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:

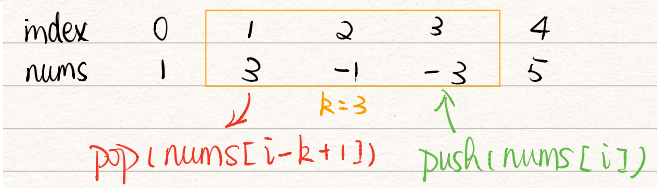
```
Input: nums = [9,11], k = 2
Output: [11]
```

Example 5:

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

Constraints:

- `1 <= nums.length <= 10^5`
- `-10^4 <= nums[i] <= 10^4`
- `1 <= k <= nums.length`



```cpp
class MonotonicQueue {
public:
    // push element n at back
    void push(int n) {
        while (!data.empty() && data.back() < n) {
            data.pop_back();
        }
        data.push_back(n);
    }

    // return the max value in queue
    int max() {
        return data.front();
    }

    // if front is n, pop it
    void pop(int n) {
        if (!data.empty() && data.front() == n) {
            data.pop_front();
        }
    }

private:
    deque<int> data;
};

class Solution {
public:
    vector<int> maxSlidingWindow(vector<int>& nums, int k) {
        MonotonicQueue window;
        vector<int> res;

        for (int i = 0; i < nums.size(); i++) {
            if (i < k - 1) {
                window.push(nums[i]);
            } else {
                window.push(nums[i]);
                res.push_back(window.max());
                window.pop(nums[i - k + 1]);
            }
        }
        return res;
    }
};
```