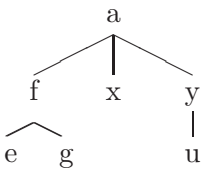
.1 Arbres : généralités

1. Un arbre *ternaire complet* est un arbre dont tous les sommets internes ont exactement 3 fils. Un arbre ternaire complet *maximal* est un arbre ternaire complet dont toutes les feuilles sont à la même distance de la racine.
   (a) Combien y a t-il de nœuds dans un arbre ternaire complet maximal de hauteur k ?
   (b) On s'intéresse à la transformation de cet arbre en arbre binaire, en décidant que pour chaque nœud x de l'arbre initial
   – le fils gauche de x reste son fils gauche dans le nouvel arbre
   – le fils du milieu de x devient le fils droit de son nouveau fils gauche, son fils droit devient le fils droit du noeud précédent
   – le fils droit de x correspond à son frère droit dans l'arbre initial.
   Dessinez l'arbre binaire image d'un arbre ternaire de hauteur 3. Étant donné un arbre ternaire complet de hauteur k, quelle sera la hauteur de l'arbre binaire ainsi obtenu ? Cette transformation conserve-t-elle la dominance des noeuds entre eux ?
   (c) Écrire l'algorithme qui réalise cette transformation.
2. Soit un arbre binaire (un arbre dont les sommets ont 0, 1 ou 2 fils). Envisager tous les cas possibles de réorganisation de l'arbre en cas de suppression d'un sommet interne.
3. On peut réaliser une implémentation chaînée d'un arbre en utilisant des tableaux, de façon analogue à la méthode du curseur vue pour les listes. Chaque sommet correspond à une ligne du tableau, on prévoit une colonne "valeur", une colonne "premier fils", et une colonne "frère". Implémenter les primitives statiques de l'arbre avec une telle méthode.
4. Étant donné un algorithme générique de parcours en profondeur (itératif ou récursif), quelles actions ajouter, et où, pour calculer la hauteur de l'arbre ?
5. Considérons l'arbre suivant.



   Quelle suite de caractères sera affichée si on ajoute dans le parcours des ordres d'affichage :
   – pour chaque feuille rencontrée, dans un parcours en profondeur
   – pour chaque sommet, la première fois qu'on le rencontre, dans un parcours en profondeur
   – pour chaque sommet, la dernière fois qu'on le rencontre, dans un parcours en profondeur
   – pour chaque sommet, dans un parcours en largeur, la première fois qu'on le rencontre.
6. On considère un sous-ensemble des formules de logique propositionnelle. Dans ce sous-ensemble une formule ψ est bien formée ssi :
   – ψ est un axiome (on considère un jeu d'axiomes restreint : p, q, r, s)
   – ψ est de la forme $\neg\varphi$, $\varphi \wedge \chi$ ou $\varphi \vee \chi$
   La vérité d'une formule est définie récursivement à partir de la vérité des axiomes :
   – $\varphi \wedge \chi$ est vraie ssi φ et χ sont vrais

– $\varphi \vee \chi$ est vraie ssi φ ou χ ou les deux sont vrais
– $\neg\varphi$ est vrai ssi φ est fausse

On considère que $p = q = 1$ (ils sont vrais) et $r = s = 0$ (ils sont faux).

(a) Donner une représentation arborescente des formules suivantes :
  – $(p \wedge q) \vee s$
  – $(p \vee s) \wedge (q \wedge r)$,
  – $\neg(p \wedge r) \vee \neg(q \vee s)$

(b) Écrivez un algorithme qui calcule la valeur de vérité d'une formule à partir de sa représentation arborescente (on peut omettre les parenthèses dans l'arbre)

.2 Arbres : parcours et contrôle

1. Ecrire un programme récursif qui affiche tous les mots de longueur n sur un alphabet donné.
2. Proposer un algorithme récursif de calcul de la hauteur d'un arbre, étant données les primitives `existe_fils(x)`, `existe_frère(x)`, `premier_fils(x)`, `frère(x)`.
3. Supposons que l'on décide de stocker un dictionnaire (par exemple le lexique des mots d'un texte) sous la forme d'un arbre dont les nœuds sont des lettres, et dont chaque branche est un mot du dictionnaire (on placera le mot ε à la racine). Calculez le coût (moyen) de recherche d'un mot dans un tel dictionnaire, par rapport au coût de recherche dans une liste ordonnée. Implémentez les opérations d'insertion et de recherche dans une telle structure de données.