TP 9 : Arbres Binaires

14 février 2024

Exercice 1

On considère, en C et en OCaml, le type `bintree` défini dans le cours.

Définir une fonction `bintree_create` en C, qui prend en entrée un pointeur vers un arbre gauche `g`, un pointeur vers un arbre droit `d` et un entier `v`, et construit l'arbre à partir du noeud d'étiquette `v` et dont les sous-arbres gauches et droits sont les arbres pointés par `g` et `d`.

Exercice 2

Définir, en C et en OCaml, une fonction `taille` qui prend en entrée un arbre et renvoie son nombre de noeuds.

Exercice 3

Définir, en C et en OCaml, une fonction `parfait` qui prend en entrée une hauteur `h` et renvoie l'arbre binaire parfait de hauteur `h`.

Exercice 4

Définir, en C et en OCaml, une fonction `peigne_gauche` qui prend en entrée un entier `n` et renvoie l'arbre peigne gauche à n noeuds (arbre de hauteur $n-1$ tel que chaque noeud a au plus un fils, placé à gauche). Les étiquettes sont les entiers de 0 à $n-1$.

Exercice 5

Définir, en C et en OCaml, une fonction `complet` qui prend en entrée un entier `n` et renvoie l'arbre complet à `n` noeuds.

Exercice 6

Définir une fonction `maximum` (respectivement `minimum`) qui prend en entrée un arbre dont les étiquettes des noeuds sont des entiers, et qui renvoie le plus grand (respectivement plus petit) élément étiqueté dans un noeud de l'arbre (et échoue avec une exception si l'arbre est vide).

Exercice 7

1) Définir un type `direction` ayant comme constructeurs `Gauche` et `Droite`.

2) Ecrire une fonction qui prend en entrée un arbre, une liste de directions et renvoie l'étiquette au noeud atteint en parcourant l'arbre en profondeur selon les directions indiquées dans la liste en entrée. L'algorithme renvoie la valeur à la racine si la liste est vide, et échoue si on arrive sur un sous-arbre vide.

Exercice 8

1) Ecrire une fonction `miroir` qui prend en entrée un arbre binaire et renvoie l'arbre binaire miroir (obtenu en intervertissant fils gauche et fils droit dans chaque sous-arbre).

2) Ecrire une fonction `symetrique` qui vérifie si un arbre binaire donné en entrée est symétrique (égal à son miroir).

Exercice 9

On définit de façon inductive une expression arithmétique sur les entiers avec les opérations d'addition, soustraction et multiplication de la façon suivante. Une telle expression est :

- Soit un entier ;
- Soit l'addition de deux expressions arithmétiques ;
- Soit la soustraction de deux expressions arithmétiques ;
- Soit la multiplication de deux pressions arithmétiques.

1) Définir une type somme `expression` permettant de représenter cette définition induction des expressions arithmétiques.

On souhaite maintenant représenter de façon arborescente une telle expression arithmétique.

2) Définir un type `operation` avec comme constructeurs `Plus`, `Moins` et `Fois`.

3) Définir un type `symbole` avec un constructeur prenant comme argument un entier, et un autre constructeur prenant comme argument une opération.

4) Définir une fonction `enum_to_tree` qui prend en entrée une expression et renvoie l'arbre binaire correspondant, dont les noeuds internes sont des symboles d'opérations et les feuilles sont des symboles d'entiers. Définir réciproquement la fonction `tree_to_enum`.