# ALGO1 : Algorithmique – DM

pour le 16 octobre 2018

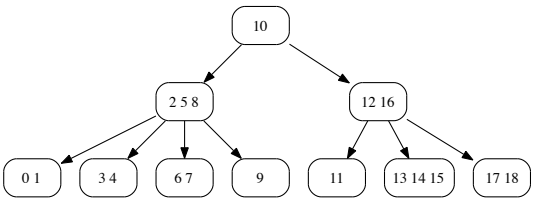
### Consignes

- *Une attention particulière devra être portée à la rédaction et la clarté des explications. Ne pas hésiter à illustrer vos explications avec des schémas et des exemples. Inutile de perdre du temps à taper votre devoir sous LaTeX, si cela doit se faire au détriment des explications et des schémas illustratifs.*
- *Le travail devra être effectué par* ***binôme****. Il devra être rendu* ***le mardi 16 octobre 2018 à 13h45*** *à Emmanuel Caruyer.*

## 1 Arbres 2-3-4

Dans ce devoir, nous nous intéressons à une famille particulière de B-arbres, la famille $\mathcal{A}$. Les nœuds des arbres de $\mathcal{A}$ sont d'arité 2, 3 ou 4. Un nœud d'arité $i$ ($i = 2..4$) comporte $i-1$ clés et $i$ arbres fils. Les feuilles ne contiennent pas de valeurs et seront notées $\emptyset$. La propriété d'arbre de recherche étudiée sur les arbres binaires s'étend naturellement à de tels arbres. Par exemple, pour les nœuds d'arité 3, la première clé doit être comprise entre les valeurs des clés des sous-arbres gauches et du centre alors que la deuxième clé doit être comprise entre celles des sous-arbres du centre et de droite. Nous imposons de plus que la profondeur de chaque feuille soit la même.

La figure ci-contre présente un exemple d'arbre de $\mathcal{A}$. Nous prendrons pour convention de ne pas faire figurer les feuilles.




Question 1 – Écrire un algorithme Recherche(v,T) permettant de rechercher une clé v dans un arbre T de $\mathcal{A}$. Donner une borne asymptotique sur le nombre de comparaisons effectuées par cet algorithme pour rechercher une clé dans un arbre comportant $n$ clés (cas le pire).

QUESTION 2 – Expliquer, sans écrire de pseudo-code, comment lors d'une descente de la racine à une feuille donnée, il est possible de modifier cet arbre de façon à enlever tous les nœuds d'arité 4 le long de ce chemin. L'ensemble des clés présentes dans l'arbre ne devra pas changer et l'arbre résultant devra encore appartenir à $\mathcal{A}$.
QUESTION 3 – Expliquer, sans écrire de pseudo-code, comment lors d'une descente de la racine à une feuille donnée, il est possible de modifier cet arbre de façon à enlever tous les nœuds d'arité 2 le long de ce chemin, sauf pour le nœud racine. L'ensemble des clés présentes dans l'arbre ne devra pas changer et l'arbre résultant devra encore appartenir à $\mathcal{A}$. Il n'est pas nécessaire de détailler tous les cas de cette méthode, s'ils sont symétriques avec des cas déjà présentés.

## 2 Plus grand sous-arbre complet

État donné un arbre binaire $T$ quelconque, un *sous-arbre* de $T$ est un sous-graphe connexe de $T$ vu comme un graphe orienté. Par ailleurs, on dit qu'un arbre est *complet* si tous ses nœuds internes ont exactement 2 fils, et que toutes ses feuilles ont la même profondeur.
QUESTION 4 – Écrire un algorithme calculant le plus grand sous-arbre complet d'un arbre donné. Il devra retourner la hauteur ainsi que la racine de cet arbre.
QUESTION 5 – Montrer que cet algorithme termine, qu'il est correct, et calculer sa complexité dans le pire cas.

## 3 Chemins tricolores

On considère un graphe orienté $G = (S, A)$, dont les sommets peuvent être soit bleus, blancs ou rouges. À partir d'un sommet $v \in S$ de ce graphe, on cherche tous les sommets atteignables en parcourant un chemin *tricolore*, c'est-à-dire un chemin $v = v_0 \to v_1 \to \ldots \to v_k$ dont la couleur des nœuds traversés est successivement bleu, blanc, rouge, bleu, etc.

Plus formellement, on dit qu'un chemin $v = v_0 \to v_1 \to \ldots \to v_k$ est tricolore ssi

$$couleur[k] = \begin{cases} bleu & \text{si } k = 0 \mod 3 \\ blanc & \text{si } k = 1 \mod 3 \\ rouge & \text{si } k = 2 \mod 3. \end{cases}$$

QUESTION 6 – Écrire un algorithme TRICOLORE$(v)$ qui calcule tous les sommets atteignables depuis $v$ par un chemin tricolore.
QUESTION 7 – Montrer que cet algorithme termine, qu'il est correct, et calculer sa complexité dans le pire cas.