Définitions :

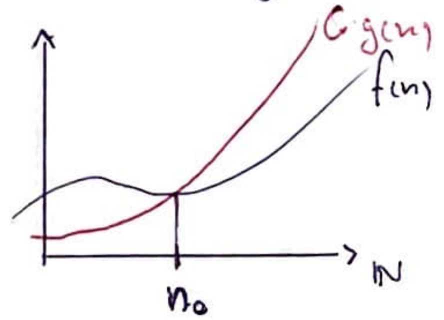
- Soient $f, g : \mathbb{N} \to \mathbb{N}$

  On dit que f est dominée par g, ou que g domine f, noté $f \in O(g)$

  si

  $\exists c > 0, \exists n_0 \in \mathbb{N}$ tels que

  $f(n) \le c \cdot g(n), \forall n \geq n_0$.



- Le fait que f est dominée par g, ou que g domine f, se note également $g \in \Omega(f)$ (i.e. $g \in \Omega(f)$ ssi $f \in O(g)$)

Lemme: Soit T un arbre binaire de hauteur h. Alors, le nb de feuilles $n(T)$ de T satisfait $n(T) \leq 2^h$.

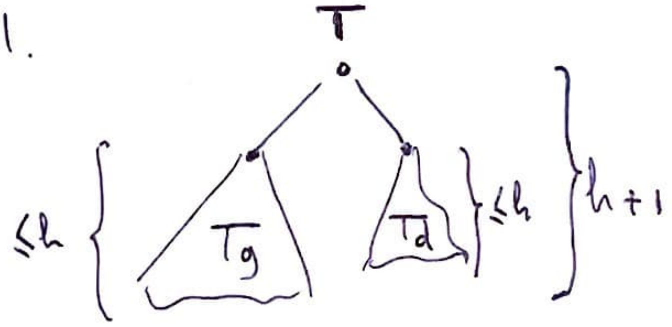
Autrement dit, un arbre binaire de hauteur h possède au plus 2^h feuilles

Preuve: Par induction sur h

Pour $h=0$: L'arbre de hauteur 0 possède une seule feuille, donc:

$$n(T) = 1 = 2^0 = 2^h \leq 2^h$$

Supposons le lemme vrai: pour tout arbre de hauteur h.
Soit T de hauteur $h+1$.



Le fils gauche T_g de T est de hauteur $\leq h$, et donc $n(T_g) \leq 2^h$ (par le lemme).

Idem, le fils droit T_d de T est de hauteur $\leq h$, et donc $n(T_d) \leq 2^h$ (par le lemme).

Ainsi, $n(T) = n(T_g) + n(T_d)$

$$\leq 2^h + 2^h = 2 \cdot 2^h = 2^{h+1}$$, cqfd. ∎

Corollaire : Soit T un arbre binaire avec $n(T) = n$ feuilles. Alors T est de hauteur $h(T) \geq \log_2(n)$.

Preuve : Par l'absurde.

Supposons que $h(T) < \log_2(n) \leq \log_2(n) - 1$

Alors, par le lemme, on a

$$n(T) \leq 2^{[\log_2(n) - 1]} = \frac{2^{\log_2(n)}}{2^1} = \frac{n}{2} < n,$$

contradiction (avec $n(T) = n$), cqfd $\blacksquare$

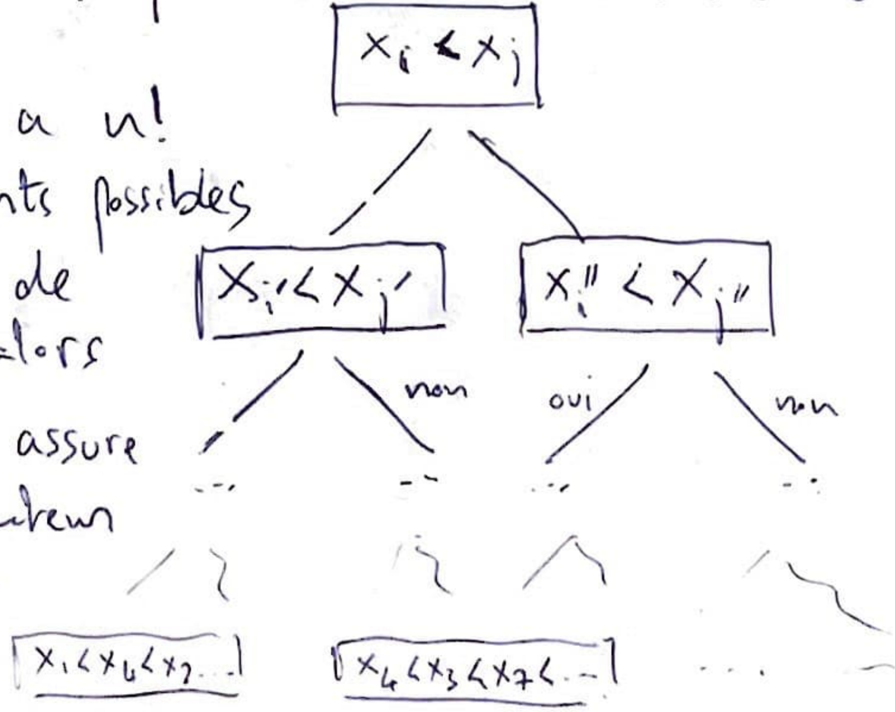
Théorème : Soit A un algorithme de tri fonctionnant par comparaisons successives de paires d'éléments (important !).
Alors, le temps d'exécution $t(A)$ de A domine $n \cdot \log(n)$, i.e.

$$t(A) \in \Omega\big(n \log(n)\big).$$

($t(A)$ est "au moins" $n \cdot \log(n)$,
$t(A)$ est "plus grand" que $n \log(n)$).

Preuve : L'algorithme A sur un tableau tab de taille n est un arbre de décision T dont chaque noeud représente une comparaison de 2 éléments x_i et x_j, et chaque feuille correspond à un ordonnancement possible des éléments de tab.

Puisqu'il y a $n!$ ordonnancements possibles de tableaux de tailles n, alors le corrolaire assure que la hauteur $h(T)$ de T satisfait



$$h(T) \geq \log_2(n!) = \log(n \cdot (n-1) \cdot \ldots \cdot 1)$$
$$= \log(n) + \log(n-1) + \ldots + \log\left(\frac{n}{2}\right) + \ldots + \log(1)$$

$$
\begin{aligned}
h(T) &\geq \log_2(n!) \\
&= \log_2(n\cdot(n-1)\cdot(n-2)\cdot\ldots\cdot 1) \\
&= \log_2(n) + \log_2(n-1) + \cdots + \log_2\left(\tfrac{n}{2}\right) + \cdots + \log_2(1) \\
&\geq \log_2(n) + \log_2(n-1) + \cdots + \log_2\left(\tfrac{n}{2}\right) \\
&\geq \underbrace{\log_2\left(\tfrac{n}{2}\right) + \log_2\left(\tfrac{n}{2}\right) + \cdots + \log_2\left(\tfrac{n}{2}\right)}_{n/2 \text{ fois}} \\
&= \frac{n}{2}\log\left(\frac{n}{2}\right)
\end{aligned}
$$

Ainsi, l'algo A devra effectuer au minimum $\frac{n}{2}\log\left(\frac{n}{2}\right)$ comparaisons pour identifier l'ordonnancement de tab, et donc son temps d'exécution $t(A) \in \Omega\left(\frac{n}{2}\log\left(\frac{n}{2}\right)\right)$.

$$\overset{\circledast}{=} \Omega(n\log(n))$$

cqfd.

$\circledast$

$$
\begin{aligned}
f \in \Omega(n\log(n)) &\Rightarrow n\log(n) \in O(f) \\
&\Rightarrow \frac{n}{2}\log\left(\frac{n}{2}\right) \in O(f) \\
&\Rightarrow f \in \Omega\left(\frac{n}{2}\log\left(\frac{n}{2}\right)\right)
\end{aligned}
$$

$$
\begin{aligned}
f \in \Omega\left(\frac{n}{2}\log\left(\frac{n}{2}\right)\right) &\Rightarrow \frac{n}{2}\log\left(\frac{n}{2}\right) \in O(f) \\
&\Rightarrow \exists c > 0 \text{ t.q. } \frac{n}{2}\log\left(\frac{n}{2}\right) \leq c f(n) \\
&\Rightarrow n\log\left(\frac{n}{2}\right) \leq 2c f(n) \\
&\Rightarrow n\log(n) - n\log(2) \leq 2c f(n) \\
&\Rightarrow n\log(n) \leq 2c f(n) + \underbrace{n}_{\leq f(n)}\underbrace{\log(2)}_{\leq 2} \\
&\qquad \leq (2c+2) f(n). \Rightarrow n\log(n) \in O(f)
\end{aligned}
$$

$\Rightarrow f \in \Omega(n\log(n))$.