

Problem 3634: Minimum Removals to Balance Array

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

You are given an integer array

nums

and an integer

k

An array is considered

balanced

if the value of its

maximum

element is

at most

k

times the

minimum

element.

You may remove

any

number of elements from

nums

without making it

empty

.

Return the

minimum

number of elements to remove so that the remaining array is balanced.

Note:

An array of size 1 is considered balanced as its maximum and minimum are equal, and the condition always holds true.

Example 1:

Input:

nums = [2,1,5], k = 2

Output:

Explanation:

Remove

$\text{nums}[2] = 5$

to get

$\text{nums} = [2, 1]$

.

Now

$\max = 2$

,

$\min = 1$

and

$\max \leq \min * k$

as

$2 \leq 1 * 2$

. Thus, the answer is 1.

Example 2:

Input:

$\text{nums} = [1, 6, 2, 9], k = 3$

Output:

2

Explanation:

Remove

$\text{nums}[0] = 1$

and

$\text{nums}[3] = 9$

to get

$\text{nums} = [6, 2]$

Now

$\max = 6$

,

$\min = 2$

and

$\max \leq \min * k$

as

$6 \leq 2 * 3$

. Thus, the answer is 2.

Example 3:

Input:

nums = [4,6], k = 2

Output:

0

Explanation:

Since

nums

is already balanced as

$6 \leq 4 * 2$

, no elements need to be removed.

Constraints:

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

5

$1 \leq \text{nums}[i] \leq 10$

9

$1 \leq k \leq 10$

5

Code Snippets

C++:

```
class Solution {  
public:
```

```
int minRemoval(vector<int>& nums, int k) {  
}  
};
```

Java:

```
class Solution {  
public int minRemoval(int[] nums, int k) {  
}  
}
```

Python3:

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

Python:

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

JavaScript:

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

TypeScript:

```
function minRemoval(nums: number[], k: number): number {  
}  
};
```

C#:

```
public class Solution {  
    public int MinRemoval(int[] nums, int k) {  
        }  
    }  
}
```

C:

```
int minRemoval(int* nums, int numsSize, int k) {  
}  
}
```

Go:

```
func minRemoval(nums []int, k int) int {  
}  
}
```

Kotlin:

```
class Solution {  
    fun minRemoval(nums: IntArray, k: Int): Int {  
        }  
    }  
}
```

Swift:

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

Rust:

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

Ruby:

```
# @param {Integer[]} nums
# @param {Integer} k
# @return {Integer}
def min_removal(nums, k)

end
```

PHP:

```
class Solution {

    /**
     * @param Integer[] $nums
     * @param Integer $k
     * @return Integer
     */
    function minRemoval($nums, $k) {

    }
}
```

Dart:

```
class Solution {
    int minRemoval(List<int> nums, int k) {
        }
    }
```

Scala:

```
object Solution {
    def minRemoval(nums: Array[Int], k: Int): Int = {
        }
```

```
}
```

Elixir:

```
defmodule Solution do
  @spec min_removal(nums :: [integer], k :: integer) :: integer
  def min_removal(nums, k) do
    end
  end
```

Erlang:

```
-spec min_removal(Nums :: [integer()], K :: integer()) -> integer().
min_removal(Nums, K) ->
  .
```

Racket:

```
(define/contract (min-removal nums k)
  (-> (listof exact-integer?) exact-integer? exact-integer?))
```

Solutions

C++ Solution:

```
/*
 * Problem: Minimum Removals to Balance Array
 * Difficulty: Medium
 * Tags: array, 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 minRemoval(vector<int>& nums, int k) {
```

```
}
```

```
} ;
```

Java Solution:

```
/**  
 * Problem: Minimum Removals to Balance Array  
 * Difficulty: Medium  
 * Tags: array, 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 minRemoval(int[] nums, int k) {  
        // Implementation  
    }  
}
```

Python3 Solution:

```
"""  
Problem: Minimum Removals to Balance Array  
Difficulty: Medium  
Tags: array, 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 minRemoval(self, nums: List[int], k: int) -> int:  
        # TODO: Implement optimized solution  
        pass
```

Python Solution:

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

JavaScript Solution:

```
/**  
 * Problem: Minimum Removals to Balance Array  
 * Difficulty: Medium  
 * Tags: array, 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  
 * @param {number} k  
 * @return {number}  
 */  
var minRemoval = function(nums, k) {  
};
```

TypeScript Solution:

```
/**  
 * Problem: Minimum Removals to Balance Array  
 * Difficulty: Medium  
 * Tags: array, 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 minRemoval(nums: number[], k: number): number {
```

```
};
```

C# Solution:

```
/*
 * Problem: Minimum Removals to Balance Array
 * Difficulty: Medium
 * Tags: array, 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 MinRemoval(int[] nums, int k) {
        return 0;
    }
}
```

C Solution:

```
/*
 * Problem: Minimum Removals to Balance Array
 * Difficulty: Medium
 * Tags: array, 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 minRemoval(int* nums, int numsSize, int k) {
    return 0;
}
```

Go Solution:

```
// Problem: Minimum Removals to Balance Array
// Difficulty: Medium
```

```

// Tags: array, 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 minRemoval(nums []int, k int) int {
}

```

Kotlin Solution:

```

class Solution {
    fun minRemoval(nums: IntArray, k: Int): Int {
        return 0
    }
}

```

Swift Solution:

```

class Solution {
    func minRemoval(_ nums: [Int], _ k: Int) -> Int {
        return 0
    }
}

```

Rust Solution:

```

// Problem: Minimum Removals to Balance Array
// Difficulty: Medium
// Tags: array, 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_removal(nums: Vec<i32>, k: i32) -> i32 {
        return 0
    }
}

```

Ruby Solution:

```
# @param {Integer[]} nums
# @param {Integer} k
# @return {Integer}
def min_removal(nums, k)

end
```

PHP Solution:

```
class Solution {

    /**
     * @param Integer[] $nums
     * @param Integer $k
     * @return Integer
     */
    function minRemoval($nums, $k) {

    }
}
```

Dart Solution:

```
class Solution {
  int minRemoval(List<int> nums, int k) {
    }
}
```

Scala Solution:

```
object Solution {
  def minRemoval(nums: Array[Int], k: Int): Int = {
    }
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_removal(nums :: [integer], k :: integer) :: integer
def min_removal(nums, k) do

end
end
```

Erlang Solution:

```
-spec min_removal(Nums :: [integer()], K :: integer()) -> integer().
min_removal(Nums, K) ->
.
```

Racket Solution:

```
(define/contract (min-removal nums k)
(-> (listof exact-integer?) exact-integer? exact-integer?))
```