Backspace String Compare 4.27 (212 votes) ★★★★☆ ···

Backspace String Compare

LeetCode Admin Jun 06, 2018

Approach #1: Build String [Accepted]

Intuition

Let's individually build the result of each string (`build(S)` and `build(T)`), then compare if they are equal.

Algorithm

To build the result of a string `build(S)` , we'll use a stack based approach, simulating the result of each keystroke.

Java Python Copy

```
class Solution {
    public boolean backspaceCompare(String S, String T) {
        return build(S).equals(build(T));
    }

    public String build(String S) {
        Stack<Character> ans = new Stack();
        for (char c: S.toCharArray()) {
            if (c != '#')
                ans.push(c);
            else if (!ans.empty())
                ans.pop();
        }
        return String.valueOf(ans);
    }
}
```

Complexity Analysis

- Time Complexity: $O(M+N)$, where M, N are the lengths of `S` and `T` respectively.
- Space Complexity: $O(M+N)$.

Approach #2: Two Pointer [Accepted]

Intuition

When writing a character, it may or may not be part of the final string depending on how many backspace keystrokes occur in the future.

If instead we iterate through the string in reverse, then we will know how many backspace characters we have seen, and therefore whether the result includes our character.

Algorithm

Iterate through the string in reverse. If we see a backspace character, the next non-backspace character is skipped. If a character isn't skipped, it is part of the final answer.

See the comments in the code for more details.

Java Python Copy

```java
class Solution {
    public boolean backspaceCompare(String S, String T) {
        int i = S.length() - 1, j = T.length() - 1;
        int skipS = 0, skipT = 0;

        while (i >= 0 || j >= 0) { // While there may be chars in build(S) or build (T)
            while (i >= 0) { // Find position of next possible char in build(S)
                if (S.charAt(i) == '#') {skipS++; i--;}
                else if (skipS > 0) {skipS--; i--;}
                else break;
            }
            while (j >= 0) { // Find position of next possible char in build(T)
                if (T.charAt(j) == '#') {skipT++; j--;}
                else if (skipT > 0) {skipT--; j--;}
                else break;
            }
            // If two actual characters are different
            if (i >= 0 && j >= 0 && S.charAt(i) != T.charAt(j))
                return false;
            // If expecting to compare char vs nothing
            if ((i >= 0) != (j >= 0))
                return false;
            i--; j--;
        }
        return true;
    }
}
```

Complexity Analysis

- Time Complexity: $O(M+N)$, where M, N are the lengths of `S` and `T` respectively.
- Space Complexity: $O(1)$.

Share

Comments (153)

Sort by: Best

Type comment here... (Markdown supported)

Preview Comment

Article Commenting Rules

1. This comment section is for questions and comments regarding this **LeetCode article**. All posts must respect our LeetCode Community Rules.

2. Concerns about errors or bugs in the article, problem description, or test cases should be posted on LeetCode Feedback, so that our team can address them.

paimei Apr 09, 2020

This question should be at least marked as Medium difficulty
if one need to come out with O(n) time and O(1) space
complexity solution.

562 Show 6 Replies Reply

cubestack Apr 27, 2021

Don't think there's any interviewer that's going to want to see itertools in the solution. No idea why this is the code in the solution.

167 Show 12 Replies Reply

cbmbbz Feb 04, 2019

solution 2 python: reversed(S) internally creates a string and that makes it not O(1). isn't it?

96 Show 10 Replies Reply

t_alex Feb 21, 2021

This should be a Medium.

▲ 58 ▼ Show 2 Replies Reply

pacheng Sep 16, 2020

This is not a easy question :(

▲ 33 ▼ Reply

sush33 Jun 15, 2018

The python solution is so beautiful.

▲ 168 ▼ Show 3 Replies Reply

10000tb Apr 15, 2019

Second solution is genius !

▲ 51 ▼ Show 1 Replies Reply

haoyangfan Jan 05, 2019

Offer my Java solution for the third idea, which factor out the logic in while loop into a separate method to make the code more readable and easy to undertand

```
class Solution {
    public boolean backspaceCompare(String S, String T) {
        int s = S.length() - 1, t = T.length() - 1;

        // start to compare string from the end
        while (s >= 0 && t >= 0) {
            s = nextNonSkipChar(S, s);
```

▲ 33 ▼ Show 5 Replies Reply

wh1210 Jul 09, 2018

Read more

▲ 33 ▼ Show 9 Replies Reply

bhisham1999 Mar 30, 2020

Surprised such simple analysis is written by @awice lol :)

▲ 15 ▼ Show 3 Replies Reply

‹ 1 2 3 4 5 6 ··· 16 ›