

5231. Remove Sub-Folders from the Filesystem

[My Submissions \(/contest/weekly-contest-159/problems/remove-sub-folders-from-the-filesystem/submissions/\)](/contest/weekly-contest-159/problems/remove-sub-folders-from-the-filesystem/submissions/)

[Back to Contest \(/contest/weekly-contest-159/\)](/contest/weekly-contest-159/)

Given a list of folders, remove all sub-folders in those folders and return in **any order** the folders after removing.

If a `folder[i]` is located within another `folder[j]`, it is called a sub-folder of it.

The format of a path is one or more concatenated strings of the form: `/` followed by one or more lowercase English letters. For example, `/leetcode` and `/leetcode/problems` are valid paths while an empty string and `/` are not.

User Accepted:	1279
User Tried:	1578
Total Accepted:	1288
Total Submissions:	2394
Difficulty:	Medium

Example 1:

Input: `folder = ["/a", "/a/b", "/c/d", "/c/d/e", "/c/f"]`

Output: `["/a", "/c/d", "/c/f"]`

Explanation: Folders `"a/b/"` is a subfolder of `"a"` and `"c/d/e"` is inside of folder `"c/d"` in our filesystem.

Example 2:

Input: `folder = ["/a", "/a/b/c", "/a/b/d"]`

Output: `["/a"]`

Explanation: Folders `"a/b/c"` and `"a/b/d/"` will be removed because they are subfolders of `"a"`.

Example 3:

Input: `folder = ["/a/b/c", "/a/b/ca", "/a/b/d"]`

Output: `["/a/b/c", "/a/b/ca", "/a/b/d"]`

Constraints:

- `1 <= folder.length <= 4 * 104`
- `2 <= folder[i].length <= 100`
- `folder[i]` contains only lowercase letters and `'/'`
- `folder[i]` always starts with character `'/'`
- Each folder name is unique.

C++



```

1 class Solution {
2 public:
3     vector<string> removeSubfolders(vector<string>& folder) {
4         int num = folder.size();
5
6         sort(folder.begin(), folder.end());
7         // => ["/a", "/a/b", "/c/d", "/c/d/e", "/c/f"]
8         vector<string> f;
9
10        if( num>=1 ) {
11            f.push_back(folder[0]);
12            int pos = 0;
13            for(int i=1; i<num; ++i)
14                if(!isPrefix(folder[pos], folder[i])) { // folder[pos] is not prefix of folder[i]
15                    f.push_back(folder[i]);
16                    pos = i;
17                }
18        }
19    }
20 }

```

```
18     }
19     return f;
20 }
21 ▼ bool isPrefix(string &a, string &b) {
22     int pos = b.find(a);
23 ▼     if(pos == 0 && b[a.length()] == '/') {
24         // cout << a << " " << b << endl;
25         return true;
26     }
27
28     return false;
29 }
30 };
```


☐ Custom Testcase

 Run

 Submit

Copyright © 2019 LeetCode

[Help Center \(/support/\)](#) | [Terms \(/terms/\)](#) | [Privacy Policy \(/privacy/\)](#)

 [United States \(/region/\)](#)