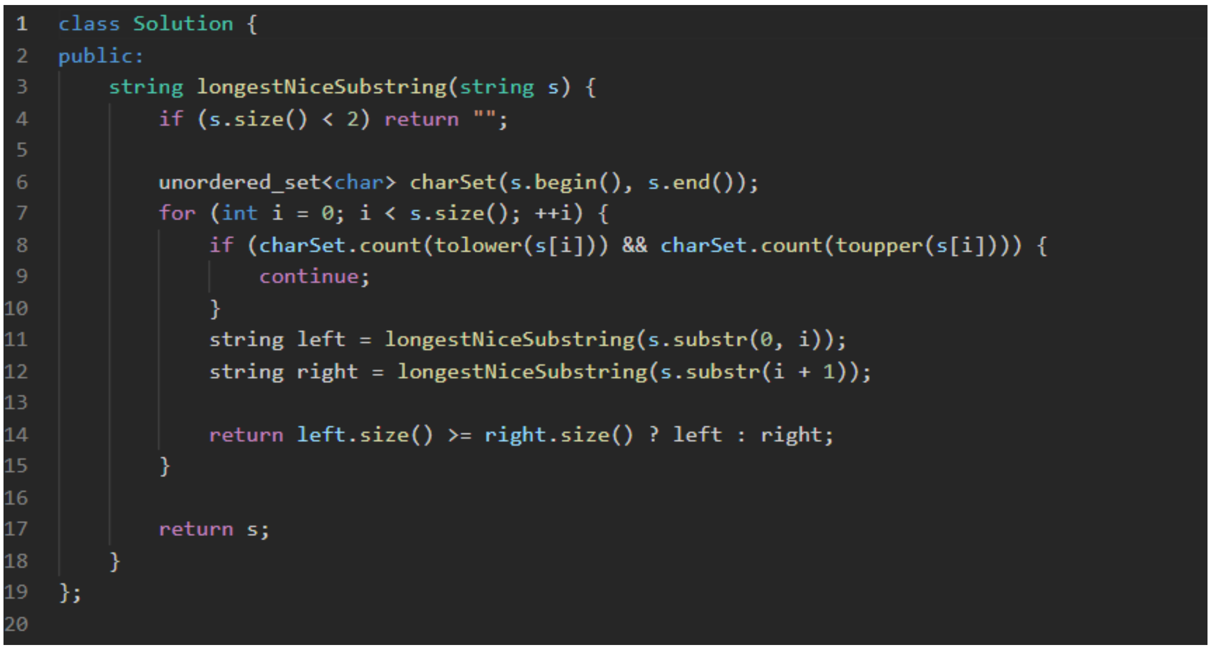
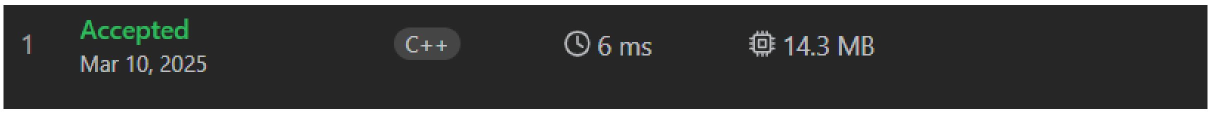
1.Longest Nice Substring.

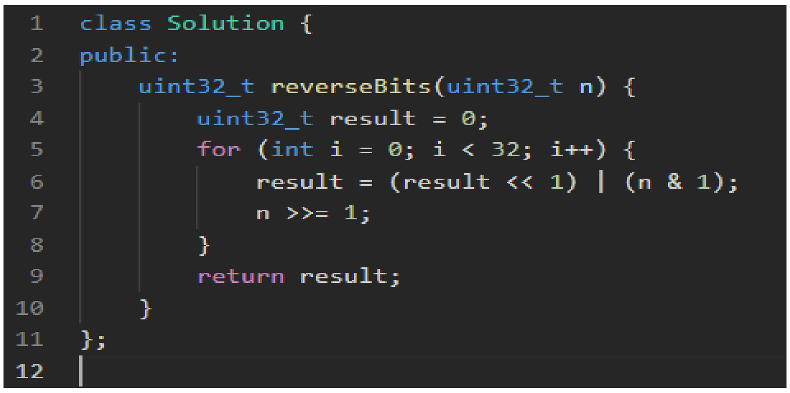


```cpp
class Solution {
public:
    string longestNiceSubstring(string s) {
        if (s.size() < 2) return "";

        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;
    }
};

```

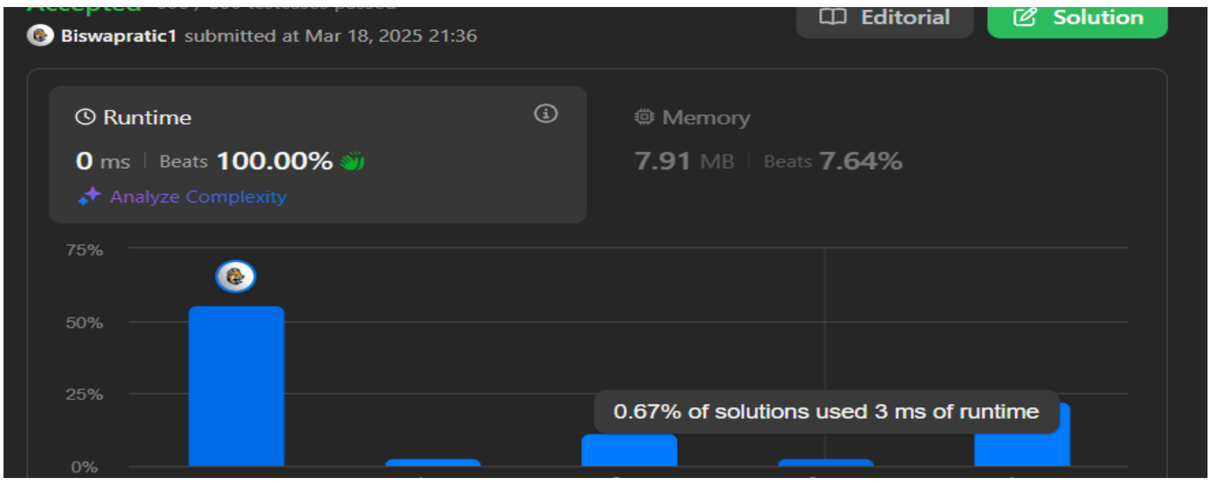


1 Accepted Mar 10, 2025 C++ 6 ms 14.3 MB

2.Reverse Bits.



```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;
    }
};

```



Biswapratic1 submitted at Mar 18, 2025 21:36

Editorial Solution

Runtime

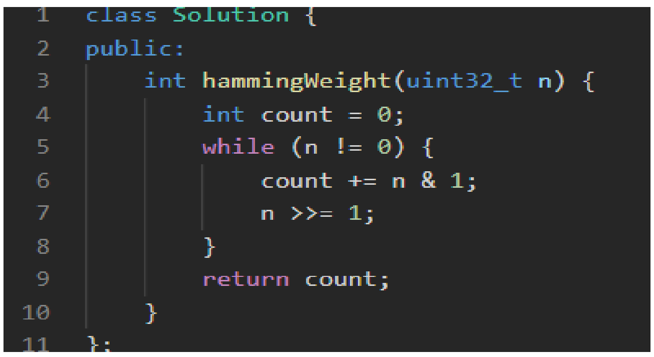
0 ms | Beats 100.00%

Analyze Complexity

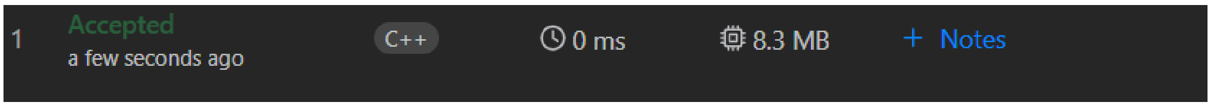
Memory

7.91 MB | Beats 7.64%

3.Number of 1 Bits.

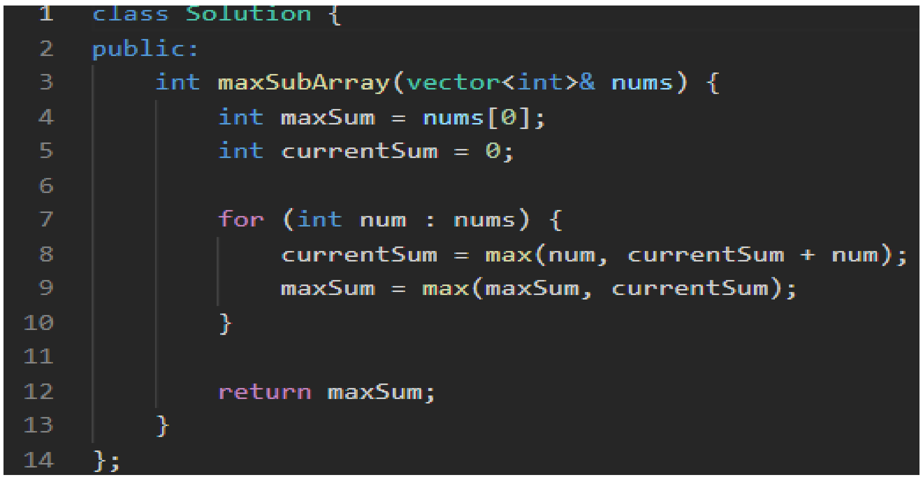


```cpp
class Solution {
public:
    int hammingWeight(uint32_t n) {
        int count = 0;
        while (n != 0) {
            count += n & 1;
            n >>= 1;
        }
        return count;
    }
};
```

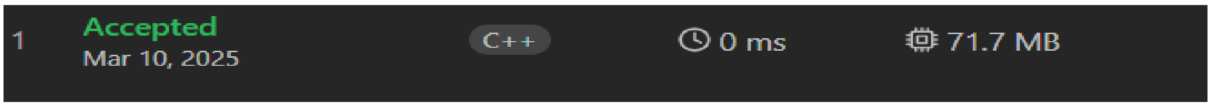


```
1   Accepted             C++    0 ms    8.3 MB    + Notes
    a few seconds ago
```

4.Maximum Subarray.

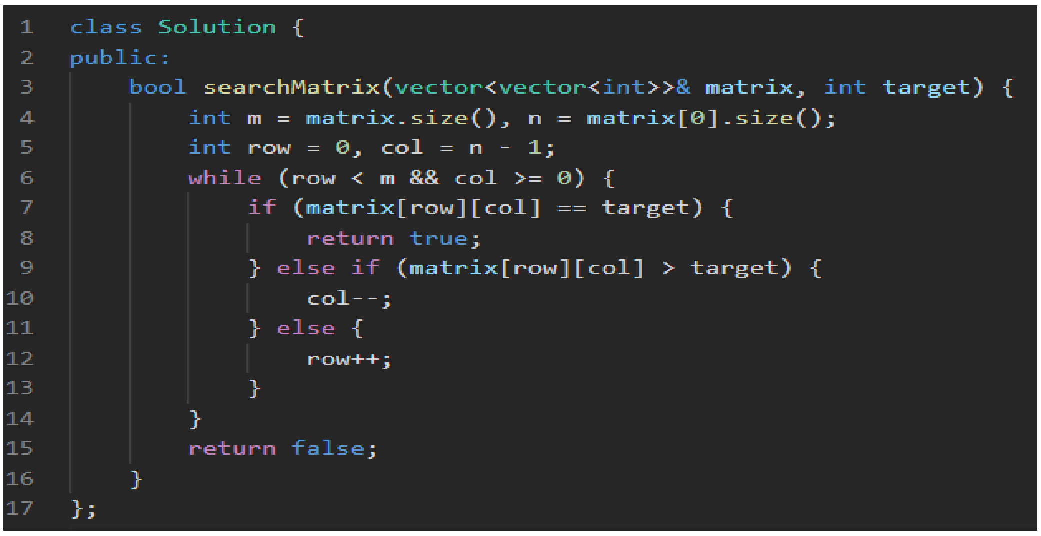


```cpp
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int maxSum = nums[0];
        int currentSum = 0;

        for (int num : nums) {
            currentSum = max(num, currentSum + num);
            maxSum = max(maxSum, currentSum);
        }

        return maxSum;
    }
};
```

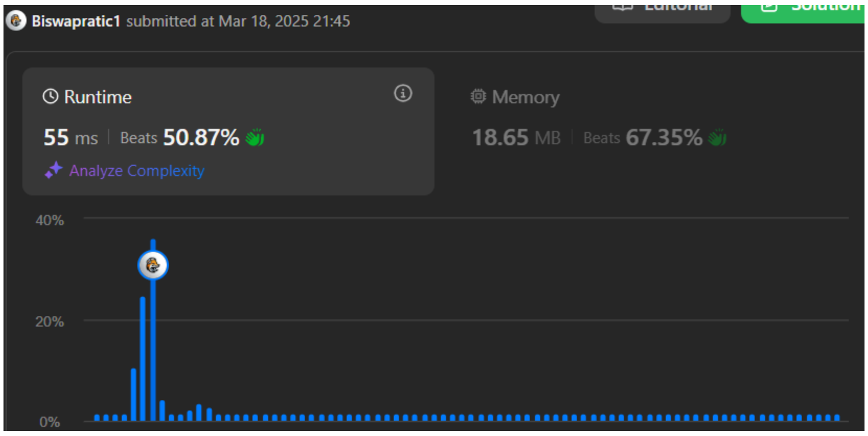


```
1   Accepted        C++    0 ms    71.7 MB
    Mar 10, 2025
```

5.Search a 2D Matrix II.



```cpp
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        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;
    }
};
```


Biswapratic1 submitted at Mar 18, 2025 21:45

Runtime

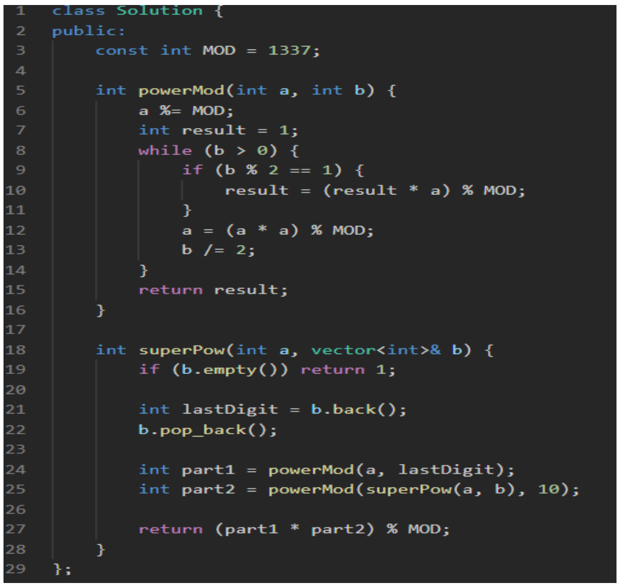
55 ms | Beats 50.87%

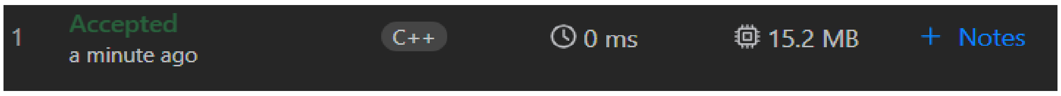
Analyze Complexity

Memory

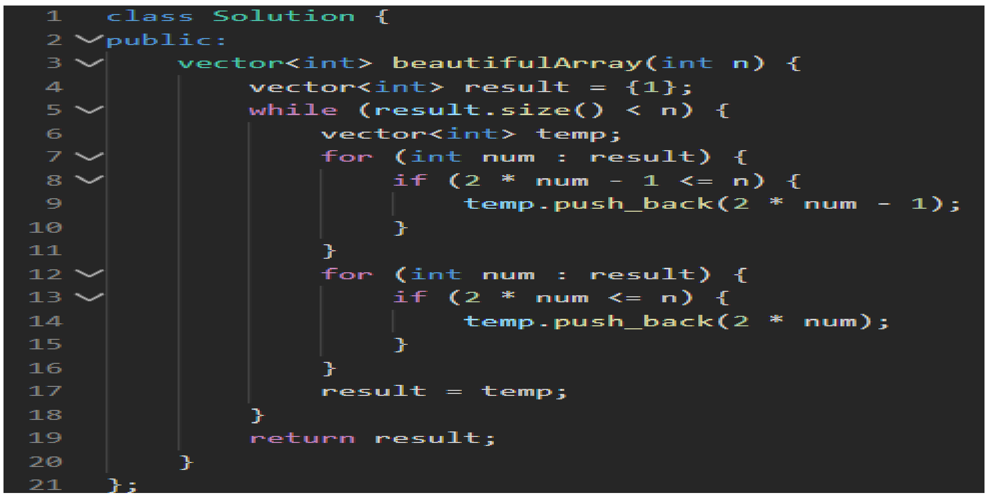
18.65 MB | Beats 67.35%

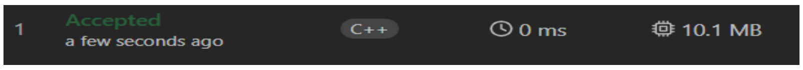
6.Super Pow.


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

    int powerMod(int a, int b) {
        a %= MOD;
        int result = 1;
        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) {
        if (b.empty()) return 1;

        int lastDigit = b.back();
        b.pop_back();

        int part1 = powerMod(a, lastDigit);
        int part2 = powerMod(superPow(a, b), 10);

        return (part1 * part2) % MOD;
    }
};
```

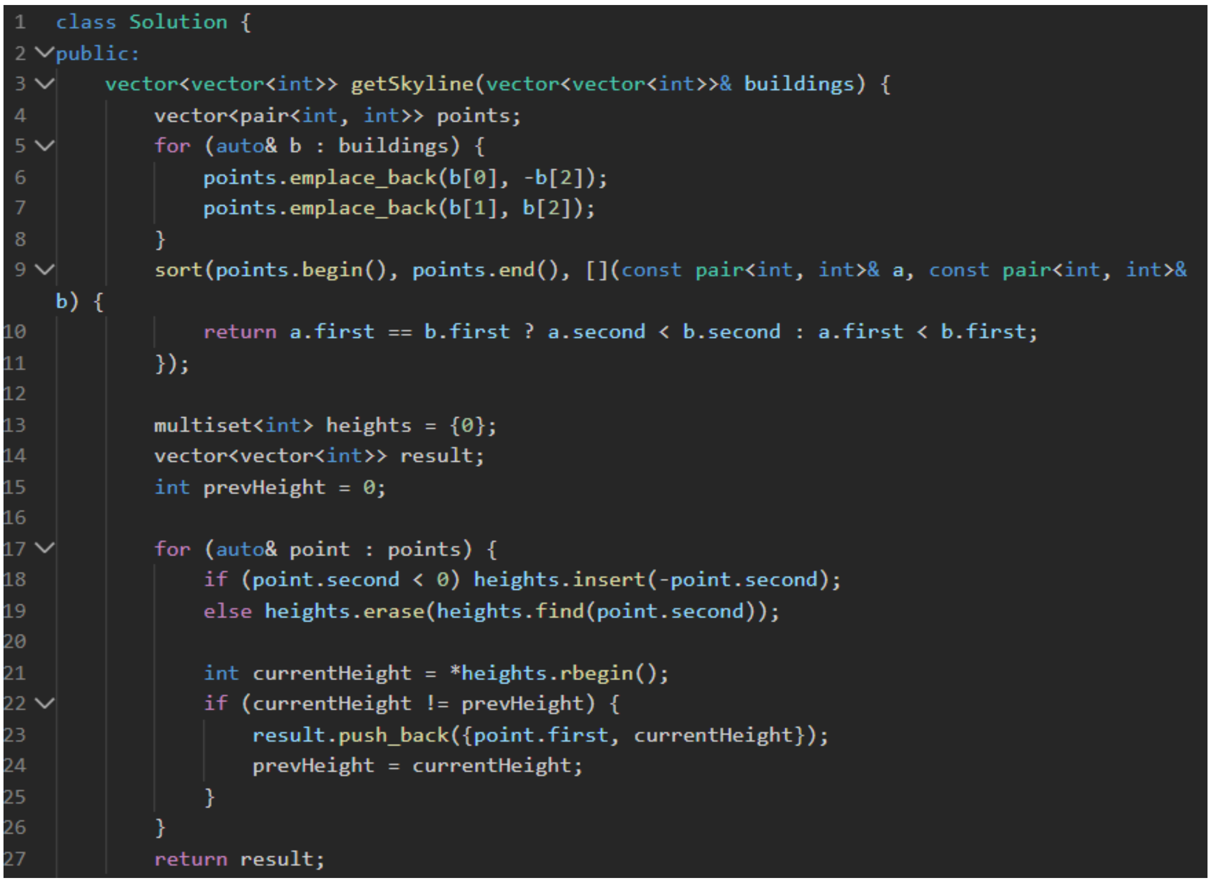

1 Accepted a minute ago C++ 0 ms 15.2 MB + Notes

7.Beautiful Array.

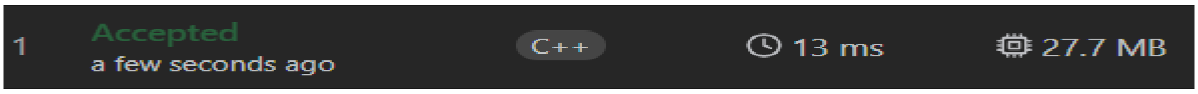

```cpp
class Solution {
public:
    vector<int> beautifulArray(int n) {
        vector<int> result = {1};
        while (result.size() < n) {
            vector<int> temp;
            for (int num : result) {
                if (2 * num - 1 <= n) {
                    temp.push_back(2 * num - 1);
  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


1 Accepted a few seconds ago C++ 0 ms 10.1 MB

8.The Skyline Problem.

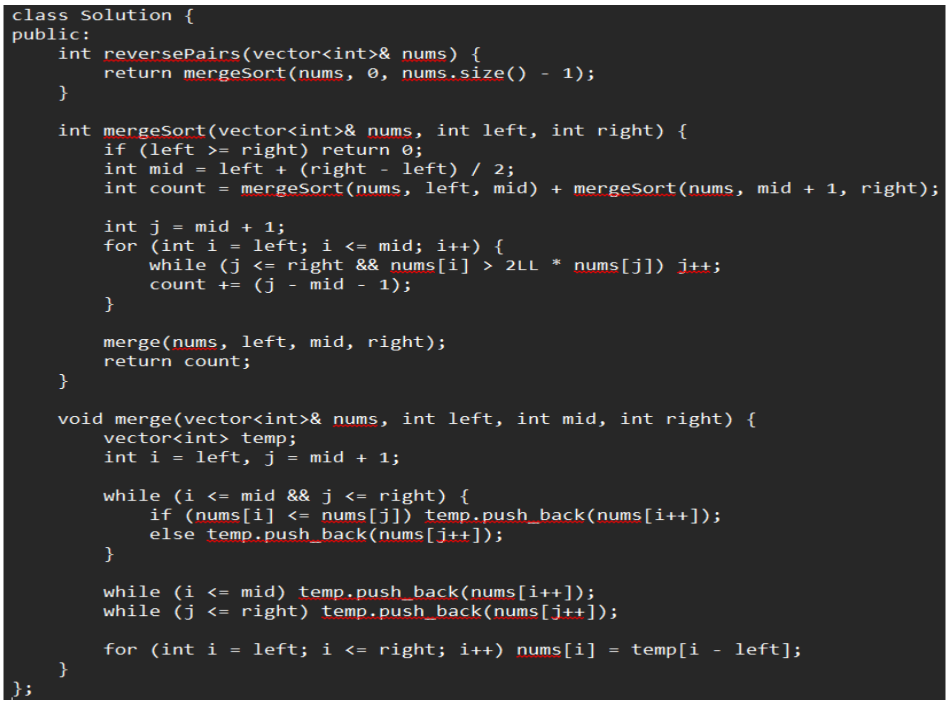


```cpp
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<pair<int, int>> points;
        for (auto& b : buildings) {
            points.emplace_back(b[0], -b[2]);
            points.emplace_back(b[1], b[2]);
        }
        sort(points.begin(), points.end(), [](const pair<int, int>& a, const pair<int, int>& b) {
            return a.first == b.first ? a.second < b.second : a.first < b.first;
        });

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

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

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

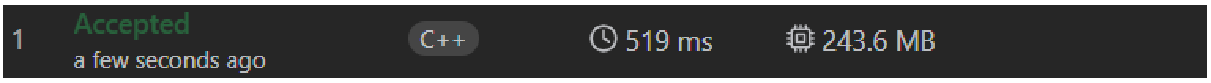


```
1    Accepted                    C++        13 ms        27.7 MB
     a few seconds ago
```

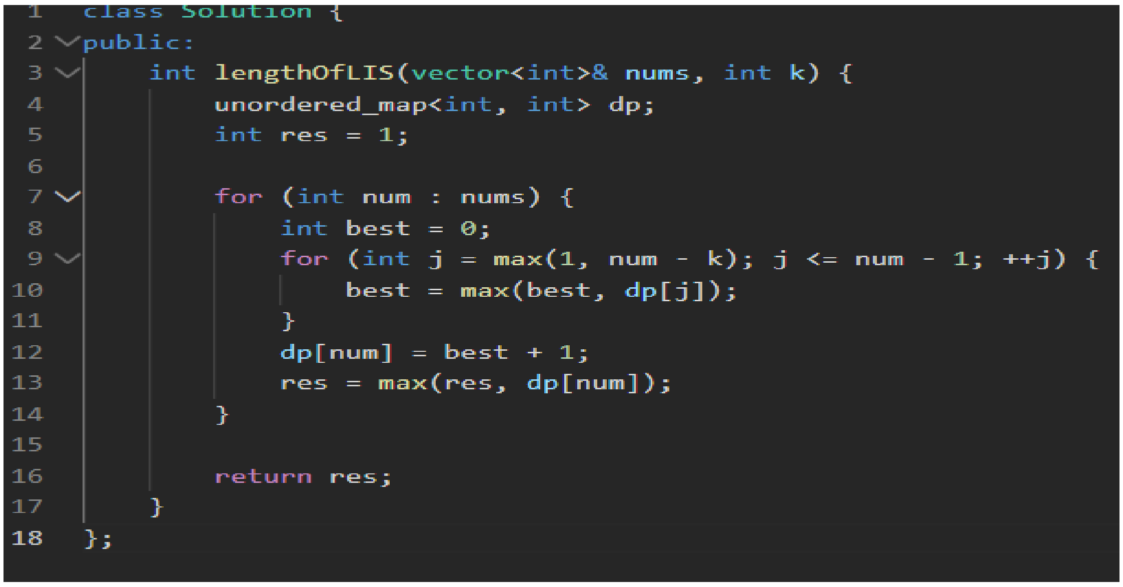
9.Reverse Pairs.

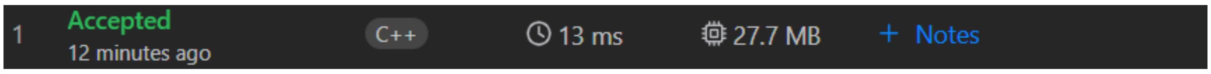


```cpp
class Solution {
public:
    int reversePairs(vector<int>& nums) {
        return mergeSort(nums, 0, nums.size() - 1);
    }

    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);

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

        merge(nums, left, mid, right);
        return count;
    }

    void merge(vector<int>& nums, int left, int mid, int right) {
        vector<int> temp;
        int i = left, j = mid + 1;

        while (i <= mid && j <= right) {
            if (nums[i] <= nums[j]) temp.push_back(nums[i++]);
            else temp.push_back(nums[j++]);
        }

        while (i <= mid) temp.push_back(nums[i++]);
        while (j <= right) temp.push_back(nums[j++]);

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


```
1    Accepted                 C++        519 ms        243.6 MB
     a few seconds ago
```

10.Longest Increasing Subsequence II.


```cpp
class Solution {
public:
    int lengthOfLIS(vector<int>& nums, int k) {
        unordered_map<int, int> dp;
        int res = 1;

        for (int num : nums) {
            int best = 0;
            for (int j = max(1, num - k); j <= num - 1; ++j) {
                best = max(best, dp[j]);
            }
            dp[num] = best + 1;
            res = max(res, dp[num]);
        }

        return res;
    }
};
```


```
1    Accepted                 C++        13 ms        27.7 MB        + Notes
     12 minutes ago
```