

# Fundamental Algorithms

## Exercise 1

Prove that the running time of an algorithm is  $\Theta(g(n))$ , if and only if its worst-case running time is  $O(n)$ , and its best-case running time is  $\Omega(n)$ .

## Exercise 2

Show that the computation of the minimum of  $n$  different integers requires at least  $n - 1$  comparisons.

### Solution:

Let  $M \subset \mathbb{Z}$  be the set of the  $n$  input integers. We will examine graphs  $G = (M, E)$ , where  $E \subset M \times M$ , and  $(a, b) \in E$ , if and only if  $a$  has been compared to  $b$ . An algorithm to find the minimum of  $M$  has to build a graph  $G$  such that all elements of  $M$  are connected by the edges in  $E$ , i.e. any two elements of  $M$  have to be connected by a path of edges. We will therefore prove (by induction over  $n$ ) that we need at least  $n - 1$  edges to connect  $n$  elements.

We prove this statement by induction over  $n$ :

**case**  $n = 1$ : if we only have one element, we do not need any edges;

**case**  $n = 2$ : to connect 2 elements, we need exactly 1 edge;

**step**  $n \rightarrow n + 1$ : we assume that our statement is correct for  $1, 2, \dots, n$ , and try to prove that this implies it is also correct for  $n + 1$ :

By removing one (or more if required) of the edges of  $E$ , we now split the graph into two non-empty subsets such that:

- the subsets are no longer connected to each other

- in both subsets, all elements are still connected to each other.

Let the sizes of the subsets be  $s_1$  and  $s_2 = (n + 1) - s_1$ . As both subsets are non-empty, both,  $s_1$  and  $s_2$ , will contain at most  $n$  elements. Therefore, the induction assumption applies, and the respective subsets will be connected by at least  $(s_1 - 1)$  and  $(s_2 - 1)$  edges. Including the one deleted edge, our original graph therefore contained at least  $(s_1 - 1) + (s_2 - 1) + 1 = s_1 + s_2 - 1 = (n + 1) - 1 = n$  edges.

### Exercise 3

Prove or disprove the following statement:

If we sort each row of a matrix, and, after that, sort each column of the matrix, the rows of the matrix will still be sorted afterwards.

#### Solution:

Let  $(a_{ij})$  be the matrix after sorting each of its rows. Without loss of generality, we will assume that the sorting is done in ascending order.

Now, assume that there is a line  $l$  that is not in sorted order, i.e. there are two indices  $j < k$  such that  $a_{lj} > a_{lk}$ :

As the  $k$ -th column of the matrix is sorted, it contains at least  $l$  elements that are  $\leq a_{lk}$  ( $a_{lk}$  included). To each of these  $l$  elements belongs an element of the  $j$ -th column that was in the same line of the matrix before the columns were sorted. These  $l$  elements are all smaller or equal to their corresponding element in the  $k$ -th column, because the lines of the matrix were already sorted at this time. Hence, these  $l$  elements of the  $j$ -th column are all  $\leq a_{lk}$ , and therefore  $< a_{lj}$ .

Hence, there are at least  $l$  elements in column  $j$  that are smaller than  $a_{lj}$ . Therefore, in column  $j$ , at least one element smaller than  $a_{lj}$  has to be placed below  $a_{lj}$ . This means that the  $j$ -th column can not be in sorted order, which contradicts our assumption.