M2 CCI – EP1 Algorithmique

TP4 (2h) – 20 oct. 2017

But de la séance : Rédaction / Codage d'algorithmes sur des tableaux 1D

Retour aux tableaux 1D. Quatre algorithmes au programme, 3 à comprendre et à traduire, le quatrième à concevoir et à écrire.

Les exercices seront traités dans l'ordre, l'important n'est pas le nombre réalisé à la fin de la séance, mais votre compréhension et votre capacité à le refaire.

Table des matières

Exercice 1 – Traduction d'algorithme (I)

On donne l'algorithme suivant :

Algorithme Algo1(T, v)
Entrée :
    T : tableau d'entiers
    v : entier
Sortie :
    p : entier
Début
    pos ← -1
    rang ← longueur(T)
    trouve ← faux
    tant que (rang>=1 et non(trouve)) faire
        si (T[rang] = v) alors
            trouve ← vrai
            pos ← rang
        fin
        rang ← rang - 1
    fin
    renvoyer (pos)
Fin

Question 1
Quel traitement est réalisé par cet algorithme ?

Question 2
Traduire cet algorithme en Java et testez le.

Exercice 2 – Traduction d'algorithme (II)

On donne l'algorithme suivant :

Algorithme Algo2(T)
Entrée :
    T : tableau d'entiers
Sortie :
    W : tableau d'entiers
Début
    pour k allant de 1 à longueur(T) faire
        W[k] ← T[longueur(T)-k+1]
    fin
    renvoyer (W)
Fin

Question 1
Quel traitement est réalisé par cet algorithme ?

Question 2
Traduire cet algorithme en Java et testez le.

Exercice 3 – Traduction d'algorithme (III)

On donne l'algorithme suivant :

Algorithme Algo3(T1, T2)
Entrée :
T1 : tableau d'entiers, trié ordre croissant
T2 : tableau d'entiers, trié ordre croissant
Sortie :
W : tableau d'entiers, trié
Début

```
LT1 ← longueur(T1) ,      LT2 ← longueur(T2)
rang1 ← 1,                rang2 ← 1
rang ← 1
tant que ( rang1 ≤ LT1 ou rang2 ≤ LT2 ) faire
        si (rang1 > LT1) alors
                tant que ( rang2 ≤ LT2 ) faire
                        W[rang] ← T2[rang2]
                        rang ← rang + 1
                        rang2 ← rang2 + 1
                fin
        sinon
                si (rang2 > LT2) alors
                        tant que ( rang1 ≤ LT1 ) faire
                                W[rang] ← T1[rang1]
                                rang ← rang + 1
                                rang1 ← rang1 + 1
                        fin
                sinon
                        si ( T1[rang1] < T2[rang2] ) alors
                                W[rang] ← T1[rang1]
                                rang ← rang + 1
                                rang1 ← rang1 + 1
                        sinon
                                W[rang] ← T2[rang2]
                                rang ← rang + 1
                                rang2 ← rang2 + 1
                        fin
                fin
        fin
fin
```

Question 1

Quel traitement est réalisé par cet algorithme ?

Question 2

Traduire cet algorithme en Java et testez le.

Exercice 4 – Ecriture d'algorithme

On souhaite réaliser un algorithme dont le comportement serait le suivant :

- il reçoit un tableau d'entiers, T, non vide
- il choisit, au hasard, une valeur de T, notée p
- il crée un tableau W de la même taille que T, rempli de 0.
- successivement, il lit chaque élément de T et
  - si la valeur lue est inférieure à p, il la recopie dans W, en partant du début (mais sans écraser celles qui auraient déjà pu être écrites) et il note la dernière position d'écriture d
  - si la valeur lue est supérieure à p, il la recopie dans W, en partant de la fin (mais sans écraser celles qui auraient déjà pu être écrites) et il note la dernière position d'écriture dans f
- il remplit l'intervalle W[d+1..f-1] avec p

Exemple : T ← {3,6,9,3,5,6,1,6,8,3,4} . On calcule longueur(T), ce qui donne 11, et on choisit un nombre entre 1 et 11 (par ex. 8). Le 8^{e} élément est 6. On initialise W ← {0,0,0,0,0,0,0,0,0,0,0}. On lit successivement tous les éléments de T et on décide de leur affectation :

- T[1] vaut 3, 3 est inférieur à 6 donc il va au début de W ← {3,0,0,0,0,0,0,0,0,0,0} et d = 1
- T[2] vaut 6, aucune règle s'applique, il est ignoré
- T[3] vaut 9, 9 est supérieur à 6 donc il va à la fin de W ← {3,0,0,0,0,0,0,0,0,0,9} et f = 11
- T[4] vaut 3, 3 est inférieur à 6 donc il va au début de W ← {3,3,0,0,0,0,0,0,0,0,9} et d = 2
- T[5] vaut 5, 5 est inférieur à 6 donc il va au début de W ← {3,3,5,0,0,0,0,0,0,0,9} et d = 3
- T[6] vaut 6, aucune règle s'applique, il est ignoré
- T[7] vaut 1, 1 est inférieur à 6 donc il va au début de W ← {3,3,5,1,0,0,0,0,0,0,9} et d = 4
- T[9] vaut 6, aucune règle s'applique, il est ignoré
- T[9] vaut 8, 8 est supérieur à 6 donc il va à la fin de W ← {3,3,5,1,0,0,0,0,0,8,9} et f = 10
- T[10] vaut 3, 3 est inférieur à 6 donc il va au début de W ← {3,3,5,1,3,0,0,0,0,8,9} et d = 5
- T[11] vaut 4, 4 est inférieur à 6 donc il va au début de W ← {3,3,5,1,3,4,0,0,0,8,9} et d = 6

On remplit ensuite W[7..9] avec la valeur de p (soit 6 ici). Donc W ← {3,3,5,1,3,4,6,6,6,8,9} , et on a d=6 et f=10

Question 1

Ecrire cet algorithme et vérifiez son fonctionnement en calculant la trace sur l'exemple donné

Question 2

- Comment définiriez-vous la variable d (ce qu'elle représente …)
- Comment définiriez-vous la variable f
- A quoi cet algorithme peut-il servir ?

Question 3

Traduire cet algorithme en Java et testez le.