Algorithmique et Programmation – CPI1

TD5 : Complexité - CORRIGE

1. Voici 2 versions différentes permettant de calculer la puissance n^k vues en TD 3. Comparer leur complexité :

```
Fonction puissance(n, k : Entier) : Entier
Début
  Si k = 0 Alors
    retourner 1
  Sinon
    retourner n*puissance(n,k-1)
  FinSi
Fin

Fonction puissance-dic(n, k : Entier) : Entier
Début
  Si k = 0 Alors
    retourner 1
  SinonSi (k mod 2 = 0)
       retourner puissance-dic(n*n,k/2)
  Sinon
       retourner x*puissance-dic(n,k-1)
  FinSi
Fin
```

Corrigé

On choisit la multiplication comme opération élémentaire. On note C(n) le nombre de multiplications au rang n.

Version 1 :

$C(0) = 0$

$C(n) = 1 + C(n-1)$

⇨ complexité linéaire (O(n)).

Version 2 :

$C(0) = 0$

$C(n) = 1 + C(n/2)$, si n pair (1)

$C(n) = 1 + C(n-1)$, si n impair (2)

Dans le pire des cas, n aura la forme $2^k + 2^{k-1} + ... + 2^1 + 1$ pour que l'on passe par (2) puis (1) puis (2) puis (1) ...

On a:

$C(n) = C(2^{k+1} -1) = 1 + C(2^{k+1} - 2)$

$C(2^{k+1} - 2) = 1 + C(2^k-1)$

$C(2^k-1) = 1 + C(2^k-2)$
$C(2^k-2) = 1 + C(2^{k-1}-1)$
...
$C(2^2 - 1) = 1 + C(2^2 - 2)$
$C(2^2 - 2) = 1 + C(2^1 - 1)$
$C(2^1 - 1) = 1 + C(0) = 1$

⇨ $C(2^{k+1} -1) = 2k+1$

⇨ $C(n) = 2\log(n+1) - 1$

⇨ complexité logarithmique O(logn)

2. 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.

Corrigé

1. Équation de récurrence dans le cas moyen :
On note C(n) le nombre de lancers avec n enfants. À chaque étape, on a une chance sur deux de retirer un enfant et une chance sur deux d'en retirer 2. En outre, on effectue un lancer par étape. Cela donne l'équation suivante, dans le cas moyen :

$$C(N) = 1/2C(N-1) + 1/2C(N-2) + 1$$

Le premier terme de la somme correspond au cas où l'on retire un enfant, le second à celui où l'on en retire deux. On sait en outre que C(1) = 0 (pas de lancer s'il n'y a qu'un enfant) et que C(2) = 1 (après un lancer, quel que soit le résultat, on se retrouve avec le gagnant).

2. Algorithme en O(N) ?
Pour montrer que cet algorithme est en O(n), on va supposer que

$$\forall i \leq n, C(i) \leq i$$

Cette propriété est clairement vraie pour n ≤ 2. Dans le cas général, on majore l'équation de C(n) par

$$C(n) \leq \frac{1}{2}(n-1) + \frac{1}{2}(n-2) + 1 = n - \frac{1}{2} \leq n$$

3. On considère le code suivant :

```
i ← 5                                   (1)
Tantque (i > 0)                         (2)
      s ← 0                             (3)
      Pour j = 1 à i                    (4)
            s ← s + 1                   (5)
      FinPour                           (6)
      i ← i - 1                         (7)
Fintantque
```

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

Corrigé

Nombre de comparaisons	Boucle 1 (ligne 2) : 6 Boucle 2 (ligne 4) : 6 + 5 + 4 + 3 + 2 **Total : 26**
Nombre d'affectations	Ligne 1 : 1 Ligne 3 : 5 * 1 (5 = nombre d'itérations de Boucle 1) Ligne 4 : 6 + 5 + 4 + 3 + 2 Ligne 5 : 5 + 4 + 3 + 2 + 1 Ligne 7 : 5 * 1 (5 = nombre d'itérations de Boucle 1) **Total : 46**
Nombre d'opérations arithmétiques	Ligne 4 (boucle) : 5 + 4 + 3 + 2 + 1 Ligne 5 (addition) : 5 + 4 + 3 + 2 + 1 Ligne 7 (soustraction) : 5 * 1 (5 = nombre d'itérations de Boucle 1) **Total : 35**

4. On considère la fonction suivante écrite en pseudo-code:

```
Fonction ex(n : Entier) : Entier
Variables a, b : Entier
Début
      a ← 2
      b ← 1
      Tantque a < n
            a ← a*2
            b ← b + 1
      FinTantque
      retourner b
Fin
```

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`.

Corrigé

a) Si $n = 2^b$, $ex(n) = b = \log_2 n$. Pour un argument n quelconque, $ex(n) = b = [\log_2 n] + 1$ ([ ] : partie entière).

b) La complexité de cette fonction en nombre de comparaisons, de multiplications et d'affectations en fonction de n

Ligne	Comparaison	Multiplication	Affectation
1	0	0	1
2	0	0	1
3 Gestion de compteur Corps de boucle	 b 0*(b-1) fois	 0 1*(b-1) fois	 0 (1+1)*(b-1) fois
4	0	1	1
5	0	0	1
6	0	0	0

Au total :

a. nombre de comparaisons : b
b. nombre de multiplications : b-1
c. nombre d'affectations : 2+2(b-1) = 2b

où $b = \log_2 n$ si n est une puissance de 2 et $b = [\log_2 n] + 1$ pour les autres cas.