Problem 2091: Removing Minimum and Maximum From Array

Problem Information

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

Problem Description

You are given a

0-indexed

array of

distinct

integers

nums

.

There is an element in

nums

that has the

lowest

value and an element that has the

highest

value. We call them the

minimum

and

maximum

respectively. Your goal is to remove

both

these elements from the array.

A

deletion

is defined as either removing an element from the

front

of the array or removing an element from the

back

of the array.

Return

the

minimum

number of deletions it would take to remove

both

the minimum and maximum element from the array.

Example 1:

Input:

nums = [2,

10

,7,5,4,

1

,8,6]

Output:

5

Explanation:

The minimum element in the array is nums[5], which is 1. The maximum element in the array is nums[1], which is 10. We can remove both the minimum and maximum by removing 2 elements from the front and 3 elements from the back. This results in 2 + 3 = 5 deletions, which is the minimum number possible.

Example 2:

Input:

nums = [0,

-4

,

19

,1,8,-2,-3,5]

Output:

3

Explanation:

The minimum element in the array is nums[1], which is -4. The maximum element in the array is nums[2], which is 19. We can remove both the minimum and maximum by removing 3 elements from the front. This results in only 3 deletions, which is the minimum number possible.

Example 3:

Input:

nums = [

101

]

Output:

1

Explanation:

There is only one element in the array, which makes it both the minimum and maximum element. We can remove it with 1 deletion.

Constraints:

1 <= nums.length <= 10

5

-10

5

<= nums[i] <= 10

5

The integers in

nums

are

distinct

.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

```
}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

```
function minimumDeletions($nums) {

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (minimum-deletions nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Removing Minimum and Maximum From Array
* 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 minimumDeletions(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Removing Minimum and Maximum From Array
* 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 minimumDeletions(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Removing Minimum and Maximum From Array
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 minimumDeletions(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Removing Minimum and Maximum From Array
* 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 minimumDeletions = function(nums) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Removing Minimum and Maximum From Array
* 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 minimumDeletions(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Removing Minimum and Maximum From Array
* 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
*/

public class Solution {
public int MinimumDeletions(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Removing Minimum and Maximum From Array
* 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
*/

int minimumDeletions(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Removing Minimum and Maximum From Array
// 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

func minimumDeletions(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Removing Minimum and Maximum From Array
// 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 minimum_deletions(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

```
}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (minimum-deletions nums)
(-> (listof exact-integer?) exact-integer?)
)
```