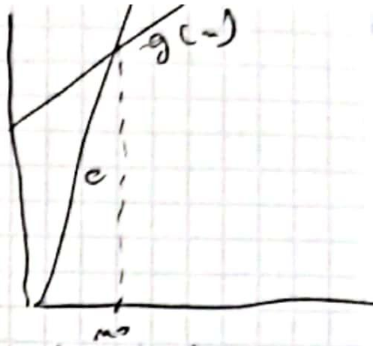
complexité - TD 1.



Exercice 01 : Majoration de la complexité :

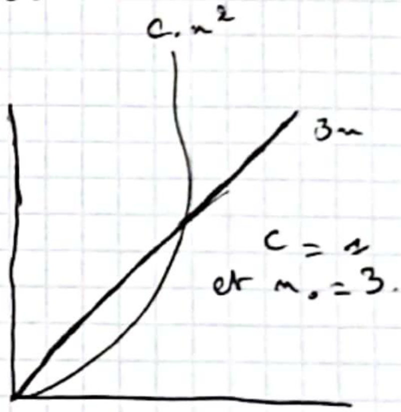
① $g(n) = 6n + 12 \in O(n)$: On cherche une fonction qui est plus grande que $g(n)$.

$\exists c\ \exists n_0\ \forall n \geq n_0 : g(n) < c.n$

pour $c = 7$ ET $n_0 = 12$.

On a pour tout $n \geq n_0$, que $6n + 12 \leq 7.n$.

equivalent $\Longleftrightarrow g(n) \leq c.n$.



② $g(n) = 3n \in O(n^2)$.

$\exists c\ \exists n_0\ \forall n \geq n_0 : g(n) \leq c.n$

pour $c = 1$ et $n_0 = 3$.

On a pour tout $n \geq n_0$, que $3n \leq 1n^2$.

$\Longleftrightarrow g(n) \leq c\,n^2$.

③ $g(n) = 10^{10^6} n \in \Theta(n)$, pour $c = 10^{10^7}$ et $n_0 = 1$.

On pour tout $n \geq n_0$, que $10^{10^6}.n \leq 10^{10^7}.n$.

$\Longleftrightarrow g(n) \in \Theta(n)$.

④ $g(n) = 5n^2 + 10n \in \Theta(n^2)$.

Pour $c = 10$ et $n_0 = 2$ on a pour tout $n \geq n_0$

$5n^2 + 10 \leq 10n^2$.

⑤ $g(n) = n^3 + 1000n^2 + n + 8 \notin O(n^2)$.

$\exists c\ \exists n_0, \forall n \geq n_0 : g(n) \leq c.n$. ⟸ Le contraire.

$\neq$ $\forall c\ \forall n_0\ \exists n \geq n_0 : g(n) \geq c\,n^2$.

Soient $c > 0$ Et n_0 Deux valeur arbitraires.

alors il existe $n = c + n_0$ $\Rightarrow$ pour assurer que l'equation est vrai.

Tel que $n \geq n_0$.

Et $g(n) \geq c.n^2$.

$$n^3 + 1000n^2 + n + 8 \geq c.n^2.$$

$$\left.\begin{aligned} &\Leftrightarrow n^3 \geq cn^2 - 1000n^2 - n - 8. \\ &\Leftrightarrow n = \frac{n^3}{n^2} \geq c - 1000 - \frac{1}{n} - \frac{8}{n^2}. \end{aligned}\right\} \text{on a divisé tout l'equation sur } n^2.$$

$$\Leftrightarrow n \geq c - 1000 - \frac{1}{n} - \frac{8}{n^2}.$$

$$\Leftrightarrow n \geq c.$$

<u>Question 2</u>: Ordre exact. Démontrer les propriétés suivantes:

a) $g(n) = 5n^2 + 10n \in \Theta(n^2)$.

$\exists\, c > 0, \exists\, c' > 0, \exists\, n_0.$

$\forall\, n \geq n_0 \quad c.n^2 \leq g(n) \leq c'.n^2.$

pour $c = 1$.

$c' = 10$

et $n_0 = 10$

On a pour $\boxed{n \geq 10}$

$$1n^2 \leq 5n^2 + 10n \leq 10n^2.$$

$$c.n^2 \leq g(n) \leq c'.n^2.$$

b) $g(n) = n^2 + 1000\,000\,n \in \Theta(n^2)$.

$$\overset{c}{\overset{\shortparallel}{1}}.n^2 \leq n^2 + 10^6 n \leq \overset{c'}{\overset{\shortparallel}{2}}\,n^2. \text{ quand } n \geq \underset{n_0.}{\underset{\shortparallel}{10^6}}$$

$$10^{12} + 10^{12} \leq 2.10^{12}.$$

③ $g(n) = 4n^2 + n.\log(n) \in \Theta(n^2)$.

$4n^2 \leq 4n^2 + n.\log(n) \leq 5.n^2$
$c = 4$, $c' = 5$ quand $n \geq 1024 = n_0$.

$4.2^{20} + 2^{10}.10 \leq 5\,2^{20}$.

④ $g(n) = 3n + 8 \notin \Theta(n^2)$.

Soient $c > 0$, $c' > 0$ et n_0.

On prend $n = \frac{3}{c} + n_0 + 1 + \frac{8}{c}$.

et alors $\underline{n \geq n_0}$.

Et $c.n^2 \geq g(n)$.

Pour $n \geq 0$ on a.

$c.n^2 \geq 3n + 8$.

$\Leftrightarrow n \geq \frac{3n+8}{c.n} = \frac{3}{c} + \frac{8}{c.n}$. // $\frac{8}{c.\frac{8}{c}} = 1$.

$+1$

Exercice 2 :

Q1) Un algorithme itératif :

```
int n, p
    res = 1
    while p > 0.
        res = res * x.
        p = p - 1.
    return res.
```

2. Non, car la taille de l'entrée est :

$t_x = \lfloor \log_2(x) \rfloor + 1$.

$+ \; t_p = \lfloor \log_2(p) \rfloor + 1$

$p = 2^{t_p}$ tours de boucle.

Donc la complexité est au moins exponentielle en l'une des deux entrées, donc pas linéaire.

3- init x, k
res $= x$. $\rightarrow A_a$
while $k > 0$. $\rightarrow A_k$

res = res x res $\rightarrow t_{res}^2 < (t_{x^p})^2$
k = k - 1. $\rightarrow t_k$ $\Big\}$ 2^{t_x}.

return res $\rightarrow t_{res}$
$\leftarrow x^p$ avec $p = 2^k$.

⊛ <u>La complexité :</u>

$t_x + t_k$ avec $t_x = (\log_2(x)) + 1$ et $t_x = (\log_2(k)) + 1$
On borne supérieurement $t_x = \log_2(x)$ et $t_k = \log_2(k)$.

$\leq t_x + 2^{t_x}(t_k + (t_{x^p})^2 + t_k) + t_k + t_{x^p}$.

$N = x^{2^k}$.

$\log_a N = \dfrac{\log_b N}{\log_b a} = b$
$a = z$

$t_x = [\log_2(x)] + 1$.

$t_{x^p} = \log_2(x^p)$

$\log_2(2^n) = n$.

$$t_{x^p} = t_{x^{2^k}} = \log_2(x^{2^k}) = \frac{\log_x(x^{2^k})}{\log_x(2)} = \frac{2^k}{\log_x(2)} = \frac{2^k}{\frac{\log_x(2)}{\log_2(x)}}$$

$\log_a N = \dfrac{\log_b N}{\log_b a}$ $N = x^{2^k}$
$a = z$

$= 2^k \log_2(x)$.

$$= \log_2(x) + k(\log(k) + 2^{2k}(\log_2(x))^2 + \log_2(k)) + \log_2(k) + 2^k \cdot \log_2(x).$$

$$= \log_2(x)[1 + 2^k] + \log_2(k)[2k + 1] + (\log_2(x))^2 \cdot [k \cdot 2^{2k}]$$

$$= O\left(k \cdot \log_2(k) + 2^{2k} \cdot k \cdot (\log_2(x))^2\right).$$

TD de complexité

La Suite.

for i = 1 to n.
for j = 1 to n.
res = [i] × [j].

EXO 3 : TD 1 : produits et puissance de matrice carrées.

Le produit de 2 Matrice A et B :

Entrée 2 Matrice A, B de taille n
Sortie 2 Matrice A, B de taille n.

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} \begin{pmatrix} 19 & 22 \\ 43 & \end{pmatrix}$$

for i=1, i=n, i++) n
for j=1, j=n, j++). n
(A[i][j]) × (B[i][j]) n²

Test.

P une matrice carrée d'ordre n.
pour i allant de 1 a n
pour j allant de 1 a n
Som = 0.
pour K allant de 1 a n
Som = A[i, K]. B[K, j].
P[i, j] = Som
retourner P.

2) La taille de l'entrée est :

$(2 . n^2)^{\frac{3}{2}} = n^3 \times 2^{\frac{3}{2}}$. Complexité entre linéaire et quadratique.

3) Un algorithme itératif : réaliser le calcule A^P:

Entrée A une matrice entier d'ordre n.
P un entier strict Sortie A^P.

n^2 Q = A
n for (K = 1, K < P - 1, K++).
$n \times n^3$ Q = PRODUIT(Q, A).
return Q.

$\Gamma(P \cdot n^3) = \Gamma(2^{tP} \cdot t_n^{\frac{3}{2}})$

$t_P = \log_2(P)$ t_P = taille de l'entier P.

t_A = taille de l'entier A.

4- ② puissance 2 (A, P) :

Entrée . A une matrice carrée d'ordre n.

k un entier positif.

Sortie : A^{2^k}.

$A^1 = A^{2^0}$.

n^2 $Q = A$

1 Pour i de 1 à k.

$k \times n^3$ Q = Produit(Q, Q).

Retourne Q.

$A^1 \times A^1$ A^2

$A^2 \times A^2$ A^4

$A^4 \times A^4$ A^8

$A^8 \times A^8$. A^{16}

$\Gamma\, k \cdot n^3 = \Gamma(2^{t_k} \cdot t_A^{\frac{3}{2}})$

- $P = 2^k$ donc cet algo est beaucoup plus rapide que le précédent.

Exo4 : le pire cas pour le nombre de comparaison.

Est celui où le tableau il est déjà trié.

ce qui donne $i+1$ comparaison pour l'élément en position i.

Donc le total :

$$\sum_{i=1}^{n-1}(i+1) = \sum_{i=1}^{n-1} i + (n-1) = \frac{(n-1)n}{2} + (n-1)$$

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$