Sujet n°3

Algorithmique & Programmation

Problèmes algorithmiques récursifs

1 Exercices simples

Exercice 1 : *Factorielle*

Écrire une fonction récursive qui calcule la factorielle d'un entier naturel n. On rappelle que $n! = 1 * 2 * 3 * \ldots n$, si $n \geq 1$ et $0! = 1$.

Exercice 2 : *PGCD*

Écrire une fonction qui retourne le plus grand commun diviseur (pgcd) de deux nombres entiers positifs par l'algorithme d'Euclide :

$$pgcd(a,b) = \begin{cases} a & \text{si } b = 0 \\ pgcd(b, a \bmod b) & sinon \end{cases}$$

Exercice 3 : *Dichotomie*

Générer aléatoirement et trier (fonction `liste.sort()`) un tableau de 100 nombre de 0 à 100 et effectuer une recherche dichotomique de nombres tapés au clavier par l'utilisateur grâce à une fonction récursive.

Vous devrez afficher l'indice du tableau contenant le nombre recherché si trouvé ou un message explicite dans le cas contraire.

Exercice 4 : *Les bases*

Le quotient de la division euclidienne d'un entier n par 10 donne le nombre de dizaines de cet entier. Le quotient de la division euclidienne de $n = 5478$ par 10 est par exemple 547.

QUESTION 1. Construire une fonction `NbChiffres(n)` prenant en paramètre un entier naturel n (écrit en décimal) et retournant le nombre de chiffres de cet entier n en base 10. Cette fonction sera définie récursivement.

QUESTION 2. Modifier cette fonction en ajoutant un deuxième paramètre `base` permettant de donner le nombre de chiffres de l'entier n (toujours fourni en base 10) en base `base`.

QUESTION 3. Créer une fonction récursive `Convert` affichant un nombre n exprimé en base 10 en une base `base` fournie en paramètre (pour simplifier l'affichage, on considèrera $base < 10$).

Ex : `Convert(10,2)` affiche `1010`

QUESTION 4. Créer une fonction récursive `ConvertInv` affichant en inverse un nombre n exprimé en base 10 en une base `base` fournie en paramètre (pour simplifier l'affichage, on considèrera $base < 10$).

Ex : `Convert(10,2)` affiche `0101`

Exercice 5 : *Implémentation de l'algorithme* `QuickSort`

QUESTION 1. Créer une fonction `Quicksort` implémentant l'algorithme *QuickSort*. Cette fonction doit prendre en paramètre le tableau des nombres à trier et permuter ces derniers de manière à les trier dans un ordre croissant.

Pour rappel, le tri *QuickSort* adopte la stratégie *« diviser pour régner »* qui consiste à réduire le problème du tri d'un tableau de taille n aux tris de 2 tableaux de taille $\frac{n}{2}$.

On choisit un élément dans le tableau (on le prend en général au hasard, mais par commodité on prendra ici le premier élément du tableau) qu'on appelle *pivot*. On sépare ensuite les autres éléments entre ceux inférieurs au pivot et ceux supérieurs au pivot. Il n'y a plus alors qu'à trier ces deux moitiés de manière récursive.

> ***Note :*** *cette opération est implémentable par simple permutation d'éléments sans recourir à plusieurs tableaux*

2 Interlude : dessin dans une fenêtre

Le module `Tkinter` de Python permet de créer très facilement une interface graphique. Il existe d'autres bibliothèques pour créer des interfaces graphiques mais `Tkinter` a l'avantage d'être disponible par défaut, sans nécessiter une installation supplémentaire.

Cette librairie permet de créer des fenêtres avec des *widgets* divers (boutons, listes déroulantes, etc). Dans le cadre de ce TP nous ne nous intéresserons qu'au widget `Canvas` qui permet de dessiner des formes géométriques à l'écran.

La création d'une fenêtre contenant un unique widget `Canvas` se fait simplement à l'aide du code suivant :

```
from tkinter import *

#window creation
window = Tk()

#canvas creation
w = Canvas(window, width=200, height=100)
w.pack()

#main loop to allow interaction
mainloop()
```

Après l'appel à la fonction `pack()`, vous pouvez dessiner des formes géométriques dans le canvas comme indiqué dans le code ci-dessous :

```
#drawing functions
w.create_line(0, 0, 200, 100)
w.create_line(0, 100, 200, 0, fill="red")
w.create_rectangle(50, 25, 150, 75, fill="blue")
```

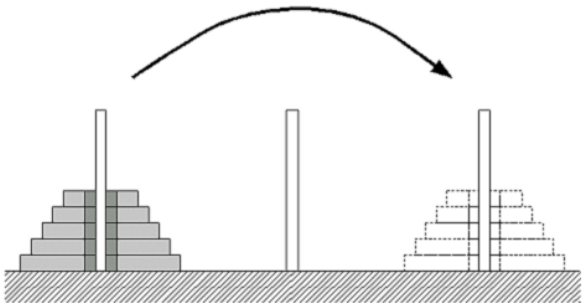
Le liste complète des figures géométriques disponibles n'est pas donnée ici, la documentation de *Tkinter* étant facilement disponible sur Internet. Cependant, il n'existe pas de fonction permettant de dessiner un point à l'écran dans ce module. Il peut cependant être émulé par la fonction suivante :

```
#function definition
def draw_point(canvas, x, y, color="black"):
    canvas.create_rectangle(x,y,x,y,fill=color,width=0)

#usage
draw_point(w,10,10)
```

3 Problèmes

Le jeu des Tours de Hanoï est constitué de trois piquets verticaux, notés 1, 2 et 3 et de n disques superposés de tailles strictement décroissantes avec un trou au centre et enfilés autour du piquet 1 ; ces disques forment les tours.



Le but du jeu consiste à déplacer l'ensemble des disques pour que ceux-ci se retrouvent enfilés autour du piquet 3 en respectant les règles suivantes :

— les disques sont déplacés un par un ;

— un disque ne doit pas se retrouver au-dessus d'un disque plus petit. (On suppose évidemment que cette dernière règle est également respectée dans la configuration de départ).

Le problème a toujours une solution en $2^n - 1$ coups (admis). Il se résout de manière récursive. En effet, supposons le problème résolu pour $n-1$ disques c'est à dire que l'on sache transférer $n-1$ disques depuis le piquet $i \in \{1,2,3\}$ jusqu'au piquet $j \in \{1,2,3\} \setminus \{i\}$ en respectant les règles du jeu. Pour transférer n disques du piquet i vers le piquet j, on procède alors comme suit :

— on amène les $n-1$ disques du haut du piquet i sur le piquet intermédiaire, qui a le numéro $6-i-j$;

— on prend le dernier disque du piquet i et on le met seul en j ;

— on ramène les $n-1$ disques de $6-i-j$ en j.

QUESTION 1. Écrire une procédure récursive `Hanoi` qui permet d'afficher dans une liste les mouvements élémentaires à accomplir pour déplacer n disques du piquet i au piquet j.

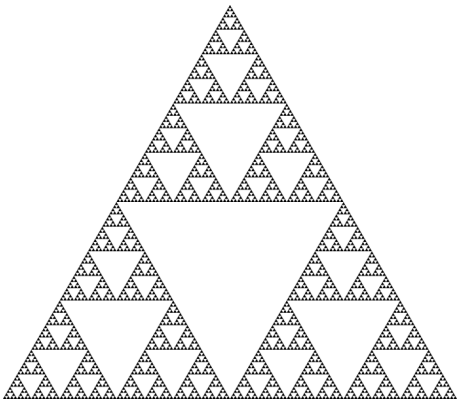
QUESTION 2. Faire un programme permettant l'affichage graphique (à l'aide de la fonction `print`) de chaque tour du jeu.

Exercice 6 : *Tracé d'un segment*

En informatique graphique, tous les points ont des coordonnées entières. Soit donc deux points A de coordonnées (x_A, y_A) et B de coordonnées (x_B, y_B). Proposez un alogrithme récursif pour tracer à l'écran le segment $[A, B]$.

Exercice 7 : *Triangle de Sierpiński*

Le triangle de Sierpiński est une fractale composée à partir de triangles illustré dans la figure suivante :



L'algorithme permettant d'obtenir cette figure consiste à tracer un triangle puis le triangle qui joint les milieux des cotés et ainsi de suite jusqu'à obtenir des segments plus petits qu'un pixel.

Écrire une fonction récursive permettant de dessiner ce triangle pour une taille de côté donnée.