239. Sliding Window Maximum

Hard 6510 246 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.

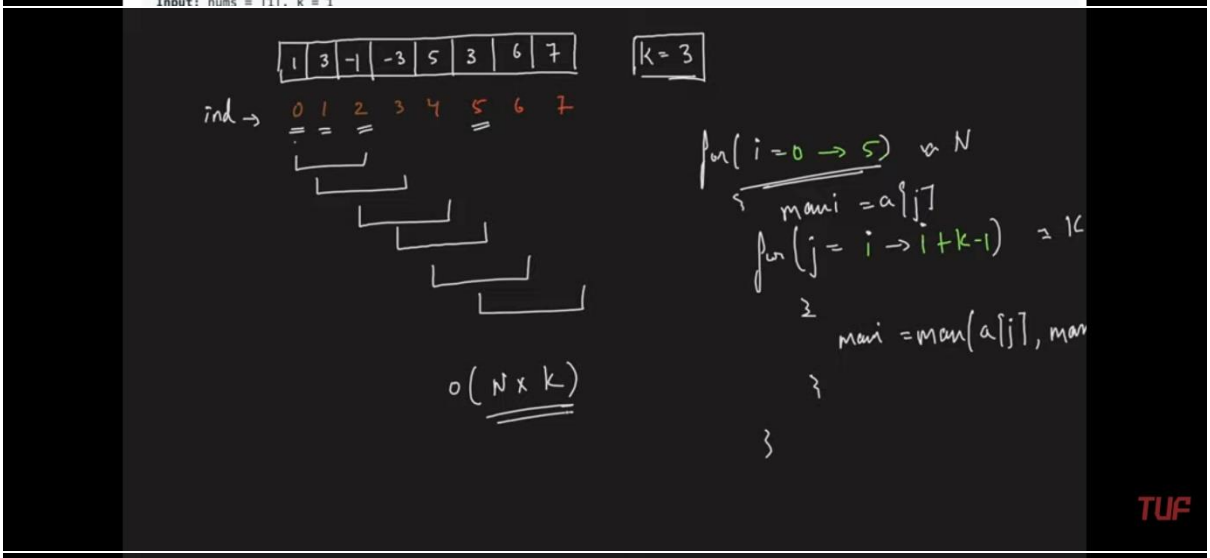
Return *the max sliding window.*

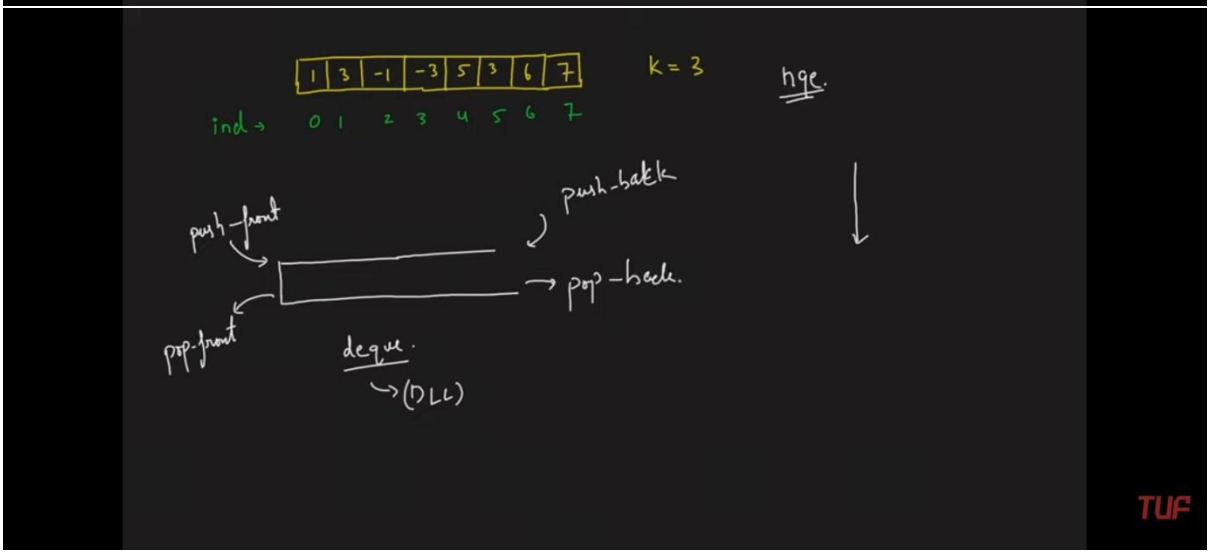
Example 1:

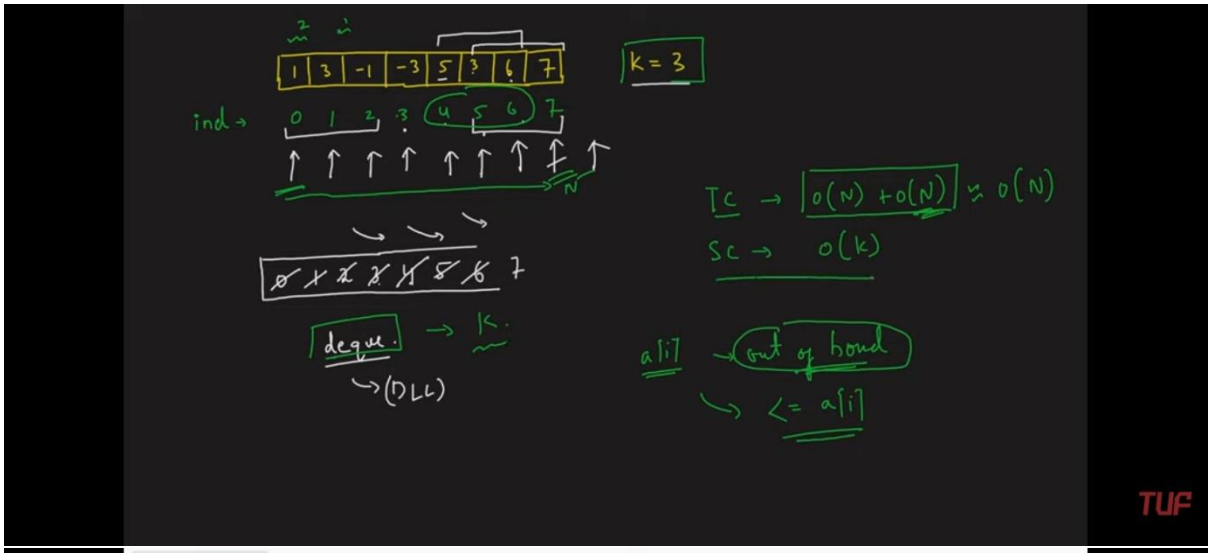
```
Input: nums = [1,3,-1,-3,5,3,6,7], k = 3
Output: [3,3,5,5,6,7]
Explanation:
Window position                Max
---------------               -----
[1  3  -1] -3  5  3  6  7       3
 1 [3  -1  -3] 5  3  6  7       3
 1  3 [-1  -3  5] 3  6  7       5
 1  3  -1 [-3  5  3] 6  7       5
 1  3  -1  -3 [5  3  6] 7       6
 1  3  -1  -3  5 [3  6  7]      7
```

Example 2:

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







C++ • Autocomplete

```cpp
class Solution {
public:
    vector<int> maxSlidingWindow(vector<int>& nums, int k) {
        deque<int> dq;
        vector<int> ans;
        for (int i=0; i<nums.size(); i++) {
            if (!dq.empty() && dq.front() == i-k) dq.pop_front();

            while (!dq.empty() && nums[dq.back()] < nums[i])
                dq.pop_back();

            dq.push_back(i);
            if (i>=k-1) ans.push_back(nums[dq.front()]);
        }
        return ans;
    }
};
```

Your previous code was restored from your local storage. Reset to default

C++ • Autocomplete

```cpp
class Solution {
public:
    vector<int> maxSlidingWindow(vector<int>& nums, int k) {
        deque<int> dq;
        vector<int> ans;
        for (int i=0; i<nums.size(); i++) {
            if (!dq.empty() && dq.front() == i-k) dq.pop_front();

            while (!dq.empty() && nums[dq.back()] <= nums[i])
                dq.pop_back();

            dq.push_back(i);
            if (i>=k-1) ans.push_back(nums[dq.front()]);
        }
        return ans;
    }
};
```


Your previous code was restored from your local storage. Reset to default