Université Bordeaux 1 LST Mention Informatique Semestre 3 2010-2011

Algorithmique 1 : Devoir Surveillé 4

Tas, Dictionnaires et Tables de hachage

Durée : 60mn
Sans documents

1 Tas et file de priorité

Exercice 4.1 *Tas*

1. Quels sont les nombres minimum et maximum d'éléments dans un tas de hauteur h.
2. La séquence `{23,17,14,6,13,10,1,5,7,12}` forme-t-elle un tas ?

Exercice 4.2 *File de priorité*
Soit le problème récurrent de gestion d'une piste d'atterrissage des avions vu en TD.
Donner un rappel comment la notion de file de priorité permet d'amorcer la gestion de la piste.

2 Dictionnaires

Exercice 4.3 *Dictionnaires*
Soit une table de symboles qui contient les mots clés :
`{"acos", "char", "define", "exp", "float", "atan", "ceil", "floor"}`
qu' on souhaite représenter par un dictionnaire.

1. Dessiner l'état du dictionnaire après l' ajout des mots dans l'ordre.
2. Dessiner l'état du dictionnaire après la suppression du mot "acos".
3. Ecrire la primitive `appartient` dans l'implémentation du type abstrait Dictionnaire par le type arbreBinaire.

3 Tables de hachage

Exercice 4.4 *Adressage ouvert*
Soit une table de symboles qui contient les mots clés :
`{"acos", "char", "define", "exp", "float", "atan", "ceil", "floor"}`.
On souhaite construire une table de hachage T_{ouv} de taille $M = 11$ en utilisant l' adressage ouvert. On utilisera la fonction de hachage $h(x)$, $h("x_1x_2 \dots x_m") = asc(x_1)\ mod\ M$,
ou $asc(x_1)$ est le code ascii du premier caractère de la clé $x = "x_1x_2 \dots x_m"$.
Pour resoudre les collisions on utilisera une fonction de sondage linéaire $s(x, i)$, $s(x, i) = (h(x) + i)\ mod\ M$, $0 \leq i \leq M - 1$.
Les codes ascii utilisés sont dans l' intervalle : $asc('a') = 97, asc('b') = 98, \dots, asc('z') = 122$.

1. Dessiner le contenu de la table de hachage T_{ouv} après l' insertion des clés dans l'ordre `{"acos", "char", "define", "exp", "float", "atan", "ceil", "floor"}`.
2. Donner l' implémentation des primitives `ajouter` et `supprimer`.

3. Dessiner le contenu de la table de hachage T_{ouv} après l' appel à `supprimer("acos")`.
4. Ecrire une fonction pour afficher toutes les clés contenues dans la table de hachage T_{ouv} en ordre lexicographique d' après leur premier caractère.
   Par exemple, pour T_{ouv} après la suppression de `"acos"`, la fonction affichera :
   `"atan", "char", "ceil", "define", "exp", "float"`.

Exercice 4.5 *Adressage chaîné*
Pour la table de symboles de l' exercice précédent, montrer comment réaliser une table de hachage T_{ch} d' adressage chaîné de taille $M = 11$.
On utilisera la même fonction de hachage $h(x)$, $h(\text{"}x_1x_2 \dots x_m\text{"}) = asc(x_1)\ mod\ M$, mais pour resoudre les collisions on utilisera le chaînage.
Dessiner le contenu de la table de hachage T_{ch} après l' insertion des clés dans l'ordre :
`{"acos", "char", "define", "exp", "float", "atan", "ceil", "floor"}`.

Annexe A : **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;
```

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 B **Type abstrait file de priorité**

Primitive supplémentaire :

```
fonction changeValeur(ref T: tas d'objet, val s: sommet, val v: objet): vide;
```

Annexe C : **Type abstrait Dictionnaire**

```
fonction appartient(val 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;
```

Implémentation du type abstrait Dictionnaire dans le type arbreBinaire

```
type dico= structure
             a: sommet;
             p: sommet;
           finstructure
```

Annexe D : **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;
```

Adressage chaîné

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

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
```