TRI EN TAS

INTRODUCTION

C'est une version élaborée du tri par sélection. Elle utilise le principe suivant :

Principe. — Une fois le plus grand élément localisé le tableau restant n'est pas « inconnu ».

1. TAS ET MISE EN TAS

1.1. Définitions, principes

Pour la suite, on suppose que A est un tableau indicé de 1 à n.

> DÉFINITION 1.1.0.1. —
> On dit que A est *organisé en tas* à la position $i \leq n$ si, et seulement si, l'une des trois conditions suivantes est satisfaite :
>
> 1. $2i > n$;
> 2. $2i = n$ et $A[i] \geq A[n]$;
> 3. $2i < n$ et $A[i] \geq \max(A[2i], A[2i+1])$ et A organisé en tas aux positions $2i$ et $2i+1$.

> DÉFINITION 1.1.0.2. —
> *A est en tas* si A est en tas en toute position $i \leq n$.

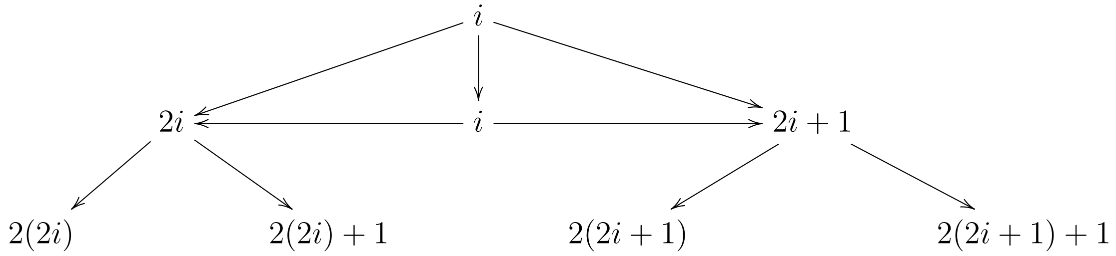
Exemple. — En prenant le tableau :

$$\begin{pmatrix} 7 & 6 & 5 & 1 & 4 & 3 & 2 \end{pmatrix}$$

— en position 1 on a $2i = 2 < n = 7$ donc il faut vérifier $7 > 6$ et $7 > 5$ ce qui est le cas et il faut vérifier que A est organisé en tas aux positions 2 et 3 :
  — en position 2, on a bien $6 > 1$ et $6 > 4$ ainsi que 1 et 4 en tas (puisqu'en positions supérieures à $n/2$) ;
  — en position 3, on a bien $5 > 3$ et $5 > 2$.

Remarques. — A est toujours organisé en tas pour les positions de $\lfloor n/2 \rfloor + 1$ à n. Si A est trié en ordre décroissant alors A est organisé en tas. Si A est en tas alors la plus grande valeur de A est en première position.

> Définition 1.1.0.3. —
> Soit $i, j \in \{1, \ldots, n\}$. On dit que j *descend* de si l'une des conditions suivantes :
> — $j = i$;
> — $j = 2i$;
> — $j = 2i + 1$;
> — $2i \leq n$ et j descend de $2i$:
> — $2i + 1 \leq n$ et j descend de $2i + 1$.



> Définition 1.1.0.4. —
> On dit que :
> — i est un descendant de *niveau* 0 de i ;
> — si x descendant du niveau $l \geq 0$ de $2i$ ou $2i + 1$ alors x descendant de *niveau* $l + 1$ de i.

Remarque. — Par définition, si la propriété de tas est vérifiée en i, alors la propriété de tas est vérifiée par tous les descendants de i. En particulier, si x descend de i alors

$$A[i] \geq A[x].$$

Par conséquent si A est en tas, $A[1] \geq A[i]$ pour tout $i \in \{1, \ldots, n\}$ car tout $x \in \{1, \ldots, n\}$ descend de 1.

Exemple. — Organiser en tas à la position A en sachant que A est organisé en tas à la position 2 et 3. On prend

$$A = \begin{pmatrix} 4 & 6 & 7 & 3 & 2 & 5 & 1 \end{pmatrix}.$$

On compare $A[1]$ avec $A[2]$ et $A[3]$, on permute le plus grand des deux :

$$A = \begin{pmatrix} 7 & 6 & 4 & 3 & 2 & 5 & 1 \end{pmatrix},$$

A est alors en tas en position 1 seulement vis-à-vis des ses voisins immédiats. A n'est plus en tas pour la position 3 : on répète le processus,

$$A = \begin{pmatrix} 7 & 6 & 5 & 3 & 2 & 4 & 1 \end{pmatrix}$$

et le tableau est alors en tas.

L'algorithme. — On fait les hypothèses suivantes pour cet algorithme : on l'invoque sur la position $p \leq \lfloor n/2 \rfloor$, fait en sorte que la propriété de tas soit satisfaite en p à condition qu'elle le soit en $2p$ et en $2p+1$ (si cela a un sens). On suppose T indicé de 1 à n (simplifie les notations).

```
function heapBubble(T,p,fin)
        test = false
        repeat
                l = 2*p
                r = 2*p+1
                m = p
                if l < fin and T[l] > T[m] then
                        m = l
                if r < fin and T[r] > T[m] then
                        m = r
                if m != p then
                        T[m], T[p] = T[p], T[m]
                        p = m
                else test = true
        until test = true
```

L'algorithme suivant met en tas sur toutes les positions.

```
function Heapify(T)
        n = longueur(T)
        for p from int(n/2) to 1 do
                heapBubble(T,p,n+1)
```

L'algorithme suivant fait le tri par tas :

```
function HeapSort(T)
        n = longueur(T)
        Heapify(T)
```

```
    for i from n to 2 do
        T[i], T[1] = T[1], T[i]
        HeapBubble(T,1,i)
```

Exemple. — On prend :

$$A = \begin{pmatrix} 1 & 6 & 5 & 7 & 3 & 2 & 4 \end{pmatrix}$$

qui en tas est :

$$\begin{pmatrix} 7 & 6 & 5 & 3 & 2 & 4 & 1 \end{pmatrix}.$$

On permute $T[1]$ et $T[n]$:

$$\begin{pmatrix} 1 & 6 & 5 & 3 & 2 & 4 & \mid & 7 \end{pmatrix}.$$

On met le sous-tableau en tas pour la position 1 :

$$\begin{pmatrix} 6 & 1 & 5 & 3 & 2 & 4 & \mid & 7 \end{pmatrix} \to \begin{pmatrix} 6 & 3 & 5 & 1 & 2 & 4 & \mid & 7 \end{pmatrix}.$$

On permute $T[1]$ et $T[n-1]$:

$$\begin{pmatrix} 4 & 3 & 5 & 1 & 2 & \mid & 6 & 7 \end{pmatrix}.$$

On met en tas pour la position 1 :

$$\begin{pmatrix} 5 & 3 & 4 & 1 & 2 & \mid & 6 & 7 \end{pmatrix}.$$

On permute $T[1]$ et $T[n-2]$:

$$\begin{pmatrix} 2 & 3 & 4 & 1 & \mid & 5 & 6 & 7 \end{pmatrix}.$$

On met en tas pour la position 1 :

$$\begin{pmatrix} 4 & 3 & 2 & 1 & \mid & 5 & 6 & 7 \end{pmatrix}.$$

On permute $T[1]$ et $T[n-3]$:

$$\begin{pmatrix} 1 & 3 & 2 & \mid & 4 & 5 & 6 & 7 \end{pmatrix}.$$

On met en tas pour la position 1 :

$$\begin{pmatrix} 3 & 1 & 2 & \mid & 4 & 5 & 6 & 7 \end{pmatrix} \to \begin{pmatrix} 2 & 1 & \mid & 3 & 4 & 5 & 6 & 7 \end{pmatrix} \to \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{pmatrix}.$$