# Développement : Tri par tas

Pierron Théo – Lacoste Cyril

13 avril 2014

**Définition** Un tas est un arbre binaire dont tous les niveaux sont complètement remplis sauf éventuellement le dernier. De plus, toutes les feuilles doivent être le plus à gauche possible.

On peut le représenter par un tableau $A$ et un entier *taille* tels que les éléments du tas soient dans $A[1 \dots taille]$.

Un tas max est un tas tel que la racine de tout sous-arbre est plus grande que les éléments du sous-arbre.

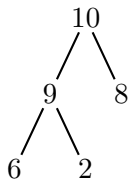


Figure 1 – Tas max représenté par $[|10; 9; 8; 6; 2|]$

Soit un tas $A$ est un indice $i$ tels que les fils gauche et droit de $i$ soient des tas max. On veut faire descendre $A[i]$ pour que l'arbre de racine $i$ soit un tas max. On utilise pour ce faire l'algorithme suivant

**Algorithme 1:** Entasser$(A, i)$

**Entrées** : Un tas $A$ et un entier $i$

**Sorties** : Une permutation de $A$ tel que l'arbre de racine $i$ soit un tas max

```
1  g := gauche(i)
2  d := droite(i)
3  max := i
4  si g ⩽ taille et A[g] > A[i] alors
5  |  max := g
6  si d ⩽ taille et A[d] > A[i] alors
7  |  max := d
8  si max ≠ i alors
9  |  échanger A[i] et A[max]
10 |  Entasser(A, max)
```

La complexité de Entasser$(A, i)$ sur un noeud $i$ de hauteur $h$ est au pire $O(h)$.

Pour construire un tas max à partir d'un tableau $A$, on va appeler Entasser sur chaque élément du tableau. On remarque que les feuilles, ie les éléments de $A[\frac{|A|}{2}, \dots, |A|]$ sont déjà des

tas max. Il suffit donc de considérer les éléments de $A[1, \ldots, \frac{|A|}{2}]$.

**Algorithme 2:** Construire($A$)

**Entrées** : Un tableau $A$
**Sorties** : Une permutation de $A$ correspondant à un tas max

```
1 taille := A
2 pour i = |A|/2 ... 1 faire
3     Entasser(A, i)
```

Quand on construit le tas, on appelle Entasser sur $\lfloor \frac{n}{2^{h+1}} \rfloor$ nœuds de hauteur $h$. La complexité est donc

$$\sum_{i=0}^{\lfloor \log_2(n) \rfloor} \left\lfloor \frac{n}{2^{h+1}} \right\rfloor O(h) = O\left( n \sum_{i=0}^{\lfloor \log_2(n) \rfloor} \frac{h}{2^h} \right) = O(n)$$

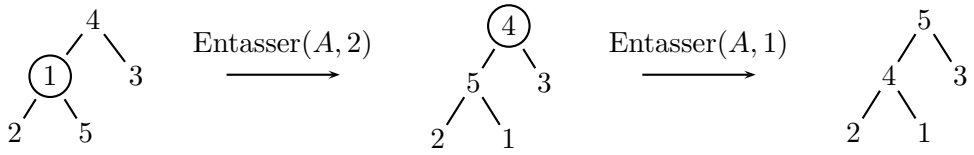


FIGURE 2 – Étapes de Construire([|4 ;1 ;3 ;2 ;5|])

On peut maintenant donner l'algorithme du tri par tas : on construit un tas puis on enlève la racine, on entasse, et on recommence.

**Algorithme 3:** Tri($A$)

**Entrées** : Un tableau $A$
**Sorties** : Le tableau trié associé à $A$

```
1 Construire(A)
2 pour i = |A| ... 2 faire
3     Échanger A[1] et A[i]
4     taille := taille − 1
5     Entasser(A, 1)
```

Le tri par tas a donc une complexité de $O(n) + n \times O(\log_2(n)) = O(n \ln(n))$ car la racine est de hauteur $\lfloor \log_2(n) \rfloor$.

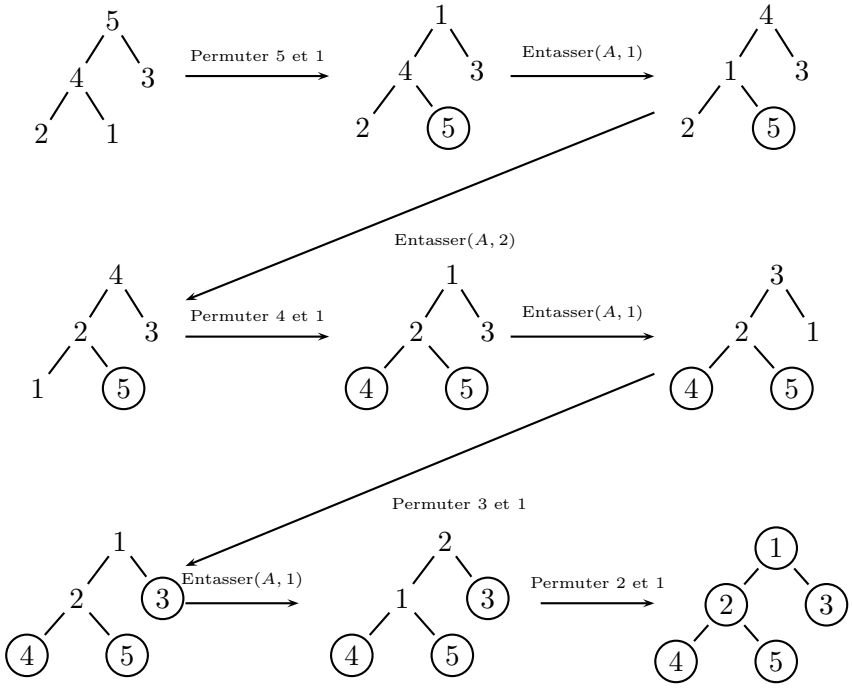


FIGURE 3 – Étapes de Tri([|4 ;1 ;3 ;2 ;5|])