Problem 1233: Remove Sub-Folders from the Filesystem

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

Given a list of folders

folder

, return

the folders after removing all

sub-folders

in those folders

. You may return the answer in

any order

.

If a

folder[i]

is located within another

folder[j]

, it is called a

sub-folder

of it. A sub-folder of

folder[j]

must start with

folder[j]

, followed by a

"/"

. For example,

"/a/b"

is a sub-folder of

"/a"

, but

"/b"

is not a sub-folder of

"/a/b/c"

.

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.

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 * 10

4

2 <= folder[i].length <= 100

folder[i]

contains only lowercase letters and

'/'

.

folder[i]

always starts with the character

'/'

.

Each folder name is

unique

.

Code Snippets

C++:

```
class Solution {
public:
vector<string> removeSubfolders(vector<string>& folder) {

}
};
```

Java:

```
class Solution {
public List<String> removeSubfolders(String[] folder) {

}
}
```

Python3:

```
class Solution:
def removeSubfolders(self, folder: List[str]) -> List[str]:
```

Python:

```
class Solution(object):
def removeSubfolders(self, folder):
"""
:type folder: List[str]
:rtype: List[str]
"""
```

JavaScript:

```
/**
* @param {string[]} folder
* @return {string[]}
*/
var removeSubfolders = function(folder) {

};
```

TypeScript:

```
function removeSubfolders(folder: string[]): string[] {

};
```

C#:

```
public class Solution {
public IList<string> RemoveSubfolders(string[] folder) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
char** removeSubfolders(char** folder, int folderSize, int* returnSize) {

}
```

Go:

```
func removeSubfolders(folder []string) []string {

}
```

Kotlin:

```
class Solution {
fun removeSubfolders(folder: Array<String>): List<String> {

}
```

```
}
```

Swift:

```
class Solution {
func removeSubfolders(_ folder: [String]) -> [String] {

}
}
```

Rust:

```
impl Solution {
pub fn remove_subfolders(folder: Vec<String>) -> Vec<String> {

}
}
```

Ruby:

```
# @param {String[]} folder
# @return {String[]}
def remove_subfolders(folder)

end
```

PHP:

```
class Solution {

/**
* @param String[] $folder
* @return String[]
*/
function removeSubfolders($folder) {

}
}
```

Dart:

```
class Solution {
List<String> removeSubfolders(List<String> folder) {

}
}
```

Scala:

```
object Solution {
def removeSubfolders(folder: Array[String]): List[String] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec remove_subfolders(folder :: [String.t]) :: [String.t]
def remove_subfolders(folder) do

end
end
```

Erlang:

```
-spec remove_subfolders(Folder :: [unicode:unicode_binary()]) ->
[unicode:unicode_binary()].
remove_subfolders(Folder) ->
.
```

Racket:

```
(define/contract (remove-subfolders folder)
(-> (listof string?) (listof string?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Remove Sub-Folders from the Filesystem
* Difficulty: Medium
* Tags: array, string, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
vector<string> removeSubfolders(vector<string>& folder) {

}
};
```

Java Solution:

```
/**
* Problem: Remove Sub-Folders from the Filesystem
* Difficulty: Medium
* Tags: array, string, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public List<String> removeSubfolders(String[] folder) {

}
}
```

Python3 Solution:

```
"""
Problem: Remove Sub-Folders from the Filesystem
Difficulty: Medium
Tags: array, string, search
```

```
Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def removeSubfolders(self, folder: List[str]) -> List[str]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def removeSubfolders(self, folder):
"""
:type folder: List[str]
:rtype: List[str]
"""
```

JavaScript Solution:

```
/**
* Problem: Remove Sub-Folders from the Filesystem
* Difficulty: Medium
* Tags: array, string, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string[]} folder
* @return {string[]}
*/
var removeSubfolders = function(folder) {

};
```

TypeScript Solution:

```
/**
* Problem: Remove Sub-Folders from the Filesystem
* Difficulty: Medium
* Tags: array, string, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function removeSubfolders(folder: string[]): string[] {

};
```

C# Solution:

```
/*
* Problem: Remove Sub-Folders from the Filesystem
* Difficulty: Medium
* Tags: array, string, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public IList<string> RemoveSubfolders(string[] folder) {

}
}
```

C Solution:

```
/*
* Problem: Remove Sub-Folders from the Filesystem
* Difficulty: Medium
* Tags: array, string, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
char** removeSubfolders(char** folder, int folderSize, int* returnSize) {

}
```

Go Solution:

```
// Problem: Remove Sub-Folders from the Filesystem
// Difficulty: Medium
// Tags: array, string, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func removeSubfolders(folder []string) []string {

}
```

Kotlin Solution:

```
class Solution {
fun removeSubfolders(folder: Array<String>): List<String> {

}
}
```

Swift Solution:

```
class Solution {
func removeSubfolders(_ folder: [String]) -> [String] {

}
}
```

Rust Solution:

```
// Problem: Remove Sub-Folders from the Filesystem
// Difficulty: Medium
// Tags: array, string, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn remove_subfolders(folder: Vec<String>) -> Vec<String> {

}
}
```

Ruby Solution:

```
# @param {String[]} folder
# @return {String[]}
def remove_subfolders(folder)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $folder
* @return String[]
*/
function removeSubfolders($folder) {

}
}
```

Dart Solution:

```
class Solution {
List<String> removeSubfolders(List<String> folder) {

}
}
```

Scala Solution:

```
object Solution {
def removeSubfolders(folder: Array[String]): List[String] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec remove_subfolders(folder :: [String.t]) :: [String.t]
def remove_subfolders(folder) do

end
end
```

Erlang Solution:

```
-spec remove_subfolders(Folder :: [unicode:unicode_binary()]) ->
[unicode:unicode_binary()].
remove_subfolders(Folder) ->
.
```

Racket Solution:

```
(define/contract (remove-subfolders folder)
(-> (listof string?) (listof string?))
)
```