



ASSIGNMENT

Student Name: Nitil Jakhar

UID: 22BCS17300

Branch: CSE

Section/Group: NTPP_IOT-602/A

Semester: 6th

Date of Performance: 05/04/2025

Subject: AP 2

1. Aim:

- a. Two Sum
- b. Longest Substring Without Repeating Characters
- c. Palindrome Number

2. Objective:

- a. Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.
- b. Given a string s, find the length of the longest substring that does not contain any repeating characters.
- c. Given an integer x, return true if x is a palindrome, and false otherwise.

3. Code:

a. Two Sum

```
from typing import List
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        num_map = {} # Dictionary to store value:index
        for i, num in enumerate(nums):
            diff = target - num
            if diff in num_map:
                return [num_map[diff], i]
            num_map[num] = i
        return [] # Return empty if no solution found (though the problem
guarantees one)
```

Example usage:

```
sol = Solution()
```

```
print(sol.twoSum([2, 7, 11, 15], 9)) # Output: [0, 1]
```

```
print(sol.twoSum([3, 2, 4], 6)) # Output: [1, 2]
```

b. Longest Substring Without Repeating Characters

```
class Solution:
```

```
    def lengthOfLongestSubstring(self, s: str) -> int:
        char_set = set()
```

```
left = 0
max_length = 0
for right in range(len(s)):
    while s[right] in char_set:
        char_set.remove(s[left])
        left += 1
    char_set.add(s[right])
    max_length = max(max_length, right - left + 1)
return max_length

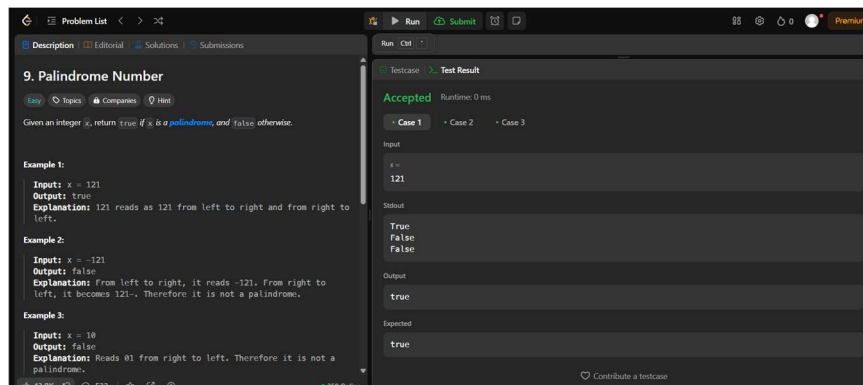
# Example usage:
sol = Solution()
print(sol.lengthOfLongestSubstring("abcabcbb")) # Output: 3
print(sol.lengthOfLongestSubstring("bbbbbb")) # Output: 1
print(sol.lengthOfLongestSubstring("pwwkew")) # Output: 3
```

c. Palindrome Number

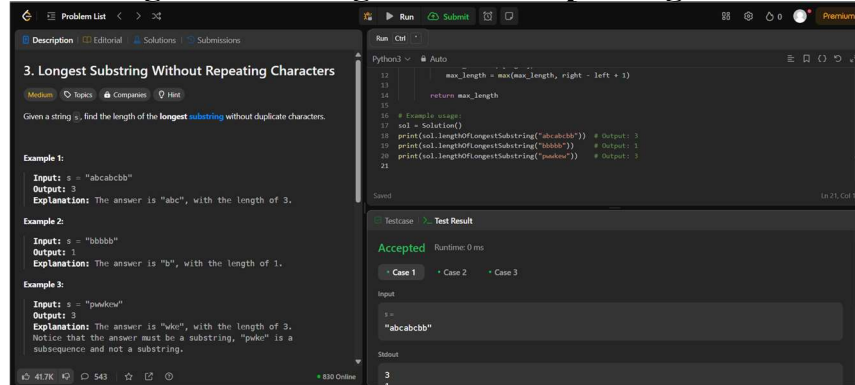
```
class Solution:
    def isPalindrome(self, x: int) -> bool:
        if x < 0:
            return False # Negative numbers are not palindromes
        return str(x) == str(x)[::-1] # Check if string representation is equal to its
reverse
# Example usage:
sol = Solution()
print(sol.isPalindrome(121)) # Output: True
print(sol.isPalindrome(-121)) # Output: False
print(sol.isPalindrome(10)) # Output: False
```

4. Output:

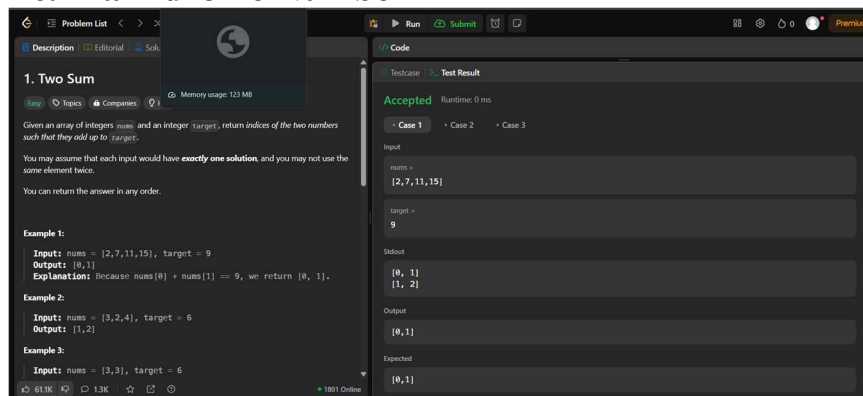
a. Two Sum



b. Longest Substring Without Repeating Characters



c. Palindrome Number



5. Learning Outcome

- 1) Understand how to implement efficient algorithms using hash maps for constant-time lookups.
- 2) Gain the ability to use sliding window techniques to solve substring-related problems.
- 3) Learn to manipulate and compare strings for palindrome validation.
- 4) Enhance problem-solving skills through real-world coding challenges.
- 5) Develop proficiency in writing clean, readable, and testable Python code.