Lab 5

**David Blumenthal, Florian Hofer,
Werner Nutt, Daniele Porello**

5. Array Algorithms

1. First Index of an Element Greater Than x

We now consider arrays that are *sorted*, like

$A = [3, 5, 5, 7, 13, 13].$

Given a number x, we want to find the first index in A with an element greater than x. For example:

- for $x = 5$, the answer is 4, since 7 is the first value in A that is greater than 5, and the first occurrence of 7 is at index 4;
- for $x = 8$, the answer is 5,

  since 13 is the first value in A that is greater than 8, and the first occurrence of 13 is at index 5;
- if $x = 15$, the answer is -1, since there is no element greater than 15.

Your task is to develop an *efficient* procedure

```
int firstOccGreater(int x, int[] A).
```

that, given an integer x and a sorted integer array A, returns the first index of an element in A that is greater than x, if such an element exists, and returns -1 otherwise.

(i) Describe in words the idea for an efficient algorithm for this task.

(ii) Write pseudocode for a recursive version of the algorithm.

(iii) What is the running time of your algorithm?

2. Positive Region

We describe a simplified machine learning problem.

Suppose we have data about *cases*, where a case has a *size*, described by an integer in the range between 1 and some maximal number m, and a classifications as either *positive* or *negative*. We can assume that the number m is less or equal to the number of cases.

The data are stored in two arrays P and N, each of length m, where $P[i]$ records the number of positive cases of size i and $N[i]$ records the number of negative cases of size i.

Our overall goal is to find the region where the positive cases are concentrated. For each range of indices $[l, r]$, where $1 \leq l,\ r \leq m$, we define

$$S(l, r) := \sum_{i=l}^{r} (P[i] - N[i]).$$

With this definition we can formally define our goal as the one of finding l and r such that $S(l, r)$ is maximal.

Note that according to our definition, the sum ranges over an empty segment for $l > r$ and then equals 0. If all differences $P[i] - N[i]$ are negative, that is, if for all sizes there are more negative than positive cases, then the empty range has the minimal sum.

(i) Develop an algorithm that returns, given m, P, and N, the value

$$\max_{l,r} S(l, r).$$

(ii) Turn this algorithm in to one that returns also the values of l and r.