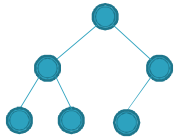
6-Tas

- Définition
- Ajouter et supprimer dans un tas Max
- Implémentation
- Files de priorité

6-1 : Définitions

Définition 6.1 : Un tas max (resp.tas min) *T* est un arbre binaire quasi-parfait étiqueté par des objets comparables (ie : il existe un ordre total) tel que tout nœud a une étiquette plus grande ou égale (resp. plus petite) que ses fils.

Propriété 6.1 : La hauteur d'un tas est O(log(n)).



L'ajout d'un élément se fait en conservant la structure ABQP

6-1 : Définitions

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;
// renvoie 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;
```

6-2-1: Ajouter dans un tas Max

Pour ajouter une valeur v dans un tas, on crée une nouvelle feuille dans l'arbre quasi-parfait en lui affectant la valeur v.

- Soit $(r=s_0,\ldots,s_k)$ 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. On fait une réorganisation montante : un exemple est ici

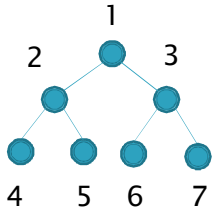
6-2-2: Supprimer dans un tas Max

Dans un tas supprimer un élément consiste toujours à supprimer la racine. 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.

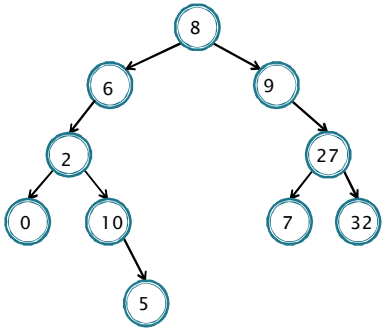
- On supprime cette feuille.
- On fait descendre la valeur v dans l'arbre par échange avec la valeur la plus grande d'un des fils si celle ci est plus grande. On fait une réorganisation descendante : un exemple est ici

6-3 : Implémentation dans un Tableau

On utilise la numérotation des nœuds dans le parcours hiérarchique d'un AB.
La racine est numérotée `1`.
Le fils gauche de la cellule numéro i a pour numéro `2*i`
Le fils droit de la cellule numéro i a pour numéro `2*i +1`



6-3 : Implémentation dans un Tableau



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
8	6	9	2			27	0	10					7	32				5		

6-3 : Implémentation dans un Tableau

On utilise cette propriété 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.

6-3 : Implémentation dans un Tableau

Les fonction ajouter et supprimer sont spécifiques au tas.

- accès

```
fonction getValeur(ref T:tas d'objet; Val s:sommet):objet;
  debut
    retourner(T.arbre[s]);
  fin;
fonction valeur(ref T:tas d'objet):objet;
  debut
    retourner(T.arbre[1]);
  fin

 fonction filsGauche(val s:sommet):sommet;
   debut
      retourner(2*s);
   fin
```

6-3 : Implémentation dans un Tableau

Les fonction ajouter et supprimer sont spécifiques au tas.

- accès

```
fonction filsDroit(val s:sommet):sommet;
  debut
    retourner(2*s+1);
  fin
fonction pere(val s:sommet):sommet;
  debut
    retourner(partieEntiere(s/2));
  fin

fonction tasPlein(ref T:tas d'objet):booleen;
  debut
     retourner(T.tailleTas==tailleStock)
   fin
```

6-3 : Implémentation dans un Tableau

- modification

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

fonction creerTas(ref T:tas d'objet; val x:objet):vide;
  debut
    T.arbre[1]=x;
    T.tailleTas=1;
  fin
```

6-3 : Implémentation dans un Tableau

- Gestion du tas

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

6-3 : Implémentation dans un Tableau

- Gestion du tas

```
fonction reorganiseTasMontant(ref T: tas d'objet;
                              val x:sommet):vide;
  var p:sommet;
  var signal:booléen;
  debut
    p=pere(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=pere(x);
      sinon
         signal=faux
      finsi
    fintantque
  fin
```

6-3 : Implémentation dans un Tableau

- Gestion du tas

```
fonction supprimer(ref T:tas d'objet):vide;
  var r:objet;
  debut
    T.arbre[1]=T.arbre[T.tailleTas];
    T.tailleTas=T.tailleTas-1;
    reorganiseTasDesc(T,1);
  fin
```

6-3 : Implémentation dans un Tableau

- Gestion du tas

```
fonction reorganiseTasDesc(ref T:tas d'objet,
                           val x:sommet):vide;
  var g,d:sommet;
  debut
    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
```

6-4 : Files de priorité

Définition 6.2 : 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 :

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

Complexité :