# Arbre binaire de recherche avec Coq

Polytech Nice-Sophia
Département informatique
Preuves de programmes
Sylvain Lippi

On souhaite implanter directement en Coq les arbres binaires de recherches : tous les noeuds internes *ie* différents d'une feuille contiennent un entier et cet entier est supérieur ou égal (resp. strictement inférieur) aux entiers contenus dans le sous-arbre gauche (resp. le sous-arbre droit).

Dans ce qui suit, on ne considère donc que des arbres binaires.

## 1 Introduction avec des arbres binaires

**Exercice 1 (tree)** *Définir le type inductif* `tree` *comme étant une feuille (sans aucune valeur) ou un noeud (interne) construit avec un entier, un premier arbre (le sous-gauche) et un deuxième arbre (le droit).*

**Exercice 2 (small-prop)**

1. *Ecrire une fonction qui donne le nombre de noeuds internes.*
2. *Ecrire une fonction qui donne le nombre de feuilles.*
3. *Prouver que dans tout arbre (binaire), le nombre de feuilles est égal au nombre de noeuds internes + 1.*

**Exercice 3 (max)**

1. *Ecrire une fonction* `max` *prenant deux entiers et qui rend le plus grand.*
2. *Prouver que l'on peut peut échanger ses deux arguments :*
   $\forall p\ q : \texttt{nat},\ \texttt{max}(p, q) = \texttt{max}(q, p)$.
3. *Prouver :* $\forall p\ q : \texttt{nat},\ p \leq \texttt{max}(p, q)$.
4. *Prouver :* $\forall p\ q : \texttt{max}(p, q) \leq p + q$.
5. *Ecrire une fonction* `height` *qui rend la hauteur d'un arbre binaire ; on suppose que la hauteur est nulle s'il y a une seule feuille.*
6. *Prouver que la hauteur d'un arbre est inférieure ou égale à son nombre de noeuds internes.*
7. *Ecrire un prédicat* `isin` *exprimant qu'un entier donné appartient appartient à un arbre donné.*

## 2 Les arbres binaires de recherche

**Exercice 4** (`searchtree`)

1. *Ecrire un prédicat prenant un entier et un arbre et qui exprime que cet entier est supérieur ou égal à tous les éléments de l'arbre donné.*
2. *Ecrire un prédicat prenant un entier et un arbre et qui exprime que cet entier est strictement supérieur à tous les éléments de l'arbre donné.*
3. *En déduire un prédicat prenant un arbre binaire quelconque et qui exprime que cet arbre est un arbre de recherche.*

**Exercice 5 (recherche)**

1. *Ecrire une fonction* `search` *qui cherche un entier dans un arbre binaire de recherche et répond vrai ou faux selon le résultat de la recherche.*
2. *Prouver que cette fonction est correcte vis-à-vis du prédicat* `isin`*.*

*Indication 1.* On n'a pas défini le type «arbre binaire de recherche» ; on doit penser à spécifier que l'argument de `search` doit vérifier le prédicat `searchtree`. On appelle de tels prédicats des pré-conditions.

*Indication 2.* On pourra utiliser la commande `SearchAbout le` pour afficher tous les théorèmes disponibles concernant les inégalités.

**Exercice 6 (insertion)**

1. *Ecrire une fonction* `insert` *qui insère un entier donné dans un arbre de recherche.*
2. *Exprimer la correction de la fonction* `insert`*; informellement, elle doit préserver la structure d'arbre binaire de recherche et doit donner un arbre contenant le nouvel entier inséré. On pourra utiliser le prédicat* `isin` *pour exprimer la seconde propriété.*
3. *En faire la preuve. On pourra prouver séparément les deux propriétés énoncées précédemment.*