Alien Dictionary

There is a new alien language which uses the latin alphabet. However, the order among letters are unknown to you. You receive a list of words from the dictionary, where **words are sorted lexicographically by the rules of this new language**. Derive the order of letters in this language.

For example,
Given the following words in dictionary,

```
[
  "wrt",
  "wrf",
  "er",
  "ett",
  "rftt"
]
```

The correct order is: `"wertf"`.

Note:

1. You may assume all letters are in lowercase.
2. If the order is invalid, return an empty string.
3. There may be multiple valid order of letters, return any one of them is fine.

Solution 1

Two similar solutions. Both first go through the word list to find letter pairs `(a, b)` where `a` must come before `b` in the alien alphabet. The first solution just works with these pairs, the second is a bit smarter and uses successor/predecessor sets. Doesn't make a big difference here, though, I got both accepted in 48 ms.

Solution 1

```
def alienOrder(self, words):
    less = []
    for pair in zip(words, words[1:]):
        for a, b in zip(*pair):
            if a != b:
                less += a + b,
                break
    chars = set(''.join(words))
    order = []
    while less:
        free = chars - set(zip(*less)[1])
        if not free:
            return ''
        order += free
        less = filter(free.isdisjoint, less)
        chars -= free
    return ''.join(order + list(chars))
```

Solution 2

```
def alienOrder(self, words):
    pre, suc = collections.defaultdict(set), collections.defaultdict(set)
    for pair in zip(words, words[1:]):
        for a, b in zip(*pair):
            if a != b:
                suc[a].add(b)
                pre[b].add(a)
                break
    chars = set(''.join(words))
    free = chars - set(pre)
    order = ''
    while free:
        a = free.pop()
        order += a
        for b in suc[a]:
            pre[b].discard(a)
            if not pre[b]:
                free.add(b)
    return order * (set(order) == chars)
```

C++ version of solution 2

```
string alienOrder(vector<string>& words) {
    map<char, set<char>> suc, pre;
    set<char> chars;
    string s;
    for (string t : words) {
        chars.insert(t.begin(), t.end());
        for (int i=0; i<min(s.size(), t.size()); ++i) {
            char a = s[i], b = t[i];
            if (a != b) {
                suc[a].insert(b);
                pre[b].insert(a);
                break;
            }
        }
        s = t;
    }
    set<char> free = chars;
    for (auto p : pre)
        free.erase(p.first);
    string order;
    while (free.size()) {
        char a = *begin(free);
        free.erase(a);
        order += a;
        for (char b : suc[a]) {
            pre[b].erase(a);
            if (pre[b].empty())
                free.insert(b);
        }
    }
    return order.size() == chars.size() ? order : "";
}
```

written by StefanPochmann original link here

Solution 2

Why do I see this error?

Input: ["wrt","wrf","er","ett","rftt"] Output: "werft" Expected: Special judge: No expected output available.

written by FerociousCodingKitten original link here

Solution 3

The question says: if the input is `[ "wrt", "wrf", "er", "ett", "rftt" ]` The correct order is: `"wertf"`. but from `"rftt"`, f should be lexicographically smaller than t? How can the result be "wertf"? Correct me if I am wrong.

written by AndyLiu0429 original link here