Arbres binaires de recherche

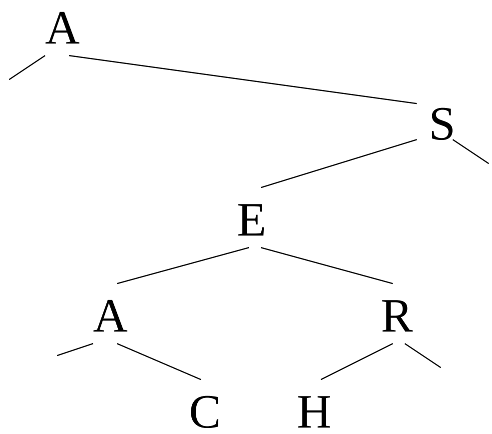
Les arbres binaires de recherche constituent un moyen courant de stockage et de recherche (pour représenter des dictionnaires par exemple). Les algorithmes impliqués sont très classiques et se retrouvent dans de nombreuses situations (parcours d'un arbre,

On définit un arbre comme étant soit une feuille (vide), soit un nœud constitué d'une information (par exemple une chaîne de caractères) et de deux sous-arbres, c'est-à-dire le type CAML suivant :

```
type 'a arbre = Feuille
              | Noeud   of 'a arbre * 'a * 'a arbre
```

où `'a` est le type de l'information. La propriété d'un arbre binaire de recherche est la suivante : *pour chaque nœud* (g, i, d)*, les informations contenues dans le sous-arbre gauche* g *sont inférieures (ou égales) à* i*, les informations contenues dans le sous-arbre droit* d *supérieures (ou égales) à* i*.*

Par exemple, l'arbre suivant est un arbre binaire de recherche :



On se propose d'écrire les principales fonctions de manipulation des arbres binaires de recherche, à savoir *la recherche*, *l'insertion* et *la suppression* dans un arbre binaire.

1 Recherche

Ecrire une fonction `recherche : 'a arbre -> 'a -> bool` testant la présence d'un élément dans un arbre binaire de recherche.

Quelle est sa complexité en fonction de la taille N de l'arbre (nombre de nœuds de l'arbre) ? Distinguer les cas en moyenne et au pire. A quel type d'arbre correspond la complexité la plus médiocre ?

2 Insertion

Ecrire une fonction `insertion : 'a arbre -> 'a -> 'a arbre` réalisant l'insertion d'un nouvel élément dans un arbre binaire de recherche. Quelle est la complexité de cette opération ?

En déduire une fonction `construit : 'a list -> 'a arbre` construisant un arbre binaire de recherche à partir d'une liste d'éléments.

3 Parcours

Ecrire une fonction `parcours : 'a arbre -> 'a list` prenant un arbre binaire de recherche en argument et renvoyant la liste *triée* de ses éléments. Un tel parcours s'appelle *parcours préfixe* d'un arbre.

4 Suppression

Ecrire une fonction `suppression : 'a arbre -> 'a -> 'a arbre` supprimant un élément dans un arbre binaire de recherche. Quelle est la complexité de cette opération ?