UNIVERSITE DE OUAGADOUGOU
-=-=-=-=-=--=-
UFR / SEA / L2S4
Parcours : Maths-Informatique-Physique
-=-=-=-=-=--=-
Enseignant : Cédric BERE

Année académique 2013-2014

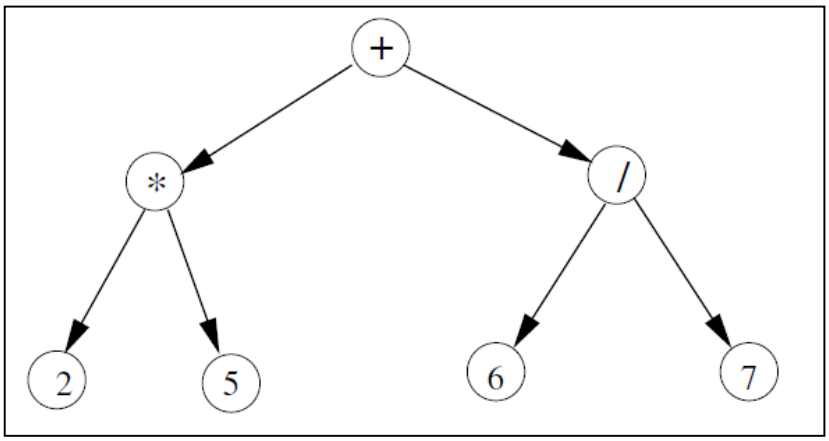
Structures de Données : TD n°3
(Arbres - Graphes)

Exercice 1
Écrire une fonction qui recherche un nombre n dans un arbre d'entiers. La fonction doit renvoyer 1 si n est présent, 0 sinon.

Exercice 2
Écrire une fonction qui calcule la somme des valeurs des nœuds dans un arbre de float.

Exercice 3
Écrire une fonction calculant le maximum des valeurs des nœuds d'un arbre d'entiers.

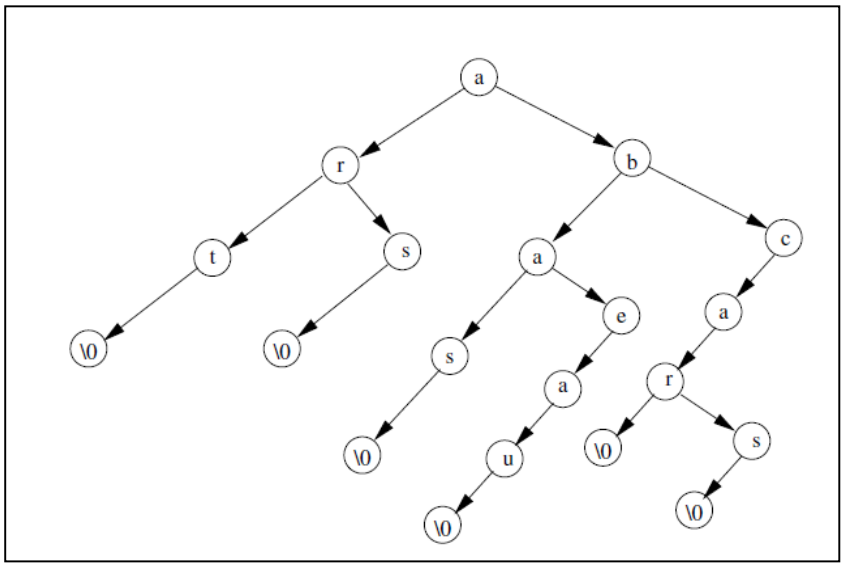
Exercice 4
On représente une expression arithmétique par un arbre dont les nœuds sont des opérateurs ou des nombres et dont les feuilles sont des nombres.



L'arbre associé à l'expression (2 ∗5) + (6/7)

1- Écrire les structures de données permettant de représenter une expression arithmétique sous forme d'arbre.
2- Écrire les structures de données permettant de représenter une expression arithmétique sous forme d'arbre.
3- Écrire une fonction d'évaluation d'une expression arithmétique sous forme d'arbre.
4- Écrire une fonction qui génère une chaîne de caractères contenant l'expression préfixée correspondant à un arbre.
5- Écrire une fonction de création d'un arbre représentant une expression arithmétique donnée au format préfixé.
6- Écrire une fonction d'écriture d'une expression arithmétique sous forme parenthésée à partir d'un arbre.
7- Écrire une fonction de création d'un arbre représentant une expression arithmétique donnée au format parenthésé.

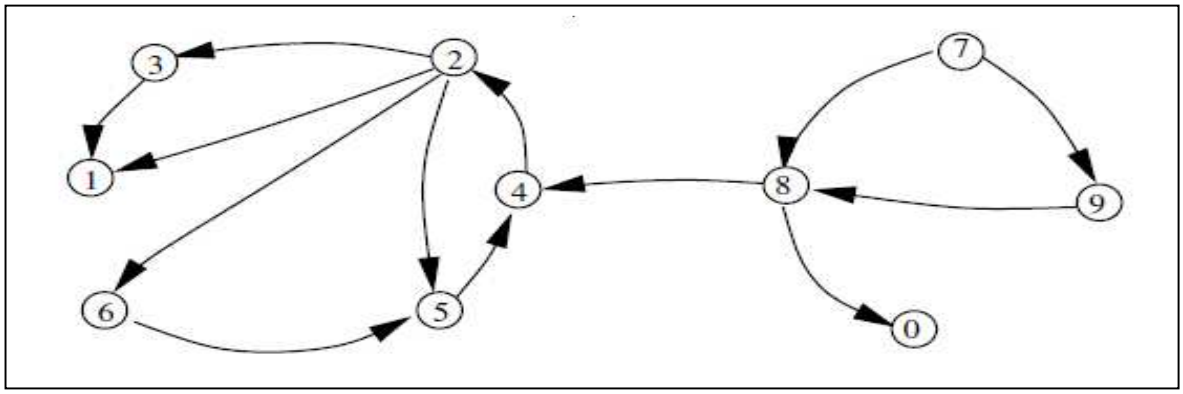
Exercice 5 (Arbres lexicographiques) [Parcours informatique]
On représente un dictionnaire sous forme d'arbre binaire. Les premières lettres possibles pour le mot sont obtenues en suivant le branche de droite à partir de la racine (par exemple, sur la figure, les premières lettres possibles sont *a*, *b* ou *c*).



Pour accéder aux deuxièmes lettres des mots, on passe au fils gauche de la première lettre. Toutes les deuxièmes lettres possibles sont alors obtenues en prenant la branche de droite (Par exemple, sur la figure, avec la première lettre *b*, les deuxième lettres possibles sont *a* et *e*). Pour avoir la troisième lettre, on passe au fils gauche de la deuxième lettre, et ainsi de suite jusqu'à trouver un \0.

1. Proposer une structure de données pour représenter un dictionnaire.
2. Écrire une fonction récursive qui vérifie si un mot est dans le dictionnaire.
3. Écrire une fonction itérative qui vérifie si un mot est dans le dictionnaire.
4. Écrire une fonction qui vérifie l'orthographe d'un fichier texte, et affiche les mots qui ne sont pas dans le dictionnaire. On suppose que les mots sont séparés par des espaces ou des retours à la ligne.
5. Écrire une fonction qui rajoute un mot *m* dans le dictionnaire. On cherchera la dernière lettre du plus long sous-mot (qui soit le début du mot *m*) qui est présent dans le dictionnaire, et on cherchera en même temps l'adresse du nœud de l'arbre correspondant. On ajoutera ensuite des descendants à ce nœud.
6. Écrire une fonction qui sauvegarde un dictionnaire sous forme de liste de mots dans un fichier texte.
7. Écrire une fonction qui charge en mémoire un dictionnaire donné sous forme de liste de mots dans un fichier texte.
8. Que faudrait-il faire pour que les mots du dictionnaire sauvegardé soient rangés dans l'ordre alphabétique ?

Exercice 6
Appliquer le parcours en profondeur au graphe de la figure en prenant pour sommet de départ le sommet 2, puis en prenant pour sommet de départ le sommet 7. On donnera la liste des sommets visités dans l'ordre où ils sont visités.

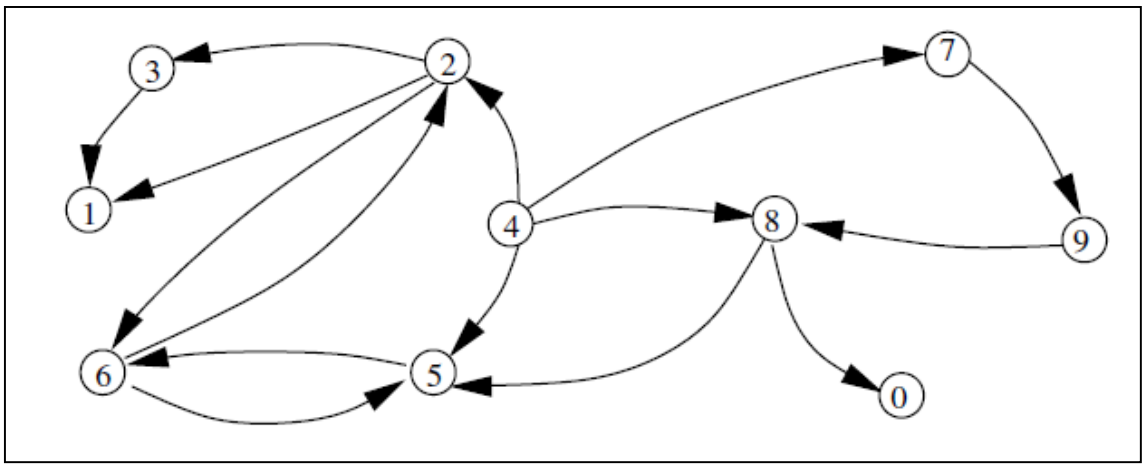


Exercice 7

Appliquer le parcours en largeur au graphe de la figure (figure de l'exercice 6)en prenant pour sommet de départ le sommet 4, puis en prenant pour sommet de départ le sommet 7. On donnera la liste des sommets visités dans l'ordre où ils sont visités et on maintiendra à jour une file.

Exercice 8 (plus court chemin)

Appliquer le parcours en largeur au graphe de la figure en prenant pour sommet de départ le sommet 4, puis en prenant pour sommet de départ le sommet 8. On donnera la liste des sommets visités dans l'ordre où ils sont visités et on maintiendra à jour une file.



Pour chaque sommet *S* inséré dans la file, on dessinera une flèche de couleur du sommet *S* vers le dernier sommet *V* défilé. Que remarque-t'on en suivant les flèches de couleur ?

Exercice 9 (Implémentation du plus court chemin)

Écrire une fonction *C* qui donne le plus court chemin entre deux sommets ***S1*** et ***S2*** d'un graphe donné sous forme de matrices d'adjacence. Pour chaque sommet *S* inséré dans la file, on mémorisera le numéro ***preced*** du dernier sommet *V* défilé avant d'insérer *S* dans la file. Pour afficher le chemin, on suivra les numéros ***preced*** à la manière des pointeurs suivant dans un parcours de liste chaînée.

Exercice 10 (Implémentation du plus court chemin)

1- Implémenter le parcours en profondeur récursif avec une représentation du graphe sous forme de matrice d'adjacence.

2- Écrire une fonction qui prend en paramètre deux sommets ***S1*** et ***S2*** dans un graphe donné sous forme de matrice d'adjacence, et qui renvoie 1 s'il existe un chemin de ***S1*** à ***S2*** et renvoie 0 sinon.