Problem 2342: Max Sum of a Pair With Equal Sum of Digits

Problem Information

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

Problem Description

You are given a

0-indexed

array

nums

consisting of

positive

integers. You can choose two indices

i

and

j

, such that

i != j

, and the sum of digits of the number

nums[i]

is equal to that of

nums[j]

.

Return the

maximum

value of

nums[i] + nums[j]

that you can obtain over all possible indices

i

and

j

that satisfy the conditions. If no such pair of indices exists, return -1.

Example 1:

Input:

nums = [18,43,36,13,7]

Output:

54

Explanation:

The pairs (i, j) that satisfy the conditions are: - (0, 2), both numbers have a sum of digits equal to 9, and their sum is 18 + 36 = 54. - (1, 4), both numbers have a sum of digits equal to 7, and their sum is 43 + 7 = 50. So the maximum sum that we can obtain is 54.

Example 2:

Input:

nums = [10,12,19,14]

Output:

-1

Explanation:

There are no two numbers that satisfy the conditions, so we return -1.

Constraints:

1 <= nums.length <= 10

5

1 <= nums[i] <= 10

9

Code Snippets

C++:

```
class Solution {
public:
int maximumSum(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public int maximumSum(int[] nums) {

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number[]} nums
* @return {number}
*/
var maximumSum = function(nums) {

};
```

TypeScript:

```
function maximumSum(nums: number[]): number {

};
```

C#:

```
public class Solution {
public int MaximumSum(int[] nums) {
```

```
}
}
```

C:

```
int maximumSum(int* nums, int numsSize) {

}
```

Go:

```
func maximumSum(nums []int) int {

}
```

Kotlin:

```
class Solution {
fun maximumSum(nums: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func maximumSum(_ nums: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn maximum_sum(nums: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @return {Integer}
def maximum_sum(nums)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
*/
function maximumSum($nums) {

}
}
```

Dart:

```
class Solution {
int maximumSum(List<int> nums) {

}
}
```

Scala:

```
object Solution {
def maximumSum(nums: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec maximum_sum(nums :: [integer]) :: integer
def maximum_sum(nums) do

end
end
```

Erlang:

```
-spec maximum_sum(Nums :: [integer()]) -> integer().
maximum_sum(Nums) ->
.
```

Racket:

```
(define/contract (maximum-sum nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Max Sum of a Pair With Equal Sum of Digits
* Difficulty: Medium
* Tags: array, hash, sort, queue, heap
*
* 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 maximumSum(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Max Sum of a Pair With Equal Sum of Digits
* Difficulty: Medium
* Tags: array, hash, sort, queue, heap
*
* 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 maximumSum(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Max Sum of a Pair With Equal Sum of Digits
Difficulty: Medium
Tags: array, hash, sort, queue, heap

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 maximumSum(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Max Sum of a Pair With Equal Sum of Digits
* Difficulty: Medium
```

```
* Tags: array, hash, sort, queue, heap
*
* 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 {number[]} nums
* @return {number}
*/
var maximumSum = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Max Sum of a Pair With Equal Sum of Digits
* Difficulty: Medium
* Tags: array, hash, sort, queue, heap
*
* 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 maximumSum(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Max Sum of a Pair With Equal Sum of Digits
* Difficulty: Medium
* Tags: array, hash, sort, queue, heap
*
* 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 MaximumSum(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Max Sum of a Pair With Equal Sum of Digits
* Difficulty: Medium
* Tags: array, hash, sort, queue, heap
*
* 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 maximumSum(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Max Sum of a Pair With Equal Sum of Digits
// Difficulty: Medium
// Tags: array, hash, sort, queue, heap
//
// 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 maximumSum(nums []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maximumSum(nums: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func maximumSum(_ nums: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Max Sum of a Pair With Equal Sum of Digits
// Difficulty: Medium
// Tags: array, hash, sort, queue, heap
//
// 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 maximum_sum(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @return {Integer}
def maximum_sum(nums)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param Integer[] $nums
* @return Integer
*/
function maximumSum($nums) {

}
}
```

Dart Solution:

```
class Solution {
int maximumSum(List<int> nums) {

}
}
```

Scala Solution:

```
object Solution {
def maximumSum(nums: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec maximum_sum(nums :: [integer]) :: integer
def maximum_sum(nums) do

end
end
```

Erlang Solution:

```
-spec maximum_sum(Nums :: [integer()]) -> integer().
maximum_sum(Nums) ->
.
```

Racket Solution:

```
(define/contract (maximum-sum nums)
(-> (listof exact-integer?) exact-integer?)
)
```