**Année 2010-2011 - Semestre de printemps**
**UE : INF152 - Algorithmes et Programmes**
Date : 17 juin 2011
Épreuve de : A. Dicky, G. Melançon

# Corrigé succinct

**Exercice 1.**

1. Appels successifs à `mystere` :

```
mystere(1,1,5)
   mystere(2,1,4)
      mystere(3,2,3)
         mystere(5,3,2)
            mystere(8,5,1)
               mystere(13,8,0)
```

   La valeur retournée est 8.

2. `mystere(1,1,n)` calcule le terme d'indice `n` de la suite de Fibonacci (définie par ses premiers termes $u_0 = u_1 = 1$, et par la relation de récurrence $u_{n+2} = u_{n+1} + u_n$).
3. L'exécution de `mystere(a,b,n)` provoque `n` appels récursifs à la fonction `mystere` (le paramètre `n` diminue de 1 à chaque appel récursif).

**Exercice 2.**

1. Trace de l'exécution de `triRapideRec(t,0,len(t))` pour `t=[7,8,42,1,5]` :

| | `indicePivot` | `debut` | `fin` | `t[debut...fin]` |
|---|---|---|---|---|
| départ | non défini | 0 | 5 | `[7,8,42,1,5]` |
| boucle 1 | 2 | | | `[1,5,7,42,8]` |
| | | 3 | | |
| boucle 2 | 3 | | | `[42,8]` |
| | | 4 | | `[8,42]` |

2. Lorsque le tableau est déjà trié, l'appel `reorganiser(t,debut,fin)` ne le modifie pas, et retourne `debut`. L'exécution de `triRapideRec(t,0,len(t))` équivaut à celle de la boucle

```
for debut in range(0,n-1):
  reorganiser(t,debut,n)
  triRapideRec(t,debut,debut)
```

   Comme `triRapideRec(t,debut,debut)` s'arrête sans appel récursif, il y aura `n-1` appels récursifs, donc en tout `n` appels, à `triRapideRec`.

   Mais la complexité de `reorganiser(t,debut,fin)` est proportionnelle à `fin-debut` (même si la fonction ne fait rien), donc la complexité de `triRapide(t)` reste quadratique (de l'ordre de $n^2$).

(En réalité, cette variante du tri rapide ne change pas les opérations élémentaires : les mêmes comparaisons et les mêmes affectations sont effectuées, dans le même ordre que dans l'algorithme d'origine.)

**Problème.**

1. Addition de polynômes :

```
def somme(p1,p2):
    p = creerTableau(DMAX + 1)
    for i in range(DMAX + 1):
        p[i] = p1[i] + p2[i]
    return p
```

   L'exécution requiert `DMAX` additions et `DMAX` affectations d'éléments de tableau, donc en tout 2`DMAX` opérations élémentaires.

2. Calcul du degré :

```
def degre(p):
    i = DMAX
    while i > 0 and p[i] == 0:
        i -= 1
    return i
```

3. Produit (le degré du produit de deux polynômes est la somme de leurs degrés) :

```
def produit(p1,p2):
    if degre(p1)+degre(p2) > DMAX:
        return None
    p = creerTableau(DMAX+1)
    for i in range(DMAX+1):
        for j in range(DMAX+1-i):
            p[i+j] += p1[i] * p2[j]
    return p
```

   L'exécution requiert $\texttt{DMAX}^2$ additions et $\texttt{DMAX}^2$ affectations d'éléments de tableau.

4. Normalisation :

```
def normaliser(p):
    nombreZerosInutiles = 0
    i = len(p) -1
    while i > 0 and p[i] == 0:
        nombreZerosInutiles += 1
        i -= 1
    supprimerNcases(p,nombreZerosInutiles)
```

5. Somme de polynômes représentés par des tableaux de tailles différentes :

```
def somme2(p1,p2):
    # initialisation
    t1 = len(p1)
    t2 = len(p2)
    if t1 < t2:
        tmin = t1
        tmax = t2
        pmin = p1
```

```
        pmax = p2
    else:
        tmin = t2
        tmax = t1
        pmin = p2
        pmax = p1
    p = creerTableau(tmax)
    # addition des monômes de même degré
    for i in range(tmin):
        p[i] = pmin[i] + pmax[i]
    # ajout des monômes de degré supérieur au degré de pmin
    for i in range(tmin,tmax):
        p[i] = pmax[i]
    normaliser(p)
    return p
```

L'initialisation s'effectue en temps constant, l'addition des monômes de même degré en temps proportionnel au plus petit des degrés, l'ajout des monômes restants en temps proportionnel à la différence des degrés, et la normalisation, dans le pire des cas, en temps proportionnel au plus grand des degrés. En définitive, l'algorithme s'exécute en temps linéaire par rapport au maximum des degrés.

6. Pour mettre sous forme canonique une somme de polynômes, on écrit les monômes dans l'ordre décroissant des degrés, en groupant les monômes de même degré (qu'on élimine si la somme des coefficients est 0).

   Remarque : l'algorithme est proche du tri par sélection (on cherche et on regroupe d'abord les monômes de plus grand degré, puis ceux de degré immédiatement inférieur, etc.)

7. La représentation par formes canoniques réduit le nombre de monômes, donc le nombre d'opérations élémentaires à effectuer (additions ou affectations d'éléments de tableaux). Mais les résultats doivent être ordonnés par degrés décroissants, ce qui nécessite des opérations supplémentaires (comparaison de degrés).

   Remarque : on peut mettre sous forme canonique la somme de deux polynômes donnés sous forme canonique en temps linéaire par rapport à la taille de ces polynômes (nombre de monômes non nuls), par un algorithme du type de la fusion de deux listes triées. Si les polynômes qu'on analyse sont «creux» (taille très inférieure au degré), l'addition est plus efficace qu'en utilisant la représentation précédente.