Année 2010-2011

Corrigé du contrôle court du module complexité

I) 1. On donne les valeurs de S et de T[i] à l'initialisation et à la fin de chaque itération :

	S	**T[i]**
Init	**2**	**2**
i=4	**5**	**1**
i=3	**11**	**1**
i=2	**22**	**0**
i=1	**52**	**8**
i=0	**107**	**3**

2. On note S_i la valeur de la variable S à la fin de l'itération i et S_n sa valeur initiale.

On a : $S_n = T[n]$ et $S_i = T[i] + x * S_{i+1}$

On montre facilement par récurrence (inversée) que :

$$S_i = \sum_{j=i}^{n} T[j] * x^{j-i}$$

On en déduit que la fonction calcul évalue le polynôme de degré n dont les coefficients sont les valeurs du tableau T, au point x :

$$S_0 = \sum_{j=0}^{n} T[j] * x^{j}$$

On vient en fait de démontrer la validité de l'algorithme. Il reste à vérifier la terminaison. Celle-ci est bien sûr triviale puisque l'algorithme n'est constitué que d'une boucle de n itérations.

3. Une fonction récursive réalisant le même calcul est donnée ci-dessous :

```
fonction Calcul(T :tableau ; x :réel ; n, j :entier) :réel ;
    si j=n  alors retourner(T[n])
    sinon retourner(T[j]+x*Calcul(T, x, n, j+1))
    finsi ;
fin ;
```

L'appel initial de la fonction Calcul doit s'effectuer avec le paramètre j=0. Si on note $Calcul_j$ la valeur retournée par l'appel de la fonction Calcul avec le paramètre j, cette fonction récursive reproduit la relation mathématique de récurrence suivante :

$$Calcul_n = T[n] \quad \text{et} \quad Calcul_j = T[j] + x * Calcul_{j+1}$$

Cette relation est exactement la même que la relation de récurrence permettant de prouver l'algorithme itératif. La preuve de l'algorithme récursif s'effectue donc de la même façon que précédemment.

II) Si $S(n) \in O(f(n))$ *et* $T(n) \in O(g(n))$, alors il existe $k>0$, $k'>0$, $n_0 \geq 0$ et $n'_0 \geq 0$

Tels que :

$S(n) \leq kf(n) \quad \forall\ n \geq n_0$

$T(n) \leq k'f(n) \quad \forall\ n \geq n'_0$

Si $f(n) \in O(g(n))$, alors il existe $k''>0$ et $n''_0 \geq 0$ tels que

$f(n) \leq k''g(n) \quad \forall\ n \geq n_0$

On définit alors $K=kk'' + k'$ et $m_0 = \max(n_0, n'_0, n''_0)$. De façon évidente :

$S(n) + T(n) \leq kf(n) + k'g(n) \leq (kk'' + k')\, g(n) = Kg(n) \quad \forall\, n \geq m_0$

Donc s(n) + T(n) est en $O(g(n))$

Si dans un algorithme, on a une première partie en $O(f(n))$ suivie (séquentiellement) d'une seconde partie en $O(g(n))$ et que $f(n) \in O(g(n))$, alors l'algorithme est globalement en $O(g(n))$.