Travaux Dirigés n°3

Exercice 1

Ecrire l'algorithme qui compte le nombre de déplacements effectués par la tête de lecture/écriture d'une machine de Turing avant qu'elle ne s'arrête. On considère une machine de Turing à 5 états de 0 à 4 et un état particulier -1 qui arrête la machine. La machine peut lire ou écrire uniquement des 0 ou des 1 sur le ruban. On modélise la machine de Turing de la façon suivante

— Un tableau d'entiers `ruban` de taille 100 qui représente le ruban (normalement infini). `ruban` sera initialisé avec que des zéros.
— Un entier `pos` représentant la position du curseur sur le tableau `ruban`. `pos` sera initialisé à 50.
— Un entier `cstate` représentant l'état courant de la machine. `cstate` sera initalisé à 0.
— Un tableau d'entiers `write0` (resp. `write1`) de taille 5 qui donne la valeur à écrire sur le ruban de la machine dans le cas où on lit un 0 (resp. 1). Par exemple, si la machine lit un 0 sur le ruban, et que son état courant est 3, alors on effectue l'affectation suivante : `ruban[pos]=write1[3];`. Notez que les tableaux `write0` et `write1` ne contiennent que des 0 ou des 1.
— Un tableau d'entiers `depl0` (resp. `depl1`) de taille 5 qui donne le déplacement de la tête de lecture avec -1 pour un déplacement vers la gauche et +1 pour un déplacement vers la droite. Par exemple, si la machine lit un 1 sur le ruban et que son état courant est 4, alors, la position de la tête de lecture est modifiée de la façon suivante : `pos=pos+depl1[4]`.
— Un tableau d'entiers `nextstate0` (resp. `nextstate1`) de taille 5 qui donne la valeur du prochain état de la machine dans le cas où on lit un 0 (resp. 1). Par exemple, si la machine lit un 1 sur le ruban et que son état courant est 2, alors, le prochain état sera l'entier contenu dans `nextstate1[2]` (`cstate=nexstate1[2]`). Attention, il y a un élément du tableau `nextstate0` ou `nextstate1` qui est égal à -1 qui arrête la machine.

Pour tester l'algorithme, on prendra la machine de Turing suivante :

— `write0={1,1,1,1,1}`
— `write1={0,1,0,0,0}`
— `depl0={1,1,1,1,-1}`
— `depl1={-1,-1,-1,-1,1}`
— `nextstate0={1,2,3,4,3}`
— `nextstate1={-1,0,1,2,4}`

Exercice 2 - factoriel

Ecrire 2 algorithmes permettant de calculer $n!$. L'un des algorithmes sera itératif et l'autre récursif.

Exercice 3 - suite de Fibonacci

La suite de Fibonacci est définie de manière récursive par la relation : $u_n = u_{n-1} + u_{n-2}$. et par les conditions initiales : $u_0 = u_1 = 1$.
Construire un algorithme d'abord itératif, puis récursif permettant de calculer le n-ième terme.

Exercice 4 - Puissance k-ième iterative

Ecrire un algorithme prenant en entrée deux entiers n et k ($k > 0$) qui renvoie n^k. Il est bien entendu interdit d'utiliser une fonction ou un opérateur calculant directement cette puissance.

Exercice 5 - Puissance k-ième récursive

Pour calculer n^k (n,k, entiers positifs), on peut utiliser le principe de récursion suivant :
- si $k = 0$ alors, le résultat est 1.
- si $k = 2 \times p$ (k est pair), alors on calcule (selon le même principe) $m = n^p$ puis le résultat est $m \times m$.
- si $k = 2 \times p + 1$ (k est impair), alors on calcule (selon le même principe) $m = n^p$ puis le résultat est $n \times m \times m$.

1. Ecrire l'algorithme récursif `puiss` qui prend en entrée deux entiers positifs n et k et renvoie n^k selon le principe décrit ci-dessus.
2. Dans le cas du calcul de n^{18} (n positif), combien d'appels seront fait à l'algorithme `puiss` ?
3. Dans le calcul de n^k avec $k = 2^p$, combien d'appels seront fait à l'algotithme `puiss` (la réponse dépend bien sûr de p).

Exercice 6

Le problème des tours de Hanoï est un jeu de réflexion imaginé par le mathématicien français Édouard Lucas, et consistant à déplacer des disques de diamètres différents d'une tour de « départ » à une tour d'« arrivée » en passant par une tour « intermédiaire » et ceci en un minimum de coups, tout en respectant les règles suivantes :
- on ne peut déplacer plus d'un disque à la fois,
- on ne peut placer un disque que sur un autre disque plus grand que lui ou sur un emplacement vide.

On suppose que cette dernière règle est également respectée dans la configuration de départ. On suppose que les trois positions des trois tours sont numérotées 1, 2 et 3.

Ecrire un algorithme qui prend en entrée n le nombre de disque sur la tour, dep la position de la tour de départ, med la position de la tour intermédiaire, et fin la position finale. Cet algorithme ne renvoie rien mais affiche les déplacement successifs sous forme d'un couple d'entier (a, b) qui signifie qu'on déplace le disque au dessus de la tour a sur la tour b.

Exemple : les mouvements nécessaires pour déplacer 3 disques de 1 vers 3 en passant par 2 :
- (1,3)
- (1,2)
- (3,2)
- (1,3)
- (2,1)
- (2,3)
- (1,3)