

L2 Informatique

Algorithmique 3 – Feuille de TD n° 2

1 Reconstruction d'arbres

Soit B un arbre binaire dont les noeuds sont étiquetés par des caractères uniques. Un parcours de l'arbre engendre un mot, selon l'ordre dans lequel les noeuds sont visités.

On suppose que :

- le mot LDNWOUREF est obtenu lors du parcours préfixe de l'arbre B ;
- WNODLERUF est obtenu lors du parcours infixe du même arbre B.

1. À partir des deux mots correspondant aux parcours préfixe et infixe, expliquer comment reconstruire l'arbre B.

2. On considère les types suivants :

- **lettre** : un caractère entre 'A' et 'Z' ;
- **mot** : une chaîne de caractères (représentant une suite de lettres), dont la première lettre est en position 1.

Implémenter les opérations associées suivantes :

- **longueur(m)** : retourne la longueur du mot m ;
- **position(m,1)** : retourne la position de la première occurrence de la lettre 1 dans m ; ($1 \leqslant \text{position}(m,1) \leqslant \text{longueur}(m)$) ;
- **caractere(m,p)** : retourne la lettre du mot m en position p (on suppose $1 \leqslant p \leqslant \text{longueur}(m)$) ;
- **extrait(m,p,1)** : retourne le mot extrait de m, formé de 1 caractères à partir de la position p.

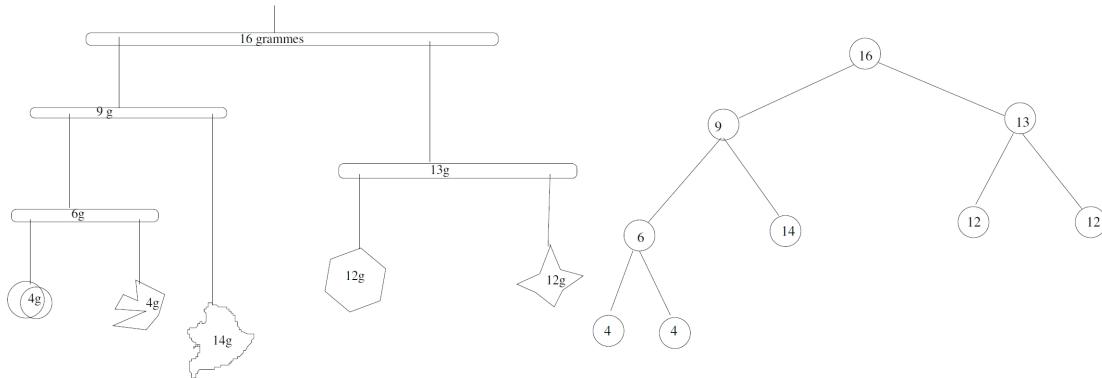
3. Écrire un sous-programme qui vérifie que deux mots sont valides (ne sont composés que de lettres), ont la même longueur, contiennent exactement les mêmes lettres, et ne comportent aucune lettre répétée.

4. En utilisant les opérations définies ci-dessus, écrire un sous-programme qui reconstruit l'arbre binaire B à partir de deux mots correspondant aux parcours préfixe et infixe de cet arbre. On supposera que les deux mots satisfont les critères de la question précédente.

2 Mobile

On considère un arbre binaire étiqueté représentant un mobile (jouet que l'on accroche au dessus du berceau des bébés) : chaque feuille correspond à un objet en plastique accroché à l'une de ses extrémités et chaque noeud interne correspond à une barre de support. L'information associée à chaque noeud est son poids (poids de la barre ou poids de l'élément accroché).

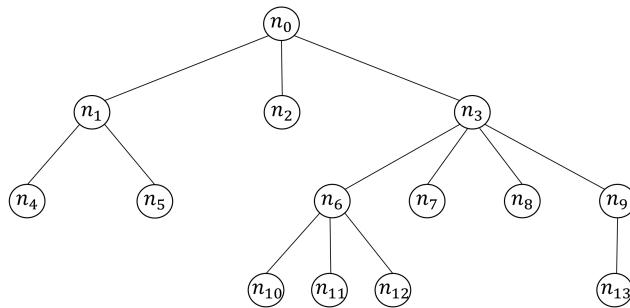
On donne ci-dessous le dessin d'un mobile et l'arbre binaire étiqueté associé.



1. Rappeler les déclarations de type nécessaires pour représenter un mobile (arbre binaire).
2. Écrire le sous-programme `poids` qui calcule le poids total d'un mobile.
→ Par exemple, le poids total du mobile de la figure est de 90 g.
3. Écrire le sous-programme `estEquilibre` qui détermine si un mobile est équilibré. Un mobile est dit *équilibré* s'il correspond à un arbre binaire localement complet et si en chaque nœud, le poids du sous-arbre gauche est égal au poids du sous-arbre droit.
→ Par exemple, le mobile de la figure est équilibré.
4. Écrire le sous-programme `existePoids` qui détermine si le mobile contient un objet en plastique de poids P donné.

3 Représentation d'un arbre n-aire en arbre binaire

1. Donner la représentation sous forme d'arbre binaire de l'arbre n-aire suivant :



2. Soit un arbre n-aire A représenté sous forme d'arbre binaire.

- (a) Écrire une fonction qui teste si un nœud est une feuille.
- (b) Écrire une fonction qui affiche tous les fils d'un nœud n donné.
→ Par exemple, pour l'arbre représenté ci-dessus, les fils du noeud n_3 sont n_6, n_7, n_8, n_9 .

4 Arbres binaires de recherche (ABR)

1. Soit A et B deux ABR vides. Dessiner les arbres obtenus après l'ajout successif aux feuilles des valeurs suivantes :
 - (a) 12 – 8 – 15 – 6 – 3 – 19 – 23 – 7
 - (b) 25 – 12 – 7 – 35 – 9 – 39 – 43 – 18 – 5 – 21 – 14 – 22 – 28
2. On considère un ABR de taille n , contenant tous les entiers 1 à n .
 - (a) Pour $n = 4$, donner tous les ABR possibles ainsi que les ordres d'insertion qui permettent de les obtenir.
 - (b) Pour $n = 10$, donner quelques exemples d'ordres d'insertion permettant d'obtenir des arbres de hauteur maximale et minimale.
 - (c) Indiquer la hauteur maximale et minimale d'un ABR en fonction de n .
3. On considère un ABR contenant des entiers entre 1 et 1000. Parmi les séquences suivantes, distinguer les parcours de recherche possibles et impossibles permettant de rechercher la valeur 363 :
 - (a) 2 – 252 – 401 – 330 – 398 – 406 – 397 – 363
 - (b) 202 – 925 – 911 – 240 – 912 – 245 – 363

- (c) 924 – 220 – 244 – 911 – 898 – 258 – 362 – 363
(d) 2 – 399 – 387 – 219 – 266 – 382 – 399 – 390 – 363
4. Expliquer comment est effectuée la suppression de la valeur 25 de l'ABR de la question 1.b. Donner l'arbre obtenu après cette suppression.
 5. Un ABR A est dit de domaine plus petit (ou égal) qu'un ABR B si le plus petit élément de A est supérieur ou égal au plus petit élément de B , et si le plus grand élément de A est inférieur ou égal au plus grand élément de B . Écrire une fonction permettant de déterminer si un ABR A est de domaine plus petit qu'un ABR B .
 6. Écrire une fonction qui, étant donné un ABR A et deux entiers e et f tels que $e < f$, affiche dans l'ordre croissant tous les éléments x de l'arbre A tels que $e \leq x \leq f$. Le parcours doit prendre en compte la propriété d'ABR de l'arbre et ainsi éviter l'exploration de certains sous-arbres lorsque c'est possible.
 7. Écrire une fonction vérifiant si un arbre binaire A est un arbre binaire de recherche.

5 Arbres AVL

Écrire les 4 sous-programmes permettant d'appliquer les opérations nécessaires au rééquilibrage d'arbres AVL :

1. rotation gauche et rotation droite ;
2. rotation gauche-droite et rotation droite-gauche.

6 Table de hachage

On considère une table de hachage de taille $m = 11$ pour stocker des enregistrements. Les clés sont des entiers compris entre 0 et 500, et la fonction de hachage utilisée est $h(i) = i \bmod 11$. Quelle table obtient-on si on insère successivement les clés 10 – 22 – 31 – 4 – 15 – 28 – 88 – 17 – 59 en résolvant les collisions :

1. par chaînage séparé ;
2. par chaînage coalescent, sans réserve ;
3. par sondage linéaire (adressage ouvert).

7 Gestion de collection

On souhaite gérer une collection ayant au maximum $m = 5000$ œuvres d'art. Chaque œuvre est repérée par un numéro de catalogue variant entre 1 et 1 000 000, et pour chaque œuvre on indique son nom et son prix. On choisit de stocker ces informations dans une table de hachage de taille m , en gérant les collisions par chaînage séparé.

1. Définir les types nécessaires, et proposer une fonction de hachage portant sur les œuvres.
2. Écrire un sous-programme permettant d'ajouter dans la collection ou de mettre à jour une œuvre à partir de son numéro, son nom et son prix. Si l'œuvre figure déjà dans la collection (représentée par la table de hachage), on met à jour son prix ; sinon on l'insère dans la table.

8 Tas

1. Donner le tas-min (tas binaire minimum) obtenu après ajout dans l'ordre des éléments suivants :
12 – 8 – 15 – 6 – 3 – 19 – 23 – 7
2. Donner le tas-max (tas binaire maximum) obtenu après ajout dans l'ordre des éléments suivants :
25 – 12 – 7 – 35 – 9 – 39 – 43 – 18 – 5 – 21 – 14 – 22 – 28
3. Donner l'état du tas-max obtenu précédemment après suppression de sa valeur maximale (43).
4. Expliquer le processus permettant de trier la seconde liste de valeurs à partir du tas.