TP N°8

Les algorithmes de tri et de recherche

Exercice 1:

L'algorithme de recherche séquentielle (ou linéaire) consiste à examiner la table éléments par éléments et voir si **info** appartient ou non à la table **T**. Si le résultat est positif (**info ∈ T**) alors cet algorithme retourne l'indice de la première occurrence de l'**info**, sinon il retourne -1.

Réaliser en C la fonction **int RechercheSequentielle (int T [] , int n, int info)**

Exercice 2:

Soit le programme suivant :

```
# include <stdio.h>
# define n 5
float a [n] ;
float info ;
unsigned P (unsigned x) {
        return a[x] = = info ;
}
void main(void){
        unsigned x ;
        unsigned y ;
        unsigned p, q ;
        unsigned i ;
        for (i=0; i<n; i++){
                printf(" donner l'element num %u \n", i) ;
                scanf("%f", &a[i]) ;
        }
        printf(" donner réel  \n") ;
        scanf("%f", &info) ;

        p=0 ;
        q=n ;
        x=p ;
        y=q ;
        while(x !=y) {
                if (P(x))
                        y=x ;
                else
                        x++ ;
        }
        printf("%u\n", x !=q) ;
}
```

Questions :

a) Expliquer l'organisation générale du programme cité ci-dessus.

b) Identifier le problème censé être résolu par le programme précédent et comparer la solution proposée par rapport aux algorithmes résolvant le même type du problème.

Exercice 3:

La technique de tri par sélection est la technique la plus simple, elle consiste à :

- chercher l'indice du plus petit élément du tableau T[0..n-1] et permuter l'élément correspondant avec l'élément d'indice 0
- chercher l'indice du plus petit élément du tableau T[1..n-1] et permuter l'élément correspondant avec l'élément d'indice 1
- …
- chercher l'indice du plus petit élément du tableau T[n-2..n-1] et permuter l'élément correspondant avec l'élément d'indice (n-2).

1) Écrire la fonction **void TriSelection (int T [] , int n)** qui permet le tri des n premiers éléments du tableau d'entier T.
2) Écrire un programme principal (la fonction main) qui lit la dimension n d'un tableau T de type int (dimension maximale: 50 composantes), remplit ce tableau par des valeurs entrées au clavier et affiche le tableau. Ensuite, trier ce tableau en utilisant la fonction TriSelection puis réafficher ce tableau.

Exercice 4:

Cette méthode consiste à prendre les éléments du tableau un par un et insérer chacun dans sa bonne place de façon que les éléments traités forment une sous-liste triée.
Pour ce faire, on procède de la façon suivante :
- comparer et permuter si nécessaire T[0] et T[1] de façon à placer le plus petit dans la case d'indice 0
- comparer et permuter si nécessaire l'élément T[2] avec ceux qui le précèdent dans l'ordre (T[1] puis T[0]) afin de former une sous-liste triée T[0..2]
- …
- comparer et permuter si nécessaire l'élément T[n-1] avec ceux qui le précèdent dans l'ordre (T[n-2], T[n-3], …) afin d'obtenir un tableau trié.

1) Écrire la fonction **void TriInsertion (int T [] , int n)** qui permet le tri des n premiers éléments du tableau d'entier T.
2) Écrire un programme principal (la fonction main) qui lit la dimension n d'un tableau T de type int (dimension maximale: 50 composantes), remplit ce tableau par des valeurs entrées au clavier et affiche le tableau. Ensuite, trier ce tableau en utilisant la fonction TriInsertion puis réafficher ce tableau.

Exercice 5:
Comparer les deux algorithmes tri par sélection et tri par insertion.

Exercice 6 :
Réaliser en C un algorithme qui permet de calculer le nombre de sous-séquences et la plus longue sous-séquences strictement croissante dans un tableau de n entiers.

Exercice 7 :

On donne un tableau appelé redondant contenant des entiers redondants et dans un ordre quelconque.
Présenter d'une façon informelle et réaliser en C un algorithme permettant de compter la fréquence de chaque élément figurant dans le tableau redondant dans une table.

Exercice 8 :

On possède un tableau de nombres entiers, classés par valeur croissante, chaque entier pouvant être répété plusieurs fois : 3 5 5 5 7 9 9 9 9 10 10 12 12 12 13

On appelle plateau la suite de tous les entiers consécutifs de même valeur : dans l'exemple ci-dessous le plateau de valeur 5 comporte 3 nombre, celui de valeur 10 comporte 2 etc. On appelle la longueur d'un plateau la valeur commune à tous ses termes. Un tableau a de n termes étant donné, trouver la longueur et la valeur de son plus long plateau.

Exercice 9:

On donne une table contenant des couples (x, y) avec x est une information de type réel et y sa fréquence (information de type entier non signé). Une telle table est triée par ordre croissant sur le champ x. De plus, on définit les opérations suivantes :

1. cherche : elle permet de voir si une information (info) de type réel appartient ou non à la table. En cas d'issue positive, cherche rend également la position de cet élément. Programmer en C cette opération.
2. incrémenter : elle permet d'incrémenter la fréquence relative à x déjà localisée. Programmer en C cette opération.
3. ajouter : elle permet d'ajouter x avec une fréquence nulle à la table. Elle laisse la table triée par ordre croissant sur le champ x.
   Donner la signature de cette opération et d'une façon informelle l'algorithme adéquat à la programmation de cette opération.

Exercice 10:

Soit T un tableau de n entiers. La méthode de tri à bulles nécessite deux étapes :

- Parcourir les éléments du tableau de 0 à (n–2) ; si l'élément i est supérieur à l'élément (i+1), alors on les permute.
- Le programme s'arrête lorsqu'aucune permutation n'est réalisable après un parcours complet du tableau.

1) Écrire la fonction **void TriBulles (int T [] , int n)** qui permet le tri des n premiers éléments du tableau d'entier T.

2) Écrire un programme principal (la fonction main) qui lit la dimension N d'un tableau T de type int (dimension maximale: 50 composantes), remplit ce tableau par des valeurs entrées au clavier et affiche le tableau. Ensuite, trier ce tableau en utilisant la fonction TriBulles puis réafficher ce tableau.

Exercice 11:

La technique de recherche dichotomique n'est applicable que si le tableau est déjà trié (par exemple dans l'ordre croissant). Le but de recherche dichotomique est de diviser l'intervalle de recherche par 2 à chaque itération. Pour cela, on procède de la façon suivante :

Soient premier et dernier les extrémités gauche et droite de l'intervalle dans lequel on cherche la valeur x, on calcule M, l'indice de l'élément médian :

M = (premier + dernier) div 2

Il y a 3 cas possibles :

- x = T[M] : l'élément de valeur x est trouvé, la recherche est terminée
- x < T[M] : l'élément x, s'il existe, se trouve dans l'intervalle [premier..M-1]
- x > T[M] : l'élément x, s'il existe, se trouve dans l'intervalle [M+1..dernier]

La recherche dichotomique consiste à itérer ce processus jusqu'à ce que l'on trouve x ou que l'intervalle de recherche soit vide.