TD n° 6 : Arbres

Dans ce TD, on considère le type `('f, 'n) arbre` suivant, pour lequel il est possible d'étiqueter les feuilles et les nœuds par des éléments de types différents :

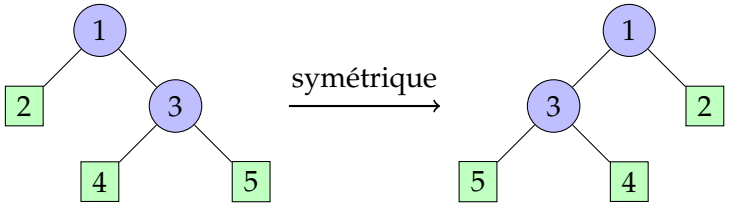
OCaml
```
type ('f, 'n) arbre =
  | Feuille of 'f
  | NoeudInterne of ('f, 'n) arbre * 'n * ('f, 'n) arbre
```

EXERCICE 1 *Pour bien commencer*

Définir en CAML, avec ce type, l'arbre de gauche de l'exercice 2.

EXERCICE 2 *Symétrisé d'un arbre*

Écrire une fonction `symetrique` de type `('f, 'n) arbre -> ('f, 'n) arbre` qui renvoie l'arbre symétrique. Par exemple :



EXERCICE 3

1. Écrire une fonction `initialise_complet : int -> 'f -> 'n -> ('f, 'n) arbre` telle que `initialise_complet h val_f val_n` créé un arbre complet de hauteur $h \geq 0$ initialisé avec les valeurs `val_f` pour les feuilles et `val_n` pour les nœuds internes.
2. On appelle *cheminement* d'un arbre la somme des profondeurs de chacune de ses feuilles. Que vaut le cheminement d'un arbre complet de hauteur h ?

EXERCICE 4 *Nombre de Strahler*

Le nombre de Strahler[1] *d'un arbre est une mesure de sa complexité de branchement. Il est utilisé par exemple pour indiquer le niveau de complexité du réseau d'affluents de fleuves et en théorie de la compilation : lors de la compilation d'un programme, le nombre minimum de registres nécessaires pour évaluer l'arbre d'une expression est le nombre de Strahler de cet arbre.*

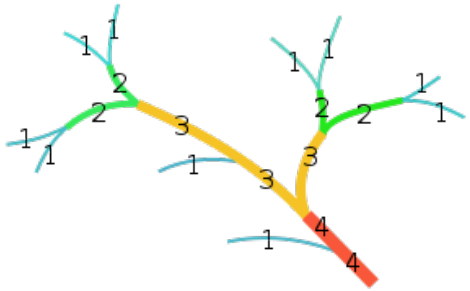


FIGURE 1 – Diagramme montrant le système de classement des cours d'eau de Strahler.

Le nombre de Strahler pour un arbre binaire strict est défini récursivement par :

- Le nombre de Strahler d'une feuille est 1 ;
- Pour un nœud interne dont les fils ont pour nombres de Strahler s_g et s_d : si $s_g = s_d$ alors le nœud a pour nombre de Strahler $s_g + 1$, sinon ce nombre est $\max(s_g, s_d)$.

1. Arthur Newell Strahler (1918-2002) est un professeur américain qui a développé en 1952 un classement des cours d'eau en fonction de la puissance de leurs affluents selon le nombre de Strahler.

1. Écrire une fonction `strahler : ('f, 'n) arbre -> int` qui calcule le nombre de Strahler d'un arbre.
2. Montrer qu'un arbre de hauteur h a un nombre de Strahler inférieur ou égal à $h+1$ avec égalité si et seulement si il est complet.
3. Montrer qu'un arbre de hauteur $h > 0$ a un nombre de Strahler supérieur ou égal à 2 avec égalité si et seulement si tous les nœuds internes à l'exception d'un seul possèdent une feuille et un nœud interne pour fils.

EXERCICE 5 *Hauteur d'un arbre binaire en fonction du nombre de nœuds*

Dans cet exercice on choisit la définition avec arbre vide et non celle d'arbre binaire strict. On note h la hauteur et n le nombre de nœuds (internes et externes) d'un arbre.

1. Montrer que l'on a toujours $\lfloor \log_2 n \rfloor \leq h \leq n-1$.
2. Donner des exemples d'arbres qui atteignent ces deux bornes.

Un arbre est dit *quasi-complet* si tous ses niveaux sont complets (entièrement remplis) sauf éventuellement le dernier.

3. Montrer que pour un arbre quasi-complet on a $h = \lfloor \log_2 n \rfloor$.

Le *facteur d'équilibrage* d'un nœud est la différence entre la hauteur de son sous-arbre gauche et celle de son sous-arbre droit. Un nœud dont le facteur d'équilibrage est -1, 0 ou 1 est considéré comme *équilibré*. Un arbre est équilibré si tous ses nœuds le sont.

4. Pour $h \in \mathbb{N}$, on note N_h le nombre minimal de nœuds d'un arbre équilibré de hauteur h. Montrer que $N_h = \mathcal{F}_{h+2} - 1$ où $\mathcal{F}_i$ est le i-ème nombre de Fibonacci.
5. En déduire que pour un arbre équilibré $h = \Theta(\log n)$.

EXERCICE 6 *Reconstruction d'arbres à partir de leurs parcours*

On définit le type suivant, qui permettra, lors du parcours d'un arbre, de se souvenir si le nœud était une feuille ou un nœud interne :

OCAML
```
type ('f, 'n) noeud = F of 'f | NI of 'n;;
```

1. Quelle est la différence entre `NoeudInterne`, `noeud` et `NI` ?

On va chercher à écrire les différentes variantes de parcours en profondeur `parcours_prefixe`, `parcours_infixe` et `parcours_suffixe`, dont le type sera `('f, 'n) arbre -> ('f, 'n) noeud list`.

2. Donner le résultat des trois parcours sur les deux arbres de l'exercice 2, sous la forme d'une liste CAML `('f, 'n) noeud list`, tels qu'ils seraient obtenus par un appel aux fonctions de parcours préfixe, infixe et suffixe.
3. Écrire la fonction `parcours_prefixe : ('f, 'n) arbre -> ('f, 'n) noeud list`. On pourra utiliser l'opérateur `@`.
4. Quelle est la complexité de cette fonction dans le pire cas en fonction du nombre de nœuds dans l'arbre ?
5. Expliquer ce qu'il faut modifier dans cette fonction pour obtenir les fonctions `parcours_infixe` et `parcours_suffixe`.
6. En utilisant un accumulateur, réécrire la fonction `parcours_prefixe` en garantissant une complexité linéaire en la taille de l'arbre. La fonction auxiliaire utilisée est-elle récursive terminale ?
7. On considère la liste `[NI 1; NI 2; F 4; F 5; F 4]` que l'on suppose obtenue par un parcours préfixe d'un arbre. Quel était cet arbre ?

On admet la proposition suivante :

Proposition 0.1

La fonction qui à un arbre associe la liste de nœuds correspondant à son parcours préfixe (respectivement suffixe) est injective.

8. Écrire une fonction `reconstruire_prefixe` de type `('f, 'n) noeud list -> ('f, 'n) arbre` qui à une liste de nœuds représentant un parcours préfixe associe l'unique arbre binaire correspondant.
9. Montrer que cette propriété est fausse pour le parcours infixe.