Arbres binaires de recherche

EXERCICE 1 *Les ABR en Caml*

Les ABR seront définis par la déclaration
`type 'a arbre=V | N of 'a*('a arbre)*('a arbre);;` suivie de :
`type ABR==int arbre;;` On utilisera les fonctions de dessins fournies dans le TD précédent.

On demande d'écrire les fonctions suivantes :

1. `ferif_abr` vérifie si un arbre binaire est un ABR.
   Tester sur de vrais ABR puis des faux bien choisis.
2. `cherche` vérifie si un entier x est dans un ABR. Le résultat n'a pas à être correct si l'arbre n'est pas un ABR. Faire des expérimentations "probantes".
3. `insert1,insert2` insèrent un entier dans un ABR, respectivement aux feuilles et à la racine.
   Pour `insert2 x`, on pourra commencer par écrire une fonction prenant en entrée un entier x un arbre et retournant le couple (g, d) correspondant à la coupure de l'arbre suivant x.
4. `build1,build2` créent un ABR à partir d'une liste d'entiers, en utilisant respectivement `insert1` et `insert2`. Comparer les résultats pour la liste `l1=[18;25;14;13;19;12;21;22;15]`.
5. `suppression` supprime un élément dans un ABR (si cet élément n'est pas présent, `suppression` retourne l'arbre initial. Faire des essais !

EXERCICE 2 *Une propriété (?) des ABR*

Le professeur G. pense avoir découvert la propriété suivante concernant les ABR : on suppose que la recherche d'une étiquette dans un ABR se termine sur une feuille. On note respectivement A, B et C l'ensemble des étiquettes situées à gauche (resp. le long et à droite) du chemin de recherche.

Le professeur G. pense que si $a \in A$, $b \in B$ et $c \in C$, alors nécessairement $a \leq b \leq c$. Qu'en pensez-vous ? (preuve ou contre-exemple **minimal**)

EXERCICE 3 *Séquences de recherche dans un ABR*

1. Les séquences suivantes peuvent-elles correspondre à la suite des étiquettes des nœuds rencontrés dans un ABR lors de la recherche de 2048 ?
   (a) 100,140,4096,150,3000,2048
   (b) 100,150,4096,140,3000,2048
   (c) 4096,100,140,150,3000,2048
   (d) 100,150,140,4096,3000,2048
   (e) 3000,4096,140,150,100,2048
2. Ecrire un programme `test` faisant ce travail de vérification (sur une liste ou un vecteur).

EXERCICE 4 *Fusion de deux ABR*

Ecrire une procédure réalisant la fusion de deux ABR (penser à la coupure lors de l'insertion à la racine dans un ABR). Complexité ?

EXERCICE 5 *Trions avec les ABR*

Ecrire une fonction prenant en entrée une liste d'entiers, et affichant ces entiers dans l'ordre croissant, en commençant par les placer dans un ABR, puis en parcourant cet ABR en ordre infixe.
Complexité dans le pire des cas ? et en moyenne ?

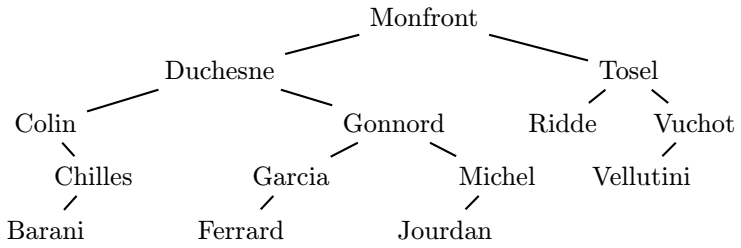
EXERCICE 6 *Commutativité de la suppression*

Dans un ABR, la suppression de x puis de y est-elle équivalente à celle de y puis de x ? (preuve ou contre-exemple...)

EXERCICE 7 *ABR équivalents/inclus*

1. Deux ABR sont dits *equivalents* lorsqu'ils comportent les mêmes étiquettes. Ecrire une fonction testant l'équivalence de deux ABR. Evaluer la complexité.
2. un ABR est *inclus* dans un autre lorsque les étiquettes du premier sont toutes dans le second. Ecrire une procédure efficace gnagnagna...

EXERCICE 8 On considère l'ABR suivant, obtenu en insérant successivement à la racine les noms d'une liste de personnes célèbres :



1. Combien de permutations différentes de la liste initiale permettent d'obtenir cet ABR ?
2. Combien de permutations différentes de la liste initiale permettent d'obtenir un arbre de hauteur 13 ?
3. Reprendre les questions précédentes avec des insertions aux feuilles...