Problem 480: Sliding Window Median

Problem Information

Difficulty: Hard
Acceptance Rate: 38.83%
Paid Only: No
Tags: Array, Hash Table, Sliding Window, Heap (Priority Queue)

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 <= 105` * `-231 <= nums[i] <= 231 - 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]:
```