TD 2 – Algorithmes classiques

Exercice 1. *Implantations*

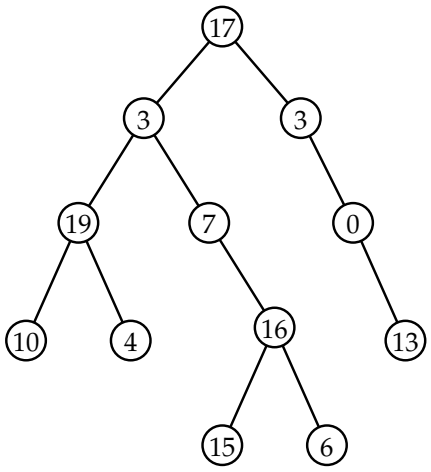
Pour les implantations, se baser sur les représentations d'arbres et de graphes vues dans le Bloc 4.

1. Implanter les parcours infixe, préfixe et suffixe d'un arbre binaire.
2. Implanter le parcours en largeur d'un arbre binaire.
3. Implanter les parcours en profondeur et en largeur d'un graphe.
4. Que se passe-t-il si on applique ces parcours sur des graphes qui sont des arbres ?
5. Utiliser le parcours en profondeur pour effectuer la détection de cycles dans un graphe.
6. Implanter l'algorithme de Dijkstra. *Il y a un choix à effectuer pour implanter la file de priorité. Dans un premier temps, utiliser une structure de données simple (tableau par exemple) sans se préoccuper de la complexité de l'obtention de l'élément de distance minimale. En bonus, il est possible d'utiliser la structure de tas, voire de tas de Fibonacci, qui fournissent les meilleures complexités* [1].

Exercice 2. *Parcours d'arbres*

1. Écrire un algorithme qui calcule la valeur maximale présente dans un arbre binaire.
2. Écrire un algorithme qui calcule le nombre de feuilles d'un arbre binaire.
3. Qu'affichent les parcours infixe, préfixe, suffixe et en largeur, pour l'arbre de l'exercice suivant ?
4. **i.** Dessiner un arbre binaire, de hauteur aussi petite que possible, dont le parcours infixe affiche 6 3 2 7 4 8 5 0 1 9.
   ii. Même question avec les parcours préfixe et suffixe.
   iii. Même question avec le parcours en largeur.

Exercice 3. *Plus proche ancêtre*



Dans un arbre binaire A, un nœud x est un *ancêtre* d'un nœud y s'il existe une suite $x_0, \dots, x_k$ avec $x_0 = y$, $x_k = x$ et $x_{i+1} = \text{père}(x_i)$ pour $i = 0, \dots, k-1$. Pour deux nœuds u et v de A, le *plus proche ancêtre commun* (PPAC) de u et v est l'ancêtre de u et de v qui a la plus grande hauteur.

1. Dans l'arbre binaire ci-contre, quels sont les ancêtres du nœud de valeur 4? Quels sont les ancêtres communs aux nœuds de valeur 19 et 15 ? Et leur plus proche ancêtre?
2. Deux nœuds u et v d'un arbre binaire A ont-ils toujours un ancêtre commun?
3. Écrire un algorithme qui étant donnés deux nœuds x et y d'un arbre binaire A décide si x est un ancêtre de y.
4. En déduire un algorithme qui calcule le plus proche ancêtre commun à deux nœuds u et v de A. 'Evaluer la complexité de votre algorithme.
5. Si ce n'est pas le cas, proposer un algorithme de complexité linéaire. *On pourra se servir d'une pile.*

Exercice 4. *Arbres binaires de recherche*

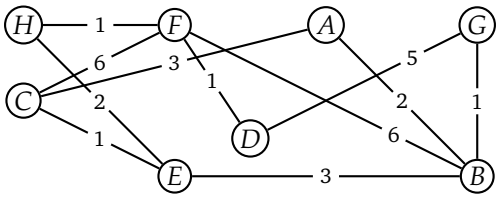
1. Écrire un algorithme qui calcule la valeur maximale dans un ABR, et analyser sa complexité.
2. Qu'affiche le parcours infixe d'un ABR ?

On appelle *successeur* d'un nœud x dans A un nœud y, distinct de x, tel que $\text{val}(x) \leq \text{val}(y)$ et pour tout nœud $z \neq x, y$, soit $\text{val}(z) < \text{val}(x)$, soit $\text{val}(z) \geq \text{val}(y)$, et $y = \emptyset$ si x est le nœud de plus grande valeur dans A. *On suppose ici que tous les nœuds ont des valeurs distinctes.*

1. Une explication du fonctionnement des tas est disponible dans les diapositives suivantes : `http://www.lirmm.fr/~grenet/AlgoComplexite/2.Arbres.pdf#page=79`.

3. Montrer que si x possède un fils droit, le successeur y de x est le minimum du sous-arbre droit de x.
4. En déduire un algorithme de suppression d'un nœud x d'un ABR. *Si x possède un fils droit, il faut le remplacer par son successeur y et faire remonter le fils droit de y, s'il existe, à la place de y. Attention au cas où x n'a pas de fils droit.*

Exercice 5. *Parcours de graphes*



Pour les deux premières questions, ne pas tenir compte des longueurs des arêtes.

1. Appliquer le parcours en profondeur sur le graphe ci-dessus, avec A comme point de départ. À quel moment détecte-t-on un cycle ? *L'insertion dans la pile ou file se fait par ordre alphabétique.*
2. Appliquer le parcours en largeur à partir du sommet A et calculer les distances de A à chacun des sommets du graphe.
3. Appliquer l'algorithme de Dijkstra pour calculer la distance de A à tous les autres sommets du graphe.

Exercice 6. *Exercice à rendre*

L'algorithme de Dijkstra fournit les distances du sommet s passé en paramètre à tous les sommets du graphe.

1. Modifier l'algorithme de Dijkstra pour qu'il calcule également les chemins les plus courts entre le sommet s et tous les autres sommets.
2. Quelle est la complexité en mémoire de l'algorithme obtenu ?