

Grokking the Coding Interview: Patterns for Coding Questions

5% completed

- Introduction
- Maximum Sum Subarray of Size K (easy)
- Smallest Subarray with a given sum (easy)
- Longest Substring with K Distinct Characters (medium)
- Fruits into Baskets (medium)
- No-repeat Substring (hard)
- Longest Substring with Same Letters after Replacement (hard)
- Longest Subarray with Ones after Replacement (hard)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2**
- Problem Challenge 3
- Solution Review: Problem Challenge 3
- Problem Challenge 4
- Solution Review: Problem Challenge 4

Pattern: Two Pointers

Pattern: Fast & Slow pointers

Pattern: Merge Intervals

Pattern: Cyclic Sort

Pattern: In-place Reversal of a LinkedList

Pattern: Tree Breadth First Search

Pattern: Tree Depth First Search

Pattern: Two Heaps

Pattern: Subsets

Pattern: Modified Binary Search

Pattern: Bitwise XOR

Pattern: Top 'K' Elements

Solution Review: Problem Challenge 2

We'll cover the following

- String Anagrams (hard)
- Solution
 - Code
 - Time Complexity
 - Space Complexity

String Anagrams (hard)

Given a string and a pattern, find **all anagrams of the pattern in the given string**.

Anagram is actually a **Permutation** of a string. For example, "abc" has the following six anagrams:

1. abc
2. acb
3. bac
4. bca
5. cab
6. cba

Write a function to return a list of starting indices of the anagrams of the pattern in the given string.

Example 1:

```
Input: String="ppqp", Pattern="pq"
Output: [1, 2]
Explanation: The two anagrams of the pattern in the given string are "pq" and "qp".
```

Example 2:

```
Input: String="abbcabc", Pattern="abc"
Output: [2, 3, 4]
Explanation: The three anagrams of the pattern in the given string are "bca", "cab", and "abc".
```

Solution

This problem follows the **Sliding Window** pattern and is very similar to [Permutation in a String](#). In this problem, we need to find every occurrence of any permutation of the pattern in the string. We will use a list to store the starting indices of the anagrams of the pattern in the string.

Code #

Here is what our algorithm will look like, only the highlighted lines have changed from [Permutation in a String](#):

JavaPython3C++JS

```
1 function find_string_anagrams(str, pattern) {
2   let windowStart = 0,
3     matched = 0,
4     charFrequency = {};
5
6   for (i = 0; i < pattern.length; i++) {
7     const chr = pattern[i];
8     if (!(chr in charFrequency)) {
9       charFrequency[chr] = 0;
10    }
11    charFrequency[chr] += 1;
12  }
13
14  const resultIndices = [];
15  // our goal is to match all the characters from the 'charFrequency' with the current window
16  // try to extend the range [windowStart, windowEnd]
17  for (windowEnd = 0; windowEnd < str.length; windowEnd++) {
18    const rightChar = str[windowEnd];
19    if (rightChar in charFrequency) {
20      // decrement the frequency of matched character
21      charFrequency[rightChar] -= 1;
22      if (charFrequency[rightChar] === 0) {
23        matched += 1;
24      }
25    }
26
27    if (matched === Object.keys(charFrequency).length) { // have we found an anagram?
28      resultIndices.push(windowStart);
29    }
30
31    // move the window forward
32    str[windowStart] += 1;
33    windowStart++;
34  }
35
36  return resultIndices;
37}
```

RUNSAVERESET

Time Complexity

The time complexity of the above algorithm will be $O(N + M)$ where 'N' and 'M' are the number of

Edpresso

Refer a Friend

Create

Pattern: K-way merge

Pattern : 0/1 Knapsack (Dynamic Programming)

Pattern: Topological Sort (Graph)

Miscellaneous

Conclusions

Where to Go from Here

Mark Course as Completed

The time complexity of the above algorithm will be $O((N + M) * M)$ where N and M are the number of characters in the input string and the pattern respectively.

Space Complexity

The space complexity of the algorithm is $O(M)$ since in the worst case, the whole pattern can have distinct characters which will go into the **HashMap**. In the worst case, we also need $O(N)$ space for the result list, this will happen when the pattern has only one character and the string contains only that character.

← Back

MARK AS COMPLETED

Next →

Problem Challenge 2

Problem Challenge 3

Report an Issue