Problem 1927: Sum Game

Problem Information

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

Problem Description

Alice and Bob take turns playing a game, with

Alice

starting first

.

You are given a string

num

of

even length

consisting of digits and

'?'

characters. On each turn, a player will do the following if there is still at least one

'?'

in

num

:

Choose an index

i

where

num[i] == '?'

.

Replace

num[i]

with any digit between

'0'

and

'9'

.

The game ends when there are no more

'?'

characters in

num

.

For Bob to win, the sum of the digits in the first half of num

must be

equal

to the sum of the digits in the second half. For Alice to win, the sums must

not be equal

.

For example, if the game ended with

num = "243801"

, then Bob wins because

2+4+3 = 8+0+1

. If the game ended with

num = "243803"

, then Alice wins because

2+4+3 != 8+0+3

.

Assuming Alice and Bob play

optimally

, return

true

if Alice will win and

false

if Bob will win

.

Example 1:

Input:

num = "5023"

Output:

false

Explanation:

There are no moves to be made. The sum of the first half is equal to the sum of the second half: 5 + 0 = 2 + 3.

Example 2:

Input:

num = "25??"

Output:

true

Explanation:

Alice can replace one of the '?'s with '9' and it will be impossible for Bob to make the sums equal.

Example 3:

Input:

num = "?3295???"

Output:

false

Explanation:

It can be proven that Bob will always win. One possible outcome is: - Alice replaces the first '?' with '9'. num = "93295???". - Bob replaces one of the '?' in the right half with '9'. num = "932959??". - Alice replaces one of the '?' in the right half with '2'. num = "9329592?". - Bob replaces the last '?' in the right half with '7'. num = "93295927". Bob wins because 9 + 3 + 2 + 9 = 5 + 9 + 2 + 7.

Constraints:

2 <= num.length <= 10

5

num.length

is

even

.

num

consists of only digits and

'?'

.

Code Snippets

C++:

```
class Solution {
public:
bool sumGame(string num) {

}
};
```

Java:

```
class Solution {
public boolean sumGame(String num) {

}
}
```

Python3:

```
class Solution:
def sumGame(self, num: str) -> bool:
```

Python:

```
class Solution(object):
def sumGame(self, num):
"""
:type num: str
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {string} num
* @return {boolean}
*/
var sumGame = function(num) {

};
```

}

TypeScript:

```
function sumGame(num: string): boolean {

};
```

C#:

```
public class Solution {
public bool SumGame(string num) {

}
}
```

C:

```
bool sumGame(char* num) {

}
```

Go:

```
func sumGame(num string) bool {

}
```

Kotlin:

```
class Solution {
fun sumGame(num: String): Boolean {

}
}
```

Swift:

```
class Solution {
func sumGame(_ num: String) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn sum_game(num: String) -> bool {

}
}
```

Ruby:

```
# @param {String} num
# @return {Boolean}
def sum_game(num)

end
```

PHP:

```
class Solution {

/**
* @param String $num
* @return Boolean
*/
function sumGame($num) {

}
}
```

Dart:

```
class Solution {
bool sumGame(String num) {

}
}
```

Scala:

```
object Solution {
def sumGame(num: String): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec sum_game(num :: String.t) :: boolean
def sum_game(num) do

end
end
```

Erlang:

```
-spec sum_game(Num :: unicode:unicode_binary()) -> boolean().
sum_game(Num) ->
.
```

Racket:

```
(define/contract (sum-game num)
(-> string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Sum Game
* Difficulty: Medium
* Tags: string, greedy, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
bool sumGame(string num) {

}
};
```

Java Solution:

```
/**
* Problem: Sum Game
* Difficulty: Medium
* Tags: string, greedy, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public boolean sumGame(String num) {

}
}
```

Python3 Solution:

```
"""
Problem: Sum Game
Difficulty: Medium
Tags: string, greedy, math

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def sumGame(self, num: str) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def sumGame(self, num):
"""
:type num: str
:rtype: bool
```

```
"""
```

JavaScript Solution:

```
/**
* Problem: Sum Game
* Difficulty: Medium
* Tags: string, greedy, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} num
* @return {boolean}
*/
var sumGame = function(num) {

};
```

TypeScript Solution:

```
/**
* Problem: Sum Game
* Difficulty: Medium
* Tags: string, greedy, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function sumGame(num: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Sum Game
* Difficulty: Medium
* Tags: string, greedy, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public bool SumGame(string num) {

}
}
```

C Solution:

```
/*
* Problem: Sum Game
* Difficulty: Medium
* Tags: string, greedy, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

bool sumGame(char* num) {

}
```

Go Solution:

```
// Problem: Sum Game
// Difficulty: Medium
// Tags: string, greedy, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach
```

```
func sumGame(num string) bool {

}
```

Kotlin Solution:

```
class Solution {
fun sumGame(num: String): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func sumGame(_ num: String) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Sum Game
// Difficulty: Medium
// Tags: string, greedy, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn sum_game(num: String) -> bool {

}
}
```

Ruby Solution:

```
# @param {String} num
# @return {Boolean}
def sum_game(num)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param String $num
* @return Boolean
*/
function sumGame($num) {

}
}
```

Dart Solution:

```
class Solution {
bool sumGame(String num) {

}
}
```

Scala Solution:

```
object Solution {
def sumGame(num: String): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec sum_game(num :: String.t) :: boolean
def sum_game(num) do

end
end
```

Erlang Solution:

```
-spec sum_game(Num :: unicode:unicode_binary()) -> boolean().
sum_game(Num) ->
.
```

Racket Solution:

```
(define/contract (sum-game num)
(-> string? boolean?)
)
```