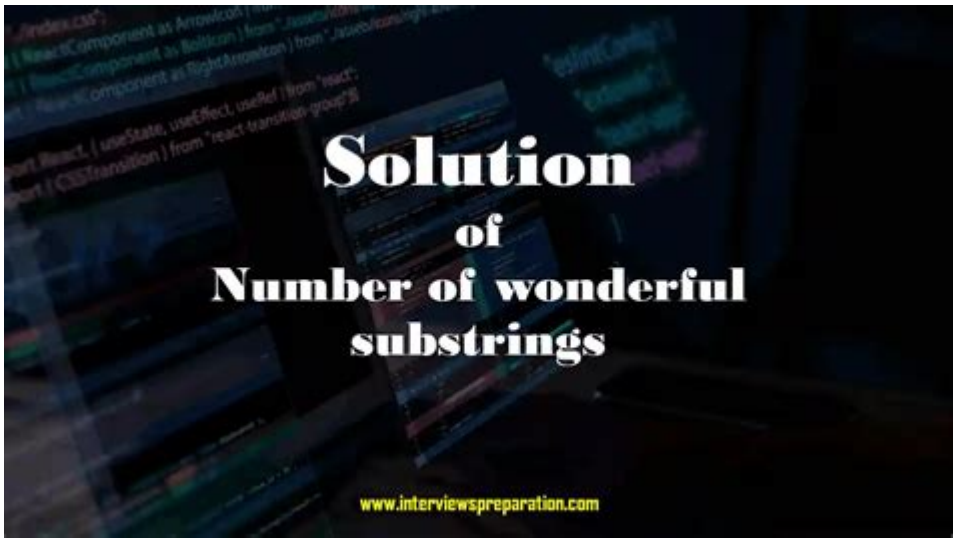
Extraordinary Substrings Hackerrank Solution



Extraordinary substrings hackerrank solution is a popular problem featured on HackerRank that challenges programmers to identify specific substrings within a given string. The task not only tests coding skills but also enhances one's ability to manipulate strings and understand various algorithms. This article delves deep into the problem, providing a comprehensive overview of its requirements, examples, and a viable solution approach, making it an essential read for anyone looking to master this challenge.

Understanding the Problem Statement

The extraordinary substrings problem asks you to find substrings within a given string that meet certain criteria. Specifically, the substrings must be classified based on their character composition. The typical requirements involve identifying substrings that contain unique characters or substrings that can be rearranged to form a palindrome.

When approaching this problem, it is crucial to clarify the requirements as they may vary slightly based on the specific challenge presented. Nevertheless, the core idea remains the same: efficiently compute valid substrings from the input string.

Key Concepts and Definitions

Before diving into the solution, it is essential to understand some key concepts:

Substring

A substring is any contiguous sequence of characters within a string. For example, in the string "abcd", the substrings include "a", "ab", "abc", "abcd", "b", "bc", "bcd", "c", "cd", and "d".

Unique Characters

A substring is said to have unique characters if no character appears more than once in that substring. For instance, "abc" has unique characters, whereas "aab" does not.

Palindrome

A palindrome is a string that reads the same forward and backward. For example, "racecar" is a palindrome, while "hello" is not.

Example Problem

To illustrate the problem, let's consider the following example:

- Input: "abca"
- Output: 4

In this example, the extraordinary substrings are "a", "b", "c", and "abca". The goal is to compute the number of such valid substrings.

Breaking Down the Solution

Finding extraordinary substrings can be achieved through various approaches, ranging from brute-force methods to more sophisticated algorithms. Here, we will explore a few methods before proposing an optimal solution.

Brute-Force Approach

The brute-force method involves generating all possible substrings and checking each one against the extraordinary condition. While this method is straightforward, it is inefficient for larger strings due to its O(n^3) time complexity.

Steps:
1. Generate all possible substrings.
2. Check each substring for unique characters or palindrome properties.

3. Count valid substrings.

Pros:
- Simple to implement.
- Easy to understand.

Cons:
- Inefficient for long strings.
- High time complexity.

Sliding Window Technique

The sliding window technique can significantly improve performance by maintaining a dynamic range of characters as we traverse the string. This method allows us to check for unique characters without generating all substrings explicitly.

Steps:
1. Use two pointers to define the current window of characters.
2. Expand the window by moving the right pointer and adding characters.
3. Contract the window from the left when a duplicate character is encountered.
4. Count the valid substrings formed in each window.

Pros:
- More efficient than brute-force.
- Optimal for many string problems.

Cons:
- More complex to implement than brute-force.
- Requires careful management of window boundaries.

Optimal Solution Implementation

Based on the understanding of the problem and the analysis of different methods, we can now present an optimal solution using the sliding window approach. This method efficiently counts extraordinary substrings while maintaining a linear time complexity.

Here is a Python implementation of the solution:

```python
def extraordinary_substrings(s):
n = len(s)
count = 0

Iterate through each character in the string
```

```
for i in range(n):
char_count = {}
unique_count = 0

Start a new substring from index i
for j in range(i, n):
char = s[j]

If the character is new, add it to the dictionary
if char not in char_count:
char_count[char] = 0
unique_count += 1

char_count[char] += 1

Check if the current substring is extraordinary
if unique_count == len(char_count) and all(v == 1 for v in
char_count.values()):
count += 1

return count

Example usage
s = "abca"
print(extraordinary_substrings(s)) Output: 4
```

Explanation of the Code:
- We iterate through each character in the string, treating it as the starting point of a new substring.
- A dictionary (`char_count`) keeps track of character frequencies to check for unique characters.
- For each possible ending index of the substring, we update the dictionary and check if all characters are unique.
- If they are, we increment our count of extraordinary substrings.

Conclusion

The extraordinary substrings problem on HackerRank is an excellent way to enhance your string manipulation and algorithmic skills. By exploring different approaches—from brute-force methods to optimized sliding window techniques—you can gain a deeper understanding of how to tackle similar programming challenges efficiently.

Remember that while the brute-force method may be easier to implement, the sliding window technique provides a more scalable solution for larger inputs. As you practice, aim to refine your skills in recognizing patterns and applying the most suitable algorithms to solve problems effectively. Happy coding!

Frequently Asked Questions

What is the 'Extraordinary Substrings' problem on HackerRank?

The 'Extraordinary Substrings' problem involves finding substrings of a given string that meet specific criteria based on the frequency of characters. The goal is to count how many substrings can be classified as extraordinary.

What are the criteria for a substring to be considered extraordinary?

A substring is considered extraordinary if all characters in it have the same frequency of occurrence. For example, 'aabb' is extraordinary because both 'a' and 'b' appear twice.

How can I approach solving the Extraordinary Substrings problem?

To solve the problem, you can iterate through all possible substrings of the input string, count the frequency of each character in those substrings, and check if all characters have the same frequency.

Are there any efficient algorithms to solve the Extraordinary Substrings problem?

Yes, using a sliding window technique along with character frequency counting can reduce the time complexity. You can maintain a count of characters in the current window and adjust as you expand or contract the window.

What are common pitfalls to avoid when solving this problem?

Common pitfalls include not considering edge cases like single-character substrings and forgetting to reset character counts properly when moving to the next substring.

Where can I find sample test cases for the Extraordinary Substrings problem?

Sample test cases can typically be found in the HackerRank problem statement, and you can also create your own test cases by manually checking substrings and their character frequencies.

Extraordinary Substrings Hackerrank Solution

Extraordinary | HiNative
Extraordinary Q&A Extraordinary 188 ...

ordinary *extraordinary*
ordinary extraor... 4 Hinative "" "" ...

exceptional **extraordinary** **...**
excepti... extraor... 2 Hinative "" "" ...

"extraordinary" "remarkable" | HiNative
extraordinary means very unusual. example: he is an extraordinary guy. remarkable means worthy of attention. example: the incident is remarkably ...

"extraordinary" "remarkable" | HiNative
extraordinary extraordinary means very unusual. example: he is an extraordinary guy. remarkable means worthy of attention. example: the ...

Extraordinary | HiNative
Extraordinary Q&A Extraordinary 188

ordinary *extraordinary* ... - *HiNative*
ordinary extraor... 4 Hinative "" "" ...

exceptional *extraordinary*
excepti... extraor... 2 Hinative "" "" ...

"extraordinary" "remarkable" | HiNative
extraordinary means very unusual. example: he is an extraordinary guy. remarkable means worthy of attention. example: the incident is remarkably different. however they both have ...

"*extraordinary*" "*remarkable*" | *HiNative*
extraordinary extraordinary means very unusual. example: he is an extraordinary guy. remarkable means worthy of attention. example: the incident is remarkably different. however ...

extraordinary *No ordinary* ... - *HiNative*
Extraordinary 1 Not ordinary ...

extraordinary *a* *an* -
extraordinary a anextraordinary 1. It is very extraordinary that I should meet him here.

"unusual" 和 **"extraordinary"** 和有什么不一样？ **| HiNative**
unusual 和 extraordinary 有什么区别？如果难以说明的话，请教我一下例句。 Danielg_ 2017年7月9日 英语 (美国) 西班牙语 (墨西哥)

"*phenomenal*" 和 "*extraordinary*" 和 "*remarkable*" 和有什么不一样？
phenomenalThey technically mean the same but I feel that they are generally used in increasing intensity. That's remarkable! - * That's pretty cool! (It's notably good) That's extraordinary! - ** ...

“EGM”指什么？_百度知道
Jun 11, 2024 · 英语缩写"EGM"在国际商务语境中代表"Extraordinary General Meeting"，中文直译为“临时股东大会”。这个术语在商业领域中颇为常见，其拼音为“lín shí gǔ dōng dà huì”，具 ...

Unlock the secrets of the Extraordinary Substrings HackerRank solution! Dive into our comprehensive guide and solve challenges effortlessly. Learn more now!

Back to Home