



Algorithmique et Complexité : TP1

MÉTAYER Stevan
GEFFROY Noam

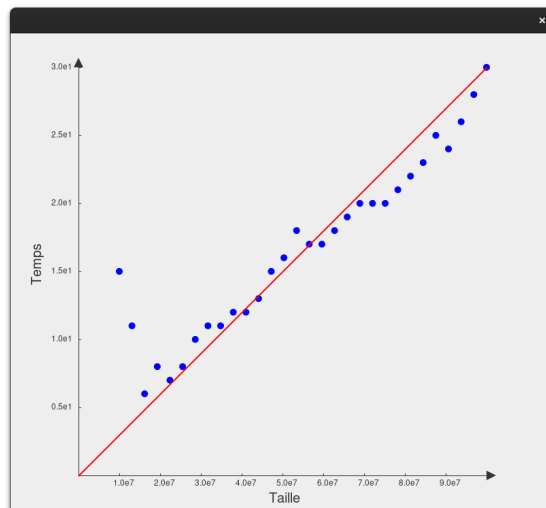
Sommaire

1	Analyse de tris	2
1.1	Tri par insertion	2
1.2	Tri fusion	3
2	Permutations	4

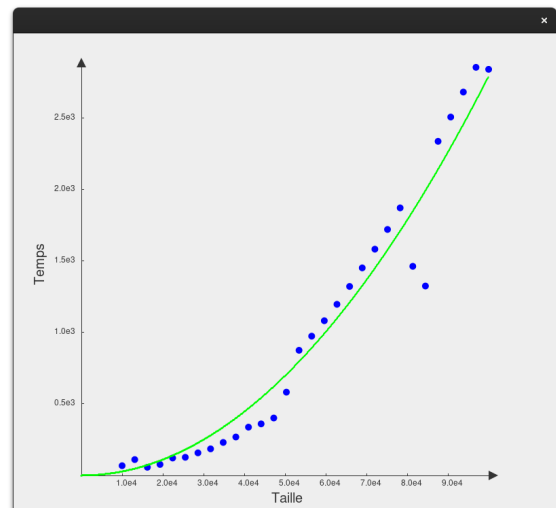
1 Analyse de tris

1.1 Tri par insertion

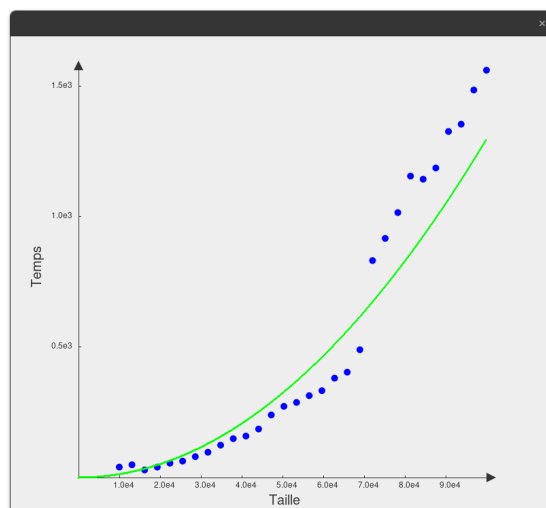
On étudie la complexité de l'algorithme de tri par insertion en fonction des différents cas possibles :



Meilleur cas



Pire cas



Cas général

On trouve des complexités différentes en fonction des différents cas. Dans le meilleur cas, c'est-à-dire quand le tableau est déjà trié, l'algorithme possède une

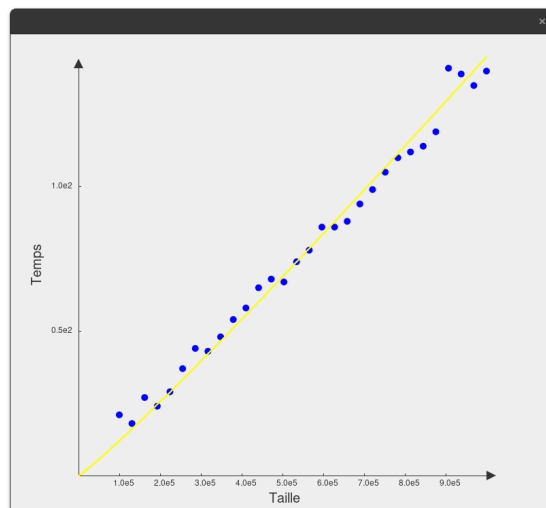
complexité linéaire. Dans le pire cas (tableau trié en sens inverse) et le cas général, l'algorithme passe en complexité quadratique. Le coefficient quadratique dans le cas général est égal à $1.3 * 10^{-7}$ dans notre cas.

Étude sur la population française

On regarde ensuite le temps estimé de l'algorithme avec un tableau de 67 millions d'éléments. On utilise le coefficient trouvé dans le cas général du tri par insertion. $t = 13 * 10^{-8} * 67 * 10^6 = 583570000$ secondes. Ce qui équivaut à 6 jours 18 heures 6 minutes 10 secondes.

1.2 Tri fusion

On étudie la complexité de l'algorithme de tri fusion uniquement dans le cas général :



Cas général

L'algorithme possède une complexité en $n \log(n)$. La courbe du temps d'exécution en fonction de la taille du tableau obtenue est similaire à une courbe linéaire. Le coefficient de la droite est environ égal à $1.05 * 10^{-5}$ dans notre cas.

Étude sur la population française

On étudie de nouveau le temps d'exécution de l'algorithme sur la population française. On utilise le coefficient de la droite trouvé dans le cas général pour estimer

le temps d'exécution. $t = 1.05 * 10^{-5} * 67 * 10^6 = 703$ secondes. Ce qui équivaut à 10 minutes 43 secondes.

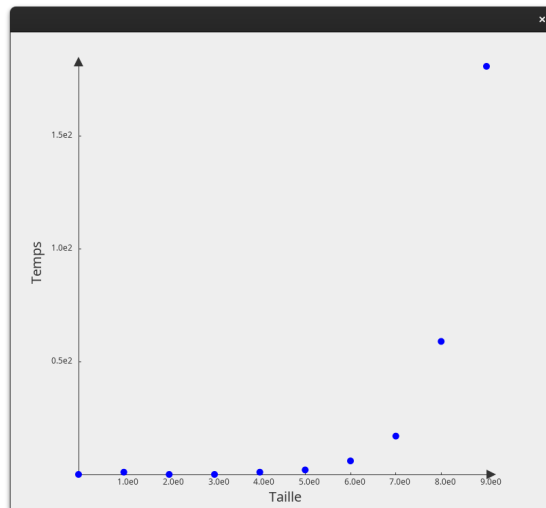
Conclusion

On peut en conclure que l'algorithme de tri fusion est meilleur que le tri par insertion dans le cas général. L'efficacité du tri fusion est surtout remarquable dans le cas de grands tableaux à trier où l'on peut passer d'un temps d'exécution d'une semaine à seulement quelques minutes.

2 Permutations

On s'intéresse ensuite aux permutations. La fonction fournie effectue des permutations sur une chaîne de caractères. En affichant successivement tous les arrangements possibles de caractères contenues dans cette chaîne. En supposant que la complexité est du même ordre de grandeur que le nombre d'affichages réalisés, il y existe, pour une chaîne de taille n , n factoriel combinaisons pour une chaîne de caractères. La complexité est donc $n!$.

On affiche l'évolution du temps de calcul en fonction de n , la taille de la chaîne de caractères :



Permutation

En se servant de la courbe obtenue on peut estimer le temps d'exécution pour $n = 26$. On trouve $2.1 * 10^{12}$ ans, ce qui est plus grand que l'âge de l'univers.