L2 Informatique

TP Algorithmique – Arbres binaires

1 Représentation contigüe d'un arbre binaire

Cette première partie consiste à implémenter une structure d'arbres binaires de chaînes de caractères au moyen de tableaux, en supposant que le nombre de nœuds dans l'arbre est fixé et connu. Un tel arbre pourra être représenté au moyen d'un enregistrement comportant :

- sa taille n ;
- l'indice de sa racine (entre 1 et n) ;
- un tableau de chaînes de caractères stockant les n étiquettes de ses nœuds indicées à partir de 1 ;
- trois tableaux de n entiers indicés à partir de 1, indiquant pour chaque nœud les indices des fils gauche et droite, ainsi que du père.

Par exemple, la structure ci-dessous permet de représenter l'arbre de la figure 1. Cet arbre représente un système de livraison tel qu'il sera utilisé dans les parties suivantes.

Taille	9									
Racine	1									
Etiquettes	-	Usine	depot_A	depot_B	depot_C	magasin_1	magasin_2	magasin_3	magasin_4	magasin_5
Gauche	-	2	5	4	7	0	0	0	0	0
Droite	-	3	6	9	8	0	0	0	0	0
Pere	-	0	1	1	3	2	2	4	4	3
Indices	0	1	2	3	4	5	6	7	8	9

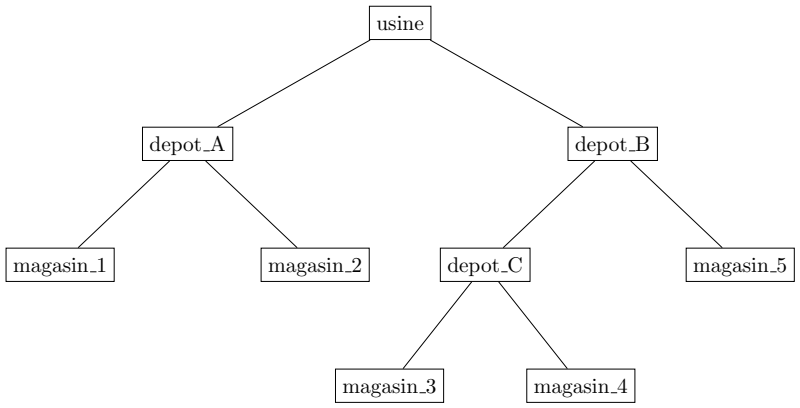

Figure 1: Exemple d'arbre représentant un système de livraison.

1. Écrire le type `ArbreTab` proposant une telle implémentation d'un arbre binaire de chaînes de caractères.
2. Écrire un sous-programme permettant de construire un arbre à partir d'un fichier texte comportant, sur la première ligne, la taille de l'arbre et l'indice de sa racine, puis pour chaque nœud (un nœud par ligne) son étiquette et les indices respectifs de son fils gauche et son fils droit (0 si absent). Les indices des pères se déduisent des informations précédentes et ne figurent pas dans le fichier.

Par exemple, l'arbre précédent pourra être construit à partir du fichier texte suivant :

```
9 1
usine 2 3
depot_A 5 6
depot_B 4 9
depot_C 7 8
magasin_1 0 0
magasin_2 0 0
magasin_3 0 0
magasin_4 0 0
magasin_5 0 0
```

3. Écrire une fonction calculant la profondeur d'un nœud (étant donné un `ArbreTab` et un indice de nœud).

4. Écrire une procédure d'affichage d'un `ArbreTab` selon un parcours préfixe. Pour chaque nœud (un nœud par ligne), on affichera son étiquette, les étiquettes de ses fils gauche, fils droit et père ("$*$" si vide), ainsi que sa profondeur.

2 Représentation chaînée d'un arbre binaire

5. Définir les types `Noeud` et `ArbreBinaire` permettant une représentation chaînée d'un arbre binaire de chaînes de caractères. Outre les références aux fils gauche et droit, on souhaite ajouter à chaque nœud une référence au nœud parent (pointeur vers le père).

6. Écrire un sous-programme permettant de convertir un `ArbreTab` en `ArbreBinaire`.
   Afin de vérifier la conversion, décliner les questions 3 et 4 (calcul de la profondeur d'un nœud et affichage de l'arbre) pour un `ArbreBinaire`.

7. Écrire des fonctions booléennes (5) testant si un nœud est : la racine, un nœud interne, une feuille, un fils gauche, un fils droit. Chaque fonction prendra comme unique paramètre l'adresse d'un nœud, donc un `ArbreBinaire`.

8. Écrire une fonction calculant le nombre de feuilles d'un `ArbreBinaire`.

9. Écrire un sous-programme qui affiche de gauche à droite les étiquettes des feuilles d'un `ArbreBinaire`.

10. Écrire une fonction calculant la hauteur d'un `ArbreBinaire`.

11. Écrire une fonction qui recherche une étiquette dans un `ArbreBinaire`. Cette fonction devra retourner l'adresse du nœud (donc un `ArbreBinaire`) dont l'étiquette correspond à celle entrée en paramètre.

12. Écrire un sous-programme qui supprime un arbre en libèrant la mémoire allouée dynamiquement pour stocker ses informations.

3 Système de livraison

Dans cette partie, on utilise cette structure pour représenter un système de livraison d'un producteur vers des magasins en passant par des entrepôts intermédiaires. Dans cet arbre la racine représente une usine qui produit des marchandises. Les nœuds internes (hors racine) sont des dépôts de marchandises, et les feuilles sont des magasins. La livraison d'une commande effectuée par un magasin passe par tous les dépôts situés entre le magasin et l'usine. À chaque commande, l'usine ainsi que les dépôts traversés doivent stocker en mémoire le fils auquel le produit doit être envoyé. L'usine et les dépôts livrent les produits par ordre chronologique des commandes passées.

13. Définir une structure de données **file de caractères** (représentation au choix), les primitives usuelles sur les files (initialiser, est_vide, consulter, ajouter, retirer), ainsi qu'une procédure d'affichage d'une file.

14. Modifier la structure `ArbreBinaire` afin que tous les nœuds comportent une file qui indique la liste des entrepôts ou magasins à livrer. Les livraisons s'effectuent en favorisant les commandes les plus anciennes, d'où l'usage d'une file (les nouvelles commandes sont ajoutées en queue, et la livraison d'une commande est traitée en priorisant la commande de tête). Chaque file contiendra des caractères qui indiqueront quel fils (site) doit être livré. La valeur 'G' indiquera qu'il faut livrer le fils gauche et la valeur 'D' indiquera qu'il faut livrer le fils droit. Les files des feuilles resteront vides.

15. Modifier l'opération de conversion `ArbreTab` vers `ArbreBinaire` afin de prendre en compte ces files et les initialiser (voir Figure 2).

16. Écrire un sous-programme qui permet à un magasin de passer une commande. Ce sous-programme prend en paramètre l'étiquette du magasin où l'on souhaite passer commande. Dans une première étape, il recherche le nœud correspondant à l'étiquette fournie, et ensuite remonte de ce magasin jusqu'à la racine en mettant à jour la file de commandes de chaque dépôt traversé ('G' ou 'D' selon le nœud d'où provient la commande) ainsi que de l'usine.

17. Écrire un sous-programme qui livre un produit depuis l'usine jusqu'au magasin. Le sous-programme doit mettre à jour les files de commandes et indiquer le trajet effectué. Par exemple, une livraison à partir de l'arbre de la Figure 3 se déroulerait comme suit:

    (a) on part de l'usine ;
    (b) on regarde quel dépôt l'usine doit livrer parmi A et B;
    (c) il faut livrer au dépôt situé à gauche, donc on livre A;
    (d) on regarde quel magasin le dépôt A doit livrer: 1 ou 2 ;
    (e) il faut livrer à gauche, donc on livre le magasin 1 ;
    (f) le magasin 1 n'a rien à livrer : fin de l'opération.

    L'affichage produit est alors $\mathbf{usine} => \mathbf{depot\_A} => \mathbf{magasin\_1}$. La figure 4 représente l'arbre mis à jour après cette livraison.

18. Écrire un sous-programme qui affiche l'état des commandes, c'est-à-dire l'étiquette et la file d'attente de chaque nœud interne de l'arbre.

19. Écrire un sous-programme permettant de simuler la série de commandes de la Figure 3, suivie des livraisons.

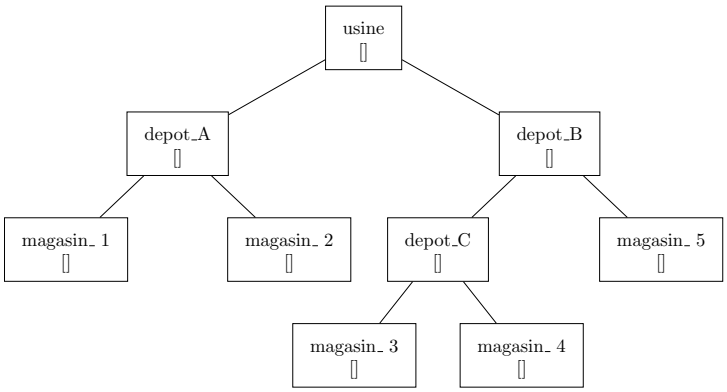

Figure 2: Arbre de départ: les files des commandes sont vides.

Série de commandes:

1. magasin_1
2. magasin_5
3. magasin_1
4. magasin_4
5. magasin_4
6. magasin_2
7. magasin_3
8. magasin_5

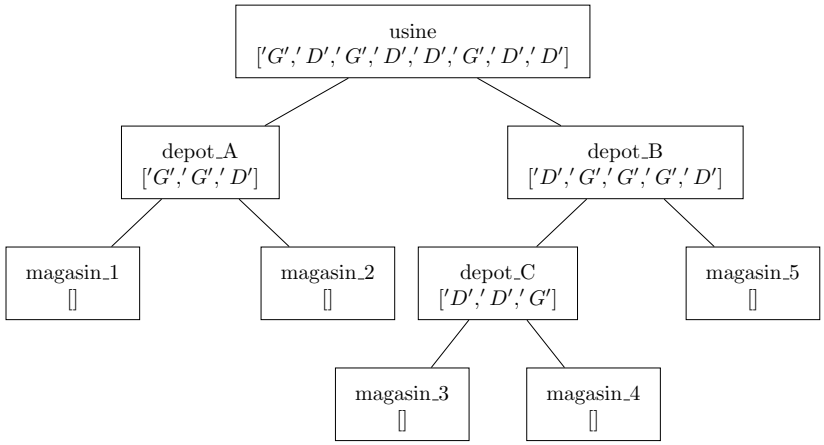

Figure 3: État de l'arbre après la série de commandes situées à gauche: Magasin1 est la première commande; il est fils gauche de Dépôt A donc 'G' est en tête de file pour dépôt A, de même pour Usine.

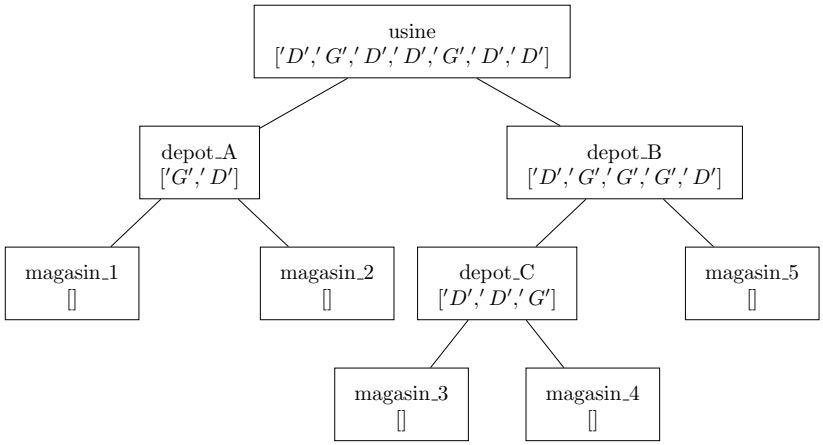

Figure 4: État de l'arbre après une livraison