PROGRAMMING METHODOLOGY - LAB TASK 2022/23

Lab Task 1. COMPLEXITY.

Session 1. Image Processing. Computing Theoretical Complexity.

It is clear that image manipulation is on the rise. Therefore, it is important that the software that manipulates images should be efficient. In this task you are going to analyse the efficiency of some existing methods of basic image processing. Therefore, you should do the following:

1. Implement a program that invokes the following methods, the code for those methods is provided in the file *Auxiliar.java*:
    a. Method **GenerarImagenGrises**(String ImagenEntrada, String ImagenSalida): takes as input the name of the ImagenEntrada file, which contains an image, transforms it to grayscale, and saves it in another file, named ImagenSalida. The name of the output image will be the same as the input image with the addition of the ending _g, e.g. if the input is "320x214.png" the output will be "320x214_g.png". It must be checked if the output image is represented in grayscale.
    b. Method int[] **HistogramaImagen**(String Ruta): receives as input the path to a file containing an image, transforms the image to grayscale, calculates its histogram and provides it as output.
    c. Method **ImprimeHistograma**(int[] Histograma): receives as input an unidimensional array of integers containing the histogram values and displays them on the screen.
    d. Method **GenerarImagenOrdenandoColumnas**(String ImagenEntrada, String ImagenSalida, int Metodo): transforms the image from the ImagenEntrada file to grayscale and sorts its columns in ascending order, according to the bubble algorithm if Metodo is 0, or according to the Quicksort algorithm if Metodo is 1. Once all the columns have been sorted, it writes the information to the ImagenSalida image file. The name of the output image will be the same as the input image with the addition of the ending "_b", if the sorting method is bubble, and the ending "_q" if the sorting method is Quicksort. For example, if the input is "320x214.png" the outputs will be "320x214_b.png" and "320x214_q.png", respectively.

    To work with the methods, you are provided with five image files of different resolutions, named "**320x214**.png", "**640x360**.png", "**640x427**.png", "**1024x1024**.png" and "**1536x1536**.png". They are available in Moodle in the file Imagenes.zip. In addition, you can make use of the methods for sorting and reading data provided in the files **Ordenar**.java and **leer**.java, respectively, together with the file **Auxiliar**.java in **Auxiliares**.zip.

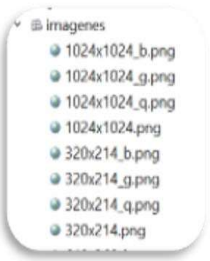
2. Compute, in a **theoretical** way, the complexity of each of the above methods.

Session 2. Calculation of Empirical Complexity and Graphical Representation.

You must do the following:

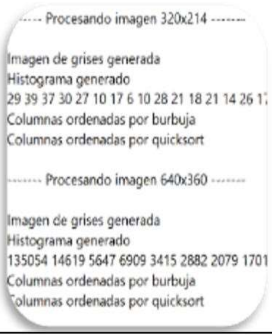
1. Create a program to empirically calculate the complexity of the methods used in Session 1. The execution time will be calculated in milliseconds or nanoseconds, depending on the user's choice. This option is chosen at the beginning of the program. The static methods **nanoTime**() and **currentTimeMillis**() of the System class can be used for the calculations, providing a long type value representing the current system time in nanoseconds and milliseconds, respectively.
2. Determine, each team member on her/his own computer, the execution times of the methods specified above, taking as input for each of them the images "320x214.png", "640x360.png", "640x427.png", "1024x1024.png" and "1536x1536.png".
3. Generate a .csv (comma separated text) file with the results.
4. Open the file with a spreadsheet program, such as MS Excel, and generate a graph with the data from the .csv file.
5. Compare the results obtained by each member of the team or even in different executions.
6. Justify, in a reasoned way, if the empirical results coincide with the theoretical ones.

The following show the contents of the image folder after the run **(1)**, part of a run of the software on a computer **(2),** and the representation of the results file together with the associated graph with MS Excel **(3)**.



```
imagenes
    1024x1024_b.png
    1024x1024_g.png
    1024x1024_q.png
    1024x1024.png
    320x214_b.png
    320x214_g.png
    320x214_q.png
    320x214.png
```

(1)) Folder with images generated



```
----- Procesando imagen 320x214 -------

Imagen de grises generada
Histograma generado
29 39 37 30 27 10 17 6 10 28 21 18 21 14 26 1
Columnas ordenadas por burbuja
Columnas ordenadas por quicksort

------- Procesando imagen 640x360 -------

Imagen de grises generada
Histograma generado
135054 14619 5647 6909 3415 2882 2079 1701
Columnas ordenadas por burbuja
Columnas ordenadas por quicksort
```

(2) Partial view of the execution results. Please, note some values of the histogram are shown.

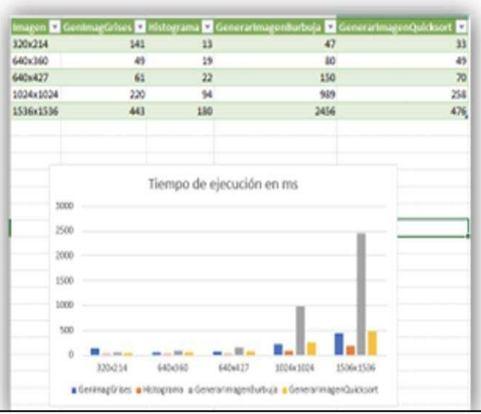


Imagen	GenImagGrises	Histograma	GenerarImagenBurbuja	GenerarImagenQuicksort
320x214	141	13	47	33
640x360	49	19	80	49
640x427	61	22	150	70
1024x1024	220	94	989	258
1536x1536	443	180	2456	476

(3) Results in Excel and Graph generated.