TD5 - Complexité, Récursivité et Tableaux

Exercice 1 - Complexité des algorithmes

Les algorithmes suivants prennent en entrée deux entiers positifs m et n. En justifiant brièvement, déterminer la complexité de ces algorithmes.

```
i:=1
j:=1
TantQue (i<=m) et (j<=n) faire
  i := i+1
  j := j+1
FinTantQue
```

Algorithme A

```
i := 1
j := 1
TantQue (i<=m) ou (j<=n) faire
  i := i+1
  j := j+1
FinTantQue
```

Algorithme B

```
i := 1
j := 1
TantQue (j<=n) faire
  Si (i<=m) alors
    i := i+1
  Sinon
    j := j+1
  FinSi
FinTantQue
```

Algorithme C

```
i := 1
j := 1
TantQue (j<=n) faire
  Si (i<=m) alors
    i := i+1
  Sinon
    j := j+1
    i := 1
  FinSi
FinTantQue
```

Algorithme D

Exercice 2 - Preuve de complexité

Déterminer la complexité de l'algorithme suivant :

```
a := 1
Pour i := 1..x faire // Boucle (1)
  k := 0
  Pour j := 1..2a faire // Boucle (2)
    k := k+1
  FinPour
  a := k
FinPour
```

Exercice 3 - Complexité et comparaison des algorithmes

Les deux relations suivantes permettent le calcul de $X^N, X \in \mathbb{R}, N \geqslant 0$.

$$X^N = \begin{cases} 1 \text{ si N=0} \\ X * X^{N-1} \text{ sinon} \end{cases} \qquad X^N = \begin{cases} 1 \text{ si N=0} \\ (X * X)^{N/2} \text{ si N est pair} \\ X * (X * X)^{N/2} \text{ si N est impair} \end{cases}$$

1. Écrire en C deux fonctions récursives de calcul de X^N correspondants à ces deux formules.
2. Déterminer et démontrer les complexités de ces deux versions.

Exercice 4 - Recherche dichotomique

On considère un tableau A de n entiers triés par valeurs croissantes. L'algorithme suivant renvoie la position dans A où se trouve la valeur *cle* (on suppose que *cle* est dans le tableau).

```
entier Cherche_Dich_it(A: tableau; n, cle: entier)
  d := 1
  f := n
  trouve := FAUX
  Repeter
    i := (d+f)/2
    Si A[i]=cle alors
      trouve := VRAI
    Si A[i]<cle alors
      d := i+1
    Si A[i]>cle alors
      f := i-1
  Tantque (trouve = FAUX)
  retourner i
```

1. Développer cet algorithme pour rechercher la clé 30 dans le tableau [1,7,8,9,12,15,18,22,30,31].
2. Indiquer un invariant de boucle pour cet algorithme et le démontrer.
3. Pour un tableau de taille $n = 2^k, k > 0$, combien d'itérations l'algorithme effectue dans le pire des cas ? En déduire (sans démontrer) la complexité de cet algorithme.
4. Pour $k = 100$, comparer les complexités des algorithmes de recherche séquentielle et de recherche dichotomique.
5. Écrire l'algorithme sous forme récursive et l'exécuter sur l'exemple.
6. Prouver la validité de la version récursive et donner sa complexité.

Exercice 5 - Preuves d'algorithmes

Soit la fonction récursive F et n un entier positif.

```
Fonction F(n : entier) : entier
  Si n = 0 alors
    retourner 2
  Sinon
    retourner (F(n-1) * F(n-1))
  FinSi
```

1. Que calcule cette fonction ? Le prouver.
2. Quelle est la complexité de cette fonction ? Le prouver.
3. Quelle modification simple permet d'améliorer cet algorithme ?

Soit la fonction itérative G et n un entier positif.

```
Fonction G(n : entier) : entier
  R : entier := cste
  Pour i := 1..n faire
    R := R * R
  FinPour
  retourner R
```

4. Que calcule cette fonction ? Le prouver.
5. Quelle est la complexité de cette fonction ? Le prouver.