616. Add Bold Tag in a String (/problems/add-bold-tag-in-string/)

June 10, 2017 | 5.6K views

Given a string **s** and a list of strings **dict**, you need to add a closed pair of bold tag `<b>` and `</b>` to wrap the substrings in s that exist in dict. If two such substrings overlap, you need to wrap them together by only one pair of closed bold tag. Also, if two substrings wrapped by bold tags are consecutive, you need to combine them.

Example 1:

```
Input:
s = "abcxyz123"
dict = ["abc","123"]
Output:
"<b>abc</b>xyz<b>123</b>"
```

Example 2:

```
Input:
s = "aaabbcc"
dict = ["aaa","aab","bc"]
Output:
"<b>aaabbc</b>c"
```

Note:

1. The given dict won't contain duplicates, and its length won't exceed 100.
2. All the strings in input have length in range [1, 1000].

Approach #1: Brute Force [Accepted]

Intuition

Let's try to learn which letters end up bold, since the resulting answer will just be the canonical one - we put bold tags around each group of bold letters.

To do this, we'll check for all occurrences of each word and mark the corresponding letters bold.

Algorithm

Let's work on first setting `mask[i] = true` if and only if the `i`-th letter is bold. For each starting position `i` in `S`, for each `word`, if `S[i]` starts with `word`, we'll set the appropriate letters bold.

Now armed with the correct `mask`, let's try to output the answer. A letter in position `i` is the first bold letter of the group if `mask[i] && (i == 0 || !mask[i-1])`, and is the last bold letter if `mask[i] && (i == N-1 || !mask[i+1])`. Alternatively, we could use `itertools.groupby` in Python.

Once we know which letters are the first and last bold letters of a group, we know where to put the `"<b>"` and `"</b>"` tags.

Java Python Copy

```java
class Solution {
    public String boldWords(String S, String[] words) {
        int N = S.length();
        boolean[] mask = new boolean[N];
        for (int i = 0; i < N; ++i)
            for (String word: words) search: {
                for (int k = 0; k < word.length(); ++k)
                    if (k+i >= S.length() || S.charAt(k+i) != word.charAt(k))
                        break search;

                for (int j = i; j < i+word.length(); ++j)
                    mask[j] = true;
            }

        StringBuilder ans = new StringBuilder();
        int anchor = 0;
        for (int i = 0; i < N; ++i) {
            if (mask[i] && (i == 0 || !mask[i-1]))
                ans.append("<b>");
            ans.append(S.charAt(i));
            if (mask[i] && (i == N-1 || !mask[i+1]))
                ans.append("</b>");
        }
        return ans.toString();
    }

    public boolean match(String S, int i, int j, String T) {
```

Complexity Analysis

- Time Complexity: $O(N \sum w_i)$, where N is the length of `S` and w_i is the sum of `W`.
- Space Complexity: $O(N)$.

Analysis written by: @awice (https://leetcode.com/awice).

Rate this article:
(/ratings/107/166/?return=/articles/add-bold-tag-in-a-string/) (/ratings/107/166/?return=/articles/

Previous (/articles/valid-triangle-number/)   Next (/articles/palindrome-permutation-ii/)

Join the conversation

Login to Reply

L **linwenboJob** commented 3 days ago
(https://discuss.leetcode.com/user/linwenbojob)

why break the outerloop

A **achuthna** commented 2 weeks ago
(https://discuss.leetcode.com/user/achuthna)

Where exactly is the second solution? I don't see it now. Is there anything better than brute force?

8 **8939123** commented 3 months ago
(https://discuss.leetcode.com/user/8939123)

Can you please explain more on time complexity analysis? The 1st method I thought it is s^2; 2nd solution where does the x come from? I thought it is O(ls). Thanks!

B (https://discuss.leetcode.com/user/berkeleysucks)

berkeleysucks commented 4 months ago

The complexity analysis for method#2 is incorrect. We also need to factor in the runtime for sorting.

C (https://discuss.leetcode.com/user/codermonk)

codermonk commented 7 months ago

I have a question about the runtime of Approach#2: will the line of sorting "Collections.sort(list, (a, b) -> a[0] == b[0] ? a[1] - b[1] : a[0] - b[0]);" be the dominant part of run time?
Since in the worst case the list size is O(s∗l), I guess sorting can take O(s/log(s*l)) ?

View original thread (https://discuss.leetcode.com/topic/92124)