Problem 480: Sliding Window Median

Problem Information

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

Problem Description

The

median

is the middle value in an ordered integer list. If the size of the list is even, there is no middle value. So the median is the mean of the two middle values.

For examples, if

arr = [2,

3

,4]

, the median is

3

.

For examples, if

arr = [1,

2,3

,4]

, the median is

(2 + 3) / 2 = 2.5

.

You are given an integer array

nums

and an integer

k

. There is a sliding window of size

k

which is moving from the very left of the array to the very right. You can only see the

k

numbers in the window. Each time the sliding window moves right by one position.

Return

the median array for each window in the original array

. Answers within

10

-5

of the actual value will be accepted.

Example 1:

Input:

nums = [1,3,-1,-3,5,3,6,7], k = 3

Output:

[1.00000,-1.00000,-1.00000,3.00000,5.00000,6.00000]

Explanation:

Window position Median ---------------- ----- [

1 3 -1

] -3 5 3 6 7 1 1 [

3 -1 -3

] 5 3 6 7 -1 1 3 [

-1 -3 5

] 3 6 7 -1 1 3 -1 [

-3 5 3

] 6 7 3 1 3 -1 -3 [

5 3 6

] 7 5 1 3 -1 -3 5 [

3 6 7

] 6

Example 2:

Input:

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

Output:

[2.00000,3.00000,3.00000,3.00000,2.00000,3.00000,2.00000]

Constraints:

1 <= k <= nums.length <= 10

5

-2

31

<= nums[i] <= 2

31

- 1

Code Snippets

C++:

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

}
};
```

Java:

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

```
}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
```

```
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
double* medianSlidingWindow(int* nums, int numsSize, int k, int* returnSize)
{

}
```

Go:

```
func medianSlidingWindow(nums []int, k int) []float64 {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn median_sliding_window(nums: Vec<i32>, k: i32) -> Vec<f64> {

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

```
defmodule Solution do
@spec median_sliding_window(nums :: [integer], k :: integer) :: [float]
```

```
def median_sliding_window(nums, k) do

end
end
```

Erlang:

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

Racket:

```
(define/contract (median-sliding-window nums k)
(-> (listof exact-integer?) exact-integer? (listof flonum?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Sliding Window Median
* Difficulty: Hard
* Tags: array, hash, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
vector<double> medianSlidingWindow(vector<int>& nums, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Sliding Window Median
* Difficulty: Hard
* Tags: array, hash, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
}
```

Python3 Solution:

```
"""
Problem: Sliding Window Median
Difficulty: Hard
Tags: array, hash, queue, heap

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Sliding Window Median
* Difficulty: Hard
* Tags: array, hash, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Sliding Window Median
* Difficulty: Hard
* Tags: array, hash, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function medianSlidingWindow(nums: number[], k: number): number[] {

};
```

C# Solution:

```
/*
* Problem: Sliding Window Median
* Difficulty: Hard
```

```
* Tags: array, hash, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
}
```

C Solution:

```
/*
* Problem: Sliding Window Median
* Difficulty: Hard
* Tags: array, hash, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
double* medianSlidingWindow(int* nums, int numsSize, int k, int* returnSize)
{

}
```

Go Solution:

```
// Problem: Sliding Window Median
// Difficulty: Hard
// Tags: array, hash, queue, heap
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
```

```
// Space Complexity: O(n) for hash map

func medianSlidingWindow(nums []int, k int) []float64 {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Sliding Window Median
// Difficulty: Hard
// Tags: array, hash, queue, heap
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn median_sliding_window(nums: Vec<i32>, k: i32) -> Vec<f64> {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

```
end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (median-sliding-window nums k)
(-> (listof exact-integer?) exact-integer? (listof flonum?))
)
```