

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.

```
Сору
Java
       Python
   class Solution {
1
2
       public boolean backspaceCompare(String S, String T) {
3
           return build(S).equals(build(T));
4
       }
5
6
       public String build(String S) {
7
           Stack<Character> ans = new Stack();
            for (char c: S.toCharArray()) {
8
9
               if (c != '#')
10
                   ans.push(c);
               else if (!ans.empty())
11
12
                   ans.pop();
13
           }
14
            return String.valueOf(ans);
15
        }
16 }
```

Complexity Analysis

- ullet Time Complexity: O(M+N), where M,N are the lengths of ${\,{\sf S}\,}$ and ${\,{\sf T}\,}$ 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.

```
Official Solution - Backspace String Compare - LeetCode
                                                                                                                                       🖺 Сору
         Python
  Java
  1
      class Solution {
  2
          public boolean backspaceCompare(String S, String T) {
              int i = S.length() - 1, j = T.length() - 1;
  3
              int skipS = 0, skipT = 0;
   4
  5
              while (i >= 0 \mid \mid j >= 0) { // While there may be chars in build(S) or build (T)
  6
                  while (i \geq= 0) { // Find position of next possible char in build(S)
  7
  8
                      if (S.charAt(i) == '#') {skipS++; i--;}
  9
                      else if (skipS > 0) {skipS--; i--;}
  10
                      else break;
  11
                  }
                  while (j \ge 0) { // Find position of next possible char in build(T)
  12
                      if (T.charAt(j) == '#') {skipT++; j--;}
  13
                      else if (skipT > 0) {skipT--; j--;}
  14
  15
                      else break;
  16
                  }
                  // If two actual characters are different
  17
                  if (i \ge 0 \&\& j \ge 0 \&\& S.charAt(i) != T.charAt(j))
  18
  19
                      return false;
                  // If expecting to compare char vs nothing
  20
                  if ((i >= 0) != (j >= 0))
  21
  22
                      return false;
  23
                  i--; j--;
  24
              }
  25
              return true;
  26
          }
Complexity Analysis
  • Time Complexity: O(M+N), where M,N are the lengths of S and T respectively.
  • Space Complexity: O(1).
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

 \times

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.



Apr 27, 2021

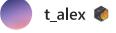
Don't think there's any interviewer that's going to want to see itertools in the solution. No idea why this is the code in the solution.

▲ 167 ▼ Q Show 12 Replies 🖎 Reply

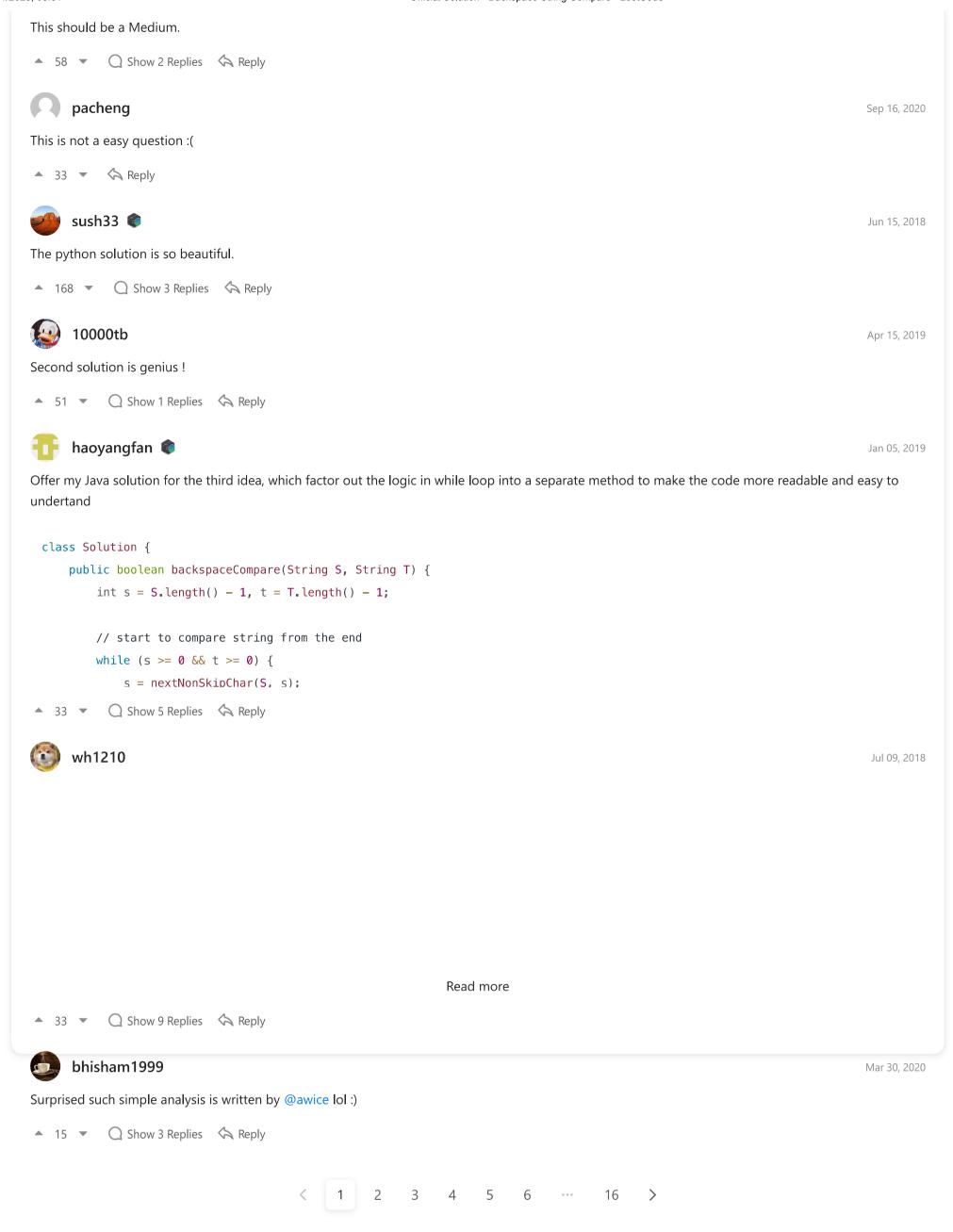


cbmbbz Feb 04, 2019

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



Feb 21, 2021



https://leetcode.com/problems/backspace-string-compare/solutions/136876/backspace-string-compare/?envType=study-plan&id=algorithm-iiwas. The problems of the