Rembrandt Reyes (He/Him)
Posted on ٣١ يوليو ٢٠٢٠

Let's solve LeetCode - Is Subsequence

#algorithms #javascript #tutorial

Problem 392 - Is Subsequence

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

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).

Examples

```
Input: s = "abc", t = "ahbgdc"
Output: true
```

```
Input: s = "axc", t = "ahbgdc"
Output: false
```

Conceptual Overview

Since we want to check if `s` is a subsequence of `t` we will want to check every character of `s` against `t` and if a character at `s` matches a character in `t` (in order) then we can move onto the next character in `s` and do the check all over again.

Looking at the example above let's run through a couple of iterations. In ECMAScript 5 we can treat the string as an array-like object, where individual characters correspond to a numerical index.

1) At s[0] = a; t[0] = a; does s[0] === t[0]? Yes, move to the next character in `s` and `t`
2) At s[1] = b; t[1] = h; does s[1] === t[0]? No, move to the next character in `t`
3) At s[1] = b; t[2] = b; does s[1] === t[2]? Yes, move to the next character in `s` and `t`
...
6) At s[2] = c; t[5] = c; does s[3] === t[5]? Yes, and since we went through the length of `s` we found `s` to be a subsequence of `t`

Code

While-loop variation

```
/**
 * @param {string} s
 * @param {string} t
 * @return {boolean}
 */
const isSubsequence = (s, t) => {
    if (s.length === 0) return true

    let sPointer = 0
    let tPointer = 0
```

```
    let tPointer = 0


    while (sPointer < s.length && tPointer < t.length) {
        if(s[sPointer] === t[tPointer]) sPointer++

        tPointer++
    }

    return sPointer === s.length

};
```

For-loop variation

```
const isSubsequence = (s, t) => {
    if (s.length === 0) return true

    let sPointer = 0

    for (let i = 0; i < t.length; i++) {
        if (s[sPointer] === t[i]) sPointer++
    }
    return sPointer === s.length
}
```

In both code solutions, we need to keep track of our position in each string so we use pointers to help with that.

Time & Space Complexity

Time: O(n) - where n is the length of the array
Space: O(1)
Both variations have the same time and space complexity

Discussion (0)

Code of Conduct • Report abuse

Rembrandt Reyes (He/Him)

I am here to make myself a better engineer. I create things for the web using the ReactJS ecosystem.

LOCATION
San Francisco, CA

WORK
Front End Engineer at Hopjump

JOINED
٢٩ يونيو ٢٠٢٠

More from Rembrandt Reyes (He/Him)

Wait... how does React.useState work?

#tutorial #react #beginners

Let's solve LeetCode! Fibonacci Number

#beginners #tutorial #javascript

eSlayers part 7 - fetching more data for math history

#javascript #react #nextjs #typescript