ALGORITHMICS - Bachelor of Software Engineering, 2021

LAB GUIDE. SESSION 0

GOALS:

- **Work with *benchmarks***
- **Set the working environment**
- **Working with matrices**

1. Benchmarking

A. Basic concepts of benchmarking

We will review some basic concepts of how to perform performance measurements.

A **benchmark** or *test bench* is a set of experiments that is performed to evaluate the performing of a system or a part of it. A typical example would be the execution of a benchmark in several computers to know which is faster or more powerful.

To make an evaluation of a system we can distinguish the following steps:

- **Analyze the factors that can influence the performance**. The factors that influence the speed of a computer would be some like the RAM, the operating system, the hard disk, the CPU, etc.

- **Selection of performance metrics**. Choose the magnitude that you want to measure. It can be a magnitude **smaller is better** as "the time it takes to run a program" or a magnitude **larger is better** as the "number of operations per second" of a CPU. It is also usual to execute several *benchmarks* and weigh the measurements to a value.

- **Perform performance experiments**. Different configurations are usually made with all the factors make constant except one that is changed. For example, a computer with Windows 10, 16 GB of RAM, 128 MB SSD hard disk may be tested with different CPUs.

In this lab, we will see several factors that make it difficult to measure the execution times of a program. The main objective is to introduce a series of guidelines that should be followed whenever performance measurements are made. Given that in this course we will study **the time complexity** of different programs and algorithms, we will analyze the **execution time** of a program versus the **size of the problem**.

In each of the tasks that make up this lab, the results and questions that you must answer will be marked in this blue color. So, please, answer them in a separate document to later deliver it.

B. Activity 1: Power of the CPUs

In this activity, we will run the same program on several computers to analyze the time it takes to run it. Regarding the factors of a computer that influence the execution time of a program, we will focus on one of the most important: its **CPU**.

Task 1

1. Find the model of the processor of your computer. For this, the simplest thing is to open the details of the system provided by Windows (`Win + Pause`). **Write down the processor model and the system memory.**

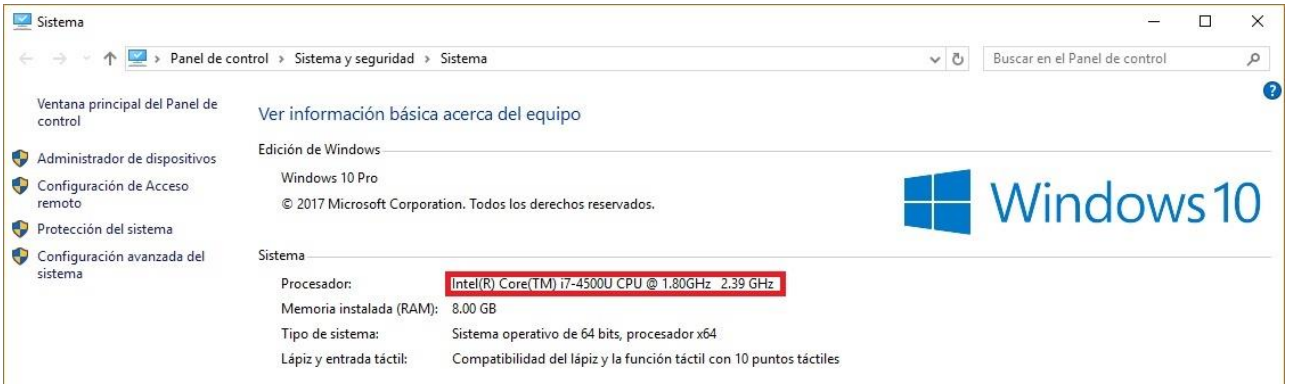

Figure 1. System information

2. Look for that processor model on the **User Benchmark** page (http://cpu.userbenchmark.com/)

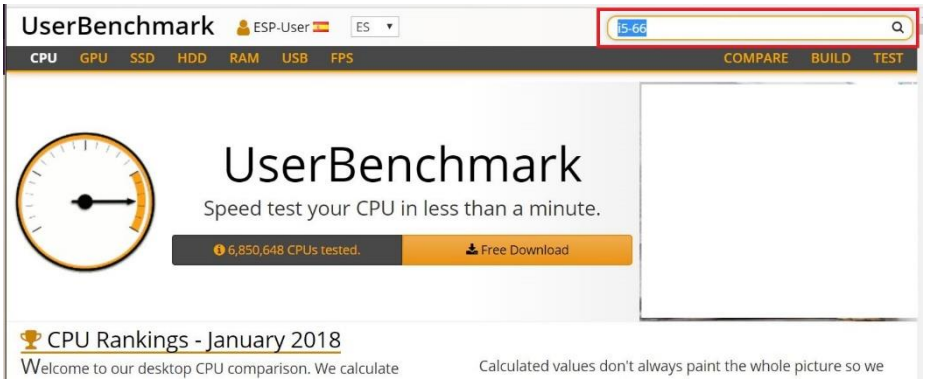

Figure 2. UserBenchmark: Searcher of CPUs

3. **Find and take note of the average index of integer and float operations per unit of time** (SC Mix Avg) performed by your processor model.

Figure 3. UserBenchmark: Details of a CPU

4. Compile and run the `Benchmarking1` program. **Write down the time it took to execute.**

5. Calculate the **approximate index of integer and float operations performed by the program**. To do this, we multiply the execution time by the *SC Mix Avg* value for that processor.

Task 2

With the data that is provided (program execution times on different machines and CPU models), perform the following tasks:

1. Complete the following table with the execution times and *SC Mix Avg* of each CPU. Then, calculate the index of integer/float operations. **Important Note**: Although the resulting index of operations is theoretically the same for all configurations, in practice there will be certain differences. **Record your results**.

2. **Extend the table with data from other computers to which you have access** (for example, your own computer).

#	CPU	milliseconds	SC Mix (avg)	Operations (aprox.)
1.	i7-4500U	285		
2.	i3-3220	267		
3.	i5-4590	219		
4.	i7-4790	207		
5.	Intel Pentium Gold G5400	215		
6.				
7.				

Table 1. CPUs comparison

Conclusion

Looking at the results in milliseconds, **do you think you could mix values from different CPUs in the same analytical study of the execution times of an algorithm?**

C. Activity 2: Influence of the operating system

Modern operating systems include many features that make it difficult to measure programs. In this activity, we will analyze the influence of the energy management policies of the processor and the simultaneous execution of several processes on the same CPU. Although both factors are closely

related to the features of the processor, much of the responsibility for managing these resources lies with the **operating system** through the energy plans and the process scheduler.

Task 1

1. Open the Task Manager and go to the CPU tab.
2. Open the Windows power configuration: `Control Panel\Hardware and sound\Energy options`
3. Change between the different plans: *High performance*, *Balanced* and *Economizer*. See how the CPU frequency varies.

Task 2

Complete this task with the program `Benchmarking1` from the previous activity.

- Sequential execution:
    1. Run the program multiple times with the `run.cmd` script. Do not do any other activity that may consume many resources (use of the browser, compilers, etc.). See how the execution time varies slightly from one execution to another.
    2. Perform the same tests with different energy plans.
- Parallel execution:
    1. Run the **cpuburn.exe** program. This program consumes 100% of the CPU so it is possible for the computer to respond more slowly (available in https://patrickmn.com/projects/cpuburn/).
    2. Run the program multiple times with the `run.cmd` script. Check whether the execution times are like those of sequential execution.
    3. When you finish, make sure you stop the *cpuburn.exe* file.

Conclusions

Answer the following questions:

1. **Which energy plan do you think is the most appropriate for making measurements?**
2. **If you had to perform a very long experiment,** could **you use the computer to, for example, watch a YouTube video in the meantime?**

3. **Do you think it is convenient to make several measurements simultaneously on the same computer?**

2. Java installation on the computer

As we are going to use the Java programming language for the purposes of this course, the most important thing we need is the JDK (Java Development Kit). It is quite possible that the JDK is already installed on the computer. For example, the JDK for Java 8 in a Windows system could be installed in the path: **C:\Program Files\Java\jdk1.8.0_131** or similar. Just in case your computer does not have installed the JDK, you should search for it on the Web – The official website can be reached at http://www.oracle.com/technetwork/java/javase/downloads/index.html.

After that, it is essential that you set the Java environment Path in your system. That way, you can invoke Java commands (e.g., for the compiling process) from different sources such as the command line console or the Eclipse IDE. The best way to know whether you already have it configured is to open a new command line terminal:

- Click **Start**
- Type `cmd` and press `enter`
- Type `javac` and press `enter`
- If you get an error, you need to configure the Java environment Path. If not, you can skip the next step.

Configuring the Java environment path:

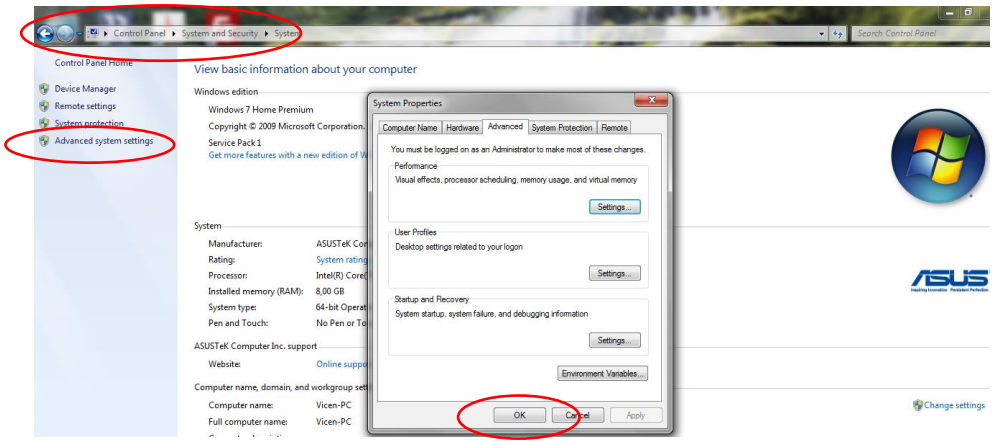

Figure 4. System properties

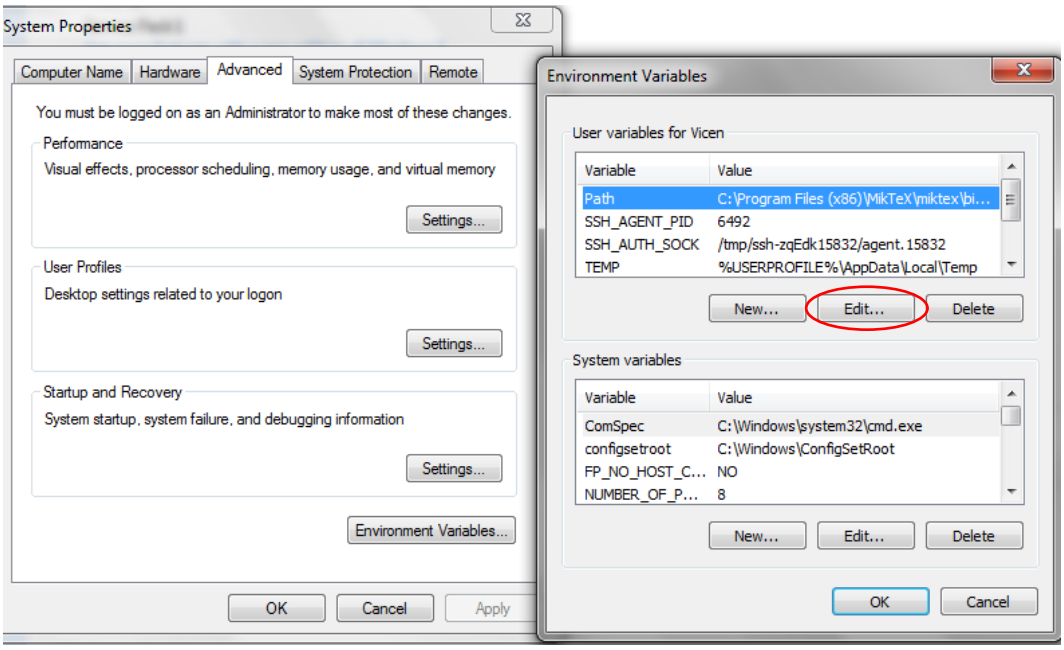

Figure 5. Environment variables

Finally, add the path in which you have installed the JDK's bin directory:

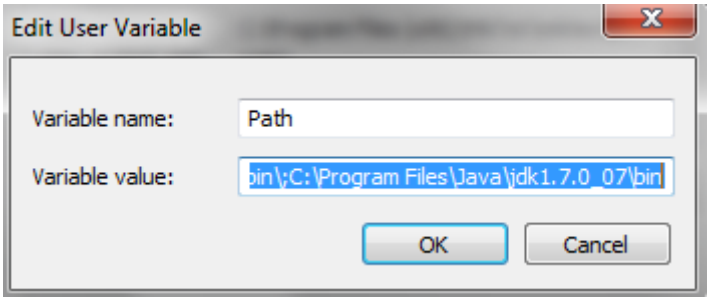

Figure 6. Edit user variable

3. Integrated development environments (IDEs)

In addition to the basic way (directly with the JDK), you can work with a Java IDE. There are lots of them: Eclipse, NetBeans, BlueJ, DrJava, jGRASP, JBuilder, JCreator, Kawa, etc. I strongly recommend you use the Eclipse IDE as it is free and used in other courses during the Degree and Master programs in our school. In addition, it is broadly used in both academia and industry.

4. Programming with Java in the command line

The basic execution of programs in Java is done from the Windows command line (DOS window). To use such a window, you should execute the executable from the start menu: `cmd`

When we want to change the hard disk drive, we just must type:

- `x: //it will open unit x:`

When we are in a folder and we want to place commands in a child folder we will do:

- `cd child //name of the folder we want to go`

To go from a folder to the parent (to the previous one hierarchically speaking):

- `cd ..`

To go to the root folder:

- `cd /`

Instead of using the instructions from the previous "Configuring the Java environment path" section, another way to configure the environment is to create a batch file (**.bat**) with the **PATH** of the computer with an editor (e.g., Notepad). That batch file should be executed every time we open a DOS window.

- The `SET PATH` command lets you know the currently active PATHs.

In that file, it is also possible to include information about the **CLASSPATH** (an option supported on the command line or using an environment variable that tells the Java Virtual Machine where to look for packages and user-defined classes when running programs).

- The `SET CLASSPATH` command lets you know the currently active CLASSPATH.

The problem with this is that if the DOS window is closed, the established paths disappear, so when we open another new DOS window there is a need to re-establish the paths.

When we create a Java class, it is usually part of a package (set of Java classes with a common purpose). The first valid statement of the class file specifies precisely in which package it is the class. In addition, the folder in which that class is stored must exactly match the package name.

To edit the files, any program editor is valid (even the Windows Notepad, in which case you must save the program with the option "All Files" and the **.java** extension, so that the **.txt** extension is not automatically added). In the labs of the school there are also installed other more powerful editors like Notepad++ and Sublime Text.

5. Programming with Java in the Eclipse environment

One of the most used environments in the Java developer community is Eclipse. This integrated development environment allows you to perform all development-related tasks from a common interface: coding, code analysis, execution and debugging.

In this environment, the minimum development unit is the **Project**, so we must create a new Java Project to start working with the code. Subsequently, you just need to copy & paste the folder corresponding to the package or packages that includes the source code of the Java classes to the `src` folder within the project.

One of the great advantages of the Eclipse environment is that it performs an automatic compilation when changing and saving the code and informs the user of possible errors using marks, both by highlighting the code involved and by establishing an icon in the margin, which provide extra information about the error or warning if we place ourselves on it with the cursor.

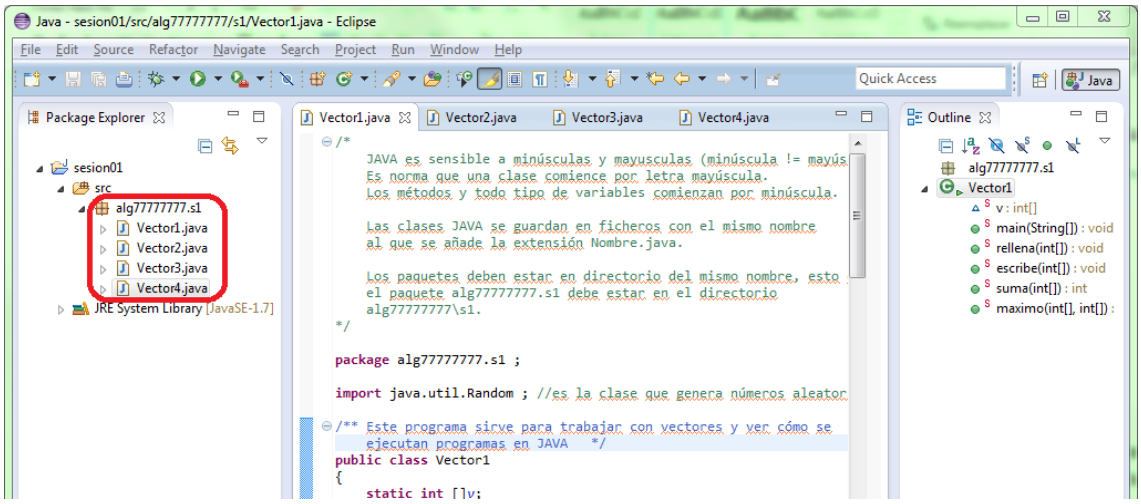

Figure 7. Eclipse workspace

To execute and debug the code, we must use the corresponding icons in the toolbar, or the context menu related to the project and use the option "**Run as ...**". To perform this operation, all the changes made to the project files must be saved and the environment will be responsible for performing the necessary operations to execute it. There are several execution modes; the one that interests us is "**Java Application**" that executes Java classes without any container or server.

In addition, Eclipse allows establishing the arguments that accompany the call to the main class to execute it. This can be done through the option "**Run configurations ...**" which can be accessed either from the toolbar execution button or from the context menu (**Right mouse button → Run As → Run Configuration → Arguments → Program Arguments**). We should remember to select in the menu on the left the related class and type the values on the tab on the right called "**Arguments**". Here we can enter the values for the arguments in the "**Program arguments**" field separated by spaces.

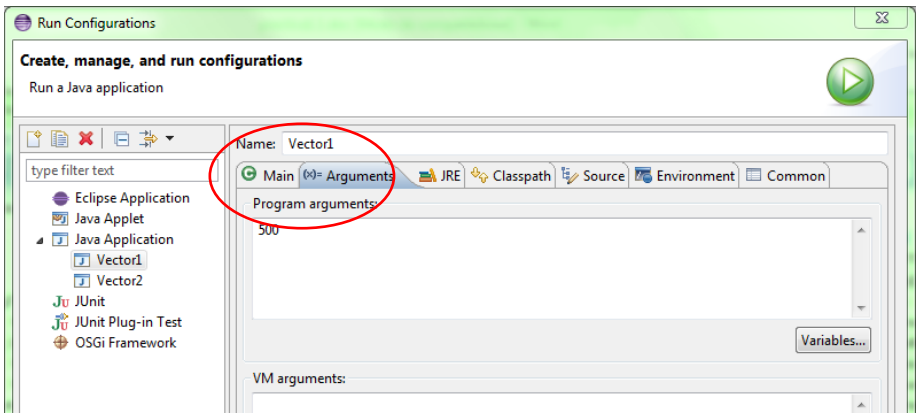

Figure 8. Execution of a program with arguments

6. Working with matrices

In the future sessions we will usually work with multi-dimensional vectors and matrices, with integers or other data types. Since these data structures are simple and easy to understand, we will use them to operate with a huge number of elements. However, to test our algorithms and implementations we should start using smaller problem sizes.

In this section you should create a Java class called `MatrixOperations.java` in the `algstudent.s0` package of your project. This represents a two-dimension square matrix. Thus, you must implement the following public methods:

- `MatrixOperations(int n, int min, int max)`. Creates a new matrix of size `n x n` and fills it with random values. These random values must be parameterizable between a maximum (`max`) and a minimum (`min`) value.
- `MatrixOperations(String fileName)`. Creates a matrix using data of the file provided as parameter. This file must have 1 integer number as the first line. Following lines contain `n` values to represent every element of the matrix row. Each of the values will be separated by a tabulator. Example:

```
10
1	1	4	1	2	2	2	2	3	4
2	4	1	1	3	4	2	1	4	3
1	4	4	4	3	2	2	3	4	3
3	2	2	3	2	1	1	2	3	3
2	3	3	4	1	1	4	3	2	3
1	2	2	3	1	4	3	2	1	2
1	2	3	3	3	1	4	3	4	3
4	3	3	4	4	2	1	2	4	2
4	2	2	3	2	1	3	1	1	1
2	4	3	2	1	3	3	2	1	1
```

- `getSize()`. Returns the matrix size (`n`).
- `write()`. Prints in the console all the matrix elements.
- `sumDiagonal1()`. Computes the summation of all the elements of the matrix diagonal. This implementation must iterate over all the matrix elements, but only sums appropriate elements. So, the complexity is quadratic.
- `sumDiagonal2()`. Computes the summation of all the elements of the matrix diagonal. This second version should only consider the elements of the main diagonal. So, the complexity is linear.

- `travelPath(int i, int j)`. Given a matrix with integer numbers between 1 and 4, this method iterates through the matrix starting at position `(i, j)` according to the following number meanings: **1 – move up; 2 – move right; 3 – move down; 4 – move left**. Traversed elements would be set to -1 value. The process will finish if it goes beyond the limits of the matrix or an already traversed position is reached. To make sure your code works, create a text file with the previous example indicated in `MatrixOperations(String fileName)` and test it. For that file, the final output for a call `travelPath(3, 0)` should be something like the following:

```
1     1     4     1     2     2     2     2     3     4
2     4     1     1     3     4     2     1     4     3
1     4     4     4     3     -1    -1    -1    4     3
-1    2     2     3     -1    -1    1     -1    -1    3
-1    -1    3     4     -1    1     4     3     -1    -1
1     -1    -1    -1    -1    -1    3     2     1     -1
1     2     3     -1    3     -1    -1    3     4     3
4     3     -1    -1    4     -1    -1    2     4     2
4     2     -1    -1    -1    -1    3     1     1     1
2     4     3     -1    -1    3     3     2     1     1

Number of movements = 32
```

TO DO:

A. Work to be done

- An Eclipse `algstudent.s0` **package** in your course project. The content of the package should be the `MatrixOperations.java` file.
- A **PDF document** using the course template. The activities of the document should be the following:
  - **Activity 1. Power of the CPUs**
    - Provide an answer to the 4 questions marked in blue color in activity 1 (task 1).
    - Provide an answer to the 2 questions marked in blue color in activity 1 (task 2).
  - **Activity 2. Influence of the operating system**
    - Provide an answer to the 3 questions marked in blue color in activity 2.

B. Delivery method

You should include in a Java project a new `algstudent.s0` package with the following content inside it:

- The requested source
- The requested PDF document called `session0.pdf` with the corresponding activities.

Deadlines:

- The delivery of this lab will be made on the same date as the following ones. More instructions will be given in the coming weeks.