Based on research it appears that
O(n) time is also referred to as O(1) time

Daily Coding Problem

Good morning! Here's your coding interview problem for today.

This problem was asked by Google.

Given an array of integers and a number k, where 1 <= k <= length of the array, compute the maximum values of each subarray of length k.

For example, given array = [10, 5, 2, 7, 8, 7] and k = 3, we should get: [10, 7, 8, 8], since:

- 10 = max(10, 5, 2)
- 7 = max(5, 2, 7)
- 8 = max(2, 7, 8)
- 8 = max(7, 8, 7)

Based on research, this has no impact on O(n) time

Do this in O(n) time and O(k) space. You can modify the input array in-place and you do not need to store the results. You can simply print them out as you compute them.

0

O(*n*) time would be that the time taken by the algorithm would be proportional to the length of the array. I.e., the running time for an array of length 1000 would be half as the running time for an array of length 2000.

RESEARCH ONLINE

Constant Time Complexity, also known as O(1), **refers to an algorithm's efficiency where the running time remains independent of the input size**. It signifies the most efficient performance as the algorithm's execution time does not increase with larger inputs.

RESEARCH ONLINE

O(1) Constant time

It denotes constant time complexity, which means that the running time of an algorithm remains constant regardless of the size of the input. An algorithm with a time complexity of O(1) performs a fixed number of operations and is not affected by the size of the input. 20 Jan 2023

RESEARCH ONLINE

To comprehend the concept of O(1) complexity it's important to recognize that the runtime of an algorithm, with this complexity remains constant regardless of the input size. This characteristic is quite impressive as it indicates that the algorithm is highly efficient and its performance remains consistent.

RESEARCH ONLINE

- **Optimizing Crucial Code Paths:** When dealing with algorithms or software systems it's quite common to come across code paths that need to be executed as swiftly, as possible. A used technique, in software development involves identifying and fine tuning these code paths to achieve a complexity of O(1).

At first instance, I am unsure of how to interpret this into the code.
I am presuming that if I traverse through the array for length k, I will be examining
(k-1) and (k+1). I can assume there will already be some proportionality (length of array with time taken processing) without taking any actions...I will be processing exactly sub-array length 3.
I am unsure of any implementations that would hinder the time execution since I will be endeavour not to store results (as stated in the query). So there will be no overheads.

The complexity will be uniform since we are interested in maximum value in sub-array size of k within the given array.