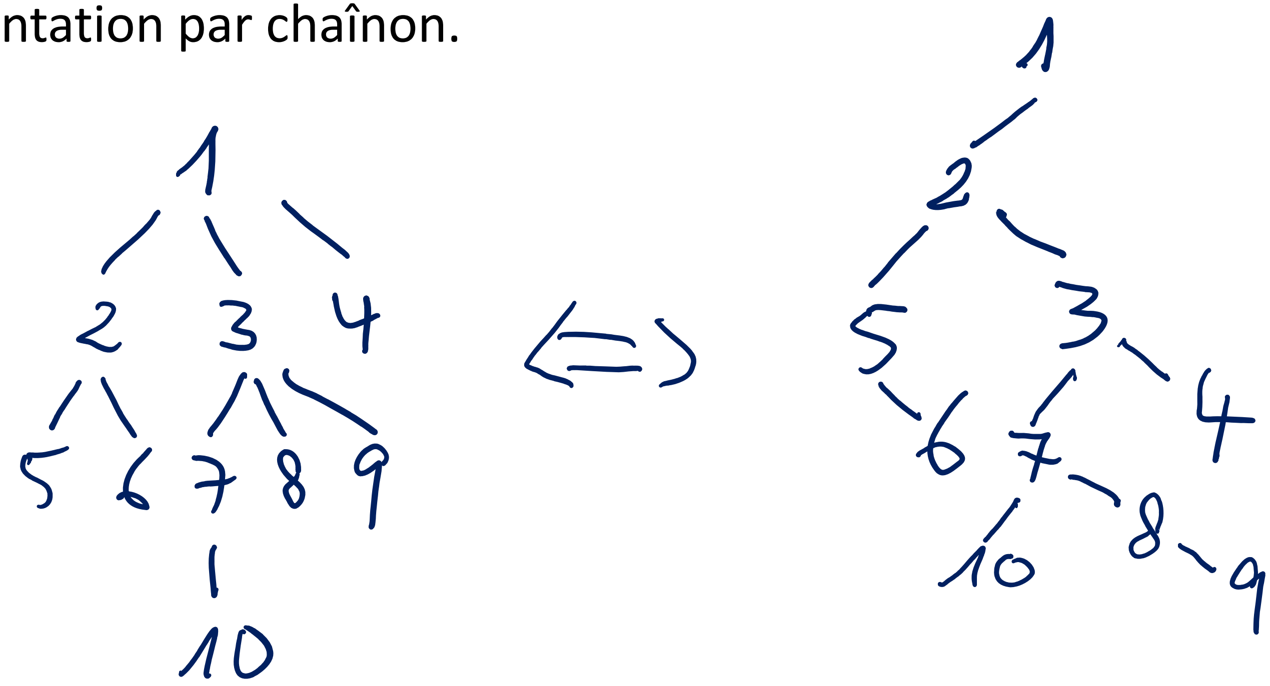
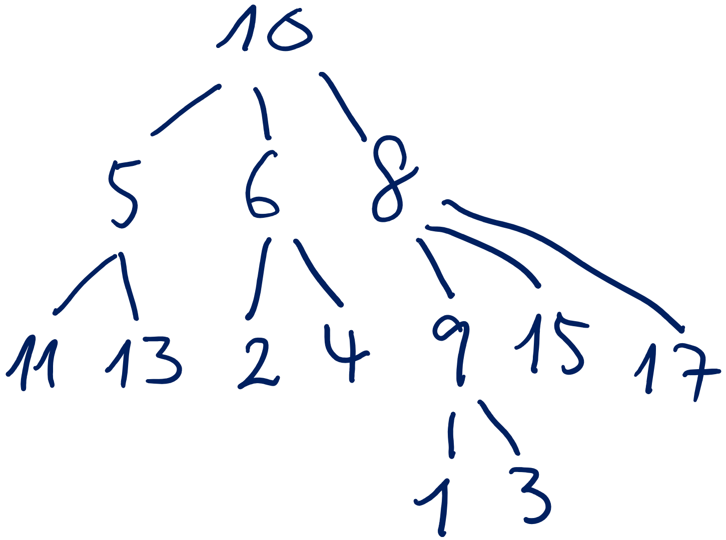
Représentations d’arbres généraux

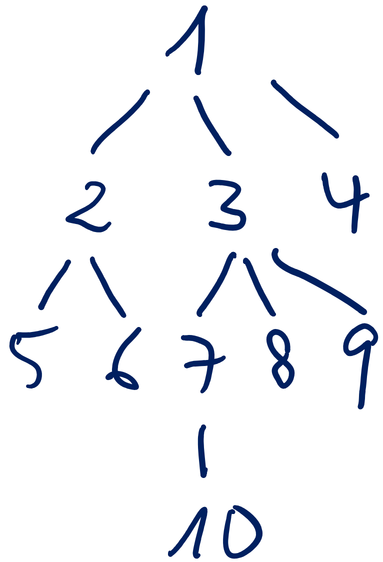
- Par premiers fils et frères droits: correspondance entre arbres généraux et arbres binaires
  - Représentation par chaînon.

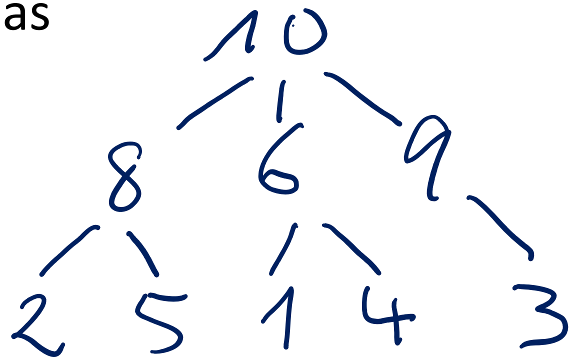


Quelques définitions

- Arbres ordonnés:
  - les fils d'un sommet sont ordonnés de gauche à droite
  - les fils d'un sommet sont ordonnés du haut vers le bas







Implémentation d’arbres

- Représentations par structures chaînées: ABR, AVL, Arbres généraux

- Représentations simple par tableau: T[i] contient l’indice du sommet père de i, l’exemple des Tas

i

$2i \quad 2i+1$

Tableau débutant par 1

i

$2i+1 \quad 2i+2$

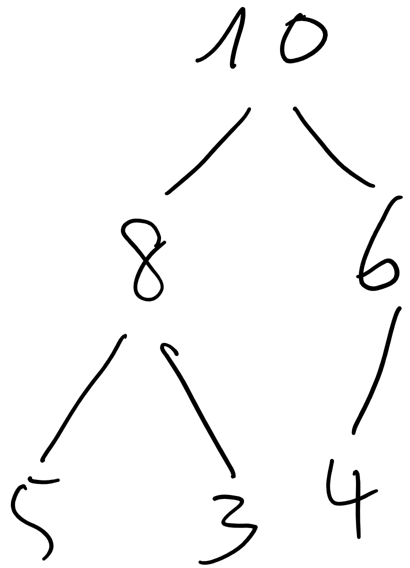
Tableau débutant par Ø

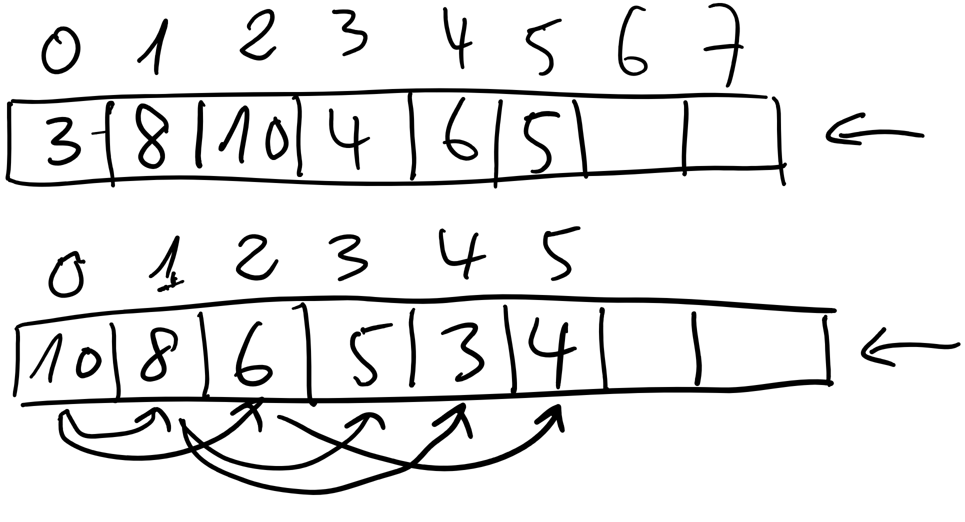
i

$ki+1 \quad ki+2 \quad \dots \quad ki+k$

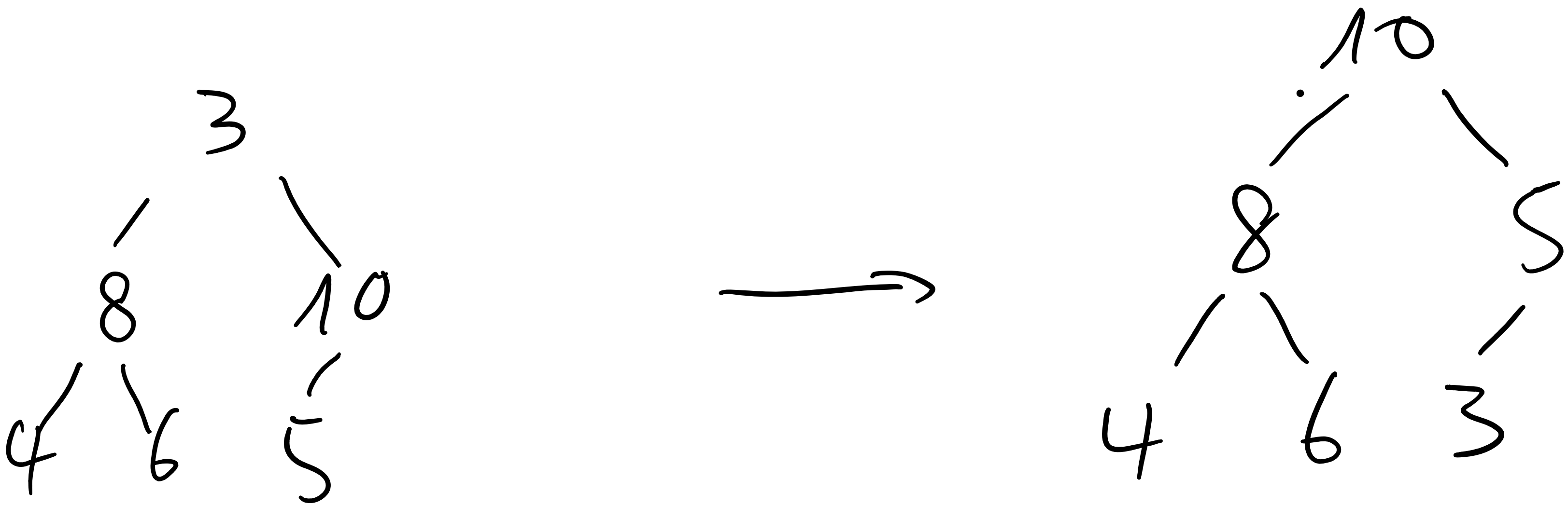
avec k fils

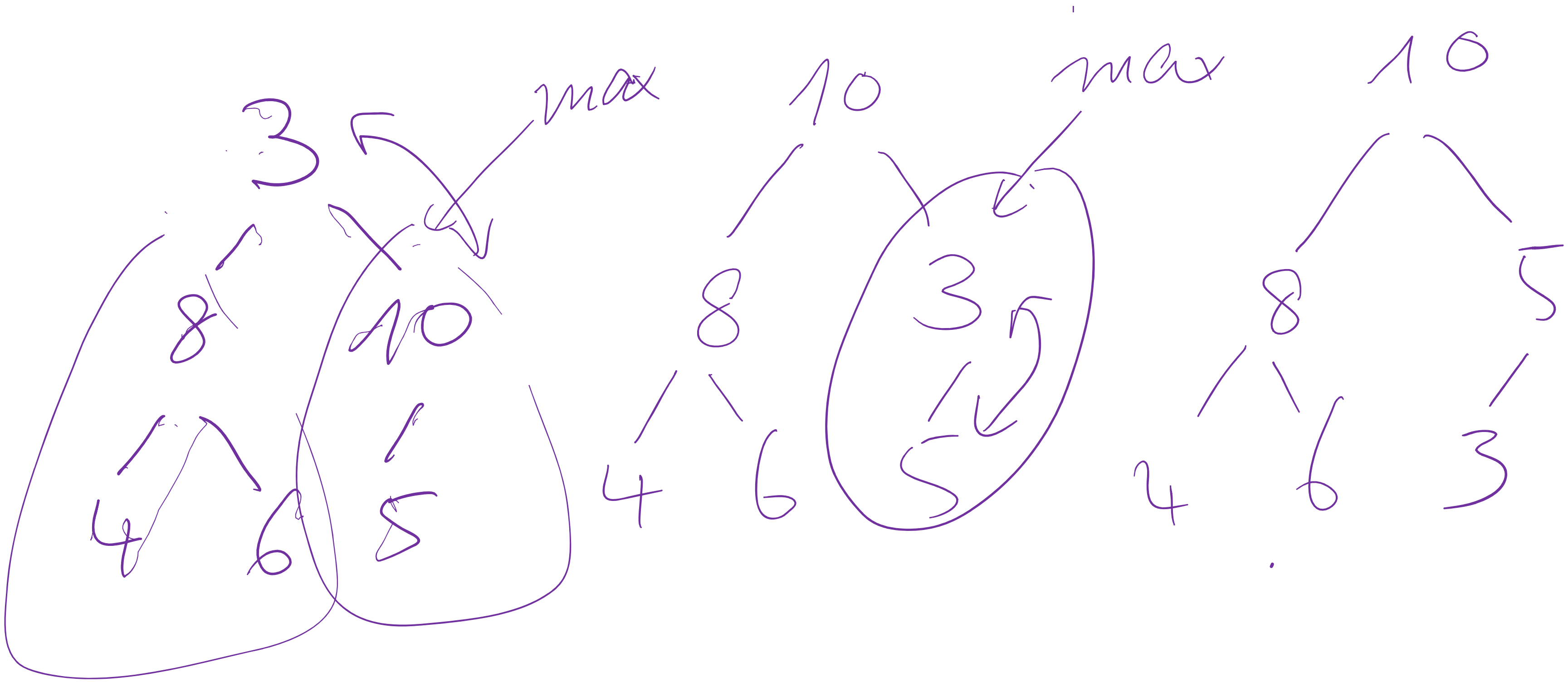
Tableau débutant par Ø


10
8
6
5
3
4


0 1 2 3 4 5 6 7
3 8 10 4 6 5
0 1 2 3 4 5
10 8 6 5 3 4

3 8 10 4 6 5



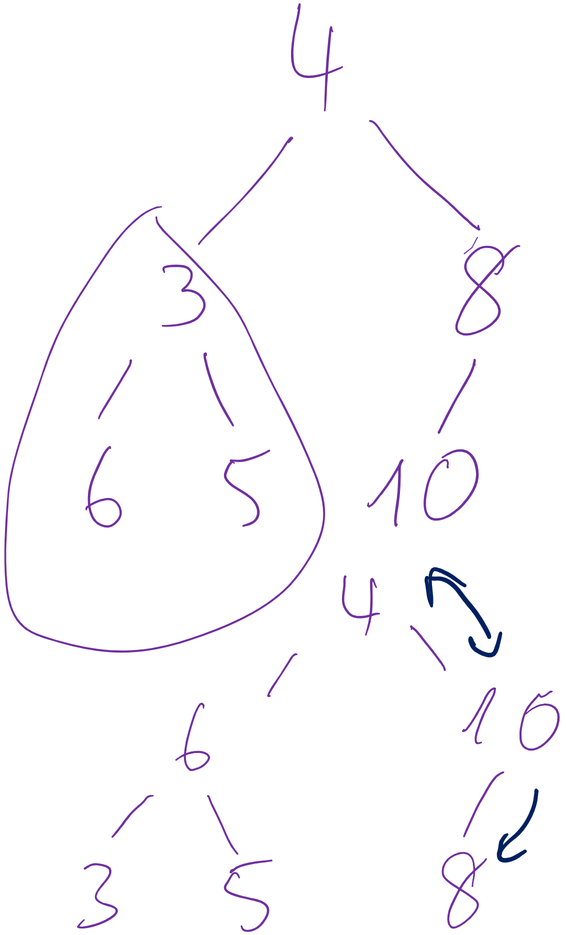


max
max

T	4	6	8	3	5	10

4 3 8 6 5 10

Construire(T) Entasser(T, i).



Tas

10 6 8 3 5 4 $\xRightarrow{\text{Tri}}$ 3 4 5 6 8 10

10 6 8 3 5 4

4 6 8 3 5 | 10

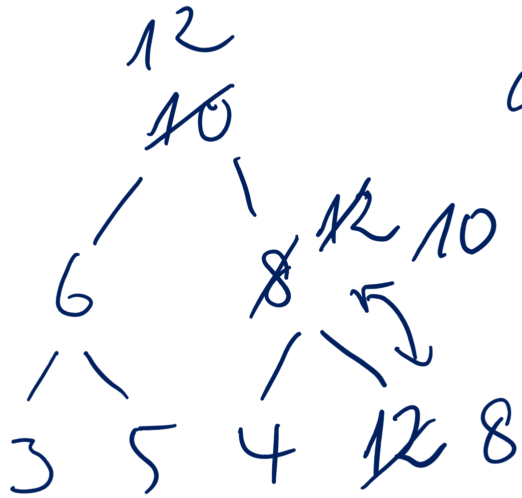
8 6 4 3 5 | 10

5 6 4 3 | 8 10

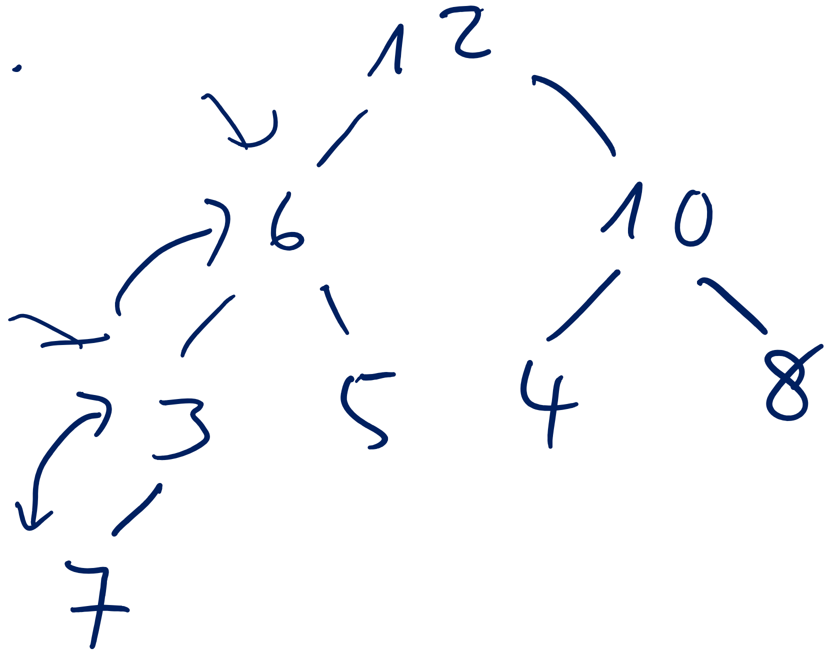
6 5 4 3 8 10

3 5 4 6 8 10

10 6 8 3 5 4 .



ajout 12.



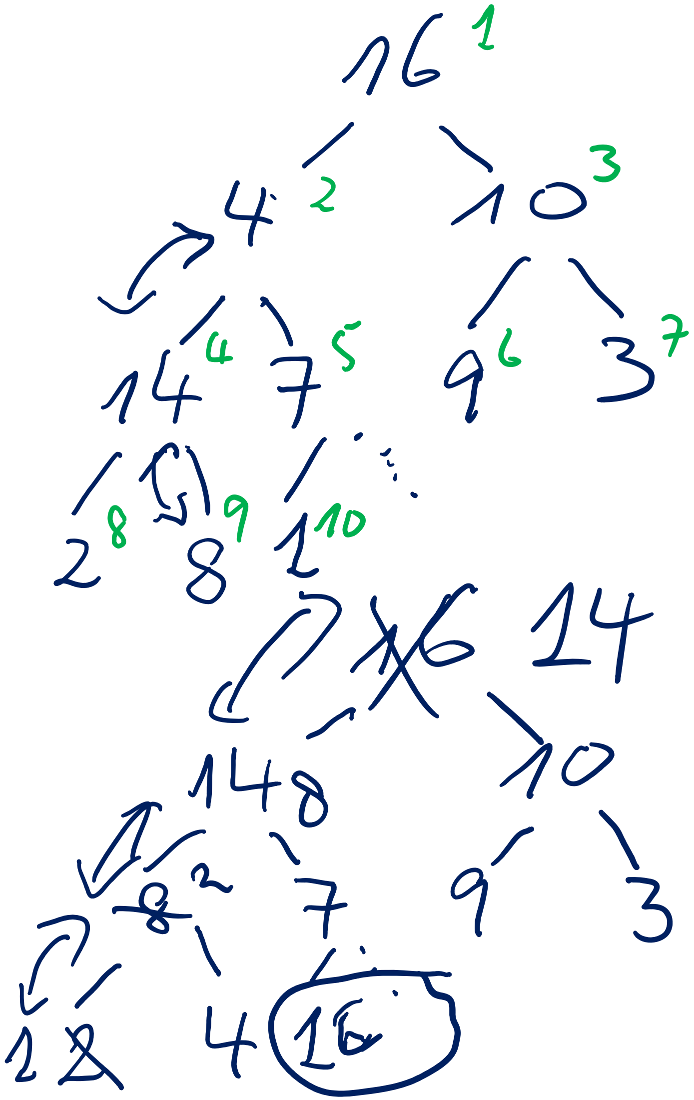
Les Tas ou (Max/Min)imiers

- (Max/Min-)Heap en anglais, mais à NE PAS CONFONDRE avec le heap système !
- Tas=Arbre k-aire avec un invariant où le père est « plus prioritaire » que les fils. En général, on utilise une implémentation dans un tableau.
- Nous verrons ensemble les 3 procédures suivantes:
  - `Entasser(T,i)` qui garantit le maintient de la propriété de Tas
  - `Construite-Tas(T)` qui produit un Tas à partir d'un tableau non ordonné
  - `Trier-Tas(T)` qui trie un tableau sur place
- A voir en plus `Extraire-PlusPrio` (Max ou Min) et `Insérer` pour utiliser les Tas comme des files de priorité

`Entassser(T,i)`

- `T` est le tableau contenant le Tas
- `i` est l'indice de l'élément de `T` qui doit être rangé au bon endroit de `T` pour que `T` ait la propriété d'un Tas
- Hypothèse, on suppose que les sous-arbres gauche et droit de `i` sont des Tas.

Maximier



	1	2	3	4	5	6	7	8	9	10
T	16	4	10	14	7	9	3	2	8	1

T 16 14 10 8 7 9 3 2 4 | 1

1 | 16

14 8 10 2 7 9 3 1 4 | 16

4

1 | 14 16

```
Entasser(T,i)
    g = gauche(i) ; d = droite(i)
    si g ≤ T.taille et T[g] > T[i] alors
        max = g ;
    sinon max = i ;
    si d ≤ T.taille et T[d] > T[max] alors
        max = d ;
    si max ≠ i alors
        Echanger(T, i, max);
        Entasser(T, max);
```

Construire-Tas(T)

Pour $i = \lfloor T.size/2 \rfloor$ à 0

Entasser(T, i);

```
Trier-Tas(T)
```

Construire-Tas(T);

Pour i = T.size à 1 Faire

Échanger(T, 0, i)

T.size = T.size - 1;

Entasser(T, 0);

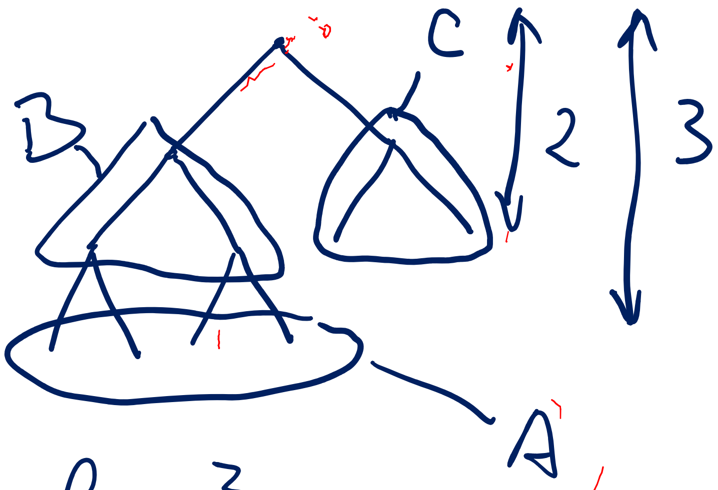
Complexité de `Entasser(T,i)` ?

`Construire-Tas(T)` et `Trier-Tas(T)` font appel à `Entasser(T,i)`.

Une idée:

`Entasser(T,i)` étant récursive, essayez d'écrire la fonction de la complexité temporelle associée T(n) de manière récurrente, à l'instar de la fonction du calcul factoriel ou encore celle du tri fusion.

Complexité de `Entasser(T,i)` ?



$h = 3$

$n = 11$

$$\left.\begin{aligned} A &= 2^{h-1} = 4 \\ B &= 2^{h-1} - 1 = 3 \\ C &= 2^{h-1} - 1 = 3 \end{aligned}\right\} \quad A + B = 7$$

$$7 \leq \frac{2 \times n}{3} = \frac{2 \times 11}{3}$$

$$T(n) \leq T\left(\frac{2n}{3}\right) + 1$$

$$T(n) \leq T\left(\frac{2n}{3}\right) + 1$$

$$T\left(\frac{2n}{3}\right) = T\left(\frac{2}{3} \times \frac{2n}{3}\right) + 1$$

$$= T\left(\left(\frac{2}{3}\right)^2 n\right) + 1$$

$$\leq T\left(\left(\frac{2}{3}\right)^h . n + \underbrace{1 + 1 + \ldots + 1}_{h \text{ fois}}\right)$$

$$\left(\frac{2}{3}\right)^h n \stackrel{?}{=} 1$$

$$n = \left(\frac{3}{2}\right)^h$$

$$\log_2 n = h \log_2 \tfrac{3}{2} \iff \log_{3/2} n = h \qquad \square$$