





- Back To Course Home

## Grokking the Coding Interview: Patterns for Coding Questions

5% completed

Search Course

(/collection/page/56686391014

Fruits into Baskets (medium) (/collection/page/56686391014

No-repeat Substring (hard) (/collection/page/56686391014

Longest Substring with Same Letters after Replacement (hard)

(/collection/page/56686391014

Longest Subarray with Ones after Replacement (hard) (/collection/page/56686391014

Problem Challenge 1 (/collection/page/56686391014

Solution Review: Problem Challenge 1

## 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





← Back To Course Home

# Grokking the Coding Interview: Patterns for Coding Questions

5% completed

C

Q Search Course

(/collection/page/56686391014

Fruits into Baskets (medium) (/collection/page/56686391014

No-repeat Substring (hard) (/collection/page/56686391014

Longest Substring with Same Letters after Replacement (hard)

(/collection/page/56686391014

Longest Subarray with Ones after Replacement (hard) (/collection/page/56686391014

Problem Challenge 1 (/collection/page/56686391014

Solution Review: Problem Challenge 1

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 "q

р".

### Example 2:

Input: String="abbcabc", Pattern="abc"

Output: [2, 3, 4]

Explanation: The three anagrams of the pattern in the given string are "bca", "ca

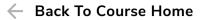
b", and "abc".

#### Solution #

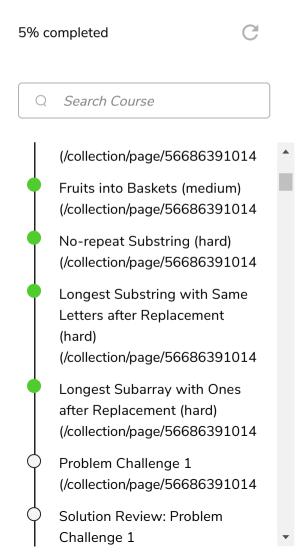
This problem follows the **Sliding Window** pattern and is very similar to Permutation in a String

(https://www.educative.io/collection/page/5668639101419520/5671464854355968/54019 34796161024/). 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 #



# Grokking the Coding Interview: Patterns for Coding Questions



Here is what our algorithm will look like, only the highlighted lines have changed from Permutation in a String

(https://www.educative.io/collection/page/5668639101419520/5671464854355968/54019 34796161024/):

```
Python3
👙 Java
                                                                                                                                                           C++
                                                                                                                                                                                                                                      ıs JS
  13
                                                            char rightChar = str.charAt(windowEnd);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                \downarrow
  14
                                                             // decrement the frequency of the matched cha
                                                            if (charFrequencyMap.containsKey(rightChar))
  15
  16
                                                                        charFrequencyMap.put(rightChar, charFrequer
                                                                       if (charFrequencyMap.get(rightChar) == 0)
  17
                                                                                   matched++;
  18
  19
  20
                                                            if (matched == charFrequencyMap.size()) // ha
  21
  22
                                                                        resultIndices.add(windowStart);
  23
  24
                                                            if (windowEnd >= pattern.length() - 1) { // s
                                                                        char leftChar = str.charAt(windowStart++);
  25
                                                                       if (charFrequencyMap.containsKey(leftChar))
  26
  27
                                                                                   if (charFrequencyMap.get(leftChar) == 0)
                                                                                               matched--; // before putting the charac
  28
                                                                                   // put the character back
  29
                                                                                   charFrequencyMap.put(leftChar, charFreque
  30
  31
  32
  33
  34
  35
                                                  return resultIndices:
  36
                                      }
  37
                                      public static void main(String[] args) {
  38
                                                 System.out.println(StringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagram
  39
  40
                                                 System.out.println(StringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagrams.findStringAnagram
  41
  42
  43
```

← Back To Course Home

# Grokking the Coding Interview: Patterns for Coding Questions

5% completed

Q Search Course

(/collection/page/56686391014

Fruits into Baskets (medium) (/collection/page/56686391014

No-repeat Substring (hard) (/collection/page/56686391014

Longest Substring with Same Letters after Replacement (hard)

(/collection/page/56686391014

Longest Subarray with Ones after Replacement (hard) (/collection/page/56686391014

Problem Challenge 1 (/collection/page/56686391014

Solution Review: Problem Challenge 1

Time Complexity #

Run

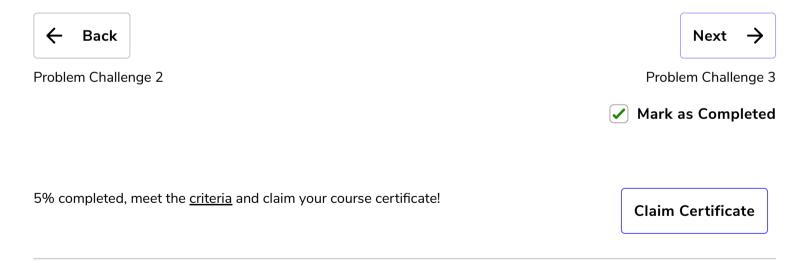
The time complexity of the above algorithm will be O(N+M) where 'N' and 'M' are the number of characters in the input string and the pattern respectively.

Save

Reset

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.



Report an Issue

? Ask a Question

 $(https://discuss.educative.io/tag/solution-review-problem-challenge-2\_pattern-sliding-window\_grokking-the-coding-interview-patterns-for-coding-questions)\\$ 





#### ← Back To Course Home

## Grokking the Coding Interview: Patterns for Coding Questions

5% completed



Q Search Course

(/collection/page/56686391014

Fruits into Baskets (medium) (/collection/page/56686391014

No-repeat Substring (hard) (/collection/page/56686391014

Longest Substring with Same Letters after Replacement (hard) (/collection/page/56686391014

Longest Subarray with Ones after Replacement (hard) (/collection/page/56686391014

Problem Challenge 1 (/collection/page/56686391014

Solution Review: Problem Challenge 1