Chapitre 14

Complexité en temps

1 Introduction :

Algorithme : lisibilité, conception, facilité de compréhension, rapidité.
Complexité d'un algorithme $\Rightarrow$ données machine (vitesse d'exécution d'une opération élémentaire).
(si réalisation effective) $\Rightarrow$ algorithme sous-jacent au programme, taille des données du programme.

Evaluation asymptotique :
Définition : T(n) : algorithme donné, données de taille n fournies est le maximum des temps d'exécution de l'algorithme sur des données de même taille (le cas le plus défavorable).

Exemple : Soient trois algorithmes qui réalisent la même tâche, de temps d'exécution $T(n) = 100n$, $5n^2$ et 2^n. On peut faire la représentation graphique de T en fonction de n pour les comparer.
Choix suivant la taille des données de l'algorithme le plus rapide.

Complexité en place mémoire
Précision et stabilité numérique des méthodes
Formules d'évaluations asymptotiques(pour n grand)

1. $\displaystyle\sum_{k=1}^{n} k^i = \frac{n^{i+1}}{i+1} + O(n^i)$
2. formule de Stirling
   $n! = \sqrt{2\pi n}(n/e)^n(1 + O(1/n))$
   $n! = O(n^n)$ d'où $\log(n!) = 0(n \log n)$
3. le n^{me} nombre harmonique défini par $H_n = 1 + \dfrac{1}{2} + \cdots + \dfrac{1}{n}$
   On a $H_n = \log n + \gamma + 0(1/n)$ avec γ la constante d'Euler (≈ 0.579, sans régularité connue).

Rappel :
Un algorithme a un temps d'exécution $T(n)$ en $O(f(n))$ s'il existe une constante C telle que pour n assez grand, on ait $T(n) \leq Cf(n)$.
Ex : $T(n) = 3n^2 + n + 56 \Rightarrow T(n) \leq 4n^2 \Rightarrow O(n^2)$
Complexité temporelle
- polynomiale ssi $T(n)$ est un $O(n^\alpha)$, $\alpha \in \mathbb{N}$
- exponentielle ssi $T(n)$ est un $O(\xi^n)$, $\xi \in \mathbb{R}^{+*}$

2 Exponentiation :

Calcul de x^n pour un entier n donné et x réel

première méthode

On utilise les relations $x^0 = 1$ et $x^n = x \times x^{n-1}$ $n \geq 1$

```
PUISSAN1(x,n)(réel : x, entier : n
debut
Si n = 0 alors retourner(1)
sinon retourner(x = PUISSAN1(x, n-1))
fin
```

Cet algorithme se termine bien puisqu'à chaque appel n va décroître.
Temps d'exécution de l'algorithme noté $T(n)$:

$$T(n) = m + T(n-1) \quad n \geq 1 \quad T(0) = c$$

où m et c sont des constantes (m est le temps nécessaire à une multiplication et c à une affectation)
Soit $T(n) = c + m \times n$
L'algorithme est donc linéaire, en $O(n)$
Remarque : Il est équivalent à l'algorithme itératif usuel.

deuxième méthode :
On utilise le principe "diviser pour régner" :

Si n est pair, $x^n = x^{n/2} \times x^{n/2}$
Si n est impair, $x^n = x \times x^{(n-1)/2} \times x^{(n-1)/2}$

```
PUISSAN2(x, n)(réel x, entier n)
debut
Si n = 0 alors retourner(1)
sinon debut
y = PUISSAN2(x, E(n/2))
Si n pair alors retourner(y*y)
sinon retourner(x*y*y)
fin
fin
```

Cet algorithme se termine bien puisqu'à chaque appel n va décroître.
Etude de la complexité en temps de l'algorithme :
Soit $T(n)$ son temps d'exécution :
$T(n) = T(n/2) + m + t$ si n est pair
(1) $\quad T(n) = T((n-1)/2) + 2m + t$ si n est impair
$T(0) = c$
où m, t et c sont des constantes (c est le temps d'un test et d'une affectation, t de deux tests et d'une affectation, et m d'une multiplication).
Etude de cette suite
Décomposons n en base 2 : $n = 2^\alpha + \sum_{\alpha > i} \alpha_i 2^i$, $\alpha_i = 0$ ou 1
La parité de n se lit sur α_0
La partie entière de n est donnée par $E(n/2) = 2^{\alpha-1} + \sum_{1 \leq i < \alpha} \alpha_i 2^{i-1}$
Par récurrence, les relations (1) deviennent :

$\forall i \geq 1 \; T(n) = T(2^{\alpha-i} + \sum_{i \leq k < \alpha} \alpha_k 2^{k-i}) + (m+t) \times Nb0(i-1) + (2m+t) \times Nb1(i-1)$
où $Nb0(i) = |\{k, \alpha_k = 0 \; et \; 0 \leq k \leq i\}|$
$Nb1(i) = |\{k, \alpha_k = 1 \; et \; 0 \leq k \leq i\}|$
On en déduit $(i = \alpha)$
$T(n) = c + m \times (Nb0(\alpha-1) + 2 \times Nb1(\alpha-1) + 2) + t \times (Nb0(\alpha-1) + Nb1(\alpha-1) + 1)$, soit :

$$(2) \; T(n) = c + m \times (\alpha + Nb1(\alpha-1) + 1) + t \times \alpha$$

On a $2^\alpha \leq n < 2^{\alpha+1}$ d'où $\alpha = \log_2(n)$
De (2) on obtient :

$$c + (m+t+1) \times \log_2(n) \leq T(n) \leq c + (2m+t) \times \log_2(n)$$

L'algorithme fonctionne, à des constantes multiplicatives près, en $\log_2(n)$.
Cette méthode est donc en $0(\log_2(n))$, bien meilleur et avec la même précision que la première.

troisième méthode :
On considère $x^n = (x^2)^{n/2}$ si n pair
$x^n = x \times (x^2)^{(n-1)/2}$ si n impair

```
PUISSAN3(x, n)(réel x, entier n)
debut
Si n = 0 alors retourner(1)
sinon debut
y = PUISSAN3(x*x, E(n/2))
Si n pair alors retourner(y)
Sinon retourner(x*y)
fin
fin
```

Cet algorithme se termine bien puisqu'à chaque appel n va décroître.
Estimation du temps de calcul $T(n)$ de cet algorithme :
Si n est pair, on ne fait qu'une unique multiplication pour le calcul de x^2, si n est impair on en fait 2. Cela montre que $T(n)$ va satisfaire la même relation de récurrence que dans la deuxième méthode en trouvant un temps de calcul similaire. On a par conséquent un algorithme en $0(\log_2(n))$.

3 Produit de deux matrices :

Soient A et $B \in \mathcal{M}_n(\mathbb{R})$.
On suppose que n est une puissance de 2.
On décompose $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ et $B = \begin{pmatrix} w & x \\ y & z \end{pmatrix}$
où les sous-matrices introduites ont une taille $n/2$.
L'algorithme standard de multiplication donne 8 multiplications et 4 additions de matrices de taille $n/2$ pour faire AB. On a alors un temps d'exécution $T(n)$ = $8{*}T(n/2)$ + $c{*}n^2$ d'où un algorithme en $O(n^3)$.
Remarque : C'est le temps de l'algorithme itératif usuel

Méthode de Strassen (1969)

7 multiplications et 15 additions à partir des sous-matrices a, b, c, d, w, x, y, z
Description : On fait les huits additions suivantes :

$$S_1 = c + d, S_2 = S_1 - a, S_3 = a - c, S_4 = b - S_2, S_5 = x - w, S_6 = z - S_5, S_7 = z - x, S_8 = S_6 - y$$

On effectue les sept multiplications :

$$P_1 = S_2S_6, P_2 = aw, P_3 = by, P_4 = S_3S_7, P_5 = S_1S_5, P_6 = S_4z, P_7 = dS_8$$

On calcule les deux additions :

$$A_1 = P_1 + P_2, A_2 = A_1 + P_4$$

Le résultat AB est donné en effectuant les 5 additions :

$$A^*B = \begin{pmatrix} P_2 + P_3 & A_1 + P_5 + P_6 \\ A_2 - P_7 & A_2 + P_5 \end{pmatrix}$$

Effectuer une addition entre deux matrices de taille n se réalise en $O(n^2)$. L'algorithme de Strassen aura un temps d'exécution $T(n)$ tel que

$$T(n) = 7 * T(n/2) + a * n^2$$

D'où un algorithme en $O(n^{\log_2 7})$ soit $O(n^{2.81})$ environ, qui a un temps d'exécution pour n assez grand moindre que celui de l'algorithme précédent en $O(n^3)$.

3.1 Construction des formules algébriques :

Soient 2 matrices $X = \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix}$ et $Y = \begin{pmatrix} y_1 & y_2 \\ y_3 & y_4 \end{pmatrix}$ et le produit $C = XY = \begin{pmatrix} f_1(x,y) & f_2(x,y) \\ f_3(x,y) & f_4(x,y) \end{pmatrix}$ avec $f_1(x,y) = x_1y_1 + x_2y_3,\ f_2(x,y) = x_1y_2 + x_2y_4,\ f_3(x,y) = x_3y_1 + x_4y_3$ et $f_4(x,y) = x_3y_2 + x_4y_4$ en notant $x = (x_1, x_2, x_3, x_4)^t$ et $y = (y_1, y_2, y_3, y_4)^t$
Les f_i sont des formes bilinéaires ($f(x,y) = \sum_{i,j=1}^{n} a_{ij}x_iy_j = x^tAy$). On a :

$$\begin{matrix} f_1(x,y) = x^tB_1y \\ f_2(x,y) = x^tB_2y \end{matrix} \quad \text{avec } B_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} B_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{matrix} f_3(x,y) = x^tB_3y \\ f_4(x,y) = x^tB_4y \end{matrix} \quad \text{avec } B_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} B_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Calculer les 4 formes bilinéaires revient à calculer la forme trilinéaire ($f(x,y,z) = \sum_{i,j,k=1}^{m} f_{ijk}x_iy_jz_k$) suivante :
(1) $f(x,y,z) = z_1f_1(x,y) + z_2f_2(x,y) + z_3f_3(x,y) + z_4f_4(x,y)$ avec $z = (z_1, z_2, z_3, z_4)^t$
$z = e_1 = (1,0,0,0)^t)$ donne $f_1(x,y)$.
Pour une forme bilinéaire $f(x,y) = x^tAy$ on peut écrire la matrice A sous la forme $UV^t = \sum_{i=1}^{p} U_iV_i^t$ avec $U, V \in \mathcal{M}_{np}(\mathbb{R})$ et les U_i, V_i étant des vecteurs de $\mathbb{R}^n$, la matrice $U_iV_i^t$ est dite antiscalaire. D'où $f(x,y) = x^t(\sum_{i=1}^{p} U_iV_i^t)y =$

$\sum_{i=1}^{p}(x^tU_i)(V_i^ty) = \sum_{i=1}^{p}(U_i^tx)(V_i^ty)$. De même on peut écrire une forme trilinéaire sous la forme suivante :
$f(x,y,z) = \sum_{i=1}^{q}(U_i^tx)(V_i^ty)(W_i^tz)$ et en notant $\varphi_i(z)$ la forme linéaire W_i^tz, on obtient $f(x,y,z) = \sum_{i=1}^{q}\varphi_i(z)x^t(U_iV_i^t)y$. Soit $f(x,y,z) = x^tM(z)y$ avec $M(z) = \sum_{i=1}^{q}\varphi_i(z)(U_iV_i^t)$, pour les formes trilinéaires $q \leq n^3$ (dénombrement des coefficients f_{ijk}).
<u>Applications</u>
La relation (1) peut s'écrire $f(x,y,z) = \sum_{i=1}^{4} z_if_i(x,y) = \sum_{i=1}^{4} x^tz_iB_iy$ d'où $M(z) = \sum_{i=1}^{4} z_iB_i = \sum_{i=1}^{q}\varphi_i(z)(U_iV_i^t)$, on a $q \leq 8$ et on peut montrer que $q < 8$. Prenons $q = 7$. On obtient $M(z) = \begin{pmatrix} z_1 & z_2 & 0 & 0 \\ 0 & 0 & z_1 & z_2 \\ z_3 & z_4 & 0 & 0 \\ 0 & 0 & z_3 & z_4 \end{pmatrix}$ et par permutation des lignes 2 et 3 en notant P la matrice associée on a $PM(z) = \begin{pmatrix} z_1 & z_2 & 0 & 0 \\ z_3 & z_4 & 0 & 0 \\ 0 & 0 & z_1 & z_2 \\ 0 & 0 & z_3 & z_4 \end{pmatrix}$. On peut écrire $PM(z) = \sum_{i=1}^{7}\varphi_i(z)H_i$ sous la forme suivante :

$$PM(z) = \begin{pmatrix} \begin{pmatrix} z_1 - z_4 + z_4 & z_2 + z_4 - z_4 \\ z_3 + z_4 - z_4 & z_4 \end{pmatrix} & 0 \\ 0 & \begin{pmatrix} z_1 & z_2 + z_1 - z_1 \\ z_3 + z_1 - z_1 & z_4 - z_1 + z_1 \end{pmatrix} \end{pmatrix}$$

$$= z_1 \begin{pmatrix} 0 & 0 \\ 0 & \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \end{pmatrix} + z_4 \begin{pmatrix} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} & 0 \\ 0 & 0 \end{pmatrix} + N(z)$$

$$N(z) = \begin{pmatrix} z_1 - z_4 & z_2 + z_4 & 0 & (z_1 - z_4) + (z_2 + z_4) - (z_2 + z_1) \\ z_3 + z_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & z_2 + z_1 \\ -(z_1 - z_4) - (z_3 + z_4) + (z_3 + z_1) & 0 & z_3 + z_1 & z_4 - z_1 \end{pmatrix}$$

$$= (z_1 - z_4) \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 \end{pmatrix} + (z_2 + z_4) \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} + (z_1 + z_3) \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

$$+ (z_2 + z_1) \begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} + (z_3 + z_4) \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix}$$

La matrice $PM(z)$ est égale à $\sum_{i=1}^{7}\varphi_i(z)H_i$ et pour retrouver $M(z)$ on pose $T_i = P^{-1}H_i = PH_i$ (P matrice de permutation $P^2 = I$, I matrice identité) et on a $M(z) = \sum_{i=1}^{7}\varphi_i(z)T_i$ avec $T_i = U_iV_i^t$. On a 7 matrices antiscalaires correspondant aux 7 multiplications de base x^tT_iy, puis en prenant $z = e_i$ (le i^{eme} vecteur de la base canonique de $\mathbb{R}^4$) on obtient le i^{eme} coefficient du produit matriciel $XY = C$, $f_i(x,y) = \sum_{j=1}^{7}\varphi_j(e_i)x^tT_jy$.
<u>exemple :</u>
Pour calculer $f_1(x,y)$, on prend e_1 et on obtient une combinaison linéaire des 7

produits (multiplications de base) $x^t T_j y$, $j = 1, \ldots, 7$.
remarque :
Il n'y a pas unicité des formules du produit de Strassen, dans notre cas si on choisit la matrice T_1 on calcule $T_1 = PH_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix}$ puis $x^t T_1 y = (x_2 - x_4)(y_3 - y_4)$, les autres produits sont $(x_1 - x_3)(y_1 - y_2)$, $(x_1 - x_4)(y_1 + y_4)$, $x_1(y_2 + y_4)$, $x_4(y_1 + y_3)$, $y_4(x_2 - x_1)$ et $y_1(x_3 - x_4)$ et on trouve $f_1(x, y)$ en prenant dans la forme $f(x, y, z)$ $z = e_1$ on obtient $f_1(x, y) = (x_2 - x_4)(y_3 - y_4) + (x_1 - x_4)(y_1 + y_4) + x_4(y_1 + y_3) + y_4(x_2 - x_1)$ qui est bien une combinaison linéaire des 4 produits (multiplications de base) déjà calculés.