Algorithmique 1 : Devoir Surveillé 3

Arbres Binaires de Recherche

Durée : 60mn
Sans documents

Pour rappel sur les primitives du type abstrait `arbreBinaire` voir l'annexe A.

Exercice 3.1

Dessiner des arbres binaires de recherche de hauteur $2, 3, 4, 5$, et 6 pour le même ensemble de clés $\{1, 4, 5, 10, 16, 17, 21\}$.

Exercice 3.2

On suppose que les entiers compris entre 1 et 1000 sont disposés dans un arbre binaire de recherche, et on souhaite retrouver le nombre 363. Parmi les séquences suivantes, lesquelles ne pourraient pas être la séquence de noeuds parcourus ?

1. $2, 252, 401, 398, 330, 344, 397, 363$.
2. $924, 220, 911, 244, 898, 258, 362, 363$
3. $925, 202, 911, 240, 912, 245, 363$
4. $2, 399, 387, 219, 266, 382, 381, 278, 363$
5. $935, 278, 347, 621, 299, 392, 358, 363$

Exercice 3.3

Soit les types abstraits `arbreBinaire` et `tas`.

1. Quelle est la différence entre la structure des arbres binaires de recherche et la structure de tas ?
2. Peut-on afficher les clés d'un tas de n noeuds dans l'ordre en $O(n)$? Expliquer pourquoi ?
3. Donner un algorithme non récursif permettant d'afficher les clés d'un arbre binaire de recherche en ordre croissant.

Exercice 3.4 *Arbres à base*

Étant donné deux chaînes $a = a_0a_1...a_p$ et $b = b_0b_1...b_q$ où chaque a_i et chaque b_i se trouvent dans un ensemble ordonnée de caractères, on dit que la chaîne a est inférieure lexicographiquement à la chaîne b si

1. il existe un entier $j, 0 \leq j \leq min(p, q)$, tel que $a_i = b_i$ pour tout $i = 0, 1, ..., j - 1$ et $a_j < b_j$, ou
2. $p < q$ et $a_i = b_i$ pour tout $i = 0, 1, ..., p$.

Par exemple, si a et b sont des chaînes de bits, alors $10100 < 10110$ d'après la règle 1 (en prenant $j = 3$) et $10100 < 101000$ d'après la règle 2.
Soit S un ensemble de chaînes binaires distinctes dont les longueurs ont pour somme n. Montrer comment utiliser un arbre à base pour trier S lexicographiquement en $O(n)$.
Pour l'exemple de la figure 1, la sortie du tri serait la séquence $0, 011, 10, 100$, et 1011.

Annexe A : rappels du cours

TYPE ABSTRAIT ARBREBINAIRE

```
arbreBinaire= curseur; sommet= curseur; fonction getValeur(val S:sommet):objet;
fonction filsGauche(val S:sommet):sommet;
fonction filsDroit(val S:sommet):sommet;
fonction pere(val S:sommet):sommet; fonction setValeur(ref S:sommet, val x:objet):vide;
fonction ajouterFilsGauche(ref S:sommet, val x:objet):vide;
fonction ajouterFilsDroit(ref S:sommet, x:objet):vide;
fonction supprimerFilsGauche(ref S:sommet):vide;
fonction supprimerFilsDroit(ref S:sommet):vide;
fonction detruireSommet(ref S:sommet):vide;
fonction creerArbreBinaire(val Racine:objet):sommet;
```

TYPE ABSTRAIT TAS Pour rappel un tas est un containeur et un arbre binaire, il dispose donc des primitives des arbres binaires ainsi que ceux d'un containeur.

```
fonction valeur(ref T: tas de objet): objet;
fonction ajouter(ref T: tas de objet, val v: objet): vide;
fonction supprimer(ref T: tas de objet): vide;
fonction creerTas(ref T: tas, val v: objet): vide;
fonction detruireTas(ref T: tas): vide;
```

TYPE ABSTRAIT CONTAINEUR

```
valeur(val c: conteneur): objet;
creerConteneur(ref c: conteneur): vide;
ajouter(ref c: conteneur, x: objet): vide;
supprimer(ref c: conteneur, x: objet): vide;
detruireConteneur(ref c: conteneur): vide;
```

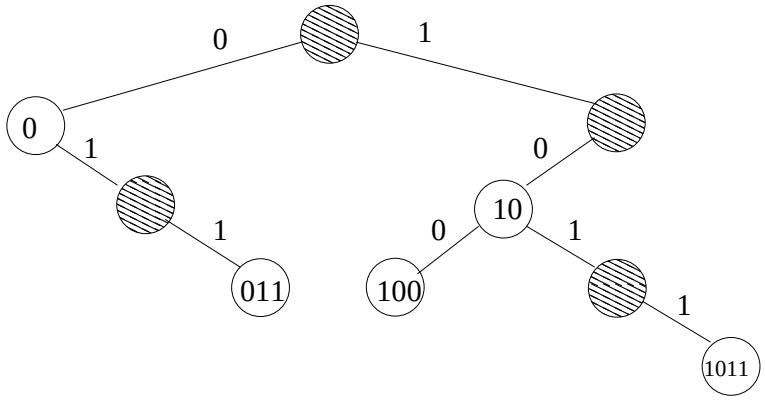

FIG. 1 – Un arbre à base contenant les chaînes de bits $1011, 10, 011, 100$, et 0.