TD 8 – Récursion (et comment l'éliminer)

Concepts informatiques (CI2)

2011-2012

1 Coupe en deux

Soit la fonction f définie par $f(0) = 1$, $f(n) = n.f(n/2)$ si n est pair et $f(n) = 1 + f((n-1)/2)$ sinon. Écrire une version récursive de f. Écrire ensuite une version itérative qui, à l'aide d'une pile, simule le comportement de la version récursive.

2 Coupe en deux (bis)

Considérons maintenant la fonction g définie par $g(0) = 1$, $g(n) = ng(n/2)$ si n est pair et $g(n) = g((n-1)/2)$ sinon. Écrire une version itérative sans pile. Pourquoi est-ce possible avec cette fonction, et non avec la fonction f ?

3 Tri rapide

Écrire une fonction `int separe(int t[],int p,int i, int j)` qui réordonne le sous-tableau de `t` compris entre les indices `i` et `j` de manière à ce que les éléments inférieurs strictement à `p` soient regroupés au début (et donc ceux supérieurs ou égaux à `p` regroupés à la fin). La fonction doit renvoyer en résultat l'indice de la première case où elle aura mis un élément supérieur ou égal à `p`.
On peut utiliser cette fonction pour trier un tableau :
- on choisit un pivot, par exemple `t[0]`
- à l'aide de `separe`, on regroupe au début du tableau les éléments plus petits que le pivot, et à la fin ceux qui lui sont supérieurs ou égaux
- on continue de même sur chacun des deux sous-tableaux.

Écrire une version récursive de cet algorithme, puis une version itérative. Pour cette dernière, il faudra maintenir une pile contenant la liste des sous-tableaux en attente de traitement.

4 Par où passer...

Un graphe orienté est un système de nœuds reliés par des flèches. On peut représenter un graphe orienté par un tableau de booléens a deux dimensions :
- les n nœuds du graphe sont numérotés de 0 à $n-1$
- la case (i, j) du tableau contient vrai si et seulement si il y a une flèche reliant le nœud d'indice i au nœud d'indice j.

On s'intéresse au problème consistant à chercher si il existe un chemin dans un graphe entre deux nœuds i et j donnés. On peut utiliser l'algorithme dit de « parcours en profondeur » :

– on se donne un tableau auxiliaire `connus` de booléens, dont toutes les cases sont initialement à `false`, et qui servira à se souvenir de l'ensemble des sommets que l'on a rencontrés
– on part du nœud i
– on met `connu[i]` à `true`
– on examine chacune des flèches sortant de i : pour chacune d'elle, en notant k son extrémité, on regarde `connu[k]`. Si cette case contient `true`, on ne fait rien. Sinon, on fait un appel récursif sur k.

Ce faisant, on explore tous les sommets du graphe que l'on peut atteindre à partir de i. Si l'on rencontre j au passage, alors on a gagné.

Écrire une méthode récursive implémentant cet algorithme.

Écrire une méthode itérative qui fait la même chose. Pour ce faire, il faut maintenir dans une pile le chemin que l'on a parcouru entre le sommet de départ et le sommet courant.