Lab 3 : Polynomial complexity

Exercise1 : Matrix Multiplication

1- Write the algorithm which allows you to calculate the product of 2 matrices A and B:

$$C(n,p) = A(n,m) \times B(m,p) \qquad n,m,p \in N \text{ and } (n \geq 1, m \geq 1, p \geq 1)$$

$$\textit{The elements } C(i,j) \textit{ are calculated with the formula}:$$

$$C(i,j) = \sum_{k=1}^{p} \big(A(i,k) \times B(k,j)\big) \quad ; \; i = 1..n \text{ and } j = 1..m$$

2- Calculate the theoretical time complexity of this program in terms of n, m and p.
In the case where n=m=p (case of square matrices), give the new formulation of the complexity.

3- Calculate the memory space required to run this problem.

4- Write the corresponding program C and measure the running times T of the product of two square matrices (nxn) for a sample of data of the variable n and represent the results in the form of a table.

5- Represent by a graph the variations of time T in relation to the values of n.

6- Compare between theoretical complexity and experimental complexity. Is there agreement between the theoretical model and experimental measurements?

Exercise 2 : Searching a submatrix

Let A(n,m), B(n',m') be two two-dimensional arrays such that n'<n and m'< m.
It's a question of searching the array (matrix) B in array A.

1- Assuming that the elements of A and B are not sorted, write a function *subMat1* which searches B in A. Evaluate its theoretical time complexity.

2- Assuming that each of the lines of A and B is sorted in ascending order (see figure), write a non-naïve *subMat2* function of minimal complexity to find B in A.
Evaluate its theoretical time complexity.

3- Measure the execution times by varying n, m then n', m' and represent the results in the form of a table for the functions *subMat1* and *subMat2*.

4- Represent by a graph the results obtained in 3- of *subMat1* and *subMat2*.

2	2	2	3	5	7	8	17	24	24	54	67	76
3	4	4	5	6	6	6	8	11	12	33	81	85
12	14	23	26	26	26	31	34	44	45	52	87	90
6	6	17	24	24	54	56	61	67	81	87	90	108
2	2	2	3	5	7	8	17	24	24	54	67	76
3	4	4	5	6	6	6	8	11	12	33	81	85
12	14	23	26	26	26	31	34	44	45	52	87	90
6	6	17	24	24	54	56	61	67	81	87	90	108
12	14	23	26	26	26	31	34	44	45	52	87	90
6	6	17	24	24	54	56	61	67	81	87	90	108

Array A

24	54	56
5	7	8
6	6	6

Array B

Figure. Example of sorted arrays A and B