TP de *Logique et Circuits* n° 7

# Arbres binaires de recherche

On représente les arbres binaires (de type I) étiquetés par des entiers par le type Caml

```
type arbre = Av | Arb of int * arbre * arbre ;;
```

**Exercice 1**

1. Écrire une fonction `ajout_abr` telle que `ajout_abr` $n$ $a$ ajoute l'étiquette $n$ à l'arbre $a$. Lorsque $a$ est un ABR, le résultat doit être encore un ABR.
2. Écrire une fonction `rech_abr`, telle que `rech_abr` $n$ $a$ détermine, lorsque $a$ est un ABR, si $n$ apparaît dans $a$.

**Exercice 2**

1. Écrire une fonction `list_of_arbre` telle que `list_of_arbre` $a$ renvoie la liste des étiquettes de $a$, et telle que lorsque $a$ est un ABR, cette liste soit triée.
2. Écrire une fonction `arbre_of_list` telle que `arbre_of_list` $l$ renvoie un arbre étiqueté par les éléments de $l$. Lorsque $l$ est triée, le résultat doit être un ABR, équilibré au sens suivant :

   > *Dans tout sous-arbre de la forme* `Arb`$(n, g, d)$*, les sous-arbres* $g$ *et* $d$ *contiennent le même nombre d'étiquettes, à (au plus) une unité près.*

   On écrira les fonctions intermédiaires suivantes :
   - `diviser` telle que `diviser` $l$ $n$ renvoie un couple $(l_1, l_2)$ où $l_1$ contient les $n$ premiers éléments de $l$, et $l_2$ les éléments suivants.
   - `separer` telle que `separer` $l$ renvoie un couple $(l_1, l_2)$ tel que $l_1$ `@` $l_2 = l$, chaque liste contenant la moitié des éléments de $l$. Lorsque $l$ est de longueur impaire, $l_2$ contiendra un élément de plus.

**Exercice 3**

1. Écrire une fonction partielle `min_abr`, telle que `min_abr` $a$ renvoie, lorsque $a$ est un ABR non vide, le minimum de $a$.
2. Écrire une fonction partielle `max_abr`, telle que `max_abr` $a$ renvoie, lorsque $a$ est un ABR non vide, le maximum de $a$.
3. À partir de ces deux fonctions, écrire une fonction `est_abr` qui détermine si un arbre quelconque est un ABR.