# Exercise Session n. 2 (3 March 2023)

**Algorithms and Data Structures**

## Monotonic Sequence

Write a function `is_monotonic(A, i, j)` that, given an array of numbers, $A$, returns `True` if $A$ contains a *monotonic* sub-sequence starting at position $i$ and ending at position $j$, or `False` otherwise. A monotonic sequence is one that is either in non-decreasing or non-increasing order.

### Examples

```
>>> is_monotonic([1,2,3], 0, 2)
True
>>> is_monotonic([1,1,7,7,9], 0, 4)
True
>>> is_monotonic([9,9,5], 0, 2)
True
>>> is_monotonic([6,6,6,6,6,6], 2, 4)
True
>>> is_monotonic([1,1,1,2,1,3], 0, 5)
False
>>> is_monotonic([1,1,1,2,1,3], 0, 3)
True
>>> is_monotonic([3,2,1,3,2,1], 0, 5)
False
>>> is_monotonic([3,2,1,3,2,1], 2, 3)
True
>>> is_monotonic([3,2,1,3,2,1], 2, 4)
False
>>> is_monotonic([3,2,1,3,2,1], 3, 5)
True
>>> is_monotonic([7,4,7], 0, 2)
False
>>> is_monotonic([7,4,7], 1, 2)
True
```

## Explain, Analyze, and Improve an Algorithm!

Consider the following algorithm:

```
def algorithm_x (A):
    x = 0
    for i in range(len(A)):
        for j in range(i+1, len(A)):
            if is_monotonic (A, i, j):
                if x < j - i:
                    x = j - i
    return x
```

## Question 1

Explain what `algorithm_x` does. Do not paraphrase the code. Instead, explain the high-level semantics of the algorithm.

## Question 2

Analyze the complexity of `algorithm_x`. Give your best characterization of the complexity of the algorithm.

## Question 3

Write an algorithm `better_algorithm_x(A)` that is functionally identical to `algorithm_x(A)` but with a strictly better time complexity.

## Question 4

Write an algorithm `linear_algorithm_x(A)` that is functionally identical to `algorithm_x(A)` and that runs in time $O(n)$.