

**Objectifs**

- À la fin de cette séance, vous devriez être capable de :
- implémenter une structure de tas dans un tableau ;
  - manipuler cette structure ;
  - maîtriser le tri par tas.

Nous avons vu en cours la structure de *tas* (arbre tassé ordonné), qui permet d’implanter efficacement une file à priorité de taille  $n$  sous forme d’un arbre de hauteur  $\lceil \log_2 n \rceil$ .

Nous avons pour l’instant décrit les opérations sur cette structure sous la forme de fonctions sur des arbres. L’objet de ce TD est de montrer comment un arbre binaire tassé peut être représenté sous forme de tableau, et de traduire les opérations du tas dans cette représentation.

**Attention :** dans le cours nous avons considéré un « tas min », dans lequel la clé de chaque nœud doit être inférieure à celles de ses fils.  
 Dans ce TD, pour que le tri par tas fonctionne, il faudra considérer un « tas max », dans lequel la clé de chaque nœud doit être **supérieure** à celles de ses fils.  
 Les opérations étudiées en cours s’adaptent facilement pour tenir compte de cet ordre sur les nœuds.

**Implantation du tas dans un tableau**

Soit un arbre binaire tassé de  $n$  nœuds et de hauteur  $h$ .

On place les étiquettes des nœuds dans un tableau (de taille au moins  $n$ ), dans l’ordre du parcours par niveaux (du niveau 0 au niveau  $h$  et de gauche à droite dans chaque niveau). Comme la forme de l’arbre est complètement déterminée par  $n$ , on a ainsi une correspondance univoque entre les tas et les tableaux.

Par ailleurs, pour pouvoir insérer de nouveaux éléments, on prévoit un tableau d’une taille suffisante  $n_{max}$ , et on maintient donc :

- $F$  : un tableau sur  $[0 \dots n_{max} - 1]$  d’entiers qui sont leur propre clé.
- $n$  : un entier de  $[0 \dots n_{max}]$ , qui donne le nombre d’éléments dans le tas

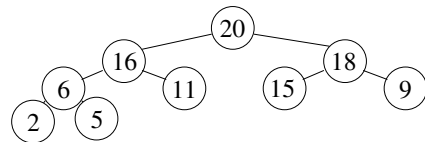
**Exemple :**

Le tas ci-contre est représenté par le tableau :

20	16	18	6	11	15	9	2	5	...
----	----	----	---	----	----	---	---	---	-----

avec  $n = 9$ .

(On remarque que la racine du tas contient l’élément maximal.)



**Exercice 1 : Indices et hauteurs**

Les indices commencent à 0.

1. Quel est l’indice de la racine ?
2. Si un nœud interne est situé à l’indice  $i$ , à quel indice est situé son fils gauche ? son fils droit ? son père ?
3. Quel est l’indice maximal d’un nœud interne ? À quelle condition un nœud d’indice  $i$  possède-t-il un fils gauche ? un fils droit ?
4. Quel est le niveau d’un nœud d’indice  $i$  ?
5. Quelle est la hauteur maximale d’un sous-arbre dont la racine est à l’indice  $i$  ?

**Exercice 2 : Opérations**

1. Reprenez (réécrivez) les opérations **Insérer** et **Extraire\_max** telles que vous les avez vues en cours
2. Traduisez chaque mot, chaque ligne ... en fonction de  $F$  et de  $n$ .

## Tri par tas

On veut trier un tableau  $T$  de  $N$  entiers (rangés dans les cases d'indices  $[0 \dots N - 1]$ ). On applique le principe suivant :

1. Dans un premier temps on modifie  $T$  de sorte qu'il devienne un tas de  $N$  éléments.
2. Dans un deuxième temps, on modifie à nouveau  $T$  de sorte que ses éléments  $y$  soient rangés en ordre croissant.

### Exercice 3 : Version “naïve”

Au cours de la première étape, un indice  $k$  parcourt les valeurs de  $0$  à  $N - 1$  et on maintient l'invariant suivant :

- $T[0 \dots k]$  est une permutation de sa valeur initiale ;
- $T[k + 1 \dots N - 1]$  est inchangé ;
- $T[0 \dots k]$  est un tas.

1. Dessinez cet invariant.
2. Écrivez l'algorithme de création du tas en utilisant une procédure similaire à **Insérer**.

Au cours de la deuxième étape, l'indice  $k$  parcourt les valeurs de  $N - 1$  à  $0$  et on maintient l'invariant suivant :

- $T$  est une permutation de sa valeur initiale ;
- $T[0 \dots k]$  est un tas.
- $T[k + 1 \dots N - 1]$  comporte les  $N - k$  plus grandes valeurs du  $T$  initial, en ordre croissant.

3. Dessinez cet invariant.
4. Écrivez l'algorithme de cette étape en utilisant une procédure similaire à **Extraire\_max**.
5. Évaluez la complexité de chacune des deux étapes et en déduire celle du tri par tas.

### Exercice 4 : Optimisation de la création du tas initial

Le tableau initial  $T$  représente un arbre binaire tassé. Les feuilles de cet arbre sont des arbres tassés ordonnés. L'idée est de donner progressivement la propriété « ordonné » à  $T$ , en procédant par niveaux, des feuilles vers la racine : le traitement d'un niveau consiste à faire percoler vers le bas (si nécessaire) les racines des sous-arbres de ce niveau.

1. Évaluez le coût de cette nouvelle procédure et montrer le gain par rapport à la version précédente.
2. Écrivez cette nouvelle procédure.

**Membres du groupe et responsabilités**

	Écrire les algorithmes sans ambiguïtés en respectant les primitives autorisées	Gérer le temps
	Évaluer l'efficacité des algorithmes	Réguler les prises de parole
	Tester les algorithmes écrits Déterminer des invariants	S'assurer que tous comprennent et sont en accord avec le rendu
	Représenter schématiquement les invariants des algorithmes	Rédiger le rendu

**Exercice 3 : Version “naïve” du tri par tas**

1. Dessinez l'invariant de la première étape du tri par tas.
2. Écrivez l'algorithme de création du tas en utilisant une procédure similaire à **Insérer**.
3. Dessinez l'invariant de la deuxième étape du tri par tas.
4. Écrivez l'algorithme de cette étape en utilisant une procédure similaire à **Extraire\_max**.