

# Problem 3496: Maximize Score After Pair Deletions

## Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

## Problem Description

You are given an array of integers

nums

. You

must

repeatedly perform one of the following operations while the array has more than two elements:

Remove the first two elements.

Remove the last two elements.

Remove the first and last element.

For each operation, add the sum of the removed elements to your total score.

Return the

maximum

possible score you can achieve.

Example 1:

Input:

nums = [2,4,1]

Output:

6

Explanation:

The possible operations are:

Remove the first two elements

$$(2 + 4) = 6$$

. The remaining array is

[1]

Remove the last two elements

$$(4 + 1) = 5$$

. The remaining array is

[2]

Remove the first and last elements

$$(2 + 1) = 3$$

. The remaining array is

[4]

.  
The maximum score is obtained by removing the first two elements, resulting in a final score of 6.

Example 2:

Input:

nums = [5,-1,4,2]

Output:

7

Explanation:

The possible operations are:

Remove the first and last elements

$$(5 + 2) = 7$$

. The remaining array is

[-1, 4]

.  
Remove the first two elements

$$(5 + -1) = 4$$

. The remaining array is

[4, 2]

Remove the last two elements

$$(4 + 2) = 6$$

. The remaining array is

[5, -1]

The maximum score is obtained by removing the first and last elements, resulting in a total score of 7.

Constraints:

$$1 \leq \text{nums.length} \leq 10$$

5

-10

4

$$\leq \text{nums}[i] \leq 10$$

4

## Code Snippets

C++:

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

```
};
```

### Java:

```
class Solution {  
    public int maxScore(int[] nums) {  
        }  
        }  
}
```

### Python3:

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

### Python:

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

### JavaScript:

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

### TypeScript:

```
function maxScore(nums: number[]): number {  
};
```

### C#:

```
public class Solution {  
    public int MaxScore(int[] nums) {  
  
    }  
}
```

**C:**

```
int maxScore(int* nums, int numsSize) {  
  
}
```

**Go:**

```
func maxScore(nums []int) int {  
  
}
```

**Kotlin:**

```
class Solution {  
    fun maxScore(nums: IntArray): Int {  
  
    }  
}
```

**Swift:**

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

**Rust:**

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

**Ruby:**

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

end
```

## PHP:

```
class Solution {

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

    }
}
```

## Dart:

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

}
```

## Scala:

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

}
```

## Elixir:

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

end
end
```

### Erlang:

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

### Racket:

```
(define/contract (max-score nums)  
  (-> (listof exact-integer?) exact-integer?)  
)
```

## Solutions

### C++ Solution:

```
/*  
 * Problem: Maximize Score After Pair Deletions  
 * Difficulty: Medium  
 * Tags: array, greedy  
 *  
 * 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 maxScore(vector<int>& nums) {  
  
    }  
};
```

### Java Solution:

```
/**  
 * Problem: Maximize Score After Pair Deletions  
 * Difficulty: Medium  
 * Tags: array, greedy  
 *  
 * 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 maxScore(int[] nums) {

}
}

```

### Python3 Solution:

```

"""
Problem: Maximize Score After Pair Deletions
Difficulty: Medium
Tags: array, greedy

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

```

### Python Solution:

```

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

```

### JavaScript Solution:

```

/**
 * Problem: Maximize Score After Pair Deletions
 * Difficulty: Medium

```

```

* Tags: array, greedy
*
* 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 maxScore = function(nums) {
}

```

### TypeScript Solution:

```

/** 
* Problem: Maximize Score After Pair Deletions
* Difficulty: Medium
* Tags: array, greedy
*
* 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 maxScore(nums: number[]): number {
}

```

### C# Solution:

```

/*
* Problem: Maximize Score After Pair Deletions
* Difficulty: Medium
* Tags: array, greedy
*
* 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

```

```
*/\n\npublic class Solution {\n    public int MaxScore(int[] nums) {\n        }\n    }\n}
```

### C Solution:

```
/*\n * Problem: Maximize Score After Pair Deletions\n * Difficulty: Medium\n * Tags: array, greedy\n *\n * Approach: Use two pointers or sliding window technique\n * Time Complexity: O(n) or O(n log n)\n * Space Complexity: O(1) to O(n) depending on approach\n */\n\nint maxScore(int* nums, int numsSize) {\n\n}
```

### Go Solution:

```
// Problem: Maximize Score After Pair Deletions\n// Difficulty: Medium\n// Tags: array, greedy\n//\n// Approach: Use two pointers or sliding window technique\n// Time Complexity: O(n) or O(n log n)\n// Space Complexity: O(1) to O(n) depending on approach\n\nfunc maxScore(nums []int) int {\n\n}
```

### Kotlin Solution:

```
class Solution {  
    fun maxScore(nums: IntArray): Int {  
        }  
    }  
}
```

### Swift Solution:

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

### Rust Solution:

```
// Problem: Maximize Score After Pair Deletions  
// Difficulty: Medium  
// Tags: array, greedy  
//  
// 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 max_score(nums: Vec<i32>) -> i32 {  
        }  
    }  
}
```

### Ruby Solution:

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

### PHP Solution:

```
class Solution {
```

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

}
```

### Dart Solution:

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

}
```

### Scala Solution:

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

}
```

### Elixir Solution:

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

end
end
```

### Erlang Solution:

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

### Racket Solution:

```
(define/contract (max-score nums)
  (-> (listof exact-integer?) exact-integer?))
)
```