Université de Nice – Sophia Antipolis — Deug MASS MIAS SM 1re année
UEF1 : Informatique & Programmation — 2000–2001

TP N° 12

Premier tri

Bonne année à vous tous !

Ça va mieux en le révisant ... Dès que l'on manipule un grand nombre de données, il faut savoir les ranger astucieusement pour pouvoir les retrouver rapidement. Le *tri* est un rangement simple et naturel. Un tri se fonde sur un *ordre* (celui des nombres, l'alphabétique, celui du dictionnaire...) : ne dit-on pas quelquefois *ordonner* à la place de ranger, dans la vie courante ?
Si des données sont conservées dans un tableau, il est utile de savoir trier ce tableau et de savoir le maintenir trié lors de l'ajout ou de la suppression d'un élément.
On vous a présenté en cours le *tri par sélection*. Il consiste à *sélectionner* la plus petite valeur d'un tableau, à la déplacer dans la première case, et à recommencer ce procédé pour le tableau privé de sa première case, jusqu'à épuisement des cases. Cet algorithme est un tri *sur place* car il ne nécessite pas de dupliquer tout ou partie du tableau pour le trier ; il a une *complexité quadratique* : cela signifie que son temps d'exécution est proportionnel au carré du nombre de cases du tableau traité.
Il existe de très très nombreux algorithmes de tri. Grosso modo, ils s'échelonnent des moins performants aux plus performants, des plus naturels aux plus sophistiqués. Le tri par sélection n'est d'ailleurs pas des plus rapides...
Dans les deux derniers cours, on vous a parlé d'algorithmes basés sur le principe de *dichotomie* : cela consiste à couper successivement en deux puis en deux puis en deux... un ensemble de données en vue de réaliser une tâche. Ce principe de dichotomie allié à la notion de tableau trié débouche sur des algorithmes performants (e.g. recherche ou insertion d'un élément dans un tableau trié en temps logarithmique [1]

1 Dichotomie et tableau trié

Exercice 1) Le but de cet exercice est de maîtriser l'algorithme d'insertion dichotomique d'un élément dans un tableau trié.

1. Déclarez un tableau de 40 entiers en ayant soin de prendre une variable `iLibre` destinée à contenir le nombre de cases effectivement utilisées.
2. Remplissez ce tableau avec 20 entiers aléatoires : on veillera à ce que le premier soit dans l'intervalle $[0, 9]$, le second dans l'intervalle $[10, 19]$...
   Faites afficher la partie utilisée de ce tableau sur une seule ligne.
3. Ce tableau est-il trié ? Implémentez l'algorithme d'*insertion dichotomique dans un tableau trié* pour insérer dans ce tableau et à sa place un entier choisi par l'utilisateur.
4. Auriez-vous une idée pour trouver où insérer l'entier donné par l'utilisateur de façon plus efficace ? (on ne vous demande pas de l'implémenter).
5. Imaginez que l'utilisateur ait inséré une dizaine de valeurs supplémentaires dans le tableau, votre idée reste-t-elle applicable ?

1. Il faut savoir que le nombre de fois que l'on doit diviser (au sens de la division entière) un entier $n > 1$ par 2 pour tomber sur 1 est égal à la partie entière de $\log_2(n)$. D'où l'apparition du logarithme dans le calcul des performances.

2 Tri par sélection

Exercice 2) Ecrivez un programme de façon à créer un tableau de 10 employés. Les caractéristiques (identification & salaire) de chaque employé seront à demander à l'utilisateur.
On décide de trier ces personnes par salaire décroissant. Adaptez le tri par sélection vu en cours à ce problème particulier.

3 Approfondissement

Exercice 3) Il paraît utile et naturel de savoir aussi trier ce tableau d'employés par ordre alphabétique. Ecrivez une variante au tri par sélection précédent pour réaliser cette tâche.
Le cas échéant, vous consulterez la documention en ligne de la classe `String`.