Problem 2023: Number of Pairs of Strings With Concatenation Equal to Target

Problem Information

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

Problem Description

Given an array of

digit

strings

nums

and a

digit

string

target

, return

the number of pairs of indices

(i, j)

(where

i != j

) such that the

concatenation

of

nums[i] + nums[j]

equals

target

.

Example 1:

Input:

nums = ["777","7","77","77"], target = "7777"

Output:

4

Explanation:

Valid pairs are: - (0, 1): "777" + "7" - (1, 0): "7" + "777" - (2, 3): "77" + "77" - (3, 2): "77" + "77"

Example 2:

Input:

nums = ["123","4","12","34"], target = "1234"

Output:

2

Explanation:

Valid pairs are: - (0, 1): "123" + "4" - (2, 3): "12" + "34"

Example 3:

Input:

nums = ["1","1","1"], target = "11"

Output:

6

Explanation:

Valid pairs are: - (0, 1): "1" + "1" - (1, 0): "1" + "1" - (0, 2): "1" + "1" - (2, 0): "1" + "1" - (1, 2): "1" + "1" - (2, 1): "1" + "1"

Constraints:

2 <= nums.length <= 100

1 <= nums[i].length <= 100

2 <= target.length <= 100

nums[i]

and

target

consist of digits.

nums[i]

and

target

do not have leading zeros.

Code Snippets

C++:

```
class Solution {
public:
int numOfPairs(vector<string>& nums, string target) {

}
};
```

Java:

```
class Solution {
public int numOfPairs(String[] nums, String target) {

}
}
```

Python3:

```
class Solution:
def numOfPairs(self, nums: List[str], target: str) -> int:
```

Python:

```
class Solution(object):
def numOfPairs(self, nums, target):
"""
:type nums: List[str]
:type target: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string[]} nums
* @param {string} target
* @return {number}
*/
var numOfPairs = function(nums, target) {

};
```

TypeScript:

```
function numOfPairs(nums: string[], target: string): number {

};
```

C#:

```
public class Solution {
public int NumOfPairs(string[] nums, string target) {

}
}
```

C:

```
int numOfPairs(char** nums, int numsSize, char* target) {

}
```

Go:

```
func numOfPairs(nums []string, target string) int {

}
```

Kotlin:

```
class Solution {
fun numOfPairs(nums: Array<String>, target: String): Int {

}
}
```

Swift:

```
class Solution {
func numOfPairs(_ nums: [String], _ target: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn num_of_pairs(nums: Vec<String>, target: String) -> i32 {

}
}
```

Ruby:

```
# @param {String[]} nums
# @param {String} target
# @return {Integer}
def num_of_pairs(nums, target)

end
```

PHP:

```
class Solution {

/**
* @param String[] $nums
* @param String $target
* @return Integer
*/
function numOfPairs($nums, $target) {

}
}
```

Dart:

```
class Solution {
int numOfPairs(List<String> nums, String target) {
```

```
}
}
```

Scala:

```
object Solution {
def numOfPairs(nums: Array[String], target: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec num_of_pairs(nums :: [String.t], target :: String.t) :: integer
def num_of_pairs(nums, target) do

end
end
```

Erlang:

```
-spec num_of_pairs(Nums :: [unicode:unicode_binary()], Target ::
unicode:unicode_binary()) -> integer().
num_of_pairs(Nums, Target) ->
.
```

Racket:

```
(define/contract (num-of-pairs nums target)
(-> (listof string?) string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Pairs of Strings With Concatenation Equal to Target
```

```
* 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:
int numOfPairs(vector<string>& nums, string target) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Pairs of Strings With Concatenation Equal to Target
* 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 int numOfPairs(String[] nums, String target) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Pairs of Strings With Concatenation Equal to Target
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 numOfPairs(self, nums: List[str], target: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def numOfPairs(self, nums, target):
"""
:type nums: List[str]
:type target: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Pairs of Strings With Concatenation Equal to Target
* 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[]} nums
* @param {string} target
* @return {number}
*/
var numOfPairs = function(nums, target) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Pairs of Strings With Concatenation Equal to Target
* 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 numOfPairs(nums: string[], target: string): number {

};
```

C# Solution:

```
/*
* Problem: Number of Pairs of Strings With Concatenation Equal to Target
* 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 int NumOfPairs(string[] nums, string target) {

}
}
```

C Solution:

```
/*
* Problem: Number of Pairs of Strings With Concatenation Equal to Target
* 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
```

```
*/

int numOfPairs(char** nums, int numsSize, char* target) {

}
```

Go Solution:

```
// Problem: Number of Pairs of Strings With Concatenation Equal to Target
// 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 numOfPairs(nums []string, target string) int {

}
```

Kotlin Solution:

```
class Solution {
fun numOfPairs(nums: Array<String>, target: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func numOfPairs(_ nums: [String], _ target: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Number of Pairs of Strings With Concatenation Equal to Target
// 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 num_of_pairs(nums: Vec<String>, target: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String[]} nums
# @param {String} target
# @return {Integer}
def num_of_pairs(nums, target)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $nums
* @param String $target
* @return Integer
*/
function numOfPairs($nums, $target) {

}
}
```

Dart Solution:

```
class Solution {
int numOfPairs(List<String> nums, String target) {

}
}
```

Scala Solution:

```
object Solution {
def numOfPairs(nums: Array[String], target: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec num_of_pairs(nums :: [String.t], target :: String.t) :: integer
def num_of_pairs(nums, target) do

end
end
```

Erlang Solution:

```
-spec num_of_pairs(Nums :: [unicode:unicode_binary()], Target ::
unicode:unicode_binary()) -> integer().
num_of_pairs(Nums, Target) ->
.
```

Racket Solution:

```
(define/contract (num-of-pairs nums target)
(-> (listof string?) string? exact-integer?)
)
```