

239. Sliding Window Maximum

Hard

5269

218

Add to List

Share

You are given an array of integers `nums`, 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 max sliding window.

Example 1:

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

Output: `[3,3,5,5,6,7]`

Explanation:

Window position	Max
[1 3 -1] -3 5 3 6 7	3
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	5
1 3 -1 -3 [5 3 6] 7	6
1 3 -1 -3 5 [3 6 7]	7

Example 2:

Input: `nums = [1]`, `k = 1`

Output: `[1]`

Example 3:

Input: `nums = [1,-1]`, `k = 1`

Output: `[1,-1]`

Example 4:

Input: `nums = [9,11]`, `k = 2`

Output: `[11]`

Example 5:

Input: `nums = [4,-2]`, `k = 2`

Output: `[4]`

Constraints:

- `1 <= nums.length <= 105`
- `-104 <= nums[i] <= 104`
- `1 <= k <= nums.length`

index	0	1	2	3	4
nums	1	3	-1	-3	5

Handwritten notes:
A red arrow points from index 1 to index 0, labeled `pop(nums[i-k+1])`.
A green arrow points from index 3 to index 4, labeled `push(nums[i])`.
The value `k=3` is written in orange above the window [3, -1, -3].

```

1 class MonotonicQueue {
2 public:
3     // push element n at back
4     void push(int n) {
5         while (!data.empty() && data.back() < n) {
6             data.pop_back();
7         }
8         data.push_back(n);
9     }
10
11     // return the max value in queue
12     int max() {
13         return data.front();
14     }
15
16     // if front is n, pop it
17     void pop(int n) {
18         if (!data.empty() && data.front() == n) {
19             data.pop_front();
20         }
21     }
22
23 private:
24     deque<int> data;
25 };
26
27 class Solution {
28 public:
29     vector<int> maxSlidingWindow(vector<int>& nums, int k) {
30         MonotonicQueue window;
31         vector<int> res;
32
33         for (int i = 0; i < nums.size(); i++) {
34             if (i < k - 1) {
35                 window.push(nums[i]);
36             } else {
37                 window.push(nums[i]);
38                 res.push_back(window.max());
39                 window.pop(nums[i - k + 1]);
40             }
41         }
42         return res;
43     }
44 };

```