Problem 2451: Odd String Difference

Problem Information

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

Problem Description

You are given an array of equal-length strings

words

. Assume that the length of each string is

n

.

Each string

words[i]

can be converted into a

difference integer array

difference[i]

of length

n - 1

where

difference[i][j] = words[i][j+1] - words[i][j]

where

0 <= j <= n - 2

. Note that the difference between two letters is the difference between their

positions

in the alphabet i.e. the position of

'a'

is

0

,

'b'

is

1

, and

'z'

is

25

.

For example, for the string

"acb"

, the difference integer array is

[2 - 0, 1 - 2] = [2, -1]

.

All the strings in words have the same difference integer array,

except one

. You should find that string.

Return

the string in

words

that has different

difference integer array

.

Example 1:

Input:

words = ["adc","wzy","abc"]

Output:

"abc"

Explanation:

- The difference integer array of "adc" is [3 - 0, 2 - 3] = [3, -1]. - The difference integer array of "wzy" is [25 - 22, 24 - 25]= [3, -1]. - The difference integer array of "abc" is [1 - 0, 2 - 1] = [1, 1]. The odd array out is [1, 1], so we return the corresponding string, "abc".

Example 2:

Input:

words = ["aaa","bob","ccc","ddd"]

Output:

"bob"

Explanation:

All the integer arrays are [0, 0] except for "bob", which corresponds to [13, -13].

Constraints:

3 <= words.length <= 100

n == words[i].length

2 <= n <= 20

words[i]

consists of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
string oddString(vector<string>& words) {

}
};
```

Java:

```
class Solution {
public String oddString(String[] words) {

}
}
```

Python3:

```
class Solution:
def oddString(self, words: List[str]) -> str:
```

Python:

```
class Solution(object):
def oddString(self, words):
"""
:type words: List[str]
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string[]} words
* @return {string}
*/
var oddString = function(words) {

};
```

TypeScript:

```
function oddString(words: string[]): string {

};
```

C#:

```
public class Solution {
public string OddString(string[] words) {

}
}
```

C:

```
char* oddString(char** words, int wordsSize) {

}
```

Go:

```
func oddString(words []string) string {

}
```

Kotlin:

```
class Solution {
fun oddString(words: Array<String>): String {

}
}
```

Swift:

```
class Solution {
func oddString(_ words: [String]) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn odd_string(words: Vec<String>) -> String {

}
}
```

Ruby:

```
# @param {String[]} words
# @return {String}
def odd_string(words)

end
```

PHP:

```
class Solution {

/**
* @param String[] $words
* @return String
*/
function oddString($words) {

}
}
```

Dart:

```
class Solution {
String oddString(List<String> words) {

}
}
```

Scala:

```
object Solution {
def oddString(words: Array[String]): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec odd_string(words :: [String.t]) :: String.t
def odd_string(words) do

end
end
```

Erlang:

```
-spec odd_string(Words :: [unicode:unicode_binary()]) ->
unicode:unicode_binary().
odd_string(Words) ->
```

```
.
```

Racket:

```
(define/contract (odd-string words)
(-> (listof string?) string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Odd String Difference
* Difficulty: Easy
* 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:
string oddString(vector<string>& words) {

}
};
```

Java Solution:

```
/**
* Problem: Odd String Difference
* Difficulty: Easy
* 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 String oddString(String[] words) {

}
}
```

Python3 Solution:

```
"""
Problem: Odd String Difference
Difficulty: Easy
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 oddString(self, words: List[str]) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def oddString(self, words):
"""
:type words: List[str]
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Odd String Difference
* Difficulty: Easy
* 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[]} words
* @return {string}
*/
var oddString = function(words) {

};
```

TypeScript Solution:

```
/**
* Problem: Odd String Difference
* Difficulty: Easy
* 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 oddString(words: string[]): string {

};
```

C# Solution:

```
/*
* Problem: Odd String Difference
* Difficulty: Easy
* 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 string OddString(string[] words) {

}
}
```

C Solution:

```
/*
* Problem: Odd String Difference
* Difficulty: Easy
* 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
*/

char* oddString(char** words, int wordsSize) {

}
```

Go Solution:

```
// Problem: Odd String Difference
// Difficulty: Easy
// 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 oddString(words []string) string {

}
```

Kotlin Solution:

```
class Solution {
fun oddString(words: Array<String>): String {

}
```

```
}
```

Swift Solution:

```
class Solution {
func oddString(_ words: [String]) -> String {

}
}
```

Rust Solution:

```
// Problem: Odd String Difference
// Difficulty: Easy
// 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 odd_string(words: Vec<String>) -> String {

}
}
```

Ruby Solution:

```
# @param {String[]} words
# @return {String}
def odd_string(words)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $words
* @return String
```

```
*/
function oddString($words) {

}
}
```

Dart Solution:

```
class Solution {
String oddString(List<String> words) {

}
}
```

Scala Solution:

```
object Solution {
def oddString(words: Array[String]): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec odd_string(words :: [String.t]) :: String.t
def odd_string(words) do

end
end
```

Erlang Solution:

```
-spec odd_string(Words :: [unicode:unicode_binary()]) ->
unicode:unicode_binary().
odd_string(Words) ->
.
```

Racket Solution:

```
(define/contract (odd-string words)
(-> (listof string?) string?)
)
```