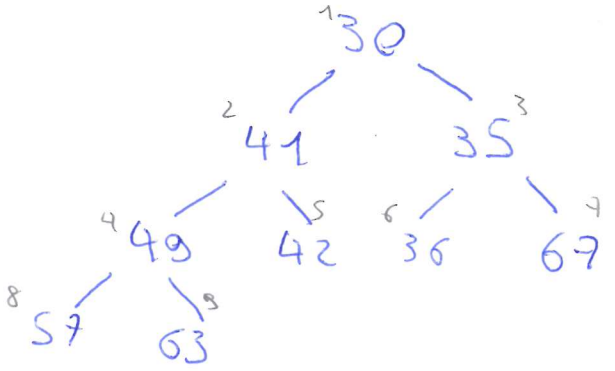
NF16 - Tas - Rappels

longueur A : taille max
taille A : nb éléments

Un tas est :

- Abre binaire parfait : les feuilles sont situées sur 2 niveaux au plus : l'avant dernier niveau est complet et les feuilles du dernier niveau sont regroupées le plus à gauche possible.
- Une clé est associée à chaque noeud
- Une relation d'ordre ($\leq$ ou $\geq$) lie la clé d'un noeud à celle des noeuds fils.



On peut représenter un tas avec un tableau A

Racine (A) ← Retourner 1

Père (i) ← Retourner ($\lfloor i/2 \rfloor$)

Gauche (i) ← Retourner (2i)

Droit (i) ← Retourner (2i+1)

$A[i] \geq A[\text{Père}(i)]$

Eléments entre les indices $\lfloor \text{taille}[A]/2 \rfloor + 1$ et taille[A] sont des feuilles

1 2 3 4 5 6 7 8 9
[30, 41, 35, 49, 42, 36, 37, 57, 63]

NF16 - Tas - 4

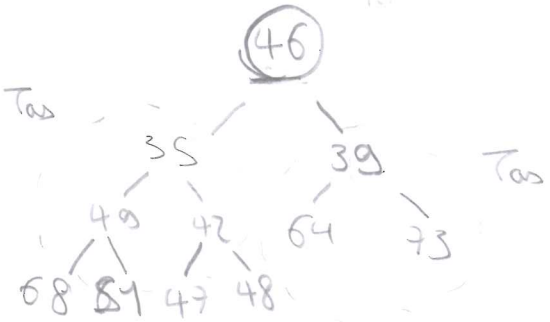
On considère la relation d'ordre "$\leq$" : le plus petit élément se trouve à la racine.

A) $L_1 = (1, 3, 1, 6, 9, 7, 12, 8, 7, 9, 10, 10)$

On a $L_1[i] \geq L_1[\lfloor i/2 \rfloor]$ $\forall i > 1$ **Ok**

$L_2 = (2, 6, 12, 4, 9, 8, 10, 7, 15, 3)$

$L_2[4] < L_2[2]$ **KO**



Si SAG[i] et SAD[i] est un tas

Entasser (A, i) ↳ Transforme le sous-arbre de racine i en un tas

l, r, i, min sont des indices

l := gauche [i]

r := droit [i]

Si l ≤ taille [A] et A[l] < A[i]

min := l

Sinon

min := i

Si r ≤ taille [A] et A[r] < A[min]

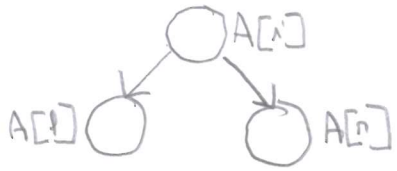
min := r

Si min ≠ i

échange (Ai, min) // Swap entre A[i] et A[min], la valeur i descend de 1 niveau

Entasser (A, min) // Continue de pair le niveau en dessous

Recherche du min entre A[i], A[l], A[r]



$O(h)$

NF16 - Tas - 2

Construire_Tas(A) Pas besoin d'entasser les feuilles

Pour i := ⌊longueur[A]/2⌋ à 1 $\Theta(n)$ non démontré ici

Entasser(A, i)

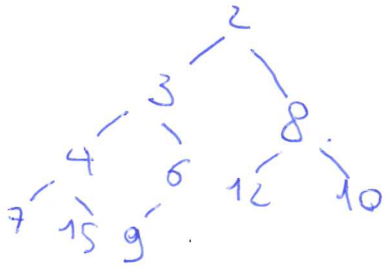
Pour L2 = (2, 6, 12, 4, 9, 8, 10, 7, 15, 3)

(2, 6, 12, 4, 3, 8, 10, 7, 15, 9)

(2, 6, 12, 4, 3, 8, 10, 7, 15, 9)

(2, 6, 8, 4, 3, 12, 10, 7, 15, 9)

(2, 3, 8, 4, 6, 12, 10, 7, 15, 9)



NF16 - Tas - 3

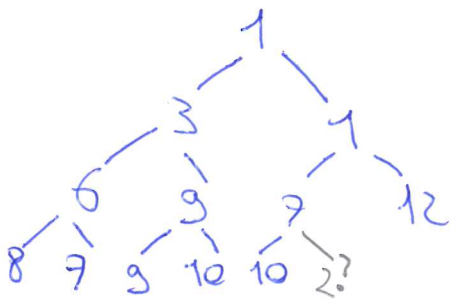
B) $L_1 = (1, 3, 1, 6, 9, 7, 12, 8, 7, 9, 10, 10)$

Insertion d'un élément de priorité 2

```
Inserer_tas (A, cle)
    taille[A] := taille[A] + 1
    i := taille[A]
    Tant que i > 1 et A[Pere[i]] > cle
        A[i] := A[Pere[i]]
        i := Pere[i]
    A[i] := cle
```

O(h)

	1	2	3	4	5	6	7	8	9	10	11	12	13	i	cle'	A[Pere(i)]
	1	3	1	6	9	(7)	12	8	7	9	10	10	⌴	13	<	7
It 1			(1)			i	⌴						7	6		1
It 2		//			//											
Fin	1	3	1	6	9	7	12	8	7	9	10	10	7			



NF16 - Tas - 4

Suppression de l'élément le plus petit → À la racine

Extraire_Min_Max_tas(A) $\mathcal{O}(h)$

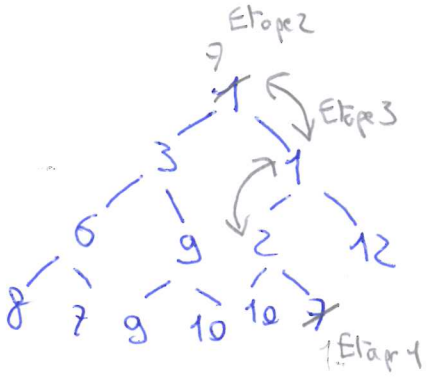
max := A[1]

A[1] := A[taille[A]]

taille[A] := taille[A] - 1

Entasser(A,1)

Retourner(max)



C)(1) Dans une liste triée (par ordre croissant) : $A[i] \geq A\left[\left\lfloor \frac{i}{2} \right\rfloor\right] \forall i > 1$ donc ainsi

(2) Contre-exemple: La liste L4 est un tas mais n'est pas triée

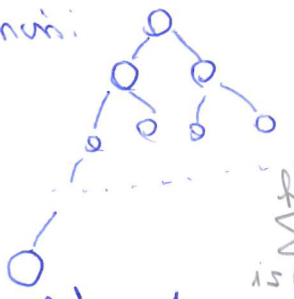
D) Plus petit élément à la racine : A[1]

Plus grand élément au niveau d'une feuille: $\left\lfloor \frac{taille[A]}{2} \right\rfloor + 1 \leq i \leq taille[A]$

E) A chaque itération, on remonte d'un niveau : $\mathcal{O}(h)$

F) // descend : $\mathcal{O}(h)$

G) Nombre min:

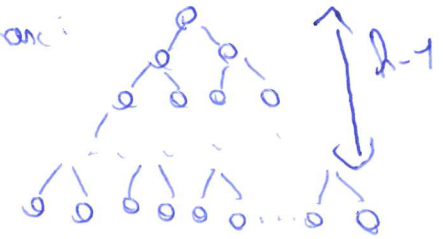


$\sum_{i=0}^{h} a^i = \frac{a^{h+1} - 1}{a - 1}$

h-1 niveaux complets + 1

$p = \left(\sum_{i=0}^{h-1} 2^i\right) + 1 = 1 + \frac{2^h - 1}{2 - 1} = 2^h$

Nombre max:



h niveaux complets

$p = \sum_{i=0}^{h} 2^i = 2^{h+1} - 1$

H) $2^h \leq p \leq 2^{h+1} - 1$

$2^h \leq p < 2^{h+1}$

$h \leq \log_2(p) < h+1$

$h = \lfloor \log_2(p) \rfloor$