Routage dans un réseau arborescent

Un n-arbre est un arbre (pas forcément binaires) possédant n noeuds et de racine 0. Chaque noeud peut avoir des fils (sauf les feuilles) dont il est le père. La racine a pour père elle-même. La hauteur de la racine vaut 0, et pour tout noeud, elle vaut 1 de plus que celle de son père.

On utilise deux tableaux, le tableau des listes des fils et le tableau des pères.

1 Pour commencer...

1. `hauteur pères e` qui, connaissant le tableau des pères *pres* calcule la hauteur d'un élément e.
2. `tabPere fils` qui calcule le tableau des pères en connaissant le tableau f des fils.
3. `ancetre_elem_h fils a b` qui calcule l'ancêtre le plus proche de deux éléments a et b de même hauteur.
4. `ancetre_elem fils a b` qui calcule l'ancêtre le plus proche de deux éléments a et b non nécessairement de même hauteur.

2 Arbres binaires complets

Tout sommet qui n'est pas une feuille a exactement deux fils, et toutes les feuilles sont à la même hauteur.

1. Quelle est la hauteur d'un arbre de n noeuds ?
2. L'arbre B est un arbre binaire complet a n sommets. A chaque sommet a on associe une étiquette $l(a)$, telle que si le sommet a a pour liste de fils $[b, c]$, avec B_x le sous arbre de B enraciné en x, $\max_{u \in B_b} l(u) < l(a) < \min_{v \in B_c} l(v)$. De plus, les étiquettes sont des entiers de 1 à n.

   Ecrire une fonction qui calcule les étiquettes des noeuds en connaissant le tableau des listes de fils.
3. Calculer l'étiquette d'un noeud quelconque de l'arbre.
4. Ecrire une fonction récursive qui calcule r en complexité $O(\ln(q-p))$ où p et q sont les étiquettes de a et b.
5. Montrer que si a et b, d'étiquettes p et q, sont deux noeuds de l'arbre, alors leur ancêtre commun le plus proche est déterminé par r, où r est l'entier entre p et q qui peut etre divisé par la plus grande puissance de 2.

3 Arbres généraux

Le poids d'un sommet est egal au nombre de sommets du sous arbre enraciné en ce sommet. Le poids d'une feuille est 1, le poids de la racine est n. Un arbre est dit *gauche* si le premier fils de chaque noeud est de poids plus important que les autres fils. On appelle *lourd* un tel sommet, et *léger* les autres. La racine est un sommet léger.

1. Ecrire `poids` qui renvoie un vecteur des poids des sommets, en connaissant le tableau des fils.
2. `gauchir` qui met en premier le fils de poids maximal pour chaque noeud père d'un arbre donné.

3. Soit a un sommet léger d'un arbre gauche et b son père, montrer que $w(b) > 2w(a)$, où $w(x)$ est le poids du sous-arbre enraciné en x. En déduire que le nombre d'ancêtres légers de a est plus petit que $1 + \log(n)$.
4. La *cime* d'un sommet est son père si le sommet est léger, sinon, c'est le plus proche ancêtre de a qui est léger.

   Ecrire `cimes` qui calcule le vecteur des cimes à partir du tableau des listes des fils.
5. On construit deux nouveaux tableaux : $t_1(0) = 0$ et $t(i) = j$ si i est le j-ème fils de son père et $t_2(a) = 0$ si a est léger et $t_2(a) = 1 + (t_2(\text{pere}(a)))$ sinon.

   On construit ensuite les étiquettes $\lambda(a)$ de la manière suivante : $\lambda(0) = []$ et $\lambda(a) = \lambda(\text{cime}(a)) :: ([t_1(a); t_2(a)])$
6. Ecrire `etiquettes` qui calcule les étiquettes des sommets d'un arbre gauche et le tableau des cimes en connaissant le tableau des fils
7. Ecrire `trouve` qui prend l'étiquette d'un sommet x d'un arbre gauche et le vecteur des fils et renvoie x.
8. Majorer la longueur de l'étiquette d'un sommet, en fonction du nombre d'ancêtres légers de ce sommet.
9. Ecrire `ancetre a b` qui calcule le plus petit ancêtre commun de a et b, à partir de leurs étiquettes.