Introduction à l'informatique

Exercices — Série 8

Exercice 1

Le *tri à bulles (bubble sort)* est un algorithme de tri d'un vecteur basé sur le principe suivant. On parcourt le vecteur du début vers la fin, en comparant à chaque étape deux éléments consécutifs. (En d'autres termes, on commence par comparer les deux premiers éléments, puis le deuxième avec le troisième, puis le troisième avec le quatrième, et ainsi de suite.) Lorsqu'une comparaison révèle que les deux éléments concernés ne sont pas dans l'ordre souhaité, on les permute. À l'issue de cette procédure, le dernier élément du vecteur a atteint sa place définitive : Il s'agit de l'élément le plus grand ou le plus petit, selon que le tri soit respectivement par ordre croissant ou décroissant. Ensuite, on répète cette procédure de façon à déterminer l'avant-dernier élément du vecteur trié, puis l'élément précédent, et ainsi de suite jusqu'à arriver à un vecteur entièrement trié.

1. Décrire cet algorithme en pseudo-code.
2. Déterminer la complexité en temps de cet algorithme, en fonction de la taille du vecteur à trier.
3. Écrire une fonction C implémentant cet algorithme, dans le cas d'un tableau d'entiers à trier par ordre croissant.

Exercice 2

Le *tri par sélection (selection sort)* est un autre algorithme de tri d'un vecteur, fonctionnant de la façon suivante. On balaye le vecteur de façon à y repérer le plus petit élément (dans le cas d'un tri par ordre croissant) ou le plus grand (pour un ordre décroissant). On permute ensuite cet élément avec le premier élément du vecteur, qui prend alors sa valeur finale. On effectue ensuite la même opération en commençant le balayage à partir du deuxième élément du vecteur, puis du troisième, et ainsi de suite.

1. Décrire cet algorithme en pseudo-code.
2. Déterminer la complexité en temps de cet algorithme, en fonction de la taille du vecteur à trier.
3. Écrire une fonction C implémentant cet algorithme, dans le cas d'un tableau d'entiers à trier par ordre croissant.

Exercice 3

En adaptant une des fonctions obtenues aux deux exercices précédents (au choix), écrire un programme C capable de lire un ensemble de chaînes de caractères entrées au clavier, et de les trier par ordre lexicographique (c'est-à-dire, l'ordre du dictionnaire).

Notes :

— La première étape consiste à écrire une fonction C capable de comparer deux chaînes de caractères données par ordre lexicographique.

— Pour lire une chaîne au clavier, on peut utiliser la fonction `fgets()` de la bibliothèque standard, comme dans l'exemple ci-dessous :

```
#include <stdio.h>

#define TAILLE_MAX  100

  ...
  char chaine[TAILLE_MAX];

  fgets(chaine, TAILLE_MAX, stdin);
  ...
```

Attention, si la saisie se termine par un retour chariot, alors le caractère correspondant (`'\n'`) est ajouté à la chaîne produite par `fgets()`.

Exercice 4

Le *tri par comptage (counting sort)* est un algorithme de tri d'un vecteur utilisable lorsque les éléments de ce vecteur ne prennent qu'un petit nombre de valeurs possibles. Son principe consiste à parcourir le vecteur une seule fois, en comptant le nombre d'occurrences de chaque valeur. Par exemple, à l'issue du parcours du vecteur $[1, 0, 1, 2, 1, 0]$, on obtient que ce vecteur contient 2 occurrences de la valeur 0, 3 occurrences de 1 et 1 occurrence de 2. À partir de ce résultat, il est ensuite facile de générer le vecteur trié en produisant le nombre de copies attendues de chaque occurrence, c'est-à-dire $[0, 0, 1, 1, 1, 2]$ pour notre exemple.

1. Décrire cet algorithme en pseudo-code.
2. Déterminer la complexité en temps et en espace de cet algorithme, en fonction de la taille du vecteur à trier et du nombre de valeurs pouvant être prises par les éléments du vecteur.
3. Écrire une fonction C appliquant cet algorithme au tri par ordre croissant des caractères contenus dans une chaîne donnée. (Par exemple, le tri de la chaîne `"algorithmique"` doit produire `"aeghiilmoqrtu"`.)

Exercice 5

Le *tri rapide (quicksort)* est un algorithme de tri d'un vecteur basé sur l'approche diviser pour régner. Son principe consiste à d'abord choisir un *pivot*, qui est un élément quelconque du vecteur. (Une stratégie simple consiste à systématiquement choisir le premier élément du vecteur comme pivot, mais d'autres choix sont possibles.) Ensuite, on réordonne les éléments du vecteur de façon à avoir un préfixe contenant tous les éléments de valeur inférieure à celle du pivot, puis le pivot, puis un suffixe contenant les éléments restants. Enfin, on applique récursivement le même algorithme de tri aux sous-vecteurs correspondant au préfixe et au suffixe.

1. Décrire cet algorithme en pseudo-code.
2. Déterminer la complexité en temps de cet algorithme, en fonction de la taille du vecteur à trier.

   Indice : Le pire cas du point de vue du temps d'exécution correspond à la plus grande profondeur possible de récursion.
3. Écrire une fonction C implémentant cet algorithme, dans le cas d'un tableau d'entiers à trier par ordre croissant.