IF111 - Algorithmes et structures de données
EI3 - Structures de données et arbres de recherche

Rohan Fossé

rfosse@labri.fr

Insertion dans les ABR, Tas min et AVL

Soit la liste de clés L = (6, 11, 26, 28, 2, 3). Pour chacune des structures, en partant d'un arbre binaire vide, vous devez dessiner l'arbre après chacune des insertions des éléments de la liste L.

1. Arbre binaire de recherche (ABR)
2. Tas min
3. AVL

Tas binaire

1. Dessiner un tas binaire max pour l'ensemble de clés $\{1, 4, 5, 16, 17, 21\}$. Pour le même ensemble dessiner un tas binaire min.
2. Quels sont les nombres minimum et maximum d'éléments dans un tas de hauteur h?
3. Démontrer que un tas avec n sommets a hauteur $\lfloor \log(n) \rfloor$.

Arbres binaires de recherche

1. Dessiner des arbres binaires de recherche de hauteur respectivement 2, 3 et 6 pour le même ensemble de clés $\{1, 4, 5, 10, 16, 17, 21\}$.
2. Dessiner tous les arbres binaires de recherche valués sur $V = \{10, 20, 30, 40\}$ ayant 20 comme valeur à la racine.
3. Quel parcours de l'arbre binaire de recherche permet d'afficher toutes les clés de l'arbre triées?
4. On suppose que des entiers compris entre 1 et 1000 sont disposés dans un arbre binaire de recherche et que l'on souhaite trouver le nombre 363. Parmi les séquences suivantes, lesquelles ne pourraient pas être la suite de sommets parcourus?
   (a) 2, 252, 401, 398, 330, 344, 397, 363
   (b) 924, 220, 911, 244, 898, 258, 362, 363
   (c) 925, 202, 911, 240, 912, 245, 363
5. Ajouter successivement à l'arbre binaire dessiné sur la Fig. 1 les valeurs : 9, 14, 16, 6, 23, 5.
6. Soit un arbre binaire de recherche T dont les clés sont distinctes. Montrer que si le sous-arbre droit d'un noeud x dans T est vide et que x a un successeur y, alors y est l'ancêtre le plus bas de x dont le fils gauche est aussi un ancêtre de x (Ne pas oublier que chaque noeud est son propre ancêtre).
7. Montrer que, si un noeud d'un arbre binaire de recherche a deux fils, alors son successeur n'a pas de fils gauche.

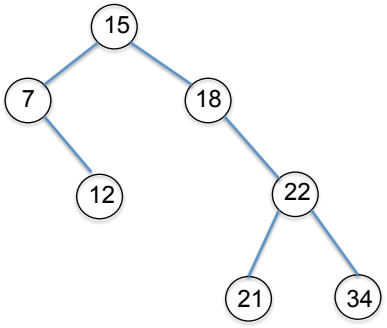

Figure 1: Arbre binaire de recherche

Recherche dans un ABR

1. Écrire une fonction qui retourne true si l'ABR qui lui est passé en paramètre est une feuille.
2. Écrire une fonction récursive qui affiche l'ABR qui lui est passé en para- mètre, par ordre croissant des valeurs.
3. Écrire une fonction récursive qui affiche l'ABR qui lui est passé en para- mètre, par ordre décroissant des valeurs.
4. Écrire une fonction qui retourne la hauteur de l'ABR qui lui est passé en paramètre.
5. Écrire une fonction qui retourne le nombre de nœuds de l'ABR qui lui est passé en paramètre.

Suppression dans un arbre binaire de recherche

Soit l'arbre binaire de recherche donné dans la figure 1

1. Dessiner l'arbre après la suppression du noeud 18.
2. L'opération suppression est-elle "commutative" au sens où la suppression de x puis de y dans un arbre binaire de recherche produit le même arbre que la suppression de y puis de x.
   Si oui dire pourquoi, sinon donner un contre exemple.

Récursivité sur les arbres binaires de recherche

On veut concevoir des algorithmes récursifs pour calculer des propriétés d'un arbre binaire donné en entrée. Les algorithmes peuvent utiliser les fonctions $estVide(T)$ qui retourne vrai si l'arborescence T est vide, $filsgauche(T)$ et $filsdroit(T)$ qui retournent respectivement le sous-arbre gauche et le sous-arbre droit de l'arbre T. Décrire l'idée des algorithmes et en écrire le pseudo-code.

1. Concevoir un algorithme récursif, nommé $Hauteur(T)$, qui calcule la hauteur d'une arborescence binaire T donnée en entrée. Si l'arborescence est vide l'algorithme retournera -1. L'algorithme s'appuie sur la technique diviser pour régner.
2. Concevoir un algorithme récursif qui calcule le nombre de feuilles d'une arborescence binaire donnée en entrée.

Parcours en largeur

1. Appliquer l'algorithme de parcours en largeur au graphe non orienté dans la Figure 2 (c) le sommet origine étant A.

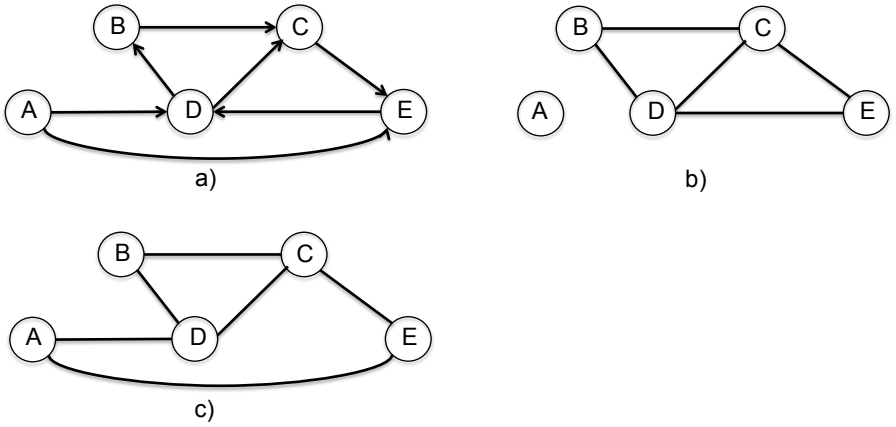

Figure 2: Graphes.

2. Un graphe non orienté $G = (V, E)$ est dit *connexe* si quels que soient les sommets u et v de V, il existe une chaîne reliant u à v. Donner un exemple d'un graphe qui n'est pas connexe. Comment utiliser l'algorithme de parcours en largeur pour tester si un graphe non orienté est *connexe*?

Algorithme de Dijkstra

On considère maintenant des graphes pondérés. Appliquer l'algorithme de Dijkstra au graphe dans la Figure 3 pour calculer la plus courte distance de A à E.

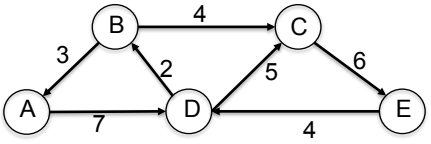

Figure 3: Graphe pour Dijkstra.

Algorithme de Connexité

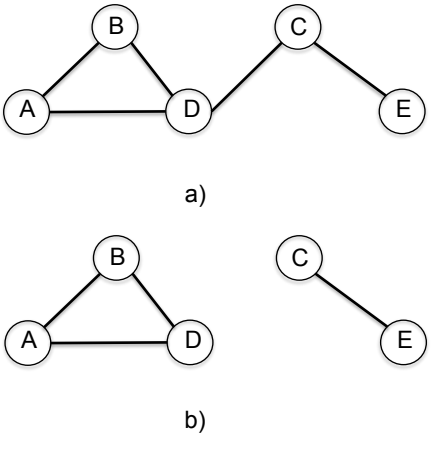

Figure 4: Graphes

Un graphe est dit *connexe* si pour toute paire de sommets s, t, il existe un chemin allant de s à t. Pour chaque graphe en Figure 4 dire s'il est connexe ou pas. Écrivez un algorithme qui teste si un graphe non orienté $G(V, E)$ est connexe ou non.
Donner la matrice d'adjacence et la liste d'adjacence des graphes en Figure 4.

Composantes connexes

Une composante connexe d'un graphe est un sous-graphe connexe tel que chaque sommet de la composante connexe n'est relié à aucun sommet n'appartenant pas à cette composante connexe.

1. Donnez la liste des composantes connexes pour chaque graphe G en Figure 4 .

2. Écrire un algorithme qui calcule la liste des composantes connexes d'un graphe non orienté. On pourra coder cette information en indiquant pour chaque sommet du graphe le numéro de la composante connexe à laquelle il appartient. On peut supposer que les sommets sont numérotés de 1 à n. L'algorithme retournera un tableau où l'indice i pour de 1 à n contiendra le numéro de la composante connexe à laquelle appartient le sommet i.

3. On considère le graphe G défini par la matrice d'adjacence suivante :

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

   Dessinez ce graphe et dire si c'est connexe.

Récursivité sur les arbres

1. Dessiner l'arborescence binaire ayant 10 noeuds {0, 1, 2, ..., 9}, telle que le parcours infixe et le parcours postfixe de cette arborescence produisent respectivement les suites suivantes : 9, 3, 1, 0, 4, 2, 6, 7, 8, 5 (infixe) et 9, 1, 4, 0, 3, 6, 7, 5, 8, 2 (postfixe). Dire quel est le raisonnement utilisé pour arriver à la solution.
   Dire quelle est la hauteur du noeud 1 et 2 dans l'arborescence dessinée et quelle est la hauteur de l'arborescence elle même.

2. Ecrire la suite d'entiers obtenue par le parcours préfixe de l'arborescence obtenue dans la question precedente. Ecrire un algorithme recursif qui prend en entrée une arborescence et affiche les valeurs des nœuds dans l'ordre du parcours préfixe de l'arborescence. L' algorithme utilisera les fonctions $estVide(T)$ qui retourne vrai si l'arborescence T est vide, $val(T)$ qui retourne l'étiquette de la racine d'un arbre non vide et $filsgauche(T)$ et $filsdroite(T)$ qui retournent respectivement la sous-arbre gauche et le sous-arbre droit de l'arborescence T.