Feuille de TD N^o 5 : Arbres

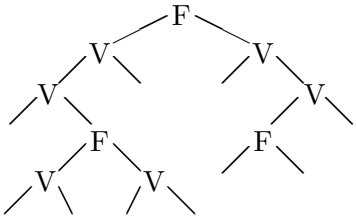
1 Fonctions et procédures sur les arbres binaires

Écrire les fonctions et procédures suivantes sur les arbres binaires :

1. **Hauteur**
2. Le miroir d'un arbre est celui que vous obtiendriez en le regardant à travers un miroir vertical. Les côtés gauche et droite s'inversent.
   - **Fmiroir** qui donne en résultat le miroir de son argument.
   - **Pmiroir** qui transforme son argument en son miroir.
   - **SontMiroirs** qui rend vrai ssi ses deux arguments sont miroirs l'un de l'autre.
3. (facultatif) Les **affichages préfixe, postfixe et infixe** en itératif.
4. À partir d'un arbre binaire A, deux joueurs jouent au jeu J1 suivant: Un pion est initialement sur la racine. Alternativement, chaque joueur déplace le pion de là où il se trouve vers l'un de ses deux fils, au choix de celui qui joue. Celui qui doit jouer alors qu'il est sur une feuille a gagné (donc celui qui joue en amenant le pion sur une feuille perd). Un arbre est J1-gagnant si celui qui commence a une stratégie gagnante, il est J1-perdant s'il n'en a pas, i.e. si c'est celui qui ne commence pas qui a une stratégie gagnante.

   Dans le jeu J2, celui qui doit jouer alors qu'il est sur une feuille a perdu (donc celui qui joue en amenant le pion sur une feuille gagne). Est-ce que A est J1-gagnant ssi il est J2-perdant ?

   Écrire la fonction **J1gagnant** qui prend A en argument et rend vrai ssi A est J1-gagnant.
5. **CompteSansAsc** qui prend A en argument et rend le nombre de nœuds internes valués par vrai mais dont aucun ascendant strict n'est valué par vrai.
   Exemple : 2 pour l'arbre ci-contre.
6. **CompteSansDesc** qui prend A en argument et rend le nombre de nœuds internes valués par vrai mais dont aucun descendant strict n'est valué par vrai. Exemple : 3 pour l'arbre ci-contre.



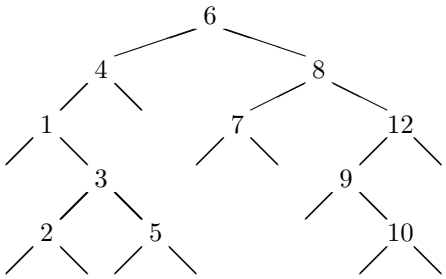
7. **CompteHEgalP** qui rend le nombre de nœuds internes de son argument dont la hauteur est égale à la profondeur.
8. **EstComplet** qui prend A en argument et rend vrai ssi il est complet (ie ssi toutes les feuilles sont à la même profondeur)
9. **EnumereArbres** qui prend en argument un entier n et rend la liste des arbres ayant n nœuds internes.

2

Réécrire l'expression préfixe $/ * + 1\,2\,3 - + * 4\,5 / - 6\,7\,8\,9$ en notation postfixe.
Réécrire l'expression postfixe $1\,2\,3 * - 4 + 5\,6 / 7 + 8\,9 * - /$ en notation préfixe.

3 ABR

3.1 Généralités



Un ABR (Arbre Binaire de Recherche) est un arbre binaire dans lequel les nœuds internes sont valués par des entiers et tel que pour tous nœuds internes x et y, on a que si x est descendant gauche, respectivement droit de y alors on a $val(x) \leq val(y)$, resp. $val(y) \leq val(x)$.

1. L'arbre ci-contre est-il un ABR ?
2. Écrire la fonction `recherche` qui dit si un élément figure dans un ABR et la procédure `ajout` qui insère un nouvel élément dans un ABR au niveau d'une feuille.
3. Décrire la procédure `TriViaABR` qui prend en argument une liste non triée et la trie. Quelle est la complexité (en nombre de comparaisons) au pire de ce tri ? Quelle complexité obtient-on si les ABR restent de profondeur $\theta(profondeur\,minimale\,possible)$?
4. Décrire la procédure `Supprime` qui cherche un élément et le supprime. Discuter selon la situation.
5. Donnez un ABR contenant tous les nombres de 1 à 11 tel que (7 est à la racine) et (1 et 5 sont frères) et (8 et 11 sont frères) et (6 est père de 4)

3.2 Rotations et recherche adaptative

On appelle rotation gauche autour du nœud n la transformation qui fait passer de l'arbre ci-dessous à gauche à celui à droite. Une rotation droite est l'opération symétrique.

1. Si A est un arbre binaire de recherche, et que l'on effectue une rotation autour de l'un de ses nœud, le résultat est-il toujours un arbre binaire de recherche ?
2. Effectuez une rotation gauche autour de 6 dans l'arbre ci-dessus.
3. Écrire la procédure `rotation` qui prend en argument un arbre, un coté (`type cote = [gauche, droite]`) et effectue dans l'arbre la rotation de sens `cote` autour de la racine, quand cela a un sens.
4. On part du principe que ce sont souvent les mêmes éléments qui sont recherchés, et qu'il est donc intéressant de rapprocher les éléments recherchés de la racine. Aussi, à chaque fois que l'on effectue la recherche d'un élément `e`, on effectue des rotations autour des ascendants de `e` pour remonter `e` à la racine. Écrire la procédure de recherche adaptative.

3.3 Arbres bicolores

Un arbre bicolore est un ABR dans lequel les nœuds internes sont coloriés en noir ou en rouge de sorte que:

- la racine est noire.
- si x est le père de y, ils ne sont pas rouges tous les deux.
- Le nombre de nœuds noirs rencontrés le long du chemin de la racine à une feuille est le même pour toutes les feuilles. On appelle hauteur noire de l'arbre ce nombre.

1. Majorez la profondeur d'un arbre bicolore en fonction du nombre de nœuds internes.
2. On insère aux feuilles un nouvel élément. Quels problèmes a-t-on si on décide de le colorier en rouge, resp. en noir ? Opter pour "un moindre mal".
3. On a un arbre bicolore, à ceci près qu'il y a un nœud x qui est rouge, de même que son père y. On veut éliminer le problème, quitte à ce qu'un nouveau problème apparaisse plus haut. Expliquez comment procéder (discuter en fonction de l'existence et de la couleur du frère de y)
4. Décrire un algorithme d'insertion dans un arbre bicolore.
5. Quelles sont les complexités de recherche, d'insertion dans un arbre bicolore, et quelle est la complexité au pire de `TriArbreBicolore` ?