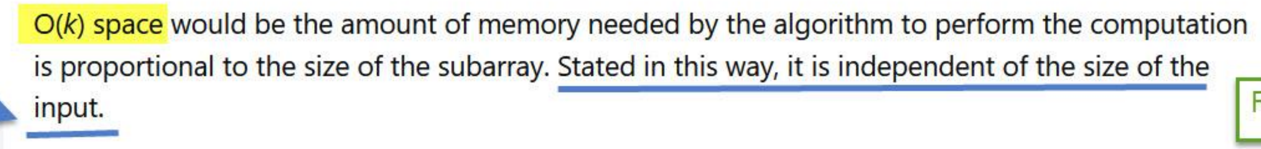
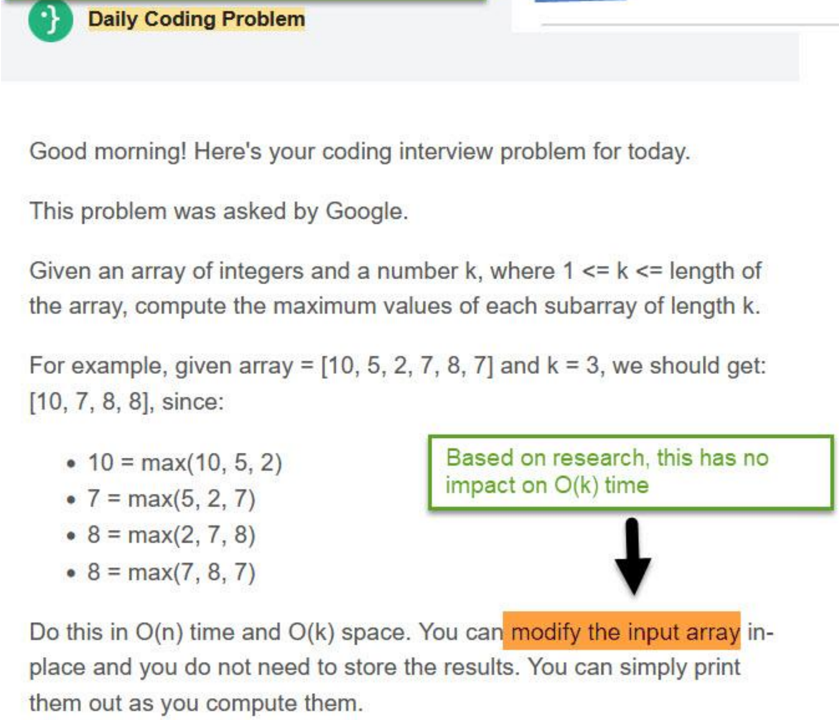
I am unsure about this bit since it mentions it is independent of the input... But in other sources it mentions it is dependent. As far as I am concerned with this challenge, there will be no change to input array once it is established from the offset.. Especially if I intend to use immutable data type.



O(*k*) space would be the amount of memory needed by the algorithm to perform the computation is proportional to the size of the subarray. Stated in this way, it is independent of the size of the input.

RESEARCH ONLINE



Daily Coding Problem

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

This problem was asked by Google.

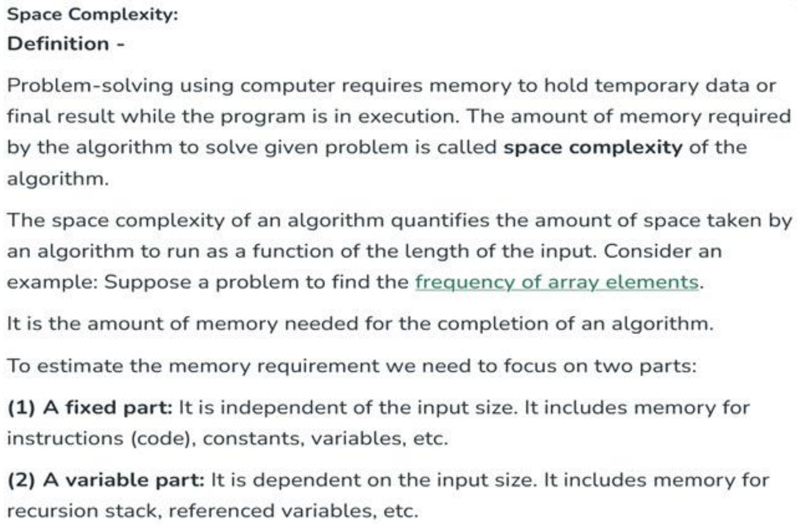
Given an array of integers and a number k, where 1 <= k <= length of the array, compute the maximum values of each subarray of length k.

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)

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.

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



Space Complexity:
Definition -

Problem-solving using computer requires memory to hold temporary data or final result while the program is in execution. The amount of memory required by the algorithm to solve given problem is called **space complexity** of the algorithm.

The space complexity of an algorithm quantifies the amount of space taken by an algorithm to run as a function of the length of the input. Consider an example: Suppose a problem to find the frequency of array elements.

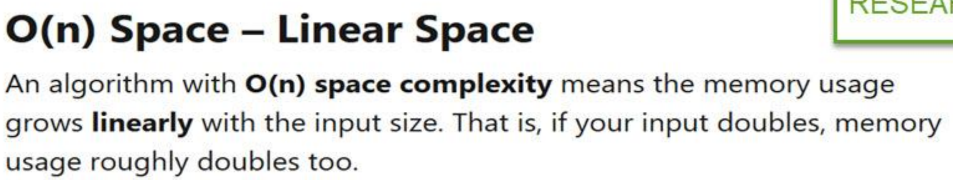
It is the amount of memory needed for the completion of an algorithm.

To estimate the memory requirement we need to focus on two parts:

(1) A fixed part: It is independent of the input size. It includes memory for instructions (code), constants, variables, etc.

(2) A variable part: It is dependent on the input size. It includes memory for recursion stack, referenced variables, etc.

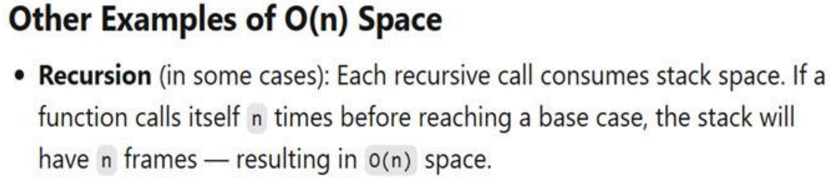
RESEARCH ONLINE



O(n) Space – Linear Space

An algorithm with **O(n) space complexity** means the memory usage grows **linearly** with the input size. That is, if your input doubles, memory usage roughly doubles too.

RESEARCH ONLINE



Other Examples of O(n) Space

- **Recursion** (in some cases): Each recursive call consumes stack space. If a function calls itself `n` times before reaching a base case, the stack will have `n` frames — resulting in `O(n)` space.

This now refers to the amount of memory. It refers to the subarray having an influence and it is independent to the size of the input.
This is quite assuring given that it provides opportunity to adjust the main array. For instance I can consider using immutability to modify the main input on the offset.
We know if results are printed and not stored it will not affect the memory storage of the algorithm. It is difficult to ascertain where the memory requirements will deviate.