TD n°6

Complexité et programmation par récurence

1 Rappel des notations

On rappelle les notations suivantes : pour deux fonctions f et g de $\mathbb{N}$ dans $\mathbb{N}$, on a :

$$f(n) = O(g(n)) \iff \exists C > 0,\ \exists n\ \geq 0,\ \forall n \geq n\ ,\ f(n) \leq C \cdot g(n) \quad (1)$$

$$f(n) = \ (g(n)) \iff \exists c > 0,\ \exists n\ \geq 0,\ \forall n \geq n\ ,\ c \cdot g(n) \leq f(n) \quad (2)$$

$$f(n) = \ (g(n)) \iff f(n) = O(g(n)) \text{ et } f(n) = \ (g(n)) \quad (3)$$

On constatera qu'il n'est pas necessare de parler de limite du quotient des fonctions.

2 Exercices

Exercice 1 Fonction en temps "constant"

Soit la fonction Java suivante

```
public static int foo(int n){
  if(n%2 == 0){
    return n;
  }
  else{
    return (n - 1);
  }
}
```

Montrer que même si sa complexite n'est pas strictement constante, elle est bien en (1).

Exercice 2 Recherche d'un élément dans un tableau

Soit la fonction de recherche d'un element dans un tableau

```
public static int search(int x, int[] t){
  for(int i = 0; i < t.length; ++i){
    if(t[i] == x){
      return i;
    }
  }
  return -1;
}
```

Montrer que la complexite de cette fonction est en $O(n)$. Est-elle en (n) ? On pourra considerer le tableau $[1, 2, \ldots, n]$ dans lequel on cherche 1.

Exercice 3 La fonction Fibonacci

Soit la fonction Fibonacci na•ve :

```
public static int fibo(int n){
  if(n<=1){
    return n;
  }
  else{
    return (fibo(n-1) + fibo(n-2));
  }
}
```

Montrer que sa complexite est $F(n)$ a une constante pres, ou $F(n)$ est le n-ieme nombre de... Fibonacci !

Montrer par recurence que

$$F(n) = \frac{\varphi^n - (1-\varphi)^n}{\sqrt{5}}$$

avec

$$\varphi = \frac{1+\sqrt{5}}{2} \approx 1,618\ldots$$

En deduire que la complexite de la fonction de Fibonacci est en (φ^n)

Exercice 4 Variation sur 6 7l La 384 fonction th

Exercice 5 Exponentiation rapide En vous inspirant de la \multiplication rapide" vue dans d'autres TDs et TPs, ecrivez une fonction recursive

```
public static int fastExp(int n, int p)
```

qui calcule n^p. On se souviendra que

$$n^{\,p} = (n)^p \quad (4)$$

$$n^p = n \cdot n^p \quad (5)$$

Montrer que sa complexite est en (log(p)).

Exercice 6 Multiplication matricielle

Ecrire une fonction *iterative* qui calcule le produit de deux matrices. On supposera que les dimensions correspondent.

```
public static int[][] multMat(int[][] mat1, int[][] mat2)
```

Montrer que si les deux matrices sont carrees, la complexitee est en (n). Apres avoir adapte les de nitions du debuts aux fonctions a plusieurs variables, montrer que la complexite est en ($n \cdot m \cdot p$) ou n, m et p sont les dimensions des deux matrices.

Exercice 7 Exponentiation de matrices

En combinant les deux exercices precedants, ecrire une fonction

```
public stativ int[][] expMat(int[][] mat, int p)
```

calculant `mat`^{p}. `mat` doit être carree. Quelle est sa complexite ?

Exercice 8 Retour à Fibonacci

En notant que

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} F(n-1) \\ F(n) \end{bmatrix} = \begin{bmatrix} F(n) \\ F(n+1) \end{bmatrix}$$

et en utilisant l'exercice precedant, ecrire une troisieme version

```
public static int fastFibo(int n)
```

calculant $F(n)$. Montrer que sa complexite est en (log(n))