complexity : Algo :

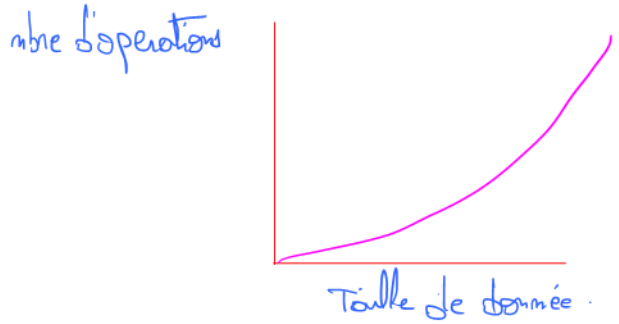
1) Mesure de complexité :

Le temp d'execution d'un algorithme donnée dépend principalement de :

- La machine utilisée : Les performance, Le Language.
- Les données auxquelles l'algorithme est appliqué : type et taille.

Dans notre cas nous allons utilisée une mesure qui ne dépend ni de la machine, ni des types, mais plutôt de la taille

ex : Le temp d'execution d'un Algo de tri depend de la longeur de la liste a trier :



2) Big O notation :

- Soit f et g deux fonctions de $\mathbb{R}$, on dit que $f(x)$ et $g(x)$: $o(g(x))$ si f est eventuellement dépassée par un multiple de g.

f et g deux fonctions :

$\mathbb{R} \longrightarrow \mathbb{R}$

$f(x)$ est $o(g(x))$

$f(x) \in o(g(x))$

$\forall x > k, \quad f(x) < o(g(x))$

} f est négligée par une autre fonction.

Avec k est un seuil qui permet d'ignorer le comportement des fonctions pour des données de petites tailles

La constante c appelée facteurs permet de faire abstraction de vitesse de la machine utilisée afin d'eviter la recherche de seuil k et de facteur c, on utilise des

Theorems 1 :

- $\lim_{+\infty} \frac{f(x)}{g(x)} = b, b > 0$, Alors $f(x) \in o(g(x))$

et $g(x) \in o(f(x))$

- Si $\lim_{+\infty} \frac{f(x)}{g(x)} = 0$, Alors $f(x) \in o(g(x))$ mais $g(x) \notin o(f(x))$

exemple:

$f(x) = 7x^2$, $g(x) = x^3$.

$f(x) \in o(g(x))$

$g(x) \notin o(f(x))$

car $\lim_{+\infty} \frac{f(x)}{g(x)} = \lim_{+\infty} \frac{7x^2}{x^3} = 0$.

Theoreme 2:

Soit la liste suivante des fonctions

$1, \log(n), n, n\log(n), n^2\log(n), n^3\log(n), \ldots, 2^n, n!$.

- Chaque fonction de la liste est o des fonctions qui se situent a droite,

L'inverse n'est pas vrai