TP Info 6 | Récursivité

Cette séance de TP constitue une initiation à la programmation récursive (au programme de seconde année). On y retrouvera les algorithmes classiques et quelques algorithmes de tri.

1. Ce qu'il faut savoir

Fonctions récursives

▷ Une fonction récursive est une fonction qui fait appel à elle-même.

▷ Très souvent un algorithme récursif est lié à une relation de récurrence permettant de calculer la valeur d'une fonction pour un argument n à l'aide des valeurs de cette fonction pour des arguments strictement inférieurs à n.

▷ Exemple : calcul par récurrence de x^n pour $x \in \mathbb{R}$ et $n \in \mathbb{N}$. Le programme en `Python` s'écrit :

```
>>> def puissance(x,n):
        if n=0:
           return(1)
        else:
           return(x*puissance(x,n-1))
```

2. Exercices

Les difficultés sont échelonnées de la manière suivante : aucune, ♪, ♪♪, ♪♪♪ et ♪♪♪♪. Certains énoncés sont tirés des annales des concours (oral et écrit) ; leur provenance est le plus souvent précisée. Les exercices notés ♪♪♪ et ♪♪♪♪ sont particulièrement délicats.

2.1. Versions récursives des algorithmes classiques

1 . [***Factorielle et suite de Fibonacci, versions récursives*** **]**
Programmer de manière récursive le calcul de :

a) la fonction factorielle ;

b) la suite de Fibonacci.

2 . [***Exponentiation rapide, version récursive*** **]**
Étant donnés un réel positif a et un entier n, remarquons que

$$a^n = \begin{cases} (a^n)^2 & \text{si } n \text{ est pair} \\ a\left(a^{\frac{n-1}{2}}\right)^2 & \text{si } n \text{ est impair} \end{cases}$$

a) En déduire une version récursive de l'algorithme d'exponentiation rapide.

b) Quel est le lien de cet algorithme avec la décomposition binaire de l'entier n ?

c) Calculer la complexité de cet algorithme et la comparer à celle de l'algorithme naïf.

3 . [*Division euclidienne et algorithme d'Euclide, versions récursives*]

a) En utilsant l'algorithme des soustractions successives, écrire une fonction récursive `deR(a,b)` qui retourne le reste de la division euclidienne de `a` par `b`.

b) En utilisant l'algorithme d'Euclide, écrire une fonction récursive `pgcdR(a,b)` qui retourne le plus grand commun diviseur de deux entiers naturels `a` et `b`.

4 . [*Coefficients binomiaux* ♪]

Écrire une fonction récursive `binome(n,k)` renvoyant le coefficient $\binom{n}{k}$.

5 . [*Décomposition binaire, version récursive* ♪]

Écrire une fonction récursive qui calcule la décomposition binaire d'un entier naturel n.

2.2. Algorithmes de tri

6 . [*Quicksort* ♪]

Quicksort is a *divide and conquer algorithm*. Quicksort first divides a large list into two smaller sub-lists : the low elements and the high elements. Quicksort can then recursively sort the sub-lists. The steps are :

- ▷ Pick an element, called a *pivot*, from the list ;
- ▷ Reorder the list so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final position. This is called the *partition* operation.
- ▷ Recursively apply the above steps to the sub-list of elements with smaller values and separately the sub-list of elements with greater values.

The base case of the recursion are lists of size zero or one, which never need to be sorted.

a) Write a function `quicksort(t)` sorting the list `t` by following this algorithm.

b) What is the complexity of the quicksort-algorithm ?

7 . [*Tri fusion* ♪♪]

L'algorithme du tri fusion est le suivant :

- ▷ On coupe en deux parties à peu près égales les données à trier ;
- ▷ On trie les données de chaque partie (pour cela, on coupe chaque partie en deux et on trie chacune) ;
- ▷ On fusionne les deux parties.

a) Écrire une fonction `fusion(A,B)` qui fusionne deux listes triées `A` et `B` en une troisième liste triée.

b) Écrire une fonction `triFusion(t)` renvoyant la liste `t` triée par l'algorithme de tri fusion.

2.3. Compléments

8 . [*Un testeur d'anagrammes* ♪]
Écrire une fonction récursives à valeurs booléennes `testeurAnagrammes(A,B)` qui teste si une chaîne de caractères `A` est un anagramme d'une chaîne de caractères `B`. Par exemple, *algorithme* est un anagramme de *logarithme*.

9 . [*Recherche dichotomique dans une liste triée, version récursive* ♪]
Écrire une fonction `rdR(a,t)` renvoyant `True` si `a` figure dans la liste triée `t` et `False` sinon. On utilisera une version récursive de l'algorithme de recherche dichotomique dans une liste triée.

10 . [*Introduction à la programmation dynamique* ♪♪]
On souhaite dénombrer le nombre de façons de décomposer une somme de n euros avec des pièces et des billets de 1, 2, 5 et 10 euros. On note $\mu_0 = 1$, $\mu_1 = 2$, $\mu_2 = 5$, $\mu_3 = 10$.

a) Notons $f(n,t)$ le nombre de façons de décomposer n euros avec des pièces et billets de valeur μ_k avec $k \leqslant t$.

   i) Que valent $f(n,0)$ et $f(0,t)$?

   ii) Justifier que $f(n,t) = f(n-1,t)$ pour $t \geqslant 1$ et $n < \mu_t$.

   iii) Pour $t \geqslant 1$ et $n \geqslant \mu_t$, trouver une expression de $f(n,t)$ en fonction des $f(i,j)$ où $i + j < n + t$.

b) Ecrire une fonction récursive qui calcule le nombre de façons de décomposer une somme de n euros avec des pièces et des billets de 1, 2, 5 et 10 euros.

11 . [*Les tours de Hanoï* ♪♪]
Sur une tablette sont disposées 3 tiges A, B, C.

Sur la tige A, on a empilé n disques de taille strictement décroissante. On veut déplacer cette pyramide vers la tige C en utilisant la règle suivante : on peut enlever un disque du sommet pour le mettre sur une autre tige à condition qu'il repose sur un disque de rayon plus grand.

a) Écrire une fonction récursive `hanoi(n,o,i,d)` d'arguments `n`, `o` (tour d'origine), `i` (tour intermédiaire) et `d` (tour de destination) renvoyant les mouvements à effectuer (sous la forme `De o en i`) pour effectuer le transfert des `n` disques de la tour `o` à la tour `d` en utilisant la tour `i` comme intermédiaire.

b) Dénombrer les manipulations effectuées par cette fonction.