Chapitre 1
Programmation récursive

TD 1

Exercices d'application

TD d'informatique du Lycée Louis Legrand – Jean-Pierre Becirspahic

http://info-llg.fr/

Savoirs et compétences :

- ❒ *Alg – C15 : Récursivité : avantages et inconvénients.*

Exercice 1 – Fonction 91 de McCarthy

On considère la fonction récursive suivante :

■ Python
```
def f(n) :
    if n>100 :
        return n−10
    return f(f(n+11))
```

Question *Prouver sa terminaison lorsque $n \in \mathbb{N}$ et déterminer ce qu'elle calcule (sans utiliser l'interpréteur de commande).*

Correction Montrons que n est un variant de boucle.

Si $n \geq 101$, l'algorithme se termine.

Si $n \in [90, 100]$, l'algorithme appelle $f(n+11)$. $n+11$ sera supérieur à 101. et $f(n+11)$ retournera donc un nombre a compris entre $[91, 101]$.

Exercice 2

Question 1 *Écrire une fonction récursive qui calcule a^n en exploitant la relation : $a^n = a^{n/2} \times a^{n/2}$.*

Correction
```
def power(a, n):
    if n == 0:
        return 1
    elif n == 1:
        return a
    return power(a, n//2) * power(a, n−n//2)
```

Question 2 *Écrire une fonction qui utilise de plus la remarque suivante :* $n/2 = \begin{cases} n/2 & \text{si } n \text{ est pair} \\ n/2+1 & \text{sinon} \end{cases}$.

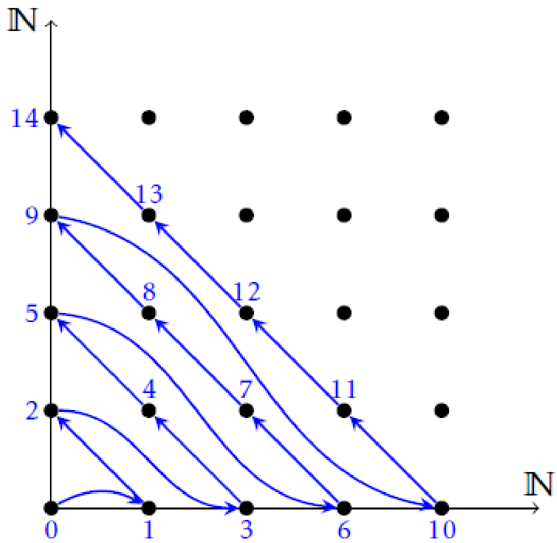
Correction
```
def power(a, n):
    if n == 0:
        return 1
    elif n == 1:
        return a
    x = power(a, n//2)
    if n % 2 == 0:
        return x * x
    else:
        return x * x * a
```

Question 3 *Déterminer le nombre de multiplications effectuées dans chacun des deux cas.*

Correction On note $C(n)$ le nombre multiplications.
Dans le premier cas, on conjecture que $\forall n > 2, C(n) = n - 1$ et on raisonne par récurrence
Dans le second cas, on conjecture que $\forall n > 2, C(n) \leq 2n$ et on raisonne par récurrence.

Exercice 3

On démontre que sur l'ensemble $\mathbb{N} \times \mathbb{N}$ est dénombrable en numérotant chaque couple $(x, y) \in \mathbb{N}^2$ suivant le procédé suggéré par la figure ci-dessous.



Question 1 *Rédiger une fonction récursive qui retourne le numéro du point de coordonnées* (x, y).

Correction
```
def numerote(x, y):
    if x == 0 and y == 0:
        return 0
    if y > 0:
        return 1 + numerote(x+1, y-1)
    return 1 + numerote(0, x-1)
```

Question 2 *Rédiger la fonction réciproque, là encore de façon récursive.*

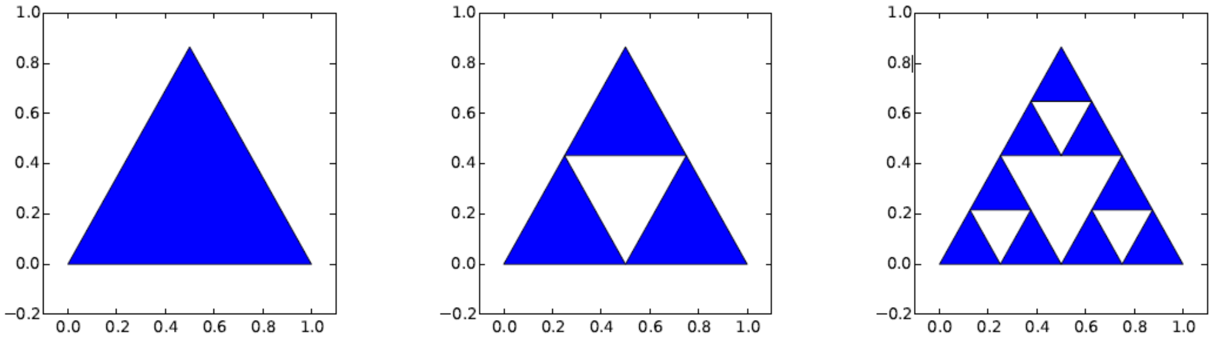
Correction
```
def reciproque(n):
    if n == 0:
        return (0, 0)
    (x, y) = reciproque(n-1)
    if x > 0:
```

```
        return (x-1, y+1)
    return (y+1, 0)
```

Exercice 4

On suppose disposer d'une fonction polygon((xa, ya), (xb, yb), (xc, yc)) qui trace le triangle plein dont les sommets ont pour coordonnées $(x_a; y_a)$, $(x_b; y_b)$, $(x_c; y_c)$.

Question 1 *Définir une fonction récursive permettant le tracé présenté figure suivante (tous les triangles sont équilatéraux).*



Correction

```
def polygon(*args):
    X, Y = [], []
    for arg in args:
        X.append(arg[0])
        Y.append(arg[1])
    plt.fill(X, Y, 'b')

from numpy import sqrt
def sierpinski(n, a=(0, 0), b=(1, 0), c=(.5, sqrt(3)/2)):
    if n == 1:
        polygon(a, b, c)
    else:
        u = ((b[0]+c[0])/2, (b[1]+c[1])/2)
        v = ((c[0]+a[0])/2, (c[1]+a[1])/2)
        w = ((a[0]+b[0])/2, (a[1]+b[1])/2)
        sierpinski(n-1, a, w, v)
        sierpinski(n-1, w, b, u)
        sierpinski(n-1, v, u, c)
```