TWO SUM PROBLEM:

CODE: class Solution {

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

        Map<Integer, Integer> numMap = new HashMap<>();

        int n = nums.length;

        for (int i = 0; i < n; i++) {

            int complement = target - nums[i];

            if (numMap.containsKey(complement)) {

                return new int[]{numMap.get(complement), i};

            }

            numMap.put(nums[i], i);

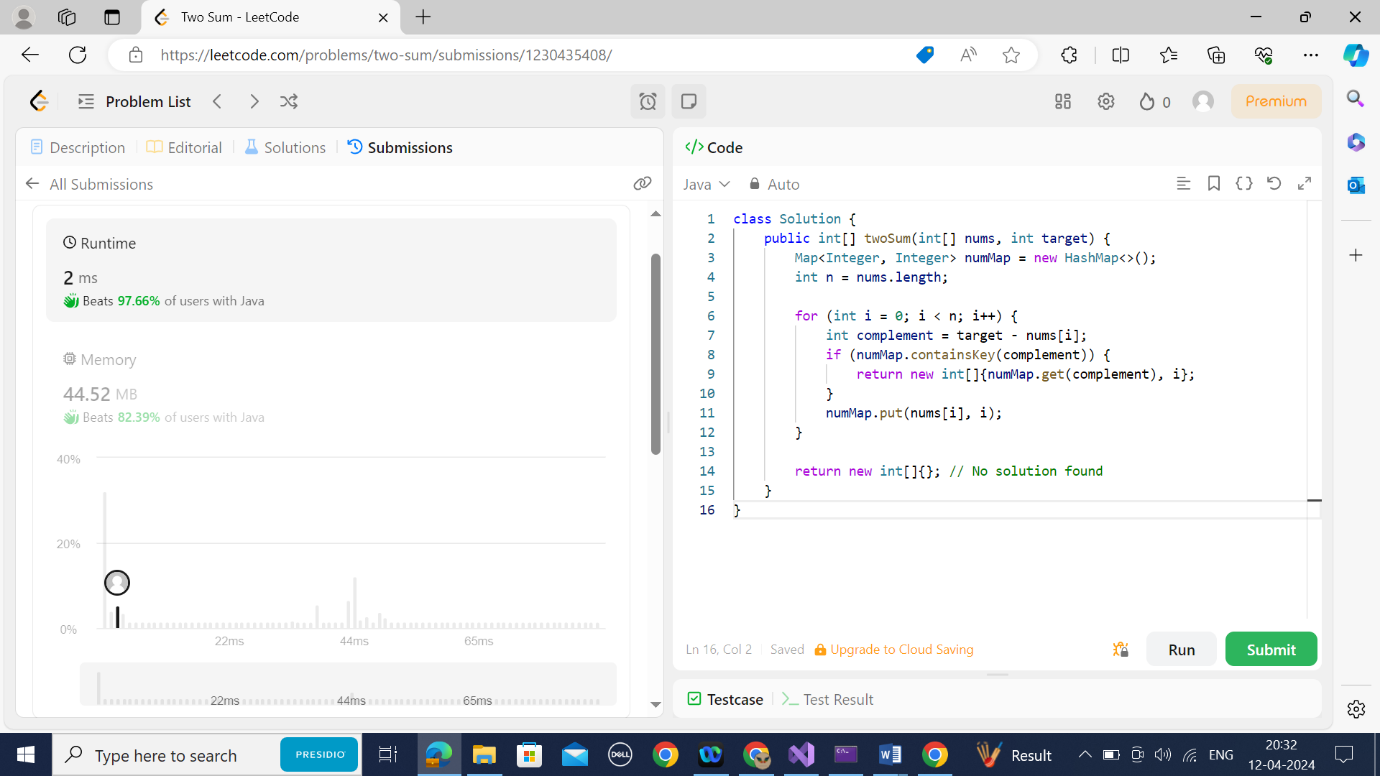
        }

        return new int[]{}; // No solution found

    }

}

OUTPUT:



VALID PARANTHESIS:

CODE:

import java.util.Stack;

class Solution {

    public boolean isValid(String s) {

        Stack<Character> stack = new Stack<>();

        for (char curr : s.toCharArray()) {

            if (curr == '(' || curr == '[' || curr == '{') {

                stack.push(curr);

            } else if (!stack.isEmpty() && isMatchingPair(stack.peek(), curr)) {

                stack.pop();

            } else {

                return false;

            }

        }

        return stack.isEmpty();

    }

    private boolean isMatchingPair(char open, char close) {

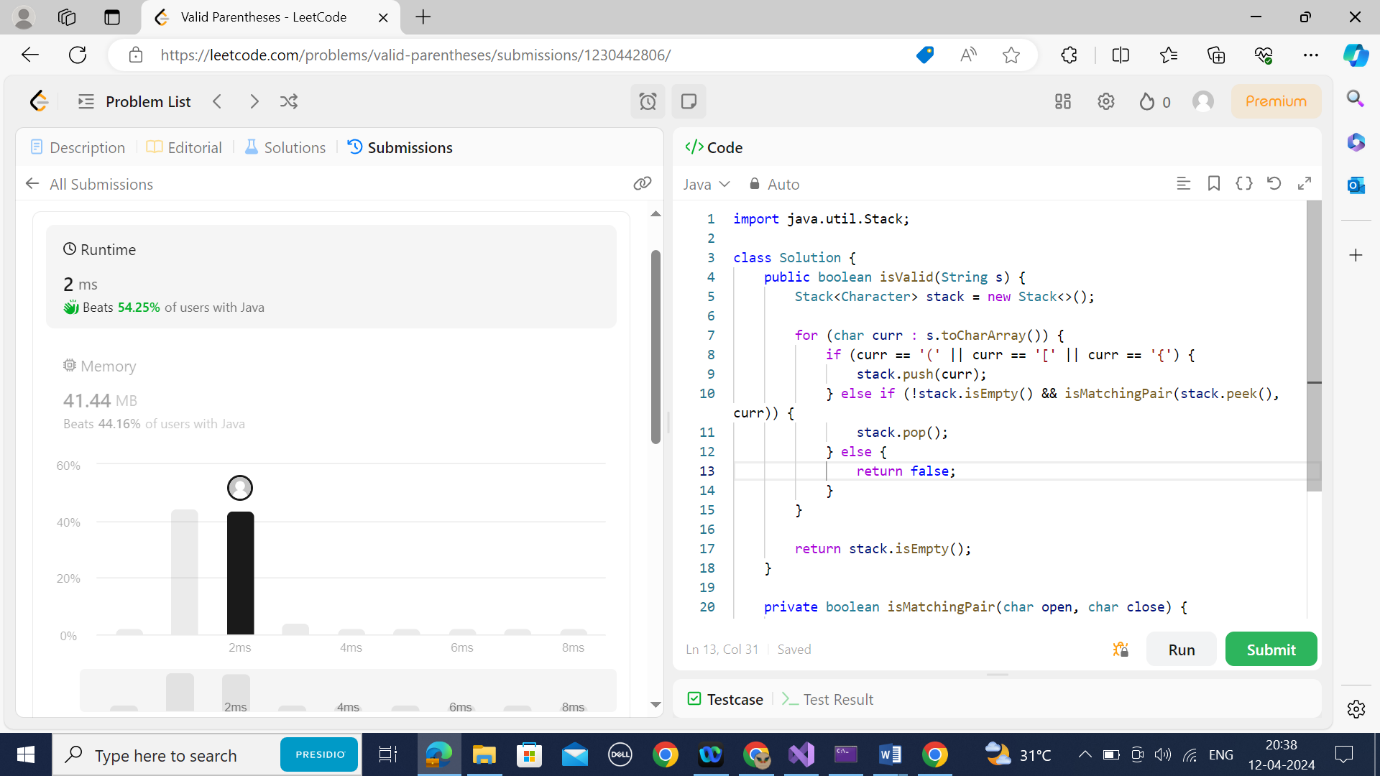
        return (open == '(' && close == ')') ||

               (open == '[' && close == ']') ||

               (open == '{' && close == '}');

    }

}

OUTPUT:

PALINDROME NUMBER:

CODE:

class Solution {

    public boolean isPalindrome(int x) {

        String num= String.valueOf(x);

        int left=0;

        int right= num.length()-1;

        while(left<right)

        {

            if(num.charAt(left)!=num.charAt(right))

            {

                return false;

            }

            left++;

            right--;

        }

        return true;

    }

}

OUTPUT:

