Problem 990: Satisfiability of Equality Equations

Problem Information

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

Problem Description

You are given an array of strings

equations

that represent relationships between variables where each string

equations[i]

is of length

4

and takes one of two different forms:

"x

i

==y

i

"

or

"x

i

!=y

i

"

.Here,

x

i

and

y

i

are lowercase letters (not necessarily different) that represent one-letter variable names.

Return

true

if it is possible to assign integers to variable names so as to satisfy all the given equations, or

false

otherwise

.

Example 1:

Input:

equations = ["a==b","b!=a"]

Output:

false

Explanation:

If we assign say, a = 1 and b = 1, then the first equation is satisfied, but not the second. There is no way to assign the variables to satisfy both equations.

Example 2:

Input:

equations = ["b==a","a==b"]

Output:

true

Explanation:

We could assign a = 1 and b = 1 to satisfy both equations.

Constraints:

1 <= equations.length <= 500

equations[i].length == 4

equations[i][0]

is a lowercase letter.

equations[i][1]

is either

'='

or

'!'

.

equations[i][2]

is

'='

.

equations[i][3]

is a lowercase letter.

Code Snippets

C++:

```
class Solution {
public:
bool equationsPossible(vector<string>& equations) {

}
};
```

Java:

```
class Solution {
public boolean equationsPossible(String[] equations) {

}
```

```
}
```

Python3:

```
class Solution:
def equationsPossible(self, equations: List[str]) -> bool:
```

Python:

```
class Solution(object):
def equationsPossible(self, equations):
"""
:type equations: List[str]
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {string[]} equations
* @return {boolean}
*/
var equationsPossible = function(equations) {

};
```

TypeScript:

```
function equationsPossible(equations: string[]): boolean {

};
```

C#:

```
public class Solution {
public bool EquationsPossible(string[] equations) {

}
}
```

C:

```
bool equationsPossible(char** equations, int equationsSize) {

}
```

Go:

```
func equationsPossible(equations []string) bool {

}
```

Kotlin:

```
class Solution {
fun equationsPossible(equations: Array<String>): Boolean {

}
}
```

Swift:

```
class Solution {
func equationsPossible(_ equations: [String]) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn equations_possible(equations: Vec<String>) -> bool {

}
}
```

Ruby:

```
# @param {String[]} equations
# @return {Boolean}
def equations_possible(equations)

end
```

PHP:

```
class Solution {

/**
* @param String[] $equations
* @return Boolean
*/
function equationsPossible($equations) {

}
}
```

Dart:

```
class Solution {
bool equationsPossible(List<String> equations) {

}
}
```

Scala:

```
object Solution {
def equationsPossible(equations: Array[String]): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec equations_possible(equations :: [String.t]) :: boolean
def equations_possible(equations) do

end
end
```

Erlang:

```
-spec equations_possible(Equations :: [unicode:unicode_binary()]) ->
boolean().
equations_possible(Equations) ->
.
```

Racket:

```
(define/contract (equations-possible equations)
(-> (listof string?) boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Satisfiability of Equality Equations
* Difficulty: Medium
* Tags: array, string, graph
*
* 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:
bool equationsPossible(vector<string>& equations) {

}
};
```

Java Solution:

```
/**
* Problem: Satisfiability of Equality Equations
* Difficulty: Medium
* Tags: array, string, graph
*
* 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 boolean equationsPossible(String[] equations) {
```

```
}
}
```

Python3 Solution:

```
"""
Problem: Satisfiability of Equality Equations
Difficulty: Medium
Tags: array, string, graph

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 equationsPossible(self, equations: List[str]) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def equationsPossible(self, equations):
"""
:type equations: List[str]
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Satisfiability of Equality Equations
* Difficulty: Medium
* Tags: array, string, graph
*
* 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[]} equations
* @return {boolean}
*/
var equationsPossible = function(equations) {

};
```

TypeScript Solution:

```
/**
* Problem: Satisfiability of Equality Equations
* Difficulty: Medium
* Tags: array, string, graph
*
* 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 equationsPossible(equations: string[]): boolean {

};
```

C# Solution:

```
/*
* Problem: Satisfiability of Equality Equations
* Difficulty: Medium
* Tags: array, string, graph
*
* 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 bool EquationsPossible(string[] equations) {

}
```

```
}
```

C Solution:

```
/*
* Problem: Satisfiability of Equality Equations
* Difficulty: Medium
* Tags: array, string, graph
*
* 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
*/

bool equationsPossible(char** equations, int equationsSize) {

}
```

Go Solution:

```
// Problem: Satisfiability of Equality Equations
// Difficulty: Medium
// Tags: array, string, graph
//
// 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 equationsPossible(equations []string) bool {

}
```

Kotlin Solution:

```
class Solution {
fun equationsPossible(equations: Array<String>): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func equationsPossible(_ equations: [String]) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Satisfiability of Equality Equations
// Difficulty: Medium
// Tags: array, string, graph
//
// 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 equations_possible(equations: Vec<String>) -> bool {

}
}
```

Ruby Solution:

```
# @param {String[]} equations
# @return {Boolean}
def equations_possible(equations)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $equations
* @return Boolean
*/
function equationsPossible($equations) {

}
}
```

Dart Solution:

```
class Solution {
bool equationsPossible(List<String> equations) {

}
}
```

Scala Solution:

```
object Solution {
def equationsPossible(equations: Array[String]): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec equations_possible(equations :: [String.t]) :: boolean
def equations_possible(equations) do

end
end
```

Erlang Solution:

```
-spec equations_possible(Equations :: [unicode:unicode_binary()]) ->
boolean().
equations_possible(Equations) ->
.
```

Racket Solution:

```
(define/contract (equations-possible equations)
(-> (listof string?) boolean?)
)
```