LU2I003 TD 3 Terminaison et validité d'un algorithme récursif

Exercice 1 – Calcul récursif du nombre d'occurrences d'un élément dans un tableau

On considère la fonction suivante :

```
def nb_occ(tab, k, x):
    print('tab=', tab[0:k], 'x=', x)
    if k == 0:
        res=0
    else :
        if tab[k - 1] == x:
            res=nb_occ(tab, k - 1, x) + 1
        else:
            res=nb_occ(tab, k - 1, x)
    print ('tab=', tab[0:k], 'x=', x, 'nboccurences=', res)
    return res
```

On rappelle que `tab[0:k]` est le sous tableau de `tab` composé des `k` premiers éléments.

Question 1

Exécutez `nb_occ(tab,k,x)` pour `tab=[3,6,7,6,2,6,3]`, `k=7` et `x=6`. Vous ne donnerez que les affichages obtenus.

Question 2

Démontrez que la fonction `nb_occ(tab,k,x)` se termine et renvoie le nombre d'occurrences de `x` dans le tableau `tab` entre les positions 0 (comprise) et `k-1` (comprise).

Exercice 2 – Algorithme récursif pour une suite récurrente linéaire de degré 2

On considère dans cet exercice le calcul de la suite récursive $u_n \geq 0$ définie par $u_0 = 3$, $u_1 = 7$ et pour tout $n \geq 2$, $u_n = u_{n-1} + u_{n-2} + 7n$.

```
def MaSuite(n):
     print 'Appel de MaSuite(', n , ')'
     if (n==0):
          res=3
     elif (n==1):
          res=7
     else:
          res=MaSuite(n-1)+MaSuite(n-2)+7*n
     print 'MaSuite(', n , ')=',  res
     return res
```

Question 1
Exécutez `MaSuite(4)` en précisant tous les affichages obtenus. En déduire l'arbre des appels avec les valeurs de retour.

Question 2
Démontrez la terminaison et la validité de la fonction `MaSuite(n)` par récurrence sur n. La validité correspond ici à démontrer que `MaSuite(n)` retourne u_n.

Exercice 3 – Recherche récursive d'un élément minimum dans un tableau

On considère la fonction suivante qui permet de renvoyer l'indice minimum du plus petit élément dans un tableau d'entiers non vide. La notation `tab[0:n-1]` est la restriction tab aux $n-1$ premières valeurs.

```
def RechercheMin(tab,n):
    print('tab=', tab[0:n])
    if (n==1):
        imin=0
    else :
        imin= RechercheMin(tab,n-1)
        if (tab[imin]>tab[n-1]):
            imin=n-1
    print('L indice du min de tab=', tab[0:n], 'vaut imin=',imin)
    return imin
```

Question 1
Exécutez la fonction pour le tableau `tab = [9, 8, 10, 6, 1, 7]`. Attention à bien effectuer les affichages correspondant à l'exécution de la fonction.

Question 2
Démontrez la terminaison et la validité de `RechercheMin` pour tout tableau non vide d'entiers.

Exercice 4 – Puissances de x

Soient x et n deux entiers positifs. Le but de cet exercice est d'étudier un algorithme récursif pour calculer x^n.

Question 1
Soit $u_n(x)$, $n \geq 0$ la suite définie par :

$$u_n = \begin{cases} 1 & \text{si} \quad n = 0 \\ (u_{\frac{n}{2}}(x))^2 & \text{si} \quad n \text{ est pair} \\ (u^{\frac{n-1}{2}}(x))^2 \times x & \text{si} \quad n \text{ est impair} \end{cases}$$

1. Calculez $u_{10}(2)$.
2. Démontrez par récurrence forte sur n que $u_n(x) = x^n$.

Question 2
Soit la fonction `PuissanceRecursive` définie comme suit :

```
def PuissanceRecursive (x, n):
        print "n vaut", n
        if (n == 0) :
                res = 1
        else :
                res = PuissanceRecursive (x, n/2)
                if (n % 2 == 0) :  # Si n est pair
                        res = res*res
                else :
                        res = res*res*x
        print "PuissanceRecursive (", x, "," , n, ")=", res
        return res
```

Exécutez `PuissanceRecursive(2,10)`. Vous préciserez l'arbre des appels et l'état de la pile des exécutions au moment où le dernier appel est empilé.

Question 3

Démontrez par récurrence la validité et la terminaison de la fonction `PuissanceRecursive`.

Exercice 5 – Fonction PGCD

Pour tout couple $(x, y) \in \mathbb{N}^2$, $pgcd(x, y)$ désigne le plus grand commun diviseur de x et de y.

Question 1

Démontrez que si $x > 0$, $pgcd(x, 0) = x$.

Question 2

Démontrer que, pour tout couple $(x, y) \in \mathbb{N}^2$ avec $y \neq 0$, $pgcd(x, y) = pgcd(y, x \ mod \ y)$.

Question 3

On considère maintenant la fonction `PGCD` définie pour $x > y \geq 0$ de la manière suivante :

```
def PGCD(x,y):
     print 'Appel de PGCD(', x ,  ',' , y ,')'
     if y==0 :
          res=x
     else:
          res=PGCD(y, x % y)
     print 'PGCD(', x ,  ',' , y ,')=', res
     return res
```

Donnez les affichages successifs de `PGCD(28,20)`. En déduire l'arbre des appels (avec les valeurs de retour).

Question 4

Démontrer que, pour tout couple $(x, y) \in \mathbb{N}^2$ tel que $x > y \geq 0$, $PGCD(x, y)$ se termine et renvoie la valeur $pgcd(x, y)$.

Exercice 6 – Problème de décomposition - Extraits de l'examen de juin 2018

Préliminaires

On dit qu'une suite d'entiers $(M_0, \ldots, M_{n-1})$ est *supercroissante* si tous ses éléments sont strictement positifs et si chaque élément est strictement supérieur à la somme de ses précédents : $M_i > M_0 + \ldots + M_{i-1}$ pour tout $i = 1, \ldots, n-1$.

étant donné un entier naturel s et une suite $M = (M_0, \ldots, M_{n-1})$ supercroissante, on cherche à savoir s'il existe n valeurs binaires $x_0, \ldots, x_{n-1}$ ($x_i = 0$ ou 1) telles que $s = x_0 M_0 + \ldots + x_{n-1} M_{n-1}$.
Dans cet exercice on appellera ce problème : *problème de décomposition*.

Question 1

Pour chacune des suites ci-dessous, dire si elle est supercroissante, en justifiant la réponse :
a) $(1, 3, 4, 7, 10)$; b) $(2, 3, 8, 14, 31)$; c) $(1, 2, 4, 8)$.

Question 2

On considère la suite supercroissante $M = (1, 3, 5, 10, 20)$. Pour chacun des entiers suivants, dire si le problème de décomposition admet une solution :
$s_1 = 0$; $s_2 = 26$; $s_3 = 17$.
S'il existe une solution, donner les valeurs des x_i.
On considère la fonction :

```
def plusGrandInd(M,s):
    n = len(M)
    k = 1
    while (k < n) and M[k] <= s:
        k = k + 1
    return k - 1
```

où M est un tableau de nombres rangés en ordre croissant et s un nombre supérieur ou égal à $M[0]$.
On rappelle que `len(M)` est le nombre d'éléments du tableau M.

Question 3

On note k_i la valeur de k à la fin de l'itération i. Initialement, c'est-à-dire à la fin de l'itération 0, $k_0 = 1$.

1. Montrer, par récurrence sur i, qu'à la fin de l'itération i, on a $M[j] \leq s$ pour tout j tel que $0 \leq j < k_i$.
2. En déduire que `plusGrandInd(M,s)` retourne le plus grand indice p tel que $M[p] \leq s$.

Question 4

Soit $M = (M_0, \ldots, M_{n-1})$ une suite supercroissante.

1. Quelle est la plus petite valeur non nulle de s pour laquelle le problème de décomposition admet une solution ?
2. Quelle est la plus grande valeur de s pour laquelle le problème de décomposition admet une solution ?

On considère la fonction :

```
def existeDec(M, s):
    print ("Valeur de s : ", s)
    if s == 0:
        return True
    if s < M[0]:
        return False
    p = plusGrandInd(M, s)
    print ("Valeur de p : ", p)
    if s - M[p] >= M[p]:
        return False
    return existeDec(M, s - M[p])
```

où M est un tableau représentant une suite supercroissante et s un entier naturel.

Question 5

Exécuter l'appel de `existeDec(ExM, 42))`, avec `ExM = [2,3,8,14,31]`, en donnant les affichages successifs et le résultat final.

Question 6

Soit $M = (M_0, \ldots, M_{n-1})$ une suite supercroissante et s un entier naturel non nul. Soit p le plus grand indice tel que $M_p \leq s$.

1. Montrer que, si $s = x_0 M_0 + \ldots + x_{n-1} M_{n-1}$ avec $x_i \in \{0, 1\}$ pour tout $i \in \{0, \ldots n-1\}$, alors :
   (a) $x_j = 0$ si $j > p$,
   (b) $x_p = 1$.

   Indication : faire ces deux preuves en utilisant un raisonnement par l'absurde.
2. En déduire que :
   (a) si $s - M_p \geq M_p$ alors le problème de décomposition n'a pas de solution pour (M, s) (faire un raisonnement par la contraposée) ;
   (b) si $s - M_p < M_p$ alors le problème de décomposition a une solution pour (M, s) ssi il en a une pour $(M, s - M_p)$.

Question 7

Prouver, par récurrence forte sur s, que `existeDec(M,s)` se termine et renvoie la valeur `True` si s admet une décomposition et la valeur `False` sinon.

Indication : pour la base, on étudiera le cas où $s = 0$ et le cas où $0 < s < M[0]$.