Lycée Louis-Le-Grand, Paris 2015/2016
MPSI 4 – Informatique commune
G. Dewaele, A. Troesch

TP n° 9 : Quelques algorithmes de tri

Pour tout ce TP, on se créera trois tableaux, l'un de longueur 10 (pour les tests de validité), les deux autres de longueur 1000 et 10000 (pour les tests d'efficacité), remplis de nombres aléatoires compris entre 0 *et* 10000*, et on déterminera le temps d'exécution des algorithmes de tri sur ces 2 tableaux, de sorte à comparer les qualités des différents algorithmes.*

Exercice 1 – (Tri par sélection) Le tri par selection consiste à rechercher le plus petit élément du tableau initial, et à le mettre en position initiale (par exemple par un échange), puis à recommencer, avec la recherche du plus petit terme parmi ceux restants etc.
Implémenter le tri par selection, et évaluer rapidement sa complexité.

Exercice 2 – (Tri par insertion)
Le tri par insertion consiste à trier les éléments d'une tableau au fur et à mesure de la lecture de ce tableau. Ainsi, supposant qu'après traitement des k premiers éléments du tableau, on les a reclassés dans l'ordre, le terme suivant est ensuite inséré à sa place parmi les précédents. Cette insertion peut se faire par échanges successifs avec les termes supérieurs déjà positionnés, jusqu'à ce qu'il se trouve à sa place. Ainsi, la recherche de la position et l'insertion se font de concert.
Implémenter l'algorithme de tri par insertion, en essayant de faire le tri dans le tableau initial. Évaluer rapidement sa complexité.

Exercice 3 – (Tri rapide, quick sort)
Le tri rapide est un tri récursif dont l'idée est la suivante : on commence par choisir un élément quelconque du tableau (appelé pivot), par exemple le premier (ou un élément choisi aléatoirement, surtout si on est amené à l'utiliser plus fréquemment sur des tableaux déjà presque triés, dans un sens ou l'autre). On parcourt une fois le tableau, de sorte à séparer les éléments inférieurs au pivot de ceux qui lui sont supérieurs. Cela fournit un tableau se découpant en trois parties : un premier sous-tableau des éléments inférieurs ou égaux au pivot, un deuxième sous-tableau contenant uniquement le pivot et un troisième sous-tableau contenant tous les éléments strictement supérieurs au pivot. On recommence la même opération sur le premier et le troisième de ces sous-tableaux, jusqu'à ce qu'ils soient de longueur 1.

1. Écrire une fonction prenant en argument une liste, et deux indices i et j, choisissant aléatoirement un pivot entre les indices i et j, et reclassant au sein du tableau initial les éléments de la façon décrite ci-dessus entre les indices i et j.

   On s'attachera à ne pas introduire de nouveau tableau pour ce faire. On pourra par exemple commencer par positionner le pivot en position j, puis construire de façon contiguë les deux autres tableaux à partir de la position i : à chaque nouvel élément considéré, s'il est supérieur au pivot, on le laisse en place (il prend alors place à la fin du second sous-tableau), sinon, on l'échange avec le premier terme du second sous-tableau (il prend alors place à la fin du premier sous-tableau, et l'ex-premier terme du second sous-tableau prend sa place à la fin du second sous-tableau. Pour cela, il faudra deux variables pour repérer la progression dans le tableau d'une part et le nombre d'éléments mis dans le premier sous-tableau d'autre part.

   Pour terminer, on remet le pivot à sa place, par un dernier échange.

2. Écrire une fonction récursive prenant en paramètres une liste T, et deux indices i et j, et triant les éléments de la liste entre les indices i et j inclus. Écrire une fonction `triRapide` prenant en paramètre une liste et retournant le tableau trié, en utilisant l'algorithme décrit ci-dessus.

3. Le tri par insertion est plutôt meilleur que le tri rapide sur les petits tableaux. On choisit un seuil n_0 à partir duquel on bascule sur le tri par insertion. Écrire un algorithme de tri prenant en paramètres une liste T et un seuil n_0, et retournant le tableau trié en basculant au tri par insertion pour le tri des sous-listes de taille inférieure ou égale à n_0.
4. Tracer le graphe du temps de réponse pour le tri d'un tableau de taille 10000 fixé une fois pour toutes, en fonction du seuil n_0. On commencera par faire varier n_0 entre 1 et 500, avec un pas de 50, puis on affinera en réduisant cet intervalle et le pas, de sorte à déterminer la valeur optimale de n_0 pour le tableau donné.
5. Une autre façon de procéder est d'utiliser le fait que le tri par insertion est rapide sur les tableaux presque triés. Une variante du basculement précédent consiste donc à abandonner le tri rapide en dessous du seuil n_0, en laissant le sous-tableau tel quel dans un premier temps. Ainsi, la fonction retourne un tableau presque trié dans le sens où tous les termes seront à une distance au plus n_0 de leur position finale (le tableau est trié par tranches d'au plus n_0). On lance alors un tri par insertion sur le tableau obtenu. Si la valeur de n_0 est vouée à ne pas être grande par rapport à n, on a intérêt à faire la recherche de la position d'insertion en partant du haut (et en effectuant les échanges successifs au fur et à mesure).

   Implémenter cette variante, et rechercher graphiquement comme précédemment la valeur optimale de n_0 pour un tableau de taille 10000. Comparer les temps de réponse avec ceux obtenus par la méthode précédente. Testez sur des tableaux de taille plus grande.

Exercice 4 – (Tri fusion)

Le tri fusion est un algorithme de tri basé sur le principe suivant :

- On coupe la liste en 2, en son milieu (à 1 près si la longueur est impaire).
- On trie récursivement les 2 moitiés en lançant l'algorithme sur chacune des deux moitiés (ce qui impose de passer en paramètre des indices de début et de fin de sous-tableau à traiter)
- On fusionne les deux sous-tableaux triés, en les parcourant simultanément, en exploitant l'idée suivante : si on a à fusionner dans l'ordre deux paquets de cartes triés, on dispose les deux paquets, cartes faibles au dessus et visibles, on on prend les cartes sur l'un paquet ou l'autre, en prenant à chaque fois la plus petite des 2 cartes visibles.

  Cette fusion devra se faire dans le tableau initial, mais pourra nécessiter de faire des copies des deux sous-tableaux initiaux.

1. Implémenter l'algorithme du tri fusion, et comparer sa rapidité à celle des autres méthodes de tri.
2. Tester l'effet d'un basculement à un tri par insertion en dessous d'un certain seuil n_0.

Il existe bien d'autres algorithmes de tri, plus ou moins efficaces, et plus ou moins adaptés à certaines situations. Ainsi, avec des hypothèses supplémentaires, on peut descendre en dessous du seuil $n\ln(n)$ pour la complexité. Par exemple, si on sait que les valeurs à trier sont toutes des valeurs contenues dans un ensemble fini connu, il n'est pas dur de trouver un algorithme de tri linéaire (tri par dénombrement). Comment feriez-vous ?