Rapport de comparaisons - Projet Algorithmique

Comparaison : Exécutable et rapport algorithmique
Nous avons eu à comparer notre projet de SAE d'algo et de C avec ceux de Jules Massonat et Antoine Voillot .

Sommaire:

Réalisé par Sadiya Niang et Killyan Batailler

I - Comparaison théorique :

Pour comparer nos deux algorithmes, nous allons en particulier nous intéresser au concept de complexité asymptotique pour les deux méthodes. En effet, Il y a là un problème difficile : la vitesse d'exécution d'un algorithme dépend de l'ordinateur (du processeur, du type et de la quantité de mémoire, de l'organisation et la taille du cache et éventuellement de la rapidité du disque). Ainsi, la comparaison asymptotique est ce qu'il y a de mieux pour contourner tous ces problèmes car avec cette dernière on ne donne pas la fonction du temps exacte, mais uniquement une approximation de celle-ci. Cette approximation est donnée sous la forme d'une fonction qui donne la forme générale de la fonction du temps.

1 - Algorithmes de Tri

Nous avons fait le choix d'un algorithme de tri appelé Quicksort tandis que l'autre groupe a choisi le tri fusion. Les principales différences entre les deux méthodes est le fait que dans le tri par fusion, le tableau doit être divisé en deux moitiés seulement (c'est-à-dire n / 2). Par contre, en tri rapide, il n'y a aucune contrainte de diviser la liste en éléments égaux.

Pire des cas:

Le cas le plus complexe du tri rapide est O (n2), car il nécessite beaucoup plus de comparaisons dans les pires conditions. mais il est possible d'éviter cela en choisissant le bon pivot. En revanche, le pire des cas du tri par fusion s'effectue lorsqu'on doit effectuer un nombre maximum de comparaisons.Ainsi la performance la plus défavorable est de O (n $\log_2$ n).

Complexité de l'espace:

L'algorithme de tri rapide utilise le même tableau lors de la division et du fusion des éléments, ainsi c'est un tri en place et donc aucun espace supplémentaire n'est nécessaire pour le tri. Tandis que le tri fusion n'est pas un tri en place, il nécessite une zone de travail telle que d'autre tableau élémentaire pour la fusion de deux sous tableaux ce qui nécessite de l'espace supplémentaire.

Conclusion:

Le tri fusion est plus efficace dans le pire des cas que le quicksort. Cependant, ce dernier est plus efficace en termes d'espace et les deux sont également efficaces dans le cas moyen. En revanche dans le cas général, le

quicksort est plus rapide et meilleur que le tri fusion sauf dans certains cas où il est inefficace comme lorsqu'il s'agit de comparer.

2 - Algorithmes de Recherche

Nous avons fait le choix d'une recherche dichotomique tout comme l'autre binôme.

Pire des cas:

Le cas le plus complexe de la recherche dichotomique se produit lorsque la valeur recherchée n'est pas dans le tableau.
Ainsi la performance la plus défavorable est de O(log2(n)).

Complexité de l'espace:

La complexité spatiale de cet algo est de o(1) dans le cas d'une implémentation itérative et de O(log n) dans le cas d'une implémentation récursive.

Conclusion:

La recherche dichotomique est l'algo de recherche la plus rapide et la plus efficace. Le seul défaut qu'il pourrait y avoir c'est la nécessité d'un tableau initialement trié pour son fonctionnement.

3 - Algorithmes de Filtre

Pour l'algorithme de filtre nous avons choisi un algo qui se base sur la recherche dichotomique tandis que l'autre binôme ont choisi une fonction du langage C appelée strstr.

Pire des cas:

Ci-dessus, nous avons présenté le pire des cas dans la recherche dichotomique. La fonction strstr elle, a pour complexité le nombre de données à vérifier (que l'on appellera n) multiplié par la longueur de la chaîne recherchée (que l'on nommera m). On aura alors pour complexité O(m.n).

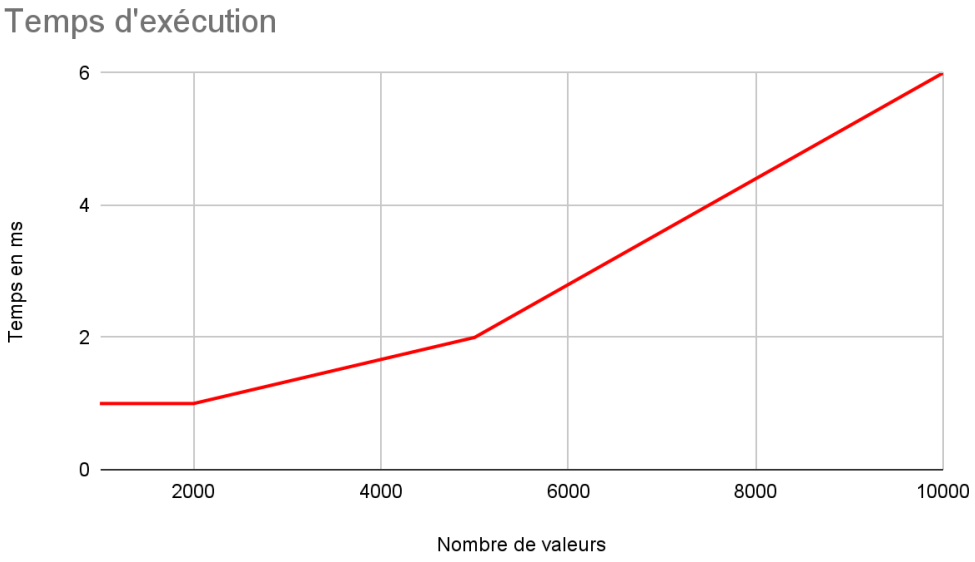
Conclusion:

La recherche dichotomique est l'algo de recherche la plus rapide et la plus efficace.Cependant la fonction stsrstr est plutôt pas mal car il permet la recherche d'une première occurrence d'une chaîne,ce qui est plutôt pratique pour le filtrage.Le défaut de la recherche dichotomique c'est la nécessité d'un tableau initialement trié tandis que la strstr n'en a pas besoin.

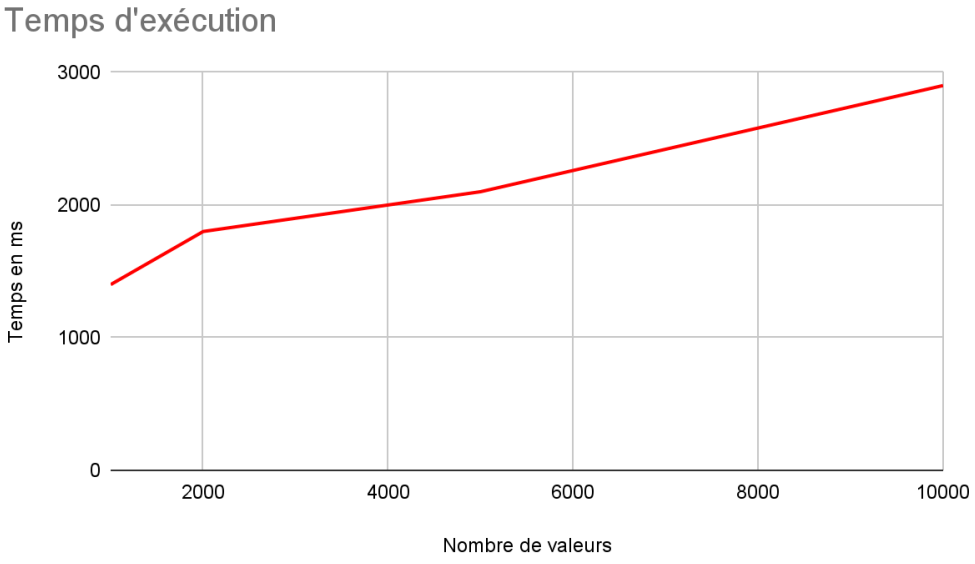
II - Comparaison pratique :

Étant donné que certaines fonctionnalités de notre exécutable ne fonctionne pas, nous ne pouvons faire de comparaison pratique. Cependant, nous avons pu faire un relevé des temps d'exécutions de l'exécutable de l'autre binôme.

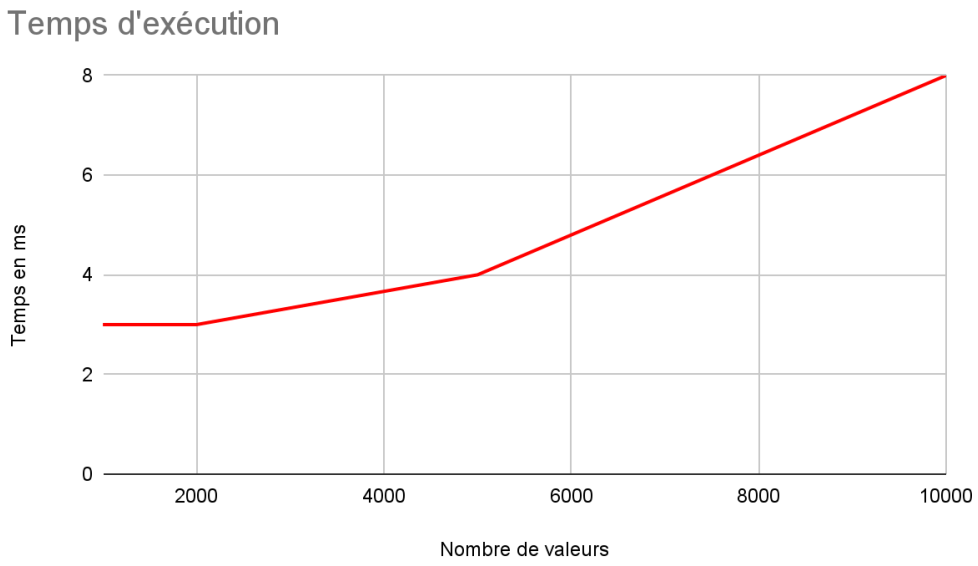
Temps d'exécution du tri :



Temps d'exécution de la recherche :



Temps d’exécution du filtre :



Le test ne va pas plus loin que 10 000 valeurs car l’exécutable crash au-delà.

III - Analyse des résultats :

Après les différentes comparaisons que nous avons pu faire, nous avons pu tirer les conclusions suivantes :

Choix du tri : Le Quicksort que nous avions choisi est plus adapté que le tri fusion de l'autre binôme car ce dernier demande plus d'espace et il est beaucoup plus long a exécuté si le nombre de valeurs est très élevé.

Choix de la recherche : Nos deux binômes ont opté pour une recherche dichotomique et nous pensons que c'est l'algorithme le plus performant et le plus adapté à la situation.

Choix du filtre : La fonction *strstr* choisie par leur binôme nous semble être le meilleur choix. Ce dernier à la particularité d'être efficace, rapide et d'afficher pas seulement la chaîne exacte qui est recherchée à l'aide d'une deuxième chaîne en paramètre du filtre mais plus particulièrement toutes les chaînes comportant la chaîne paramètre dans son champ.