table des matières

Université Bordeaux 1

Licence Semestre 3 - Algorithmes et structures de données 1

Dernière mise à jour effectuée le 1 Septembre 2013

Tas

1. Définition

Définition 5.1 Un tas max (resp.tas min) *T* est un arbre binaire quasi-parfait étiquetté par des objets comparables (ie : il existe un ordre total) tel que tout noeud a une étiquette plus grande (resp. plus petite) que ces fils.

Propriété 5.2 La hauteur d'un tas est O(log(n)).

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 d'objet): objet;
// renvoit l'objet stocké à la racine de l'arbre
fonction ajouter(ref T:tas de objet,
                 val v:objet):vide;
// ajoute l'objet dans le tas
fonction supprimer(val T:tas de objet):vide;
// suppression de la racine et tassement de l'arbre
fonction creerTas(ref T:tas,val:v:objet):vide;
fonction detruireTas(ref T:tas):vide;
```

Création d'un tas à partir d'une liste d'entiers non vide

```
fonction tableauTas(ref L:liste d'entier):tas d'entier;
  var T:tas d'entier;
  début
    premier(L);
    creerTas(T,valeur(L));
    suivant(L);
    tantque valeur(L)!=NULL faire
      ajouter(T,valeur(L));
      suivant(L);
    fintantque
    retourner(T)
  fin
```

Tri par tas

Propriété 5.3 Si *n* est le nombre de sommets du tas, on obtient la liste triée

en O(n log(n)).

```
fonction listeTas(val T:tas d'entier):liste de entier;
  var L: listeSC de entier;
  var r:entier;
  début
    creerListe(L);
    tantque !estFeuille(T) faire
      r= valeur(T);
      supprimer(T);
      insérerEnTete(L,r);
    fintantque
    insérerEnTete(L,valeur(T));
    detruireTas(T);
    retourner(L);
  fin
```

Propriété 5.4. Le tri qui consiste à
1. Construire un tas
2. Extraire la liste triée

a une complexité O(n log(n)).

---

2. Ajouter et supprimer dans un tas Max

Pour ajouter une valeur v dans un tas,
- on crée une nouvelle feuille dans l'arbre quasi-parfait en affectant lui affectant la valeur v.
- Soit ($r=s_0,...,s_k$) le chemin de la racine à cette nouvelle feuille. Pour i allant de k à 1 si la valeur stockée dans s_i est plus grande que celle stockée dans s_{i-1} alors on échange ces valeurs.

Un exemple est ici
Pour supprimer la valeur dans un tas,
- on remplace la valeur de la racine par la valeur v de la dernière feuille de l'arbre.
- on supprime cette feuille.
- on fait descendre la valeur v dans l'arbre par échange avec la valeur la plus grande d'un des fils si celle ci est plus grande.

Un exemple est ici

---

3. Implémentation

On utilise la numérotation des noeuds dans le parcours hiérarchique. La racine est numérotée 1. Le fils gauche (resp. droit) de la cellule numéro i a pour numéro 2*i (resp. 2*i+1). On utilise cette propriéte pour représenter un tas dans un tableau. De plus dans les opérations d'ajout et suppression des valeurs, on devra pouvoir parcourir l'arbre. Un curseur sera donc utile.

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

De ce fait, les primitives arbre binaire prennent comme paramètre un tas et non un sommet. On a

```
sommet=entier;
```

Les fonction ajouter et supprimer sont spécifiques au tas.

- accès

```
fonction getValeur(ref T:tas d'objet;val s:sommet):objet;
  début
    retourner(T.arbre[s]);
  fin;
fonction valeur(ref T:tas d'objet):objet;
  début
    retourner(T.arbre[1]);
  fin
fonction filsGauche(val s:sommet):sommet;
  début
    retourner(2*s);
  fin
fonction filsDroit(val s:sommet):sommet;
  début
    retourner(2*s+1);
  fin
fonction pere(val s:sommet):sommet;
  début
    retourner(partieEntiere(s/2));
  fin
fonction tasPlein(ref T:tas d'objet):booleen;
  début
    retourner(T.tailleTas==tailleStock)
  fin
```

- modification

```
fonction setValeur(ref T:tas d'objet;val s:sommet;val x:objet):vide;
  début
    T.arbre[s]=x;
  fin
fonction créerTas(ref T:tas d'objet; val x:objet):vide;
  début
    T.arbre[1]=x;
    T.tailleTas=1;
  fin
```

- Gestion du tas

```
fonction ajouter(ref T:tas d'objet, val v:entier):vide
  début
    T.tailleTas=T.tailleTas+1;
    T.arbre[T.tailleTas]=v;
    reorganiseTasMontant(T,tailleTas);
  fin

fonction reorganiseTasMontant(ref T: tas d'objet;val x:sommet):vide;
  var p:sommet;
  var signal:booléen;
  début
    p=père(x);
    signal=vrai;
    tantque x!=1 et signal faire
      si getValeur(T,x)>getValeur(T,p) alors
         échanger(T.arbre[p],T.arbre[x])
         x=p;
         p=père(x);
      sinon
         signal=faux
      finsi
    fintantque
  fin

fonction supprimer(ref T:tas d'objet):vide;
  var r:entier;
  début
```

```
    r=valeur(T);
    T.arbre[1]=T.arbre[T.tailleTas];
    T.tailleTas=T.tailleTas-1;
    reorganiseTasDes(T,1);
  fin

fonction reorganiseTasDesc(ref T:tas d'objet,x:sommet):vide;
  var g,d:sommet;
  début
    g= filsGauche(x);
    d= filsDroit(x);
    si g!=NIL alors
      si d!=NIL alors
        si getValeur(T,d)>getValeur(T,g) alors
          g=d;
        finsi;
      finsi
      si getValeur(T,x)<getValeur(T,g) alors
        échanger(T.arbre[x],T.arbre[g]);
        reorganiseTasDesc(T,g)
      finsi
    finsi
  fin
```

---

4. Files de priorité

Définition 7.5 Une file de priorité est un tas dans lequel on a la possibilité de modifier les valeurs des sommets.

On dispose donc d'une primitive supplémentaire que l'on implémente et qui enrichit le type tas.

```
fonction changeValeur(ref T:tas d'objet,val s:sommet,val v:objet):vide;
  début
    setValeur(T,s,v);
    si v> getValeur(T,pere(s)) alors
      reorganiseTasMontant(T,s)
    sinon
      si v<getValeur(T,filsDroit(s))
          ou v<getValeur(T,filsGauche(s)) alors
        reorganiseTasDesc(T,s)
      finsi
    finsi
  fin
```

Propriété 7.6 Le changement de valeur dans une file de priorité de taille n s'effectue en O(log(n)).