Algorithms and data structures

Unit Laboratory 1 - 2019-2

Christian Flor - A00355624, Daniel Villota - A00356255, Santiago Zuñiga - A00352139

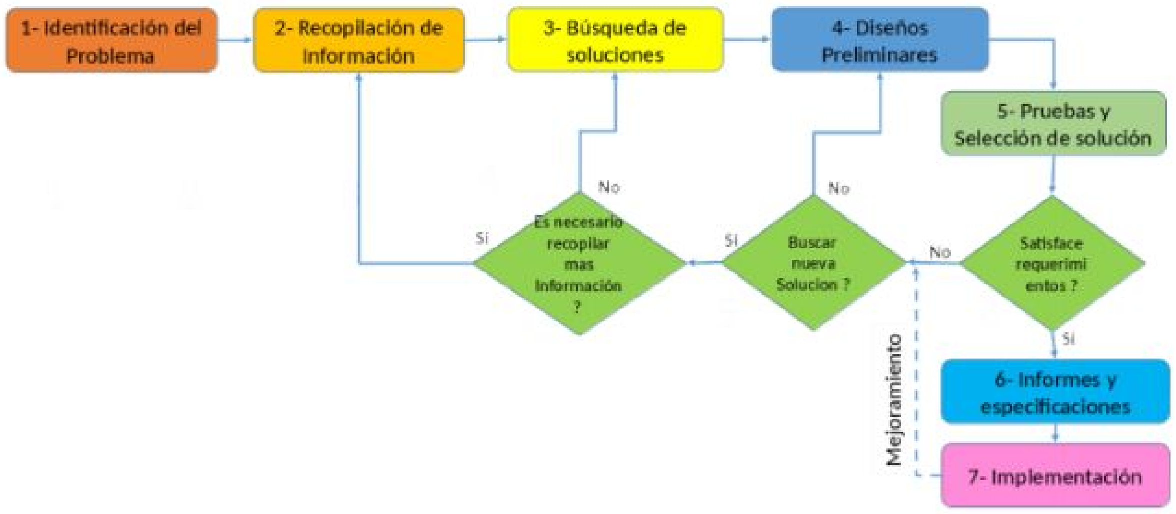
Space War.

Context of the Problem.

For the past 4 years, Venus has been at war with Mars disputing the territorial sovereignty of the 53 moons of Saturn and its rings. The earth has allied itself with Venus and has sent our team to help create a plan that will help them win.

Solution development.

To solve the previous situation, the Engineering Method was chosen to develop the solution, following a systematic approach in accordance with the problematic situation.
Based on the description of the Engineering Method of the book "Introduction to Engineering" by Paul Wright, it presents the following flow chart, whose steps we will follow in the development of the solution.


1- Identificación del Problema
2- Recopilación de Información
3- Búsqueda de soluciones
4- Diseños Preliminares
5- Pruebas y Selección de solución
Satisface requerimientos ?
Sí
No
Buscar nueva Solucion ?
Sí
No
Es necesario recopilar mas Información ?
Sí
No
6- Informes y especificaciones
7- Implementación
Mejoramiento

Step 1. Problem Identification

The specific needs of the problematic situation as well as its symptoms and conditions under which it must be resolved are specifically recognized.

<u>Identification of needs and symptoms</u>

- ➢ Identify the ships of Mars since they are strategically located secretly and invisible to the troops of Venus. In addition, it is possible to calculate the future coordinates of the ships with the positions of previous wars. (Using the matrix multiplication technique to find the exact coordinates)
- ➢ Matrix multiplication technique must be done quickly on very large matrices.

- ➢ A graphical user interface that allows you to enter the values of the number of rows and columns of the two matrices to be multiplied (battle board)
- ➢ A graphical interface that allows you to randomly generate the values of the positions of each matrix. It must allow to indicate if the numbers to be generated must be all different or may have repeated.
- ➢ A graphical interface that allows generating a random list of matrices (they can be more than 2 matrices, but their dimensions must be compatible in order to perform the requested operation) to be subsequently multiplied.
- ➢ A graphical interface that shows the result of multiplication with the exact positions of the Mars troops..

Definition of the problem

Venus requires the creation of a software plan that allows them to locate Mars fleets to attack them.

Step 2. Information Collection

In order to have total clarity in the concepts involved, a search is made of the definitions of the mathematical terms related to the problem posed. It is important to perform this search in recognized and reliable sources to know which elements are part of the problem and which are not.

Definitions

Source:
https://es.wikipedia.org

Matrix

An array is a two-dimensional array of numbers. Since both the sum and the product of matrices can be defined, it is generally said that they are elements of a ring.
They can be added, multiplied and decomposed in various ways, which also makes them a key concept in the field of linear algebra.

Matrix Multiplication

The multiplication or product of matrices is the operation of composition carried out between two matrices, or the multiplication between a matrix and a scalar according to certain rules.
Given two matrices A and B, such that the number of columns in matrix A is equal to the number of rows in matrix B; that is to say:

$$A := (a_{ij})_{m\times n} \ y \ B := (b_{ij})_{n\times p}$$

the multiplication of A by B, which is denoted $A \cdot B,\ A \times B,\ A \circ B$ or simply AB, the result of the product is a new matrix C:

$$C = AB := (c_{ij})_{m\times p}$$

where each item c_{ij} it is defined by:

$$c_{ij} = \sum_{r=1}^{n} a_{ir} b_{rj}$$

that is to say:

$$C = AB = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & \cdots & b_{1p} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{np} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + \cdots + a_{1n}b_{n1} & \cdots & a_{11}b_{1p} + \cdots + a_{1n}b_{np} \\ \vdots & \ddots & \vdots \\ a_{m1}b_{11} + \cdots + a_{mn}b_{n1} & \cdots & a_{m1}b_{1p} + \cdots + a_{mn}b_{np} \end{pmatrix}$$

Matrix multiplication is very useful for solving systems of equations of many variables, since they are very comfortable to be implemented by a computer. The numerical calculation is based largely on these operations, as are applications such as MATLAB and Octave. It is also currently widely used in the calculation of microarrays, in the area of bioinformatics.

Positive integers

An integer is an element of the numerical set that contains natural numbers $N = \{1, 2, 3, 4, \cdots\}$, its opposites and zero preceded by a plus sign. «+».Whole numbers can be added, subtracted, multiplied and divided, following the model of natural numbers by adding rules for the use of the sign.

Prime number

It is a natural number greater than 1 that has only two different divisors: himself and the. The number 1, by agreement, is not considered as cousin or compound.
The property of being a prime number is called primality.
In algebraic number theory, prime numbers are known as prime rational numbers to distinguish them from prime Gaussian numbers.

Random Number Generation

(RNG) is a computer or physical device designed to produce sequences of numbers without apparent order. It is a device that generates a sequence of numbers or symbols that cannot be predicted reasonably better than by a random possibility.
Random number generators can be true hardware random number generators (HRNG), which generate genuinely random numbers, or pseudorandom number generators (PRNG) that generate numbers that appear random, but are actually deterministic, and can be reproduced if the PRNG status is known.

There are several computational methods for generating pseudorandom numbers. All do not achieve the objective of true randomness, although they may meet, with greater or lesser success, some of the statistical tests of randomness aimed at measuring how unpredictable their results are (that is, to what extent their patterns are discernible).

- ❑ Computational methods

Most computer-generated random numbers use pseudorandom number generators (PRNGs) that are algorithms that can automatically create long series of numbers with good random properties, but eventually the sequence repeats itself (or memory usage grows without limit) . The series of values generated by such algorithms is usually determined by a fixed number called seed. One of the most common PRNGs is the congruent linear generator, which uses recurrence.

$$X_{n+1} = (aX_n + b)(mod\, m)$$

The success of this type of generator of values of a random variable depends on the choice of the four parameters that intervene initially in the previous expression:

- ❖ The initial value or seed: x_0
- ❖ The multiplicative constant: a
- ❖ The additive constant: c
- ❖ The number m with respect to which the remains are calculated

These four values must be non-negative integers that meet the following condition: $x_0, a, c < m$.

The recurrence relationship can be extended to the matrices to have much longer periods and better statistical properties. In order to avoid certain non-random properties of a single congruent linear generator, several of these random number generators with slightly different values of the multiplier coefficient, a, can be used in parallel, with a "master" random number generator that selects between Several different generators.
Most computer programming languages include library functions or routines that provide random number generators. They are often designed to provide a random byte or word, or a floating point number evenly distributed between 0 and 1.

Step 3. Search for Creative Solutions

For this step, although we can think of our own solutions, and as it turns out that the problem is a classic one in mathematics, we search the web for various matrix multiplication algorithms, algorithms for finding cousins in an array of numbers and generating random numbers. The algorithms found are the following:

★ Algorithms found for matrix multiplication:

Alternative 1. *Strassen's Algorithm*

In Linear Algebra, the Strassen Algorithm (named after Volker Strassen) is used for matrix multiplication. It's time complexity is better to the standard Matrix product algorithm (O(n^3) solution), but slower than the best. It is also useful for big matrices unlike its successors (Coppersmith-Winograd Algorithm) due to the large constant factors in their running times.

Source:
https://github.com/srinivasgumdelli/Algorithms/blob/master/Strassen.java
https://es.wikipedia.org/wiki/Algoritmo_de_Strassen

Alternative 2. *Coppersmith-Winograd Algorithm*

It was the asymptotically fastest known matrix multiplication algorithm from 1990 to 2010. It can multiply two $n \times n$ matrices in $O(n^{2.375477})$ time. This is an improvement over the naive $O(n^3)$ approach and $O(n^{2.807355})$ Strassen algorithm.

This algorithm is frequently used in construction blocks in other algorithms to test theoretical time boundaries. However unlike Strassen's algorithm its not used in praxis solutions. The latter is due as it only provides and advantage when matrices so great, that they can't possibly be handled by modern hardware.

Source:
https://www.sanfoundry.com/java-program-implement-schonhage-strassen-algorithm-for-multiplicatitwo-numbers/

Alternative 3. *Naive*

Standard Algorithm that uses the linear algebra definition of matrix multiplication.
TIme Complexity: O(n^3).

Source:
https://en.wikipedia.org/wiki/Matrix_multiplication_algorithm#Iterative_algorithm

Alternative 4. *Divide and conquer Algorithm*

This is the recursive alternative to matrix product which is based in block partition. There is a version which only works with square matrices $2^n \times 2^n$ for any n, and another which works with matrices with arbitrary dimensions.We will review the latter.

Source:
https://en.wikipedia.org/wiki/Matrix_multiplication_algorithm#Divide_and_conquer_algorithm
https://en.wikipedia.org/wiki/Cache-oblivious_algorithm

Alternative 5. *Parallelized Divide and conquer*

It uses parallel technology in the Divide and Conquer algorithm to multiply non square matrices. This algorithm is not practical as it has a high cost concerning transference of data (movements).

Source:
https://en.wikipedia.org/wiki/Matrix_multiplication_algorithm#Shared-memory_parallelism

★ **Algorithms to find a prime number:**

Alternative A. *Sieve of Eratosthenes (optimized variant):*

Time complexity: O(n).

Source:
https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes ,
https://www.geeksforgeeks.org/sieve-eratosthenes-0n-time-complexity/

Alternative B. *Sieve of Atkin*

Time Complexity: O(n/log(log (n))). It is a little better than Eratosthenes alternative, but only for a reduced number of cases.

Source:
https://en.wikipedia.org/wiki/Sieve_of_Atkin
https://www.geeksforgeeks.org/sieve-of-atkin/

Alternative C. *Sundaram*

The Sieve of Sundaram has a time complexity of O(nLog(n)).

Source:
https://stackoverflow.com/questions/5235865/comparison-of-sieve-of-sundaram-and-sieve-of-atkin-for-generating-a-list-of-prim

★ **Generating random numbers in Java:**

Alternative Z. *Math.random()*

It returns a random number between 0.0 and 1.0. For an integer we remove decimals with the class *Math.floor().*

```
int value = Math.floor(Math.random()*6+1);
```

Source::
https://www.mkyong.com/java/java-generate-random-integers-in-a-range/
http://chuwiki.chuidiang.org/index.php?title=Generar_n%C3%BAmeros_aleatorios_en_Java

Alternative Y. *java.util.Random*

An instance of this class is used to generate a stream of pseudorandom numbers. The class uses a 48-bit seed, which is modified using a linear congruential formula.

```
Random r = new Random();
int valorDado = r.nextInt(6)+1;
```

Source:
https://www.mkyong.com/java/java-generate-random-integers-in-a-range/
http://chuwiki.chuidiang.org/index.php?title=Generar_n%C3%BAmeros_aleatorios_en_Java

Alternative X. *Java 8 Random.ints*

Since Java 8, the Random class has various methods that return an IntStream or a random integer stream. If the first parameter is not specified, the IntStream will be infinite. If the minimum and maximum bound are not specified, any integer number will be included (as big as it may be).

Source:
https://www.mkyong.com/java/java-generate-random-integers-in-a-range/
http://chuwiki.chuidiang.org/index.php?title=Generar_n%C3%BAmeros_aleatorios_en_Java

Alternative W. *javax.crypto.SecureRandom*

This class provides a cryptographically strong random number generator (RNG). SecureRandom must produce non-deterministic output.

Source:
https://howtodoinjava.com/java8/secure-random-number-generation/

Alternative V. *randomX*
The numbers are generated from radioactive disintegration.

Source:
http://www.fourmilab.ch/hotbits/source/randomX/randomX.html
http://www.fourmilab.ch/hotbits/source/randomX/doc/randomX/randomX.html#nextInt--

Alternative U. *Libreria de Random.org*
It is a service that produces random numbers based on atmospheric noise.

Source:
https://stackoverflow.com/questions/381037/true-random-generation-in-java
https://sourceforge.net/projects/trng-random-org/

★ **Generating random numbers without repetition Java:**

Alternative T. Choose and remove randomly from a set
*I*n our case we multiply the number of rows by columns to obtain the total number of positions that the matrix has, we put in a matrix / list the numbers of the positions and each number represents the position it has in the matrix. Then we randomly choose a valid index from the set or list to extract the position of the matrix and remove it from the set / list. The array / list size will be reduced by 1.
In this way, we simulate that we are not repeating the numbers in each position of the matrix.

Source:
http://chuwiki.chuidiang.org/index.php?title=Generar_n%C3%BAmeros_aleatorios_en_Java

Alternative S. Store the random numbers generated
The other Alternative is to generate random numbers and save them in a Java set. We generate a new random number and before taking out the number on the screen (or doing what we have to do with it), we must verify if it already exists in the set. If it exists, we discard it and generate another. If it does not exist, we use it and add it to the set.
In the code there is the possibility, possible but not probable, that the loop never ends, or takes a long time just to get 10 numbers. This will be more likely the more numbers we want to get and fewer numbers to choose from.

Source:
http://chuwiki.chuidiang.org/index.php?title=Generar_n%C3%BAmeros_aleatorios_en_Java

Step 4. Transition from ideas to preliminary design

In this step we discard unfeasible options.

★ **Algorithms for matrix multiplication.**

Alternative 1. *Strassen's Algorithm*

- Presents improvements over the traditional approach.
- It is considered efficient for large entries.
- It is restricted to multiply $2^n \times 2^n$ matrices.

Alternative 2. *Coppersmith Winograd*

- No es útil para la dimensión del problema que se va a trabajar.
- Presenta mejoras en el rendimiento pero sólo para tamaños de matriz enormes.

Alternative 3. *Naive*

- Useful to multiply non square matrices.
- Has a high time complexity..

Alternative 4. *Divide and conquer*

- Recursive form to multiply both squared and non square matrices by dividing the matrix in smaller sections..
- Its time complexity is the same as the naive approach, however it is better due to cache memory behaviour..

Alternative 4. *Divide and conquer paralelizado*

- In theory in takes advantage of the shared memory of multiprocessors.
- Has a high cost in resources due to "data movement".

★ **Algorithms to find prime numbers in an array:**

First of all, we discard the alternative C (Sundaram) and the alternative B (Atkin), the first one because its performance in terms of its temporal complexity is the worst compared to the other two both theoretically and practically. The second because its complexity can be decent but in practice the improved variant of the Eratosthenes screen turns out to be better. The careful review of the alternatives leads us to the following:

Alternative A. *Sieve of Eratosthenes (optimized variant):*

- It has a temporal complexity of O (n).
- Its implementation is simple.

Alternative B. *Sieve of Atkin*

- It has a temporal complexity of O (n / log (log (n))), it is optimal for large entries but it is not the most practical because it has greater efficiency only in a few cases, without far exceeding the optimized sieve of Eratosthenes.

Alternative C.*Sieve of Sundaram*

- It has a temporal complexity of O (nLog (n)). It has no theoretical or practical advantage.

Step 5. Evaluation and Selection

Standard

The criteria that will allow the evaluation of the solution alternatives to be able to choose the solution that best meets the needs of the problem must be defined. The criteria we chose in this case are the ones listed below. Next to each one a numerical value has been established in order to establish a weight.

Criteria 1A. Temporal complexity analysis:
1: O(n^3) , 1.5: O($n^{2.8074}$) , 2: O($n^{2.3755}$).

Criteria 1B. The algorithm is practical.

0: No , 1: Sí.
Criteria 1C. For all possible inputs in the problem, the algorithm gives a correct answer.
0: No , 1: Sí.

Evaluation
Evaluating with the above criteria the alternatives that are maintained, we obtain the following table:

Evaluation for matrix multiplication algorithms.				
	Criteria 1A	**Criteria 1B**	**Criteria 1C**	**Total**
Alternative 1. Strassen	1.5	1	0	2.5
Alternative 2. Coppersmith Winograd	2	0	0	2
Alternative 3. naïve	1	1	1	3
Alternative 4. Divide and conquer	1	1	1	3
Alternative 5. Divide and conquer parallelized	1	0	1	2

Criteria 2A. Temporal complexity analysis:
1: O(nLog(n)) , 1.5: O(n/Log(Log(n))) , 2: O(n).
Criteria 2B. The algorithm is practical.
0: No , 1: Sí.
Criteria 2C. For all possible inputs in the problem, the algorithm gives a correct answer.
0: No , 1: Sí.

Evaluation of algorithms to find prime numbers in an array.				
	Criteria 2A	**Criteria 2B**	**Criteria 2C**	**Total**
Alternative A. Optimized sieve of Eratosthenes	2	1	1	4
Alternative B. sieve of Atkin	1.5	0	1	2.5
Alternative C. sieve of Sundaram	1	0	1	2

Selection

According to the previous evaluations, alternatives 1, 3 and 4 must be selected first for matrix multiplication algorithms since they obtained the highest score according to the defined criteria. It is necessary to clarify that the final software must use the Strassen algorithm in a restricted way, since in theory it is the best algorithm for the problem but for it to deliver a correct result its inputs should only be matrices of dimensions $2^n \times 2^n$ for some n, in any other case, one of the other two algorithms that effectively work with matrices of arbitrary dimensions must be used. Secondly, for the algorithms of finding primes in an array of numbers, alternative A was chosen, since it obtained the highest score according to the defined criteria.

Conclusions.

Based on the previously mentioned criteria, the algorithm selected for matrix multiplication is that of (Divide and conquer). This is because although we know that its temporal complexity is equal to the Naive algorithm, it is better in practice because of its cache behavior, also considering the dimensions of the Martians' problem and that their results must be able to be visible on a limited space screen. On the other hand, in the case of searching for prime numbers in an array, Alternative A. (Optimized Eratosthenes Screen) was chosen because it has a temporal complexity O (n) that is more efficient than the algorithms sought. In addition, in the case of random number generation in Java, Alternative Y will be implemented (Class java.util.Random) since it is more general for what the problem needs. Finally, for the generation of random numbers without repetition in Java, the Alternative T. (Choose and delete randomly from a set) was chosen as the most efficient. Since although there is a probability that it will take a long time, in the end it always ends with the desired result and also in practice the mentioned probability is very low.

Step 6. Preparation of Reports and Specifications

1. **Specification of Functional Requirements. (in terms of input / output)**

Name	**R.F# 1. Allow to define the rows and columns of two matrices**
Summary	The system must allow you to enter the row and column number you want for the multiplication of two matrices.
Input	
Row and column value	
Output	
Result of the multiplication of the two matrices	

Name	**R.F# 2. Generate positions for each matrix**
Summary	The system must randomly generate the values of the positions of each matrix depending on what is chosen by the user.
Input	
Do you want repeated values? Yes or no	
Resultados	
Value of the positions of each matrix.	

Name	**R.F# 3. Generate a list of matrices**
Summary	The system must make a list of randomly generated matrices with different values in their positions but with the same dimension.
Input	
Any.	
Output	
Matrix list.	

Name	**R.F# 4. Mars Troops Report**
Resumen	The system must show the result of the multiplication of two matrices with the exact positions of the Mars troops.
Input	
Any.	
Output	
Matrix of Mars troops.	