Problem 1984: Minimum Difference Between Highest and Lowest of K Scores

Problem Information

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

Problem Description

You are given a

0-indexed

integer array

nums

, where

nums[i]

represents the score of the

i

th

student. You are also given an integer

k

.

Pick the scores of any

k

students from the array so that the

difference

between the

highest

and the

lowest

of the

k

scores is

minimized

.

Return

the

minimum

possible difference

.

Example 1:

Input:

nums = [90], k = 1

Output:

0

Explanation:

There is one way to pick score(s) of one student: - [

90

]. The difference between the highest and lowest score is 90 - 90 = 0. The minimum possible difference is 0.

Example 2:

Input:

nums = [9,4,1,7], k = 2

Output:

2

Explanation:

There are six ways to pick score(s) of two students: - [

9

,

4

,1,7]. The difference between the highest and lowest score is 9 - 4 = 5. - [

9

,4,

1

,7]. The difference between the highest and lowest score is 9 - 1 = 8. - [

9

,4,1,

7

]. The difference between the highest and lowest score is 9 - 7 = 2. - [9,

4

,

1

,7]. The difference between the highest and lowest score is 4 - 1 = 3. - [9,

4

,1,

7

]. The difference between the highest and lowest score is 7 - 4 = 3. - [9,4,

1

,

7

]. The difference between the highest and lowest score is 7 - 1 = 6. The minimum possible difference is 2.

Constraints:

1 <= k <= nums.length <= 1000

0 <= nums[i] <= 10

5

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Minimum Difference Between Highest and Lowest of K Scores
* Difficulty: Easy
* 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 minimumDifference(vector<int>& nums, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Difference Between Highest and Lowest of K Scores
* Difficulty: Easy
* 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 minimumDifference(int[] nums, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Difference Between Highest and Lowest of K Scores
Difficulty: Easy
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 minimumDifference(self, nums: List[int], k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Difference Between Highest and Lowest of K Scores
* Difficulty: Easy
* 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 minimumDifference = function(nums, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Difference Between Highest and Lowest of K Scores
* Difficulty: Easy
* 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 minimumDifference(nums: number[], k: number): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Difference Between Highest and Lowest of K Scores
* Difficulty: Easy
* 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 MinimumDifference(int[] nums, int k) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Difference Between Highest and Lowest of K Scores
* Difficulty: Easy
* 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 minimumDifference(int* nums, int numsSize, int k) {

}
```

Go Solution:

```
// Problem: Minimum Difference Between Highest and Lowest of K Scores
// Difficulty: Easy
// 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 minimumDifference(nums []int, k int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Minimum Difference Between Highest and Lowest of K Scores
// Difficulty: Easy
// 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 minimum_difference(nums: Vec<i32>, k: i32) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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