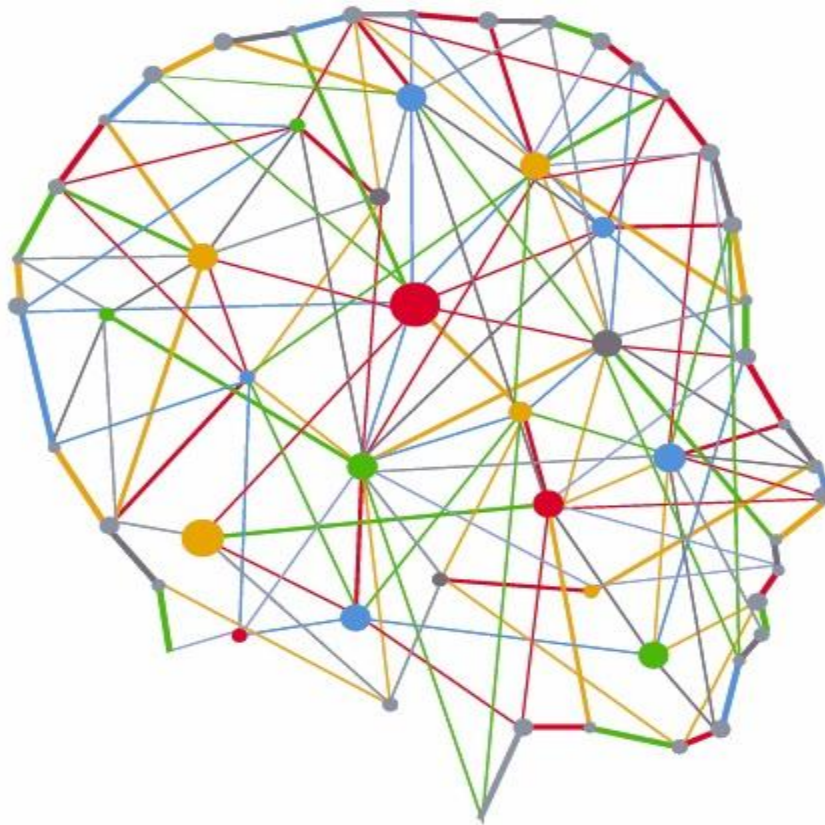


Rapport travaux pratiques n° 1

Mesure du temps d'exécution d'un programme



Travail présenté par:

Membres de la team 13 :

- BOUADI Nassima. 191931012438
- FERKOUS Sarah. 191931043867
- MOKHTARI Mohamed Rayane. 191931069009
- GUERBAS Thinhinane. 191933000894

2022/2023

Sommaire :

Introduction générale	7
1.Solution apportée	8
1.1. Partie I : Développement de l'algorithme et du programme correspondant	8
1.1.1. Question 01	8
1.1.1.1. Algorithme 01	8
1.1.1.2. Algorithme 02	9
1.1.1.3. Algorithme 03	9
1.1.1.4. Algorithme 04	10
1.1.1.5. Algorithme 05	10
1.1.1.6. Algorithme 06	11
1.2. Partie II : Mesure du temps d'exécution	12
1.2.1. Question 01	12
1.2.1.1. Table des résultats d'exécution	12
1.2.1.2. Représentation graphique	13
1.2.1.3. Analyse	14
1.2.1.4. Conclusion	14
1.2.2. Question 02	15
1.2.2.1. Table des temps d'exécution	15
1.2.2.2. Représentation graphique	17
1.2.2.3. Analyse	17
1.2.2.4. Conclusion	18
1.2.3. Question 03	18

1.2.3.1. Table des résultats numériques	18
1.2.3.2. Représentation graphique	19
1.2.3.3. Analyse	19
1.2.3.4. Conclusion	20
1.2.4. Description du code utilisé pour la représentation graphique.....	20
1.3. Partie III : Environnement expérimental	21
1.3.1. Question 01	21
1.3.1.1. L'environnement expérimental	21
2.Analyse générale des résultats	23
Conclusion générale	24
Annexe	25
Bibliographie	44

Liste des tableaux :

- **Tableau 01** : Résultats d'exécution des six algorithmes en prenant des nombres de longueur <12.12
- **Tableau 02** : Résultats d'exécution des six algorithmes en prenant 20 nombres de même longueur.15
- **Tableau 03** : Résultats d'exécution des six algorithmes en prenant des nombres de longueur <12 ,50 fois.....18
- **Tableau 04** : Description de l'environnement expérimental.....21
- **Tableau 05** : Version de langage de programmation.22

Liste des figures :

- **Figure 01** : Représentation graphique des résultats d'exécution des six algorithmes en prenant des nombres de longueur <1214
- **Figure 02** : Représentation graphique des résultats d'exécution des six algorithmes en prenant 20 nombres de même longueur17
- **Figure 03** : Représentation graphique des résultats d'exécution des six algorithmes en prenant des nombres de longueur <20 ,50 fois19
- **Figure 04** : Code source correspondant a l'algorithme 01.....25
- **Figure 05** : Code source correspondant a l'algorithme 0225
- **Figure 06** : Code source correspondant a l'algorithme 0326
- **Figure 07** : Code source correspondant a l'algorithme 0426
- **Figure 08** : Code source correspondant a l'algorithme 0527
- **Figure 09** : Code source correspondant a l'algorithme 0627
- **Figure 10** : Code source correspondant au main28
- **Figure 11** : Code source correspondant au main question 1 partie II capture 1.....29
- **Figure 12** : Code source correspondant au main question 1 partie II capture 2.....30
- **Figure 13** : Code source correspondant au main question 1 partie II capture 3.....30
- **Figure 14** : Code source correspondant au main question 1 partie II capture 4.....31
- **Figure 15** : Code source correspondant au main question 1 partie II capture 5.....31

• Figure 16 : Code source correspondant au main question 2 partie II capture 1.....	32
• Figure 17 : Code source correspondant au main question 2 partie II capture2.....	33
• Figure 18 : Code source correspondant au main question 2 partie II capture 3.....	33
• Figure 19 : Code source correspondant au main question 2 partie II capture 4.....	34
• Figure 20 : Code source correspondant au main question 2 partie II capture 5.....	34
• Figure 21 : Code source correspondant au main question 3 partie II capture 1.....	35
• Figure 22 : Code source correspondant au main question 3 partie II capture 2.....	36
• Figure 23 : Code source correspondant au main question 3 partie II capture 3.....	36
• Figure 24 : Code source correspondant au main question 3 partie II capture 4.....	37
• Figure 25 : Code source correspondant au main question 3 partie II capture 5.....	37
• Figure 26 : Code source correspondant au main question 3 partie II capture 6.....	38
• Figure 27 : fenêtre d'exécution.....	39
• Figure 28 : fenêtre d'exécution question 1 algorithme 01.....	39
• Figure 29 : fenêtre d'exécution question 1 algorithme 02.....	40
• Figure 30 : fenêtre d'exécution question 1 algorithme 03	40
• Figure 31 : fenêtre d'exécution question 1 algorithme 04.....	41
• Figure 32 : fenêtre d'exécution question 1 algorithme 05.....	41
• Figure 33 : fenêtre d'exécution question 1 algorithme 06.....	42

Introduction générale

Trouver un algorithme qui résout un problème c'est trouver une succession d'opérations qui permet de fournir une solution de ce problème. Ce n'est pas toujours une chose facile, mais ce qui peut être encore plus difficile et surtout bien plus intéressant, c'est de trouver un algorithme qui fournit cette solution rapidement.

Lorsque l'on considère un problème, il peut être intéressant de trouver plusieurs algorithmes pour le résoudre, afin d'exploiter le meilleur d'entre eux. Une question se pose alors : **comment déterminer lequel est le plus efficace?** L'approche expérimentale consiste à programmer ces algorithmes et à les essayer sur plusieurs données différentes.

L'approche présentée dans ce TP consiste à déterminer la quantité de ressources (temps) nécessaires à l'exécution de ces algorithmes en fonction de la taille des données considérées, ainsi que le calcul de la complexité temporelle de chaque algorithme qui consiste en une mesure théorique de son temps d'exécution en fonction de la taille de l'instance du problème à traiter (nombre de chiffres dans notre cas).

1. Solution apportée

1.1. Partie I : Développement de l'algorithme et du programme correspondant

1.1.1. Question 01 :

Écrire six algorithmes différents pour déterminer si un nombre entier est premier ou composé. Évaluer la complexité pour chacun des algorithmes proposés (en langage c).

Réponse 01 :

1.1.1.1. Algorithme 01 :

Dans cet algorithme on va s'appuyer sur la définition d'un nombre premier; il va donc consister en une boucle dans laquelle on va tester si le nombre n est divisible par $2, 3, \dots, n-1$ (on épargne le 1 car tous les nombres sont divisibles par 1).

Complexité de l'algorithme 01 :

Le pire cas c'est là où le n est premier donc la boucle s'exécute $(n-2)$ fois.

⇒ $O(n)$.

Pseudo code :

```
Algorithme A1 ;  
début  
    premier := vrai ;  
    i := 2 ;  
    tant que (i <= n-1) et premier faire  
        si (n mod i = 0) alors premier := faux sinon i := i+1 ;  
    fin.
```

1.1.1.2. Algorithme 02 :

Dans cet algorithme on va arrêter la boucle à $n/2$ car si n est divisible par i pour $i = 1 \dots$ partie entière de $(n/2)$ il est aussi divisible par n/i vérifier qu'il est divisible par un nombre supérieur à $n/2$ n'est donc pas nécessaire.

Complexité de l'algorithme 02 :

Le cas le plus défavorable c'est là où n est premier mais dans ce cas le nombre d'itérations est $[n/2]$ où $[n/2]$ dénote la partie entière de $n/2$.

$\Rightarrow O(n)$.

Pseudo code :

```
Algorithme A2 ;
début
    premier := vrai ;
    i := 2 ;
    tant que ((i <= ⌊n/2⌋) et premier) faire
        si (n mod i = 0) alors premier := faux sinon i := i+1 ;
fin
```

1.1.1.3. Algorithme 03 :

Dans ce troisième algorithme on va tester si n est impair dans ce cas il ne faudra tester la divisibilité de n que par les nombres impairs.

Complexité de l'algorithme 03 :

Le pire cas correspond au n est premier le nombre d'itérations $[n/2] - 2$.

$\Rightarrow O(n)$.

Pseudo code :

```
Algorithme A3 ;
début
    premier := vrai ;
    si ((n <> 2) et (n mod 2 = 0)) alors premier := faux
    sinon si (n <> 2) alors
        début
            i := 3 ;
            tant que ((i <= n-2) et premier) faire
                si (n mod i = 0) alors premier := faux sinon i := i+2 ;
        fin
fin
```

1.1.1.4. Algorithme 04 :

L'algorithme 04 consiste à l'association de Algorithme 02 et Algorithme 03.

Complexité de l'algorithme 04 :

Le nombre d'itérations de la boucle pour un nombre premier est égale à la moitié du nombre d'itérations de L'Algorithme 03 $[n/4] - 1$.

$\Rightarrow O(n)$.

Pseudo code :

```
Algorithme A4 ;  
début  
    premier := vrai ;  
    si (n <> 2) et (n mod 2 = 0) alors premier := faux  
    sinon si (n <> 2) alors  
        début  
            i := 3 ;  
            tant que (i <= [n/2]) et premier faire  
                si (n mod i = 0) alors premier := faux sinon i := i+2 ;  
        fin  
fin.
```

1.1.1.5. Algorithme 05 :

Dans ce présent algorithme on va arrêter la boucle à racine de n au lieu de n/2, car si n est divisible par x il est aussi divisible par n/x, donc il est inutile d'aller au-delà de $x = \sqrt{n}$.

Complexité de l'algorithme 05 :

Le nombre maximum d'itérations est $[\sqrt{n}] - 1$.

$\Rightarrow O(\sqrt{n})$.

Pseudo code :

```
Algorithme A5 ;  
début    premier := vrai ;  
        i := 2 ;  
        tant que ((i <= [√n]) et premier) faire  
            si (n mod i = 0) alors premier := faux sinon i := i+1 ;  
fin
```

1.1.1.6. Algorithme 06 :

L'algorithme 06 consiste à l'association de Algorithme 03 et Algorithme 05.

Complexité de l'algorithme 06 :

Le nombre maximum d'itérations est $(\lfloor \sqrt{n} \rfloor / 2) - 1$.

$\Rightarrow O(\sqrt{n})$.

Pseudo code :

```
Algorithme A6 ;
début  premier = vrai ;
        si ( $n \neq 2$ ) et ( $n \bmod 2 = 0$ ) alors premier := faux
        sinon si ( $n \neq 2$ ) alors
            début   $i := 3$  ;
                    tant que ( $i \leq \lfloor \sqrt{n} \rfloor$ ) et premier faire
                        si ( $n \bmod i = 0$ ) alors premier := faux sinon  $i := i + 2$  ;
                    fin
        fin
```

1.2 Partie II : Mesure du temps d'exécution.

1.2.1.Question 01 :

Mesurer les temps d'exécution T pour chacun des six algorithmes en faisant des tests sur des nombres premiers ayant au plus 12 chiffres.

Réponse 01 :

1.2.1.1. Table des résultats d'exécution :

algo nb premiers	A1	A2	A3	A4	A5	A6
8768477	0.0940 00	0.04700 0	0.032000	0.03100 0	0.000000	0.000000
20910101	0.1880 00	0.12500 0	0.093000	0.06300 0	0.000000	0.000000
65229067	0.5310 00	0.37500 0	0.282000	0.18700 0	0.000000	0.000000
501027419	4.1250 00	2.89100 0	2.187000	1.40600 0	0.000000	0.000000
716786933	5.9530 00	4.12700 0	3.109000	2.01600 0	0.000000	0.000000
4617403429	38.357 000	27.5110 00	20.23300 0	12.9850 00	0.016000	0.000000

5544625589	51.148 000	35.9460 00	24.04500 0	15.6240 00	0.000000	0.000000
12469345529	103.21 9000	79.9750 00	54.26400 0	35.0720 00	0.000000	0.016000
46885138957	401.03 7000	270.683 000	204.0100 00	140.712 000	0.000000	0.000000
180327126037	1498.7 69000	1054.82 5000	794.3010 00	564.270 000	0.000000	0.015000

Tableau 01 : Résultats d'exécution des six algorithmes en prenant des nombres de longueur <12.

1.2.1.2. Représentation graphique :

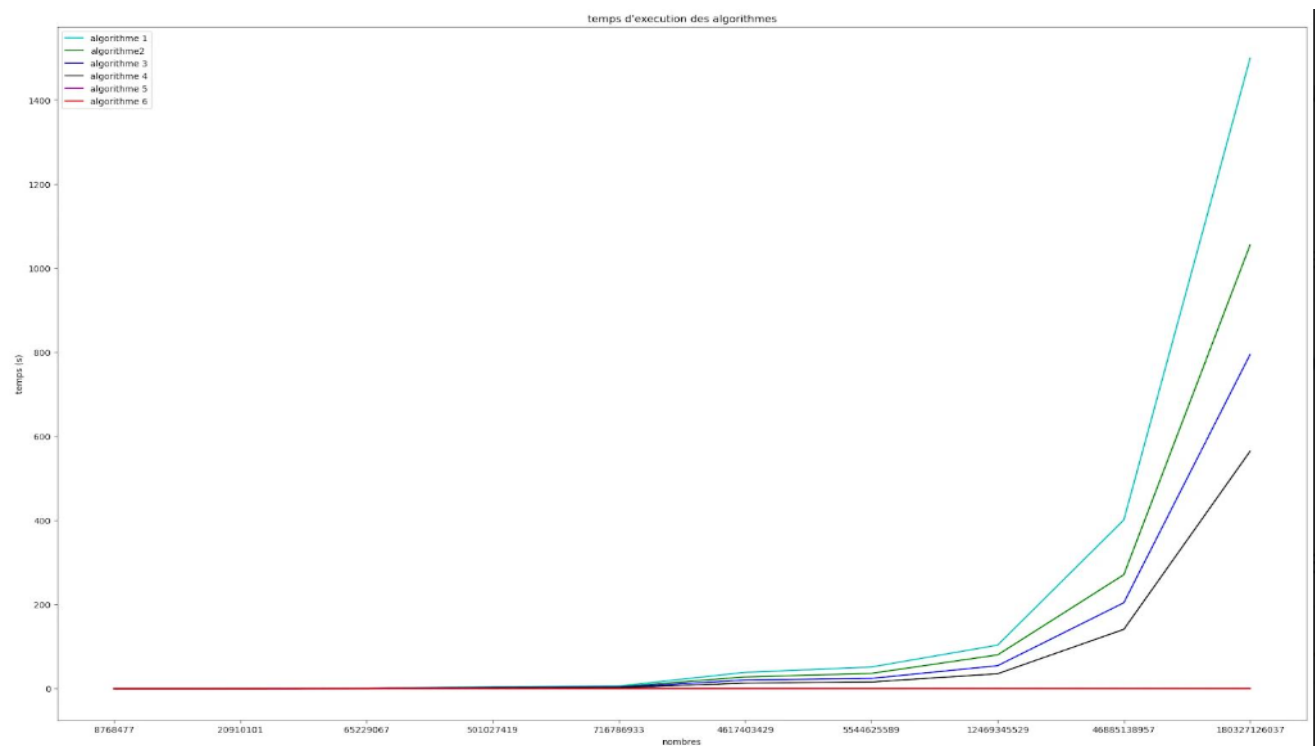


Figure 01 : Représentation graphique des résultats d'exécution des six algorithmes en prenant des nombres de longueur <12.

1.2.1.3. Analyse :

D'après le graphe (figure 01) on remarque qu'en donnant des nombres de longueur 7 à 9, tel que 20910101, la courbe n'évolue pas, elle est constante pour les algorithmes 1,2,3,4,5 et 6. Par contre en donnant des nombres plus grands (10 chiffres) le temps d'exécution correspondant aux 6 algorithmes accroît, c'est le cas pour le nombre 12469345529 par exemple.

On aperçoit aussi qu'au bout du 9ème nombre la courbe correspondante à l'algorithme n°06 voit une croissance supérieure aux autres courbes désignant les autres algorithmes.

1.2.1.4. Conclusion :

Le coût (en temps) d'un algorithme est l'ordre de grandeur du nombre d'opérations arithmétiques ou logiques que doit effectuer un algorithme pour résoudre le problème auquel il est destiné.

Cet ordre de grandeur dépend évidemment de la taille N des données en entrée (la taille du nombre donné en entrée dans notre cas). Plus notre nombre en entrée est grand en nombre de chiffres, plus le temps d'exécution accroît.

Ainsi, pour une entrée à 10000 chiffres, les algorithmes mettront beaucoup plus de temps à s'exécuter que pour une entrée à 10 chiffres.

1.2.2. Question 02 :

Pour une même longueur de nombres (20 chiffres par longueur), dresser la table des temps d'exécution que vous obtenez, puis donner un graphique traduisant les résultats obtenus. Que pouvez-vous conclure ?

Réponse 02 :

1.2.2.1. Table des temps d'exécution :

algo nb premiers	A1	A2	A3	A4	A5	A6
105989621	0.8900 00	0.60900 0	0.453000	0.308000	0.000000	0.000000
133329107	1.1250 00	0.78200 0	0.594000	0.375000	0.000000	0.000000
135741521	1.1410	0.78100	0.594000	0.384000	0.000000	0.000000

	00	0				
177112589	1.4690 00	1.02900 0	0.765000	0.485000	0.016000	0.000000
277151089	2.3320 00	1.62900 0	1.187000	0.766000	0.000000	0.000000
324784529	2.7230 00	1.89000 0	1.411000	0.906000	0.000000	0.000000
378535273	3.1410 00	2.18700 0	1.641000	1.062000	0.000000	0.000000
379866931	3.1400 00	2.18900 0	1.656000	1.063000	0.000000	0.000000
469498597	3.8900 00	2.71100 0	2.065000	1.312000	0.015000	0.000000
508145639	0,4.238 000	2.95300 0	2.266000	1.421000	0.000000	0.000000
509305891	4.1980 00	2.95300 0	2.218000	1.421000	0.000000	0.000000
644920949	5.3600 00	3.76600 0	2.812000	1.797000	0.000000	0.000000
653273063	5.4670 00	3.79000 0	2.813000	1.828000	0.000000	0.000000
666183283	5.5620 00	3.85900 0	2.937000	1.858000	0.000000	0.000000
698207291	5.7810 00	4.04600 0	3.033000	1.937000	0.000000	0.000000
698779387	5.7920 00	4.06300 0	3.031000	2.000000	0.000000	0.000000
778377773	6.4240 00	4.54300 0	3.375000	2.185000	0.000000	0.000000
836627747	7.0570 00	4.85400 0	3.656000	2.344000	0.000000	0.000000
964521797	8.1200	5.59400	4.250000	2.687000	0.000000	0.000000

	00	0				
987952681	8.1680 00	5.73000 0	4.359000	2.750000	0.000000	0.000000

Tableau 02 : Résultats d'exécution des six algorithmes en prenant 20 nombres de même longueur.

1.2.2.2. Représentation graphique :

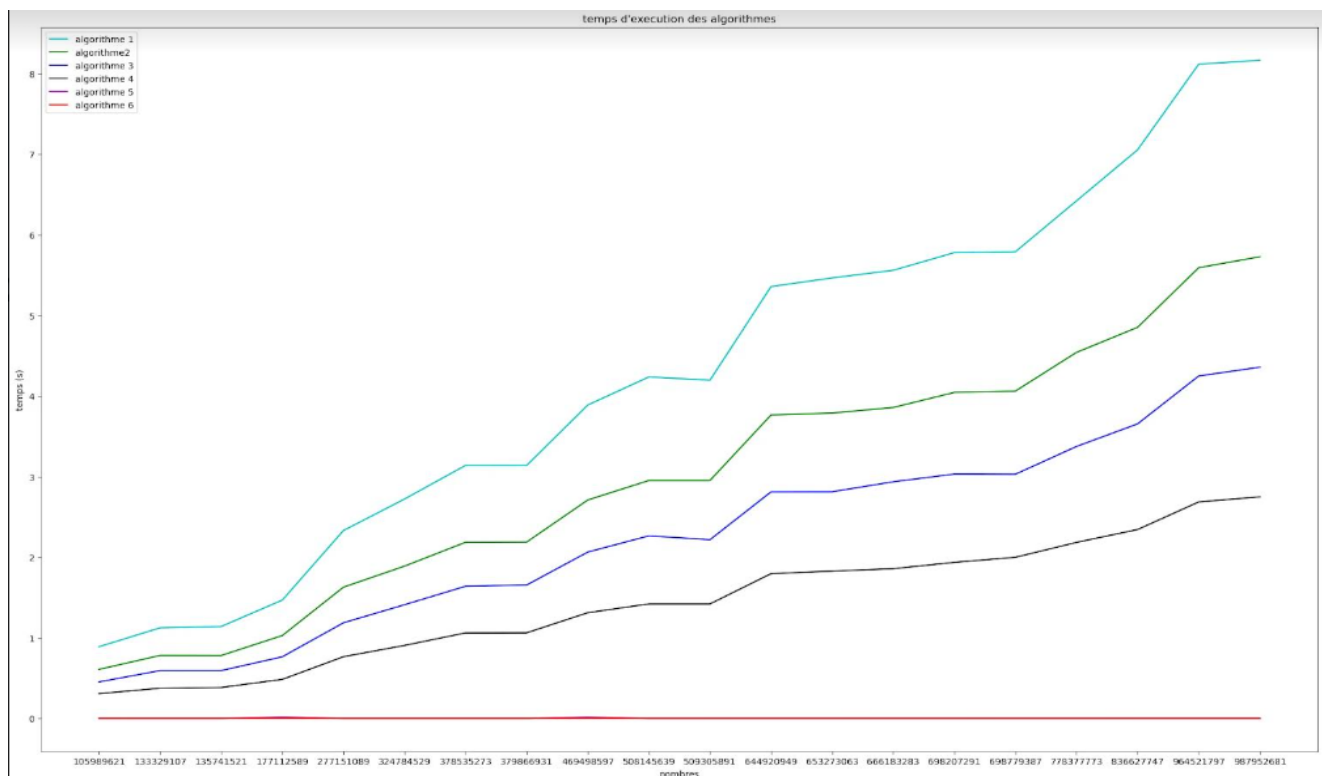


Figure 02 : Représentation graphique des résultats d'exécution des six algorithmes en prenant 20 nombres de même longueur.

1.2.2.3. Analyse :

On remarque à travers ce graphe (figure 02) que pour les nombres de même longueur (ici on choisit longueur = 9 chiffres) les courbes des algorithmes 1, 2, 3 et 4 sont très similaires. De même pour les algorithmes 5 et 6.

Si l'on prend $T(A)_i$ le temps d'exécution d'un algorithme A (en s) pour un nombre $i \in$ l'ensemble des entiers à 9 chiffres alors :

$$\forall i [T(A)_1 > T(A)_2 > T(A)_3 > T(A)_4 > T(A)_5 \approx T(A)_6 = 0.000000 \text{ s}]$$

Autrement dit plus les données sont grandes plus les temps d'exécution des algorithmes 1, 2, 3 et 4 augmentent de façon très rapide.

Alors que pour les algorithmes 5 et 6 qui ne testent que jusqu'à la racine carré de l'instance donnée en entrée, leur temps d'exécution se rapproche de 0.00000 s pour des nombres à neuf chiffres.

Résumé :

- Algorithmes: plus la complexité est faible, plus le temps d'exécution diminue.
- Nombres: plus la taille d'un nombre augmente, plus le temps d'exécution est élevé.

1.2.2.4. Conclusion :

On conclut que l'augmentation de la performance des algorithmes améliore le temps de calcul.

Les algorithmes 5 et 6 sont très rapides en vue de leur temps d'exécution et donc beaucoup plus optimaux que les autres algorithmes, plus particulièrement l'Algorithme 1 qui est très long en raison du nombre d'instructions exécutées qui accroît très vite pour de grandes instances.

Pour de grands nombres de même longueur le temps d'exécution accroît très vite en augmentant leurs tailles car le nombre 900000000 est 9 fois plus grand que le nombre 100000000 alors le temps d'exécution est multiplié par 9 ce qui voudrait dire que si le deuxième nombre mets 5 min en terme de temps d'exécution le premier mettrait 5×9 min ce qui équivaut à 45 min.

$O(n)$ la complexité linéaire est simplement proportionnelle à la grandeur de n , c'est la complexité des algorithmes 1,2,3 et 4.

$O(\sqrt{n})$ la complexité racinaire est la complexité des algorithmes 5 et 6.

Cette différence de complexité explique les comportements opposés des courbes des algorithmes 1,2,3,4 et des algorithmes 5,6.

1.2.3. Question 03 :

Pour des longueurs différentes de nombres, allant de 6 à 12, exécuter les 6 programmes 50 fois (si possible) et reporter la moyenne du temps d'exécution. Dresser la table des résultats numériques puis le graphique correspondant. Que pouvez-vous conclure ?

Réponse 03 :

1.2.3.1. Table des résultats numériques :

algo nb premiers	A1	A2	A3	A4	A5	A6
7474387	0.0000 00	0.08125 7	0.061870	0.044480	0.028436	0.000000
8768477	0.0000 00	0.08125 7	0.061870	0.044480	0.028436	0.000000
20910101	0.0000 07	0.08125 7	0.061870	0.044479	0.028435	0.000000
65229067	0.0006 06	0.08119 0	0.061840	0.044436	0.028360	0.000000
501027419	0.1356 72	0.06187 0	0.044480	0.028436	0.000000	0.000000

Tableau 03 : Résultats d'exécution des six algorithmes en prenant des nombres de longueur <12 ,50 fois.

1.2.3.2. Représentation graphique :

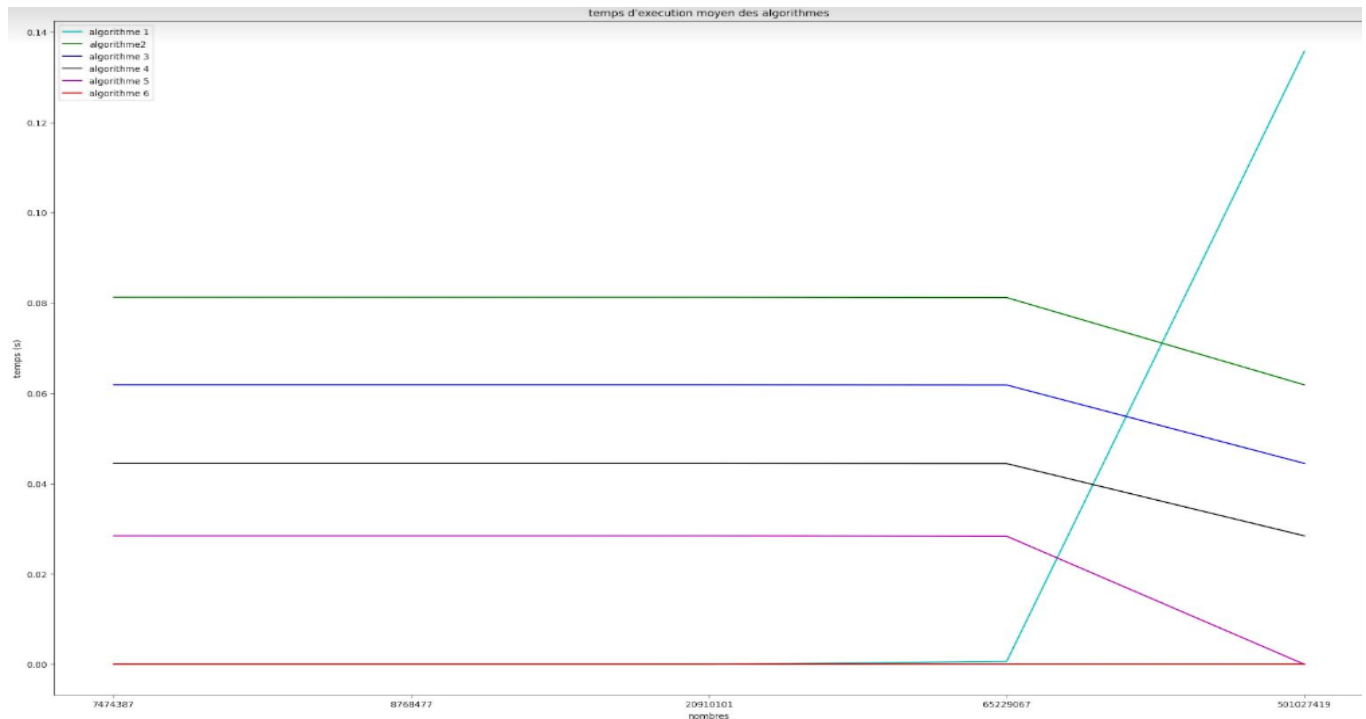


Figure 03 : Représentation graphique des résultats d'exécution des six algorithmes en prenant des nombres de longueur <12 , 50 fois.

1.2.3.3. Analyse :

On remarque à travers ce graphe ("figure 03") que pour des nombres de différentes longueurs (entre 6 et 12 chiffres) les courbes des différents algorithmes sont stables pour des nombres allant de 6 à 8 chiffres. À partir de 9 chiffres elles diminuent (Algorithmes 2, 3, 4, et 5) où augmente de façon exponentielle (Algorithme 1) ou reste stable (Algorithme 6).

On remarque aussi que pour des nombres de 6 à 9 chiffres la moyenne du temps d'exécution de l'Algorithme 6 reste approximativement proche de zéro.

1.2.3.4. Conclusion :

Suite à l'analyse effectuée on conclut qu'exécuter 50 fois les six algorithmes tout en leur donnant différentes données en entrée permettrait de calculer les moyennes d'exécution qui sont plus significatives pour déterminer l'algorithme le plus performant, car une exécution dépend de l'état actuel du système qui diffère d'une exécution à une autre.

Remarque :

Le lien du code pour effectuer les graphes correspondant à chaque algorithme [graphes.ipynb](#) (ouvrir avec google colab).

Pour nos tests on les a effectués sur une machine (DELL i7) avec un processeur x64 Windows 10 Pro N ,système d'exploitation 64 bits.

1.2.4. Description du code utilisé pour la représentation graphique:

Pour réaliser les graphes de temps d'exécution de chaque algorithme on a eu recours au langage python à travers la bibliothèque **matplotlib** conçue spécialement pour dresser des courbes de natures différentes.

Le script trace les graphes en se basant sur une liste de nombres qui représente les temps d'exécution récupérés à partir des fichiers générés par le programme C.

Les paramètres **xlabel**, **ylabel**, **title** et **legend** servent à personnaliser les graphes et à donner une meilleure compréhension et lisibilité des fonctions tracées.

1.3. Partie III : Environnement expérimental

1.3.1.Question 01 :

Décrire l'environnement expérimental : caractéristiques de la machine utilisée, version du langage de programmation, etc.

Réponse 01 :

1.3.1.1. L'environnement expérimental :

Caractéristiques de la machine	BOUADI Nassima	FERKOUS Sarah	MOKHTARI Mohamed Rayane	GUERBAS Tinhinane
Marque	DELL i3	DELL i5	HP G5	DELL i7
Système d'exploitation	Système d'exploitation 64 bits, processeur x64. Windows 10 Professionnel.	Système d'exploitation 64 bits, processeur x64. Windows 10 Professionnel.	Système d'exploitation 64 bits, processeur x64. Windows 10 Professionnel.	Système d'exploitation 64 bits, processeur x64 Windows 10 Pro N.
Processeur	Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz 2.00 GHz	Intel(R) Core(TM) i5-7300U CPU @ 2.60GHz 2.70 GHz	Intel® Core(TM) i3-6006U CPU 2.00GHz	Intel(R) Core(TM) i7-4610M CPU @ 3.00GHz 3.00 GHz
Mémoire Ram installée	4,00 Go	8,00 Go	4,00 Go	4,00 Go

Tableau 04 : Description de l'environnement expérimental.

Version langage de programmation	BOUADI Nassima	FERKOUS Sarah	MOKHTARI Mohamed Rayane	GUERBAS Tinhinane
	code block 17.12	code blocks 20.03	code block 17.12	dev c++ 5.11

Tableau 05 : Version de langage de programmation.

2. Analyse générale des résultats obtenus

D'après les résultats obtenus à partir des graphes précédents on remarque que plus l'algorithme est optimal (exécute un minimum d'itération ie. un minimum d'instructions) plus le temps d'exécution est faible.

Nous avons vu aussi que le temps d'exécution d'un programme dépendait de différentes caractéristiques, principalement :

- De la taille des données en entrée.
- De la complexité temporelle de l'algorithme.

Le calcul de la **complexité** d'un algorithme permet de mesurer sa **performance**. Il existe deux types de complexité :

- **complexité spatiale** : permet de quantifier l'utilisation de la **mémoire**
- **complexité temporelle** : permet de quantifier la **vitesse** d'exécution

Conclusion générale

Notre objectif dans ce TP était de pouvoir prévoir le temps d'exécution d'un algorithme et pouvoir comparer l'efficacité de six algorithmes résolvant le même problème (le test de la primalité d'un entier). L'analyse des résultats obtenus ainsi que le calcul théorique de leurs complexités temporelles nous a permis d'établir lequel d'entre eux est le plus optimal et ainsi de comprendre pourquoi il est nécessaire de trouver l'algorithme le plus rapide en termes de temps d'exécution. Cependant plus la complexité tend vers l'exponentielle plus le temps d'exécution devient prohibitif et impossible à réaliser, dans le cas où elle est de type polynomiale, le temps effectif peut rester raisonnable.

Annexe

Code source des six algorithmes:

Algorithme 01 :

```
8 //-----Algo1-----
9
10 /***on va faire une boucle dans laquelle on va tester si le nombre n est
11 //divisible par 2,3...,n-1(on epargne le 1 car tout les nombre sont divisible par 1
12
13 int Algorithme1(unsigned long long nbr){
14     int premier = 1;
15     unsigned long long i=2;
16     while((i<=nbr-1) && (premier)){
17         if(nbr%i == 0) premier = 0;
18         else i++;
19     }
20     return premier;
21 }
```

Figure 04 : Code source correspondant à l'algorithme 01.

Algorithme 02 :

```
23 //-----Algo2-----
24
25 /***on va arrêter la boucle a n/2 car si n est divisible par 2 il est aussi divisible par n/2
26
27 int Algorithme2(unsigned long long nbr){
28     int premier=1;
29     unsigned long long i=2;
30     while(i<=floor(nbr/2) && (premier)){
31         if(nbr%i==0) premier = 0;
32         else i++;
33     }
34     return premier;
35 }
36
37
```

Figure 05 : Code source correspondant à l'algorithme 02.

Algorithme 03 :

```
38 //-----Algo3-----
39
40
41 //****tester si n est impair dans ce cas il faut dans la boucle tester la divisibilite de n que par les nb impair
42
43 int Algorithme3(unsigned long long nbr){
44     int premier = 1;
45     if((nbr!=2) &&(nbr%2 == 0)) premier = 0;
46     else if(nbr!=2){
47         unsigned long long i=3;
48         while((i<=nbr-2) && (premier)){
49             if(nbr % i ==0) premier = 0 ;
50             else i=i+2;
51         }
52     }
53     return premier;
54 }
55
```

Figure 06 : Code source correspondant à l'algorithme 03.

Algorithme 04 :

```
56 //-----Algo4-----
57
58
59 //****melange de A2 ET A3
60
61 int Algorithme4(unsigned long long nbr){
62     int premier = 1;
63     if((nbr!=2) &&(nbr%2 == 0)) premier = 0;
64     else if(nbr!=2){
65         unsigned long long i=3;
66         while((i<=floor(nbr/2)) && (premier)){
67             if(nbr % i ==0)
68                 premier = 0 ;
69             else
70                 i = i+2;
71         }
72     }
73     return premier;
74 }
```

Figure 07 : Code source correspondant à l'algorithme 04.

Algorithme 05 :

```
76 //-----Algo5-----
77
78 //****arrêter la boucle a racine de n au lieu de n/2 car si n est divisible par x il est
79 //aussi divisible par n/x donc ça ne sert a rien d'aller au dela de x=n/x
80
81 int Algorithme5(unsigned long long nbr){
82     int premier = 1;
83     unsigned long long i = 2;
84     while((i<=floor(sqrt(nbr))) && (premier)){
85         if(nbr%i==0) premier = 0 ;
86         else i++;
87     }
88     return premier;
89 }
90
```

Figure 08 : Code source correspondant à l'algorithme 05.

Algorithme 06 :

```
91 //-----Algo6-----
92
93 //****melange de A3 et A5
94
95 int Algorithme6(unsigned long long nbr){
96     int premier = 1;
97     if((nbr!=2) && (nbr%2 == 0))
98         premier = 0;
99     else
100         if(nbr!=2){
101             unsigned long long i=3;
102             while((i<=floor(sqrt(nbr))) && (premier)){
103                 if(nbr%i == 0)
104                     premier = 0 ;
105                 else
106                     i = i+2;}
107         }
108     return premier;
109 }
110
```

Figure 09 : Code source correspondant à l'algorithme 06.

Explications 01 :

Dans les captures précédentes on a écrit 6 algorithmes différents que l'on a transformés sous forme de fonctions permettant de définir si un nombre donné en entrée est premier ou pas.

Code source de mesure des temps d'exécution T pour chacun des six algorithmes en faisant des tests sur des nombres premiers ayant au plus 12 chiffres:

```
115 int main()
116 {
117     //DECLARATION DE 2 FICHIERS POUR LES RESULTATS
118     FILE* fichier=NULL;
119     FILE* fichierQst3=NULL;
120     int nbrfois;
121     //DECLARATION DES TABLEAUX
122     //QST1----
123     unsigned long long Tab1[] = {8768477,20910101,65229067,501027419,716786933,4617403429,5544625589,12469345529,46885138957,180327126037};
124     //QST2- ---
125     unsigned long long Tab2[] = {105989621,133329107,135741521,177112589,277151089,324784529,378535273,379866931,469498597,508145639,
126     509305891,644920949,653273063,666183283,698207291,698779387,778377773,836627747,964521797,987952681};
127     //QST3- ---
128     unsigned long long Tab3[] = {7474387,8768477,20910101,65229067,501027419};
129     //CALCULE DES TAILLES DES 3 TABLEAUX
130     int j=0;
131     int taille1 = sizeof(Tab1)/sizeof(Tab1[0]);
132     int taille2 = sizeof(Tab2)/sizeof(Tab2[0]);
133     int taille3 = sizeof(Tab3)/sizeof(Tab3[0]);
134     double Moyenne[20];
135     double moy;
136     //DECLARATION DES VARIABLES DE MESURE DU TEMPS
137     clock_t t1,t2;
138     double delta;
139     float reel;
140     int choix;
141     printf("\n Choisissez votre chiffre:\n1 -> Pour executer la question 1 et la question 2.\n2 -> Pour executer la question 3 50 fois.\n");
142     printf("\nMon choix est:\t");
143     scanf("%d",&choix);
144     switch(choix){
145     case 1:
146         //CREATION DU FICHIER QUI VA STOCKER LES RESULTATS DE LA QST 1 ET 2
147         fichier = fopen("C:/Users/DELL/Desktop/TempPrem12.txt","w");
148         /* //-----QST1
149         printf("-----QST 1-----\n\n");
150         printf("\t\t ALGORITHME 1\n");
151         //BOUCLE DE PARCOUR D'UN TABLEAU
152     }
```

Figure 10 : Code source correspondant au main .


```

192 //APPEL AU FONCTION
193 int test=Algorithme2(Tab1[j]);
194
195 //AFFICHAGE (PREMIER OU NON)
196 if(test == 0){
197     printf("***** Le nombre ( %llu ) n'est pas premier!\n",Tab1[j]);
198 }else{
199     printf("***** Le nombre ( %llu ) est premier.\n",Tab1[j]);
200 }
201
202 //2EME VAR DE TEMPS
203 t2 = clock();
204
205 //LA MOYENNE DE TEMPS D'EXECUTION
206 delta = (double)(t2-t1)/CLOCKS_PER_SEC;
207 reel = (float)delta;
208
209 //AFFICHAGE DU TEMPS D'EXECUTION
210 printf("***** Le temps d'execution de (%llu) est: ( %lf )\n\n",Tab1[j],delta);
211
212 //SAISIE DANS LE FICHIER
213 fprintf(fichier,"%lf,",delta);
214 }
215 fprintf(fichier,"\n");
216
217 //-----ALGO3-----
218 printf("\n \t\t ALGORITHME 3\n");
219 //BOUCLE DE PARCOUR D'UN TABLEAU
220 for(j=0;j<taille1;j++){
221
222     //1ERE VAR DE TEMPS
223     t1 = clock();
224
225     //APPEL AU FONCTION
226     int test=Algorithme3(Tab1[j]);
227
228     //AFFICHAGE (PREMIER OU NON)
229     if(test == 0){
230         printf("***** Le nombre ( %llu ) n'est pas premier!\n",Tab1[j]);
231     }else{
232         printf("***** Le nombre ( %llu ) est premier.\n",Tab1[j]);
233     }

```

Figure 12 : Code source correspondant au main question 1 partie II capture 2.

```

235 //2EME VAR DE TEMPS
236 t2 = clock();
237
238 //LA MOYENNE DE TEMPS D'EXECUTION
239 delta = (double)(t2-t1)/CLOCKS_PER_SEC;
240 reel = (float)delta;
241
242 //AFFICHAGE DU TEMPS D'EXECUTION
243 printf("***** Le temps d'execution de (%llu) est: ( %lf )\n\n",Tab1[j],delta);
244
245 //SAISIE DANS LE FICHIER
246 fprintf(fichier,"%lf,",delta);
247 }
248 fprintf(fichier,"\n");
249
250 //-----ALGO4-----
251 printf("\n \t\t ALGORITHME 4\n");
252 //BOUCLE DE PARCOUR D'UN TABLEAU
253 for(j=0;j<taille1;j++){
254
255     //1ERE VAR DE TEMPS
256     t1 = clock();
257
258     //APPEL AU FONCTION
259     int test=Algorithme4(Tab1[j]);
260
261     //AFFICHAGE (PREMIER OU NON)
262     if(test == 0){
263         printf("***** Le nombre ( %llu ) n'est pas premier!\n",Tab1[j]);
264     }else{
265         printf("***** Le nombre ( %llu ) est premier.\n",Tab1[j]);
266     }
267
268     //2EME VAR DE TEMPS
269     t2 = clock();
270
271     //LA MOYENNE DE TEMPS D'EXECUTION
272     delta = (double)(t2-t1)/CLOCKS_PER_SEC;
273     reel = (float)delta;
274
275

```

Figure 13 : Code source correspondant au main question 1 partie II capture 3.

```

276 //AFFICHAGE DU TEMPS D'EXECUTION
277 printf("***** Le temps d'execution de (%llu) est: ( %lf )\n\n",Tab1[j],delta);
278
279 //SAISIE DANS LE FICHIER
280 fprintf(fichier,"%lf",delta);
281 }
282 fprintf(fichier,"\n");
283 //-----ALG05-----
284 printf("\n \t\t ALGORITHME 5\n");
285 //BOUCLE DE PARCOUR D'UN TABLEAU
286 for(j=0;j<taille1;j++){
287
288     //1ERE VAR DE TEMPS
289     t1 = clock();
290
291     //APPEL AU FONCTION
292     int test=Algorithme5(Tab1[j]);
293
294     //AFFICHAGE (PREMIER OU NON)
295     if(test == 0){
296         printf("***** Le nombre ( %llu ) n'est pas premier!\n",Tab1[j]);
297     }else{
298         printf("***** Le nombre ( %llu ) est premier.\n",Tab1[j]);
299     }
300
301     //2EME VAR DE TEMPS
302     t2 = clock();
303     //LA MOYENNE DE TEMPS D'EXECUTION
304     delta = (double)(t2-t1)/CLOCKS_PER_SEC;
305     reel = (float)delta;
306
307     //AFFICHAGE DU TEMPS D'EXECUTION
308     printf("***** Le temps d'execution de (%llu) est: ( %lf )\n\n",Tab1[j],delta);
309
310     //SAISIE DANS LE FICHIER
311     fprintf(fichier,"%lf",delta);
312 }
313 fprintf(fichier,"\n");

```

Figure 14 : Code source correspondant au main question 1 partie II capture 4.

```

314 //-----ALG06-----
315 printf("\n \t\t ALGORITHME 6\n");
316 //BOUCLE DE PARCOUR D'UN TABLEAU
317 for(j=0;j<taille1;j++){
318
319     //1ERE VAR DE TEMPS
320     t1 = clock();
321
322     //APPEL AU FONCTION
323     int test=Algorithme6(Tab1[j]);
324
325     //AFFICHAGE (PREMIER OU NON)
326     if(test == 0){
327         printf("***** Le nombre ( %llu ) n'est pas premier!\n",Tab1[j]);
328     }else{
329         printf("***** Le nombre ( %llu ) est premier.\n",Tab1[j]);
330     }
331
332     //2EME VAR DE TEMPS
333     t2 = clock();
334
335     //LA MOYENNE DE TEMPS D'EXECUTION
336     delta = (double)(t2-t1)/CLOCKS_PER_SEC;
337     reel = (float)delta;
338
339     //AFFICHAGE DU TEMPS D'EXECUTION
340     printf("***** Le temps d'execution de (%llu) est: ( %lf )\n\n",Tab1[j],delta);
341
342     //SAISIE DANS LE FICHIER
343     fprintf(fichier,"%lf",delta);
344 }
345 fprintf(fichier,"\n\n");
346
347

```

Figure 15 : Code source correspondant au main question 1 partie II capture 5.

Explications 02 :

Dans cette partie on a déclaré 3 tableaux : tab1[] , tab2[] et tab3[].

tab1[] correspond aux entiers premiers à 12 chiffres.

Pour connaître le choix de l'utilisateur on a utilisé un switch:

S'il tape 1 il exécute la première question qui consiste à la mesure du temps d'exécution de chacun des six algorithmes pour des nombres premiers à 12 chiffres et la deuxième question qui consiste à l'exécution des six algorithmes en prenant 20 nombres de même longueur.

S'il tape 2 il exécute la troisième question ("**Voir explications 03**").

Code source de mesure des temps d'exécution T pour chacun des six algorithmes en faisant des tests sur des nombres premiers ayant au plus 20 chiffres :

```
349 //-----QST2
350 printf("\n-----QST 2-----\n\n");
351 printf("\t\t ALGORITHME 1\n");
352 //BOUCLE DE PARCOUR D'UN TABLEAU
353 for(j=0;j<taille2;j++){
354
355     //1ERE VAR DE TEMPS
356     t1 = clock();
357
358     //APPEL AU FONCTION
359     int test=Algorithme1(Tab2[j]);
360
361     //AFFICHAGE (PREMIER OU NON)
362     if(test == 0){
363         printf("***** Le nombre ( %llu ) n'est pas premier!\n",Tab2[j]);
364     }else{
365         printf("***** Le nombre ( %llu ) est premier.\n",Tab2[j]);
366     }
367
368     //2EME VAR DE TEMPS
369     t2 = clock();
370
371     //LA MOYENNE DE TEMPS D'EXECUTION
372     delta = (double)(t2-t1)/CLOCKS_PER_SEC;
373     reel = (float)delta;
374
375     //AFFICHAGE DU TEMPS D'EXECUTION
376     printf("***** Le temps d'execution de (%llu) est: ( %lf )\n\n",Tab2[j],delta);
377
378     //SAISIE DANS LE FICHIER
379     fprintf(fichier,"%lf,",delta);
380 }
381 fprintf(fichier,"\n");
382
383 printf("\t\t ALGORITHME 2\n");
384 //BOUCLE DE PARCOUR D'UN TABLEAU
385 for(j=0;j<taille2;j++){
386
387     //1ERE VAR DE TEMPS
388     t1 = clock();
```


Figure 16 : Code source correspondant au main question 2 partie II capture 1.

```
387 //1ERE VAR DE TEMPS
388 t1 = clock();
389
390 //APPEL AU FONCTION
391 int test=Algorithme2(Tab2[j]);
392
393 //AFFICHAGE (PREMIER OU NON)
394 if(test == 0){
395     printf("***** Le nombre ( %llu ) n'est pas premier!\n",Tab2[j]);
396 }else{
397     printf("***** Le nombre ( %llu ) est premier.\n",Tab2[j]);
398 }
399 //2EME VAR DE TEMPS
400 t2 = clock();
401
402 //LA MOYENNE DE TEMPS D'EXECUTION
403 delta = (double)(t2-t1)/CLOCKS_PER_SEC;
404 reel = (float)delta;
405 //AFFICHAGE DU TEMPS D'EXECUTION
406 printf("***** Le temps d'execution de (%llu) est: ( %lf )\n\n",Tab2[j],delta);
407
408 //SAISIE DANS LE FICHIER
409 fprintf(fichier,"%lf,",delta);
410 }
411 fprintf(fichier,"\n");
412 printf("\t\t ALGORITHME 3\n");
413 //BOUCLE DE PARCOUR D'UN TABLEAU
414 for(j=0;j<taille2;j++){
415     //1ERE VAR DE TEMPS
416     t1 = clock();
417     //APPEL AU FONCTION
418     int test=Algorithme3(Tab2[j]);
419     //AFFICHAGE (PREMIER OU NON)
420     if(test == 0){
421         printf("***** Le nombre ( %llu ) n'est pas premier!\n",Tab2[j]);
422     }else{
423         printf("***** Le nombre ( %llu ) est premier.\n",Tab2[j]);
424     }
425 }
```

Figure 17 : Code source correspondant au main question 2 partie II capture 2.

```
426 //2EME VAR DE TEMPS
427 t2 = clock();
428
429 //LA MOYENNE DE TEMPS D'EXECUTION
430 delta = (double)(t2-t1)/CLOCKS_PER_SEC;
431 reel = (float)delta;
432 //AFFICHAGE DU TEMPS D'EXECUTION
433 printf("***** Le temps d'execution de (%llu) est: ( %lf )\n\n",Tab2[j],delta);
434 //SAISIE DANS LE FICHIER
435 fprintf(fichier,"%lf,",delta);
436 }
437 fprintf(fichier,"\n");
438
439 printf("\t\t ALGORITHME 4\n");
440
441 //BOUCLE DE PARCOUR D'UN TABLEAU
442 for(j=0;j<taille2;j++){
443     //1ERE VAR DE TEMPS
444     t1 = clock();
445     //APPEL AU FONCTION
446     int test=Algorithme4(Tab2[j]);
447     //AFFICHAGE (PREMIER OU NON)
448     if(test == 0){
449         printf("***** Le nombre ( %llu ) n'est pas premier!\n",Tab2[j]);
450     }else{
451         printf("***** Le nombre ( %llu ) est premier.\n",Tab2[j]);
452     }
453
454     //2EME VAR DE TEMPS
455     t2 = clock();
456     //LA MOYENNE DE TEMPS D'EXECUTION
457     delta = (double)(t2-t1)/CLOCKS_PER_SEC;
458     reel = (float)delta;
459     //AFFICHAGE DU TEMPS D'EXECUTION
460     printf("***** Le temps d'execution de (%llu) est: ( %lf )\n\n",Tab2[j],delta);
461     //SAISIE DANS LE FICHIER
462     fprintf(fichier,"%lf,",delta);
463 }
464 fprintf(fichier,"\n");
465 }
```

Figure 18 : Code source correspondant au main question 2 partie II capture 3.

```

458     reel = (float)delta;
459     //AFFICHAGE DU TEMPS D'EXECUTION
460     printf("***** Le temps d'execution de (%llu) est: ( %lf )\n\n",Tab2[j],delta);
461     //SAISIE DANS LE FICHIER
462     fprintf(fichier,"%lf,",delta);
463 }
464 fprintf(fichier,"\n");
465
466     printf("\t\t ALGORITHME 5\n");
467
468     //BOUCLE DE PARCOUR D'UN TABLEAU
469     for(j=0;j<taille2;j++){
470
471         //1ERE VAR DE TEMPS
472         t1 = clock();
473
474         //APPEL AU FONCTION
475         int test=Algorithme5(Tab2[j]);
476
477         //AFFICHAGE (PREMIER OU NON)
478         if(test == 0){
479             printf("***** Le nombre ( %llu ) n'est pas premier!\n",Tab2[j]);
480         }else{
481             printf("***** Le nombre ( %llu ) est premier.\n",Tab2[j]);
482         }
483
484         //2EME VAR DE TEMPS
485         t2 = clock();
486
487         //LA MOYENNE DE TEMPS D'EXECUTION
488         delta = (double)(t2-t1)/CLOCKS_PER_SEC;
489         reel = (float)delta;
490
491         //AFFICHAGE DU TEMPS D'EXECUTION
492         printf("***** Le temps d'execution de (%llu) est: ( %lf )\n\n",Tab2[j],delta);
493
494         //SAISIE DANS LE FICHIER
495         fprintf(fichier,"%lf,",delta);
496     }

```

Figure 19 : Code source correspondant au main question 2 partie II capture 4.

```

495         fprintf(fichier,"%lf,",delta);
496     }
497     fprintf(fichier,"\n");
498     printf("\t\t ALGORITHME 6\n");
499     //BOUCLE DE PARCOUR D'UN TABLEAU
500     for(j=0;j<taille2;j++){
501         //1ERE VAR DE TEMPS
502         t1 = clock();
503         //APPEL AU FONCTION
504         int test=Algorithme6(Tab2[j]);
505         //AFFICHAGE (PREMIER OU NON)
506         if(test == 0){
507             printf("***** Le nombre ( %llu ) n'est pas premier!\n",Tab2[j]);
508         }else{
509             printf("***** Le nombre ( %llu ) est premier.\n",Tab2[j]);
510         }
511
512         //2EME VAR DE TEMPS
513         t2 = clock();
514
515         //LA MOYENNE DE TEMPS D'EXECUTION
516         delta = (double)(t2-t1)/CLOCKS_PER_SEC;
517         reel = (float)delta;
518
519         //AFFICHAGE DU TEMPS D'EXECUTION
520         printf("***** Le temps d'execution de (%llu) est: ( %lf )\n\n",Tab2[j],delta);
521
522         //SAISIE DANS LE FICHIER
523         fprintf(fichier,"%lf,",delta);
524     }
525     fprintf(fichier,"\n\n");
526
527     fclose(fichier);
528
529     break;
530

```

Figure 20 : Code source correspondant au main question 2 partie II capture 5.

Explications 03 :

Ici on mesure le temps d'exécution de chacun des six algorithmes avec des nombres premiers à 20 chiffres en utilisant la méthode citée dans le support du TP (**"page1 partie II"**).

Code source pour des longueurs différentes de nombres, allant de 6 à 12, exécuter les 6 programmes 50 fois:

```
533 //-----QST3
534 //CREATION DU FICHIER QUI VA STOCKER LES RESULTATS DE LA QST 3
535 fichierQst3 = fopen("C:/Users/DELL/Desktop/TempPrem3.txt","w");
536
537 printf("-----QST 3-----\n\n");
538 printf("\t\t ALGORITHME 1\n");
539 //BOUCLE DE PARCOUR D'UN TABLEAU
540 for(j=0;j<taille3;j++){
541     for(nbrfois=0;nbrfois<30;nbrfois++){
542         //1ERE VAR DE TEMPS
543         t1 = clock();
544         //APPEL AU FONCTION
545         int test=Algorithme1(Tab3[j]);
546
547         //AFFICHAGE (PREMIER OU NON)
548         if(test == 0){
549             printf("***** Le nombre ( %llu ) n'est pas premier!\n",Tab3[j]);
550         }else{
551             printf("***** Le nombre ( %llu ) est premier.\n",Tab3[j]);
552         }
553         //2EME VAR DE TEMPS
554         t2 = clock();
555         //LA MOYENNE DE TEMPS D'EXECUTION
556         delta = (double)(t2-t1)/CLOCKS_PER_SEC;
557         reel = (float)delta;
558         Moyenne[j] = delta;
559     }
560     int t;
561     for(t=0;t<taille3;t++){
562         moy = (moy + Moyenne[t])/30;
563     }
564     printf("La moyenne du temps d'execution de %llu est %lf\n\n",Tab3[j],moy);
565     fprintf(fichierQst3,"%lf",moy);
566 }
567 fprintf(fichierQst3,"\n");
568
569 printf("\t\t ALGORITHME 2\n");
570 //BOUCLE DE PARCOUR D'UN TABLEAU
571 for(j=0;j<taille3;j++){
```

Figure 21 : Code source correspondant au main question 3 partie II capture 1.

```

560 int t;
561 for(t=0;t<taille3;t++){
562     moy = (moy + Moyenne[t])/30;
563 }
564 printf("La moyenne du temps d'execution de %llu est %lf\n\n",Tab3[j],moy);
565 fprintf(fichierQst3,"%lf",moy);
566 }
567 fprintf(fichierQst3,"\n");
568 printf("\t\t ALGORITHME 2\n");
569 //BOUCLE DE PARCOUR D'UN TABLEAU
570 for(j=0;j<taille3;j++){
571     for(nbrfois=0;nbrfois<50;nbrfois++){
572         //1ERE VAR DE TEMPS
573         t1 = clock();
574         //APPEL AU FONCTION
575         int test=Algorithme2(Tab3[j]);
576         //AFFICHAGE (PREMIER OU NON)
577         if(test == 0){
578             printf("***** Le nombre ( %llu ) n'est pas premier!\n",Tab3[j]);
579         }else{
580             printf("***** Le nombre ( %llu ) est premier.\n",Tab3[j]);
581         }
582         //2EME VAR DE TEMPS
583         t2 = clock();
584         //LA MOYENNE DE TEMPS D'EXECUTION
585         delta = (double)(t2-t1)/CLOCKS_PER_SEC;
586         reel = (float)delta;
587         Moyenne[j] = delta;
588     }
589     int t;
590     for(t=0;t<taille3;t++){
591         moy = (moy + Moyenne[t])/50;
592     }
593     printf("La moyenne du temps d'execution de %llu est %lf\n\n",Tab3[j],moy);
594     fprintf(fichierQst3,"%lf",moy);
595 }
596 fprintf(fichierQst3,"\n");
597
598 printf("\t\t ALGORITHME 3\n");

```

Figure 22 : Code source correspondant au main question 3 partie II capture 2.

```

598 printf("\t\t ALGORITHME 3\n");
599 //BOUCLE DE PARCOUR D'UN TABLEAU
600 for(j=0;j<taille3;j++){
601     for(nbrfois=0;nbrfois<50;nbrfois++){
602         //1ERE VAR DE TEMPS
603         t1 = clock();
604         //APPEL AU FONCTION
605         int test=Algorithme3(Tab3[j]);
606         //AFFICHAGE (PREMIER OU NON)
607         if(test == 0){
608             printf("***** Le nombre ( %llu ) n'est pas premier!\n",Tab3[j]);
609         }else{
610             printf("***** Le nombre ( %llu ) est premier.\n",Tab3[j]);
611         }
612         //2EME VAR DE TEMPS
613         t2 = clock();
614         //LA MOYENNE DE TEMPS D'EXECUTION
615         delta = (double)(t2-t1)/CLOCKS_PER_SEC;
616         reel = (float)delta;
617         Moyenne[j] = delta;
618     }
619     int t;
620     for(t=0;t<taille3;t++){
621         moy = (moy + Moyenne[t])/50;
622     }
623     printf("La moyenne du temps d'execution de %llu est %lf\n\n",Tab3[j],moy);
624     fprintf(fichierQst3,"%lf",moy);
625 }
626 fprintf(fichierQst3,"\n");
627 printf("\t\t ALGORITHME 4\n");
628 //BOUCLE DE PARCOUR D'UN TABLEAU
629 for(j=0;j<taille3;j++){
630     for(nbrfois=0;nbrfois<50;nbrfois++){
631         //1ERE VAR DE TEMPS
632         t1 = clock();
633         //APPEL AU FONCTION
634         int test=Algorithme4(Tab3[j]);
635
636

```

Figure 23 : Code source correspondant au main question 3 partie II capture 3.

```

637 //AFFICHAGE (PREMIER OU NON)
638 if(test == 0){
639     printf("***** Le nombre ( %llu ) n'est pas premier!\n",Tab3[j]);
640 }else{
641     printf("***** Le nombre ( %llu ) est premier.\n",Tab3[j]);
642 }
643 //2EME VAR DE TEMPS
644 t2 = clock();
645 //LA MOYENNE DE TEMPS D'EXECUTION
646 delta = (double)(t2-t1)/CLOCKS_PER_SEC;
647 reel = (float)delta;
648 Moyenne[j] = delta;
649 }
650 int t;
651 for(t=0;t<taille3;t++){
652     moy = (moy + Moyenne[t])/50;
653 }
654 printf("La moyenne du temps d'execution de %llu est %lf\n\n",Tab3[j],moy);
655 fprintf(fichierQst3,"%lf",moy);
656 }
657 fprintf(fichierQst3,"\n");
658 printf("\t\t ALGORITHME 5\n");
659 //BOUCLE DE PARCOUR D'UN TABLEAU
660 for(j=0;j<taille3;j++){
661     for(nbrfois=0;nbrfois<50;nbrfois++){
662         //1ERE VAR DE TEMPS
663         t1 = clock();
664         //APPEL AU FONCTION
665         int test=Algorithme5(Tab3[j]);
666         //AFFICHAGE (PREMIER OU NON)
667         if(test == 0){
668             printf("***** Le nombre ( %llu ) n'est pas premier!\n",Tab3[j]);
669         }else{
670             printf("***** Le nombre ( %llu ) est premier.\n",Tab3[j]);
671         }
672         //2EME VAR DE TEMPS
673         t2 = clock();

```

Figure 24 : Code source correspondant au main question 3 partie II capture 4.

```

675 //LA MOYENNE DE TEMPS D'EXECUTION
676 delta = (double)(t2-t1)/CLOCKS_PER_SEC;
677 reel = (float)delta;
678 Moyenne[j] = delta;
679 }
680 int t;
681 for(t=0;t<taille3;t++){
682     moy = (moy + Moyenne[t])/50;
683 }
684 printf("La moyenne du temps d'execution de %llu est %lf\n\n",Tab3[j],moy);
685 fprintf(fichierQst3,"%lf",moy);
686 }
687 fprintf(fichierQst3,"\n");
688 printf("\t\t ALGORITHME 6\n");
689 //BOUCLE DE PARCOUR D'UN TABLEAU
690 for(j=0;j<taille3;j++){
691     for(nbrfois=0;nbrfois<50;nbrfois++){
692         //1ERE VAR DE TEMPS
693         t1 = clock();
694         //APPEL AU FONCTION
695         int test=Algorithme6(Tab3[j]);
696         //AFFICHAGE (PREMIER OU NON)
697         if(test == 0){
698             printf("***** Le nombre ( %llu ) n'est pas premier!\n",Tab3[j]);
699         }else{
700             printf("***** Le nombre ( %llu ) est premier.\n",Tab3[j]);
701         }
702         //2EME VAR DE TEMPS
703         t2 = clock();
704         //LA MOYENNE DE TEMPS D'EXECUTION
705         delta = (double)(t2-t1)/CLOCKS_PER_SEC;
706         reel = (float)delta;
707         Moyenne[j] = delta;
708     }
709     int t;
710     for(t=0;t<taille3;t++){
711         moy = (moy + Moyenne[t])/50;
712     }
713 }

```

Figure 25 : Code source correspondant au main question 3 partie II capture 5.

```

691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729

        for(j=0;j<taille3;j++){
            for(nbrfois=0;nbrfois<50;nbrfois++){
                //1ERE VAR DE TEMPS
                t1 = clock();
                //APPEL AU FONCTION
                int test=Algorithme6(Tab3[j]);
                //AFFICHAGE (PREMIER OU NON)
                if(test == 0){
                    printf("***** Le nombre ( %llu ) n'est pas premier!\n",Tab3[j]);
                }else{
                    printf("***** Le nombre ( %llu ) est premier.\n",Tab3[j]);
                }
                //2EME VAR DE TEMPS
                t2 = clock();
                //LA MOYENNE DE TEMPS D'EXECUTION
                delta = (double)(t2-t1)/CLOCKS_PER_SEC;
                reel = (float)delta;
                Moyenne[j] = delta;
            }
            int t;
            for(t=0;t<taille3;t++){
                moy = (moy + Moyenne[t])/50;
            }
            printf("La moyenne du temps d'execution de %llu est %lf\n\n",Tab3[j],moy);
            fprintf(fichierQst3,"%lf",moy);
        }
        fprintf(fichierQst3,"\n");

        fclose(fichierQst3);
        break;
    default:
        printf("\nErreur! Choix n'existe pas.\n\n");
}

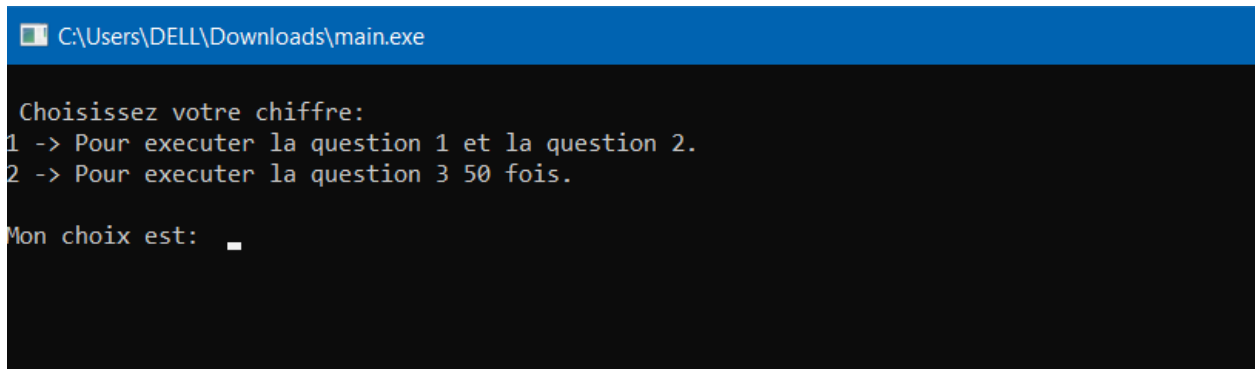
```

Figure 26 : Code source correspondant au main question 3 partie II capture 6.

Explications :

On passe maintenant à l'exécution des six algorithmes 50 fois ou il est question d'évaluer le temps d'exécution de ces derniers en indiquant en entrée des nombres premiers de longueur variant entre 6 et 12 chiffres.

Capture correspondante à la fenêtre d'exécution :



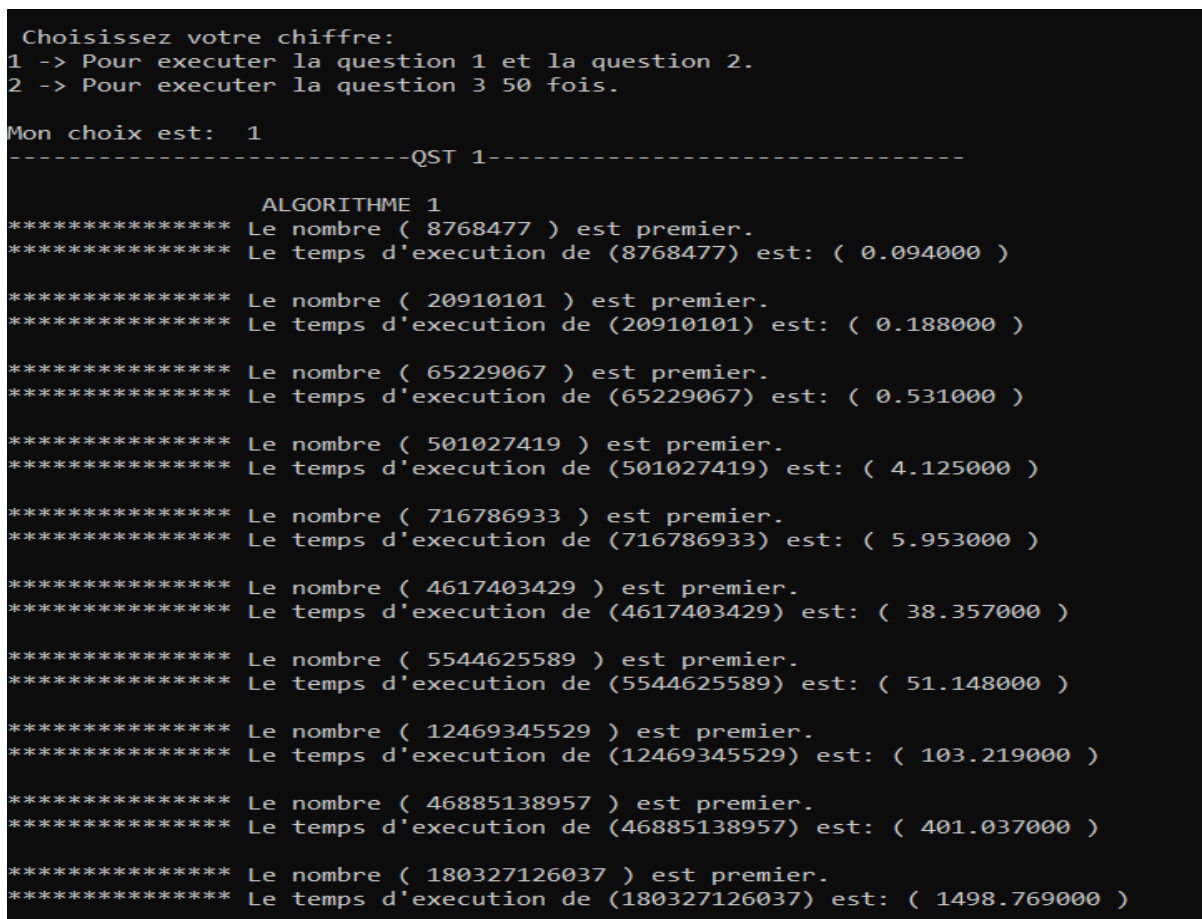
```
C:\Users\DELL\Downloads\main.exe

Choisissez votre chiffre:
1 -> Pour executer la question 1 et la question 2.
2 -> Pour executer la question 3 50 fois.

Mon choix est: _
```

Figure 27 : fenêtre d'exécution.

Exécutons par exemple la question 1 :



```
Choisissez votre chiffre:
1 -> Pour executer la question 1 et la question 2.
2 -> Pour executer la question 3 50 fois.

Mon choix est: 1
-----QST 1-----

          ALGORITHME 1
*****
***** Le nombre ( 8768477 ) est premier.
***** Le temps d'execution de (8768477) est: ( 0.094000 )

*****
***** Le nombre ( 20910101 ) est premier.
***** Le temps d'execution de (20910101) est: ( 0.188000 )

*****
***** Le nombre ( 65229067 ) est premier.
***** Le temps d'execution de (65229067) est: ( 0.531000 )

*****
***** Le nombre ( 501027419 ) est premier.
***** Le temps d'execution de (501027419) est: ( 4.125000 )

*****
***** Le nombre ( 716786933 ) est premier.
***** Le temps d'execution de (716786933) est: ( 5.953000 )

*****
***** Le nombre ( 4617403429 ) est premier.
***** Le temps d'execution de (4617403429) est: ( 38.357000 )

*****
***** Le nombre ( 5544625589 ) est premier.
***** Le temps d'execution de (5544625589) est: ( 51.148000 )

*****
***** Le nombre ( 12469345529 ) est premier.
***** Le temps d'execution de (12469345529) est: ( 103.219000 )

*****
***** Le nombre ( 46885138957 ) est premier.
***** Le temps d'execution de (46885138957) est: ( 401.037000 )

*****
***** Le nombre ( 180327126037 ) est premier.
***** Le temps d'execution de (180327126037) est: ( 1498.769000 )
```

Figure 28 : fenêtre d'exécution question 1 algorithme 01.


```

          ALGORITHME 2
*****
Le nombre ( 8768477 ) est premier.
*****
Le temps d'execution de (8768477) est: ( 0.047000 )

*****
Le nombre ( 20910101 ) est premier.
*****
Le temps d'execution de (20910101) est: ( 0.125000 )

*****
Le nombre ( 65229067 ) est premier.
*****
Le temps d'execution de (65229067) est: ( 0.375000 )

*****
Le nombre ( 501027419 ) est premier.
*****
Le temps d'execution de (501027419) est: ( 2.891000 )

*****
Le nombre ( 716786933 ) est premier.
*****
Le temps d'execution de (716786933) est: ( 4.127000 )

*****
Le nombre ( 4617403429 ) est premier.
*****
Le temps d'execution de (4617403429) est: ( 27.511000 )

*****
Le nombre ( 5544625589 ) est premier.
*****
Le temps d'execution de (5544625589) est: ( 35.946000 )

*****
Le nombre ( 12469345529 ) est premier.
*****
Le temps d'execution de (12469345529) est: ( 79.975000 )

*****
Le nombre ( 46885138957 ) est premier.
*****
Le temps d'execution de (46885138957) est: ( 270.683000 )

*****
Le nombre ( 180327126037 ) est premier.
*****
Le temps d'execution de (180327126037) est: ( 1054.825000 )

```

Figure 29 : fenêtre d'exécution question 1 algorithme 02.

```

          ALGORITHME 3
*****
Le nombre ( 8768477 ) est premier.
*****
Le temps d'execution de (8768477) est: ( 0.032000 )

*****
Le nombre ( 20910101 ) est premier.
*****
Le temps d'execution de (20910101) est: ( 0.093000 )

*****
Le nombre ( 65229067 ) est premier.
*****
Le temps d'execution de (65229067) est: ( 0.282000 )

*****
Le nombre ( 501027419 ) est premier.
*****
Le temps d'execution de (501027419) est: ( 2.187000 )

*****
Le nombre ( 716786933 ) est premier.
*****
Le temps d'execution de (716786933) est: ( 3.109000 )

*****
Le nombre ( 4617403429 ) est premier.
*****
Le temps d'execution de (4617403429) est: ( 20.233000 )

*****
Le nombre ( 5544625589 ) est premier.
*****
Le temps d'execution de (5544625589) est: ( 24.045000 )

*****
Le nombre ( 12469345529 ) est premier.
*****
Le temps d'execution de (12469345529) est: ( 54.264000 )

*****
Le nombre ( 46885138957 ) est premier.
*****
Le temps d'execution de (46885138957) est: ( 204.010000 )

*****
Le nombre ( 180327126037 ) est premier.
*****
Le temps d'execution de (180327126037) est: ( 794.301000 )

```

Figure 30 : fenêtre d'exécution question 1 algorithme 03.


```

ALGORITHME 4
*****
Le nombre ( 8768477 ) est premier.
*****
Le temps d'execution de (8768477) est: ( 0.031000 )

*****
Le nombre ( 20910101 ) est premier.
*****
Le temps d'execution de (20910101) est: ( 0.063000 )

*****
Le nombre ( 65229067 ) est premier.
*****
Le temps d'execution de (65229067) est: ( 0.187000 )

*****
Le nombre ( 501027419 ) est premier.
*****
Le temps d'execution de (501027419) est: ( 1.406000 )

*****
Le nombre ( 716786933 ) est premier.
*****
Le temps d'execution de (716786933) est: ( 2.016000 )

*****
Le nombre ( 4617403429 ) est premier.
*****
Le temps d'execution de (4617403429) est: ( 12.985000 )

*****
Le nombre ( 5544625589 ) est premier.
*****
Le temps d'execution de (5544625589) est: ( 15.624000 )

*****
Le nombre ( 12469345529 ) est premier.
*****
Le temps d'execution de (12469345529) est: ( 35.072000 )

*****
Le nombre ( 46885138957 ) est premier.
*****
Le temps d'execution de (46885138957) est: ( 140.712000 )

*****
Le nombre ( 180327126037 ) est premier.
*****
Le temps d'execution de (180327126037) est: ( 564.270000 )

```

Figure 31 : fenêtre d'exécution question 1 algorithme 4.

```

ALGORITHME 5
*****
Le nombre ( 8768477 ) est premier.
*****
Le temps d'execution de (8768477) est: ( 0.000000 )

*****
Le nombre ( 20910101 ) est premier.
*****
Le temps d'execution de (20910101) est: ( 0.000000 )

*****
Le nombre ( 65229067 ) est premier.
*****
Le temps d'execution de (65229067) est: ( 0.000000 )

*****
Le nombre ( 501027419 ) est premier.
*****
Le temps d'execution de (501027419) est: ( 0.000000 )

*****
Le nombre ( 716786933 ) est premier.
*****
Le temps d'execution de (716786933) est: ( 0.000000 )

*****
Le nombre ( 4617403429 ) est premier.
*****
Le temps d'execution de (4617403429) est: ( 0.016000 )

*****
Le nombre ( 5544625589 ) est premier.
*****
Le temps d'execution de (5544625589) est: ( 0.000000 )

*****
Le nombre ( 12469345529 ) est premier.
*****
Le temps d'execution de (12469345529) est: ( 0.000000 )

*****
Le nombre ( 46885138957 ) est premier.
*****
Le temps d'execution de (46885138957) est: ( 0.000000 )

*****
Le nombre ( 180327126037 ) est premier.
*****
Le temps d'execution de (180327126037) est: ( 0.000000 )

```

Figure 32 : fenêtre d'exécution question 1 algorithme 05.

```

                                ALGORITHME 6
*****
***** Le nombre ( 8768477 ) est premier.
***** Le temps d'execution de (8768477) est: ( 0.000000 )
*****
***** Le nombre ( 20910101 ) est premier.
***** Le temps d'execution de (20910101) est: ( 0.000000 )
*****
***** Le nombre ( 65229067 ) est premier.
***** Le temps d'execution de (65229067) est: ( 0.000000 )
*****
***** Le nombre ( 501027419 ) est premier.
***** Le temps d'execution de (501027419) est: ( 0.000000 )
*****
***** Le nombre ( 716786933 ) est premier.
***** Le temps d'execution de (716786933) est: ( 0.000000 )
*****
***** Le nombre ( 4617403429 ) est premier.
***** Le temps d'execution de (4617403429) est: ( 0.000000 )
*****
***** Le nombre ( 5544625589 ) est premier.
***** Le temps d'execution de (5544625589) est: ( 0.000000 )
*****
***** Le nombre ( 12469345529 ) est premier.
***** Le temps d'execution de (12469345529) est: ( 0.016000 )
*****
***** Le nombre ( 46885138957 ) est premier.
***** Le temps d'execution de (46885138957) est: ( 0.000000 )
*****
***** Le nombre ( 180327126037 ) est premier.
***** Le temps d'execution de (180327126037) est: ( 0.015000 )

```

Figure 33 : fenêtre d'exécution question 1 algorithme 06.

Répartition des tâches :

- **BOUADI Nassima**

- Le code algorithmique des 3 fonctions de la question 1 partie I.
- Les pseudo code.
- La rédaction du rapport et son organisation (sommaire, table de figure).
- Analyse du graphe et conclusion de la question 1 partie II.
- Participation aux réponses des questions.

- **FERKOUS Sarah**

- Écriture du code source main() (langage c).
- Faire des seconds tests d'exécution.
- Participation aux réponses des questions.
- Des modifications dans le rapport
- Participation à l'introduction et participation à l'analyse générale et la conclusion générale.

- **GUERBAS Tinhinane**

- Le code algorithmique des 3 autres fonctions de la question 1 partie I.
- Générer les résultats d'exécution.
- Analyse des graphes et conclusions des questions 2 et 3 partie II.
- Participation aux réponses des autres questions.
- Effectuation des recherches.
- Ecriture de la conclusion.

- **MOKHTARI Mohamed Rayane**

- Le dessin des graphes en python.
- Participation aux réponses des questions.
- Le choix des nombres premiers.

Bibliographie:

[1]http://www.monlyceenumerique.fr/nsi_premiere/algo_a/a2_complexite.php[2]
<https://www.ionos.fr/digitalguide>

[2] <https://buzut.net/cours/computer-science/time-complexity>

[3] livre de Mme DARIAS