Université Paris 7 — Licence d'informatique
Algorithmique — Année 2003/04 — TD2

Eric Colin de Verdiere, Yan Jurski, Sylvain Peyronnet

Rappels

La notation $f(x) = \Theta(g(x))$ signifie qu'il existe c_1, $c_2 > 0$ tels que $0 \leq c_1 \times g(x) \leq f(x) \leq c_2 \times g(x)$ dès que x est suffisamment grand. Noter qu'elle ne s'applique qu'à des fonctions positives à partir d'un certain rang. On a :

1. Si f et g sont strictement positives pour x assez grand, $f(x) = \Theta(g(x))$ équivaut à $\frac{f}{g}$ et $\frac{g}{f}$ bornées pour x assez grand.
2. $f(x) = \Theta(a \times f(x))$ pour tout réel $a > 0$.
3. Θ est une relation d'équivalence, c'est-à-dire :
   - $f(x) = \Theta(f(x))$ (réflexivité),
   - si $f(x) = \Theta(g(x))$ alors $g(x) = \Theta(f(x))$ (symétrie),
   - si $f(x) = \Theta(g(x))$ et $g(x) = \Theta(h(x))$, alors $f(x) = \Theta(h(x))$ (transitivité).
4. Si $\lim f(x) = +\infty$ quand $x \to +\infty$, alors $f(x) + b = \Theta(f(x))$ pour toute constante réelle b.
5. Si $f_1(x) = \Theta(f_2(x))$ et $g_1(x) = \Theta(g_2(x))$, alors $f_1(x) + g_1(x) = \Theta(f_2(x) + g_2(x))$.

Exercice 1 Démontrer les propriétés (1) à (5). Pour (4), étudier d'abord le cas $b > 0$.

Exercice 2 Regrouper les fonctions suivantes suivant la relation d'équivalence Θ.

- $f_1(n) = n$;
- $f_2(n) = 2^n$;
- $f_3(n) = n \log_2(n)$;
- $f_4(n) = n - n^3 + 7n^5$;
- $f_5(n) = n^2 + \log_2(n)$;
- $f_6(n) = n^2$;
- $f_7(n) = 2^{2n}$;
- $f_8(n) = n^5$;
- $f_9(n) = \sqrt{n} + \log_2(n)$;
- $f_{10}(n) = \ln(2n)$;
- $f_{11}(n) = \ln(n)$;
- $f_{12}(n) = 3^n + n^2$;
- $f_{13}(n) = \log_2(n)$.

Exercice 3 Que calculent les deux fonctions suivantes? Quelle est leur complexité dans le pire, le meilleur, des cas ? En moyenne ?

```
inconnue1 (a) {
   p <- 0;
   Tant que (a modulo 2 = 0) Faire {
      a <- a div 2;
      p <- p + 1;
   }
   Si (a = 1) Alors
      retourner p;
   Sinon
      retourner -1;
}
```

```
inconnue2 (n, b) {
   p <- 1;
   r <- 0;
   Tant que (n > 0) Faire {
      r <- r + (n modulo 10) * p;
      n <- n div 10;
      p <- p * b;
   }
   retourner r;
}
```

Exercice 4 Soient `début` et `fin` deux constantes entières avec `début`$<$`fin`, et I l'ensemble des entiers entre `début` et `fin`. On code une fonction f de I dans I par un tableau `f` indexé par l'ensemble des entiers I, à valeurs dans I. Ainsi, la valeur de $f(i)$ est codée en pseudo-langage par `f[i]`.

On considère le programme suivant :

```
PROGRAMME inconnu
    CONSTANTES entières : début = 3, fin = 8
    TYPE TabFonc : tableau d'entiers indexé par [début..fin]

DÉBUT inconnu
    VARIABLES
        n,i,j : entiers
        f,g : TabFonc
    AFFICHER "Entrez un entier n"    LIRE n
    AFFICHER "Entrez la fonction f"  LIRE f

    COPIER f dans g

    POUR i <- 2 jusqu'à n FAIRE
        POUR j <- début jusqu'à fin FAIRE
            g[j] <- f[g[j]]
        FIN POUR
    FIN POUR

    AFFICHER g
FIN inconnu
```

1. Que vaut `g` à la fin ?
2. L'algorithme fonctionne-t-il encore si l'on remplace la ligne : `g[j] <- f[g[j]]` par : `g[j] <- g[f[j]]` ? Justifiez.

Exercice 5 Soit n un entier strictement positif et soit T un tableau de n nombres distincts. On appelle *maximum provisoire* de T tout indice i dans $[1, n]$ tel que, pour tout j dans $[1, i-1]$, $T[i]$ est plus grand que $T[j]$.

On veut calculer le nombre moyen de maxima provisoires, sur tous les tableaux T qui sont des permutations de $[1, n]$ (*i.e.*, des bijections de $[1, n]$ dans $[1, n]$).

1. On appelle $P_{n,k}$ le nombre de permutations de $[1, n]$ qui ont k maxima provisoires. Montrer que l'on a les relations :
   - $P_{1,1} = 1$, et $P_{1,k} = 0$ pour tout $k \neq 1$;
   - $P_{n,k} = P_{n-1,k-1} + (n-1)P_{n-1,k}$.
2. Soit S_n le nombre moyen de maxima provisoires d'une permutation de $[1, n]$. Montrer que :

$$S_n = \frac{1}{n!} \sum_{k=1}^{n} k P_{n,k}.$$

3. Calculer S_n.