Algorithmique et Programmation – CPI1

TD5 : Complexité

1. Soit le jeu suivant consistant à tirer au sort un "vainqueur" parmi un groupe d'enfants, au moyen d'une pièce de monnaie :
   - S'il n'y a qu'un enfant, il est évidemment vainqueur.
   - S'il y en a plusieurs, on lance la pièce. Si c'est face qui sort, on élimine au hasard (par un jeu de type "chaises musicales" par exemple) un des enfants ; si c'est pile on en élimine deux (à moins qu'il n'en reste que deux, auquel cas on n'en élimine qu'un).
   - Le gagnant est le dernier enfant non éliminé.

   Si l'on considère le lancer d'une pièce comme opération fondamentale, trouver l'équation de récurrence correspondant à ce jeu dans le cas moyen. Montrer par récurrence, à partir de cette équation, que l'algorithme précédent est en O(n), où n est le nombre initial d'enfants.

2. On considère le code suivant :

```
i := 5;                              (1)
WHILE (i > 0) DO BEGIN               (2)
      s := 0;                        (3)
      FOR j := 1 TO i DO             (4)
            s := s + 1;              (5)
      i := i – 1;                    (6)
END ;
```

   Calculer sa complexité en nombre de comparaisons, d'affectations et d'opérations arithmétiques.

3. On considère la fonction suivante :

```
FUNCTION ex(n: INTEGER): INTEGER;
VAR a,b : INTEGER;
BEGIN
      a := 2;
      b := 1;
      WHILE (a < n) DO BEGIN
            a := a*2;
            b := b+1;
      END;
      ex := b;
END;
```

   a. Que vaut ex(n) si n est une puissance de 2 ? Que vaut ex(n) pour un argument n quelconque?

   b. Quelle est la complexité de cette fonction en nombre de comparaisons, de multiplications et d'affectations en fonction de n.

4. Donner l'ordre de complexité des algorithmes suivants (en fonction des variables n et m).

a)
```
x := 1;
FOR i := 1 TO n DO
   FOR j := 1 TO m DO
      x := x+1;
```

b)
```
x := 1;
FOR i := 1 TO n DO
   x := x+1;
FOR j := 1 TO m DO
   x := x+1;
```

c)
```
x := 1;
FOR i := n TO n+6 DO
   FOR j := m TO m+3 DO
      x := x+1;
```

d)
```
x := 1;
i := 1;
j := 1;
WHILE (i<=n) AND (j<=m) DO BEGIN
   x := x+1;
   i := i+1;
   j := j+1;
END;
WHILE (i<=n) DO BEGIN
   x := x+1;
   i := i+1;
END;
WHILE (j<=m) DO BEGIN
   x := x+1;
   j := j+1;
END;
```

5. Quel est le nombre d'<u>additions</u> réalisées par l'algorithme suivant dans
   a) le meilleur cas
   b) le pire cas
   c) le cas moyen, en supposant que les valeurs de T sont aléatoires entre 1 et 100 inclus.

```
type tabstat = array[1..N] of INTEGER;
FUNCTION foo(T: tabstat) : INTEGER;
VAR i,s : INTEGER;
BEGIN
        s := 0;
        FOR i := 1 TO N DO
                IF (T[i]*T[i]>100) THEN
                        s := s + T[i];
        foo := s;
END;
```

6. Voici deux versions différentes permettant de calculer la puissance n^k.

   Comparer leur ordre de complexité (il suffira de s'intéresser à un seul type d'opération élémentaire):

```
FUNCTION puissance(n,k:INTEGER):INTEGER;
BEGIN
  IF k = 0 THEN
    puissance := 1
  ELSE
    puissance := n*puissance(n,k-1) ;
END ;
```

```
FUNCTION puissance-dic(n,k: INTEGER): INTEGER;
BEGIN
  IF k = 0 THEN
    puissance-dic := 1
  ELSE IF (k mod 2 = 0)
    puissance-dic := puissance-dic(n*n,k DIV 2)
  ELSE
    puissance-dic := x*puissance-dic(n,k-1) ;
END;
```