UNIVERSITE DE CERGY-PONTOISE
DEPARTEMENT D'INFORMATIQUE
Algorithmique III- L3I-S5, 2019-20
© B. Derdouri

Lundi 4 Novembre 2019
DUREE 2h00 HEURES
DOCUMENTS NON AUTORISES

Partiel D'ALGORITHMIQUE III L3I

EXERCICE 1 (6.0PTS)

1) Soient un algorithme A et D(n) l'ensemble des jeux possibles de données de taille n.
Notons par Coût(A(d)) la complexité en temps de l'algorithme A sur la donnée d (d ∈ D(n)).

a) Définir les complexités suivantes :
- Complexité dans le meilleur des cas
- Complexité dans le pire des cas
- Complexité dans les moyenne des

b) Donner un exemple d'algorithme où les trois cas de complexités sont **différents** (Justifier votre réponse).

2) Décrire le principe de partitionnement d'un tableau donné.

3) Ecrire en langage algorithmique ou C, un algorithme **récursif** de partitionnement d'un tableau donné composé de n éléments (des entiers par exemple).

4) Proposer en C, un programme principal qui fait appel à l'algorithme de partitionnement.

5) Illustrer graphiquement le déroulement du tri rapide au tableau suivant : [3, 2, 7, 8, 1, 5, 4, 9, 0].

EXERCICE 2 (3PTS)

***Notes** : Pour chaque équation de la complexité, vous devriez expliquer la démarche suivie et de justifier les termes qui la composent.*

1) En ne comptabilisant que **les additions (lignes I et II)**, trouver et analyser en fonction de **n** l'équation de la complexité en temps de la fonction suivante :

```
Fonction tata(n) → Entier
Début
    Si n < 0 alors
         renvoyer 1
    Sinon
       som←0
       Pour i variant de 0 à n faire
            Som ← som + i          (I)
       FinPour
       renvoyer som + tata(n-1)  (II)
    FinSi
FIN
```

Paramètres Formels

n : entier. Paramètre en entré

Variables intermédiaires :

i, som : Entier.

2) En ne comptabilisant que **les multiplications (lignes I et II)** ; trouver puis résoudre en fonction de n l'équation de la complexité de la fonction toto.

```
Fonction toto(n, m) → Entier
Début
   Si n = 0 alors
      renvoyer 1
   Sinon
      Si estPair(n) alors
         renvoyer toto(n/2, m*m)   (I)
      Sinon
          renvoyer m*toto(n/2, m*m)  (II)
      FinSi
   FinSi
FIN
```

Paramètres Formels

n, m : entier. Paramètre en entré

EXERCICE 3 (1.5 PTS)

1) Traduire en C l'algorithme ci-dessous d'insertion en queue d'un élément donné dans une liste simplement chaînée donnée. La liste chaînée est définie par l'adresse de sa première cellule.

```
procédure inserQueue(elt, l)
Début
   Si estVide(l) alors
        inserTete(elt, l)
   Sinon
        inserQueue(elt, suivant(l))
   Fsi
Fin
```

Paramètres Formels
elt : TElement (E)
l : Liste (E/S)

Note : On suppose connue la fonction estVide et la procédure inserTete.

2) Ecrire en C, l'algorithme d'insertion en queue sous forme de fonction.

PROBLÈME (9.5 PTS)

Un multi-ensemble est constitué d'un ensemble de couples : (valeur, occurrence)**.** Chaque couple sera représenté par une structure composée de deux membres : valeur (un entier) et son occurrence (un entier positif non nul).

On suppose que les couples seront représentés par une **liste linéaire simplement chaînée** et classée dans **un ordre croissant suivant la valeur de l'élément (couple)**.

On souhaite construire un multi-ensemble à partir d'une suite d'entiers où certaines valeurs peuvent se répéter. La suite d'entiers sera représentée également par une **liste linéaire simplement chaînée.**

Exemple : à partir de la liste L d'entiers suivante : {13, 4, 1, 2, 13, 1, 0, 9, 0, 19, 12, 13, 1, 13}, on obtient le multi-ensemble ME suivant : { (0 , 2) , (1, 3) , (2, 1) , (4 , 1), (9 , 1) , (12, 1) , (13, 4) , (19 , 1) }

1) Ecrire en C les structures données nécessaires pour résoudre le problème ci-dessus.

Ecrire en langage algorithmique (ou C), les algorithmes suivants :

2) Construction d'un multi-ensemble à partir d'une liste chaînée d'entiers donnée.
3) A Partir d'un multi-ensemble donné, déterminer les valeurs qui possèdent une occurrence donné. Pour répondre à cette question, il faut justifier le choix de la structure qui mémorise les valeurs.

Dans ce qui suit, nous souhaitons représenter le multi-ensemble par une **liste linéaire doublement chaînée caractérisée avec deux adresses : l'adresse de la tête et l'adresse de la queue**.

Dans l'écriture des algorithmes ci-dessous ; vous devriez profiter pleinement des avantages des deux adresses. Par exemple :

-on utilise l'adresse de la queue de la liste pour effectuer les insertions en queue.

-on utilise les deux adresses pour parcourir simultanément dans les deux sens une liste doublement chainée.

-etc

4) Ecrire en C les structures de données nécessaires pour représenter le multi-ensemble.

Ecrire en langage algorithmique (ou C), les algorithmes suivants :

5) Construction d'un multi-ensemble à partir d'une liste chaînée d'entiers donnée.
6) A Partir d'un multi-ensemble donné, déterminer l'adresse d'un couple donné s'il existe et null s'il n'existe pas.