DS 2

Option informatique, première année

Julien Reichert

Exercice 1 : Écrire en Caml une fonction qui détermine le minimum d'un tableau. Donner aussi sa signature.

Exercice 2 : Même exercice mais cette fois-ci avec une liste.

Exercice 3 : On souhaite calculer le maximum de la somme d'une zone carrée d'une matrice de flottants (`float vect vect`), et la complexité sera un élément important dans l'écriture du programme. Les nombres considérés seront tous positifs ou nuls. Dans un premier temps, on cherchera donc la plus grande somme d'un élément de sa matrice et de ses voisins (diagonale comprise).

Question 3.0 : Justifier qu'on n'a pas besoin de se poser la question de la gestion des bords de la matrice au moment de faire les tests.

On exclut dans un premier temps les cas dégénérés et on suppose que les matrices étudiées auront toujours au moins trois lignes et trois colonnes.

Question 3.1 : Commencer par écrire une fonction qui localise la plus grande somme de trois éléments successifs d'un tableau de flottants. La valeur de retour doit être l'indice de la position centrale.

Question 3.2 : Écrire une fonction qui calcule la somme des éléments du carré centré en la position `(i,j)` d'une matrice de flottants, la position `(i,j)` n'étant pas sur le bord de la matrice.
Le type doit être `float vect vect -> int -> int -> float`.

Question 3.3 : À l'aide de la fonction précédente, écrire une fonction qui localise la plus grande somme d'une zone carrée de côté trois dans une matrice.
Le type doit être `float vect vect -> (int * int)`.

Désormais, plutôt que de chercher la plus grande somme d'une zone, on va récupérer des sommes à des positions quelconques. On admet que la matrice étudiée est une variable globale appelée `mat` et qu'elle n'est pas un paramètre de la fonction qu'on attend à la question 3.5.

Question 3.4 : On se place dans un contexte où le nombre d'appels la fonction écrite à la question 3.2 est très grand (bien au-delà du nombre total d'éléments dans la matrice) et où les éléments de la matrice ne sont jamais modifiés. Quelle solution pratique peut-on proposer ? (Penser à la section sur la programmation dynamique.)

Question 3.5 : Écrire un programme qui implémente la solution proposée dans l'exercice précédent.

Question 3.6 : Étendre le programme de la question 3.3 au cas où la zone carrée ne contient pas forcément 3x3 cases mais au plus (2n+1)x(2n+1) éléments, où la valeur `n` est un paramètre. Cette fois-ci, le nombre de lignes de la matrice ou son nombre de colonnes peuvent être inférieurs à 2n+1.

Exercice 4 : Le but de cet exercice est d'implémenter une structure d'arbre binaire à l'aide d'un tableau. Les nœuds internes de l'arbre seront d'un certain type et les feuilles d'un type qui peut être différent. L'implémentation proposée sera particulièrement utile pour le projet de programmation « codage de Huffman ».

L'idée de base est que la racine sera à la position zéro du tableau, et tous les nœuds seront munis de trois informations : la position dans le tableau du fils gauche, la valeur associée au nœud et la position du fils droit dans le tableau. Il suffira donc de suivre les indices pointés pour descendre une branche de l'arbre.

Cette implémentation était utilisée dans le sujet d'option informatique du concours Centrale en 2013, associée à des formules logiques.

Question 4.0 : Expliquer les lignes ci-après imposées pour la suite de l'exercice.

```
type ('a,'b) arbre_bin = Vide | Feuille of 'a
| Noeud of ('a,'b) arbre_bin * 'b * ('a,'b) arbre_bin;;

type ('a,'b) elt_arbre_bin_tab = Rien | Pos_feuille of 'a | Pos_noeud of int * 'b * int;;

type ('a,'b) arbre_bin_tab == ('a,'b) elt_arbre_bin_tab vect;;
```

Question 4.1 : Écrire une fonction qui initialise un tableau représentant l'arbre binaire vide et ayant de la place pour `n` feuilles, où la valeur `n` est un paramètre. Autant le dire tout de suite, il y a un piège.

Question 4.2 : Écrire une fonction qui transforme un arbre binaire en un tableau qui le représente.
Le type doit être `('a,'b) arbre_bin -> ('a,'b) arbre_bin_tab`. [1]

L'exercice ci-avant est relativement indépendant du reste. Il est également plus difficile que les autres.

Question 4.3 : Écrire une fonction d'ajout d'une feuille à une certaine position dans la représentation d'un arbre binaire. La position est fournie en tant que liste d'entiers, l'entier zéro indiquant qu'on envoie la feuille sur le fils gauche du nœud en question et l'entier un indiquant qu'on envoie la feuille sur le fils droit. Une fois la liste parcourue, il faut qu'on tombe sur une case contenant `Rien` dans le tableau. Donner aussi la signature de la fonction.

Question 4.4 : Écrire une fonction d'ajout d'un nœud à une certaine position dans la représentation d'un arbre binaire. La position est fournie comme dans la question précédente. Une fois la liste parcourue, si la case du tableau atteinte ne contient pas `Rien`, le contenu devient le fils gauche du nouveau nœud. Donner aussi la signature de la fonction.

Question 4.5 : Écrire une fonction de fusion de deux représentations d'arbres binaires. La fonction crée un nouvel arbre dont la racine est fournie en paramètre, le fils gauche de la racine est le premier arbre et le fils droit est le second.
Le type doit être `'a -> ('b,'a) arbre_bin_tab -> ('b,'a) arbre_bin_tab -> ('b,'a) arbre_bin_tab` [2].

1. Mais Caml ne tiendra pas compte de ce dernier type obtenu par renommage.
2. Ordre alphabétique oblige...