TD/TP 4 : Algorithmes de Tri

IUT CAEN – Département Informatique

Module M1103

Exercice 1

1. Écrire une fonction qui teste si un tableau de n entiers est trié ou non.
2. Écrire une fonction qui compte le nombre de fois où des nombres successifs dans le tableau ne sont pas correctement ordonnés.

Exercice 2 (Tri par sélection)

Le tri par sélection consiste en la recherche du plus petit élément du tableau qui va être placé à sa position définitive c'est-à-dire en première position. L'algorithme continue en recherchant le plus petit élément du tableau privé de son premier élément, qui va être placé en seconde position. Et ainsi de suite pour tous les éléments du tableau.

1. Trier le tableau ci-dessous selon l'algorithme de tri par sélection en donnant toutes les étapes intermédiaires réalisées.

1	4	3	5	2	0	6	8	7

2. Écrire la fonction `tri_selection(int T[], int n)` qui prend en paramètre un tableau et sa taille, le trie selon l'algorithme par sélection.
3. Donner la complexité de cet algorithme.

Exercice 3 (Tri par insertion)

Le principe du tri par insertion peut être résumé comme suit: on tri d'abord les deux premiers éléments, puis on insère le 3ème à sa place pour faire une liste triée de 3 éléments, etc. De manière générale on suppose les $(i-1)$ premiers éléments triés. On prend le $i$ème élément, et on essaie de le mettre à sa place dans les $(i-1)$ éléments déjà triés. Et on continue jusqu'à $i = n$.

1. Trier le tableau ci-dessous selon l'algorithme de tri par insertion en donnant toutes les étapes intermédiaires réalisées.

1	4	3	5	2	0	6	8	7

2. Écrire la fonction `tri_insertion(int T[], int n)` qui prend en paramètre un tableau et sa taille, le trie selon l'algorithme du tri par insertion.
3. Donner la complexité de cet algorithme.

Exercice 4 (Tri à bulles)

L'algorithme de tri à bulles consiste à parcourir le tableau à trier et à comparer chaque élément avec son voisin de droite. Lorsque deux éléments consécutifs ne sont pas dans l'ordre croissant, ils sont échangés. Après un parcours complet du tableau, on recommence l'opération. Lorsqu'un parcours n'a donné lieu à aucun échange, cela signifie que le tableau est trié : l'algorithme s'arrête.

1. Trier le tableau suivant selon l'algorithme du tri à bulles en donnant toutes les étapes intermédiaires réalisées.

1	4	3	5	2	0	6	8	7

2. Écrire la fonction `tri_bulles(int T[], int n)` qui prend en paramètre un tableau et sa taille, le trie selon l'algorithme du tri à bulles.

3. Donner la complexité de cet algorithme.

Exercice 5 (À faire en TP)

On va chercher à mesurer les performances sur un même jeu de valeurs des algorithmes de tri suivants : Tri par sélection, Tri par insertion et Tri à bulles. Ecrivez un programme qui :

1. initialise trois tableaux identiques de N = 1000 valeurs à l'aide de la fonction `rand()`;

2. trie chacun des tableaux avec l'un des algorithmes implémentés ci-dessous;

3. mesure le temps d'exécution de chacun de ces algorithmes à l'aide de la fonction `clock()`;

   Cette fonction renvoie le temps processeur utilisé depuis le début du programme courant. Donc, pour mesurer que le temps d'exécution d'un des algorithmes, il faut appeler cette fonction juste avant et juste après celui-ci (par exemple: `clock_t debut_algo1 = clock();` etc.), puis procéder par différences;

4. réalise ces mêmes mesures en faisant croître le nombre d'éléments des tableaux.