**Name –** Sidhant

**UID –** 22BCS15880

**Section –** 601-‘B’

**Experiment –** 1

1. **Remove duplicates from a sorted array:**

**Problem Link :** <https://leetcode.com/problems/remove-duplicates-from-sorted-array/description/>

**Solution:**

class Solution {

public int removeDuplicates(int[] nums) {

if(nums == null || nums.length == 0) return 0;

int res = 0;

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

if(nums[res] == nums[i]) continue;

res++;

nums[res] = nums[i];

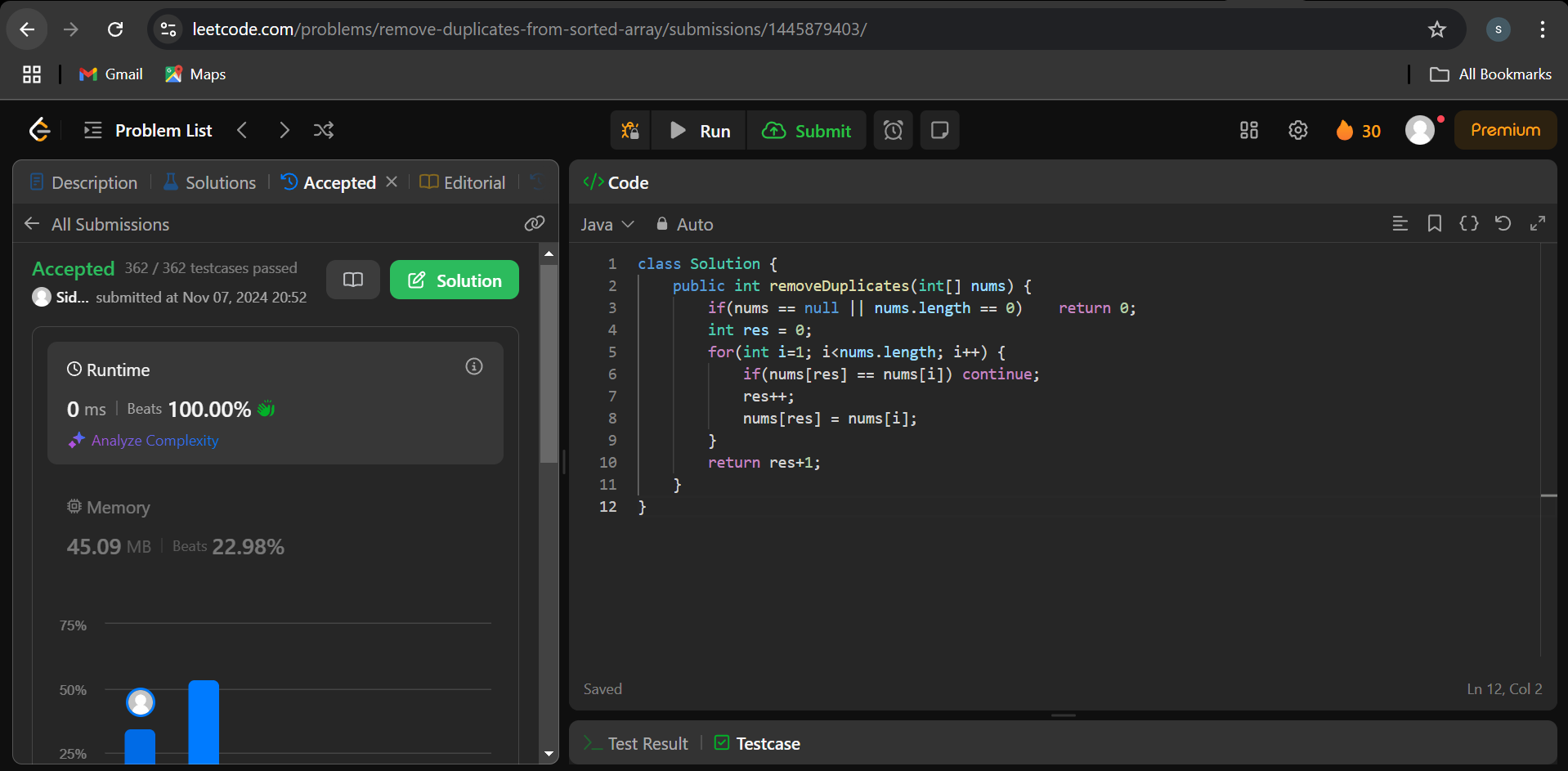
}

return res+1;

}

}

**Output:**

****

1. **Implementing insertion sort:**

**Problem Link :** <https://www.geeksforgeeks.org/problems/insertion-sort/1>

**Solution:**

void insertionSort(int arr[], int n)

{

for (int i = 1; i < n; ++i) {

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

j = j - 1;

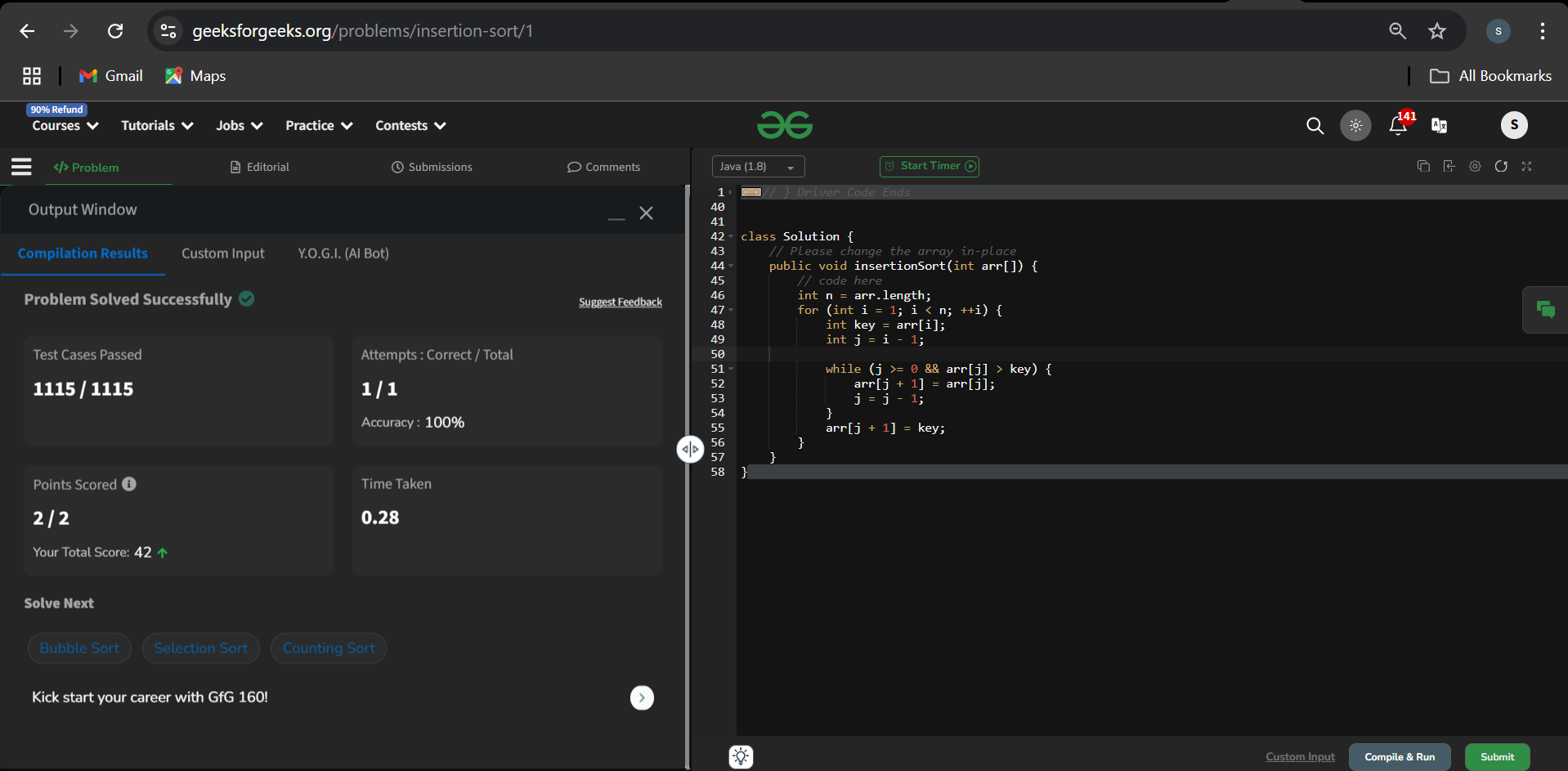
}

arr[j + 1] = key;

}

}

**Output:**



1. **Contains duplicate:**

**Problem Link :** <https://leetcode.com/problems/contains-duplicate/description/>

**Code:**

class Solution {

public boolean containsDuplicate(int[] nums) {

int n = nums.length;

HashSet<Integer> set = new HashSet<>();

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

set.add(nums[i]);

}

if(n > set.size()) {

return true;

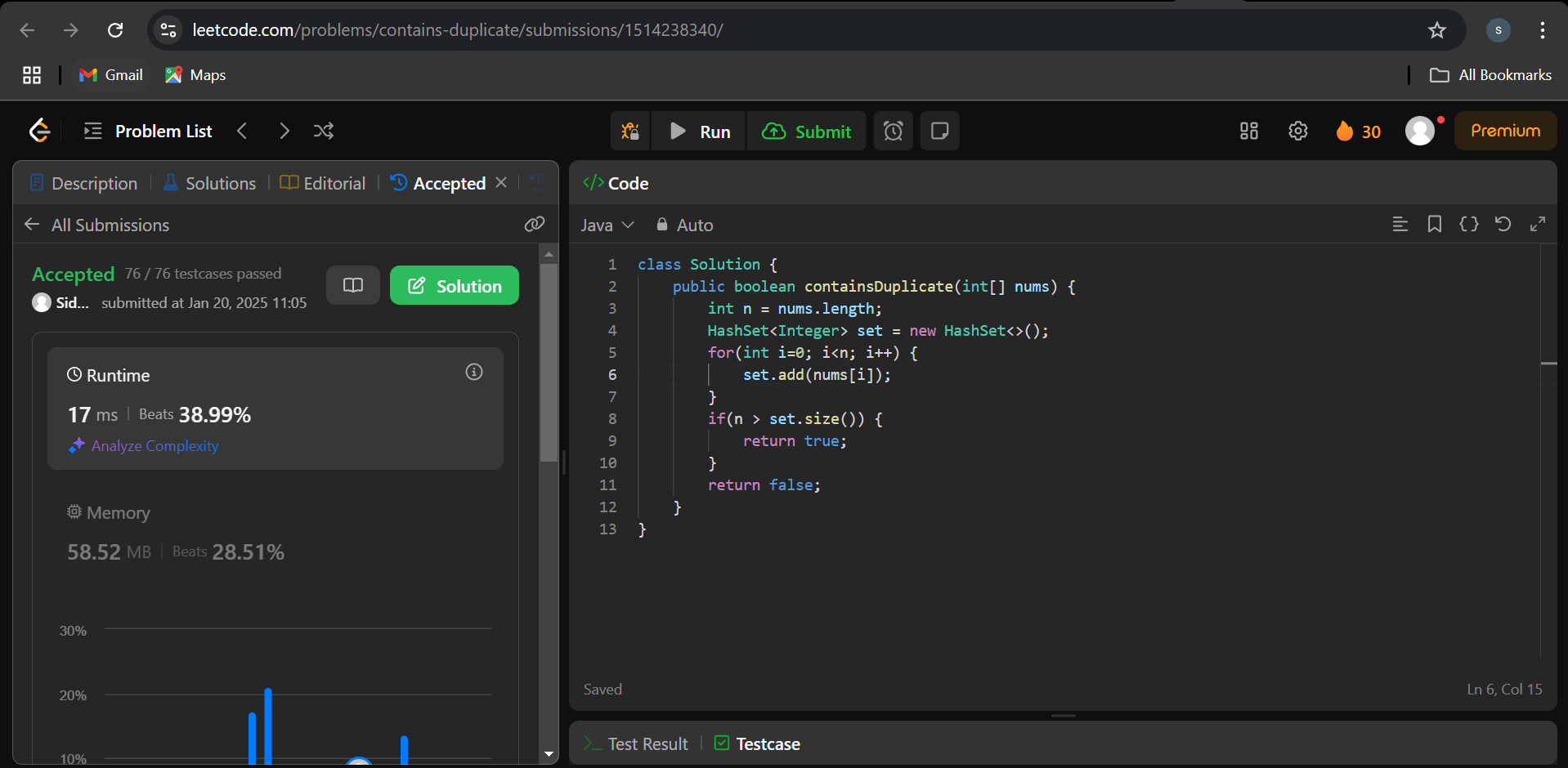
}

return false;

}

}

**Output:**



1. **Two Sum:**

**Problem Link :** <https://leetcode.com/problems/two-sum/description/>

**Code:**

class Solution {

public int[] twoSum(int[] nums, int target) {

int n = nums.length;

// int ans[] = new int[2];

HashMap<Integer,Integer> map = new HashMap<>();

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

int left = target - nums[i];

if(map.containsKey(left)) {

return new int[] {map.get(left), i};

} else {

map.put(nums[i],i);

}

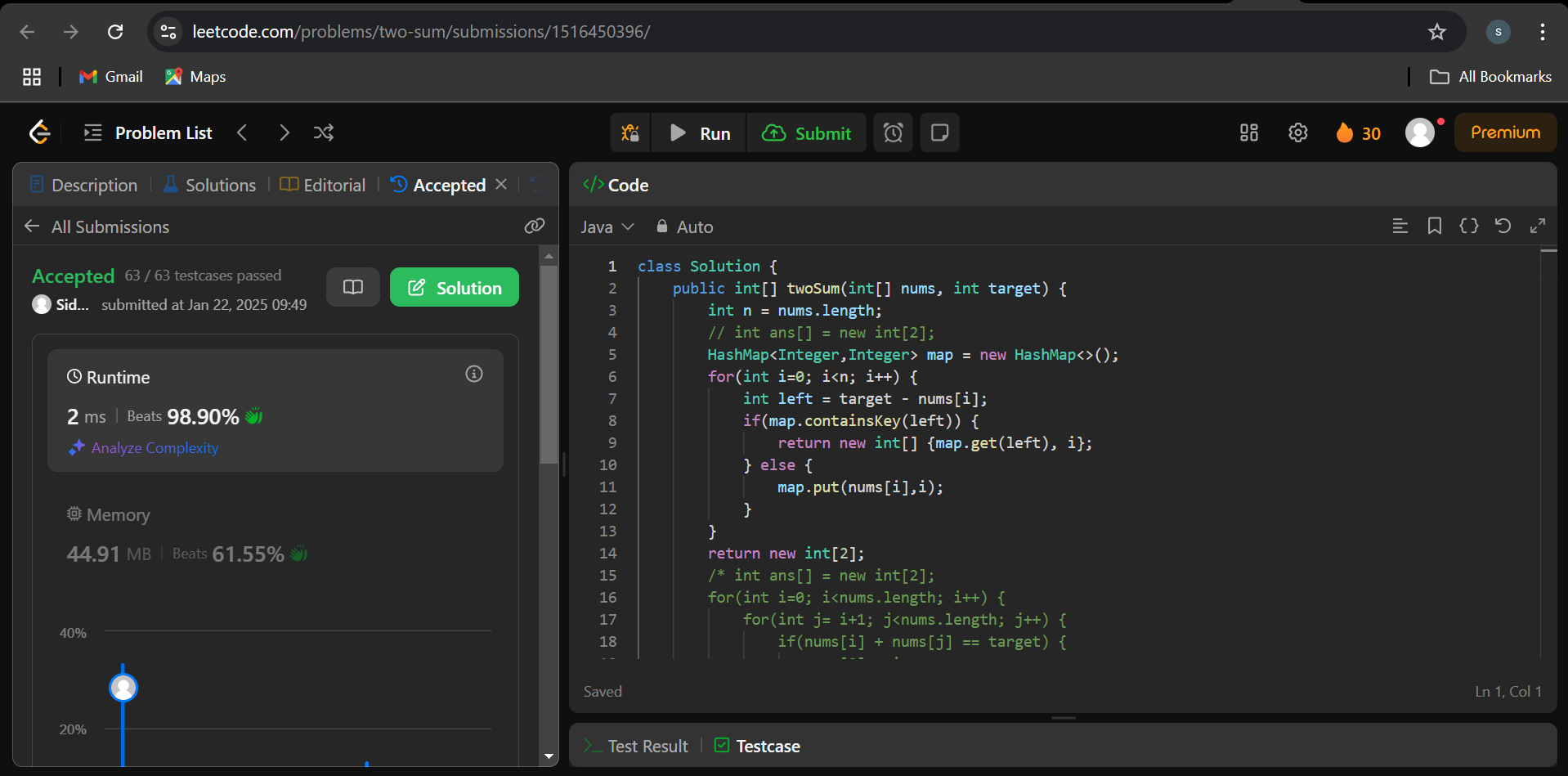
}

return new int[2];

}

}

**Output:**



1. **Jump Game:**

**Problem Link :** <https://leetcode.com/problems/jump-game/description/>

**Code:**

class Solution {

public boolean canJump(int[] nums) {

int jump = 0;

for(int i=0;i<nums.length;i++){

if(i>jump){

return false;

}

jump = Math.max(jump,i + nums[i]);

}

return true;

/\* int n = nums.length;

int[] dp = new int[n+1];

Arrays.fill(dp,-1);

return canJumpMemo(0,nums,dp);

\*/

}

private boolean canJumpMemo(int idx, int[] nums, int[] dp) {

if(idx >= nums.length-1) {

return true;

}

if(dp[idx] != -1) {

return dp[idx] == 0 ? false : true;

}

for(int i=1; i<=nums[idx]; i++) {

if(canJumpMemo(idx+i, nums, dp) == true) {

dp[idx] = 1;

return true;

}

}

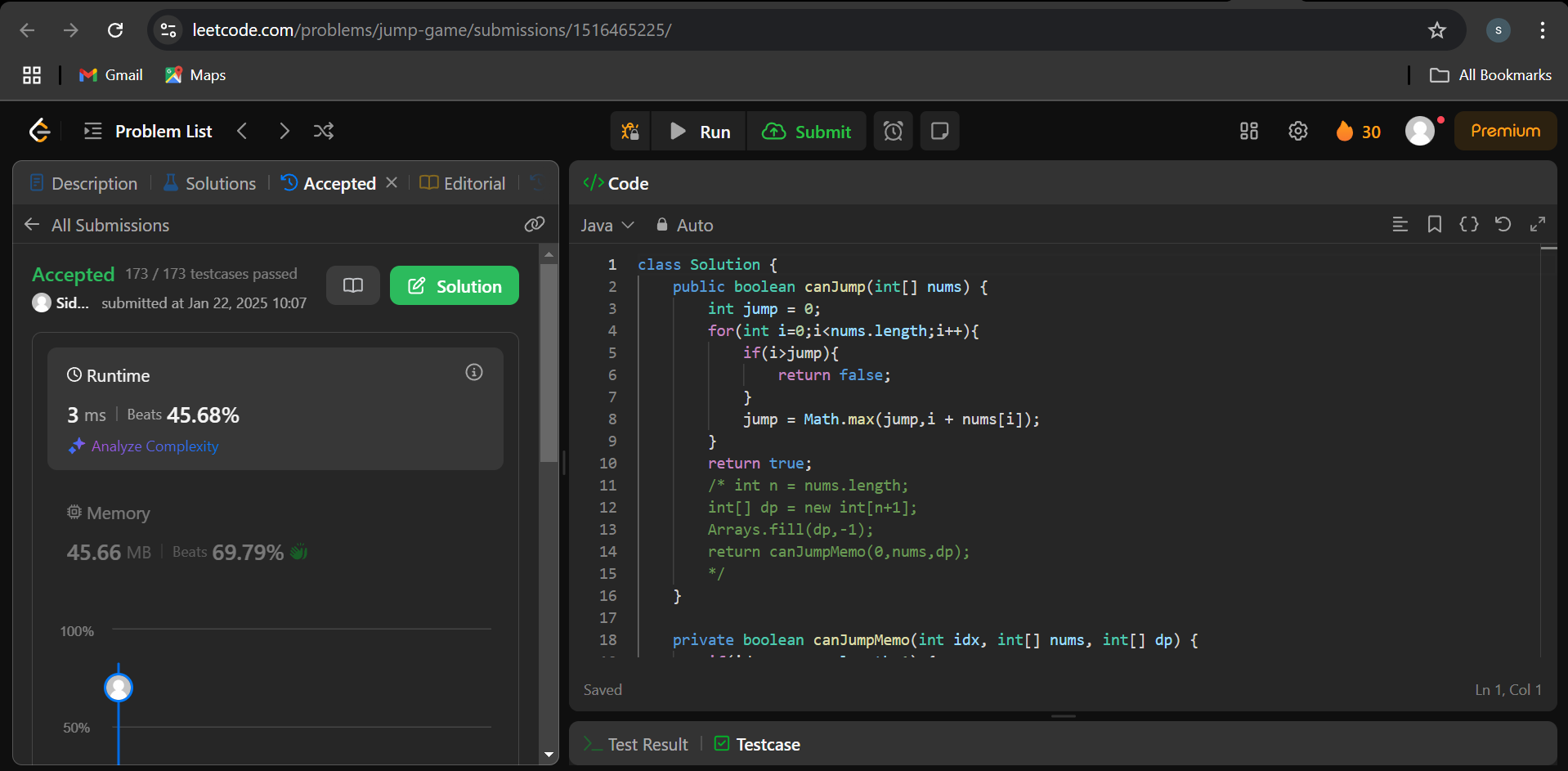
dp[idx] = 0;

return false;

}

}

**Output:**



1. **Majority Elements:**

**Problem Link :** <https://leetcode.com/problems/majority-element/description/>

**Code:**

class Solution {

public int majorityElement(int[] nums) {

HashMap<Integer, Integer> map = new HashMap<>();

for(int i=0; i<nums.length; i++) {

map.put(nums[i], map.getOrDefault(nums[i],0)+1);

}

for(Integer key : map.keySet()) {

if(map.get(key) > nums.length/2) {

return key;

}

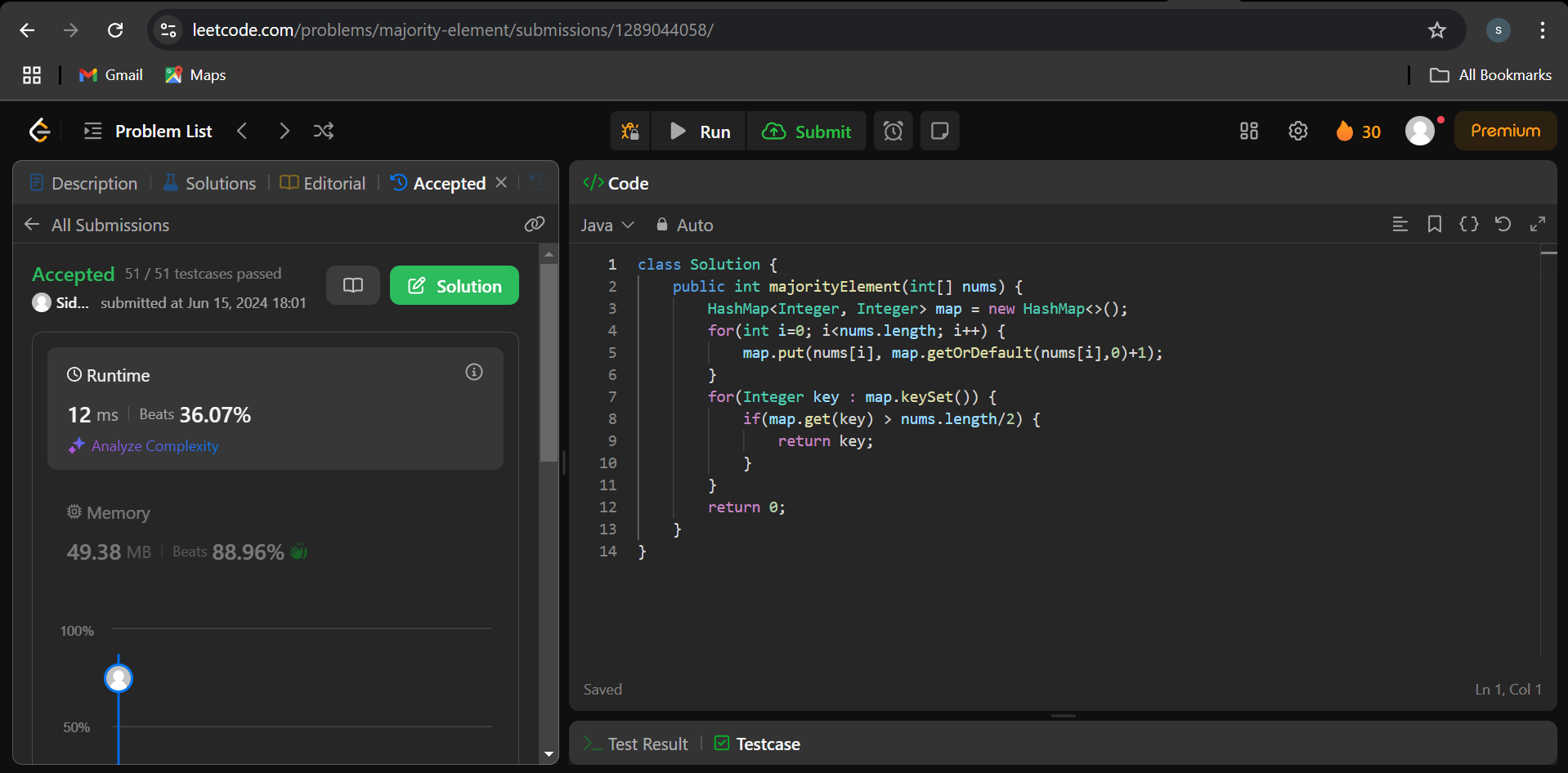
}

return 0;

}

}

**Output:**

****

1. **Valid Palindrome:**

**Problem Link :** <https://leetcode.com/problems/valid-palindrome/description/>

**Code:**

class Solution {

public boolean isPalindrome(String s) {

int i = 0, j = s.length() - 1;

while (i < j) {

if (!Character.isLetterOrDigit(s.charAt(i))) {

++i;

} else if (!Character.isLetterOrDigit(s.charAt(j))) {

--j;

} else if (Character.toLowerCase(s.charAt(i)) != Character.toLowerCase(s.charAt(j))) {

return false;

} else {

++i;

--j;

}

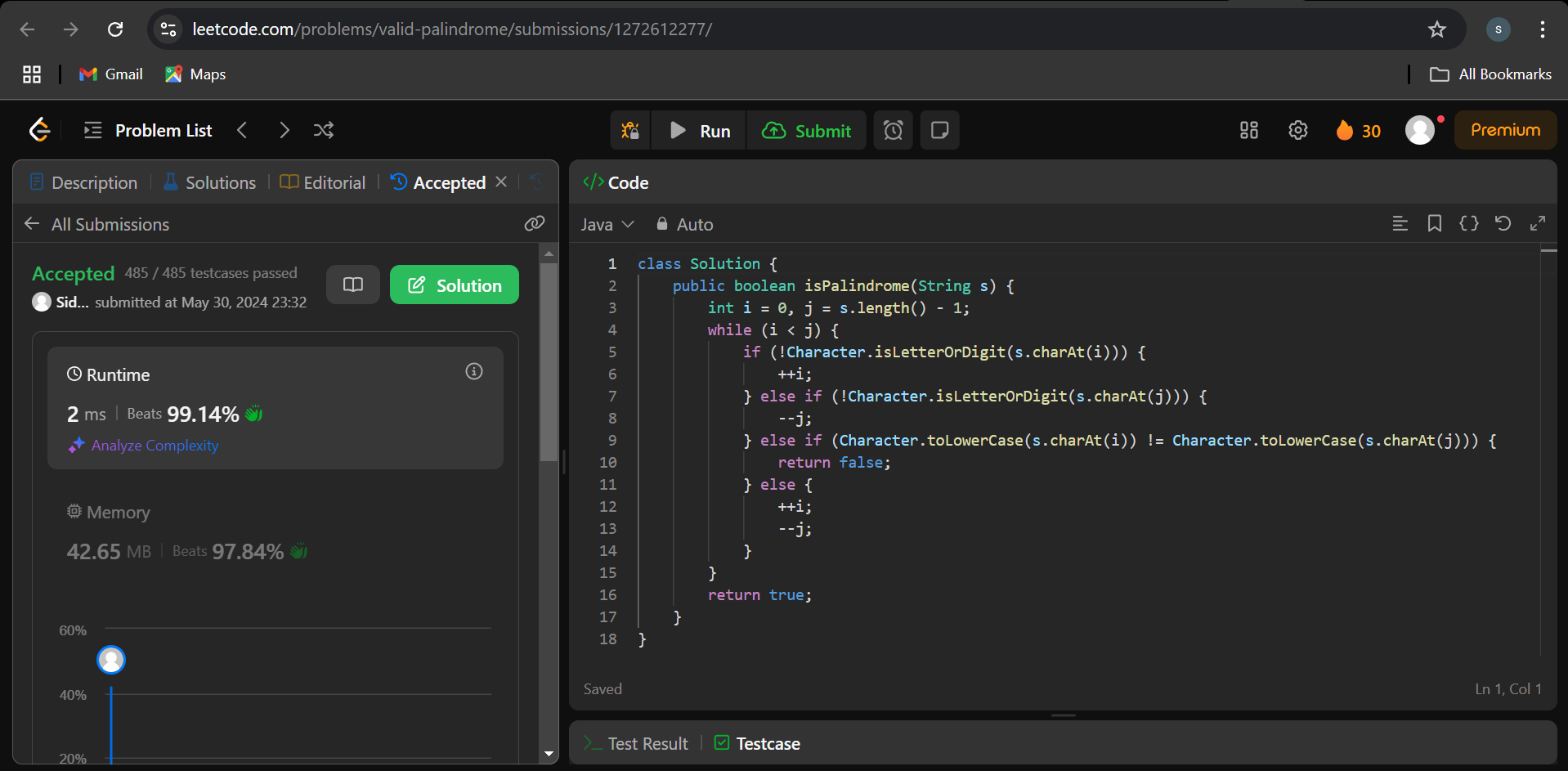
}

return true;

}

}

**Output:**



1. **Jump Game 2:**

**Problem Link :** <https://leetcode.com/problems/jump-game-ii/description/>

**Code:**

class Solution {

public int jump(int[] nums) {

// Max. no. of jumps given -> iska mtlb chotta jump bhi maar skta

int n = nums.length;

int dp[] = new int[n];

Arrays.fill(dp, -1);

dp[n-1] = 0;

for(int i=n-2; i>=0; i--) {

int steps = nums[i];

int ans = Integer.MAX\_VALUE;

for(int j=i+1; j<=i+steps && j<n; j++) {

if(dp[j] != -1) {

ans = Math.min(ans, dp[j]+1);

}

}

if(ans != Integer.MAX\_VALUE) {

dp[i] = ans;

}

}

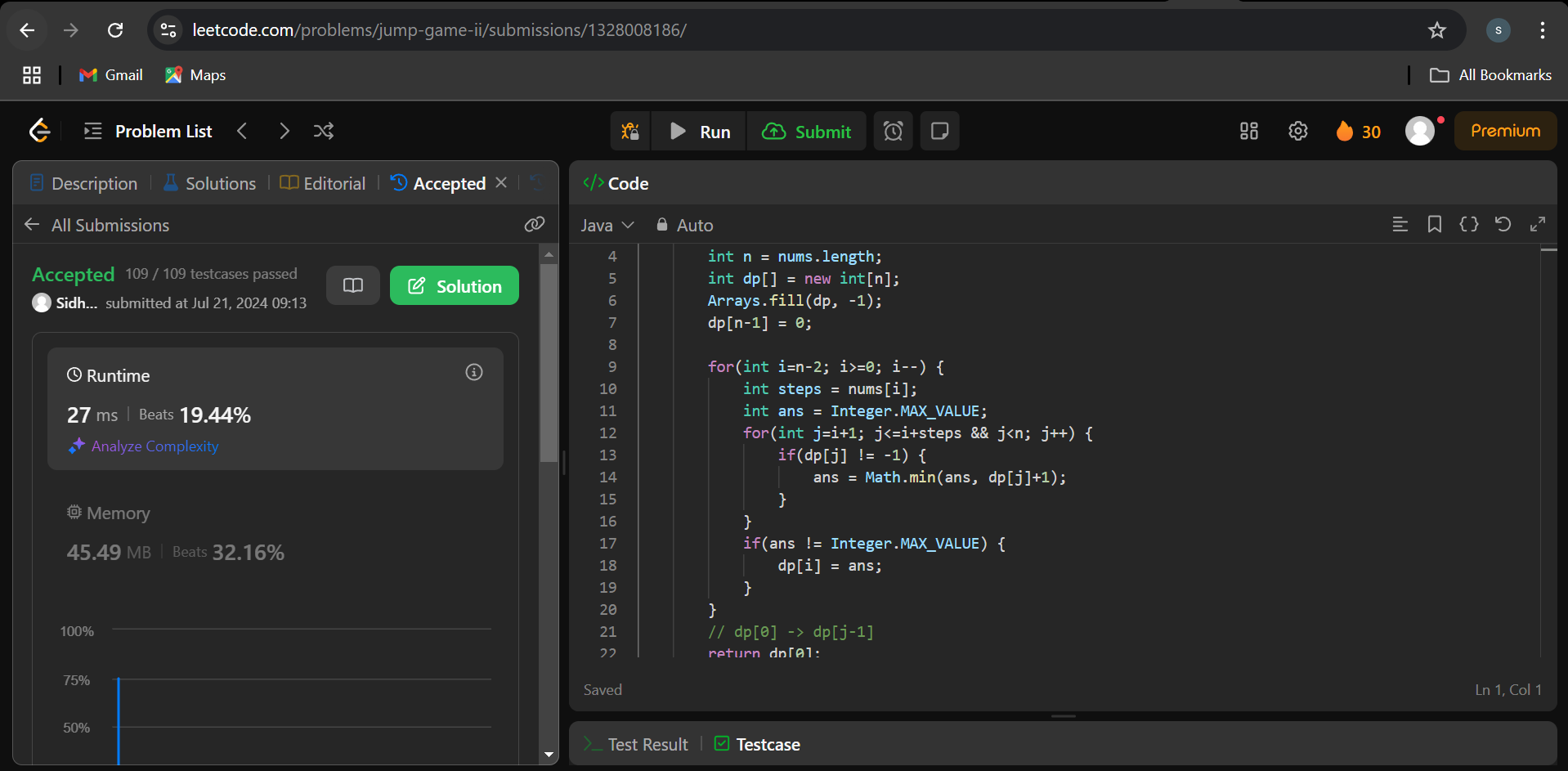
// dp[0] -> dp[j-1]

return dp[0];

}

}

**Output:**



1. **3Sum:**

**Problem Link :** <https://leetcode.com/problems/3sum/description/>

**Code:**

class Solution {

public List<List<Integer>> threeSum(int[] nums) {

List<List<Integer>> result = new ArrayList<>();

Arrays.sort(nums);

for(int i=0; i<nums.length-2; i++) {

// skip duplicates

if(i>0 && nums[i] == nums[i-1]) {

continue;

}

int left = i+1;

int right = nums.length-1;

while(left < right) {

int sum = nums[i] + nums[left] + nums[right];

if(sum == 0) {

result.add(Arrays.asList(nums[i], nums[left], nums[right]));

while(left < right && nums[left] == nums[left+1]) {

left++;

}

while(left < right && nums[right] == nums[right-1]) {

right--;

}

left++;

right--;

}

else if(sum < 0) {

left++;

}

else {

right--;

}

}

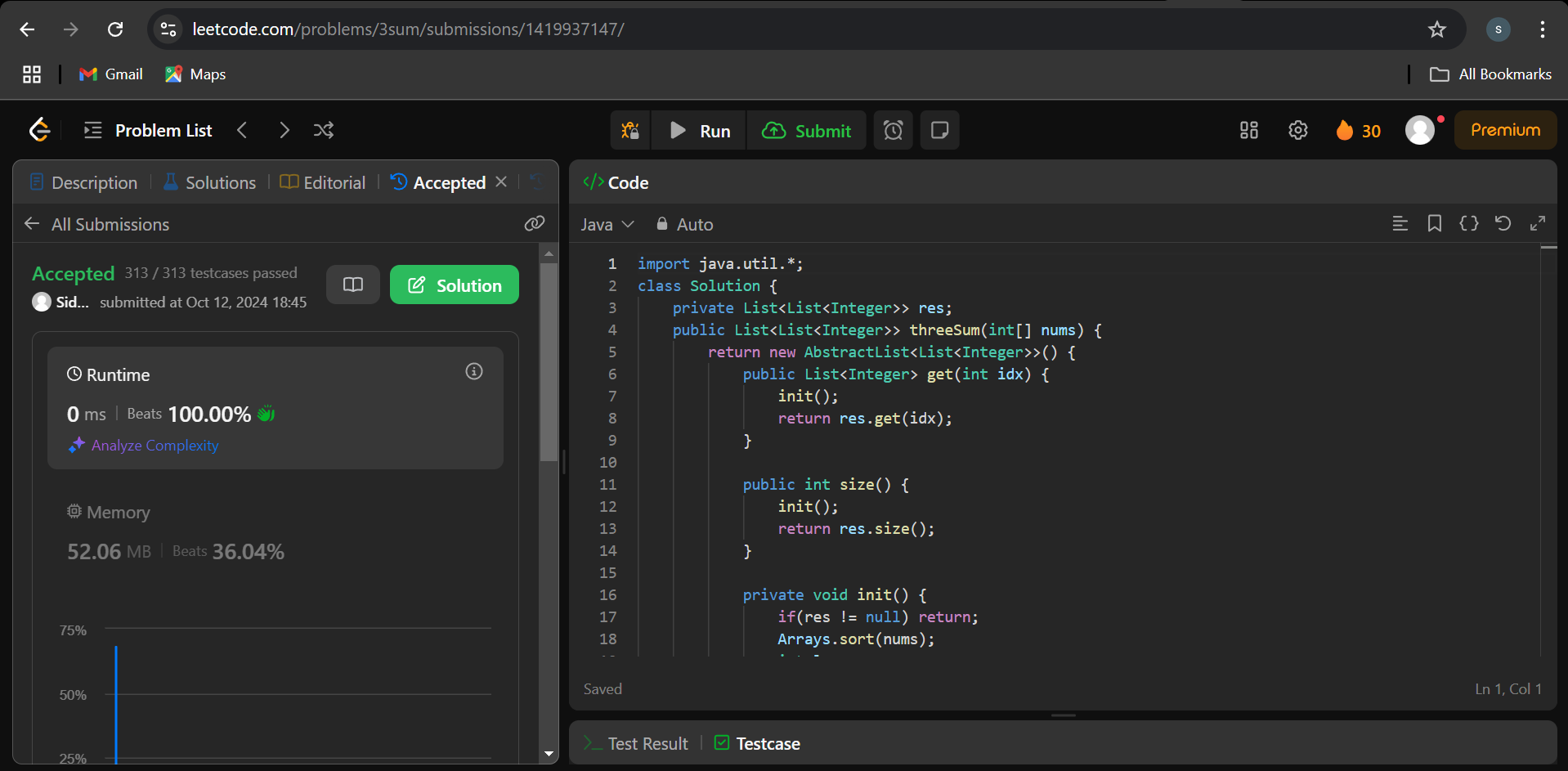
}

return result;

}

}

**Output :**



1. **Set Matrix Zeroes:**

**Problem Link :** <https://leetcode.com/problems/set-matrix-zeroes/description/>

**Code:**

class Solution {

public void setZeroes(int[][] matrix) {

boolean fr = false, fc = false;

for(int i=0; i<matrix.length; i++) {

for(int j=0; j<matrix[0].length; j++) {

if(matrix[i][j] == 0) {

if(i == 0) fr = true;

if(j == 0) fc = true;

matrix[0][j] = 0;

matrix[i][0] = 0;

}

}

}

for(int i=1; i<matrix.length; i++) {

for(int j=1; j<matrix[0].length; j++) {

if(matrix[i][0] == 0 || matrix[0][j] == 0) {

matrix[i][j] = 0;

}

}

}

if(fr) {

for(int j=0; j<matrix[0].length; j++) {

matrix[0][j] = 0;

}

}

if(fc) {

for(int i=0; i<matrix.length; i++) {

matrix[i][0] = 0;

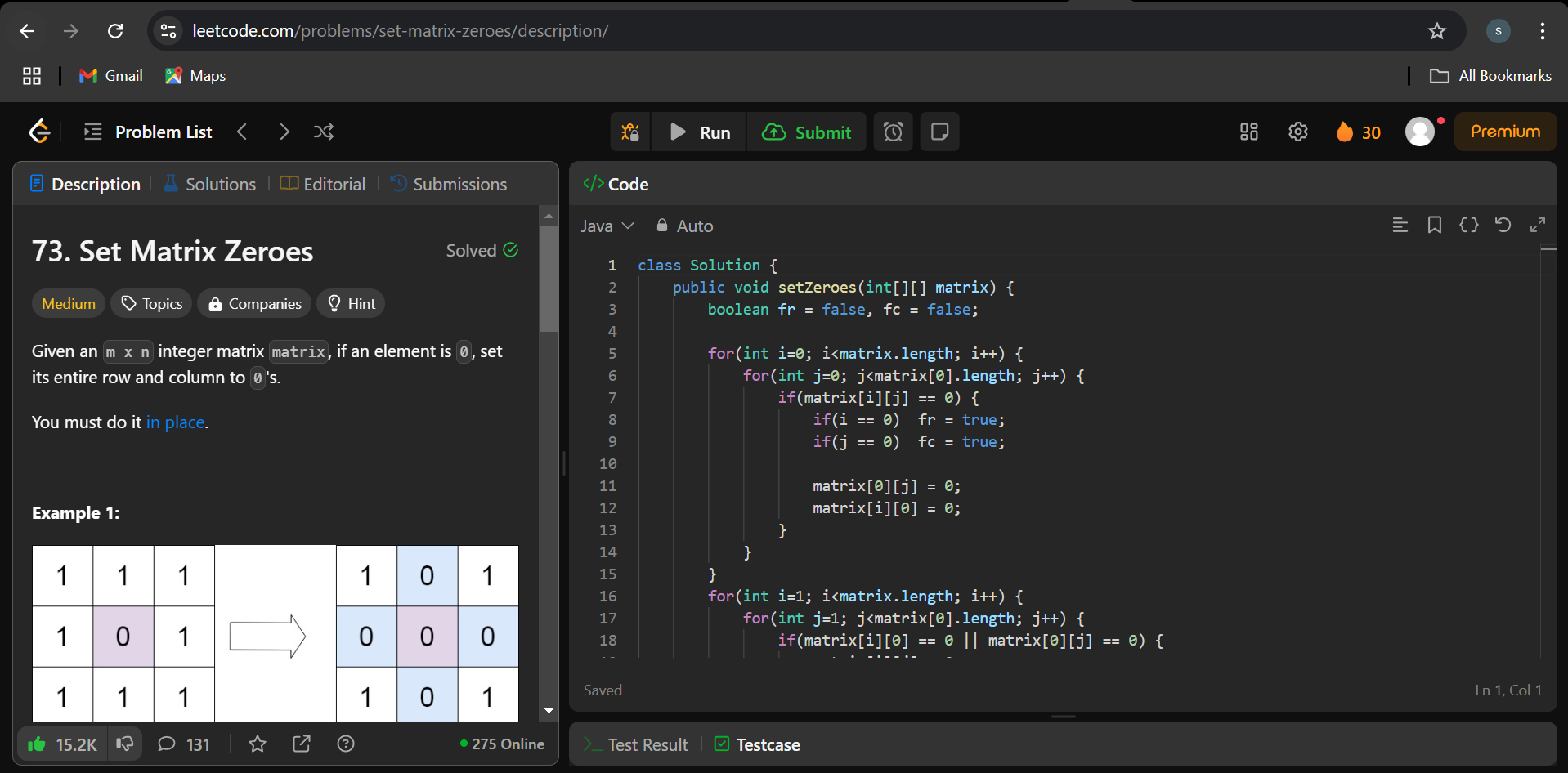
}

}

}

}

**Output:**



1. **Longest substring without repeating characters:**

**Problem Link :** <https://leetcode.com/problems/longest-substring-without-repeating-characters/description/>

**Code:**

class Solution {

public int lengthOfLongestSubstring(String s) {

int n = s.length();

int maxLength = 0;

Set<Character> charSet = new HashSet<>();

int left = 0;

for (int right = 0; right < n; right++) {

if (!charSet.contains(s.charAt(right))) {

charSet.add(s.charAt(right));

maxLength = Math.max(maxLength, right - left + 1);

} else {

while (charSet.contains(s.charAt(right))) {

charSet.remove(s.charAt(left));

left++;

}

charSet.add(s.charAt(right));

}

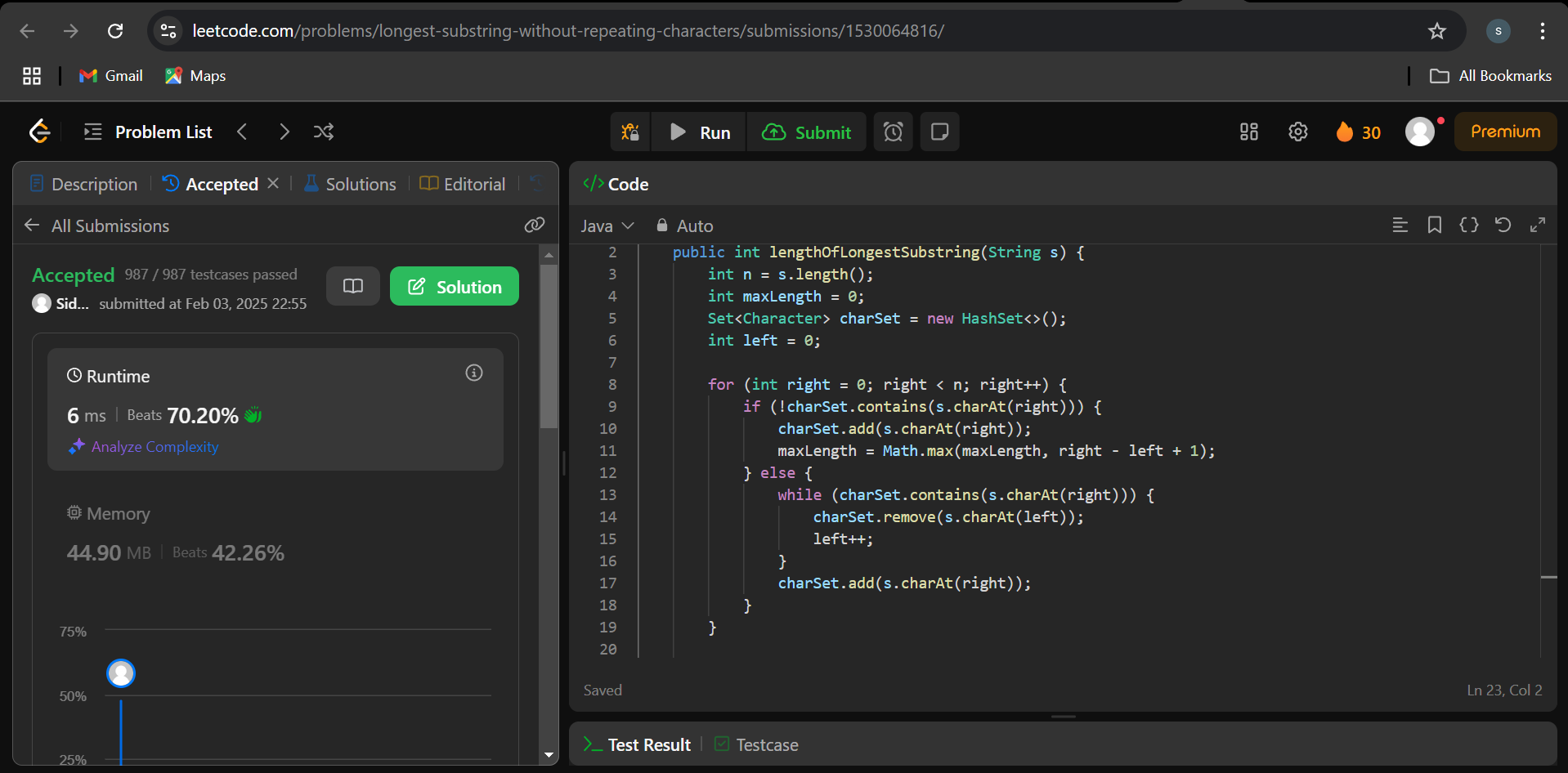
}

return maxLength;

}

}

**Output:**



1. **Finding duplicate number:**

**Problem Link :** <https://leetcode.com/problems/find-the-duplicate-number/description/>

**Code:**

class Solution {

public int findDuplicate(int[] nums) {

int not\_unique = -1;

int n = nums.length;

boolean[] isUnique = new boolean[n];

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

if(isUnique[nums[i]]) return nums[i];

else isUnique[nums[i]] = true;

}

return not\_unique;

}

}

**Output:**

