Université Bordeaux 1 Licence Informatique 2011/2012

Algorithmique 1

DS 4

Documents **non** autorisés

Ensemble du cours

Exercice 4.1

Question du cours Donner un exemple d'un AVL dont la suppresion d'un sommet nécessite plus d'une rotation (simple ou double) pour maintenir la propriété d'AVL, indiquer aussi le sommet à supprimer.

Exercice 4.2

Structure de données Dans cet exercice, on se propose de manipuler de figures géométriques régulières et irrégulières, caractérisées par leur nombre de côtés.

Définire les structures de données suivantes :

1. triangle
2. rectangle
3. pentagone
4. hexagone
5. dégagone
6. dodégagone

Exercice 4.3

Table de hachage On veut stocker dans une table de hachage de taille m = 13 les figures suivantes :

1. carreA de taille 3
2. rectangleA de taille 2 3
3. carreB de taille 2
4. triangle de taille 3 4 5
5. pentagone de taille 5
6. hexagone de taille 4
7. degagone de taille 3
8. dodega de taille 2
9. isocele de taille 3 2
10. equilateral de taille 3

En utilisant l'adressage ouvert [1] avec la fonction de hachage `h(x)`,

$$h(x) = (asc(x_1) * 2^{l-1} + asc(x_2) * 2^{l-2} + ... + asc(x_l)) \ \ mod \ \ m$$

$$x = "x_1 x_2 \dots x_l"$$

Pour rechercher une nouvelle case libre en cas de collision, on utilisera une seconde fonction

$$h'(x) = 1 + h(x) \ mod \ (m-1)$$

En cas de nouvelle collision, on appliquera la recherche linéaire. Les clés sont les identifiants de figures.

Exercice 4.4
Construction de tas

1. Construire un tas (min) avec les identifiants de figures successives stokées dans la table ci-dessus, on utilisera comme relation d'ordre, le périmètre de chaque figure

   $$F1 \leq F2 \equiv perimetre(F1) \leq perimetre(F2)$$

   On détaillera les cinq premières étapes de construction.
2. Faire la trace de l'exécution de la fonction supprimer appliquée au tas construit ci-dessus,

Exercice 4.5
Construction de AVL

1. Construire un AVL avec les identifiants de figures successives stokées dans la table ci-dessus, on utilisera comme relation d'ordre, le périmètre de chaque figure

   $$F1 \leq F2 \equiv perimetre(F1) \leq perimetre(F2)$$

   On détaillera les cinq premières étapes de construction.
2. Faire la trace de l'exécution de la fonction supprimer la racine appliquée à l'AVL construit ci-dessus.

1. En adressage ouvert tous les éléments sont conservés dans la table de hachage elle-même.

Annexe A : **Type abstrait tableHash**

```
   fonction creerTableHachage(ref T: tableHash de cle, ref h: fonction): vide;
fonction charcher(ref T: tableHash de cle, val v: cle): curseur;
ajouter(ref T: tableHash de cle, val v: cle): booleen;
supprimer(ref T: tableHash de cle, val v: cle): vide;
```

Annexe B : **Adressage ouvert**

```
tableHash de cle= structure
                    table: tableau[0..m-1] de cle;
                    h: fonction(val v: cle): curseur;
                    s: fonction(val v: cle, i: entier): curseur;
                  finstructure
```

Annexe C : **Type abstrait tas.**

```
fonction valeur(val T: tas d'objet): objet;
fonction ajouter(ref T: tas d'objet, val v: objet): vide;
fonction supprimer(ref T: tas d'objet): vide;
fonction creerTas(ref T: tas d'objet, val v: objet): vide;
fonction detruireTas(ref T: tas d'objet): vide;
```

Annexe D : **Implémentation du type abstrait tas.**

```
tas=structure
        arbre:tableau[1..tailleStock] d'objet;
        tailleTas:entier;
     finstructure;
curseur=entier;
sommet=entier;

fonction getValeur(val T: tas d'objet, val s: sommet): objet;
fonction valeur(val T: tas d'objet): objet;
fonction filsGauche(val s: sommet): sommet;
fonction filsDroit(val s: sommet): sommet;
fonction pere(val s: sommet): sommet;
fonction setValeur(ref T: tas d'objet, val s: sommet, val x:objet): vide;
fonction tasPlein(val T: tas d'objet): booleen
fonction creerTas(ref T: tas d' objet, val racine: objet): vide;
fonction ajouter(ref T: tas d'objet, val v: objet): vide;
fonction supprimer(ref T: tas d'objet): vide;
```

Annexe E : **Type abstrait arbreAVL**

```
type celluleAVL= structure
                   info: objet;
                   hauteur: entier;
                   gauche: sommetAVL;
                   droit: sommetAVL;
                   pere: sommetAVL;
                 finstructure
sommetAVL= ^celluleAVL;
arbreAVL= sommetAVL;

fonction getHauteur(ref s: sommetAVL): entier;
fonction setHauteur(ref s: sommetAVL): entier;
fonction rotationDroite(ref s: sommetAVL): vide;
fonction rotationGauche(ref s: sommetAVL): vide;
fonction ajouter(ref s: sommetAVL, val e: objet): vide;
fonction supprimer(ref s: sommetAVL): booleen;
fonction equilibrerApresInsertion(ref s: sommetAVL, val cote: entier): vide;
fonction equilibrerApresSuppression(ref s: sommetAVL): vide;
fonction equilibrerUnSommet(ref p,s : sommetAVL): vide;
```