Problem 1877: Minimize Maximum Pair Sum in Array

Problem Information

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

Problem Description

The

pair sum

of a pair

(a,b)

is equal to

a + b

. The

maximum pair sum

is the largest

pair sum

in a list of pairs.

For example, if we have pairs

(1,5)

,

(2,3)

, and

(4,4)

, the

maximum pair sum

would be

max(1+5, 2+3, 4+4) = max(6, 5, 8) = 8

.

Given an array

nums

of

even

length

n

, pair up the elements of

nums

into

n / 2

pairs such that:

Each element of

nums

is in

exactly one

pair, and

The

maximum pair sum

is

minimized

.

Return

the minimized

maximum pair sum

after optimally pairing up the elements

.

Example 1:

Input:

nums = [3,5,2,3]

Output:

7

Explanation:

The elements can be paired up into pairs (3,3) and (5,2). The maximum pair sum is max(3+3, 5+2) = max(6, 7) = 7.

Example 2:

Input:

nums = [3,5,4,2,4,6]

Output:

8

Explanation:

The elements can be paired up into pairs (3,5), (4,4), and (6,2). The maximum pair sum is max(3+5, 4+4, 6+2) = max(8, 8, 8) = 8.

Constraints:

n == nums.length

2 <= n <= 10

5

n

is

even

.

1 <= nums[i] <= 10

5

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number[]} nums
* @return {number}
*/
```

```
var minPairSum = function(nums) {

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Scala:

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

}
}
```

Racket:

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

)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimize Maximum Pair Sum in Array
* Difficulty: Medium
* Tags: array, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Minimize Maximum Pair Sum in Array
* Difficulty: Medium
* Tags: array, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
```

```
public int minPairSum(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimize Maximum Pair Sum in Array
Difficulty: Medium
Tags: array, greedy, sort

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def minPairSum(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimize Maximum Pair Sum in Array
* Difficulty: Medium
* Tags: array, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Minimize Maximum Pair Sum in Array
* Difficulty: Medium
* Tags: array, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function minPairSum(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Minimize Maximum Pair Sum in Array
* Difficulty: Medium
* Tags: array, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

```
}
}
```

C Solution:

```
/*
* Problem: Minimize Maximum Pair Sum in Array
* Difficulty: Medium
* Tags: array, greedy, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/



int minPairSum(int* nums, int numsSize){

}
```

Go Solution:

```
// Problem: Minimize Maximum Pair Sum in Array
// Difficulty: Medium
// Tags: array, greedy, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func minPairSum(nums []int) int {

}
```

Kotlin Solution:

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

}
```

```
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Minimize Maximum Pair Sum in Array
// Difficulty: Medium
// Tags: array, greedy, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
```

```
*/
function minPairSum($nums) {

}
}
```

Scala Solution:

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

}
}
```

Racket Solution:

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

)
```