# Rapport de projet : Exponentiation rapide

## Année : 2016/2017

## S4 MA – IUT de Villetaneuse



## Groupe : Gatto Nathan, Wong Jason et Dos Santos Jérémy

## Tuteur : M. Bonino

# Sommaire :

INTRODUCTION ………………………………………………………………………………………………………………………………….3

1. Présentation du projet ………………………………………………………………………………………………………
2. Contexte du projet …………………………………………………………………………………………………………...
3. Objectif du projet ……………………………………………………………………………………………………………..

Développements mathématiques ……………………………………………………………………………………………………….4

Présentation des différents outils informatique …………………………………………………………………………………..

1. PHP……………………………………………………………………………………………………………………………………
2. GMP………………………………………………………………………………………………………………………………….
3. WAMP………………………………………………………………………………………………………………………………

Présentation et commentaires des résultats obtenus………………………………………………………………………….

Conclusion et commentaires personnels sur le projet………………………………………………………………………….

Annexes………………………………………………………………………………………………………………………………………………

Bibliographie………………………………………………………………………………………………………………………………………..

# Introduction

1 – Présentation du projet

Tout d’abord nous allons commencer par expliquer ce qu’est l’exponentiation rapide. C’est une méthode mathématique utilisée pour calculer rapidement de grandes puissances entières qu’on peut éventuellement mettre à un certain modulo. On l’appelle aussi « Square and Multiply ». Cette méthode est très utilisée en informatique notamment pour des méthodes de cryptage (RSA).

2- Contexte du projet

Ce projet présente des algorithmes de calcul rapide de nombres de la forme mn, où m et n sont deux entiers positifs. De tels algorithmes sont utilisés dans de nombreux domaines où l’on a besoin de manipuler de grands entiers, par exemple en cryptographie.

3-Objectif du projet

Analyser et programmer plusieurs algorithmes d’exponentiation rapide de nombres entiers (et de classes de congruence d’entiers). Le but était de concevoir plusieurs façons de calculer des exponentiations et de comparés leur rapidité et leurs complexités afin de savoir lequel est le plus optimal et donc le moins couteux.

# Développements mathématiques

1er algorithme (Écriture de Hörner) :

On souhaite faire le calcul rapide d’une puissance de la forme , pour cela, on veut tout d’abord décomposer la puissance c’est-à-dire n.On commence passer le nombre n à la base 2 (en binaire), qui correspond à un polynôme de la forme :

Où : - les correspondent au nombre de rang i donnés dans l’écriture de n. - x correspond à la base, ici on utilise la base 2. - d le nombre de chiffres dans l’écriture binaire de n.

Puis on met en facteur les x le plus possible.

Puis

On continue jusqu’à avoir l’écriture de Hörner du polynôme P

Enfin, on va utiliser cette écriture pour notre calcul de puissance. Pour cela, on remplace n par son écriture de Hörner et on calcule.

Exemple :

On veut calculer

Le nombre 9 correspond à 1001 en base 2, soit au polynôme :

Où l’on va mettre les x en facteurs :

Puis

On obtient donc où x correspond à la base utilisée, soit 2 dans notre cas.

Pseudo Code algorithme 1 : Voir annexe 1.

Question de l’énoncé :

1. A) Ici on travaille en base de 2, on a donc :

B) Le nombre d’opérations additions et multiplication par boucle ne comptant que pour une seule opération on a :

1. Pour justifier l’encadrement suivant il faut comprendre la chose suivante : quand on met un nombre en base 2, il est composé de 1 et/ou de 0. Donc grâce à la formule (\*) alors 2d est inférieur ou égale à n

Explications :

Si la puissance n = 0, la taille de la chaine bin qui est la longueur du nombre en base 2 sera 0 et l’algorithme retournera 1 car .

Sinon, on remarque que quelques soit le chiffre au rang i du nombre binaire, on fait toujours le même calcul : on met à la puissance x le résultat, on fait une multiplication du nombre m seulement si le chiffre au rang i est 1. Donc on crée une boucle dans laquelle on calcule res=res^x et on vérifie si le chiffre au rang i est 1, si c’est le cas, on fait l’opération res=res\*m.

3ème Algorithme numéro 3 (parité de la puissance)

Explication

Le troisième algorithme programmé de manière récursive, consiste à utiliser les égalités basiques comme mn = (m2) n/2 lorsque n est pair ou mn = (m2) n-1/2 lorsque n est impair. Comme l’algorithme est récursif, il fallait trouver une condition d’arrêt : on avait le choix entre n=0 ou n=1. J’ai préféré garder le second choix car calculer un nombre puissance 0 est inutile dans le sens que le résultat sera toujours 1 et ça nous évite un appel supplémentaire (sachant qu’on passe toujours à n=1 avant n=0). La condition d’arrêt de l’algorithme est donc lorsque la puissance est égale à 1, on renvoie la valeur du nombre. La partie concrète restait à programmer.

En reprenant le TP d’algorithmique avancé, j’ai écrit le code en réfléchissant à ce que je faisais. La première chose à faire est de tester si la puissance est un nombre pair ou impair. Si elle est paire, on rappelle la fonction avec comme paramètre le nombre mis au carré et la puissance divisé par 2. Dans le cas où la puissance est impaire, on fait pareil que dans le cas précédent sauf que l’on change les paramètres : on garde le nombre au carré pour le premier paramètre mais on donne l’indice de puissance – 1 divisé par 2 afin de garder une puissance entière. Cet algorithme s’enchaîne jusqu’à ce que la puissance vaille 1.

La complexité de l’algorithme est dans le pire des cas en logarithme en base 2 de la puissance. En effet, à chaque passage dans la fonction, on divise soit par 2, soit on soustrait 1 avant de diviser par 2 (ce qui est plus rapide pour terminer l’algorithme). Le pire cas de figure serait donc que la puissance donnée soit paire : on retrouverait une puissance égale à 1 après plus d’itération que si l’on avait fait -1 avant de diviser par 2. Par conséquent la complexité se déclare en combien de fois on divise par 2 successivement avant d’obtenir 1, c’est-à-dire le log en base 2.

Exemple avec 16474

On voit que l’indice de puissance est pair, donc on relance la fonction avec 1642 et 37 (74/2).

Seconde itération, la puissance est impaire. Cette fois, on appelle la fonction avec (1642)2 et 18 ((37-1)/2)

Ensuite, la puissance étant à nouveau paire, on appelle la fonction avec ((1642)2)2 et 9 (18/2)

De même avec (((1642)2)2)2 et 4 ((9-1)/2) puis avec ((((1642)2)2)2)2 et 2 (4/2)

Enfin l’algorithme arrive à son terme car le dernier appel se fait avec les paramètres suivant (((((1642)2)2)2)2)2 et 1 (2/2). Lors du prochain passage, la fonction retournera 164 car la puissance vaut 1 et on remonte la valeur aux anciens appels récursifs et le dernier résultat correspond à ce que l’on voulait au départ.

# Présentation des différents outils informatique

1. PHP



Pour réaliser ce projet nous avons décidé d’utiliser le langage de programmation PHP, PHP est un langage de programmation libre multi plateforme, principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. PHP est un langage impératif orienté objet.

PHP est un langage qui a été créé en 1994 par Rasmus Lerdorf. C’est un langage extrêmement utilisé, il a permis la création de site tels que Facebook, Wikipédia et bien plus encore.

1. GMP

De plus nous avons utilisé une librairie de fonctions PHP qui se nomme GMP, *GNU Multiple Précision*, qui permet de manipuler des entiers d’une taille arbitraire. Parmi ces fonctions nous avons notamment utilisé *gmp\_mod* une fonction qui permet de mettre un nombre de type GMP au modulo souhaiter. *gmp\_mul* une fonction qui permet de multiplier deux nombres de type GMP.

1. WAMP



Wamp est une plate-forme de développement Web sous Windows pour des applications Web dynamiques à l’aide du serveur Apache2, du langage de scripts PHP et d’une base de données MySQL. (Mamp pour OS, LAMP pour Linux). Elle nous a permis de travailler avec GMP car cette librairie est déjà installée. Nous l’avons donc utilisé dans le but de tester et voir le rendu de notre code.

# Présentation et commentaires des résultats obtenus

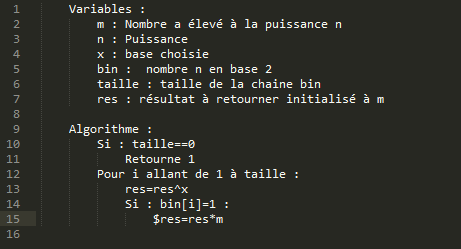
Voici un graphique représentant les temps d’exécution de chaque algorithme sur 6 exemples différents. Bien qu’on observe que les temps d’exécution sont plus importants pour le 3ème algorithme en moyenne, la différence n’est pas significative.

# Conclusion

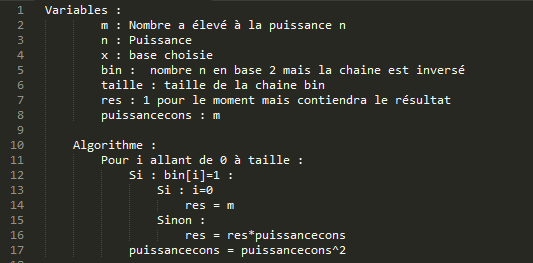
1. PHP

# Annexes

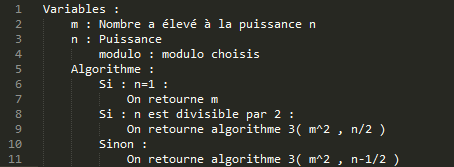
**Annexe 1 : Pseudo code** 1er **algorithme (Hornër)**



**Annexe 2 : Pseudo code 2ème algorithme (puissances successives)**



Annexe 3 : Pseudo code 3ème algorithme (parité des puissances)



# Bibliographie

Site web consultés :

<https://hal.inria.fr/inria-00540485v2/document>

<http://www.maths-algo.fr/index.php/informatique/problemes-d-informatique/arithmetique/exponentiation-rapide>

<http://php.net/manual/fr/book.gmp.php>

Cours sur la cryptographie de David Hébert.