CHAP 4 :

Les tas (Heaps)

Université Sétif 1
Faculté des sciences
Département d'informatique

Algorithmique et Structures de Données

2017-2018 / Dr. L.Douidi

Introduction

Pour implémenter une file d'attente avec priorité, souvent utilisée dans les systèmes d'exploitation, on peut utiliser :

- Une file d'attente ordinaire (sans priorité), l'insertion sera alors simple (à la fin) en O(1), mais le retrait nécessitera la recherche de l'élément le plus prioritaire, en O(n).
- Un tableau (ou une liste) trié où le retrait sera en O(1) (le premier élément), mais l'insertions nécessitera O(n).

- Les tas apportent la solution à ce problème.

Définition

Un tas (*heap en anglais) est un arbre qui vérifie les deux propriétés suivantes :*

1. C'est un arbre binaire complet c'est-à-dire un arbre binaire dont tous les niveaux sont remplis sauf éventuellement le dernier où les éléments sont rangés le plus à gauche possible.

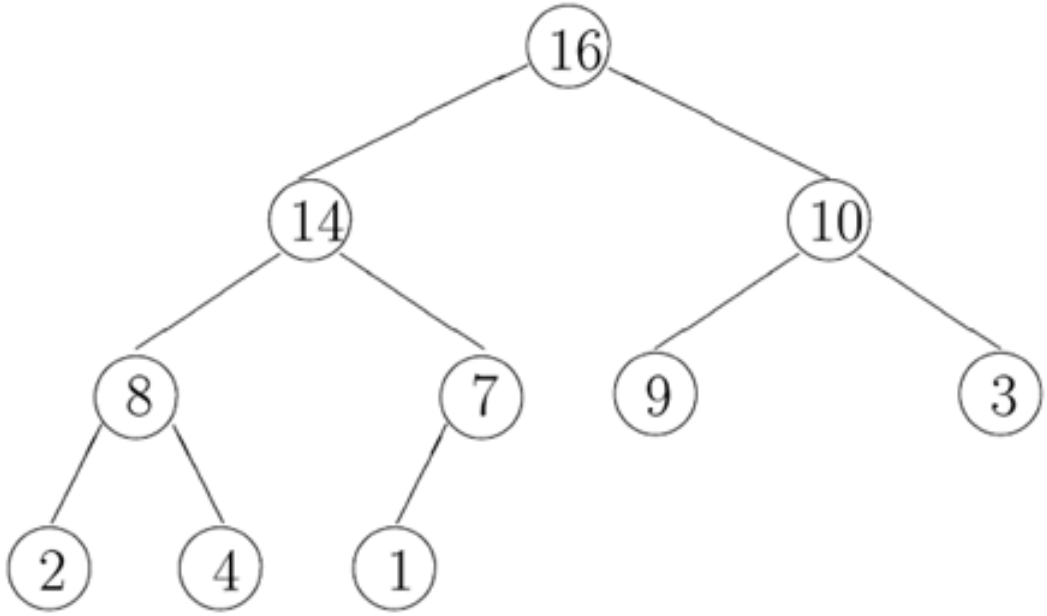
2. il est **ordonné en tas**

   On dit qu'un arbre est ordonné en tas lorsque la propriété suivante est vérifiée : les nœuds sont ordonnés par leurs clés respectives,

- On parle d'un **tas-max** (**Max-heap)** si tout nœud a une valeur plus grande ou égale à celles de ses deux fils. Pour tous A et B nœuds de l'arbre tels que B soit un fils de A **clé(A)** $\geq$ **clé(B)**

  Dans ce cas, le *plus grand* élément de l'arbre est à la *racine*

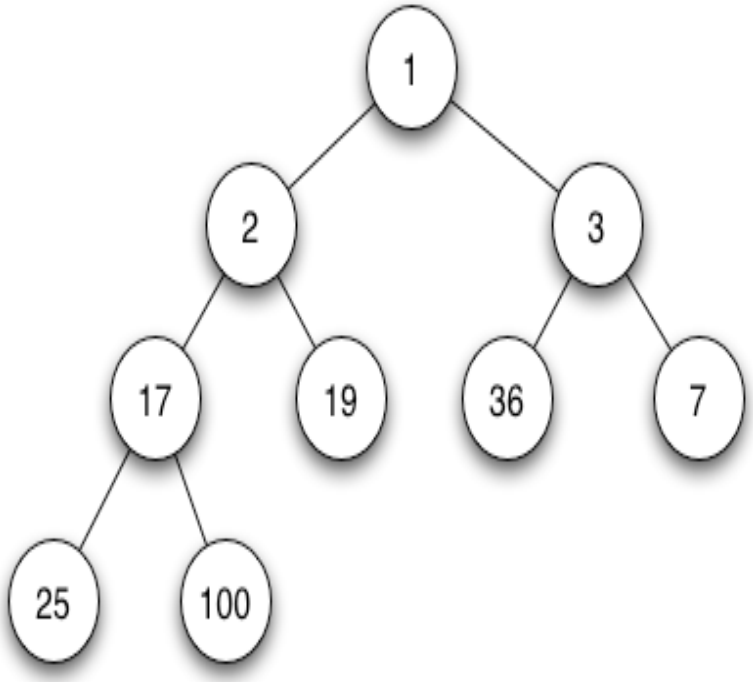
  Exemple d'un tas-max :



Tas-min

- On parle d'un **tas-min** (**Min-heap)** si tout nœud a une valeur plus petite ou égale à celles de ses deux fils.

  Dans ce cas, le plus petit élément de l'arbre est à la racine.
- Exemple :



Ils sont ainsi très utilisés pour implémenter les **files à priorités** car ils permettent des insertions en temps logarithmique et un accès direct au plus grand élément.

- Ils sont ainsi très utilisés pour implémenter les **files à priorités** car ils permettent des insertions en temps logarithmique et un accès direct au plus grand élément.
- L'efficacité des opérations effectuée sur des tas est très importante dans de nombreux algorithmes sur les graphes.
- ***En d'autre terme :***

  Un tas (heap) est un arbre binaire T qui emmagasine une collection de clés (ou paires clé-élément) comme nœuds internes et qui satisfait les deux propriétés suivantes:

➢ ***Propriété d'ordre:***

  - *clé(parent) ≤ clé(enfant) pour le tas-min*
  - *clé(parent) ≥ clé(enfant) pour le tas-max*

➢ ***Propriété structurelle:*** *arbre binaire complet*

Opérations sur les tas

- Pour coder une file d'attente avec priorité par un tas, on doit définir les opérations d'ajout et de retrait.

I Ajout:

- Pour ajouter un nouvel élément dans la file avec priorité c-à-d dans le tas on doit :

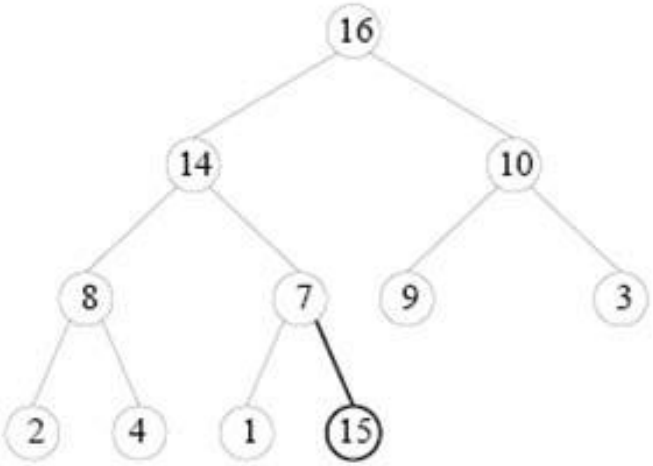
1. Créer un nœud contenant la valeur de cet élément,
2. Attacher ce nœud dans le dernier niveau dans la première place vide le plus à gauche possible (créer un nouveau niveau si nécessaire). On obtient toujours un arbre binaire complet mais pas nécessairement un tas.

- 3. Comparer la clé du nouveau nœud avec celle de son père et les permuter si nécessaire,

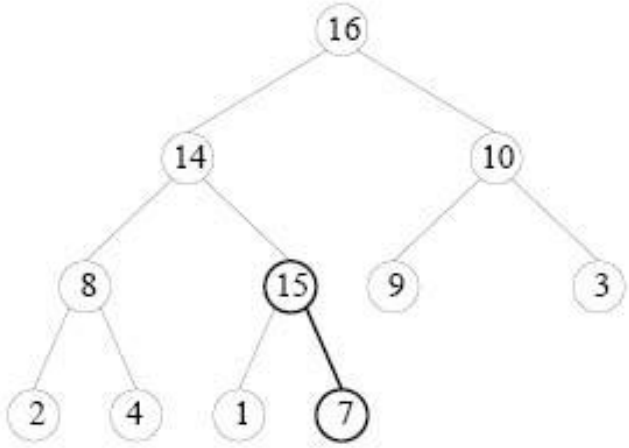
  puis recommencer le processus jusqu'il n'y ait plus d'éléments à permuter.

Ajout d'élément

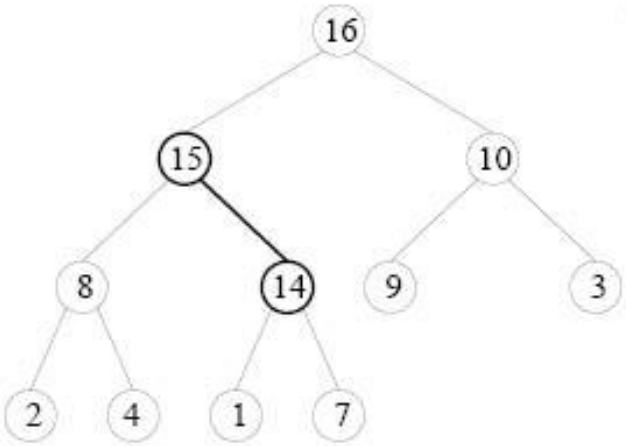
Exemple : soit à ajouter l'élément de priorité 15 :



- 1 -



- 2 -



- 3 -

Ajout d'élément

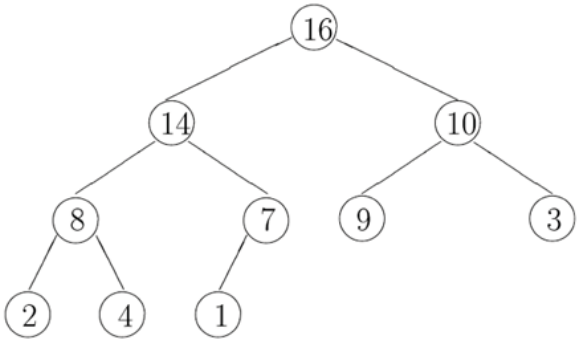
il suffit de l'ajouter tout en bas de l'arbre (1ère feuille libre) et de le faire remonter à sa place, c'est-à-dire à l'échanger avec son père tant que le poids de celui-ci (le père) est supérieur à son propre poids (le nouveau nœud). Cette opération porte généralement le nom de **percolation**.

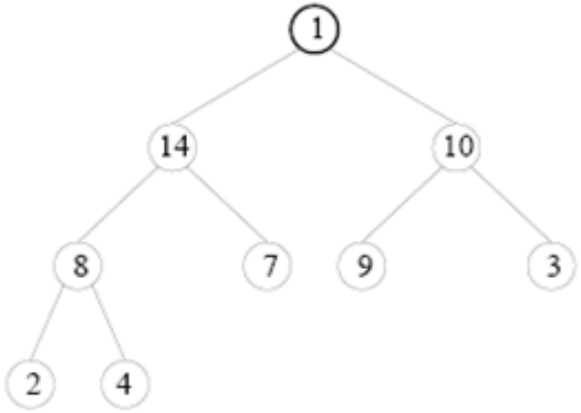
Retrait

- L'élément le plus prioritaire se trouve toujours à la racine, donc le retrait consiste à lire la racine puis la supprimer. Pour ce faire on doit :

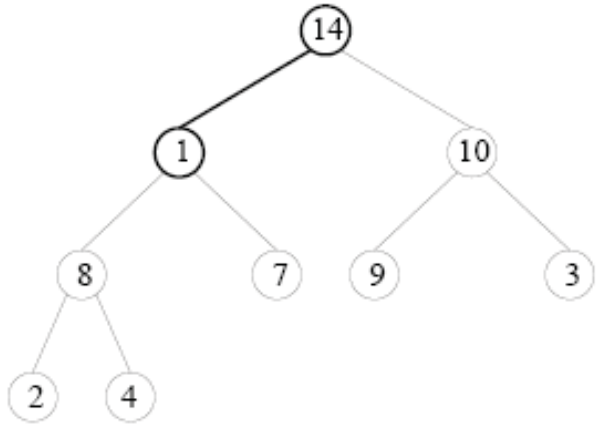
1. Remplacer la valeur de la racine par la valeur de l'élément le plus à droite dans le dernier niveau.
2. Supprimer de l'arbre cet élément (le plus à droite dans le dernier niveau), on obtient un arbre binaire mais pas nécessairement un tas.
3. On compare la valeur de la racine avec les valeurs de ses deux fils et on la permute avec la plus grande. On recommence le processus jusqu'il n'y ait plus d'éléments à permuter.

Exemple

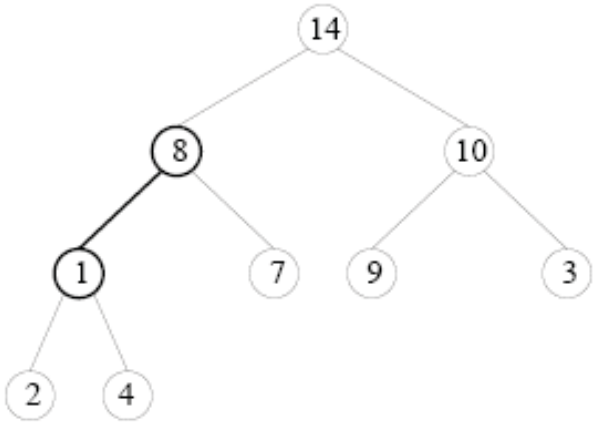




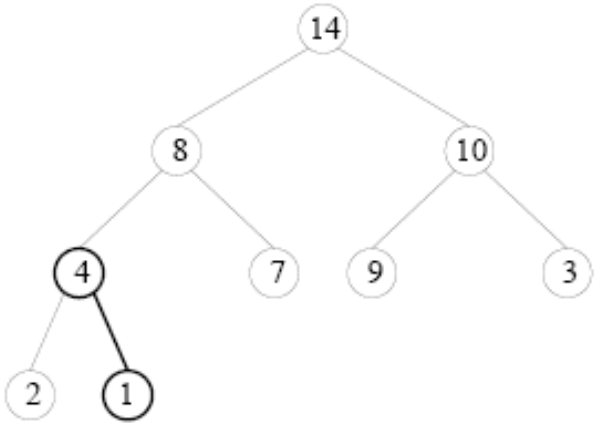
- 1 -



- 2 -



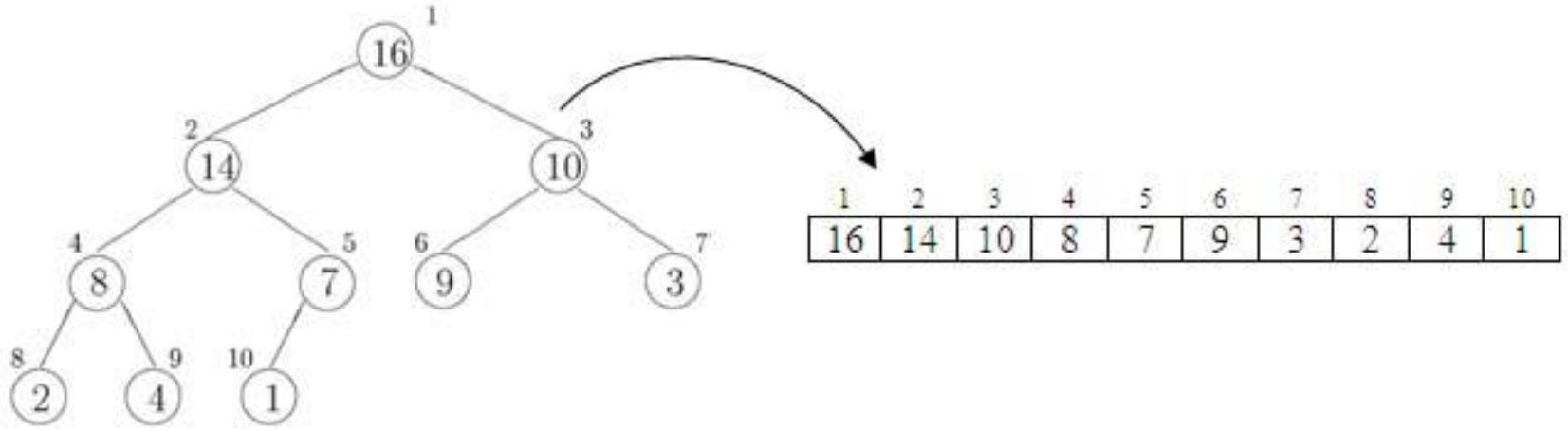
- 3 -



- 4 -

Implémentation des tas

- Les tas peuvent être implémentés dynamiquement exactement comme les ARB, et sont utilisés par le même modèle.
- Une représentation statique très efficace utilisant des tableaux est très utilisée en pratique, elle consiste à ranger les éléments du tas dans un tableau selon un **parcours en largeur** :

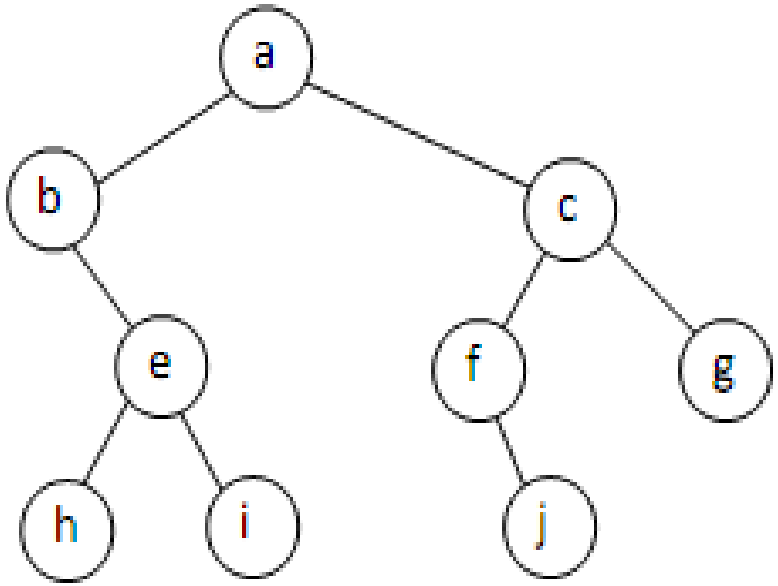


Remarque

- le fils gauche d'un élément d'indice ***i*** *se trouve* toujours s'il existe à la position ***2i****, et*
- *son fils droit se trouve à la position (**2i + 1**) et*
- *son* père se trouve à la position ***i/2***.
- *Les opérations d'ajout et de retrait sur le tas statique se* font de la même façon que dans le cas du tas dynamique. Avec ce principe les opérations d'ajout et de retrait se font d'une manière très simple et extrêmement efficace.

Exercice

- Donner la représentation de l'arbre suivant dans un tableau



- Dite si c'est un TAS ? Pourquoi ?