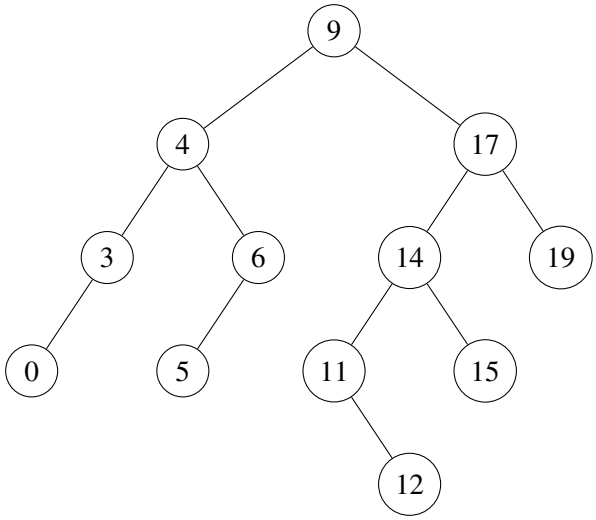
Travaux dirigés n^o 6
Structures de données

Exercice 6.1 — Parcours d'arbres



1. Effectuez les parcours demandés sur l'arbre ci-dessus et donnez leur complexité :
   - parcours en largeur
   - parcours en profondeur *préfixe*
   - parcours en profondeur *infixe*
   - parcours en profondeur *postfixe*.
2. Cet arbre représente-t-il un tas-max ou un tas-min ? Pourquoi ?
3. Cet arbre est-il un arbre binaire de recherche (ABR) ? Pourquoi ?

Exercice 6.2 — Recherche en profondeur (DFS)

1. Ecrivez la fonction DFS(A,x) qui consiste à rechercher l'élément x en profondeur dans un arbre binaire A quelconque. Quelle est sa complexité ?
2. Adaptez cette fonction de recherche en profondeur à un ABR. Quelle est la nouvelle complexité ?

Exercice 6.3 — Type abstrait Pile

On utilise une pile pour évaluer des expressions arithmétiques sur les entiers écrites en *notation polonaise inversée* (NPI). On suppose que chaque opération est binaire. Ses deux opérandes vont précéder l'opérateur dans l'expression. Cette notation est sans ambiguïté et le parenthésage devient superflu. Voici un exemple d'une telle expression suivie de sa transcription en polonaise inversée :

$$(3 \ * \ (5 \ + \ 1)) \ - \ (4 \ / \ 2)$$

$$3 \ 5 \ 1 \ + \ * \ 4 \ 2 \ / \ -$$

Une expression est donnée sous la forme d'un tableau E dont chaque case contient une chaîne de caractères représentant soit un entier soit un opérateur.

1. Rémémorez-vous les primitives qui permettent de manipuler une pile.
2. Ecrivez une fonction Evalue(E) qui utilise une pile pour évaluer l'expression E passée en paramètre.

Exercice 6.4 — Type abstrait File

Vous avez vu en cours l'implantation d'une file d'attente sur les listes chaînées. On se propose cette fois de modéliser une file avec un tableau de n cases (numérotées de 0 à $n-1$) que l'on imagine comme étant *circulaire*. Ainsi, on suppose que la case suivant la case d'indice $n-1$ est la case d'indice 0.

1. Rappelez les différentes primitives d'une file.
2. Dans la file circulaire, on doit garder une case toujours une vide comme tampon (sa capacité est donc de $n-1$ éléments). Deux indices *début* et *fin* assurent la gestion de la file.
   Implémentez les primitives de la file d'attente circulaire.

Exercice 6.5 — Suppression dans un ABR

Ecrivez l'algorithme de suppression d'un élément présent dans un ABR. Vous aurez peut-être besoin d'une fonction auxiliaire pour cela.