## 438. Find All Anagrams in a String 438.

Feb. 3, 2020 | 41.9K views

\*\*\* Average Rating: 4.43 (21 votes)

**6 0 0** 

### Given a string s and a non-empty string p, find all the start indices of p's anagrams in s.

Strings consists of lowercase English letters only and the length of both strings s and p will not be larger than 20,100.

The order of output does not matter.

## Example 1:

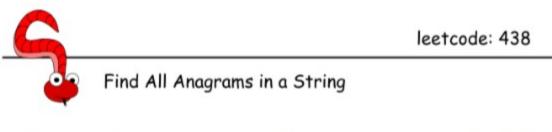
```
Input:
s: "cbaebabacd" p: "abc"
Output:
[0, 6]
Explanation:
The substring with start index = 0 is "cba", which is an anagram of "abc".
The substring with start index = 6 is "bac", which is an anagram of "abc".
```

Example 2:

```
Input:
s: "abab" p: "ab"
Output:
[0, 1, 2]
Explanation:
The substring with start index = 0 is "ab", which is an anagram of "ab".
The substring with start index = 1 is "ba", which is an anagram of "ab".
The substring with start index = 2 is "ab", which is an anagram of "ab".
```

# Solution

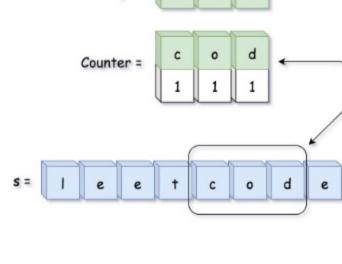
Solution Template



linear time. If the patterns are not known in advance, i.e. it's "find duplicates" problem, one could use one of two ways to implement constant-time slice: Bitmasks or Rabin-Karp. Please check article Repeated DNA Sequences for the

This is a problem of multiple pattern search in a string. All such problems usually could be solved by sliding window approach in a linear time. The challenge here is how to implement constant-time slice to fit into this

Here the situation is more simple: patterns are known in advance, and the set of characters in the patterns is very limited as well: 26 lowercase English letters. Hence one could allocate array or hashmap with 26 elements and use it as a letter counter in the sliding window.



### The first hashmap is a reference counter pcount for string p, and the second one is a counter scount for

Approach 1: Sliding Window with HashMap

detailed comparison of these two algorithms.

string in the sliding window. The idea is to move sliding window along the string s, recompute the second hashmap sCount in a constant time and compare it with the first hashmap pCount . If sCount == pCount , then the string in the

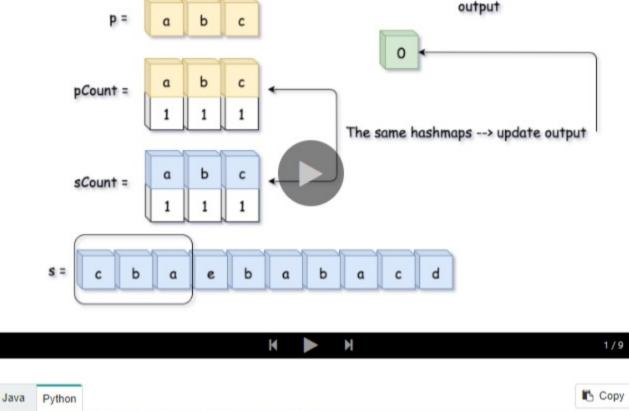
Let's start from the simplest approach: sliding window + two counter hashmaps letter -> its count.

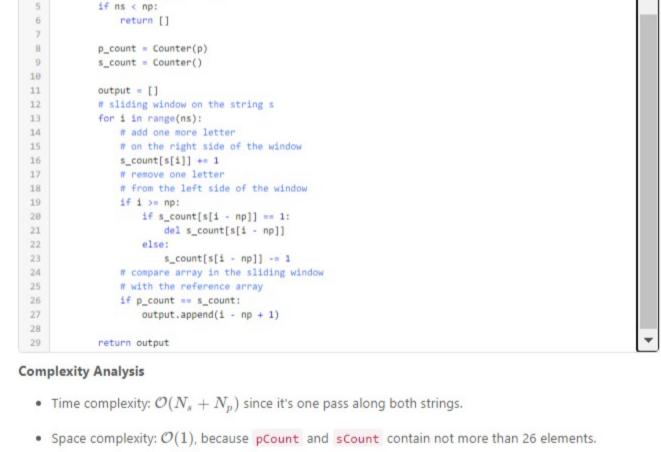
sliding window is a permutation of string p, and one could add its start position in the output list. Algorithm Build reference counter pCount for string p.

- Move sliding window along the string s: Recompute sliding window counter sCount at each step by adding one letter on the right and
- removing one letter on the left.
- o If sCount == pCount, update the output list. · Return output list.
- Implementation

def findAnagrams(self, s: str, p: str) -> List[int]:

ns, np = len(s), len(p)





- Approach 2: Sliding Window with Array Algorithm
- Hashmap is quite complex structure, with known performance issues in Java. Let's implement approach 1 using 26-elements array instead of hashmap:
- Element number 26 contains count of letter z.

Element number 0 contains count of letter a.

Element number 1 contains count of letter b.

- Algorithm Build reference array pCount for string p. Move sliding window along the string s:
  - Recompute sliding window array sCount at each step by adding one letter on the right and removing one letter on the left. If sCount == pCount, update the output list.

ns, np = len(s), len(p)

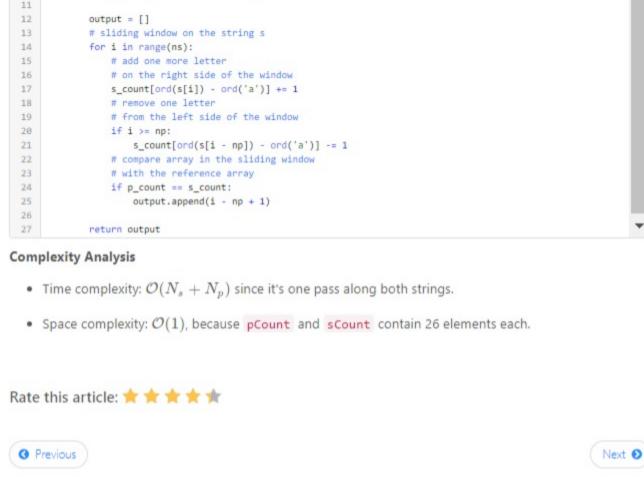
- **Сору** Java Python 1 class Solution: def findAnagrams(self, s: str, p: str) -> List[int]:
- if ns < np: return [] p\_count, s\_count = [0] \* 26, [0] \* 26 # build reference array using string p

Return output list.

Implementation

### for ch in p: 10 p\_count[ord(ch) - ord('a')] += 1 11 12 13

Comments: 23



operation i.e. O(1). If we didn't have a bound on the number of input characters we might want to consider a more 56 ∧ ∨ lt Share ← Reply SHOW 4 REPLIES

@ Preview

Type comment here... (Markdown is supported)

bshaibu 🛊 175 🗿 February 6, 2020 9:53 PM

monhoshum \* 15 February 27, 2020 10:45 AM

alphabet then it's O(string.size() \* 26) == O(string.size())

harsh014 ★ 61 ② February 23, 2020 4:21 AM Why is this solution O(S + P)? Shouldn't it just be O(S) since S will always be larger than P? 26 A V & Share Share Reply **SHOW 4 REPLIES** 

Why is the runtime O(S + P)? In the for loop, it compares the p\_count and s\_count, which has a runtime

It's probably worth noting: we can compare p\_count and s\_count in constant time because they are both at most size 26 (as they only contain the 26 lowercase characters). This makes comparing an O(26)

Sort By \*

Post

of O(P). should the runtime be O(P + SP) = O(SP)? 14 A V & Share A Reply SHOW 4 REPLIES Should time complexity be O(string.size() \* alphabet.size()) ? because a comparison of two arrays/hashmaps is not const time operation . Sure, if we assume that our alphabet is always English

ntkw \$ 58 @ April 14, 2020 8:54 PM I'm not convinced about the reported time complexity, isn't time complexity NxP where P is the number of unique characters in p String? once hashmap equals is time complexity O(n)? Or Am I mis interpretating? Thank you 3 A V Et Share A Reply

asish\_cse # 4 @ April 6, 2020 9:12 PM Time complexity should be O(S + P) or O(S) as S will always be larger than P? 2 A V & Share A Reply SHOW 1 REPLY

if (s == null || s.length() == 0) return res; 2 A V 🖸 Share 🦘 Reply SHOW 1 REPLY

CoderJoe \* 702 @ February 5, 2020 7:13 PM public List findAnagrams(String s, String p) {

user4717V \* 1 @ July 1, 2020 7:16 PM

1 A V E Share A Reply

Pasting the entire code below

venki07 🛊 18 🗿 July 13, 2020 6:47 PM

SHOW 1 REPLY

( 1 2 3 )

List res = new ArrayList<>();

p\_count == s\_count is faster since we have to check the array is empty in O(1) instead of comparing all of the character counts worst case in O(p). 1 A V Et Share A Reply wengkeat575 \* 0 O May 20, 2020 5:37 AM Can someone explained: Why is the space complexity is O(1) in sliding window with array? When you create new array and using it to store the count, won't it be using the space?

Hi, not able to wrap my head around why 2nd code snippet has better performance than the 1st.

instead of calculating both p\_count and s\_count we can keep track of the difference between s and p and add the index to the output everytime the difference array is all 0. This way the comparison of

1. Read More 0 A V E Share + Reply