UNIVERSITE ABDERRAHMANE MIRA DE BEJAIA
Faculté des Sciences Exactes
Département d'Informatique

Module : ASD
Niveau : 2e Année Licence
Année : 2020/2021

TD N° 1 : La complexité algorithmique

Exercice N° 1 : calculer la complexité de l'algorithme suivant :

```
Algorithme complexité1 ;
Type Tab = Tableau [1..100] de réel ;
Var A : Tab ;
        n, j : entier ;
        p, x :réel ;
Fonction puissance (x : réel ; p : entier) : réel ;
Var i : entier ;
        puis : réel ;
Début
        i←1 ;
        puis←1 ;
        Tq (i≤p) faire
                puis←puis*x ;
                i←i+1 ;
        FTq
        puissance←puis ;
Fin ;
Fonction polynome (T: Tab ; x: réel ; n: entier): réel ;
Var i : entier ;
        poly : réel ;
Début
        poly←T[0] ;
        i←1 ;
        Tq (i≤n) faire
                poly←poly+T[i]*puissance(x,i) ;
                i←i+1 ;
        FTq
        polynome←poly ;
Fin ;
Début
        Lire(n) ;
        pour i allant de 0 à n faire
                lire(A[i]) ;
        FinPour ;
        Lire(x) ;
        p←plynome(A,x,n) ;
        Ecrire(p);
Fin.
```

Exercice N° 2
Calculer la complexité de la suite de Fibonacci en O(?) qui est définie comme suit :

$$f(n) = \begin{cases} 1, & x = 0 \; ou \; x = 1 \\ f(n-1) + f(n-2), & x > 1 \end{cases}$$

UNIVERSITE ABDERRAHMANE MIRA DE BEJAIA
Faculté des Sciences Exactes
Département d'Informatique

Module : ASD
Niveau : 2e Année Licence
Année : 2020/2021

TD N° 2 : Les algorithmes de tri

Exercice N° 1 :

Soit l'algorithme de « Tri par Fusion » suivant :

```
Procédure triFusion(var A : Tableau [1..n] d'entiers,
début, fin : Eentier) ;
Variables milieu : entier
début
        si (début < fin) Alors      // au moins deux cases
                milieu ←(début + fin)/2 ;
                triFusion(A, début, milieu) ;
                triFusion(A, milieu+1, fin) ;
                fusionner(A,debut,milieu,fin) ;
        finSi ;
fin ;
```

```
Procédure Fusionner(var A[n] : Tableau, début, milieu,
fin : entier) ;
Variables i, i1, i2 : entier ; tmp: Tableau [1..n] d'entiers ;
début
        i ←0 ;
        i1 ←début ;
        i2 ←milieu + 1 ;
        Tantque (i1 <= milieu) et (i2 <= fin) faire
                si A[i1] < A[i2] Alors
                        tmp[i] ←A[i1] ;
                        i1 ←i1 +1 ;
                Sinon
                        tmp[i] ←A[i2] ;
                        i2 ←i2 + 1
                finSi ;
                i ←i + 1 ;
        FinTantque ;
        Tantque i1 <= milieu faire
                tmp[i] ←A[i1] ;
                i ←i + 1 ; i1 ←i1 + 1 ;
        finTantque ;
        Tantque i2 <= fin faire
                Tmp[i] ←A[i2] ;
                i ←i + 1 ; i2 ←i2 + 1 ;
        finTantque ;
        A ← Tmp ;
fin ;
```

Q1- Dérouler cet algorithme sur le tableau A suivant :
A=[38, 27, 43, 3, 9, 82, 10] ;
Q2- Calculer sa complexité.

Exercice N° 2 :

Soit l'algorithme de « Tri Rapide » suivant :

```
Procédure triRapide(var A : Tableau [1..n] d'entiers,
début, fin : entier) ;
Variables indexPivot : entier ;
début
        si (début < fin) Alors   // au moins deux cases
                partitionner(A, début, fin, indexPivot) ;
                triRapide(A, début, indexPivot -1) ;
                triRapide(A, indexPivot +1, fin) ;
        finSi ;
fin ;
```

```
procédure Partitionner(var A :Tableau [1..n] d'entiers,
début, fin, var indexPivot: entier) ;
Variables i, pivot, cpt : entier ;
début
        cpt← début ;  pivot ← t[début] ;
        pour i ←début+1 à fin faire
                        si A[i] < pivot alors
                                cpt ← cpt+1
                                permuter(A[i],t[cpt]);
                finSi ;
        finPour ;
        permuter(A,cpt,début) ;
        indexPivot ← cpt ;

fin ;
```

Q1- Dérouler cet algorithme sur le tableau A suivant :
A=[27, 63, 1, 71, 64, 58, 94, 9] ;
Q2- Calculer sa complexité ;
Q3- Comparer cet algorithme à l'algorithme de « Tris par fusion » ; justifiez votre répons, conclure.

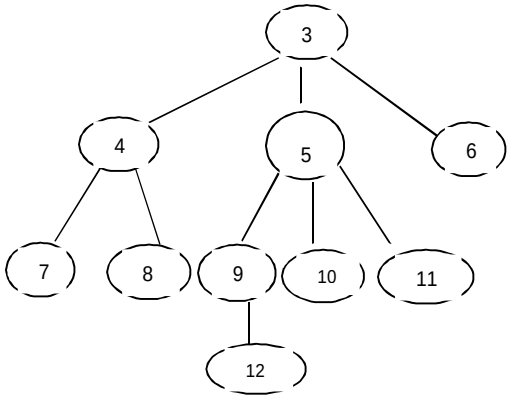
UNIVERSITE ABDERRAHMANE MIRA DE BEJAIA
Faculté des Sciences Exactes
Département d'Informatique

Module : ASD
Niveau : 2e Année Licence
Année : 2020/2021

TD N° 3 : les Arbres

Exercice 1 :

Considérer l'arbre ci-contre :



1) Quelle est la hauteur de cet arbre ?
2) Quel est son degré ?
3) Transformer cet arbre en arbre binaire B.
4) Afficher l'arbre binaire résultant des parcours : préfixée, infixée et postfixée.
5) Ecrire une procédure qui permet de construire la liste des feuilles d'un arbre.
6) Ecrire une fonction qui renvoie *vrai* si chaque élément de l'arbre est supérieur au double de ses fils, et *faux* sinon.
7) Transformer l'arbre **B** en un arbre binaire de recherche **R.**
8) Ecrire un sous-programme permettant de créer un arbre binaire de recherche **R** à partir de la liste linéaire chainée **L**.

Exercice 2 :

Soit une personne représentée par les informations suivantes : num, nom, prénom, sexe et âge. On souhaite construire l'arbre binaire généalogique d'une famille.

1) Définir les types adéquats ;
2) Ecrire des sous-programmes permettant de :
   - Créer la liste de tous les descendants d'une personne donnée ;
   - Afficher toutes les personnes ayant gardé le même nom de famille que la racine ;
   - Ajouter un enfant pour une personne donnée en gardant l'arbre binaire.

UNIVERSITE ABDERRAHMANE MIRA DE BEJAIA
Faculté des Sciences Exactes
Département d'Informatique

Module : ASD
Niveau : 2e Année Licence
Année : 2020/2021

TD N° 4 : les graphes

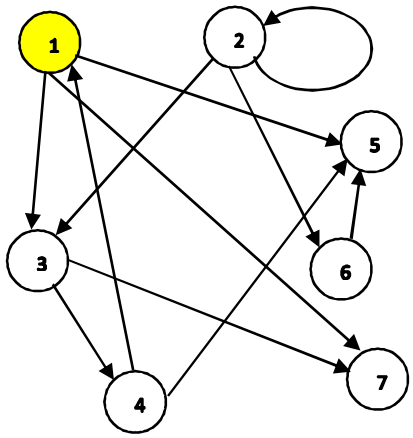
Exercice 1 :

Soit le graphe ci-dessus.

1- Donner la représentation de ce graphe en utilisant les deux approches : statique et dynamique ;

2- Appliquer les algorithmes de parcours en largeur et en profondeur sur ce graphe (on considère le sommet 1 comme sommet de départ) ;

3- Donner la complexité des deux algorithmes en O(?) pour chacune des représentations, conclure ?



Exercice 2 :

Soit un graphe g=(A, S), avec |S|=n et |S|=m.

Ecrire des sous-programmes permettant de :

1- Vérifier s'il existe un chemin entre 2 sommets x et y ;

2- Calculer les successeurs et les prédécesseurs d'un sommet x ;

3- Calculer la composante fortement connexe d'un sommet x ;

4- Calculer les composantes connexes d'un graphe (non orienté) ;

5- Calculer le niveau d'un sommet x par rapport à un sommet y.