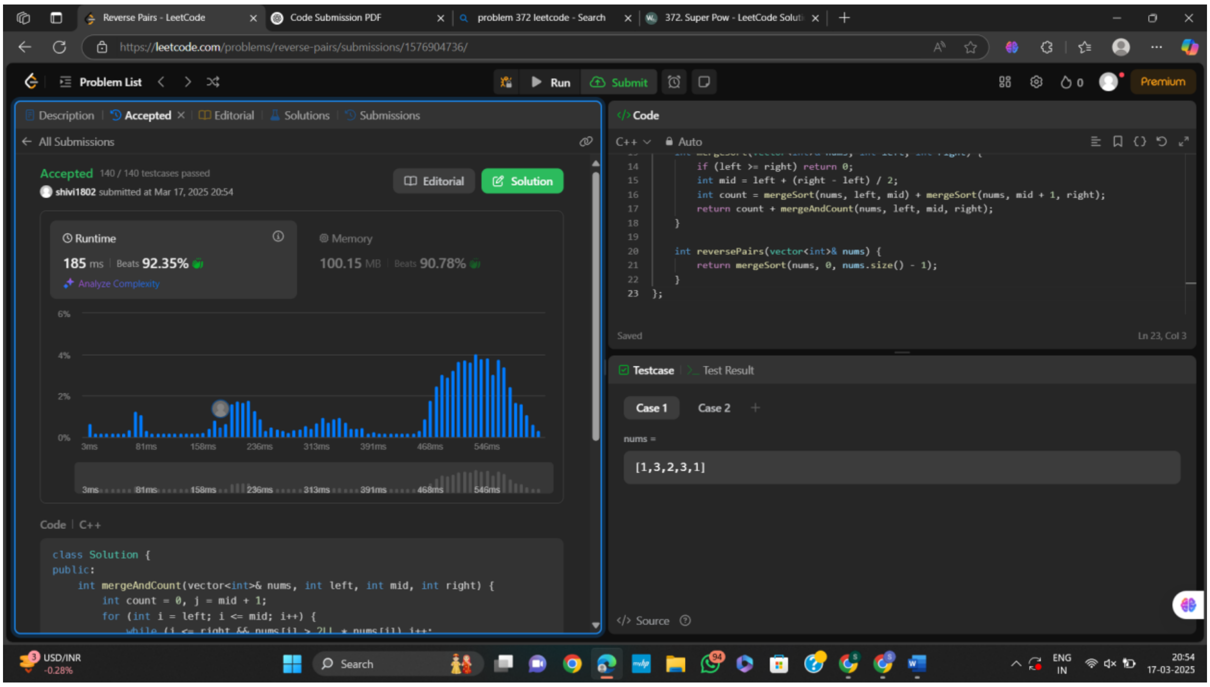
```
[[2,9,10],[3,7,15],[5,12,12],[15,20,10],[19,24,8]]
```

Code | C++

```
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<pair<int, int>> events;
        for (auto& b : buildings) {
            events.emplace_back(b[0], -b[2]); // Start of a building
```

Code 9: class Solution {

```
public:
    int mergeAndCount(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);
        }
        inplace_merge(nums.begin() + left, nums.begin() + mid + 1, nums.begin() + right + 1);
        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);
        return count + mergeAndCount(nums, left, mid, right);
    }

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



Accepted 140 / 140 testcases passed

shivi1802 submitted at Mar 17, 2025 20:54

Runtime 185 ms | Beats 92.35%

Memory 100.15 MB | Beats 90.78%

Code | C++

```
class Solution {
public:
    int mergeAndCount(vector<int>& nums, int left, int mid, int right) {
        int count = 0, j = mid + 1;
        for (int i = left; i <= mid; i++) {
            while (i <= right && nums[i] > 2LL * nums[j]) j++;
```

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

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

Saved

Ln 23, Col 3

nums =

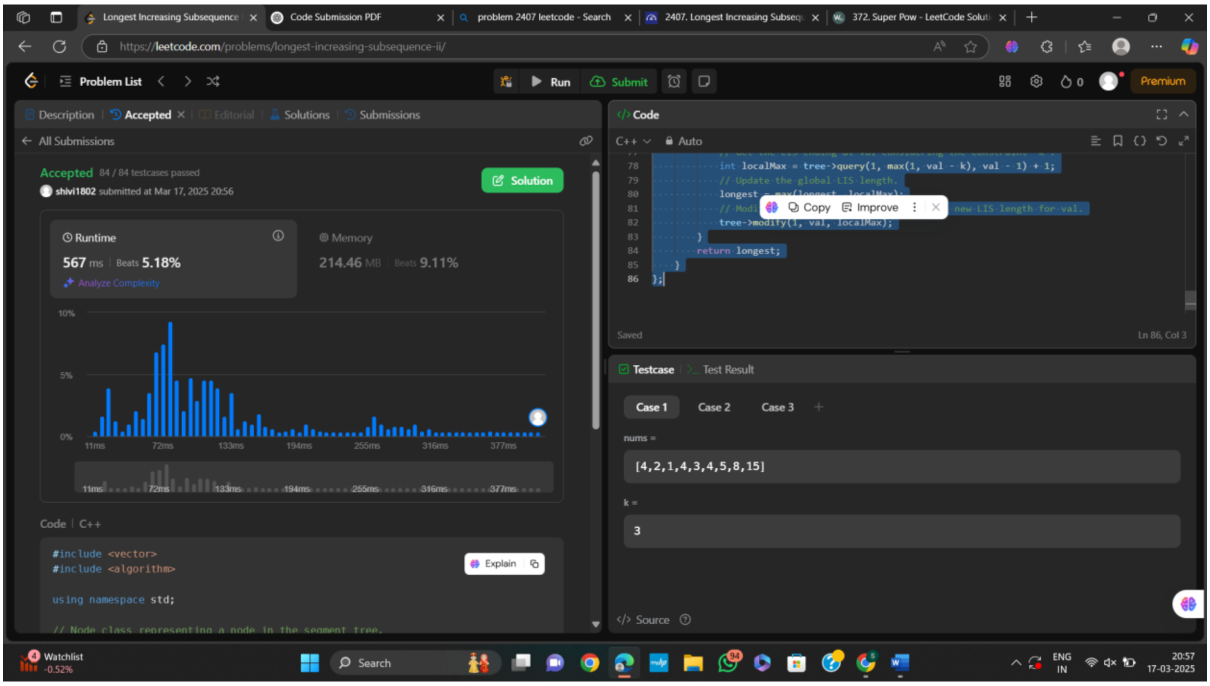
[1,3,2,3,1]

Code 10:

```
#include <vector>
#include <algorithm>


using namespace std;
```

```
// Node class representing a node in the segment tree.
class Node {
public:
    int left;
    int right;
    int value;
};

// SegmentTree class representing a segment tree data structure.
class SegmentTree {
private:
    vector<Node*> tree; // Vector of Node pointers representing the segment tree structure.

    // Private helper method to push-up the value to the parent node after modifications.
    void pushUp(int index) {
        tree[index]->value = max(tree[index * 2]->value, tree[index * 2 + 1]->value);
    }

public:
    // Constructor to initialize segment tree with `n` elements.
    SegmentTree(int n) {
        tree.resize(4 * n);
        for (int i = 0; i < tree.size(); ++i) tree[i] = new Node();
        build(1, 1, n);
    }

    // Method to build the segment tree recursively.
    void build(int index, int left, int right) {
        tree[index]->left = left;
        tree[index]->right = right;
        if (left == right) return; // Base case: reach a leaf node.
        int mid = (left + right) >> 1;
        build(index * 2, left, mid);
        build(index * 2 + 1, mid + 1, right);
    }

    // Method to modify the value at a specific position in the segment tree.
    void modify(int index, int position, int value) {
        if (tree[index]->left == position && tree[index]->right == position) {
            tree[index]->value = value;
            return;
        }
        int mid = (tree[index]->left + tree[index]->right) >> 1;
        if (position <= mid)
            modify(index * 2, position, value);
        else
            modify(index * 2 + 1, position, value);
        pushUp(index);
    }

    // Method to query the maximum value within a range [left, right] in the segment tree.
    int query(int index, int left, int right) {
        if (tree[index]->left >= left && tree[index]->right <= right) return tree[index]->value;
        int mid = (tree[index]->left + tree[index]->right) >> 1;
        int maxValue = 0;
        if (left <= mid) maxValue = query(index * 2, left, right);
        if (right > mid) maxValue = max(maxValue, query(index * 2 + 1, left, right));
        return maxValue;
    }
};

// Solution class to solve the problem.
```

```
class Solution {
public:
    // Method to find the length of the Longest Increasing Subsequence (LIS)
    // where the difference between adjacent elements is at most `k`.
    int lengthOfLIS(vector<int>& nums, int k) {
        int maxNum = *max_element(nums.begin(), nums.end());
        SegmentTree* tree = new SegmentTree(maxNum);
        int longest = 1;
        for (int val : nums) {
            // Get the LIS ending at val considering the constraint 'k'.
            int localMax = tree->query(1, max(1, val - k), val - 1) + 1;
            // Update the global LIS length.
            longest = max(longest, localMax);
            // Modify the segment tree to include the new LIS length for val.
            tree->modify(1, val, localMax);
        }
        return longest;
    }
};
```



Accepted 84 / 84 testcases passed

shivi1802 submitted at Mar 17, 2025 20:56

Runtime: 567 ms | Beats 5.18%

Memory: 214.46 MB | Beats 9.11%

Code | C++

```
#include <vector>
#include <algorithm>

using namespace std;

// Node class representing a node in the segment tree.
```

```
            int localMax = tree->query(1, max(1, val - k), val - 1) + 1;
            // Update the global LIS length.
            longest
            // Modi                          new LIS length for val.
            tree->modify(1, val, localMax);
        }
        return longest;
    }
};
```

nums =

```
[4,2,1,4,3,4,5,8,15]
```

k =

```
3
```