Problem 1452: People Whose List of Favorite Companies Is Not a Subset of Another List

Problem Information

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

Problem Description

Given the array

favoriteCompanies

where

favoriteCompanies[i]

is the list of favorites companies for the

ith

person (

indexed from 0

).

Return the indices of people whose list of favorite companies is not a

subset

of any other list of favorites companies

. You must return the indices in increasing order.

Example 1:

Input:

favoriteCompanies = [["leetcode","google","facebook"],["google","microsoft"],["google","facebo
ok"],["google"],["amazon"]]

Output:

[0,1,4]

Explanation:

Person with index=2 has favoriteCompanies[2]=["google","facebook"] which is a subset of favoriteCompanies[0]=["leetcode","google","facebook"] corresponding to the person with index 0. Person with index=3 has favoriteCompanies[3]=["google"] which is a subset of favoriteCompanies[0]=["leetcode","google","facebook"] and favoriteCompanies[1]=["google","microsoft"]. Other lists of favorite companies are not a subset of another list, therefore, the answer is [0,1,4].

Example 2:

Input:

favoriteCompanies =
[["leetcode","google","facebook"],["leetcode","amazon"],["facebook","google"]]

Output:

[0,1]

Explanation:

In this case favoriteCompanies[2]=["facebook","google"] is a subset of favoriteCompanies[0]=["leetcode","google","facebook"], therefore, the answer is [0,1].

Example 3:

Input:

favoriteCompanies = [["leetcode"],["google"],["facebook"],["amazon"]]

Output:

[0,1,2,3]

Constraints:

1 <= favoriteCompanies.length <= 100

1 <= favoriteCompanies[i].length <= 500

1 <= favoriteCompanies[i][j].length <= 20

All strings in

favoriteCompanies[i]

are

distinct

.

All lists of favorite companies are

distinct

, that is, If we sort alphabetically each list then

favoriteCompanies[i] != favoriteCompanies[j].

All strings consist of lowercase English letters only.

Code Snippets

C++:

```
class Solution {
public:
vector<int> peopleIndexes(vector<vector<string>>& favoriteCompanies) {

}
};
```

Java:

```
class Solution {
public List<Integer> peopleIndexes(List<List<String>> favoriteCompanies) {

}
}
```

Python3:

```
class Solution:
def peopleIndexes(self, favoriteCompanies: List[List[str]]) -> List[int]:
```

Python:

```
class Solution(object):
def peopleIndexes(self, favoriteCompanies):
"""
:type favoriteCompanies: List[List[str]]
:rtype: List[int]
"""
```

JavaScript:

```
/**
* @param {string[][]} favoriteCompanies
* @return {number[]}
*/
var peopleIndexes = function(favoriteCompanies) {

};
```

TypeScript:

```
function peopleIndexes(favoriteCompanies: string[][]): number[] {
};
```

C#:

```
public class Solution {
public IList<int> PeopleIndexes(IList<IList<string>> favoriteCompanies) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* peopleIndexes(char*** favoriteCompanies, int favoriteCompaniesSize, int*
favoriteCompaniesColSize, int* returnSize) {

}
```

Go:

```
func peopleIndexes(favoriteCompanies [][]string) []int {

}
```

Kotlin:

```
class Solution {
fun peopleIndexes(favoriteCompanies: List<List<String>>): List<Int> {

}
}
```

Swift:

```
class Solution {
func peopleIndexes(_ favoriteCompanies: [[String]]) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn people_indexes(favorite_companies: Vec<Vec<String>>) -> Vec<i32> {

}
}
```

Ruby:

```
# @param {String[][]} favorite_companies
# @return {Integer[]}
def people_indexes(favorite_companies)

end
```

PHP:

```
class Solution {

/**
* @param String[][] $favoriteCompanies
* @return Integer[]
*/
function peopleIndexes($favoriteCompanies) {

}
}
```

Dart:

```
class Solution {
List<int> peopleIndexes(List<List<String>> favoriteCompanies) {

}
}
```

Scala:

```
object Solution {
def peopleIndexes(favoriteCompanies: List[List[String]]): List[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec people_indexes(favorite_companies :: [[String.t]]) :: [integer]
def people_indexes(favorite_companies) do

end
end
```

Erlang:

```
-spec people_indexes(FavoriteCompanies :: [[unicode:unicode_binary()]]) ->
[integer()].
people_indexes(FavoriteCompanies) ->
.
```

Racket:

```
(define/contract (people-indexes favoriteCompanies)
(-> (listof (listof string?)) (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: People Whose List of Favorite Companies Is Not a Subset of Another
List
* Difficulty: Medium
* Tags: array, string, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
```

```
public:
vector<int> peopleIndexes(vector<vector<string>>& favoriteCompanies) {

}
};
```

Java Solution:

```
/**
* Problem: People Whose List of Favorite Companies Is Not a Subset of Another
List
* Difficulty: Medium
* Tags: array, string, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public List<Integer> peopleIndexes(List<List<String>> favoriteCompanies) {

}
}
```

Python3 Solution:

```
"""
Problem: People Whose List of Favorite Companies Is Not a Subset of Another
List
Difficulty: Medium
Tags: array, string, hash, sort

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def peopleIndexes(self, favoriteCompanies: List[List[str]]) -> List[int]:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

```
class Solution(object):
def peopleIndexes(self, favoriteCompanies):
"""
:type favoriteCompanies: List[List[str]]
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: People Whose List of Favorite Companies Is Not a Subset of Another
List
* Difficulty: Medium
* Tags: array, string, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string[][]} favoriteCompanies
* @return {number[]}
*/
var peopleIndexes = function(favoriteCompanies) {

};
```

TypeScript Solution:

```
/**
* Problem: People Whose List of Favorite Companies Is Not a Subset of Another
List
* Difficulty: Medium
* Tags: array, string, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function peopleIndexes(favoriteCompanies: string[][]): number[] {

};
```

C# Solution:

```
/*
* Problem: People Whose List of Favorite Companies Is Not a Subset of Another
List
* Difficulty: Medium
* Tags: array, string, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public IList<int> PeopleIndexes(IList<IList<string>> favoriteCompanies) {

}
}
```

C Solution:

```
/*
* Problem: People Whose List of Favorite Companies Is Not a Subset of Another
List
* Difficulty: Medium
* Tags: array, string, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
```

```
* Note: The returned array must be malloced, assume caller calls free().
*/
int* peopleIndexes(char*** favoriteCompanies, int favoriteCompaniesSize, int*
favoriteCompaniesColSize, int* returnSize) {

}
```

Go Solution:

```
// Problem: People Whose List of Favorite Companies Is Not a Subset of
Another List
// Difficulty: Medium
// Tags: array, string, hash, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func peopleIndexes(favoriteCompanies [][]string) []int {

}
```

Kotlin Solution:

```
class Solution {
fun peopleIndexes(favoriteCompanies: List<List<String>>): List<Int> {

}
}
```

Swift Solution:

```
class Solution {
func peopleIndexes(_ favoriteCompanies: [[String]]) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: People Whose List of Favorite Companies Is Not a Subset of
Another List
// Difficulty: Medium
// Tags: array, string, hash, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn people_indexes(favorite_companies: Vec<Vec<String>>) -> Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {String[][]} favorite_companies
# @return {Integer[]}
def people_indexes(favorite_companies)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[][] $favoriteCompanies
* @return Integer[]
*/
function peopleIndexes($favoriteCompanies) {

}
}
```

Dart Solution:

```
class Solution {
List<int> peopleIndexes(List<List<String>> favoriteCompanies) {

}

}
```

Scala Solution:

```
object Solution {
def peopleIndexes(favoriteCompanies: List[List[String]]): List[Int] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec people_indexes(favorite_companies :: [[String.t]]) :: [integer]
def people_indexes(favorite_companies) do

end
end
```

Erlang Solution:

```
-spec people_indexes(FavoriteCompanies :: [[unicode:unicode_binary()]]) ->
[integer()].
people_indexes(FavoriteCompanies) ->
.
```

Racket Solution:

```
(define/contract (people-indexes favoriteCompanies)
(-> (listof (listof string?)) (listof exact-integer?))
)
```