**INF8775 – Analyse et conception d’algorithmes**

Rapport TP1 – Hiver 2022

| **Nom, prénom, matricule des membres** | Mohammed Amin, Nawras, 1962832  Patel, Pritam, 1933097 |
| --- | --- |
| **Note finale / 30** |  |

# Informations techniques

* Répondez directement dans ce document DOCX. Veuillez ne pas inclure le texte en italique servant de directive.
* La correction se fait sur ce même rapport.
* Vous devez faire une remise électronique sur Moodle avant le 23 Février à 23h59 en suivant les instructions suivantes :
  + Vos fichiers doivent être remis dans une archive zip à la racine de laquelle on retrouve :
    - Ce rapport au format DOCX.
    - 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.
    - Si le langage que vous utilisez nécessite une phase de compilation, veuillez joindre un Makefile afin que nous puissions le compiler en cas de problème avec vos exécutables. Si nous ne sommes pas en mesure de tester votre code, vous perdrez des points de respect d’interface et de qualité de code !
* Vous avez le choix du langage de programmation utilisé mais vous devrez utiliser les mêmes langage, compilateur et ordinateur pour toutes vos implantations. Le code et les exécutables soumis devront être compatibles avec les ordinateurs de la salle L-4714.
* Si vous utilisez des extraits de codes (programmes) trouvés sur Internet, vous devez en mentionner la source, sinon vous serez sanctionnés pour plagiat.

# Présentation des résultats

|  | / 1,5pts |
| --- | --- |

## Tableau des résultats

Tableau 1: Temps d’exécution moyen pour chaque algorithme en fonction de la taille d’exemplaire.

|  | Temps d’exécution moyen pour chaque algorithme (ms) | | |
| --- | --- | --- | --- |
| Taille de l’exemplaire | Force brute | Diviser pour régner | Diviser pour régner avec seuil = 85 |
| 1000 | 25.80838203 | 0.05520195961 | 0.02640419006 |
| 5000 | 231.1214447 | 0.1103949547 | 0.1131776333 |
| 10000 | 835.3308201 | 0.1284232616 | 0.1273598194 |
| 100000 | 23976.86615 | 0.3104996681 | 0.1327002645 |
| 500000 | 4306888.575 | 1.178253174 | 1.184924746 |

## 

# Analyse et discussion

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

|  | / 1,5 pt |
| --- | --- |

L’algorithme naïf est d’ordre O(n2). L’algorithme appliquant le patron de conception de diviser pour régner est d’ordre O(nlog(n)). L’algorithme appliquant le patron de conception de diviser pour régner avec seuil est d’ordre O(nlog(n)).

## Tests de puissance

|  | / 5 pt |
| --- | --- |

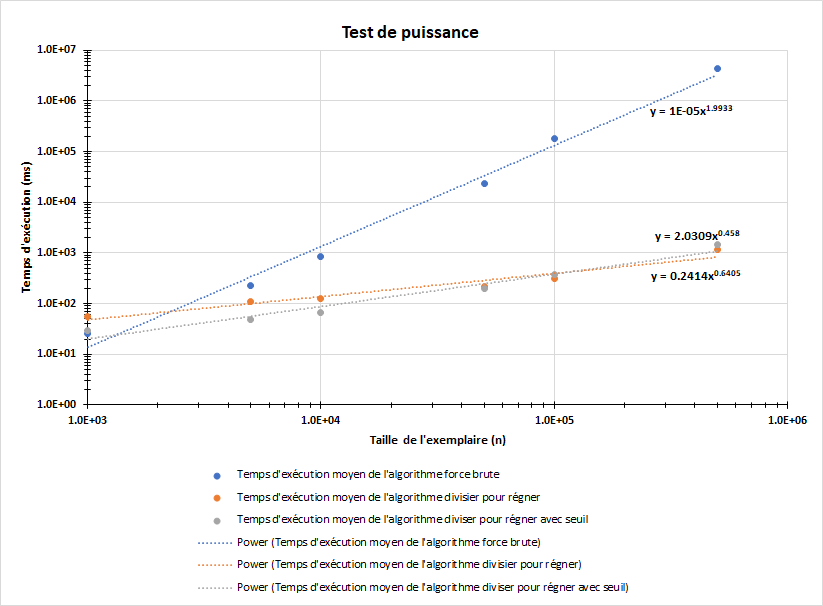


Figure 1: Représentation graphique du test de puissance effectué sur l’algorithme naïf (algorithme appliquant la force brute), l’algorithme diviser pour régner, et l’algorithme diviser pour régner avec seuil.

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

Le test de puissance est utile pour avoir une idée générale du taux de croissance de la consommation de ressources des algorithmes utilisés. En regroupant les résultats présentés dans le tableau 1, il est possible de former un graphique sous une échelle logarithmique qui révèle une régression linéaire pour chacun des algorithmes. Ainsi, on remarque que les algorithmes ont une croissance polynomiale de degré précise. Les résultats des tests de puissance avec la croissance polynomiale sont rapportés dans le tableau 2 ci-dessous.

* 1. Tableau 2: Comparaison des résultats des tests de puissances des algorithmes

| Algorithme | Fonction polynomiale | Degré de croissance théorique | Degré de croissance expérimentale |
| --- | --- | --- | --- |
| Force brute |  | 2 | 1.9933 |
| Diviser pour régner |  | [0, 1] | 0.4580 |
| Diviser pour régner avec seuil |  | [0, 1] | 0.6405 |

Pour ce qui est de l'algorithme de force brute, étant de complexité O(n2), on s'attend théoriquement à un degré de croissance égale à 2. Nos résultats expérimentaux affirment la valeur théorique. En fait, pour cet algorithme, on obtient un degré de croissance très proche à 2, soit de 1.9933. Ainsi, on peut utiliser cette valeur pour les prochains tests. Pour ce qui est des algorithmes diviser pour régner et diviser pour régner avec seuil, étant de complexité O(nlog(n)), on s'attend théoriquement à un taux de croissance qui est entre 0 et 1. Nos résultats confirment encore une fois la théorie. Effectivement, on obtient respectivement un degré polynomial de 0.4580 et 0.6405. Il est donc possible de garder ces valeurs expérimentales.

En analysant le graphique, on remarque bel et bien que l'algorithme de force brute est avantageux pour certaines tailles d'exemplaires. On note également que l'algorithme diviser pour régner avec seuil qui utilise l'algorithme de force brute a un taux de croissance légèrement plus élevé, mais est plus avantageux pour les tailles d'exemplaires plus petits.

## 

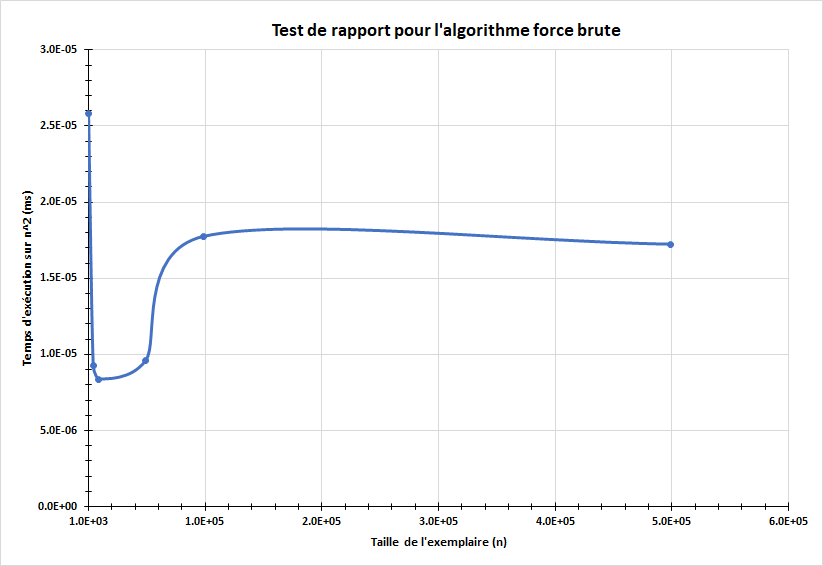
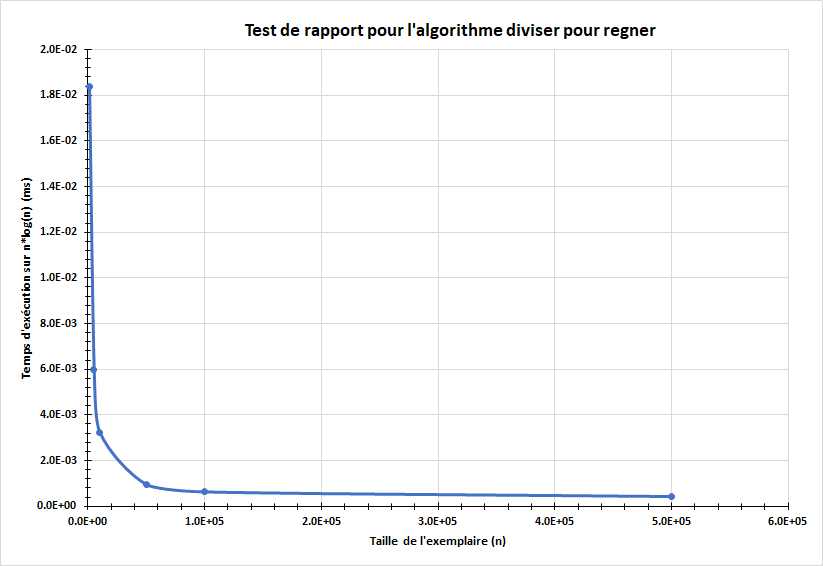
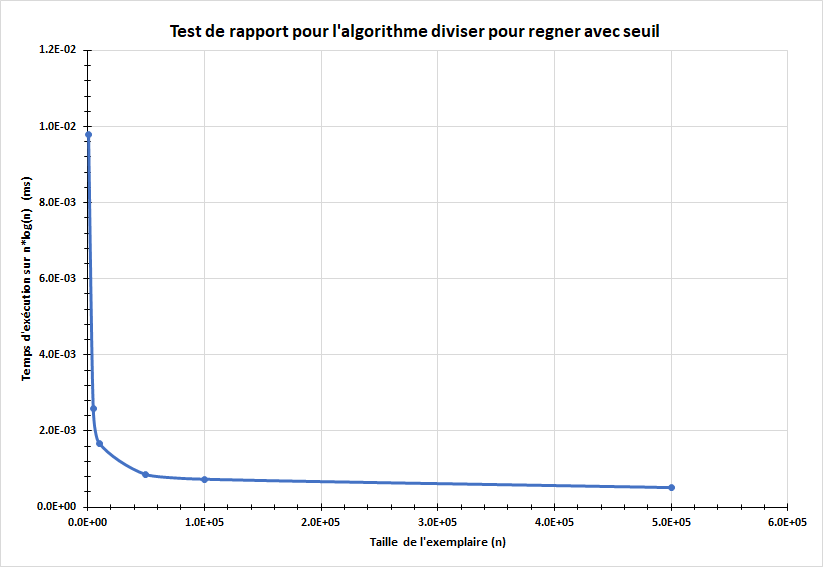
## 

## 

## 

## Test du rapport

|  | / 5 pt |
| --- | --- |

1. 
2. Figure 2: Représentation graphique du test de de rapport effectué sur l’algorithme naïf (algorithme appliquant la force brute).
3. 
4. Figure 3: Représentation graphique du test de constantes effectué sur l’implantation de la solution avec l’algorithme récursif (algorithme appliquant le patron de conception de diviser pour régner).
5. 
6. Figure 4: Représentation graphique du test de rapport effectué sur l’implantation de la solution avec l’algorithme récursif avec seuil (algorithme appliquant le patron de conception de diviser pour régner avec seuil)

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

Le test de rapport nous permet de vérifier si notre hypothèse selon laquelle le f(x), représenté par le rapport de la consommation de ressources sur la complexité temporelle, est croissant ou non.   
  
En ce qui concerne l'algorithme de force brute, notre test de rapport montre qu'il y a une convergence vers 0 pour les premières tailles d'entrée, notamment de 1000 à 50000 inclusivement. Cependant, cette convergence a été interrompue lorsque la taille d'entrée était de 100000. Ainsi, nous pouvons conclure que l'hypothèse est une sous-estimation.

En ce qui concerne les deux algorithmes récursifs, nous pouvons voir qu'il y a effectivement une convergence vers une constante b qui est plus grande que 0. Ceci conclut que l'hypothèse est confirmée et que la valeur vers laquelle on converge est la constante multiplicative f(x).

## Test des constantes

|  | / 5 pt |
| --- | --- |

## 

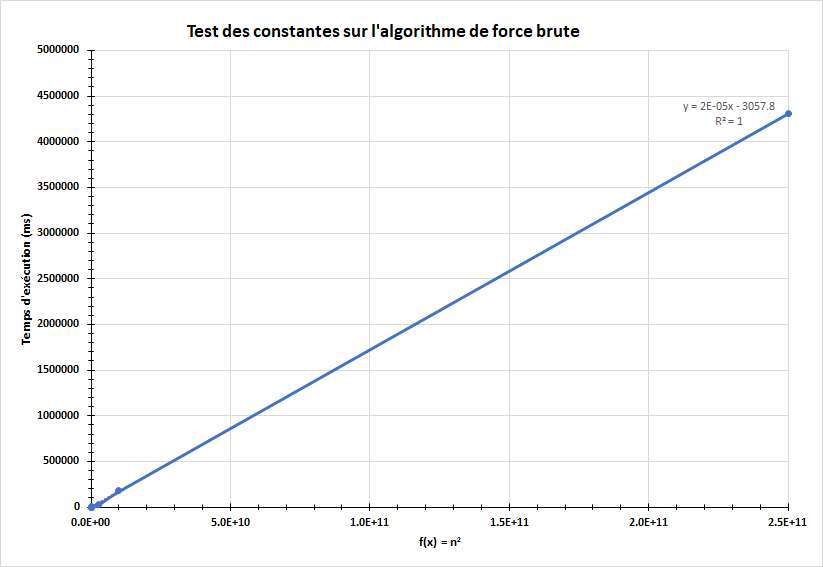
**

Figure 5: Représentation graphique du test de constantes effectué sur l’algorithme naïf (algorithme appliquant la force brute).

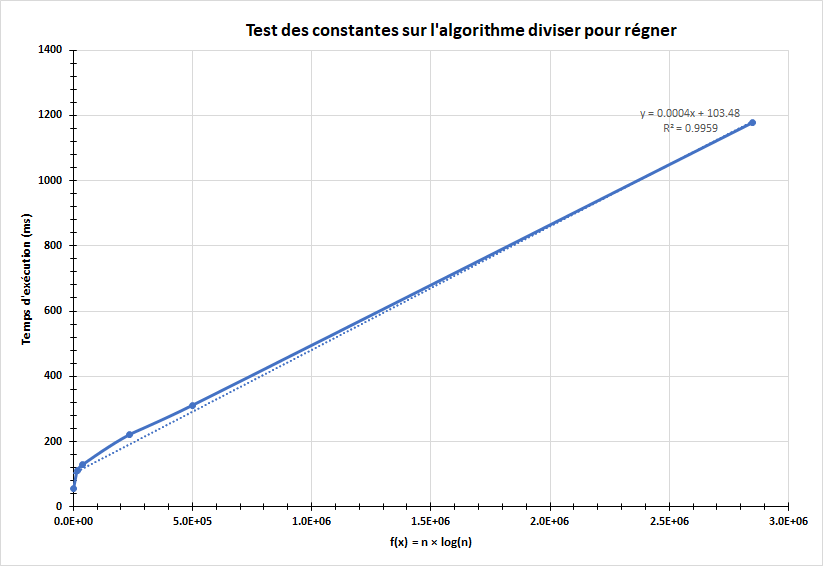
**

Figure 6: Représentation graphique du test de constantes effectué sur l’implantation de la solution avec l’algorithme récursif (algorithme appliquant le patron de conception de diviser pour régner).

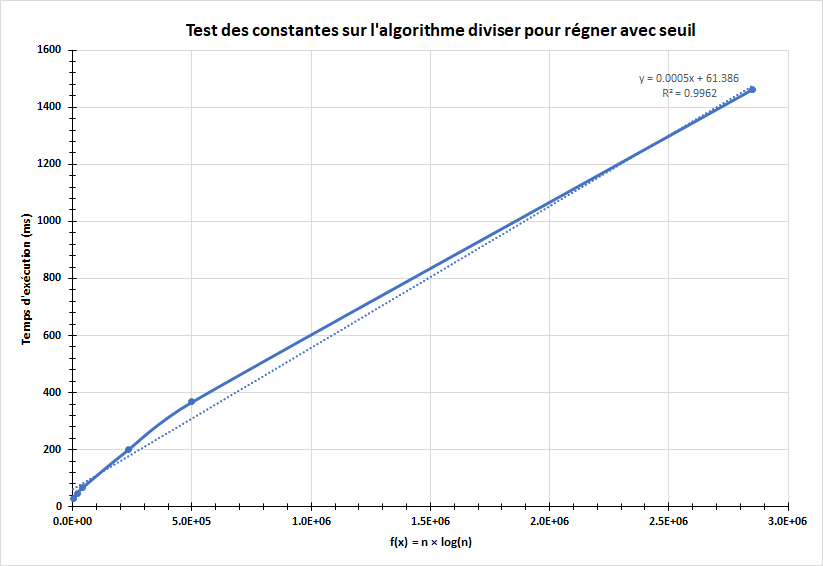
**

Figure 7: Représentation graphique du test de constantes effectué sur l’implantation de la solution avec l’algorithme récursif avec seuil (algorithme appliquant le patron de conception de diviser pour régner avec seuil)

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

En ce qui concerne l'algorithme de force brute, nous pouvons voir une bonne corrélation entre le test et les résultats expérimentaux, tels que l’augmentation de la taille de l’exemplaire augmente de façon linéaire l’utilisation des ressources. étant donné l'équation y = a(f(x)) + b, la pente du graphique étant de 2x10-5 et l'ordonnée à l'origine étant de -3057,8, nous pouvons comprendre que le coût initial en temps de l'approche par force brute est de -3057,8 et qu'il augmente à mesure que l'entrée augmente d'une constante fixe d’environ 2x10-5. Cela peut également être vu à travers les résultats observés, car les entrées plus petites ont tendance à être plus efficaces que les autres algorithmes, ce qui est quelque peu représenté par le "remboursement" du coût fourni par la valeur négative de B.

En ce qui concerne les algorithmes "diviser pour régner", nous constatons une apparence similaire à leurs graphiques de test des constantes. Les résultats nous montrent que la consommation des ressources est plus élevée dans les cas des exemplaires moins importants, ce qui est le cas vu expérimentalement aussi. La régression linéaire des graphiques des deux algorithmes récursifs montrent que l’algorithme sans seuil tend à être plus rapide alors que la taille des exemplaires augmente, mais est moins rapide si le nombre d'exemplaires est peu, ce qui est aussi vu expérimentalement.

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

|  | / 2 pt |
| --- | --- |

## 

Les seuils dans les algorithmes de type diviser pour régner sont importants pour déterminer exactement à quel moment il est plus efficace d'utiliser un algorithme différent pour résoudre le problème subdivisé, plutôt que de le diviser davantage.

Théoriquement, les résultats devraient montrer que le seuil fournit une augmentation des performances (ou une diminution du temps d'exécution) lorsque le seuil est défini pour utiliser l'algorithme de force brute pour les cas ayant une entrée inférieure à plus ou moins 100. En effet, l'algorithme de force brute a un temps d'exécution moyen inférieur lorsque la taille de l'entrée est inférieure ou égale à 1000, par rapport au temps d'exécution de l'algorithme de diviser pour régner.

Expérimentalement, nous constatons que c'est plus ou moins le cas. L'approche récursive avec seuil montre une légère diminution du temps d'exécution par rapport à l'approche récursive autonome lorsque la taille d'entrée est de 1000. Cette diminution est toutefois négligeable lorsque la taille de l'entrée augmente.   
  
Trouver le bon seuil est une tâche très difficile à réaliser en théorie, compte tenu du nombre de facteurs qui interviennent dans la sélection, notamment l'implémentation de l'algorithme, ainsi que la puissance du système qui exécute le programme. Pour nos tests, nous avons fixé le seuil à 85, car à travers plusieurs cas de test, nous avons constaté que le seuil optimal se situe entre 70 et 95.

## Suite à cette analyse, indiquez sous quelles conditions (taille d’exemplaire ou autre) vous utiliseriez chacun de ces algorithmes. Prenez en compte la complexité spatiale et temporelle, le temps de calcul et la difficulté d'implémentation. Justifiez.

|  | / 3 pt |
| --- | --- |

Compte tenu des résultats de nos tests présentés dans la première section, nous pouvons clairement voir que l'approche "diviser pour régner" est significativement supérieure à l'approche par force brute, en particulier lorsque la quantité de données augmente. L'approche par force brute a tendance à être plus rapide lorsqu'elle exécute la solution avec des tailles d'entrée comprises entre 0 et plus ou moins 1000. L'approche récursive avec un seuil s'est révélée plus rapide que l'approche récursive lorsque la taille de l’entrée était entre 0 et 50000, surtout au vu du test de puissance. Les exemplaires de tailles de plus ou moins 1000 peuvent être résolus avec l’algorithme de force brute ou l’algorithme récursif avec seuil.

Il est important de noter qu’un seuil mal choisi pourrait diminuer de manière significative les performances de l'approche récursive avec seuil, en particulier avec des quantités d'entrées plus importantes. En effet, la complexité temporelle théorique de cette méthode est calculée par l'équation suivante T(n) = 2T(n/2) + n2. Ce qui, en utilisant le théorème du maître, donne un ordre de complexité de O(n2). Cependant, ceci n'est pas exact. Par une analyse amortie, nous pouvons déduire que le coût général des opérations de l'algorithme tend à être de l'ordre de O(nlog(n)) pour la majorité des cas, en dehors des ceux où le seuil est suffisamment bas pour que l'algorithme soit calculé avec la méthode de la force brute. Dans ces cas, l'entrée doit être suffisamment petite pour ne pas tenir compte de la complexité théorique d'O(n^2) de l'algorithme naïf. Ainsi, si l'entrée n'est pas petite, alors le seuil n'a pas été choisi correctement, et la complexité augmentera davantage, pour atteindre éventuellement le pire cas de O(n2).

En termes de complexité spatiale, les trois algorithmes ont une complexité spatiale de O(n). La quantité de points stockés correspond au double du nombre d'entrées dans le pire des cas, ce qui d’ordre O(n). En termes de difficultés d’implémentation, les trois algorithmes ont une difficulté d’implémentation relativement similaire.

Enfin, un autre élément important à prendre en compte est la taille de la pile. Avec une taille d'entrée suffisamment grande, la pile peut déborder et ainsi réduire les performances du programme, voire l'arrêter complètement. Cependant, aucune taille d'entrée de ce type n'a été donnée ou testée pour ce TP.

Ainsi, dans les cas où nous avons un taille d’entrée entre 1000 et 50000 inclusivement, il serait idéal d’utiliser l’algorithme récursif avec seuil. Dans les cas où nous avons une taille d’entrée entre 100000 et 500000, il serait idéal d’utiliser l’algorithme récursif sans seuil. Finalement, si la taille est de 1000 ou moins inclusivement, l’algorithme de force brute est aussi une option valide.

# Autres critères de correction

## Respect de l’interface tp.sh

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

## Qualité du code

|  | / 5 pt |
| --- | --- |

* + - * 1. Validité des solutions
        2. Qualité de l'implémentation

Présence de commentaires

## Présentation générale

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

* Concision
* Qualité du français

## Pénalité retard

| 0 |
| --- |

* -15% de la note / journée de retard, arrondi vers le haut. Les TPs ne sont plus acceptés après 3 jours.