Travaux Dirigés d'informatique n°4

Algorithmique

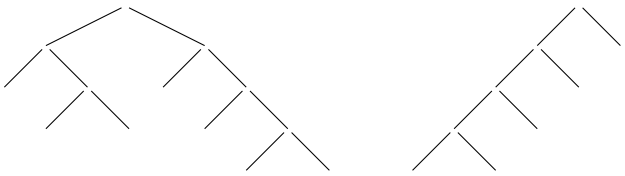
Codage d'un arbre complet et arbre binaire de recherches.

Exercice 1 codage d'un arbre complet

Un arbre binaire complet non étiqueté peut être codé en effectuant un parcours en profondeur en affichage préfixe. On décrit alors chaque nœud visité par:

- *0 pour un nœud interne,*
- *1 pour une feuille.*

1. *Donner le codage des arbres binaires pleins de hauteur* 0, 1 *et* 2, *ainsi que des arbres binaires suivants:*



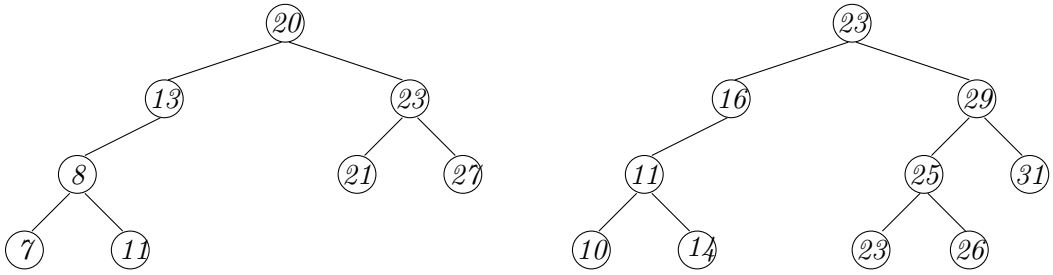
2. *Donner, lorsque c'est possible, l'arbre correspondant au codage suivant:*
   (a) *100101101010101*
   (b) *010101010101011*
   (c) *010011011011001*
   (d) *00000111101011001111.*

   Pour les codages ne donnant pas un arbre complet, pouvions nous le prévoir.

Exercice 2 fonction de codage et décodage

1. *Ecrire une fonction* `void AffichCodage(Arbre A)` *qui affiche le codage de l'arbre A;*
2. *écrire une fonction* `Arbre ConstArbre(void)` *qui construit un arbre à partir d'un codage que l'on saisit au fur et à mesure de la construction;*
3. *modifier les fonctions précédentes afin d'étiqueter les feuilles par des caractères. Pour ce faire, dans le codage, chaque 1 sera suivi du caractère associé.*

Exercice 3 Arbre binaire de recherche



1. *Les arbres de la figure ci-dessus sont-ils des arbres binaires de recherche?*

2. *Donner la liste des valeurs correspondantes aux parcours en profondeur en affichage préfixe, suffixe et infixe de l'arbre A.*

3. *Ecrire une fonction* `Arbre Recherche(Arbre A, int n)` *qui recherche un élément n dans l'arbre A. Elle renvoie l'adresse du nœud contenant l'élément s'il est présent, NULL sinon. On pourra supposer qu'il n'y a pas de doublon.*

Exercice 4 Ajout

1. *Construire un arbre binaire de recherche par ajout successif des valeurs: 42 18 16 12 23 45 14 13 32 62 11.*
   Donner plusieurs ordres d'ajout pour lesquel l'arbre a la plus grande hauteur et un pour lequel l'arbre a la plus petite hauteur.

2. *Ecrire une fonction* `void Ajout(Arbre *A, int n)` *qui effectue l'ajout de l'élément n dans l'arbre binaire de recherche A.*

Exercice 5 Suppression

1. *Supprimer successivement dans l'arbre construit dans la question précédente les valeurs: 62 16 18.*

2. *Ecrire une fonction* `Arbre Extraitmin(Arbre *A)` *qui effectue l'extraction du nœud contenant la plus petite étiquette de l'arbre binaire de recherche A. Elle renvoie l'adresse du nœud correspondant et NULL en cas d'echec, et l'arbre devra être modifié en conséquent. Que faut-il changer pour extraire le nœud contenant la plus grande étiquette.*

3. *Ecrire une fonction* `Extrait(Arbre A, int n)` *qui effectue l'extraction du nœud contenant une étiquette donnée d'un arbre binaire de recherche. Elle renvoie l'adresse du nœud et NULL en cas d'échec.*

Exercice 6 Vérification
Ecrire une fonction qui détermine si un arbre est un arbre binaire de recherche.