AlgoStats

Le but de ce projet était de mettre en place une plateforme d'évaluation de puissance de tri sur laquelle il fallait tester divers algorithmes de tri :

I - AlgoStats 1 :

- *Le tri par insertion*
- *Le tri par sélection*
- *Le tri à bulle*

1. LE TRI PAR INSERTION :

L'algorithme du tri par insertion consiste à insérer un élément dans une liste déjà triée. En général, le tri par insertion est beaucoup plus lent que d'autres algorithmes mais il est cependant considéré comme le tri le plus efficace sur des entrées de petite taille.

2. LE TRI PAR SELECTION :

Ici, le but est de parcourir une liste non triée (dans notre cas, liste de nombres) afin de récupérer le plus grand élément (ou le plus petit) et de le placer en fin de liste (ou au début). L'action est répétée jusqu'à ce que la liste soit triée.

3. LE TRI A BULLE :

Le principe de ce tri, est de parcourir la liste deux à deux (cad, on compare chaque case avec la suivante) et on permute les valeurs si cela est nécessaire. On répète ainsi ces étapes jusqu'à ce que la liste soit triée.

II - AlgoStats 2 :

1. LE TRI SHELL:

Le tri de Shell trie chaque liste d'éléments séparés de n positions chacun avec le tri par insertion. L'algorithme effectue plusieurs fois cette opération en diminuant n jusqu'à n=1 ce qui équivaut à trier tous les éléments ensemble.

2. LE TRI FUSION:

Il s'agit à nouveau d'un tri suivant le paradigme diviser pour régner. Le principe du tri fusion (ou tri par interclassement) en est le suivant :

Première étape: On divise en deux moitiés la liste à trier (en prenant par exemple, un élément sur deux pour chacune des listes).
Deuxième étape: On trie chacune d'entre elles.
Troisième étape: On fusionne les deux moitiés obtenues pour reconstituer la liste triée.

3. LE TRI RAPIDE:

Le tri rapide - aussi appelé "tri de Hoare" (du nom de son inventeur Tony Hoare) ou "tri par segmentation" ou "tri des bijoutiers" ou, en anglais "quicksort" - est certainement l'algorithme de tri interne le plus efficace.

Le principe de ce tri est d'ordonner le vecteur T.(0)..T.(n) en cherchant dans celui-ci une clé pivot autour de laquelle réorganiser ses éléments. Il est souhaitable que le pivot soit aussi proche que possible de la clé relative à l'enregistrement central du vecteur, afin qu'il y ait à peu près autant d'éléments le précédant que le suivant, soit environ la moitié des éléments du tableau.

4. LE TRI A PEIGNE:

L'idée de base du tri à peigne est de comparer un élément avec un autre élément plus lointain, espacé de celui-ci d'un certain intervalle. Au départ, cet intervalle est relativement grand, puis on le réduit progressivement à chaque passe de l'algorithme jusqu'à ce qui ne soit plus que de 1 en fin de traitement. Le tri à bulles peut être considéré comme une variante du comb sort dans lequel l'intervalle est toujours de 1.

Nombre d'itération:

Algostats 1:

Lorsque l'on compare le nombre d'itération pour les 3 tris et qu'on les classe ensuite dans un ordre décroissant, on obtient :

Tri à bubble > Tri select = Tri insert

Algostats 2:

Lorsque l'on compare le nombre d'itération pour les 4 tris et qu'on les classe ensuite dans un ordre décroissant :

Tri à peigne > Tri shell > Tri rapide > Tri fusion

Temps de traitement :

Algostats 1:

En ce qui concerne le temps d'exécution, le tri par sélection et le tri à bulle ont un temps d'exécution plus faible que celui du tri par insertion qui est légèrement supérieur.

Algostats 2:

En ce qui concerne le temps d'exécution, si nous classons dans l'ordre décroissant :

Tri fusion > Tri à peigne > tri rapide > Tri shell