Université Bordeaux 1 LST Mention Informatique Semestre 3 2008-2009

Algorithmique 1 : Devoir Surveillé 4

Arbres

Durée : 1h20
Sans documents.
Les primitives nécessaires à l'écriture des fonctions demandées sont données en annexe.

1 Tas et file de priorité

Exercice 4.1 *Tas maximier*
Dessinez le maximier obtenu après que les opérations suivantes ont été effectuées sur un tas maximier initialement vide.

```
var T: tas d'entiers
creerTas(T)
ajouter(T,1);
ajouter(T,5);
ajouter(T,2);
ajouter(T,6);
setValeur(T,2,4)
ajouter(T,8);
supprimer(T);
ajouter(T,7);
ajouter(T,3);
```

Exercice 4.2
Ecrire une fonction pour supprimer l'élément d'indice `d` dans un maximier.

Exercice 4.3
Quels indices pourraient être occupés par la troisième plus grande clé dans un maximier de taille `32`.

2 Dictionnaires

Exercice 4.4 *Insertion*
Dessiner le dictionnaire résultat de la suite d'opérations donnée :

```
var d: dictionnaire;
creerDictionnaire(d);
ajouter(d,"abois");
ajouter(d,"abord");
ajouter(d,"aborder");
supprimer(d,"abord" );
ajouter(d,"chic");
ajouter(d,"suisse");
supprimer(d,"suis");
```

Exercice 4.5 *Recherche*
Ecrire une fonction qui renvoie le nombre de mots contenu dans un dictionnaire ayant un prefixe donné.

```
fonction nmb_mots_prefixe_donne(var d: dicionnaire, var prefixe: mot): entier
```

3 Tables de hachage

Exercice 4.6 *Adressage chaîné*
Soit une table de hachage d'adressage ouvert :

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

Ecrire les primitives d'accès, de recherche et de suppression :

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

Exercice 4.7
Quelle fonction d'adressage doit-on utiliser si l'on sait à l'avance que les valeurs de clé appartiennent toutes à un intervalle d'assez petite amplitude ?

4 Arbres AVL

Exercice 4.8
Dessinez tous les arbres AVL qui contiennent les valeurs `1,2,3,4,5`.

5 Arbres équilibrés

Exercice 4.9 *Arbres 2-3*
Définir un type de données permettant de construire des arbres 2-3. Ecrire une fonction qui parcourt un arbre 2-3 en ordre infixe.

Annexe

```
tas=structure
        arbre:tableau[1..tailleStock] d'objet;
        tailleTas:entier;
     finstructure;
curseur=entier;
sommet=entier;
fonction valeur(ref T:tas d'objet): objet;
fonction ajouter(ref T:tas d'objet, val v:objet):vide;
fonction supprimer(val T:tas d'objet):vide;
fonction creerTas(ref T:tas d'objet,val:v:objet):vide;
fonction detruireTas(ref T:tas d'objet):vide;
fonction setValeur(ref T:tas d'objet, val s: sommet, val x: objet): vide;

cellule=structure
         info:objet;
         gauche:^cellule;
         droit:^cellule;
         pere:^cellule;
       finstructure
sommet=^cellule;
fonction appartient(ref d: dictionnaire, val M: mot) : booleen;

fonction creerDictionnaire(ref d: dictionnaire): vide;
fonction ajouter(ref d: dictionnaire, val M: mot): vide;
fonction supprimer(ref d: dictionnaire, val M: mot): vide;
fonction detruireDictionnaire(ref d: dictionnaire): vide;
```