Find all Anagram

27 April 2022 23:27

#Acquire_release_strategy
#Sliding_window

https://pepcoding.com/resources/data-structures-and-algorithms-in-java-levelup/hashmap-and-heaps/find-all-anagrams-in-a-string-official/ojquestion

ANAGRAM - Two strings are in anagram, when all characters must be present maybe the order is different.
EX- say
Anagram - ysa

Question is - Let "S1" is string, find all the anagram in the string "S2", and print the starting index of it.
Ex -
cbaebaba**bac**d -->S2
abc --> S1

Out put -
2 -- > Total anagram found
0 6 --> Their starting indexes

APPROACH IS - 1. Create a hashmap for string "s1" .

2. Using #Sliding_window approach or #Acquire_release strategy of same length as string s1. Run it on s2.
3. As we get same frequencies of each character, that will be one of the result. Then move ahead, till last.

```
import java.util.*;
public class Main {
        public static void findAnagrams(String s, String p) {
    // write your code here
```

```
    int[] map2 = new int[26];
    for(int i = 0; i < p.length(); i++){
        char ch = p.charAt(i);
        map2[ch - 'a']++;
    }
```

Create a hashmap for p.

```
    int[] map1 = new int[26];
    for(int i = 0; i < p.length(); i++){
        char ch = s.charAt(i);
        map1[ch - 'a']++;
    }
```

Create a hashmap for s, which size must be of p.
- Start from 0 to p.size, make a hashmap

```
            char ch = s.charAt(i);
            map1[ch - 'a']++;
        }
```

```
        ArrayList<Integer> res = new ArrayList<>();
```

Create an arraylist to store the starting points

```
        if(areAnagram(map1, map2) == true){
            res.add(0);
        }
```

Check whether map1 and map2 are same or not. If yes add 0 to the arraylist.

```
        int i = p.length();
        int j = 0;
        while(i < s.length()){
            char chi = s.charAt(i);
            map1[chi - 'a']++;
            char chj = s.charAt(j);
            map1[chj - 'a']--;
            if(areAnagram(map1, map2) == true){
                res.add(j + 1);
            }
            i++;
            j++;
        }
```

Now do the samething for p.size+1 to s.size.
Add one character to it, and remove the current first character,
And check everytime.

```
        System.out.println(res.size());
        for(int val: res){
            System.out.print(val + " ");
        }
    }


    // constant time complexity
    public static boolean areAnagram(int[] map1, int[] map2){
        for(int i = 0; i < 26; i++){
            if(map1[i] != map2[i]){
                return false;
            }
        }
        return true;
    }
```

Alagram function to check whether two string are
Alagram or not.

```
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String s = scn.next();
        String p = scn.next();
        findAnagrams(s, p);
    }
}
```