Polytechnique Sousse EUR-ACE	**Algorithmique Avancée et langage C** **Classe :** 3ème Année Ingénieur Génie Informatique **Enseignant :** F. HAMDAOUI & S. Ben Ahmed

TD N° I

Exercice 1

Soit l'algorithme ci-dessous

```
fonction avec retour booléen rechercheElement(chaine[] tab, chaine x)
        entier i;
début
        i <- 0;
        tantque (i < tab.longueur) faire
            si (tab[i] = x) alors
                retourne VRAI;
            finsi
            i <- i + 1;
        fintantque
        retourne FAUX;
fin
```

Et soient : n = la taille du tableau, ae = affectation d'entier, ce = comparaison d'entier, oe=opération sur les entiers. Calculez le temps d'exécution pour les 3 cas suivants ainsi que l'ordre de grandeur correspondant O:

1. Complexité au pire (x n'est pas dans le tableau) :
2. Complexité au mieux (x est dans la première case du tableau) :
3. Complexité en moyenne : considérons qu'on a 50% de chance que x soit dans le tableau, et 50% qu'il n'y soit pas, et, s'il y est sa position moyenne est au milieu.

Exercice 2

Soit l'aalgorithme de la recherche dichotomique ci-dessous

```
fonction avec retour booléen rechercheDicho(chaine[] tab, chaine x)
        entier i, j;
début
        i <- 0;
        j <- tab.longueur-1;
        tantque (i <= j) faire
            si (tab[(j+i)/2] = x) alors retourne VRAI;
            sinon
                si (tab[(j+i)/2] > x) alors j <- (j+i)/2 - 1;
                sinon i <- (j+i)/2 + 1;
                finsi
            finsi
        fintantque
        retourne FAUX;
fin
```

1. Calculez la complexité au pire des cas : x n'est pas dans le tableau.
2. Pour des données de grande taille, si le tableau est trié, quel algorithme de recherche vaut mieux utiliser ?

Exercice 3

Vous disposez d'un tableau T de taille n. Etudiez la complexité en temps des fonctions suivantes sur les tableaux :

1. Tri par insertion
2. Tri par sélection
3. Tri à bulles
4. Tri rapide
5. Tri par fusion

Exercice 4

L'algorithme suivant détermine le quotient et le reste de la division entière de n par m (On suppose que n>m). Donner sa complexité algorithmique.

```
Procedure division (n, m : entier ; var q, r : entier )
 Début
    q←0
    r←n
    tant que (m <= r) faire
            q←q+1
            r←r-m
     fin tant que
 fin
```

Exercice 5

Soit un polynôme P tq : $P(x)=a^n x^n + a^{n-1} x^{n-1} + \ldots + a^1 x + a^0$

Ecrire une fonction permettant d'évaluer la valeur du polynôme pour un x donné. Calculer sa complexité en nombre d'additions et de multiplications. Peut-on mieux faire ?

Exercice 6

Ecrire la fonction minmax qui donne pour une matrice d'entiers m x n (m et n >=1), la valeur minmax représentant la valeur minimale des maximas de chaque ligne. Calculez sa complexité.

Exercice 7

En utilisant une représentation contigüe d'une liste, nous cherchons à manipuler un annuaire des clients d'une entreprise où chaque client est caractérisé par un nom de longueur 20 caractères et un numéro de téléphone de longueur 15 caractères. Les clients sont organisés au sein de l'annuaire selon l'ordre alphabétique des noms. Dans cet ordre, on ne prend en considération que la première lettre du nom ; les noms qui ont la même première lettre sont rangés successivement d'une façon aléatoire.
Indication : Dans cet exercice, vous pouvez faire appel aux primitives suivantes vues en cours si vous en aurez besoin : Vide, Init, taille, insérerDébut et insérerFin.

1. Définir les types ***Client*** et ***Annuaire***.
2. Ecrire une procédure ***ajouterClient*** qui permet d'ajouter un nouveau client ***C*** à un annuaire ***A*** entrés en paramètres.
   Procédure ajouterClient (C : Client, var A : Annuaire)
3. Calculer la complexité de la procédure ***ajouterClient*** de la question précédente.

Exercice 8

L'algorithme suivant se propose d'extraire la racine carrée entière (par défaut) d'un nombre *n* non négatif. Pour ce faire, nous allons procéder comme suit :
Par définition, si *r* est la racine carrée de *n* alors la relation suivante est vérifiée : $r^2 \leq n \leq (r+1)^2$

En se basant sur cette relation, on peut utiliser l'idée de la recherche dichotomique sur l'intervalle *[0, d]* où $d = 2^k$ et $2^{k-1} \leq \sqrt{n} \leq 2^k$.
La valeur de la racine est initialisée à 0. Tant que la longueur de l'intervalle *d* est plus grande que l'unité, on diminue de moitié cet intervalle et on ajoute à la valeur courante de la racine, la longueur *d* de l'intervalle trouvé, bien entendu dans le cas où $(r+d)^k \leq n$.
Ecrire l'algorithme et donner sa complexité.

Exercice 9

Étant données deux matrices A et B, respectivement, de dimensions (n,p) et (p,m), la matrice C produit de la matrice A par la matrice B. Rappelons que chaque élément C[i,j] (1<=i <=n, 1<=j <=m) est calculé selon la formule suivante :

$$C[i,j] = \sum_{k=1}^{p} A[i,k] \times B[k,j]$$

Donner la complexité algorithmique de la procédure Prod_Mat (n,m,p : entier)