Problem 3520: Minimum Threshold for Inversion Pairs Count

Problem Information

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

Problem Description

You are given an array of integers

nums

and an integer

k

.

An inversion pair with a

threshold

x

is defined as a pair of indices

(i, j)

such that:

i < j

nums[i] > nums[j]

The difference between the two numbers is

at most

x

(i.e.

nums[i] - nums[j] <= x

).

Your task is to determine the

minimum

integer

min_threshold

such that there are

at least

k

inversion pairs with threshold

min_threshold

.

If no such integer exists, return

-1

.

Example 1:

Input:

nums = [1,2,3,4,3,2,1], k = 7

Output:

2

Explanation:

For threshold

x = 2

, the pairs are:

(3, 4)

where

nums[3] == 4

and

nums[4] == 3

.

(2, 5)

where

nums[2] == 3

and

nums[5] == 2

.

(3, 5)

where

nums[3] == 4

and

nums[5] == 2

.

(4, 5)

where

nums[4] == 3

and

nums[5] == 2

.

(1, 6)

where

nums[1] == 2

and

nums[6] == 1

.

(2, 6)

where

nums[2] == 3

and

nums[6] == 1

.

(4, 6)

where

nums[4] == 3

and

nums[6] == 1

.

(5, 6)

where

nums[5] == 2

and

nums[6] == 1

.

There are less than

k

inversion pairs if we choose any integer less than 2 as threshold.

Example 2:

Input:

nums = [10,9,9,9,1], k = 4

Output:

8

Explanation:

For threshold

x = 8

, the pairs are:

(0, 1)

where

nums[0] == 10

and

nums[1] == 9

.

(0, 2)

where

nums[0] == 10

and

nums[2] == 9

.

(0, 3)

where

nums[0] == 10

and

nums[3] == 9

.

(1, 4)

where

nums[1] == 9

and

nums[4] == 1

.

(2, 4)

where

nums[2] == 9

and

nums[4] == 1

.

(3, 4)

where

nums[3] == 9

and

nums[4] == 1

.

There are less than

k

inversion pairs if we choose any integer less than 8 as threshold.

Constraints:

1 <= nums.length <= 10

4

1 <= nums[i] <= 10

9

1 <= k <= 10

9

Code Snippets

C++:

```
class Solution {
public:
int minThreshold(vector<int>& nums, int k) {

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function minThreshold(nums: number[], k: number): number {

};
```

C#:

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

}
}
```

C:

```
int minThreshold(int* nums, int numsSize, int k) {

}
```

Go:

```
func minThreshold(nums []int, k int) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
```

```
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Minimum Threshold for Inversion Pairs Count
* Difficulty: Medium
* Tags: array, tree, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public:
int minThreshold(vector<int>& nums, int k) {
```

```
}
};
```

Java Solution:

```
/**
* Problem: Minimum Threshold for Inversion Pairs Count
* Difficulty: Medium
* Tags: array, tree, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public int minThreshold(int[] nums, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Threshold for Inversion Pairs Count
Difficulty: Medium
Tags: array, tree, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Threshold for Inversion Pairs Count
* Difficulty: Medium
* Tags: array, tree, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Threshold for Inversion Pairs Count
* Difficulty: Medium
* Tags: array, tree, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function minThreshold(nums: number[], k: number): number {
```

```
};
```

C# Solution:

```
/*
* Problem: Minimum Threshold for Inversion Pairs Count
* Difficulty: Medium
* Tags: array, tree, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

public class Solution {
public int MinThreshold(int[] nums, int k) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Threshold for Inversion Pairs Count
* Difficulty: Medium
* Tags: array, tree, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

int minThreshold(int* nums, int numsSize, int k) {

}
```

Go Solution:

```
// Problem: Minimum Threshold for Inversion Pairs Count
// Difficulty: Medium
```

```
// Tags: array, tree, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func minThreshold(nums []int, k int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Minimum Threshold for Inversion Pairs Count
// Difficulty: Medium
// Tags: array, tree, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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