Description | Solution | Discuss (296) | Submissions

Java | Autocomplete

737. Sentence Similarity II

Medium 656 39 Add to List Share

We can represent a sentence as an array of words, for example, the sentence `"I am happy with leetcode"` can be represented as `arr = ["I","am",happy","with","leetcode"]`.

Given two sentences `sentence1` and `sentence2` each represented as a string array and given an array of string pairs `similarPairs` where `similarPairs[i] = [xi, yi]` indicates that the two words `xi` and `yi` are similar.

Return `true` *if* `sentence1` *and* `sentence2` *are similar, or* `false` *if they are not similar*.

Two sentences are similar if:

- They have **the same length** (i.e., the same number of words)
- `sentence1[i]` and `sentence2[i]` are similar.

Notice that a word is always similar to itself, also notice that the similarity relation is transitive. For example, if the words `a` and `b` are similar, and the words `b` and `c` are similar, then `a` and `c` are **similar**.

Example 1:

```
Input: sentence1 = ["great","acting","skills"], sentence2 =
["fine","drama","talent"], similarPairs = [["great","good"],["fine","good"],
["drama","acting"],["skills","talent"]]
Output: true
Explanation: The two sentences have the same length and each word i of sentence1
is also similar to the corresponding word in sentence2.
```

Example 2:

```
Input: sentence1 = ["I","love","leetcode"], sentence2 = ["I","love","onepiece"],
similarPairs = [["manga","onepiece"],["platform","anime"],["leetcode","platform"],
["anime","manga"]]
Output: true
Explanation: "leetcode" --> "platform" --> "anime" --> "manga" --> "onepiece".
Since "leetcode is similar to "onepiece" and the first two words are the same, the
two sentences are similar.
```

Example 3:

```
Input: sentence1 = ["I","love","leetcode"], sentence2 = ["I","love","onepiece"],
similarPairs = [["manga","hunterXhunter"],["platform","anime"],
["leetcode","platform"],["anime","manga"]]
Output: false
Explanation: "leetcode" is not similar to "onepiece".
```

Constraints:

- `1 <= sentence1.length, sentence2.length <= 1000`
- `1 <= sentence1[i].length, sentence2[i].length <= 20`
- `sentence1[i]` and `sentence2[i]` consist of lower-case and upper-case English letters.
- `0 <= similarPairs.length <= 2000`
- `similarPairs[i].length == 2`
- `1 <= xi.length, yi.length <= 20`
- `xi` and `yi` consist of English letters.

Accepted 54,497 | Submissions 114,915

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Amazon | 2

Related Topics

Array Hash Table String Depth-First Search Breadth-First Search Union Find

Similar Questions

Number of Provinces	Medium
Accounts Merge	Medium
Sentence Similarity	Easy

Hide Hint 1

Consider the graphs where each pair in "pairs" is an edge. Two words are similar if they are the same, or are in the same connected component of this graph.

```java
class Solution {
    public boolean areSentencesSimilarTwo(
            String[] words1, String[] words2, String[][] pairs) {
        if (words1.length != words2.length) return false;
        Map<String, List<String>> graph = new HashMap();
        for (String[] pair: pairs) {
            for (String p: pair) if (!graph.containsKey(p)) {
                graph.put(p, new ArrayList());
            }
            graph.get(pair[0]).add(pair[1]);
            graph.get(pair[1]).add(pair[0]);
        }

        for (int i = 0; i < words1.length; ++i) {
            String w1 = words1[i], w2 = words2[i];
            Stack<String> stack = new Stack();
            Set<String> seen = new HashSet();
            stack.push(w1);
            seen.add(w1);
            search: {
                while (!stack.isEmpty()) {
                    String word = stack.pop();
                    if (word.equals(w2)) break search;
                    if (graph.containsKey(word)) {
                        for (String nei: graph.get(word)) {
                            if (!seen.contains(nei)) {
                                stack.push(nei);
                                seen.add(nei);
                            }
                        }
                    }
                }
                return false;
            }
        }
        return true;
    }
}
```