LAB TASK 1. COMPLEXITY.

Programming Methodology

Universidad de Castilla-La Mancha

Escuela Superior de Informática de Ciudad Real

Goals

Session 1. Image Processing. Computing Theoretical Complexity.

Session 2. Calculation of Empirical Complexity and Graphical Representation.

Introduction

- 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**.

Goals

Goals

- To perform an **empirical** and **theoretical** study of the **complexity** of a set of algorithms applied to **image processing**.
- **Display** the results in **graphs** in a **spreadsheet** to facilitate their **analysis**.

Session 1. Image Processing. Computing Theoretical Complexity.

Tasks.

- Implement a program that invokes the methods described below. The code of these methods is provided in the file Auxiliar.java.
    1. The **pictures** 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**.
- **Compute**, in a **theoretical** way, the **complexity** of each of the above methods.

GenerarImagenGrises

- 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**.

HistogramaImagen

- 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.

ImprimeHistograma

- Method **ImprimeHistograma**(int[] Histograma): receives as input an **unidimensional array** of integers containing the **histogram** values and **displays** them on the **screen**.

GenerarImagenOrdenandoColumnas

- 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.

Imput Pictures. Imagenes.zip i

Figure 1: Resolution 320x214. Photo of Namroud Gorguis on unsplash.

Imput Pictures. Imagenes.zip ii

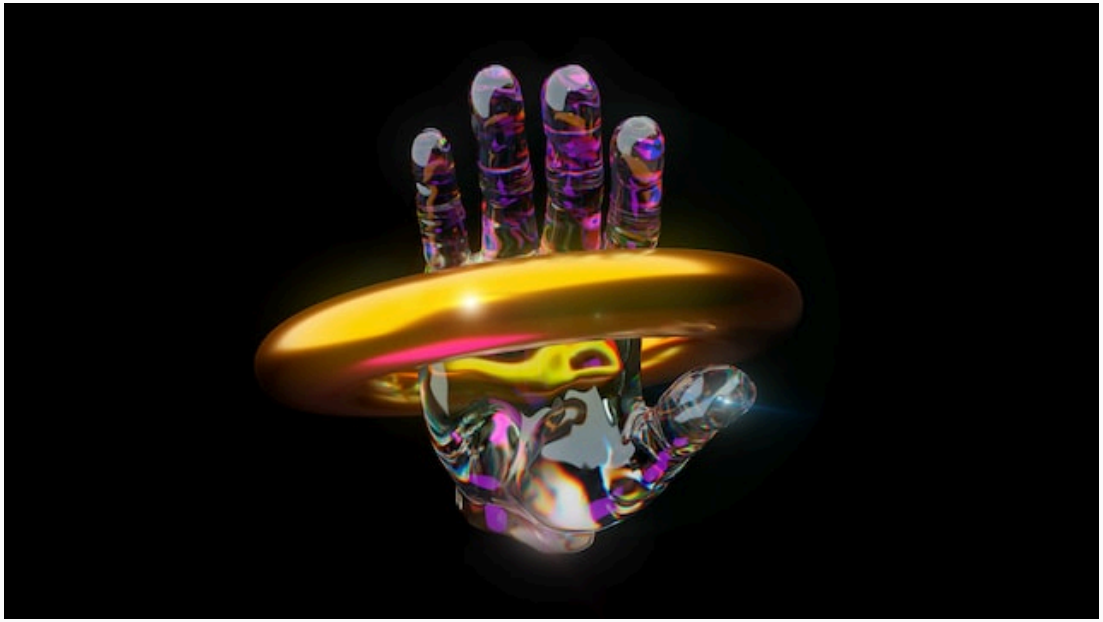

Figure 2: Resolution 640x360. Photo of Axel Ruffini on unsplash

Imput Pictures. Imagenes.zip iii

Figure 3: Resolution 640x427. Photo of Blake Wisz on unsplash

Imput Pictures. Imagenes.zip iv

Figure 4: Resolution 1024x1024. Generated by Luis Jiménez on Midjourney.

Imput Pictures. Imagenes.zip v

Figure 5: Resolution 1536x1536. Generated by Luis Jiménez on Midjourney.

Session 2. Calculation of Empirical Complexity and Graphical Representation.

Computational Complexity

- Create a **program** to **empirically calculate** the complexity of the **methods** used in Session 1.
- Determine, **each team member on her/his own computer**, the execution times of the methods
- The **input images** are: ”320x214.png”, ”640x360.png”, ”640x427.png”, ”1024x1024.png” y ”1536x1536.png”.

Execution time of the methods i

- 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.

Execution time of the methods ii

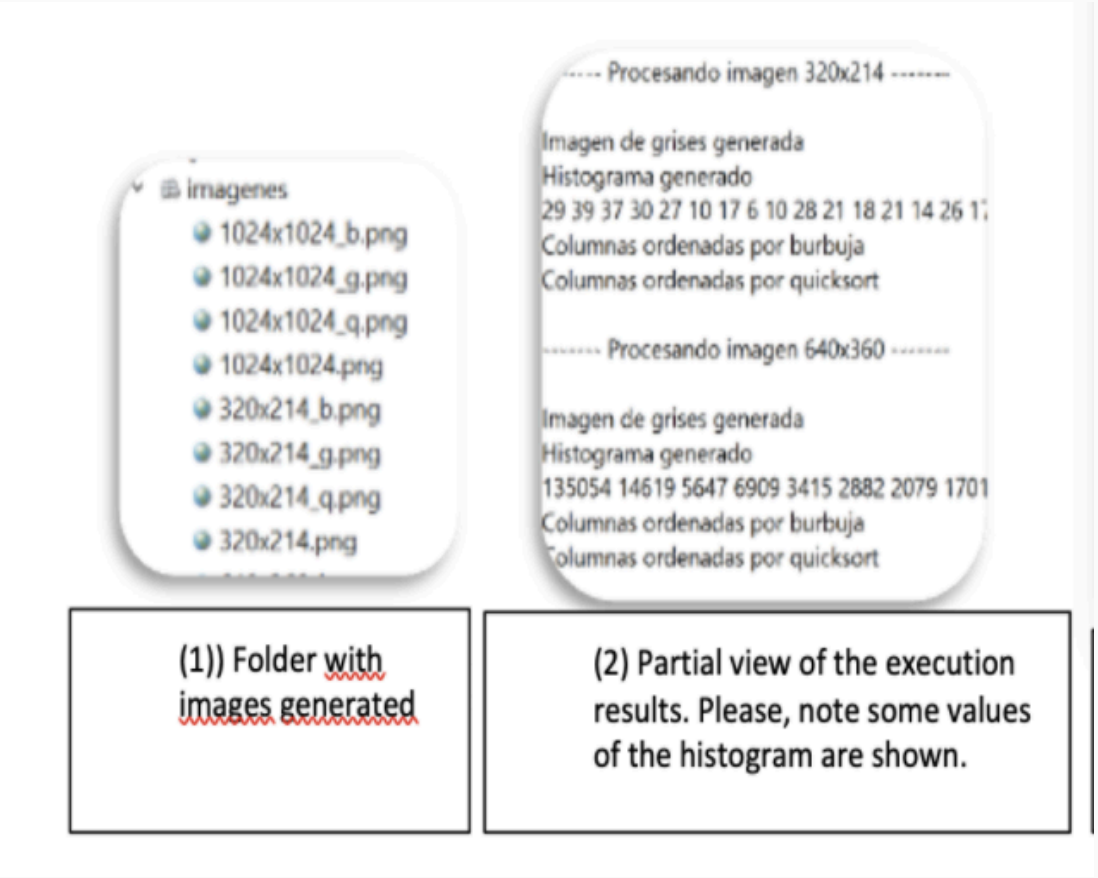

Figure 6: Output of the programa and new images generated

Results in Excel i

- **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 **given images**.
- **Generate** a **.csv** (comma separated text) **file** with the results.

Results in Excel ii

- **Open** the file with a **spreadsheet** program, such as MS Excel, and generate a **graph** with the data from the **.csv file**.
- **Compare** the results obtained by **each member** of the team or even in **different executions**..
- **Justify**, in a reasoned way, if the **empirical** results coincide with the **theoretical** ones.

Results in Excel iii

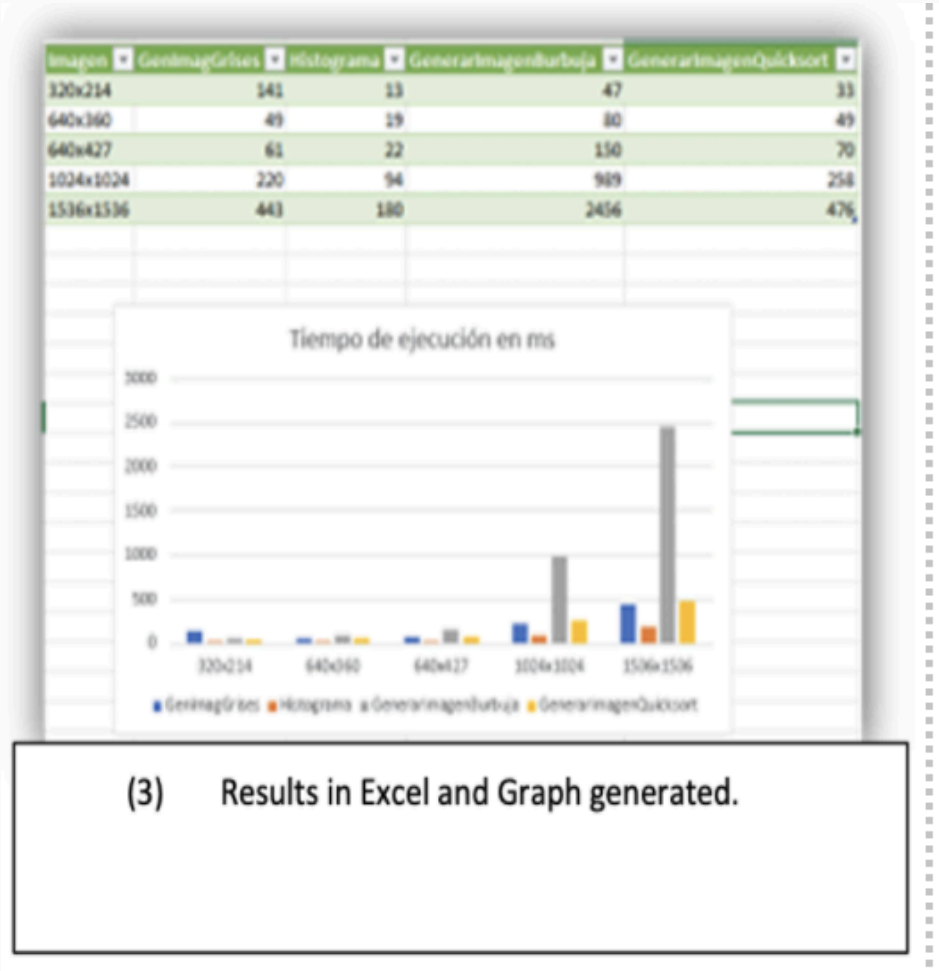

Figure 7: Graphs in Excel