**Experiment - 2**

**Lokesh Gupta**

**22BCS16079**

**IOT\_602\_A**

**1.[Longest Nice Substring](https://leetcode.com/problems/longest-nice-substring/description/)**

class Solution {

public:

bool isNice(string s) {

for (char c : s) {

if (find(s.begin(), s.end(), tolower(c)) == s.end() ||

find(s.begin(), s.end(), toupper(c)) == s.end()) {

return false;

}

}

return true;

string longestNiceSubstring(string s) {

int n = s.size();

string longest = "";

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) && sub.size() > longest.size()) {

longest = sub;

}

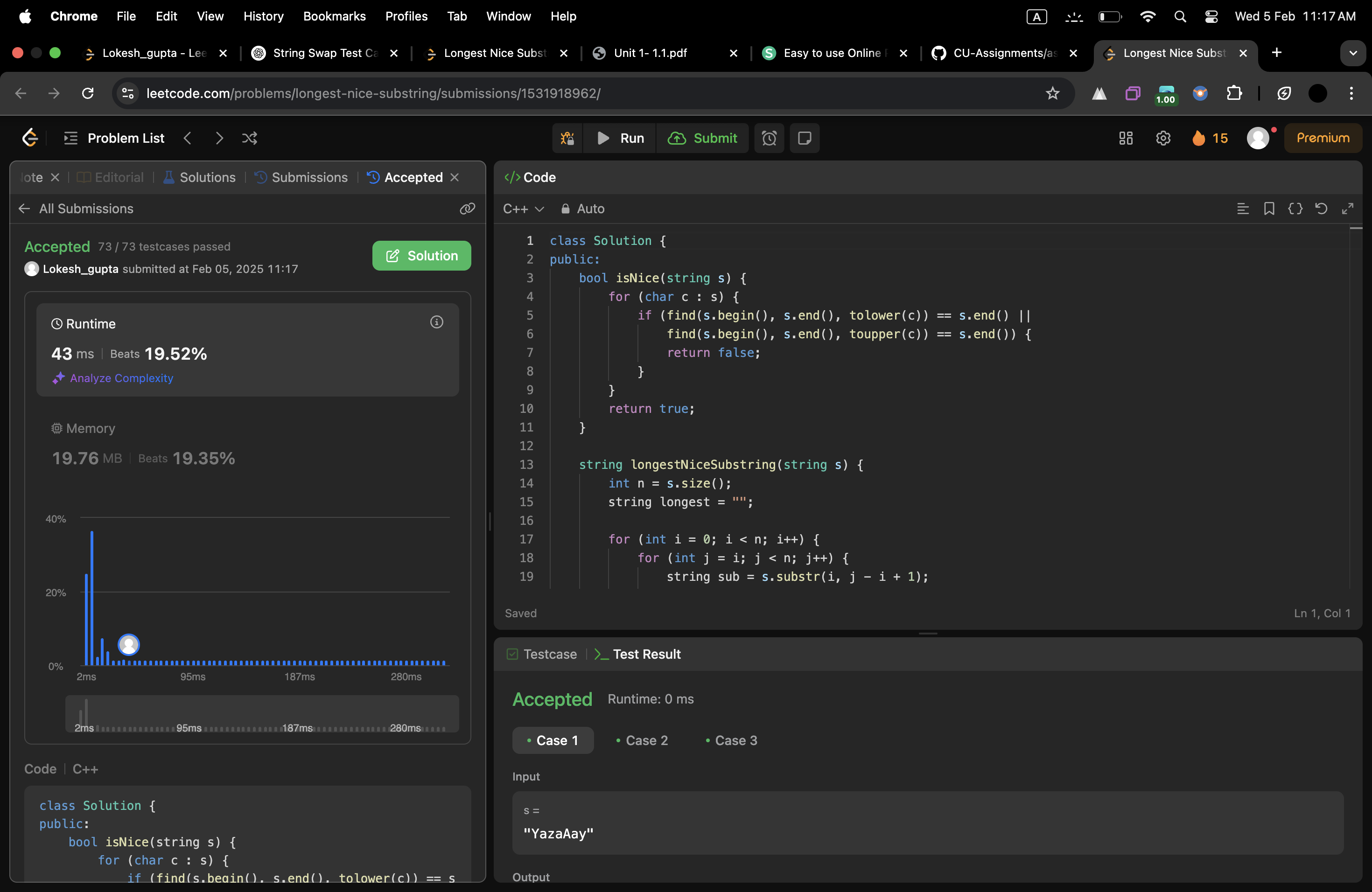
}

}

return longest;

}

};

****

**2.[Reverse Bits](https://leetcode.com/problems/reverse-bits/description/)**

class Solution {

public:

uint32\_t reverseBits(uint32\_t n) {

uint32\_t result = 0;

for (int i = 0; i < 32; i++) {

int lastBit = n & 1;

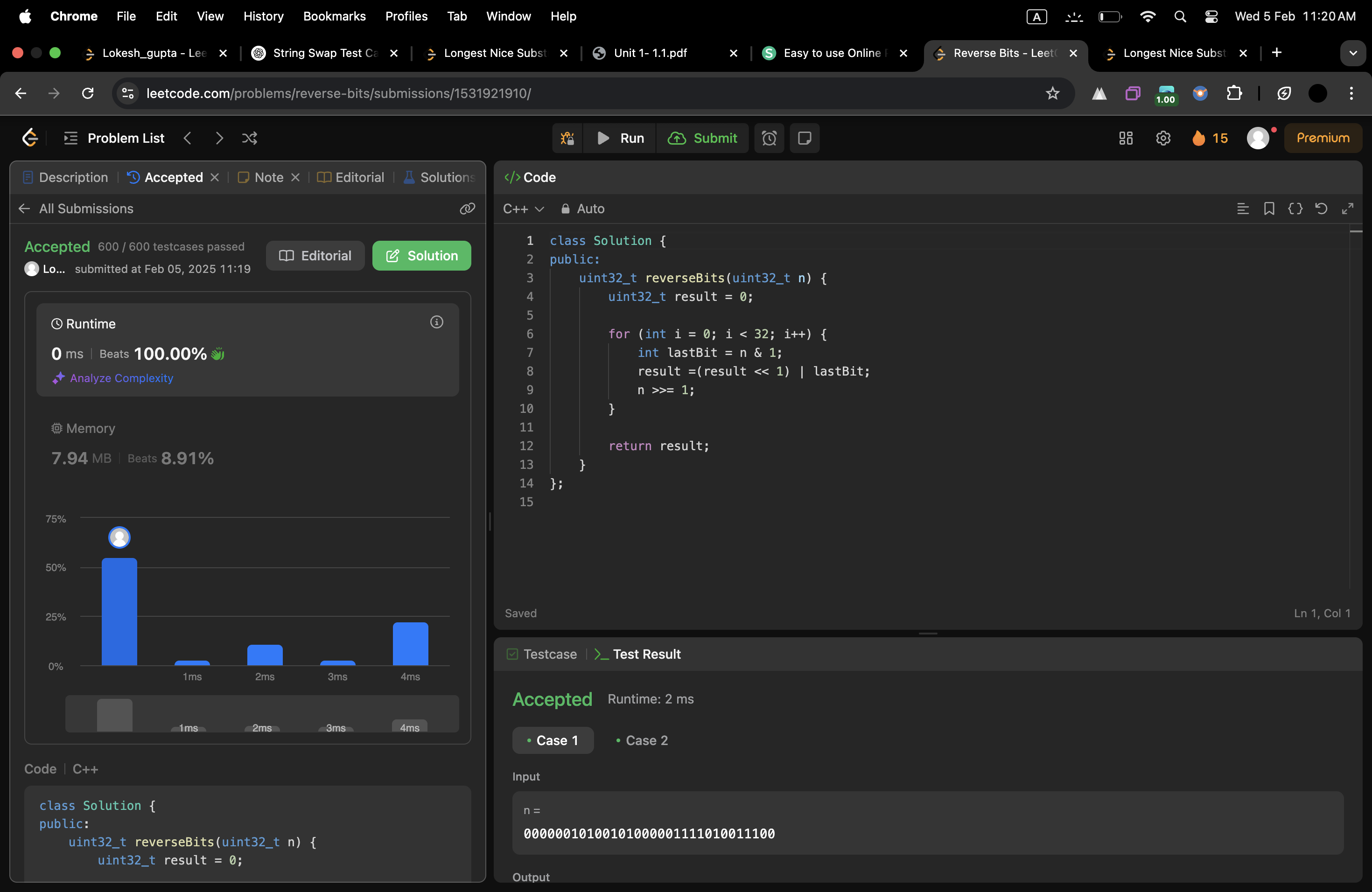
result =(result << 1) | lastBit;

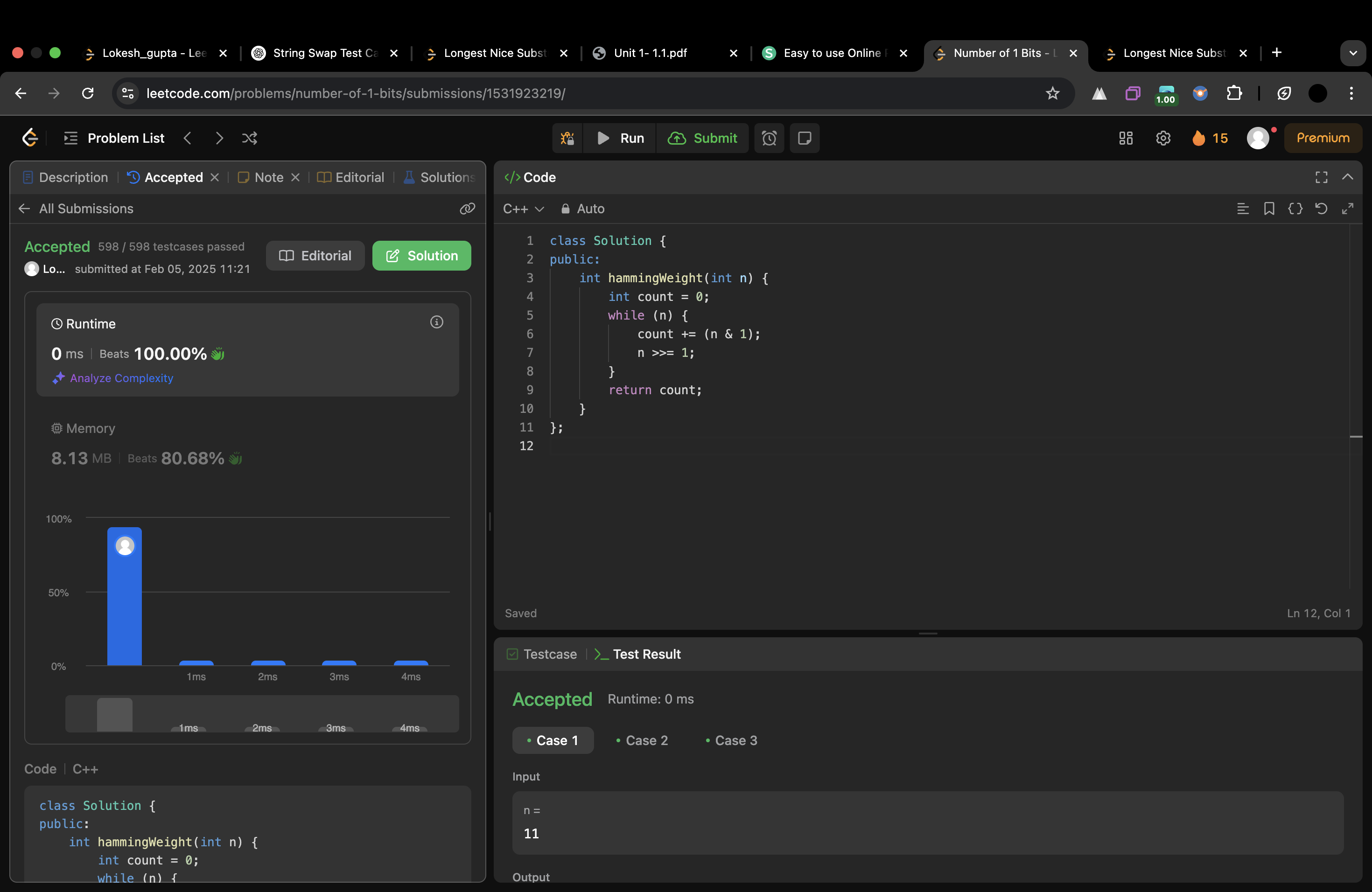
n >>= 1;

}

return result;

}

};

**3.[Number of 1 Bits](https://leetcode.com/problems/number-of-1-bits/description/)**

class Solution {

public:

int hammingWeight(int n) {

int count = 0;

while (n) {

count += (n & 1);

n >>= 1;

}

return count;

}

};

**4.[Maximum Subarray](https://leetcode.com/problems/maximum-subarray/description/)**

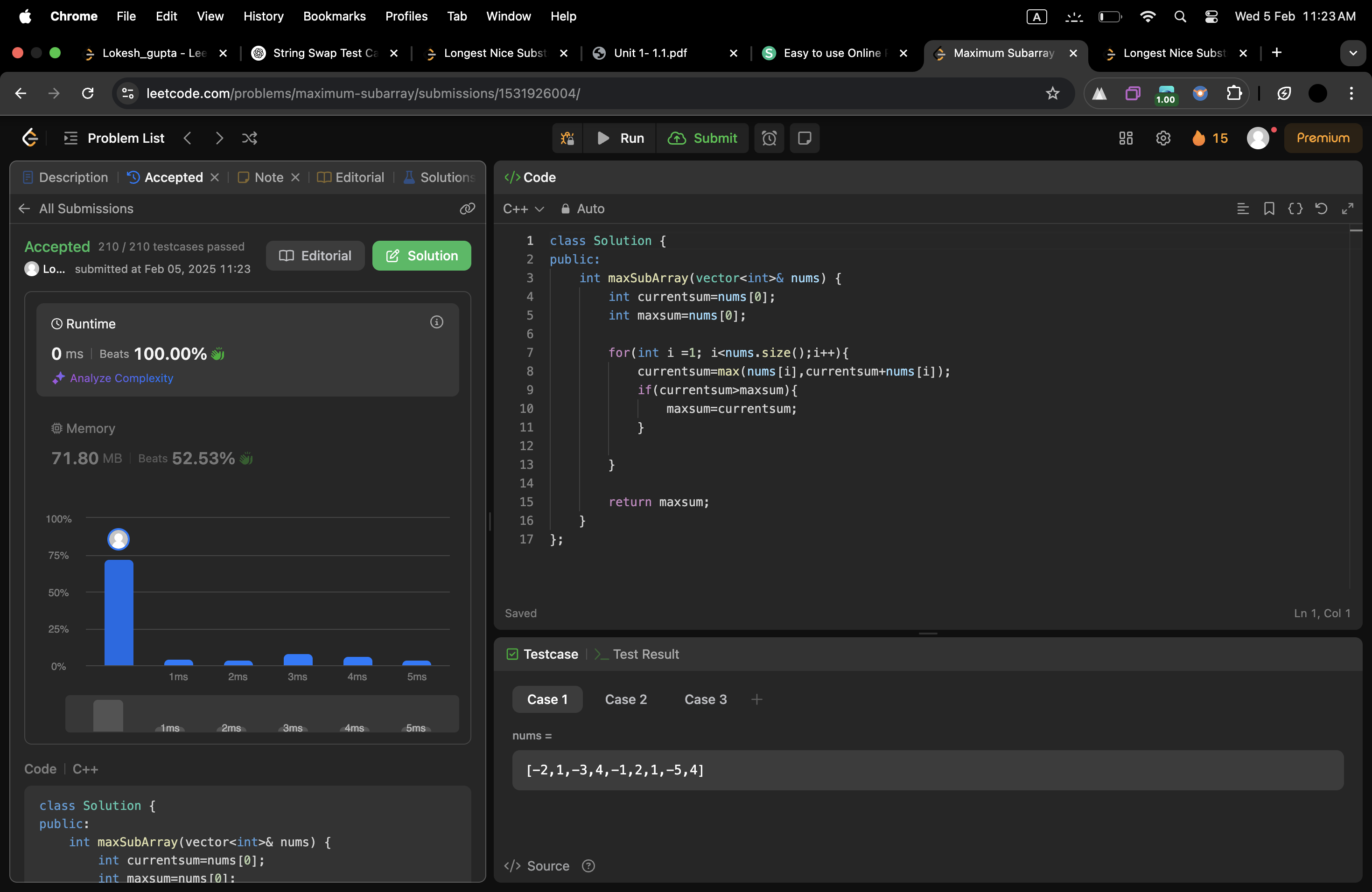
class Solution {

public:

int maxSubArray(vector<int>& nums) {

int currentsum=nums[0];

int maxsum=nums[0];



for(int i =1; i<nums.size();i++){

currentsum=max(nums[i],currentsum+nums[i]);

if(currentsum>maxsum){

maxsum=currentsum;

}

}

return maxsum;

}

};

**5.[Search a 2D Matrix II](https://leetcode.com/problems/search-a-2d-matrix-ii/description/)**

class Solution {

public:

bool searchMatrix(vector<vector<int>>& matrix, int target) {

int m = matrix.size(), n = matrix[0].size();

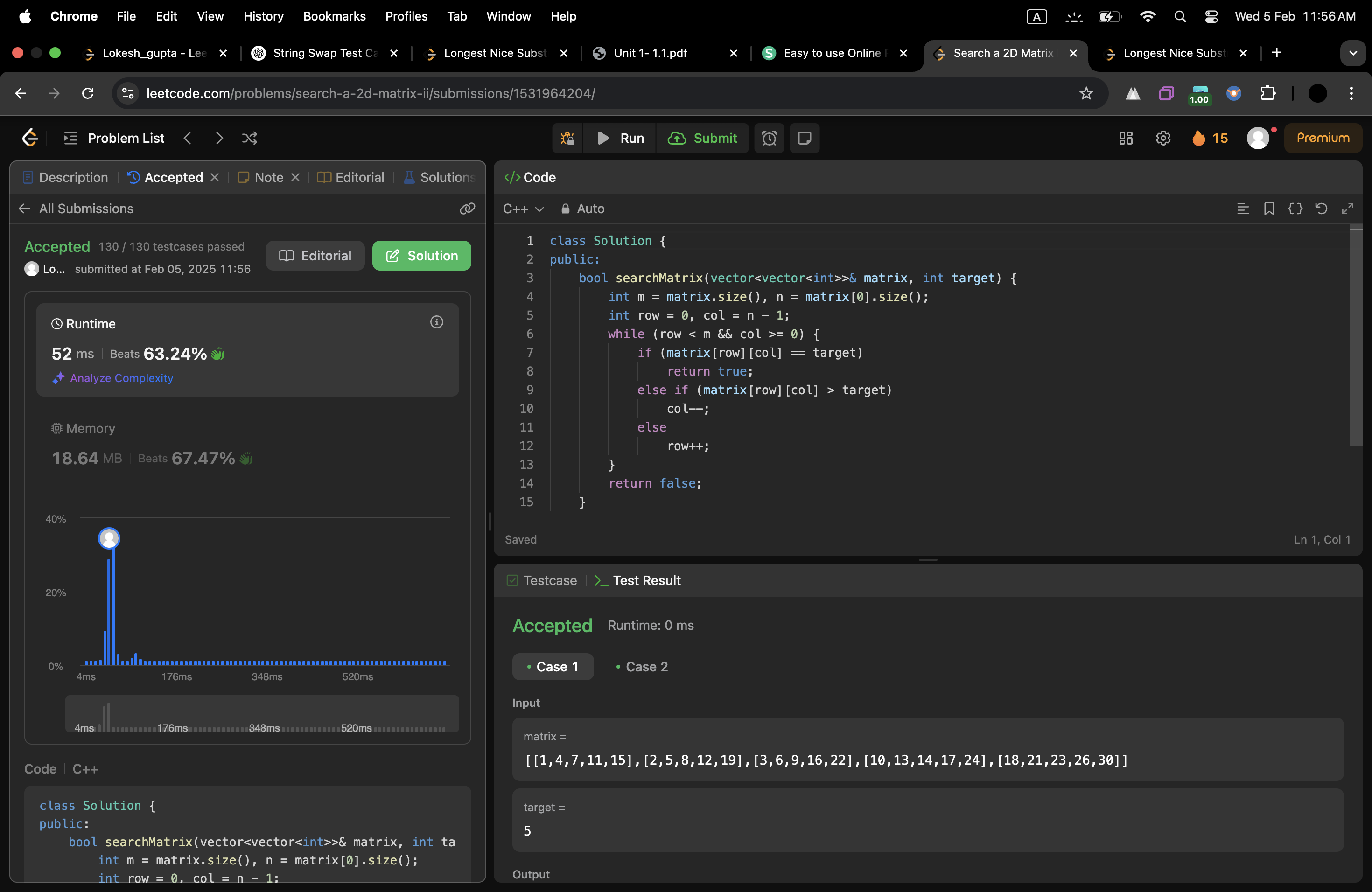
int row = 0, col = n - 1;

while (row < m && col >= 0) {

if (matrix[row][col] == target)

return true;

else if (matrix[row][col] > target)



col--;

else

row++;

}

return false;

}

};

**6.[Super Pow](https://leetcode.com/problems/super-pow/)**

class Solution {

public:

int modPow(int a, int b, int mod) {

int res = 1;

a %= mod;

while (b) {

if (b % 2) res = (res \* a) % mod;

a = (a \* a) % mod;

b /= 2;

}

return res; }

int superPow(int a, vector<int>& b) {

int mod = 1337, res = 1;

for (int digit : b) {

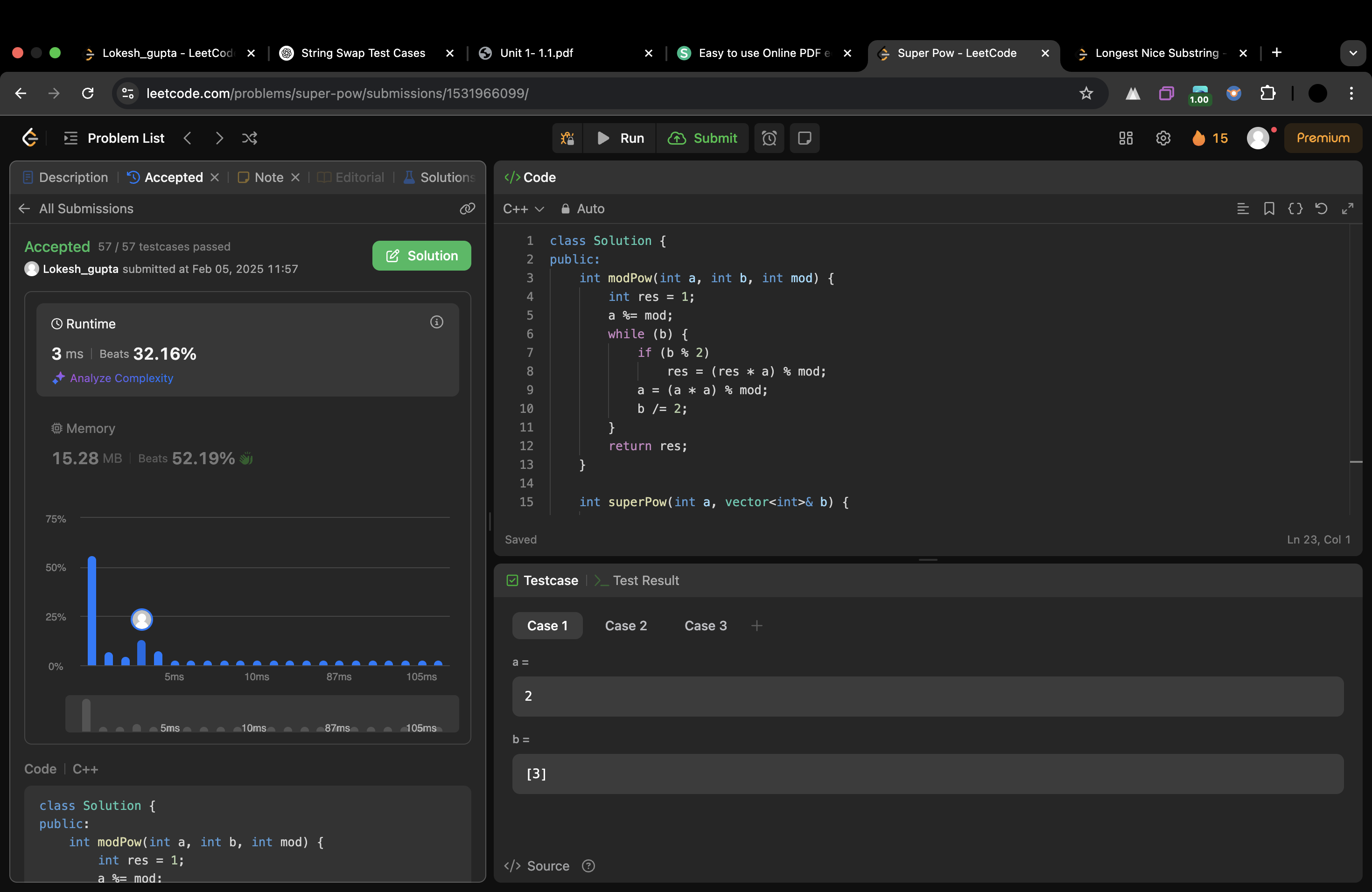
res = modPow(res, 10, mod) \* modPow(a, digit, mod) % mod;

}

return res;

}

};

****

**7.[Beautiful Array](https://leetcode.com/problems/beautiful-array/)**

class Solution {

public:

std::vector<int> beautifulArray(int n) {

std::vector<int> result={1};

while (result.size() < n) {

std::vector<int> temp;

for (int i : result) {

if (i \* 2 - 1 <= n) {

temp.push\_back(i \* 2 - 1);

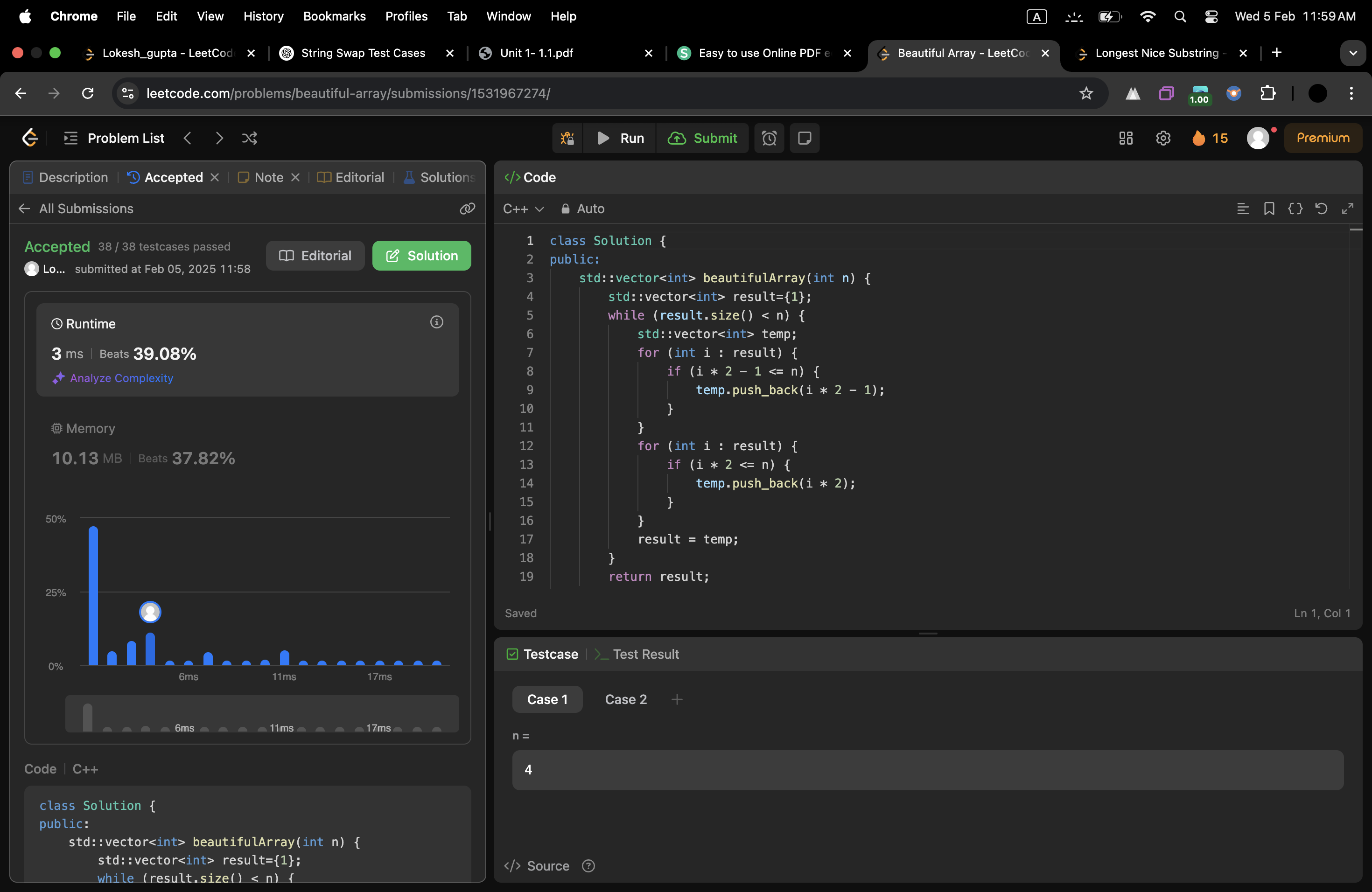
}

}

for (int i : result) {

if (i \* 2 <= n) {

temp.push\_back(i \* 2);

}

}

result = temp;

}

return result;

}

};

**8.[The Skyline Problem](https://leetcode.com/problems/the-skyline-problem/)**

class Solution {

public:

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

vector<tuple<int,int,int>> events;

for(auto& b : buildings){

events.push\_back({b[0], -b[2], b[1]});

events.push\_back({b[1], b[2], 0});

}

sort(events.begin(), events.end(), [](auto a, auto b){

if(get<0>(a) != get<0>(b))

return get<0>(a) < get<0>(b);

return get<1>(a) < get<1>(b);

});

vector<vector<int>> res;

priority\_queue<pair<int,int>> pq;

pq.push({0, INT\_MAX});

int prev = 0;

for(auto& e : events){

int x, h, r;

tie(x, h, r) = e;

while(!pq.empty() && x >= pq.top().second)

pq.pop();

if(h < 0)

pq.push({-h, r});

int cur = pq.top().first;

if(cur != prev){

res.push\_back({x, cur});

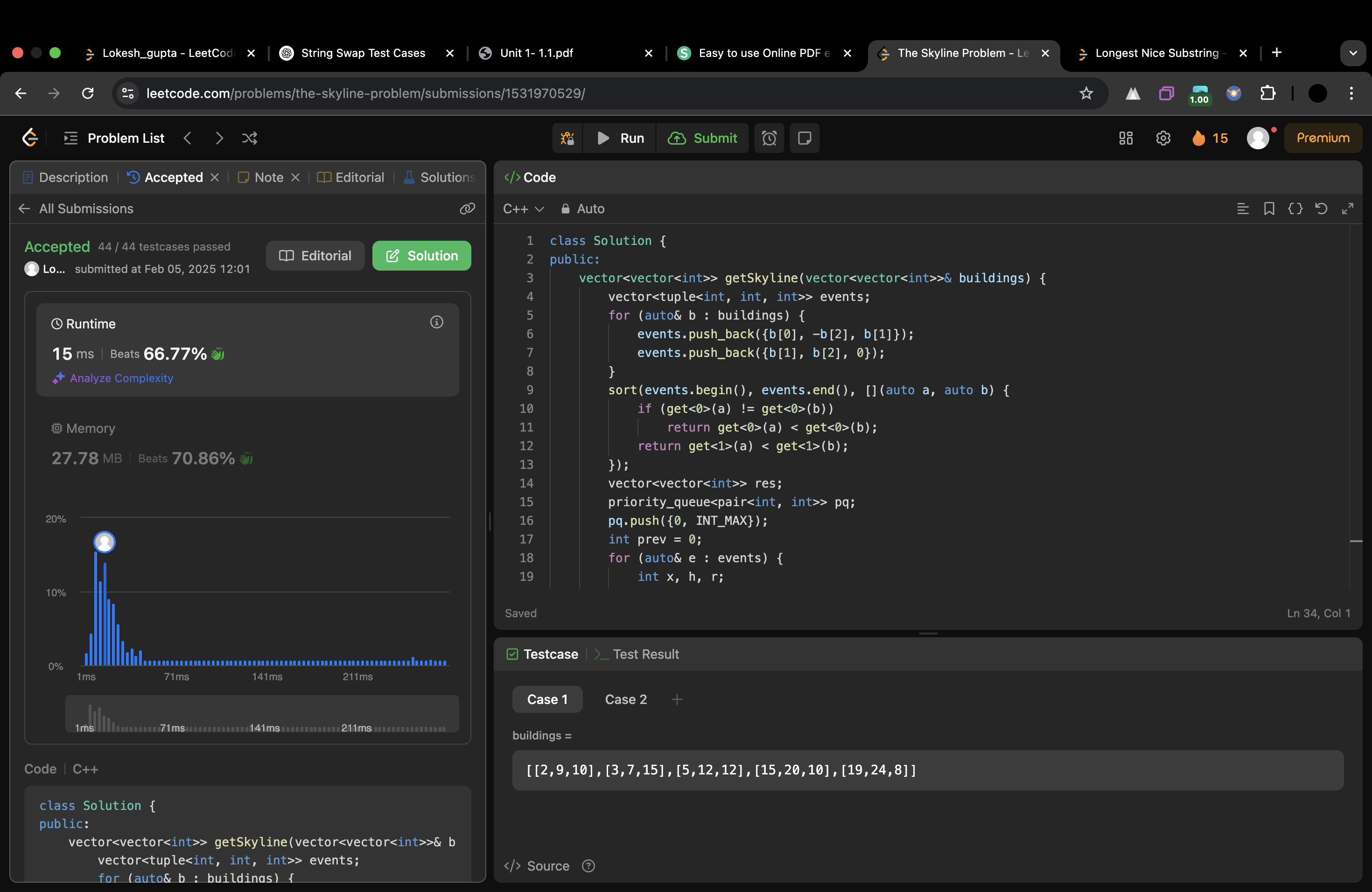
prev = cur;

}

}

return res;

}

};

**9.[Reverse Pairs](https://leetcode.com/problems/reverse-pairs/)**

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));

}

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 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;

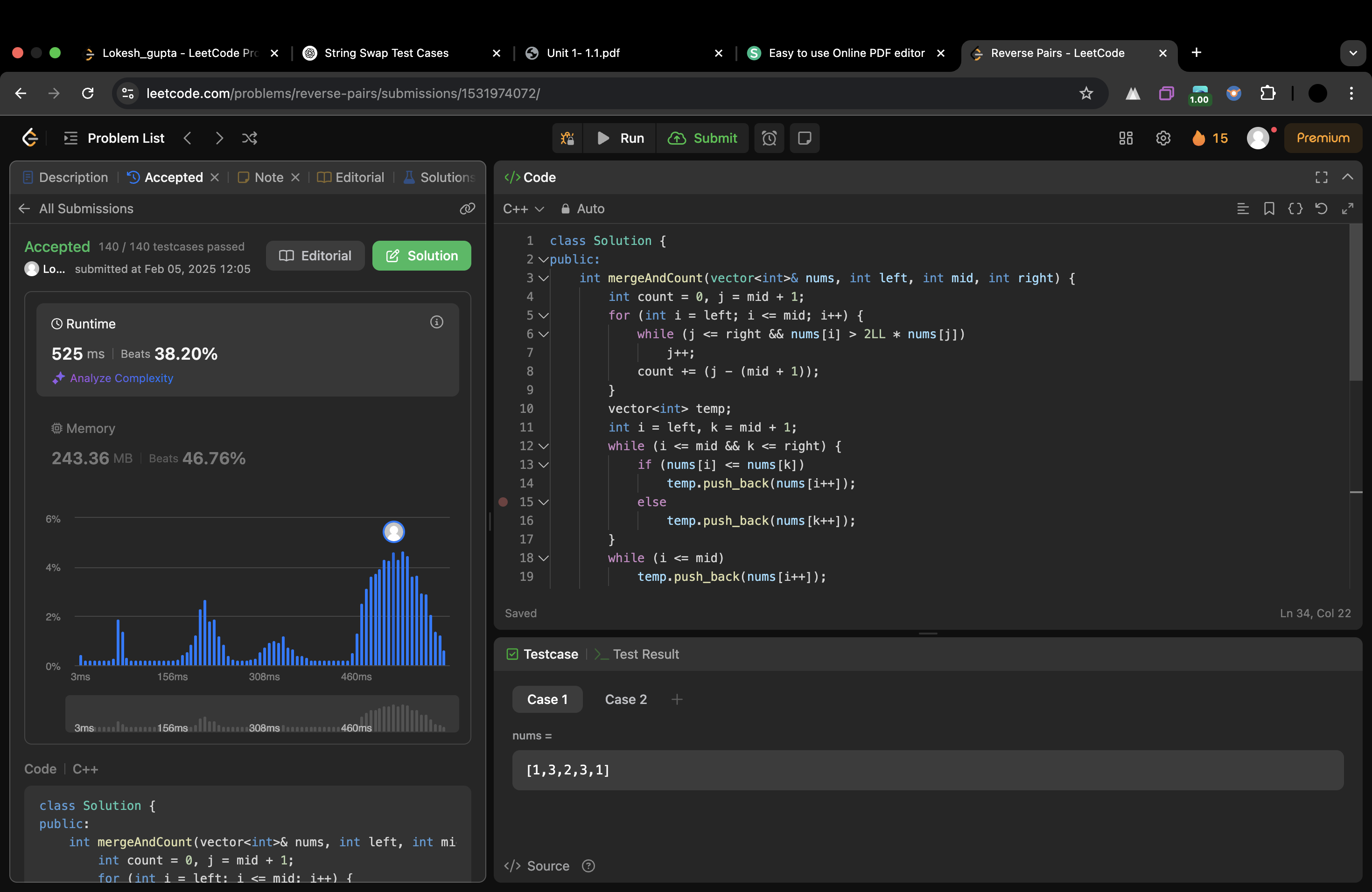
}

int reversePairs(vector<int>& nums) {

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

}

};



**10.[Merge Sorted Array](https://leetcode.com/problems/merge-sorted-array/)**

class Solution {

public:

void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {

int i=0,j=0,k=0;

vector<int> ans(m+n);

while(i<m && j <n){

if(nums1[i]<=nums2[j]){

ans[k++]=nums1[i++];

}

else{

ans[k++]=nums2[j++];

}

}

while(i < m){

ans[k++] = nums1[i++];

}

while(j < n){

ans[k++] = nums2[j++];

}

for (int x = 0; x < m + n; ++x) {

nums1[x] = ans[x];

}

}

};