Révisions arbres

Ex. 1 – Arbres génériques

On définit des arbres étiquetés en OCaml avec le type suivant :

```
type 'a arbre =
  N of 'a * 'a arbre list
```

1. Proposer une fonction déterminant la taille de l'arbre.

2. Proposer une fonction déterminant la hauteur de l'arbre.

3. Proposer une fonction déterminant l'arité de l'arbre.

4. Proposer une fonction déterminant le plus grand élément de l'arbre (on suppose les étiquettes comparables avec les opérateurs de comparaison usuels).

5. Proposer une fonction renvoyant la liste des éléments de l'arbre dans un parcours en profondeur suffixe.

Ex. 2 – Numérotation et nombre de Strahler d'un arbre binaire strict

On considère des arbres binaires stricts étiquetés par des objets de type `'a`, représentés en OCaml par le type suivant :

```
type 'a arbre =
  | F of 'a
  | N of 'a * 'a arbre * 'a arbre
```

1. Proposer une fonction `numerote` dont la signature OCaml serait `'a arbre -> (int * 'a) arbre`, prenant en argument un arbre binaire strict étiqueté et créant un nouvel arbre dans lequel, à chaque élément (nœud ou feuille) de l'arbre fourni en paramètre, on adjoint un entier entre 1 et la taille de l'arbre, correspondant à l'ordre de visite pour un parcours en profondeur infixe de l'arbre, les sous-arbres étant visités de gauche à droite (on débutera la numérotation à 0).

2. Faire de même avec une numérotation dans l'ordre d'un parcours en largeur (hiérarchique).

On définit le *nombre de Strahler* d'un arbre binaire strict par induction structurelle :

- le nombre de Strahler d'une feuille est 1 ;
- pour un arbre non réduit à une feuille dont les deux sous-arbres ont le même nombre de Strahler s, le nombre de Strahler de l'arbre est $s+1$;
- pour un arbre non réduit à une feuille dont les deux sous-arbres ont les nombres de Strahler s et s' avec $s < s'$, le nombre de Strahler est s'.

3. Déterminer les valeurs possibles pour le nombre de Strahler d'un arbre de hauteur h (on précisera un exemple pour les valeurs extrêmes)

4. Proposer une fonction prenant en argument un arbre binaire strict et renvoyant son nombre de Strahler.

Ex. 3 – Arbres binaires de recherche

On considère un arbre binaire de recherche A étiqueté par $\mathcal{E}$, représenté en OCaml par le type

```
type 'a abr =
  | Nil
  | N of 'a arbre * a * 'a arbre
```

1. Quelles complexités en temps, dans le pire des cas (exprimée en fonction de $h(A)$ et $|A|$), peut-on espérer pour des fonctions réalisant les opérations suivantes :

- vérifier la présence d'un élément x dans A ;
- compter les occurrences d'un élément x dans A ;
- déterminer le cinquième plus petit élément de A ;
- déterminer la médiane de l'ensemble représenté par A.

2. Comment peut-on modifier la structure d'un arbre binaire de recherche pour pouvoir déterminer en $O(h(A))$ l'obtention du « k^e » plus petit élément ou de la médiane d'un ensemble représenté par un arbre binaire de recherche A équilibré ?

3. Proposer une fonction `nieme` qui prend en argument un entier k et un tel arbre et renvoie, en $O(h(A))$, l'élément qui se trouverait dans la case d'index k dans un tableau où les éléments auraient été ordonné par ordre croissant. En déduire une fonction `mediane` renvoyant la médiane des éléments de l'arbre en $O(h(A))$.

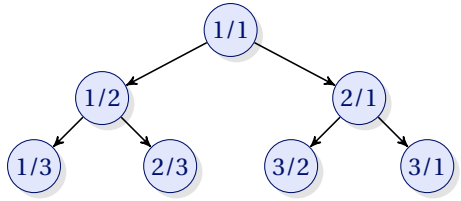
4. Proposer une fonction `insere` travaillant sur un tel arbre.

Ex. 4 – Arbre de Stern-Brocot

Un arbre de Stern-Brocot est un arbre binaire de recherche étiqueté avec des fractions irréductibles, chaque fraction n'apparaissant qu'une seule fois. Dans sa description mathématique, il s'agit d'un arbre infini dans lequel tout élément de $\mathbb{Q}$ figure dans l'arbre (en ce sens, il est similaire aux arbres de Calkin-Wilf, avec un placement différent des éléments). Il a été décrit indépendamment par le mathématicien allemand Moritz Stern (en 1858) et l'horloger français Achille Brocot (en 1861).

Il s'agit d'un arbre binaire strict, dont la racine porte l'étiquette 1. On ne peut évidemment construire intégralement un arbre infini, mais on peut s'intéresser à l'arbre binaire

strict complet regroupant les éléments dont la profondeur est inférieure ou égale à h. Par exemple, voici l'arbre de Stern-Brocot restreint aux éléments de profondeur inférieure ou égale à 2 :



On peut construire un arbre restreint aux éléments de profondeur inférieure ou égale à h de la façon suivante :

- on construit l'arbre restreint aux éléments à une profondeur inférieure ou égale à $h-1$;
- pour toute feuille à une profondeur h, on détermine les étiquettes u/v et u'/v' qui la précède (respectivement la succède) immédiatement dans un parcours en profondeur infixe de l'arbre, et on lui attribue l'étiquette $(u+u')/(v+v')$. Pour la feuille la plus à gauche, on choisira $u=0$ et $v=1$, et pour la plus à droite, $u'=1$ et $v'=0$.

1. Construire l'arbre restreint aux éléments de profondeur inférieure ou égale à 3 en déterminant les huit fractions devant étiqueter les huit nouvelles feuilles.

2. Quels sont les plus petits et plus grands éléments contenus dans l'arbre de Stern-Brocot à une profondeur h ?

3. Montrer que si x est une étiquette d'un nœud de l'arbre de Stern-Brocot à une profondeur h, alors $1/x$ est également une étiquette d'un nœud de l'arbre de Stern-Brocot à une profondeur h.

On définit le type suivant :

```
type arbre_stern_brocot =
  | Nil
  | N of arbre_stern_brocot * int * int * arbre_stern_brocot
```

4. Proposer une fonction de signature `int -> arbre_stern_brocot` prenant en argument une profondeur h et renvoyant l'arbre de Stern-Brocot restreint aux éléments à une profondeur inférieure ou égale à h.

Indication : il peut être plus malin de construire itérativement l'ensemble des étiquettes dans une liste (où elles figurent dans un ordre correspondant au parcours infixe des arbres de hauteur croissante), avec les deux éléments supplémentaires $0/1$ et « $1/0$ » aux deux extrémités, et de ne construire l'arbre qu'une fois toutes les étiquettes déterminées !

La première moitié des listes ainsi construite est appelée suite de Farey, en hommage au géologue britannique Sir John Farey qui a posé une conjecture concernant cette suite, ultérieurement prouvée par Augustin Cauchy, même si le mathématicien Charles Haros est le premier à avoir étudié cette suite.

Ex. 5 – Coupure et échanges d'éléments

On considère des arbres binaires de recherche, représentés en OCaml par le type

```
type 'a abr =
  | Nil
  | N of 'a arbre * a * 'a arbre
```

1. Proposer une fonction OCaml prenant en argument un arbre binaire de recherche A et un élément x et retournant deux arbes binaires de recherche contenant une partition des étiquettes de A : les étiquettes inférieures à x dans le premier et les autres dans le second.

On considère un arbre binaire de recherche A contenant au moins deux éléments. On suppose toutes les étiquettes dans cet arbre distinctes. On sélectionne au hasard deux sommets de l'arbre et on échange leurs étiquettes x et y.

2. Proposer une fonction OCaml prenant en argument un tel arbre et retournant les étiquettes x et y ayant été échangées en un temps linéaire en la taille de l'arbre A.

3. Qu'en est-il si l'on retire la condition que toutes les étiquettes sont distinctes ?