INF8775 – Analyse et conception d'algorithmes

TP1 – Hiver 2020

Nom, prénom, matricule des membres	***MASSUCCI -- TEMPLIER, Axel, 1837967*** ***SOUMOY, Astrid, 1902835***
Note finale / 20	

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.

0	/ 2 pt

PRÉALABLE :

Afin de trouver le seuil du troisième algorithme, nous avons testé ces valeurs pour les seuils de 1 à 6, pour une matrice de taille 6.

1	2	3	4	5	6
848,991	258,142	260,822	111,654	110,804	110,588
844,846	254,140	253,414	105,932	105,655	106,189
839,815	252,765	252,247	104,850	104,829	105,245
840,603	254,855	255,363	105,241	105,465	105,597
839,526	257,146	256,409	105,565	107,278	106,695
841,843	255,788	253,848	106,708	105,578	106,542
839,745	254,342	253,054	105,969	105,779	106,868
839,957	253,993	253,047	106,064	105,668	106,140
842,082	257,560	256,580	105,170	104,795	106,201
840,755	255,115	257,058	104,941	105,762	106,121
840,170	257,633	256,387	106,408	104,865	107,072
845,170	253,208	254,494	106,288	105,218	107,109
846,002	253,121	253,432	105,632	105,770	106,033
851,064	254,383	255,460	106,513	105,867	106,216
844,200	256,507	253,477	105,921	105,591	106,030
842,985	**255,247**	**255,006**	**106,190**	**105,928**	**106,576**

Figure 1.1 : Test du seuil pour une matrice de taille 6

Nous avons remarqué qu'à partir du seuil 4, le temps d'exécution moyen était beaucoup plus rapide et constant. Notre troisième algorithme utilise donc Strassen jusqu'à ce que la taille des sous-matrices soit plus petit ou égal à 4. Auquel cas, il changera pour l'algorithme conventionnel.

RÉSULTATS :

Pour chacun des algorithmes, nous avons insérer des minuteurs. Ces derniers démarrent au début de l'exécution de l'algorithme et se termine une fois la matrice finale obtenue. Cela nous permet de mesurer le temps d'exécution de chaque algorithme.

Pour une précision maximale, nous avons généré des matrices allant de la taille 1 (2^1 = 2) à la taille 9 (2^9 = 512), avec 5 matrices par taille. Pour chaque taille, nous avons mesuré le temps d'exécution toutes les combinaisons possibles de multiplication, soit 15 mesures par taille. Nous avons ensuite fait la moyenne par taille pour chaque algorithme, vous pouvez retrouver le détail des mesures dans les tableaux ci-dessous.

taille1	taille2	taille3	taille4	taille5	taille6	taille7	taille8	taille9
0,008	0,019	0,098	0,644	4,304	34,199	255,362	2705,602	19546,19002
0,007	0,019	0,122	0,638	4,329	33,382	252,631	2771,760	21426,55802
0,006	0,019	0,097	0,638	4,398	34,017	253,724	2503,750	19545,09401
0,006	0,018	0,097	0,642	4,301	33,915	254,548	2348,057	19369,79699
0,005	0,019	0,102	0,636	4,547	33,956	254,262	2352,191	18419,56186
0,006	0,018	0,098	0,633	4,556	33,644	252,732	2274,880	18536,45015
0,006	0,019	0,097	0,634	4,713	33,870	255,578	2288,432	18506,68001
0,006	0,018	0,098	0,639	4,503	34,403	253,069	2383,841	18551,43595
0,005	0,018	0,097	0,638	4,580	34,050	252,415	2137,717	18607,41711
0,005	0,019	0,095	0,783	4,358	33,696	253,268	2267,216	18567,91306
0,006	0,019	0,097	0,638	4,288	34,492	252,998	2403,481	18880,56803
0,005	0,018	0,097	0,644	4,437	33,659	252,843	2239,244	19376,6129
0,005	0,018	0,096	0,650	4,271	34,551	254,017	2341,956	18947,91818
0,006	0,018	0,097	0,628	4,381	34,533	254,268	2351,861	19360,28695
0,005	0,018	0,096	0,598	4,404	34,670	252,831	2281,646	18879,03404
0,006	**0,018**	**0,099**	**0,646**	**4,425**	**34,069**	**253,636**	**2376,776**	**19101,434**

Figure 1.2: Tableau des mesures pour l'algorithme conventionnel

taille1	taille2	taille3	taille4	taille5	taille6	taille7	taille8	taille9
0,009	0,159	0,744	5,407	37,009	257,127	1780,146	13206,290	91735,736
0,007	0,112	0,732	6,356	37,378	258,858	1780,413	13449,261	89219,910
0,006	0,101	0,738	6,136	37,051	257,044	1783,283	13461,418	89391,810
0,005	0,101	0,739	6,130	36,988	254,646	1782,935	13234,214	92167,985
0,005	0,101	0,727	6,131	38,163	252,645	1787,551	13266,102	89712,378
0,005	0,100	0,729	5,438	36,605	254,510	1792,701	13167,793	89565,118
0,005	0,104	0,734	5,491	36,984	254,755	1784,295	13183,392	89373,576
0,006	0,101	0,741	5,037	36,950	256,004	1787,856	12999,241	92095,640
0,006	0,099	0,753	5,038	37,085	254,739	1797,126	12944,302	89776,769
0,005	0,100	0,727	4,999	36,656	257,700	1791,341	12696,310	90686,289
0,005	0,100	0,733	5,038	37,301	262,209	1784,831	13118,683	92227,254
0,006	0,099	0,729	5,112	36,904	259,337	1792,501	13015,020	89810,760
0,006	0,100	0,728	5,342	36,713	258,662	1800,825	13018,726	89587,675
0,005	0,099	0,729	5,028	36,872	254,997	1926,792	13111,051	92065,926
0,005	0,099	0,732	5,0149	37,198	254,025	1902,798	12736,660	89599,257
0,006	**0,105**	**0,734**	**5,447**	**37,057**	**256,484**	**1805,026**	**13107,231**	**90467,739**

Figure 1.3 : Tableau des mesures pour l'algorithme Strassen

taille1	taille2	taille3	taille4	taille5	taille6	taille7	taille8	taille9
0,009	0,019	0,329	2,558	15,381	108,010	768,196	5593,155	39200,134
0,006	0,018	0,251	2,129	14,583	109,012	768,660	5675,420	39351,630
0,006	0,018	0,251	2,084	15,092	109,839	766,117	5696,118	39704,438
0,005	0,018	0,257	2,086	14,513	109,065	766,421	5860,252	39258,004
0,005	0,018	0,249	2,078	15,221	108,882	785,505	5715,159	39348,156
0,005	0,018	0,248	2,0709	14,619	109,368	848,512	5692,845	39265,692
0,005	0,017	0,263	2,0721	14,982	110,099	801,141	5828,357	42421,796
0,004	0,017	0,248	2,513	14,450	108,665	787,409	5769,581	39284,428
0,005	0,017	0,248	2,081	14,900	109,737	781,265	5837,114	39484,058
0,005	0,018	0,249	2,072	14,592	109,281	813,297	5777,227	39583,676
0,005	0,018	0,249	2,074	15,821	109,732	820,628	6042,538	39458,699
0,005	0,017	0,247	2,075	14,392	109,392	801,841	6358,508	39489,813
0,005	0,018	0,250	2,079	14,976	108,692	797,302	6416,288	40667,520
0,005	0,018	0,250	2,073	14,641	107,693	791,545	6398,572	42002,972
0,005	0,017	0,248	2,081	15,136	108,374	802,387	6145,165	43079,510
0,005	**0,018**	**0,256**	**2,142**	**14,887**	**109,056**	**793,348**	**5920,420**	**40106,702**

Figure 1.4 : Tableau des mesures pour l'algorithme StrassenSeuil

Nous avons ensuite mis toutes les moyennes dans un tableau récapitulatif afin de pouvoir facilement les comparer.

Poids	Conv	Strassen	StrassenSeuil
1	0,006	0,006	0,005
2	0,018	0,105	0,018
3	0,099	0,734	0,256
4	0,646	5,447	2,142
5	4,425	37,057	14,887
6	34,069	256,484	109,056
7	253,636	1805,026	793,348
8	2376,776	13107,231	5920,420
9	19101,434	90467,739	40106,702

Figure 1.5 : Tableau récapitulatif des temps d'exécutions par algorithme

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

PRÉALABLE :

Pour une harmonisation des résultats, nous avons choisis de négliger les poids inférieurs à 16. En effet, si la taille est de 2, 4 ou 8, nous avons conclu que les valeurs de temps étaient trop petites (comparées aux autres) et donc n'étaient pas pertinentes.

Poids	log(x)	Conv	Conv (log)	Strassen	Strassen (log)	StrassenSeuil	StrassenSeuil (log)
16	4	0,646	-0,630	5,447	2,445	2,142	1,099
32	5	4,425	2,146	37,057	5,212	14,887	3,896
64	6	34,069	5,090	256,484	8,003	109,056	6,769
128	7	253,636	7,987	1805,026	10,818	793,348	9,632
256	8	2376,776	11,215	13107,231	13,678	5920,420	12,531
512	9	19101,434	14,221	90467,739	16,465	40106,702	15,292

Figure 2.1 : Tableau des valeurs logarithmiques pour le test de puissance

Test de puissance de l'algorithme conventionnel :

Nous pouvons voir ci-dessous le graphique à l'échelle log-log de l'algorithme conventionnel. Celui-ci forme une courbe linéaire d'équation $y = 2{,}98x - 12{,}71$.

Puisque nous avons pris ici $y = log_2(\text{y})$ et x = $log_2(\text{x})$, nous nous retrouvons avec la formule suivante :

$log_2(\text{y}) = 2{,}98 \times log_2(\text{x}) - 12{,}71$.

Or, on sait que : $y = a^b \times x^m$

Donc : $a = 2; b = -12{,}71; m = 2{,}98$ Ainsi $y = 2^{-12{,}71}x^{2{,}98}$

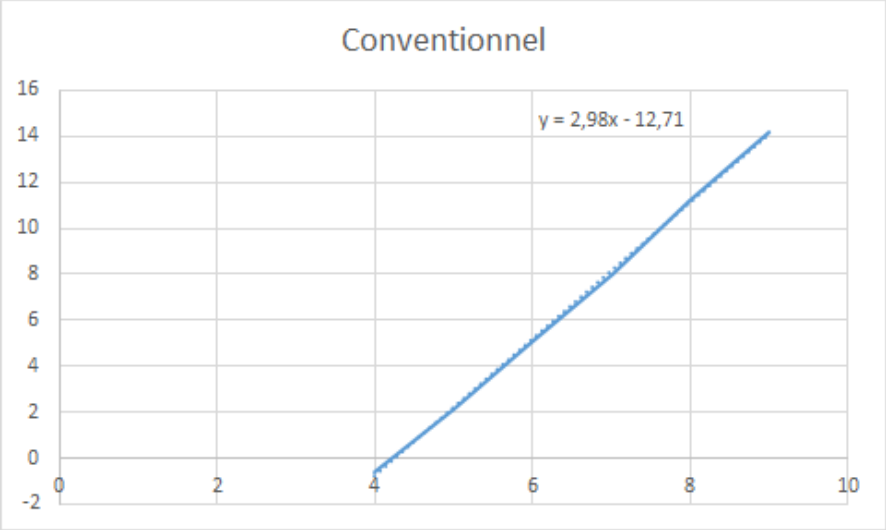

Figure 2.2 : Graphique des valeurs logarithmique de l'algorithme conventionnel

Ce que l'on peut en déduire :

Étant donné que l'on peut faire passer une droite, on peut en déduire que la consommation croît de manière polynomiale avec un exposant de 2,98 et une constante multiplicative évaluée proche de $2^{-12,71}$.

Test de puissance de l'algorithme Strassen :

Nous pouvons voir ci-dessous le graphique à l'échelle log-log de l'algorithme conventionnel. Celui-ci forme une courbe linéaire d'équation $y = 2{,}81x - 8{,}82$.

Puisque nous avons pris ici $y = log_2$(y) et x = log_2(x), nous nous retrouvons avec la formule suivante :

log_2(y) = 2,81 $\times log_2$(x) – 8,82.

Or, on sait que : $y = a^b \times x^m$

Donc : $a = 2; b = -8{,}82 ; m = 2{,}81$. Ainsi $y = 2^{-8,82}x^{2,81}$.

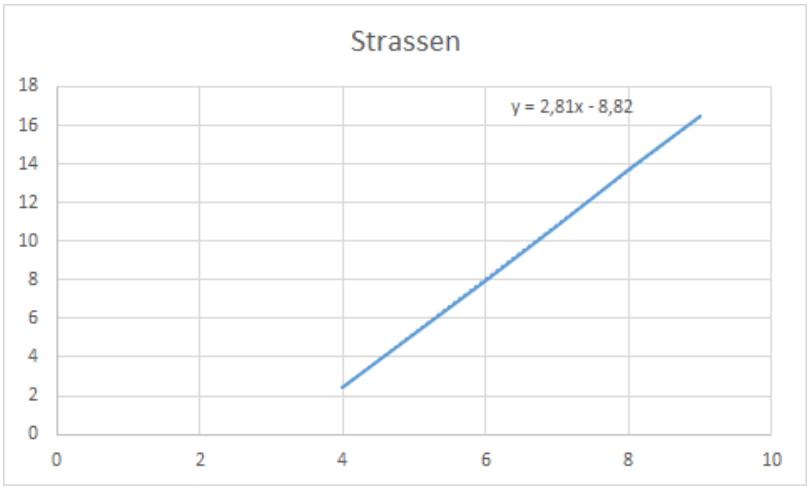

Figure 2.2 : Graphique des valeurs logarithmique de l'algorithme conventionnel

Ce que l'on peut en déduire :

Étant donné que l'on peut faire passer une droite, on peut en déduire que la consommation croît de manière polynomiale avec un exposant de 2,81 et une constante multiplicative évaluée proche de $2^{-8,82}$.

Test de puissance de l'algorithme StrassenSeuil :

Nous pouvons voir ci-dessous le graphique à l'échelle log-log de l'algorithme conventionnel. Celui-ci forme une courbe linéaire d'équation $y = 2{,}85x - 10{,}32$.

Puisque nous avons pris ici $y = log_2(\text{y})$ et x = $log_2(\text{x})$, nous nous retrouvons avec la formule suivante :

$log_2(\text{y}) = 2{,}85 \times log_2(\text{x}) - 10{,}32$.

Or, on sait que : $y = a^b \times x^m$

Donc : $a = 2; b = -10{,}32 ; m = 2{,}85$. Ainsi $y = 2^{-10,32}x^{2,85}$.

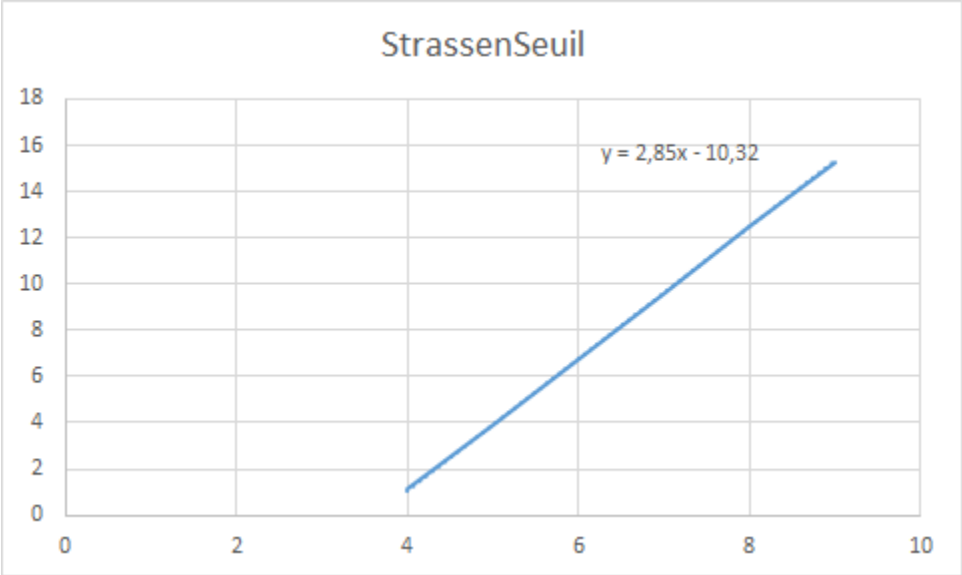

Figure 2.2 : Graphique des valeurs logarithmique de l'algorithme conventionnel

Ce que l'on peut en déduire :

Étant donné que l'on peut faire passer une droite, on peut en déduire que la consommation croit de manière polynomiale avec un exposant de 2,85 et une constante multiplicative évaluée proche de $2^{-10,32}$.

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

Soit n la taille de l'exemplaire à trier.

La consommation théorique du temps de calcul pour l'algorithme conventionnel est :

$\boldsymbol{Conv} \in \boldsymbol{\Theta}(\boldsymbol{n^{2,98}})$

La consommation théorique du temps de calcul pour l'algorithme Strassen est :

$\boldsymbol{Strassen} \in \boldsymbol{\Theta}(\boldsymbol{n^{2,81}})$

La consommation théorique du temps de calcul pour l'algorithme StrassenSeuil est :

$\boldsymbol{StrassenSeuil} \in \boldsymbol{\Theta}(\boldsymbol{n^{2,85}})$

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

PRÉALABLE :

Pour une harmonisation des résultats, nous avons choisis de négliger les poids inférieurs à 16. En effet, si la taille est de 2, 4 ou 8, nous avons conclu que les valeurs de temps étaient trop petites (comparées aux autres) et donc n'étaient pas pertinentes. Nous avons par la suite mis dans un tableau les couples x et $\frac{y}{f(X)}$, pour chaque algorithme. Ici y représente le temps d'exécution de l'algorithme, x représente la taille de la matrice et $f(x)$ la fonction trouvée lors du test de puissance.
Avec ce test du rapport, on cherche à trouver la constante multiplicative précisant $f(x)$. Ainsi, dans notre tableau, $f(x)$ n'inclue pas la constante multiplicative évaluée approximativement au test précédent.

Test du rapport pour l'algorithme conventionnel :

$f(x) = x^{2,98}$

Poids (x)	f(x)	y / f(x)	Conv (y)
2	7,88986164	0,00076047	0,006
4	62,2499166	0,00028916	0,018
8	491,143229	0,00020157	0,099
16	3875,05212	0,00016671	0,646
32	30573,6251	0,00014473	4,425
64	241221,671	0,00014124	34,069
128	1903205,61	0,00013327	253,636
256	15016028,9	0,00015828	2376,776
512	118474391	0,00016123	19101,434

Figure 4.1 : Tableau des valeurs pour le test du rapport de l'algorithme conventionnel

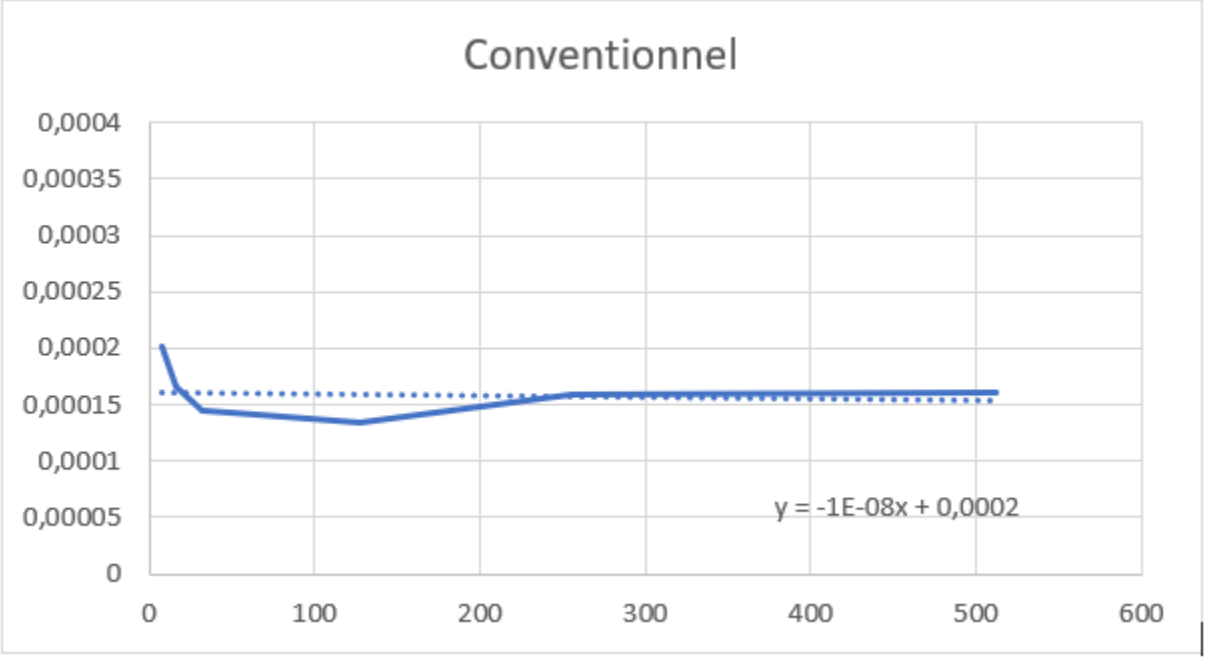

Figure 4.2 : Graphe de l'algorithme conventionnel pour le test du rapport

Ce que l'on peut en déduire :

Nous remarquons que l'on peut tracer une courbe qui converge vers une constante b > 0 où b = 0,0002. De plus, avec le test de puissance nous avions estimé la constante multiplicative à $2^{-12,71} = 0,00014925$. Notre résultat est donc cohérent avec notre approximation de départ. Ainsi, l'hypothèse est confirmée et 0,0002 est la constante multiplicative précisant $f(x)$.

Test du rapport pour l'algorithme Strassen :

$f(x) = x^{2,81}$

Poids (x)	f(x)	y / f(x)	Strassen
2	7,01284577	0,00085557	0,006
4	49,1800058	0,00213501	0,105
8	344,891796	0,0021282	0,734
16	2418,67297	0,00225206	5,447
32	16961,7805	0,00218474	37,057
64	118950,351	0,00215623	256,484
128	834180,464	0,00216383	1805,026
256	5849978,94	0,00224056	13107,231
512	41025000,1	0,00220519	90467,739

Figure 4.3 : Tableau des valeurs pour le test du rapport de l'algorithme Strassen

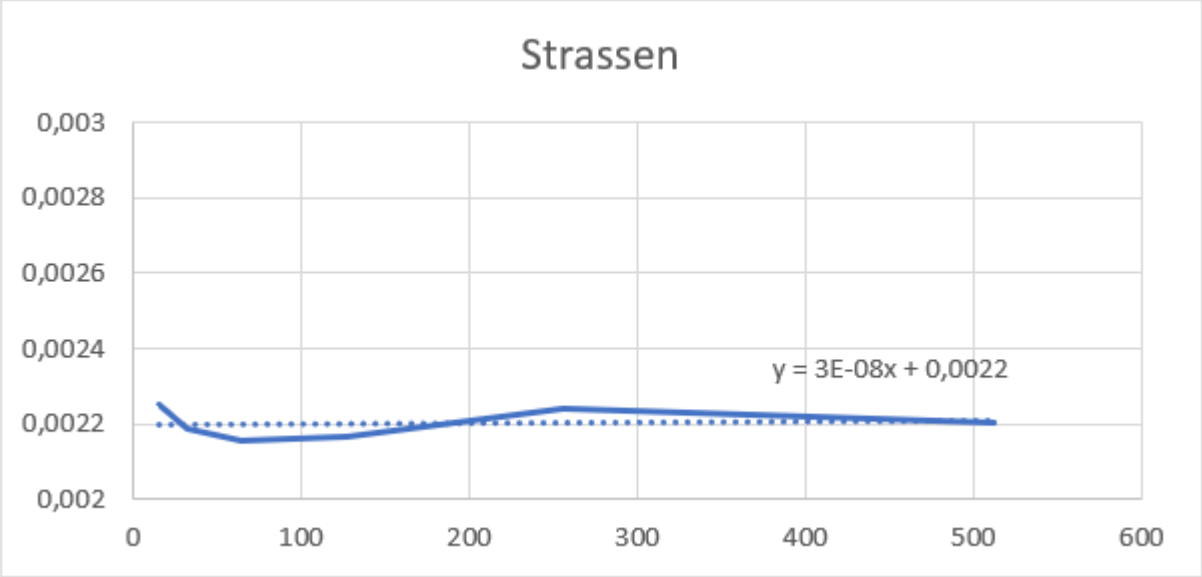

Figure 4.4 : Graphe de l'algorithme Strassen pour le test du rapport

Ce que l'on peut en déduire :

Nous remarquons que l'on peut tracer une courbe qui converge vers une constante b > 0 où b = 0,0022. De plus, avec le test de puissance nous avions estimé la constante multiplicative à $2^{-8,82} = 0{,}00221266$. Notre résultat est donc cohérent avec notre approximation de départ. Ainsi, l'hypothèse est confirmée et 0,0022 est la constante multiplicative précisant $f(x)$.

Test du rapport pour l'algorithme StrassenSeuil :

$f(x) = x^{2,85}$

Poids (x)	f(x)	y / f(x)	StrassenSeuil
2	7,2100037	0,00069348	0,005
4	51,9841534	0,00034626	0,018
8	374,805938	0,00068302	0,256
16	2702,3522	0,00079264	2,142
32	19483,9694	0,00076406	14,887
64	140479,491	0,00077631	109,056
128	1012857,65	0,00078328	793,348
256	7302707,42	0,00081072	5920,42
512	52652547,5	0,00076172	40106,702

Figure 4.5 : Tableau des valeurs pour le test du rapport de l'algorithme StrassenSeuil

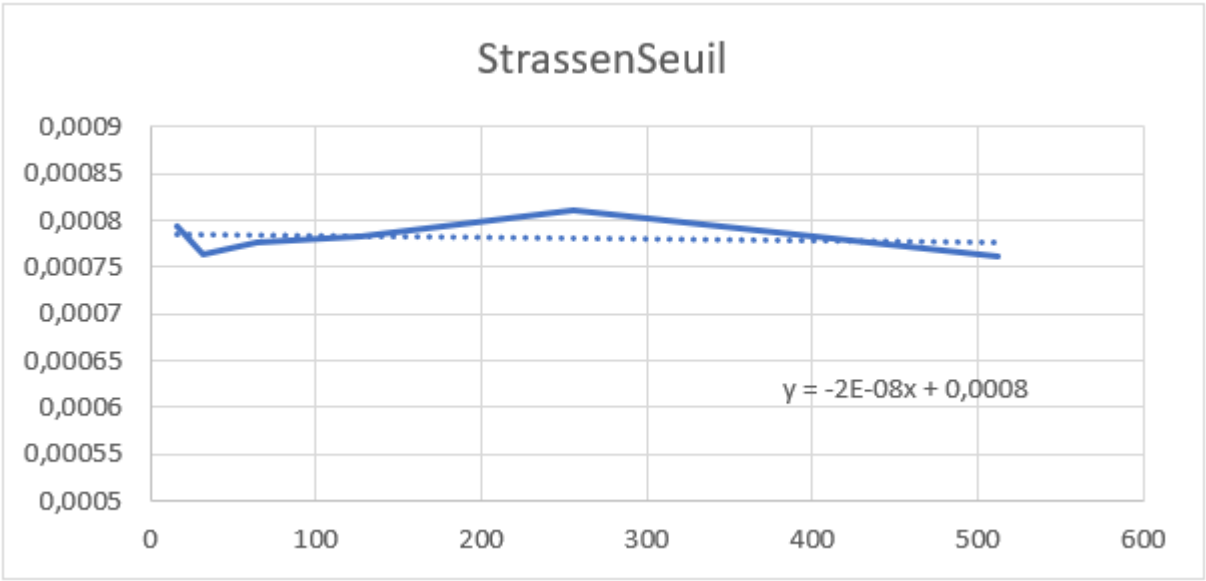

Figure 4.6 : Graphe de l'algorithme StrassenSeuil pour le test du rapport

Ce que l'on peut en déduire :

Nous remarquons que l'on peut tracer une courbe qui converge vers une constante b > 0 où b = 0,0008. De plus, avec le test de puissance nous avions estimé la constante multiplicative à $2^{-10,32} = 0,00078229$. Notre résultat est donc cohérent avec notre approximation de départ. Ainsi, l'hypothèse est confirmée et 0,0008 est la constante multiplicative précisant $f(x)$.

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

PRÉALABLE :

Pour une plus grande précision, nous n'avons pas retiré les mesures des tailles 2, 4 et 8. En effet, dans ce cas cela ne devrait pas influencer la courbe finale. Nous avons par la suite mis dans un tableau les couples y et $f(x)$, pour chaque algorithme. Ici y représente le temps d'exécution de l'algorithme, et $f(x)$ la fonction trouvée lors du test de puissance (sans tenir compte de la constante multiplicative).
Avec ce test des constantes , on cherche à trouver la constante multiplicative précisant $f(x)$, celle-ci doit être la même que celle trouvée lors du test du rapport.

Test des constantes pour l'algorithme Conventionnel :

$f(x) = x^{2,98}$

Poids (x)	f(x)	Conv
2	7,88986164	0,006
4	62,2499166	0,018
8	491,143229	0,099
16	3875,05212	0,646
32	30573,6251	4,425
64	241221,671	34,069
128	1903205,61	253,636
256	15016028,9	2376,776
512	118474391	19101,434

Figure 5.1 : Tableau des valeurs pour le test des constantes de l'algorithme conventionnel

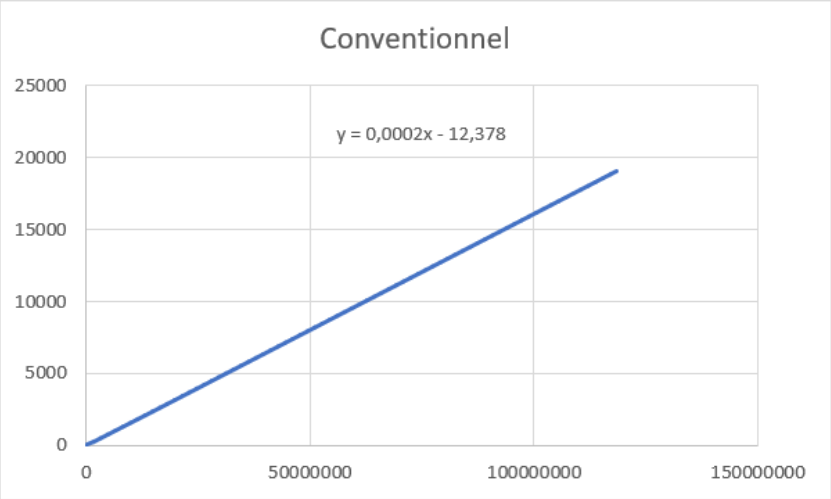

Figure 5.2 : Graphe de l'algorithme conventionnel pour le test des constantes

Ce que l'on peut en déduire :

Nous pouvons constater que ce graphique représente très précisément une droite. La courbe faites avec les valeurs est tellement droite qu'elle se confond avec la courbe de tendance. Cette dernière a pour équation $y = 0{,}0002x - 12{,}378$. Selon le test des constantes, la pente de la droite correspond à la constante multiplicative de la fonction $f(x)$. Celle-ci vaut pour cet algorithme 0,0002, ce qui est égal à la constante multiplicative trouvée lors du test du rapport et proche de celle approximée au test des puissance. Nous pouvons donc prendre cette valeur comme constante multiplicative. De plus, son ordonnée à l'origine correspond à un coût fixe, qui ici vaut -12,378.

La fonction de croissance de l'algorithme conventionnel est donc :

$f(x) = 0{,}0002x^{2{,}98}$

Test des constantes pour l'algorithme Strassen :

$f(x) = x^{2,81}$

Poids (x)	f(x)	Strassen
2	7,01284577	0,006
4	49,1800058	0,105
8	344,891796	0,734
16	2418,67297	5,447
32	16961,7805	37,057
64	118950,351	256,484
128	834180,464	1805,026
256	5849978,94	13107,231
512	41025000,1	90467,739

Figure 5.3 : Tableau des valeurs pour le test des constantes de l'algorithme Strassen

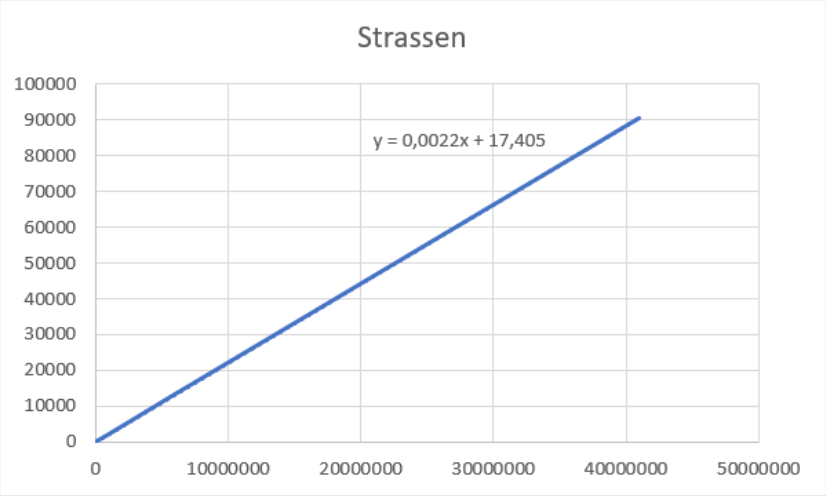

Figure 5.4 : Graphe de l'algorithme Strassen pour le test des constantes

Ce que l'on peut en déduire :

Nous pouvons constater que ce graphique représente très précisément une droite. La courbe faites avec les valeurs est tellement droite qu'elle se confond avec la courbe de tendance. Cette dernière a pour équation $y = 0{,}0022x + 17{,}407$. Selon le test des constantes, la pente de la droite correspond à la constante multiplicative de la fonction $f(x)$. Celle-ci vaut pour cet algorithme 0,0022, ce qui est égal à la constante multiplicative trouvée lors du test du rapport et proche de celle approximée au test des puissance. Nous pouvons donc prendre cette valeur comme constante multiplicative. De plus, son ordonnée à l'origine correspond à un coût fixe, qui ici vaut 17,407.

La fonction de croissance de l'algorithme Strassen est donc :

$f(x) = 0{,}0022x^{2{,}81}$

Test des constantes pour l'algorithme StrassenSeuil :

$$f(x) = x^{2{,}85}$$

Poids (x)	f(x)	StrassenSeuil
2	7,2100037	0,005
4	51,9841534	0,018
8	374,805938	0,256
16	2702,3522	2,142
32	19483,9694	14,887
64	140479,491	109,056
128	1012857,65	793,348
256	7302707,42	5920,42
512	52652547,5	40106,702

Figure 5.5 : Tableau des valeurs pour le test des constantes de l'algorithme StrassenSeuil

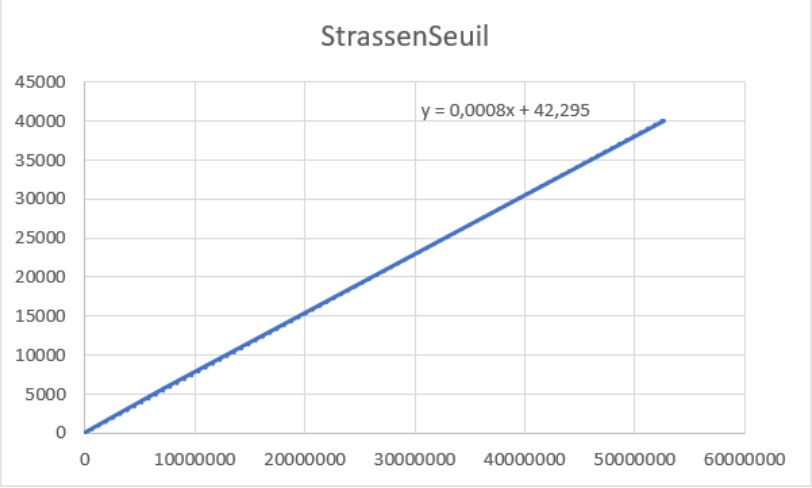

Figure 5.6 : Graphe de l'algorithme StrassenSeuil pour le test des constantes

Ce que l'on peut en déduire :

Nous pouvons constater que ce graphique représente très précisément une droite. La courbe faites avec les valeurs est tellement droite qu'elle se confond avec la courbe de tendance. Cette dernière a pour équation $y = 0{,}0008x + 42{,}295$. Selon le test des constantes, la pente de la droite correspond à la constante multiplicative de la fonction $f(x)$. Celle-ci vaut pour cet algorithme 0,0008, ce qui est égal à la constante multiplicative trouvée lors du test du rapport et proche de celle approximée au test des puissance. Nous pouvons donc prendre cette valeur comme constante multiplicative. De plus, son ordonnée à l'origine correspond à un coût fixe, qui ici vaut 42,295.

La fonction de croissance de l'algorithme conventionnel est donc :

$f(x) = 0{,}0008x^{2{,}85}$

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

Nous avons testé 16 multiplications de matrices de taille 6 (poids 64). Pour ce faire, nous avons pris les valeurs de seuil de récursivité allant de 1 à 6. À noter que nos seuils sont implémentés de la façon suivante : soit un k la variable correspondant au seuil de récursivité et n à la taille des matrices que l'on multiplie. Si n <= k nous appliquons l'algorithme conventionnel. Sinon, nous appliquons Strassen, la taille est donc divisée et la condition est appelé de nouveau. Voici les moyennes des temps d'exécution que nous avons calculé :

1	2	3	4	5	6
848,991	258,142	260,822	111,654	110,804	110,588
844,846	254,140	253,414	105,932	105,655	106,189
839,815	252,765	252,247	104,850	104,829	105,245
840,603	254,855	255,363	105,241	105,465	105,597
839,526	257,146	256,409	105,565	107,278	106,695
841,843	255,788	253,848	106,708	105,578	106,542
839,745	254,342	253,054	105,969	105,779	106,868
839,957	253,993	253,047	106,064	105,668	106,140
842,082	257,560	256,580	105,170	104,795	106,201
840,755	255,115	257,058	104,941	105,762	106,121
840,170	257,633	256,387	106,408	104,865	107,072
845,170	253,208	254,494	106,288	105,218	107,109
846,002	253,121	253,432	105,632	105,770	106,033
851,064	254,383	255,460	106,513	105,867	106,216
844,200	256,507	253,477	105,921	105,591	106,030
842,985	**255,247**	**255,006**	**106,190**	**105,928**	**106,576**

Figure 6.1 : Test du seuil pour une matrice de taille 6

Observation :

Nous pouvons constater que plus le seuil augmente, plus le temps d'exécution diminue. Plus précisément, ce dernier est très élevé au seuil 1, diminue assez rapidement puis semble devenir constant à partir du seuil 4. Il converge alors vers 106ms.

Impact du seuil :

Ainsi, le seuil de récursivité à un impact considérable sur le temps d'exécution de l'algorithme de multiplication de matrices. Utiliser Strassen sur des matrices trop petites pourrait coûter en termes de temps à l'algorithme. Si l'on choisit un seuil plus grand que 2, qui est le seuil par défaut de l'algorithme Strassen, alors la multiplication de matrices de grandes tailles sera optimisée car les matrices plus petites seront multipliées par l'algorithme conventionnel. Ainsi, sa complexité va légèrement diminuer car le temps d'exécution est plus faible pour un même échantillon de matrice. Par ailleurs, même si théoriquement le seuil ne va pas modifier la complexité asymptotique de l'algorithme, nous avons constaté qu'il diminue grandement sa constante multiplicative. Elle passe de 0,0022 à 0,0008 grâce à un seuil plus grand.

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

La création du troisième algorithme (StrassenSeuil) nous permet de mieux comprendre sous quelles conditions nous devrions utilisés chaque algorithme. L'algorithme conventionnel devrait être réservé pour des algorithmes de petites tailles (inférieur à 4). Son temps d'exécution est plus rapide, cependant sa complexité est plus grande. L'algorithme Strassen, lui devrait être utilisé pour des algorithmes de grande taille car sa complexité est plus faible mais son temps d'exécution et sa constante multiplicatrice son plus élevés. Enfin, le dernier algorithme permet d'associer les deux. Ce dernier à un temps d'exécution plus grand que le conventionnel et plus faible que Strassen et une complexité plus faible que conventionnel mais plus grande que Strassen. Il devrait donc être utilisé en tout temps car il utilise le bon algorithme en fonction de la taille de la matrice/sous-matrice.

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.)

Sources :

- Notes de cours abrégées de la session 3, par Gilles Pesant.
- Approche d'analyse empirique de la session 3, par Gilles Pesant.
- The Strassen algorithm, par martin Thoma:

  https://martin-thoma.com/strassen-algorithm-in-python-java-cpp/
- Analyse d'algorithmes, du site UOttawa :

  http://www.site.uottawa.ca/~stan/csi2514/trs/csi2514-4x.pdf