737. Sentence Similarity II

Given two sentences `words1, words2` (each represented as an array of strings), and a list of similar word pairs `pairs`, determine if two sentences are similar.

For example, `words1 = ["great", "acting", "skills"]` and `words2 = ["fine", "drama", "talent"]` are similar, if the similar word pairs are `pairs = [["great", "good"], ["fine", "good"], ["acting","drama"], ["skills","talent"]]`.

Note that the similarity relation **is** transitive. For example, if "great" and "good" are similar, and "fine" and "good" are similar, then "great" and "fine" **are similar**.

Similarity is also symmetric. For example, "great" and "fine" being similar is the same as "fine" and "great" being similar.

Also, a word is always similar with itself. For example, the sentences `words1 = ["great"], words2 = ["great"], pairs = []` are similar, even though there are no specified similar word pairs.

Finally, sentences can only be similar if they have the same number of words. So a sentence like `words1 = ["great"]` can never be similar to `words2 = ["doubleplus","good"]`.

Note:

- The length of `words1` and `words2` will not exceed `1000`.
- The length of `pairs` will not exceed `2000`.
- The length of each `pairs[i]` will be `2`.
- The length of each `words[i]` and `pairs[i][j]` will be in the range `[1, 20]`.

Seen this question in a real interview before?

- Difficulty:Medium
- Total Accepted:2.4K
- Total Submissions:5.9K
- Contributor: 1337c0d3r
-
- Subscribe to see which companies asked this question.

  Related Topics Union Find

  Similar Questions

  Friend Circles Accounts Merge Sentence Similarity

```
// C++
// let map<string a,string b> record the parent of string a--> string b
string find(string s, unordered_map<string,string> &map)
{
     if(map.count(s)==0)
     {
          map[s]=s;
          return s;
     }
     if(map[s]!=s)
     {
          map[s] = find(map[s],map);
     }
     return map[s];
}

void build_union(unordered_map<string,string> &map, vector<pair<string,
string>> pairs)
{
     for(auto elem:pairs)
     {
          string s1 = elem.first;
          string s2 = elem.second;
          string g1 = find(s1,map);
          string g2 = find(s2,map);
          if(g1!=g2)
          {
               map[g2]=g1;
          }
     }
}
bool areSentencesSimilarTwo(vector<string>& words1, vector<string>&
words2,
vector<pair<string, string>> pairs)
{
     unordered_map<string,string> union_map;
     build_union(union_map,pairs);
     for(int i=0;i<(int)words1.size();++i)
     {
          string s1 = words1[i];
          string s2 = words2[i];
          string g1 = find(s1,union_map);
          string g2 = find(s2,union_map);
          if(g1!=g2) return false;
     }
     return true;
}
```