Problem 3295: Report Spam Message

Problem Information

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

Problem Description

You are given an array of strings

message

and an array of strings

bannedWords

.

An array of words is considered

spam

if there are

at least

two words in it that

exactly

match any word in

bannedWords

.

Return

true

if the array

message

is spam, and

false

otherwise.

Example 1:

Input:

message = ["hello","world","leetcode"], bannedWords = ["world","hello"]

Output:

true

Explanation:

The words

"hello"

and

"world"

from the

message

array both appear in the

bannedWords

array.

Example 2:

Input:

message = ["hello","programming","fun"], bannedWords = ["world","programming","leetcode"]

Output:

false

Explanation:

Only one word from the

message

array (

"programming"

) appears in the

bannedWords

array.

Constraints:

1 <= message.length, bannedWords.length <= 10

5

1 <= message[i].length, bannedWords[i].length <= 15

message[i]

and

bannedWords[i]

consist only of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
bool reportSpam(vector<string>& message, vector<string>& bannedWords) {

}
};
```

Java:

```
class Solution {
public boolean reportSpam(String[] message, String[] bannedWords) {

}
}
```

Python3:

```
class Solution:
def reportSpam(self, message: List[str], bannedWords: List[str]) -> bool:
```

Python:

```
class Solution(object):
def reportSpam(self, message, bannedWords):
"""
:type message: List[str]
```

```
:type bannedWords: List[str]
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {string[]} message
* @param {string[]} bannedWords
* @return {boolean}
*/
var reportSpam = function(message, bannedWords) {

};
```

TypeScript:

```
function reportSpam(message: string[], bannedWords: string[]): boolean {

};
```

C#:

```
public class Solution {
public bool ReportSpam(string[] message, string[] bannedWords) {

}
}
```

C:

```
bool reportSpam(char** message, int messageSize, char** bannedWords, int
bannedWordsSize) {

}
```

Go:

```
func reportSpam(message []string, bannedWords []string) bool {

}
```

Kotlin:

```
class Solution {
fun reportSpam(message: Array<String>, bannedWords: Array<String>): Boolean {

}
}
```

Swift:

```
class Solution {
func reportSpam(_ message: [String], _ bannedWords: [String]) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn report_spam(message: Vec<String>, banned_words: Vec<String>) -> bool {

}
}
```

Ruby:

```
# @param {String[]} message
# @param {String[]} banned_words
# @return {Boolean}
def report_spam(message, banned_words)

end
```

PHP:

```
class Solution {

/**
* @param String[] $message
* @param String[] $bannedWords
* @return Boolean
*/
function reportSpam($message, $bannedWords) {
```

```
}
}
```

Dart:

```
class Solution {
bool reportSpam(List<String> message, List<String> bannedWords) {

}
}
```

Scala:

```
object Solution {
def reportSpam(message: Array[String], bannedWords: Array[String]): Boolean =
{

}
}
```

Elixir:

```
defmodule Solution do
@spec report_spam(message :: [String.t], banned_words :: [String.t]) ::
boolean
def report_spam(message, banned_words) do

end
end
```

Erlang:

```
-spec report_spam(Message :: [unicode:unicode_binary()], BannedWords ::
[unicode:unicode_binary()]) -> boolean().
report_spam(Message, BannedWords) ->
.
```

Racket:

```
(define/contract (report-spam message bannedWords)
(-> (listof string?) (listof string?) boolean?)
```

```
)
```

Solutions

C++ Solution:

```
/*
* Problem: Report Spam Message
* Difficulty: Medium
* Tags: array, string, hash
*
* 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:
bool reportSpam(vector<string>& message, vector<string>& bannedWords) {

}
};
```

Java Solution:

```
/**
* Problem: Report Spam Message
* Difficulty: Medium
* Tags: array, string, hash
*
* 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 boolean reportSpam(String[] message, String[] bannedWords) {

}
}
```

Python3 Solution:

```
"""
Problem: Report Spam Message
Difficulty: Medium
Tags: array, string, hash

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 reportSpam(self, message: List[str], bannedWords: List[str]) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def reportSpam(self, message, bannedWords):
"""
:type message: List[str]
:type bannedWords: List[str]
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Report Spam Message
* Difficulty: Medium
* Tags: array, string, hash
*
* 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[]} message
* @param {string[]} bannedWords
```

```
 * @return {boolean}
 */
var reportSpam = function(message, bannedWords) {

};
```

TypeScript Solution:

```
/**
* Problem: Report Spam Message
* Difficulty: Medium
* Tags: array, string, hash
*
* 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 reportSpam(message: string[], bannedWords: string[]): boolean {

};
```

C# Solution:

```
/*
* Problem: Report Spam Message
* Difficulty: Medium
* Tags: array, string, hash
*
* 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 bool ReportSpam(string[] message, string[] bannedWords) {

}
}
```

C Solution:

```
/*
* Problem: Report Spam Message
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

bool reportSpam(char** message, int messageSize, char** bannedWords, int
bannedWordsSize) {

}
```

Go Solution:

```
// Problem: Report Spam Message
// Difficulty: Medium
// Tags: array, string, hash
//
// 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 reportSpam(message []string, bannedWords []string) bool {

}
```

Kotlin Solution:

```
class Solution {
fun reportSpam(message: Array<String>, bannedWords: Array<String>): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func reportSpam(_ message: [String], _ bannedWords: [String]) -> Bool {
```

```
}
}
```

Rust Solution:

```
// Problem: Report Spam Message
// Difficulty: Medium
// Tags: array, string, hash
//
// 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 report_spam(message: Vec<String>, banned_words: Vec<String>) -> bool {

}
}
```

Ruby Solution:

```
# @param {String[]} message
# @param {String[]} banned_words
# @return {Boolean}
def report_spam(message, banned_words)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $message
* @param String[] $bannedWords
* @return Boolean
*/
function reportSpam($message, $bannedWords) {

}
}
```

Dart Solution:

```
class Solution {
bool reportSpam(List<String> message, List<String> bannedWords) {

}
}
```

Scala Solution:

```
object Solution {
def reportSpam(message: Array[String], bannedWords: Array[String]): Boolean =
{

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec report_spam(message :: [String.t], banned_words :: [String.t]) ::
boolean
def report_spam(message, banned_words) do

end
end
```

Erlang Solution:

```
-spec report_spam(Message :: [unicode:unicode_binary()], BannedWords ::
[unicode:unicode_binary()]) -> boolean().
report_spam(Message, BannedWords) ->
.
```

Racket Solution:

```
(define/contract (report-spam message bannedWords)
(-> (listof string?) (listof string?) boolean?)
)
```