UNIVERSITE DE CERGY-PONTOISE
DEPARTEMENT D'INFORMATIQUE
Algorithmique III- L3I-S5, 2018-19
© B. Derdouri

Jeudi 27 Juin 2019
DUREE 2h00 HEURES
**DOCUMENTS NON AUTORISES**

# SESSION 2 - EXAMEN D'ALGORITHMIQUE III - L3I

## *Exercice 1 (5Pts)*

**1)** Expliquer les différentes étapes de l'approche 'Diviser Pour Régner' dans le cas du tri rapide.
**2)** Illustrer graphiquement le déroulement du tri rapide au tableau suivant : [11, 1, 6, 9, 10, 15, 18, 17, 3, 0, 14].
**3)** Ecrire en langage algorithmique ou en C, le partitionnement d'un tableau composé de n éléments.
**4)** Ecrire en langage algorithmique ou en C, le tri rapide d'un tableau.
**5)** En ne comptabilisant que **les comparaisons**, déterminer en fonction de la taille du tableau n, que l'équation de la complexité du tri rapide.

## *Exercice 2 (1.5 Pts)*

En ne comptabilisant que les exécutions de l'affichage produit par la procédure **écrire**; étudier la complexité en temps de la procédure `tita` en fonction de **n** (pour faciliter la résolution de l'équation de la complexité on suppose que n est une **puissance de 3**).

**`Procédure tita(n, s)`**

Paramètres Formels :
n : Entier (E )
s : Entier (E/S)

Variables intermédiaires :
m : Entier

```
DEBUT
      si n > 1 alors
            m ← n div 3
            pour i variant de 1 à n faire
                écrire('tita-tita')
            fpour
           tita(m, s)
           …
           tita(m, s)
           …
           tita(m, s)
            …
      sinon
                écrire('tata-tata')
     fsi
FIN
```

## *Exercice 3 (4.0Pts)*

On souhaite écrire un programme effectuant diverses opérations sur des polynômes d'une variable réelle. On rappelle qu'un polynôme de degré « n » ( $P(X)=\sum_{i=0}^{i=n} C_i X^i$ ) est constitué d'une somme de **monômes non nuls** ( $C_i X^i$ ). Chaque monôme est représenté par une structure composée de deux membres : degré (un entier) et coefficient (un réel).

Dans cet exercice, les polynômes seront représentés par une liste doublement chaînées de monômes non nuls et dans **l'ordre croissant suivant le degré**. Et pour faciliter la gestion des polynômes, on rajoutera deux adresses **: la première adresse pour indiquer la tête de la liste et la seconde adresse pour indiquer la queue de la liste.**

La structure C suivante représente un polynôme à coefficients réels :

```
typedef struct{
                double coef ;
                int degre ;
        }Monome ;
```



```
typedef struct cellule{
                Monome donnee;
                struct cellule *suivant ;
                struct cellule *précédent ;
        }*DListe;
typedef struct cellule{
                DListe tete;
                DListe queue;
        }Polynome;
```

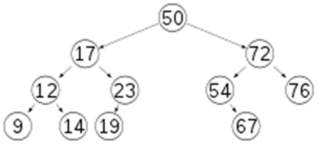
Notes:
Dans l'écriture des algorithmes ci-dessous ; vous devriez profiter pleinement des avantages des deux adresses. Par exemple :
-on utilise l'adresse de la queue de la Dliste pour effectuer les insertions en queue.
-on utilise les deux adresses pour parcourir simultanément dans les deux sens une Dliste.
-etc

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

**1)** En parcourant le polynôme dans les deux sens ; évaluer un polynôme donné pour un réel donné.
**2)** En parcourant le polynôme dans les deux sens ; vérifier l'existence d'un degré donné dans un polynôme donné.
**3)** Insérer un monôme donné, dans un polynôme donné.

## *Exercice 4 (3.5Pts)*

1) Dessinez tous les AVL qui contiennent les valeurs 1, 2, 3, 4, 5.
2) Insérer dans l'AVL ci-contre les valeurs 18 puis 60.
4) Ecrire en langage algorithmique ou en C, l'insertion d'un élément donné dans un AVL donné (note : n'oublier pas de gérer l'équilibrage de l'AVL).




## *Exercice 5 (3Pts)*

On implémente les arbres généraux avec la structure de données suivantes :
typedef struct noeud * ArbreG;
struct noeud{
TElement donnee;
ArbreG filsAine;
ArbreG freres;
};

Forme condensée avec une seule structure : un lien vers le fils aîné et un lien vers le frère

Le degré d'un nœud est le nombre des fils du nœud. Le degré d'un arbre est égal au plus grand des degrés de ses noeuds : d°(Arbre) = max { d° ( nd ) / nd est un noeud de l'Arbre }

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

1) Déterminer le degré d'un arbreG donné.
2) Déterminer le nombre d'occurrence d'un élément donné dans un arbreG donné.

## *Exercice 6 (3Pts)*

1) Décrire uniquement le principe de la résolution des collisions avec les méthodes suivantes :
   a) Hachage coalescent sans réserve.
   b) Hachage linéaire.
2) Application : soit les éléments E1, E2, E3, E4, E5, E6, E7, E8, E9, E10 dont les valeurs de hachage respectivement sont : 1, 5, 2, 5, 1, 8, 11, 9, 5, 10.

   On suppose que la table de hachage commence à l'indice 1 et de taille M=11.

   Décrire graphiquement pour chacune des méthodes (a) et (b) le remplissage de la table de hachage.