INF8775 – Analyse et conception d'algorithmes

TP1 – Hiver 2020

Nom, prénom, matricule des membres	Gaulin, Antoine, 1845639 Guertin, Léandre, 1841782
Note finale / 20	**0**

Informations sur la correction

- Répondez directement dans ce document. La correction se fait à même le rapport.
- La veille de 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 au 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.
    - 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.
- 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.
- 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*

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 deçà 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 dans le dossier *gen_matrix* un générateur de matrices (*Gen*). Ce générateur prend comme paramètres sur la ligne de commande N (la taille de la matrice sera $2^N \times 2^N$), 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. Pour aller plus vite dans la génération des fichiers, vous pouvez utiliser le script *Gen.sh* qui exécute plusieurs fois le fichier *Gen*. Modifiez les valeurs *min* et *max* dans le fichier pour choisir les tailles de matrices générées et exécutez la commande :

```
chmod +x Gen.sh ; chmod +x Gen ; ./Gen.sh
```

Remarque : Il se peut que vous ne puissiez pas lancer le script Gen.sh depuis vos ordinateurs personnels. Utilisez alors les machines du laboratoire pour récupérer les matrices générées.

Q1 – Tableau des résultats

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.

Un tutoriel « guide bash » est disponible sur Moodle si vous souhaitez automatiser le lancement de vos algorithmes.

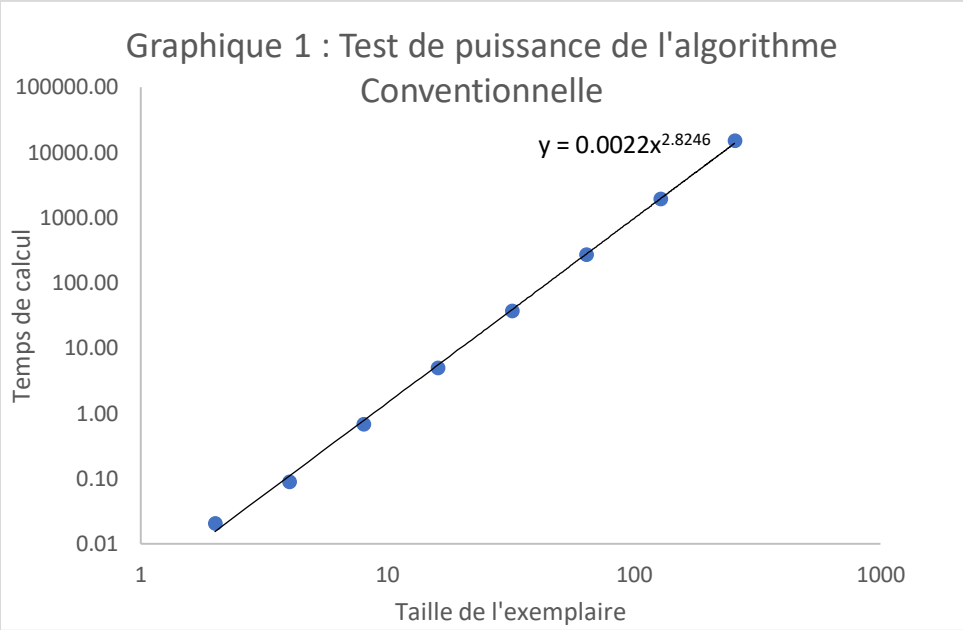
Tableau 1 : Temps d'exécution par algorithmes selon la taille du jeu de donnée

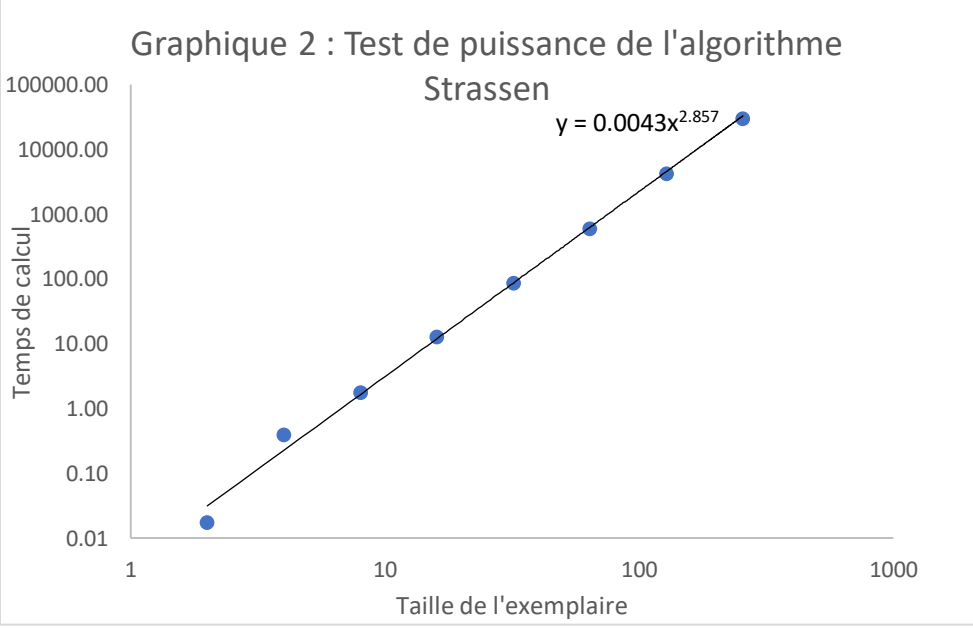
Taille \ Type	**Conventionnelle**	**Strassen**	**Strassen avec seuil de 16**
2x2	0.0207901 ms	0.0174999 ms	0.0232538 ms
4x4	0.0906785 ms	0.4002250 ms	0.0853697 ms
8x8	0.6840230 ms	1.7864900 ms	0.5561670 ms
16x16	5.0810200 ms	13.001200 ms	4.3421600 ms
32x32	37.322900 ms	87.353200 ms	40.369000 ms
64x64	274.47600 ms	610.65000 ms	270.18300 ms
128x128	1950.3500 ms	4320.6400 ms	1795.3800 ms
256x256	15379.900 ms	30087.500 ms	12159.500 ms

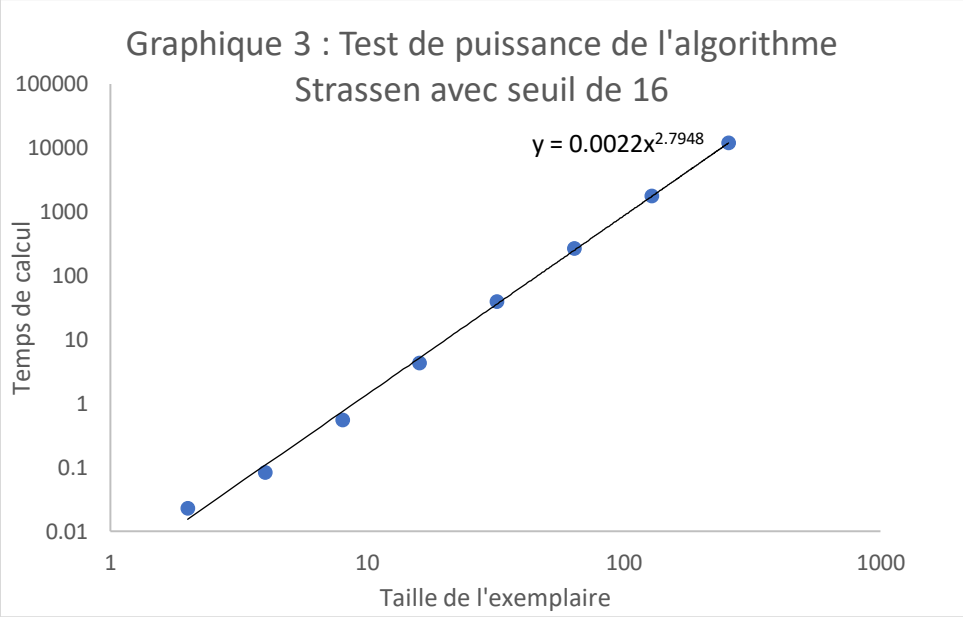
Q2 – Test de puissance

*Pour chacun des algorithmes, appliquez le test de puissance et rapportez les graphiques ici. En quelques lignes, décrivez ce que vous pouvez déduire du test de puissance. Vous pouvez vous référer flow chart « **Approche d'analyse empirique** » disponible sur Moodle (section 3).*

0	/ 3 pt







Puisque les droites passent par tous les points sur les graphiques log-log, nous pouvons déduire que la consommation est polynomiale dans tous les cas et suivent une équation $y = a^b * x^m$, où m est l'exposant du polynôme et a^b est la constante multiplicative.

Tableau 2 : Consommation pratique du temps de calcul pour les algorithmes

Conventionnelle	Strassen	Strassen avec seuil de 16
$O(n^{2,8246})$	$O(n^{2,857})$	$O(n^{2,7948})$

Q3 – Consommation théorique

Citez la consommation théorique du temps de calcul pour les algorithmes, en notation asymptotique. Nul besoin de faire une preuve, on demande seulement de citer.

0	/ 1 pt

Tableau 3 : Consommation théorique du temps de calcul pour les algorithmes

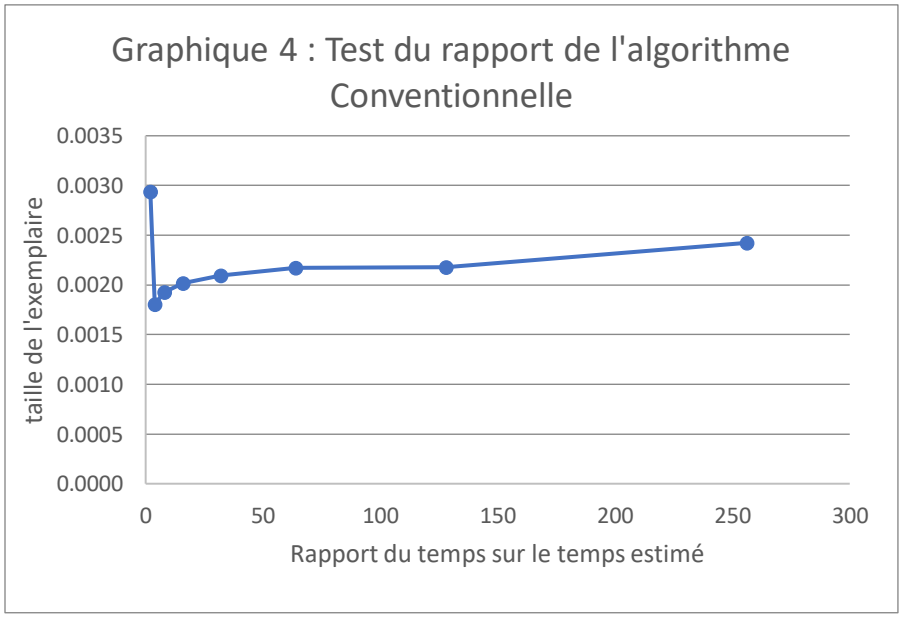
Conventionnelle	Strassen	Strassen avec seuil de 16
$O(n^3)$	$O(n^{2,807})$	Indéterminé, dépendant du seuil choisi.

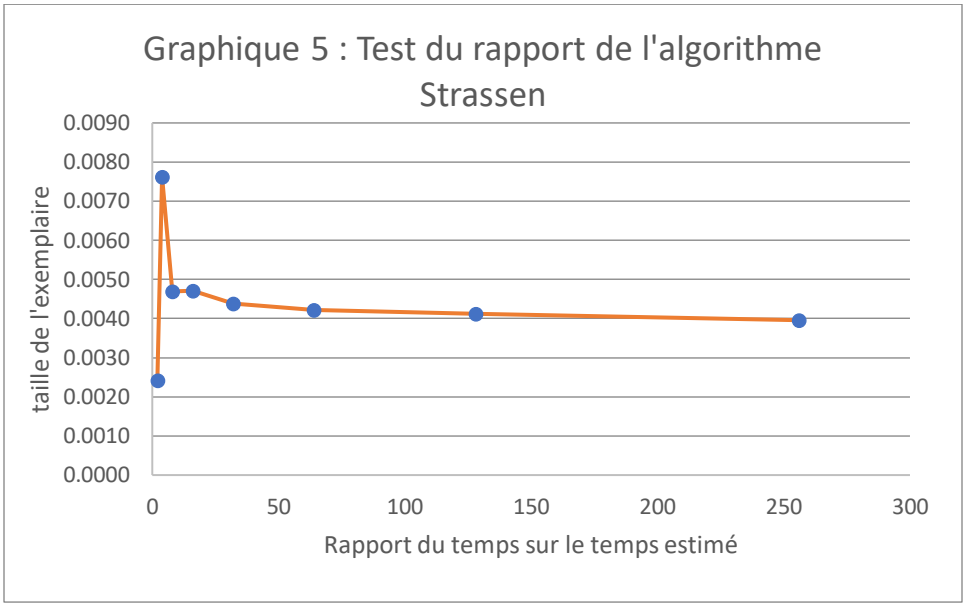
Q4 – Test du rapport

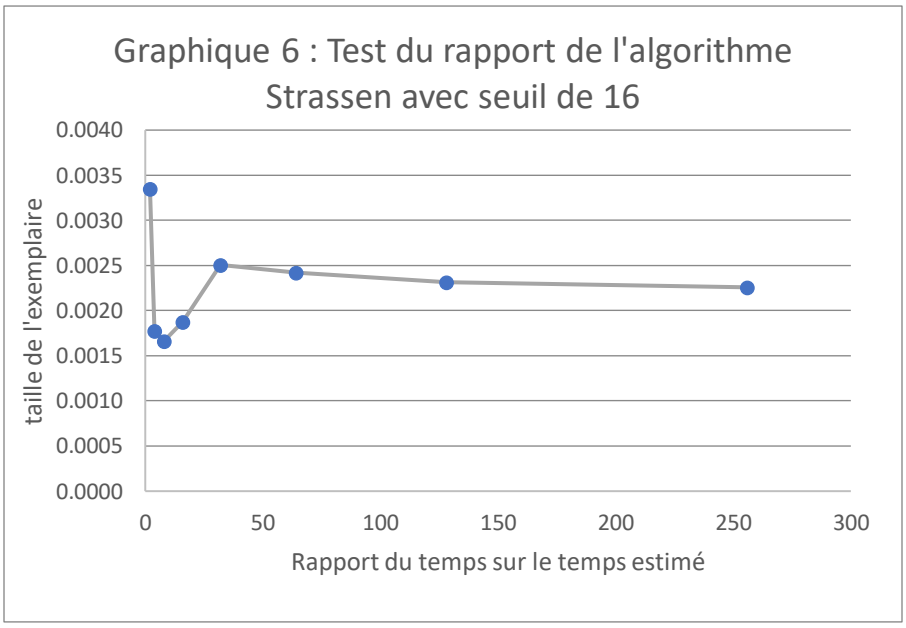
Pour chacun des algorithmes, appliquez le test du rapport et rapportez les graphiques ici. En quelques lignes, décrivez ce que vous pouvez déduire du test du rapport.

0	/ 3 pt

Nous posons l'hypothèse que la consommation de ressource en temps croit selon les fonctions polynomiales trouvées à la suite du test de puissance.







Pour l'algorithme Conventionnelle au graphique 4, l'hypothèse est confirmée et 0.0022 est la constante multiplicative précisant la fonction polynomiale trouvé précédemment.

Pour l'algorithme Strassen au graphique 5, l'hypothèse est confirmée et 0.0040 est la constante multiplicative précisant la fonction polynomiale trouvé précédemment.

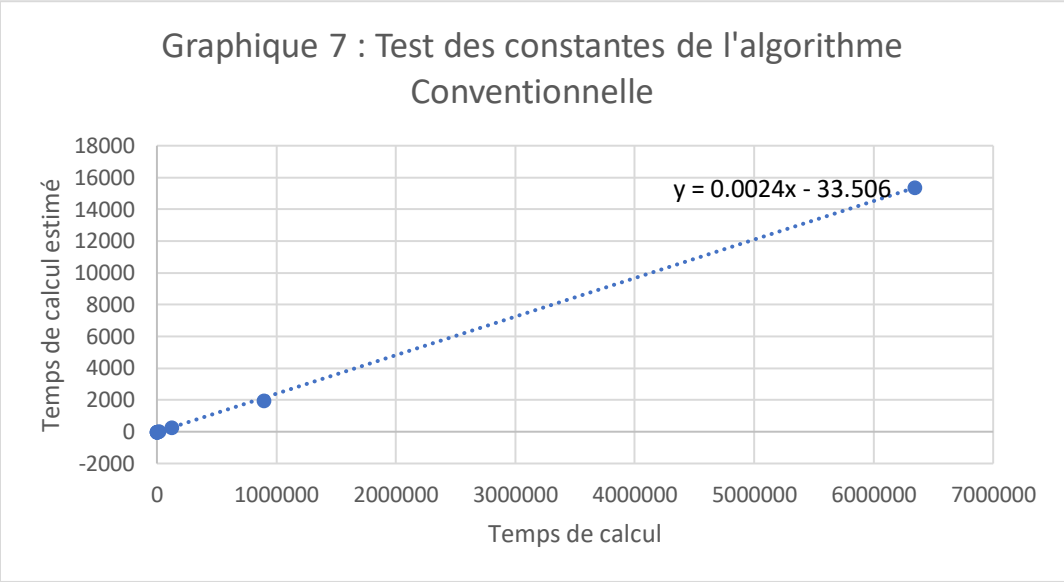
Pour l'algorithme Strassen avec seuil au graphique 6, l'hypothèse est confirmée et 0.0023 est la constante multiplicative précisant la fonction polynomiale trouvé précédemment.

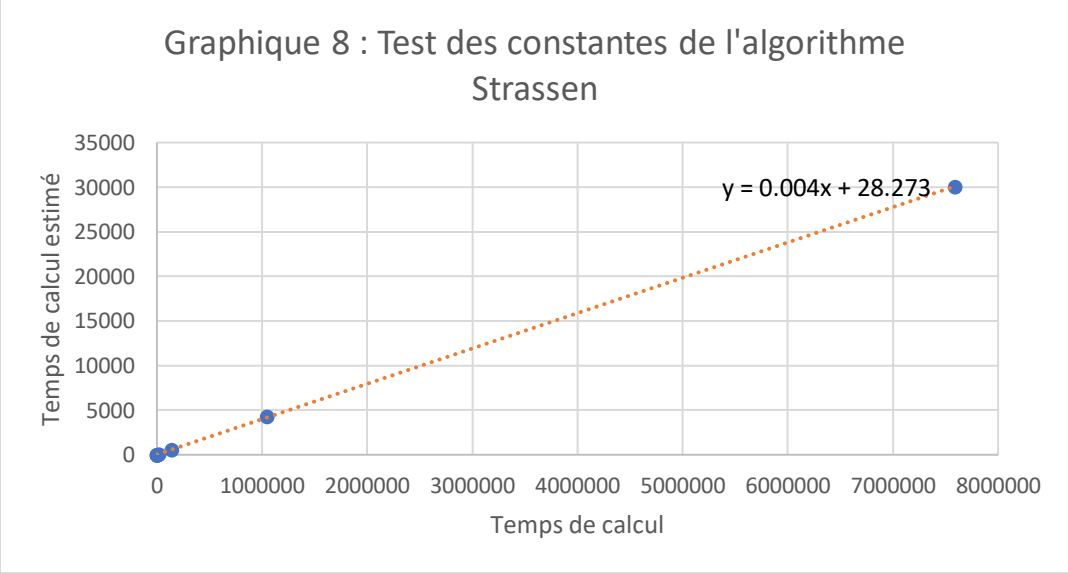
Q5 – Test des constantes

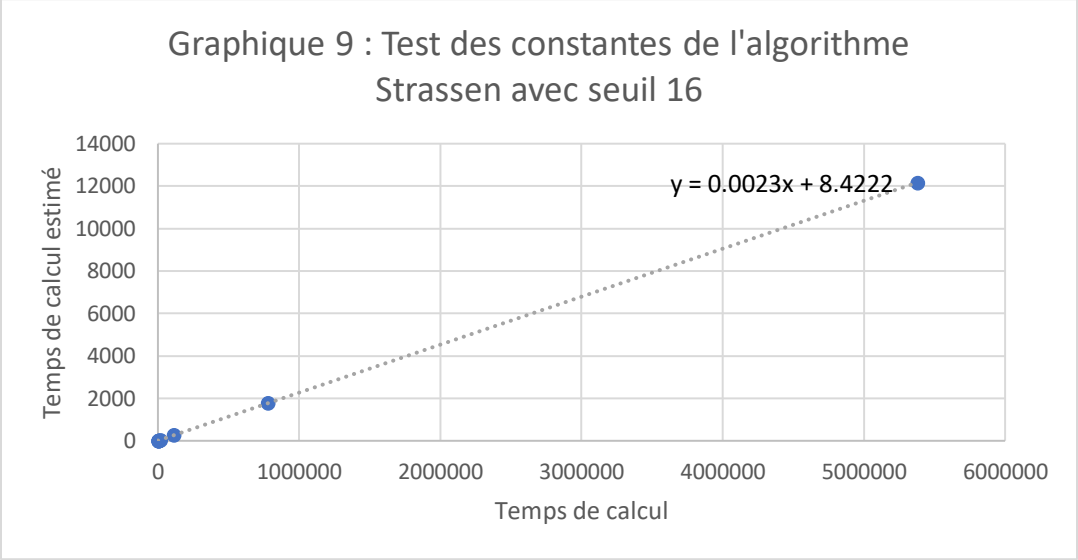
Pour chacun des algorithmes, appliquez le test des constantes et rapportez les graphiques ici. Expliquez brièvement la signification des valeurs que vous pouvez déduire de ce test.

0	/ 3 pt

Pour donner suite à nos tests qui se sont trouvés concluant, nous reprenons la même équation comme hypothèse.







Nous remarquons que la pente de la droite est la constante multiplicative et son ordonnée à l'origine correspond à un coût fixe.

Pour l'algorithme conventionnel au graphique 7, nous avons reçu une valeur de constante multiplicative de 0,0024, ce qui est environ conforme à l'hypothèse. Nous pouvons alors confirmer l'hypothèse.

Pour l'algorithme Strassen au graphique 8, nous avons reçu une valeur de constante multiplicative de 0,0040, ce qui est conforme à l'hypothèse. Nous pouvons alors confirmer l'hypothèse.

Pour l'algorithme Strassen avec seuil au graphique 9, nous avons reçu une valeur de constante multiplicative de 0,0023, ce qui est conforme à l'hypothèse. Nous pouvons alors confirmer l'hypothèse.

Q6 – Impact du seuil de récursivité.

Pour l'algorithme diviser pour régner avec seuil, testez différentes valeurs de seuil de récursivité. Discutez de l'impact du seuil de récursivité.

0	/ 2 pt

Tableau 4 : Temps d'exécution de Strassen avec différents seuils selon la taille du jeu de données

Taille \ Type	**Seuil 8**	**Seuil 16**	**Seuil 32**	**Seuil 64**	**Seuil 128**
2x2	0.0239531 ms	0.0232538 ms	0.0222683 ms	0.0226021 ms	0.0223955 ms
4x4	0.0850995 ms	0.0853697 ms	0.0842571 ms	0.0845273 ms	0.0841618 ms
8x8	0.5578040 ms	0.5561670 ms	0.5640820 ms	0.5584080 ms	0.5632080 ms
16x16	6.0901300 ms	4.3421600 ms	4.2918200 ms	4.2657700 ms	4.3118500 ms
32x32	42.744800 ms	40.369000 ms	33.348100 ms	33.695700 ms	34.220800 ms
64x64	281.59900 ms	270.18300 ms	278.80000 ms	243.68700 ms	248.04700 ms
128x128	1973.5300 ms	1795.3800 ms	1863.6800 ms	2156.1200 ms	1925.7000 ms
256x256	13904.400 ms	12159.500 ms	12285.800 ms	14041.700 ms	16875.900 ms

Parmi les seuils de tableau précédent, nous voyons que le seuil idéal est celui de 16. Ainsi, nous remarquons que les temps sont généralement meilleurs que tous les autres seuils, sauf pour les matrices de taille 32. Nous remarquons que pour les matrices de grandes tailles, le seuil de 16 performe beaucoup mieux que les autres. Cependant, nous ne pouvons rien affirmer pour les matrices plus grandes que celles que nous avons essayées étant donné que nous ne l'avons pas essayé.

Q7 – Conclusion sur l'utilisation des algorithmes

Suite aux réponses précédentes, indiquez sous quelles conditions (taille d'exemplaire ou autre) vous utiliseriez chacun de ces algorithmes. Justifiez en un court paragraphe.

0	/ 2 pt

Nous pouvons alors conclure, pour donner suite à nos expériences, que pour les matrices de petite taille, il est mieux de seulement implémenter l'algorithme conventionnel puisqu'il est simple et demande moins de consommation de ressource. Cependant, pour les matrices de grande taille, il est préférable d'implémenter l'algorithme de Strassen avec seuil. Nous avons remarqué qu'un seuil de 16 est le choix prédominant selon les expériences que nous avons faites. L'algorithme de Strassen sans seuil n'est jamais une bonne option dans le cadre de nos expériences.

Autres critères de correction

Respect de l'interface tp.sh

0	/ 2 pt

Utilisation :

tp.sh -a [conv | strassen | strassenSeuil] -e1 [path_vers_ex_1] -e2 [path_vers_ex_2]

Arguments optionnels :

- -p affiche la matrice résultat contenant uniquement les valeurs, **sans texte superflu**
- -t affiche le temps d'exécution en ms, **sans unité ni texte superflu**

Par exemple, pour deux exemplaires de taille 3 qui se trouvent dans un dossier « exemplaires », la commande de la première ligne fournit la sortie suivante :

```
./tp.sh -e1 ./exemplaires/ex_3.1 -e2 ./exemplaires/ex_3.2 -a strassen -p -t
72 -95 -5 7 -33 46 -155 -58
16 -91 -168 53 -123 86 -170 -135
-58 41 12 120 -16 -17 -24 39
97 -16 60 31 -72 -103 -42 -78
2 -20 33 -105 33 18 62 70
68 -52 7 57 99 -125 44 20
159 -31 101 -113 -81 25 8 -18
-198 47 4 64 -29 87 -64 69
3.949403762817383
```

On y trouve d'abord la matrice de taille 8*8 puis le temps d'exécution en ms.

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, etc.)