INF4705 – Analyse et conception d’algorithmes

TP1 – Automne 2018

|  |  |
| --- | --- |
| **Nom, prénom, matricule des membres** | Badirou, Salim, 1733231  Badirou, Wajiha, 1770039 |
| **Note finale / 13** | 0 |

# Informations par rapport à la correction

* Répondez directement dans ce document ODT avec LibreOffice. La correction se fait à même le rapport.
* Avant votre troisième séance de laboratoire, vous devez faire une *remise électronique sur Moodle* en suivant les instructions suivantes :
  + Le dossier remis doit se nommer matricule1\_matricule2\_tp1 et doit être compressé sous format zip.
  + À la racine de ce dernier, on doit retrouver :
    - Ce rapport sous format ODT.
    - Un script nommé *tp.sh* servant à exécuter les différents algorithmes du TP. L’interface du script est décrite à la fin du rapport.
    - Un fichier texte nommé *emails.txt* contenant le courriel de chaque membre de l’équipe.
    - Le code source et les exécutables.
* Vous avez le choix du langage de programmation utilisé mais vous devrez utiliser les mêmes langage, compilateur et ordinateur pour toutes vos implantations. Notez que le code et les exécutables soumis seront testés sur les ordinateurs de la salle L-4714 et doivent être compatibles avec cet environnement. En d’autres mots, tout doit fonctionner correctement lorsque le correcteur exécute votre script *tp.sh* sur un des ordinateurs de la salle.

# Mise en situation

Ce travail pratique se répartit sur deux séances de laboratoire et porte sur l'analyse empirique et hybride des algorithmes. À la section 3.2 des notes de cours, trois approches d'analyse de l'implantation d'un algorithme sont décrites. Vous les mettrez en pratique pour des algorithmes de multiplication de matrices.

# Implantation

Vous implanterez les algorithmes de multiplication de matrices *conventionnel* et *diviser-pour-régner* (algorithme de Strassen, section 4.4 des notes de cours). Vous ferez deux versions de ce dernier, avec et sans un seuil de récursivité déterminé expérimentalement par essai-erreur. Pour la version avec seuil de récursivité, les (sous-)exemplaires dont la taille est en deça de ce seuil ne seront plus résolus récursivement mais plutôt directement avec l’algorithme conventionnel. Assurez-vous que vos implantations sont correctes en comparant les résultats des trois algorithmes.

# Jeu de données

Vous trouverez sur le site Moodle du cours un générateur de matrices. Ce générateur prend comme paramètres sur la ligne de commande *N* (la taille de la matrice sera 2*N* × 2*N*), le nom du fichier de sortie (par exemple *ex\_8.1* pour une première matrice avec *N* = 8), et le germe du générateur de nombres aléatoires. Le fichier généré débute avec la valeur de *N* sur la première ligne et les lignes suivantes correspondent aux lignes de la matrice où chaque nombre est séparé par une tabulation. Voici un exemple pour *N* = 2 :

2

1 3 2 1

0 1 2 2

3 3 3 1

3 0 1 1

Pour chaque valeur de *N*, générez cinq matrices que vous pourrez multiplier deux à deux, ce qui vous donnera dix exemplaires. Utilisez au moins cinq valeurs consécutives de *N* pour votre analyse; ce choix pourra varier d'une équipe à l'autre selon la qualité de vos implémentations.

# Présentation des résultats

|  |  |
| --- | --- |
| 0 | / 4 pt |

### Tableau des résulats

*Pour chacun des trois algorithmes, mesurez le temps d’exécution pour chaque exemplaire et rapportez dans un tableau le temps moyen par taille d’exemplaire. Vous vous servirez de ces résultats pour l’analyse qui suit. Lorsque vous calculez les temps d’exécution, vous devez séparer le temps de chargement du jeu de test du temps d’exécution de votre algorithme. Vous devrez donc insérer les sondes temporelles à l’intérieur de votre code.*

### Tests de puissance

*Pour chacun des algorithmes, appliquez le test de puissance et rapportez les graphiques ici.*

Algorithme conventionnel :

Algorithme Strassen :

Algorithme StrassenSeuil :

### Test du rapport

*Pour chacun des algorithmes, appliquez le test du rapport et rapportez les graphiques ici.*

### Test des constantes

*Pour chacun des algorithmes, appliquez le test des constantes et rapportez les graphiques ici.*

Algorithme conventionnel :

Algorithme Strassen :

Algorithme StrassenSeuil :

*Note : Pour effectuer vos régressions linéaires, vous pouvez utiliser l'outil de votre choix : Excel, LibreOffice, gnuplot, R, matplotlib, etc. Il vous serait probablement plus rapide en bout de ligne d’utiliser une méthode qui vous permette d'automatiser la génération de vos graphiques, mais ce choix vous revient. Le guide intitulé “guide bash” sur Moodle peut vous servir d’inspiration si vous voulez automatiser la génération de graphiques.*

# Analyse et discussion

|  |  |
| --- | --- |
| 0 | / 6 pt |

### Que pouvez-vous déduire du test de puissance?

À partir du test de puissance, on peut déduire que chacun des algorithmes ont une consommation polynomiale. Cette consommation polynomiale sous forme de fonction nous permet d’avoir une idée vague de la consommation de ressource en temps des divers algorithmes.

La consommation polynomiale de l’algorithme conventionnel est : y = 1.3597x - 57.611

La consommation polynomiale de l’algorithme de strassen est : y = 8.9064x - 370.86

La consommation polynomiale de l’algorithme de strassen avec seuil est : y = 0.7497x - 31.022

### Citez la consommation théorique du temps de calcul pour les algorithmes, en notation asymptotique.

Consommation théorique de l’algorithme conventionnel : O(n3)

Consommation théorique de l’algorithme de strassen : O(n2.807)

Consommation théorique de l’algorithme de strassen avec un seuil : O(n2.807)

### Que pouvez-vous déduire du test du rapport?

On veut confirmer les hypothèses que les fonctions … sont représentent respectivement la croissance de la consommation de ressource en temps. Les courbes de l’algorithme conventionnel, de strassen et de strassen avec seuil convergent respectivement vers : … Les hypothèses peuvent alors être confirmés et les constantes trouvées sont les constantes multiplicatives respectivent de chaque fonction.

### Que pouvez-vous déduire du test des constantes?

Pour le test des constantes, on cherche à confirmer que les fonctions trouvées lors du test de puissance pour chaque algorithme est en effet la fonction selon laquelle chaque algorithme croit. Pour cela nous avons trouvé la constante multiplicative de chaque algorithme. Les constances multiplicatives de l’algorithme conventionnel, de strassen et de strassen avec seuil sont respectivement :… Cela nous permet ainsi de confirmer que les hypothèses que la consommation de ressource en temps de l’algorithme conventionnel, de strassen et de strassen avec seuil sont respectivement : …

### Discutez de l’impact du seuil de récursivité.

Le seuil de récursivité permet de combiner l’algorithme de strassen et le conventionnel le rendant plus efficace. C’est en effet le cas d’un algorithme diviser pour régner. À partir d’un certain seuil de récursivité, on applique l’algorithme conventionnel, et on remarque bien que pour des matrices suffisamment grande le temps de calcul est réduit de manière non négligeable. Le seuil de récursivité permet donc de rendre un algorithme plus efficace.

### Suite à cette analyse, indiquez sous quelles conditions (taille d’exemplaire ou autre) vous utiliseriez chacun de ces algorithmes. Justifiez.

Grâce aux divers temps de calculs, on peut tout d’abord déduire qu’utiliser l’algorithme de strassen sans seuil de récursivité n’est pas avantageux. En outre, on utilisera l’algorithme conventionnel en temps normal mais à partir de matrices suffisamment grandes, il est plus avantageux d’utiliser l’algorithme de Strassen. Avec les tailles de matrice que nous avons, on peut dire que à partir de matrice de taille supérieure à 50, nous commençons à sentir une différence dans le temps de calcul.

# Autres critères de correction

### Respect de l’interface tp.sh

|  |  |
| --- | --- |
| 0 | / 1 pt |

Utilisation :

tp.sh -a [conv | strassen | strassenSeuil] -e [path\_vers\_exemplaire\_1 path\_vers\_exemplaire\_2]

Arguments optionnels :

-p affiche la matrice résultat dans le format des exemplaires, sans texte superflu

-t affiche le temps d’exécution en ms, sans unité ni texte superflu

Important : l’option -e doit accepter des fichiers avec des paths absolus.

### Qualité du code

|  |  |
| --- | --- |
| 0 | / 1 pt |

### Présentation générale (concision, qualité du français, etc.)

|  |  |
| --- | --- |
| 0 | / 1 pt |

### Pénalités

|  |
| --- |
| 0 |

* Retard : -1 pt / journée de retard, arrondi vers le haut. Les TPs ne sont plus acceptés après 3 jours.
* Autres : Le correcteur peut attribuer d’autres pénalités (par exemple si les exécutables sont manquants, fichier *emails.txt* manquant, etc.)

# Fichiers fournis

Outre le présent rapport que vous devez compléter, les autres fichiers fournis sont les suivants :

### generer\_exemplaire.py

Ce script prend en entrée *N*, un nom de fichier de sortie, et un germe, et génère un exemplaire (une matrice).

### generer\_serie.sh

Ce script prend en entrée un *N* de départ, un nombre de *N*,et un germe, et génère une série d’exemplaires suivant les consignes du TP (cinq matrices par taille d’exemplaire pour un nombre donné de *N* consécutifs, commençant avec le *N* de départ).

### Le répertoire “*exemple”*

Ce répertoire est *fourni à titre d’exemple* et vous êtes libres de vous en servir, de le modifier, ou de vous en inspirer. Il inclut un script *tp.sh* fonctionnel ainsi que le script *autotests.sh* automatisant les tests sur plusieurs exemplaires (notez qu’il s’agit d’exemplaires et d’implémentations bidon pour les besoins de l’exemple).

# Notes techniques

* La commande *chmod +x mon\_script.sh* rendra le script *mon\_script.sh* exécutable. Pour l’exécuter il s’agira de faire *./mon\_script.sh*
* Le guide “guide bash” sur le site Moodle du cours n’est pas spécialement adapté au présent TP, mais il vous aidera à mieux comprendre la façon dont les différents scripts fonctionnent.