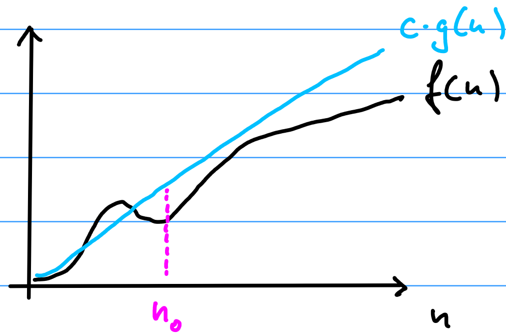
NOTATION ASYMPTOTIQUE

<u>motivation</u>: analyser le coût d'un algorithme

$f : \mathbb{N} \to \mathbb{N}$
$f(n)$: coût d'un algo (par ex : temps [s], énergie [mW])
pour une entrée de taille n.



1) on s'intéresse au comportement de f pour n grand ($n \geq n_0$)
2) on ne considère pas des facteurs constants : $2f$, $5f$, $10f$, c'est la même chose (*)
3) le comportement de f peut être compliqué → on utilise souvent de bornes sup/inf (g)

(*) pourquoi ?
Exécuter un algo sur des matériels différents ou utiliser des langages de programmation différents peut entraîner des différences du coût d'un facteur constant.

cad : le facteur est indépendant de la taille de l'entrée

En tenant compte de 1) – 3), on arrive aux définitions suivantes :

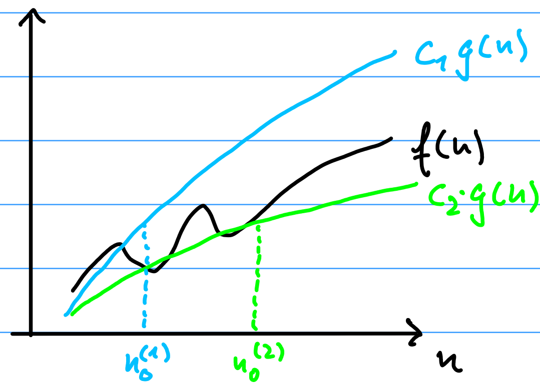
•) $f(n) = O(g(n)) :\Leftrightarrow \exists n_0 \in \mathbb{N}\ \exists c > 0\ \forall n \geq n_0 : f(n) \leq c \cdot g(n)$
(Grand O : f est dominé par g à un facteur près)

•) $f(n) = \Omega(g(n)) \;:\Leftrightarrow\; g(n) = O(f(n))$

(Grand Omega : f est minorée par g à un facteur près)

•) $f(n) = \Theta(g(n)) \;:\Leftrightarrow\; f(n) = O(gn)$ et $f(n) = \Omega(g(n))$

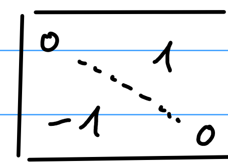
(Grand Theta : f est dominée et soumise à g asymptotiquement)



$f(n) = O(g(n))$ et $f(n) = \Omega(g(n))$

$\Rightarrow f(n) = \Theta(g(n))$

Exemple: initialiser une $n \times n$ matrice

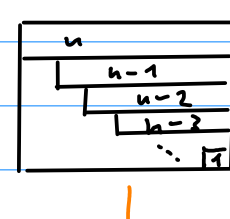


on suppose que le coût d'accès est constant

```
for i in range(n):
    a[i][i] = 0          ← coût constant : c1
    for j in range(i+1, n):
        a[i][j] = 1      } coût constant : c2
        a[j][i] = -1     }
```

le coût $f(n) = \sum_{i=0}^{n-1} \left(c_1 + \sum_{j=i+1}^{n-1} c_2 \right)$

$$\overset{c_3 := \max\{c_1, c_2\}}{\leq} \sum_{i=0}^{n-1} \left(c_3 + \sum_{j=i+1}^{n-1} c_3 \right)$$



$$= \sum_{i=0}^{n-1} \sum_{j=i}^{n-1} c_3 = c_3 \cdot \sum_{i=0}^{n-1} \sum_{j=i}^{n-1} 1 = c_3 \sum_{i=0}^{n-1} (n-i)$$

$n \geq 1 \Rightarrow n+1 \leq 2n$

$$= c_3 \cdot \sum_{i=1}^{n} i = c_3 \cdot \frac{n(n+1)}{2} \leq \frac{2c_3(n+1)^2}{2} \leq 4 c_3 n^2$$

$= O(n^2)$ $[n_0 = 1,\ c = 4c_3 :\ \forall n \geq 1,\ c \geq 4c_3 > 0 : f(n) \leq cn^2]$

On prend $c_4 := \min\{c_1, c_2\}$ et obtient : $f(n) \geq c_4 n^2 = \Omega(n^2)$
$[n_0 = 1,\ c = 1/c_4 > 0 :\ \forall n \geq 1,\ c \geq 1/c_4 : n^2 \leq c f(n)]$

On conclut que $f(n) = \Theta(n^2)$

Si $f(n) = \Theta(g(n))$ alors $g(n)$ a le même ordre de croissance que f.
On peut indiquer qu'une fonction n'a pas le bon ordre de croissance en utilisant la notation o et ω :

- $f(n) = o(g(n)) :\Leftrightarrow \lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} = 0$

  (Petit O : f est négligeable devant g asymptotiquement)

- $f(n) = \omega(g(n)) :\Leftrightarrow g(n) = o(f(n))$
  (Petit Omega : f domine g asymptotiquement)

Exemple : on montre : $f(n) = O(n^2) \Rightarrow f(n) = o(n^3)$

$\exists\, n_0 \in \mathbb{N},\ c > 0 :\ \forall n \geq n_0 :\ f(n) \leq cn^2$

$$\Rightarrow \lim_{n\to\infty} \frac{f(n)}{n^3} \leq \lim_{n\to\infty} \frac{cn^2}{n^3} = \lim_{n\to\infty} \frac{c}{n} = 0$$

Exercice : Montrer que

(a) $30n^2 + 104\, n \cdot \log_2 n = \Theta(n^2)$

(b) $\max\{f_1(n), f_2(n)\} = \Theta(f_1(n) + f_2(n))$

(c) $\log_2(n!) = O(n \log_2 n)$

(d) $2^n = o(2{,}01^n)$