PROGRAMMING TECHNIQUES, A.A. 2023/2024

Laboratory 7

Objectives

- Solve iterative verification / selection / **sorting** problems using vectors and matrices

Tecnical contents

- Basics of Input Output
- Use of functions
- Conditional and iterative constructs
- Elementary manipulations of vectors and matrices (of int and float)

Exercise 1.

Skills: reading / writing files, manipulation of matrices
Category: verification and selection problems (Problem solving with arrays - complex problems)

Detection of black regions

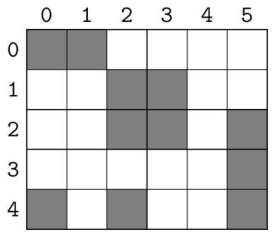
A text file contains an array of integers (either 0 or 1) with the following format:

- the first row of the file specifies the real dimensions of the matrix (number of rows `nr` and number of columns `nc`, separated by a space). Assume that both values are at most 50
- each of the subsequent rows reports the `nc` values corresponding to a row of the matrix, separated by one or more spaces
- each cell of the matrix can either contain the value 0 (associated with the white color) or the value 1 (associated with the black color)
- the black cells are organized into homogeneous rectangular regions, where each black region is either surrounded by a frame of white cells, or by the edge (s) of the matrix. To this end, you should consider the adjacent cells only along the four main cardinal points (North, South, West, East) and neglect the diagonals.

Write a C program that:

- **reads the matrix from the input file**. You can assume that the file does not contain any format errors (i.e., all the black rectangles respect the given formatting constraints).
- **identifies the largest black regions**, respectively in terms of (1) height, (2) width and (3) area. In case of tie, the program should report just one of the identified regions that meet the corresponding criterion.
- for each of the three regions identified at the previous point, **outputs the following characteristics on the screen**: coordinates (row,column) of the upper left corner, height, width and area.

Example of input file:	Corresponding map:

`5 6`
`1 1 0 0 0 0`
`0 0 1 1 0 0`
`0 0 1 1 0 1`
`0 0 0 0 0 1`
`1 0 1 0 0 1`
	

Output on the screen:

```
Max-height region: upper left corner=(2,5), height=3, width=1, area=3
Max-width region: upper left corner=(0,0), height=1, width=2, area=2
Max-area region: upper left corner=(1,2), height=2, width=2, area=4
```

Exercise 2.

Skills: iterative sorting algorithms, empirical complexity analysis

Evaluation of sorting algorithms

Consider the following sorting algorithms to sort vectors of integers in ascending order:

- Selection Sort
- Insertion Sort
- Shell Sort

Write a C program that acquires numerical sequences from a file (`sort.txt`) and calls all the above-mentioned sorting algorithms on each sequence, printing the following output on the screen:

- the number of swaps
- the number of iterations of the outer loop
- per each iteration of the outer loop, the number of iterations of the inner loop
- the total number of iterations.

The file `sort.txt` has the following format:

- the first line reports the number `S` of numerical sequences that will follow
- in the following `S` lines, one per line:
  `<length> <sequence>`
  where `<length>` is a positive integer (at most 100) representing the length of the sequence reported in the current line, and `<sequence>` is a sequence of `<length>` integer values separated by a space.

Example of file `sort.txt`:

```
4
5 1 2 3 4 5
5 1 2 3 4 0
5 5 4 3 2 1
5 1 5 2 4 3
```