PYTHON PROBLEM

Longest Substring Without Repeating Characters

Problem Statement

Given a string s, find the length of the longest substring without repeating characters.

Constraints

- 0≤s.length≤5×1040 \leq s.length \leq 5 \times 10^40≤s.length≤5×104
- s consists of English letters, digits, symbols, and spaces.

Solution

The solution uses the sliding window technique with a hash map to track characters and their indices. This approach allows us to efficiently find the longest substring without repeating characters.

Code Implementation

```
class Solution:
    def lengthOfLongestSubstring(self, s: str) -> int:
        char_index_map = {}
        longest = 0
        start = 0

        # Iterate over the string with the 'end' pointer.
        for end, char in enumerate(s):
            if char in char_index_map and char_index_map[char] >= start:
                start = char_index_map[char] + 1
            # Update the character's index in the dictionary to the
current position.
            char_index_map[char] = end
            longest = max(longest, end - start + 1)


        return longest
```

SQL PROBLEM

SQL Query to Find Numbers That Appear at Least Three Times Consecutively

Problem Statement

Given a table Logs with columns id and num, where id is an auto increment primary key, find all numbers that appear at least three times consecutively. The result table should list each such number in any order.

Constraints

- id is the primary key and is auto-incremented.
- nums can be any valid value and is not guaranteed to be unique.

SOLUTION

The provided solution uses a self-join approach to find numbers that appear consecutively at least three times. This method involves joining the `Logs` table with itself multiple times to compare consecutive rows.

SQL Query

```sql
SELECT DISTINCT l1.num AS ConsecutiveNums
FROM Logs l1, Logs l2, Logs l3
WHERE
    l1.id = l2.id-1 AND
    l2.id = l3.id-1 AND
    l1.num = l2.num AND
    l2.num = l3.num;
```