DEUG 1^{ere} année

MIAS 1 et MASS 1 – Programmation fonctionnelle

TD n° 8 – Corrigé

Exercice 1 :

a) Le paramètre d'induction de la fonction `facto` est l'entier `n`. Le temps ainsi que l'espace mémoire nécessaire pour calculer le résultat de cette fonction ne dépend que de `n`.

Pour la complexité en temps, notons :

- C_f le temps (maximum et constant) consommé pour évaluer le corps de la fonction, sans compter le temps nécessaire pour évaluer l'éventuel appel récursif;
- $T_f(n)$ le temps consommé pour calculer le résultat de `(facto n)`.

Si $n = 0$ le calcul de `facto` nécessite uniquement d'évaluer le corps de la fonction. Sinon, le temps consommé est celui nécessaire pour évaluer le corps de la fonction auquel on ajoute le temps nécessaire au calcul de `(facto (- n 1))` (l'appel récursif). Le temps consommé est donc donné par les équations :

$$\begin{cases} T_f(0) &= C_f \\ T_f(n) &= C_f + T_f(n-1) \quad \text{, pour } n \geq 1 \end{cases}$$

$T_f(n)$ est une suite arithmétique de premier terme C_f et de raison C_f, on a donc $T_f(n) = (n+1)C_f = O(n)$.

Pour la complexité en espace, notons $E_f(n)$ le nombre de symboles de la plus longue expression que l'interpréte doit évaluer. Si $n = 0$ cette expression est `1`, et pour $n > 0$ c'est l'expression `(* n (facto (- n 1)))`. On a donc :

$$\begin{cases} E_f(0) &= 1 \\ E_f(n) &= 2 + E_f(n-1) \quad \text{, pour } n \geq 1 \end{cases}$$

C'est une suite arithmétique et on a $E_f(n) = 1 + 2n = O(n)$.

b)
```
(define approx
   (lambda (n)
      (if (= n 0)
          1
         (+ (/ 1 (facto n)) (approx (- n 1))))))
```

La complexité de cette fonction ne dépend que de la valeur de n. Pour la complexité en temps, notons :

- C_a le temps (maximum et constant) consommé pour évaluer le corps de la fonction, sans compter le temps nécessaire pour évaluer l'éventuel appel récursif;
- $T_a(n)$ le temps consommé pour calculer le résultat de `(approx n)`.

Cette fois, le temps consommé pour calculer `(approx n)` est fonction du temps consommé pour calculer `(approx (- n 1))` mais aussi du temps consommé par le calcul de `(facto n)`. On a donc :

$$\begin{cases} T_a(0) &= C_a \\ T_a(n) &= C_a + T_a(n-1) + T_f(n) = C_a + T_a(n-1) + (n+1)C_f \quad \text{, pour } n \geq 1 \end{cases}$$

Équations qu'on peut résoudre par sommation :

$$\begin{array}{lcllll} T_a(n) & = & C_a & + & T_a(n-1) & + & (n+1)C_f \\ T_a(n-1) & = & C_a & + & T_a(n-2) & + & nC_f \\ T_a(n-2) & = & C_a & + & T_a(n-3) & + & (n-1)C_f \\ \ldots & & & & & & \\ T_a(2) & = & C_a & + & T_a(1) & + & 3C_f \\ T_a(1) & = & C_a & + & T_a(0) & + & 2C_f \\ T_a(0) & = & C_a & & & & \\ \hline T_a(n) & = & (n+1)C_a & + & & & C_f \sum_{k=2}^{n+1} k \end{array}$$

En se souvenant que $\sum_{k=1}^{n} = \frac{n(n+1)}{2}$, on obtient : $T_a(n) = (n+1)C_a + C_f(\frac{(n+1)(n+2)}{2} - 1) = O(n^2)$

Pour la complexité en espace, notons $E_a(n)$ le nombre de symboles de la plus longue expression que l'interpréte doit évaluer. Si $n = 0$ cette expression est `1`, et pour $n > 0$ c'est l'expression `(+ (/ 1 (facto n)) (approx (- n 1)))`. On a donc :

$$\begin{cases} E_a(0) &= 1 \\ E_a(n) &= 3 + E_a(n-1) + E_f(n) = 3 + E_a(n-1) + 1 + 2n \quad \text{, pour } n \geq 1 \end{cases}$$

On résoud de nouveau ces équations par sommation et on trouve : $E_a(n) = 4n + 1 + 2\sum_{k=2}^{n} = O(n^2)$.

Exercice 2 :

Comme on admet que les fonctions prédéfinies `sqrt` et `remainder` ont des complexités en temps en $O(1)$, on considérer que l'évaluation du corps de la fonction (sans l'appel récursif) prend un temps constant indépendant et de p. Notons C ce temps.

n étant fixé, si $p \geq \sqrt{n}$, le temps maximum consommé pour calculer `(trouv-aux p)` est C.

Maintenant, si $p < n$, deux cas se présentent : soit p divise n et le calcul est terminé, soit p ne divise pas n et nécessaire de faire un appel récursif. Comme on veut évaluer le temps maximum consommé, on se place dans où p ne divise jamais n. Dans ce cas on a :

$$\begin{cases} T(p,n) &= C & \text{, si } p \geq n \\ T(p,n) &= C + T(p+1,n) & \text{, pour } n < p \end{cases}$$

Le temps consommé pour le calcul de `(trouvdiv n)` est égal à $T(2,n)$ qu'on peut calculer à l'aide des équ précédentes :

$T(2,n) = C + T(3,n) = C + C + T(4,n) = C + C + ... + T(s,n)$ où $s = \lceil\sqrt{n}\rceil$) est l'entier immédiatement supérieur

C'est à dire : $T(2,n) = C\lceil\sqrt{n}\rceil = O(\sqrt{n})$

La complexité en temps de la fonction `trouvdiv`, et par conséquent de la fonction `testprem`, est donc proportio à la racine carrée de n.