Problem 3491: Phone Number Prefix

Problem Information

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

Problem Description

You are given a string array

numbers

that represents phone numbers. Return

true

if no phone number is a prefix of any other phone number; otherwise, return

false

.

Example 1:

Input:

numbers = ["1","2","4","3"]

Output:

true

Explanation:

No number is a prefix of another number, so the output is

true

.

Example 2:

Input:

numbers = ["001","007","15","00153"]

Output:

false

Explanation:

The string

"001"

is a prefix of the string

"00153"

. Thus, the output is

false

.

Constraints:

2 <= numbers.length <= 50

1 <= numbers[i].length <= 50

All numbers contain only digits

'0'

to

'9'

.

Code Snippets

C++:

```
class Solution {
public:
bool phonePrefix(vector<string>& numbers) {

}
};
```

Java:

```
class Solution {
public boolean phonePrefix(String[] numbers) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function phonePrefix(numbers: string[]): boolean {

};
```

C#:

```
public class Solution {
public bool PhonePrefix(string[] numbers) {

}
}
```

C:

```
bool phonePrefix(char** numbers, int numbersSize) {

}
```

Go:

```
func phonePrefix(numbers []string) bool {

}
```

Kotlin:

```
class Solution {
fun phonePrefix(numbers: Array<String>): Boolean {

}
}
```

Swift:

```
class Solution {
func phonePrefix(_ numbers: [String]) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn phone_prefix(numbers: Vec<String>) -> bool {

}
}
```

Ruby:

```
# @param {String[]} numbers
# @return {Boolean}
def phone_prefix(numbers)

end
```

PHP:

```
class Solution {

/**
* @param String[] $numbers
* @return Boolean
*/
function phonePrefix($numbers) {

}
}
```

Dart:

```
class Solution {
bool phonePrefix(List<String> numbers) {

}
}
```

Scala:

```
object Solution {
def phonePrefix(numbers: Array[String]): Boolean = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec phone_prefix(Numbers :: [unicode:unicode_binary()]) -> boolean().
phone_prefix(Numbers) ->
.
```

Racket:

```
(define/contract (phone-prefix numbers)
(-> (listof string?) boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Phone Number Prefix
* Difficulty: Easy
* Tags: array, string, sort
*
```

```
* 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 phonePrefix(vector<string>& numbers) {

}
};
```

Java Solution:

```
/**
* Problem: Phone Number Prefix
* Difficulty: Easy
* Tags: array, string, sort
*
* 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 phonePrefix(String[] numbers) {

}
}
```

Python3 Solution:

```
"""
Problem: Phone Number Prefix
Difficulty: Easy
Tags: array, string, sort

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Phone Number Prefix
* Difficulty: Easy
* Tags: array, string, sort
*
* 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[]} numbers
* @return {boolean}
*/
var phonePrefix = function(numbers) {

};
```

TypeScript Solution:

```
/**
* Problem: Phone Number Prefix
* Difficulty: Easy
* Tags: array, string, sort
```

```
*
* 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 phonePrefix(numbers: string[]): boolean {

};
```

C# Solution:

```
/*
* Problem: Phone Number Prefix
* Difficulty: Easy
* Tags: array, string, sort
*
* 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 PhonePrefix(string[] numbers) {

}
}
```

C Solution:

```
/*
* Problem: Phone Number Prefix
* Difficulty: Easy
* Tags: array, string, sort
*
* 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 phonePrefix(char** numbers, int numbersSize) {
}
```

Go Solution:

```
// Problem: Phone Number Prefix
// Difficulty: Easy
// Tags: array, string, sort
//
// 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 phonePrefix(numbers []string) bool {

}
```

Kotlin Solution:

```
class Solution {
fun phonePrefix(numbers: Array<String>): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func phonePrefix(_ numbers: [String]) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Phone Number Prefix
// Difficulty: Easy
// Tags: array, string, sort
//
// 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 phone_prefix(numbers: Vec<String>) -> bool {

}
}
```

Ruby Solution:

```
# @param {String[]} numbers
# @return {Boolean}
def phone_prefix(numbers)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $numbers
* @return Boolean
*/
function phonePrefix($numbers) {

}
}
```

Dart Solution:

```
class Solution {
bool phonePrefix(List<String> numbers) {

}
}
```

Scala Solution:

```
object Solution {
def phonePrefix(numbers: Array[String]): Boolean = {



}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec phone_prefix(Numbers :: [unicode:unicode_binary()]) -> boolean().
phone_prefix(Numbers) ->
.
```

Racket Solution:

```
(define/contract (phone-prefix numbers)
(-> (listof string?) boolean?)
)
```