Is Subsequence

Given a string **s** and a string **t**, check if **s** is subsequence of **t**.

You may assume that there is only lower case English letters in both **s** and **t**. **t** is potentially a very long (length ~= 500,000) string, and **s** is a short string (

A subsequence of a string is a new string which is formed from the original string by deleting some (can be none) of the characters without disturbing the relative positions of the remaining characters. (ie, `"ace"` is a subsequence of `"abcde"` while `"aec"` is not).

Example 1:
s = `"abc"`, **t** = `"ahbgdc"`

Return `true`.

Example 2:
s = `"axc"`, **t** = `"ahbgdc"`

Return `false`.

Follow up:
If there are lots of incoming S, say S1, S2, ... , Sk where k >= 1B, and you want to check one by one to see if T has its subsequence. In this scenario, how would you change your code?

Credits:
Special thanks to @pbrother for adding this problem and creating all test cases.

Subscribe to see which companies asked this question

Solution 1

```
bool isSubsequence(char* s, char* t) {
    while (*t)
        s += *s == *t++;
    return !*s;
}
```

Just go through t and "cross off" the matching characters in s. Then return whether there's nothing left in s.

Sometimes, C makes things easier... here it's helpful that C strings are null terminated and that I can easily pop off a string's first character in constant time just by incrementing the pointer.

written by StefanPochmann original link here

Solution 2

Just use two pointers:

```
public class Solution {
    public boolean isSubsequence(String s, String t) {
        if (s.length() == 0) return true;
        int indexS = 0, indexT = 0;
        while (indexT < t.length()) {
            if (t.charAt(indexT) == s.charAt(indexS)) {
                indexS++;
                if (indexS == s.length()) return true;
            }
            indexT++;
        }
        return false;
    }
}
```

written by sampsonchan original link here

Solution 3

```
def isSubsequence(self, s, t):
    t = iter(t)
    return all(c in t for c in s)
```

Just testing whether all characters in s are also in t (in order).

Based on falsetru's code on StackOverflow which I improved a while ago,see here.

written by StefanPochmann original link here

From Leetcoder.