Rapport

Jean-Didier Pailleux - Maxence Joulin - Damien Thenot - Romain Robert - Robin Feron

Test de primalité

27/12/2017

UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES
université PARIS-SACLAY

Projet de Programmation numérique

Table des matières

1 Introduction

Ce document est le compte-rendu de notre travail qui s'inscrit dans le cadre du module *Projet de Programmation numérique* du master *Calcul Haute Performance Simulation* de l'***UVSQ***. Le sujet de ce projet a été proposé par l'encadrant Sébastien Gougeau.

Ce projet découle d'un thème qui est le test de primalité, et consiste à tester si un nombre est bien premier ou non. Un nombre premier est un entier qui admet uniquement deux diviseurs distincts et positifs (1 et lui même). Les nombres premiers prennent une place importante dans le domaine des mathématiques et ont des propriétés très utiles particulièrement dans le domaine de la cryptographie. La recherche de très grands nombres premiers est devenu de plus en plus captivé par beaucoup, de nombreux tests de primalité ont pu émerger, voir évoluer, devenir plus performants, et de plus en plus rapides.

Il existe actuellement deux types de tests de primalités, les déterministes qui permettent d'établir avec certitude le résultat et les tests probabilistes qui émettent un résultat n'ont fiable avec une certaine probabilité d'erreur mais sont plus rapide que les tests déterministes.

L'objectif de ce projet consiste à implémenter plusieurs de ces tests (déterministes) et ainsi comparer leur vitesse d'exécution. Ceci et pour une exécution sur le test de primalité pour un nombre premier, et pour une exécution sur le test de n nombres premiers. De plus un test probabiliste sera programmer pour ainsi comparer la fiabilité du résultat et le temps que nécessite de faire cet algorithme.
Une partie bonus nous a été proposé lors de notre entretient avec Monsieur Gougeau qui consiste à implémenter une fonctionnalité pour déterminer si un nombre est hautement composé (c'est à dire que le nombre de ses diviseurs est supérieur strictement à tout les nombres inférieur lui).

Dans la première partie de ce document, on présentera l'architecture de notre application, illustrée par un organigramme qui a été préalablement établi (A voir si on en fait un).
Ensuite Le fonctionnement de l'application sera décrit. Après cela, on fera le bilan technique de notre projet, c'est à dire, les outils utilisés (bibliothèque), les difficultés rencontrés et autre.
Puis dans une autre partie l'analyse des résultats établis lors des tests de l'application.
Finalement, dans les deux dernières parties, on établira un bilan quant à l'organisation interne au sein du groupe et un bilan comparatif des coûts présumés de chaque test de primalité.

2 Architecture de projet

2.1 Organigramme

2.2 Tests de primalité et Hautement composé

Au cours de ce projet il nous a été demandé d'implémenter plusieurs algorithme pour déterminer si un nombre est premier ou non. Voici les différents tests utilisés pour ce travail :

- Méthode naïve le crible d'Ératosthène utilisé pour le memory bound(Déterministe).
- Méthode naïve l'algorithme d'Euclide (Déterministe).
- Pocklington (Déterministe)
- AKS 2002 (Déterministe)
- Miller-Rabin (Probabiliste).

Également, deux fonctions ont été implémenté pour les nombres hautement composés. La première est une méthode naïve qui consiste à calculer le nombre de diviseur d'un nombre n et de comparer ce nombre avec le nombre de diviseurs pour tout entier k compris entre 1 et $n-1$.
La seconde qui utilise la propriété sur la forme qu'un nombre hautement composé doit avoir. Un nombre hautement composé possèdent des facteurs premiers les plus petits possible. Si l'on doit prendre en considération que la décomposition d'un entier $n > 1$ en produit de facteurs premiers comme suit $n = p_1^{c_1} * p_2^{c_2} * ... * p_k^{c_k}$ où $p_1 = 2 < p_2 = 3 < \ldots < p_k$ sont les k plus petits nombres premiers, avec c_k le dernier exposant non nul.
En conséquence, pour que n soit hautement composé, il faut que $c_1 \geq c_2 \geq ... \geq c_k$. En cas de non respect de cette règle si nous échangeons deux exposants on diminue le nombre n tout en conservant exactement le même nombre de diviseurs. Un exemple pour illustrer : $18 = 2^1 * 3^2$ peut être remplacé par $12 = 2^2 * 3^1$, ces deux nombres ont tout les deux 6 diviseurs). De plus il a été montré que le dernier exposant $c_k = 1$, sauf dans deux cas particuliers $n = 4$ et $n = 36$.

3 Fonctionnement du projet

4 Bilan technique du projet

5 Analyse des résultats

6 Coûts(Complexité)(facultatif ?)

7 Organisation interne du groupe

Pour débuter la programmation de ce projet, il nous fallait en premier lieu établir la répartition du travail de groupe pour que le projet puisse avancer de façon efficace et de manière rapide. Le tableau

ci-dessus va ainsi indiquer pour chaque membre du groupe la ou les fonctionnalités pour laquelle il a pu contribuer à l'élaboration :

TABLEAU ORGANISATION INTERNE

8 Conclusion