# IFT2125 - Introduction à l'algorithmique

## Diviser pour régner (B&B chapitre 7)

Pierre McKenzie

DIRO, Université de Montréal

Automne 2017

# Structure générale d'un algo diviser-pour-régner

B&B Section 7.2

**function** $DC(x)$

  **if** $x$ is sufficiently small or simple **then return** $adhoc(x)$

  decompose $x$ into smaller instances $x_1, x_2, \ldots, x_\ell$

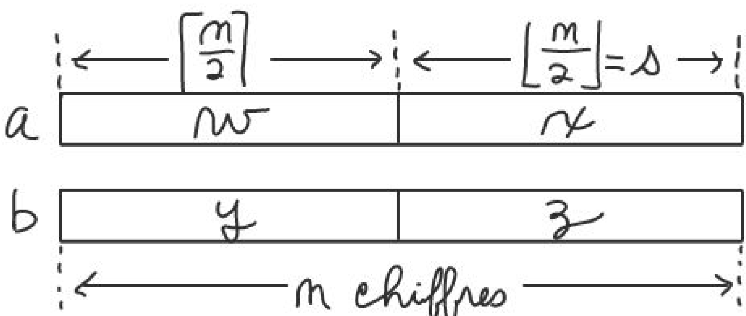
  **for** $i \leftarrow 1$ **to** $\ell$ **do** $y_i \leftarrow DC(x_i)$

  recombine the $y_i$ 's to obtain a solution $y$ for $x$

  **return** $y$

# Multiplication de grands entiers

B&B Section 7.1




$$
\begin{aligned}
a &= 2^s w + x \\
b &= 2^s y + z \\
\hline
ab &= 2^{2s} wy + 2^s (wz + xy) + xz
\end{aligned}
$$

PRODUIT

**DONNÉE:** entier $a$ de $n$ chiffres, entier $b$ de $m$ chiffres
**CALCULER:** $a \times b$

- L'algo diviser pour régner analysé en cours :
  $O(\max(n, m) \cdot [\min(m, n)]^{\alpha})$ où $\alpha = \log_2 3 - 1 = 0,58496$
  Donc $O(n^{1,58496})$ lorsque $n = m$.
  On peut montrer aussi $\Theta$.
  (L'algo classique donnerait $\alpha = 1$ : exercice.)

- B&B problème 7.2 (avec $m = n$) demande de calculer $a \times b$ à l'aide de sommes, de décalages et de 5 produits de nombres de $\lceil \frac{n}{3} \rceil$ chiffres.
  Coût de cet algo vu en démo ?

PRODUIT

**DONNÉE:** entier $a$ de $n$ chiffres, entier $b$ de $m$ chiffres
**CALCULER:** $a \times b$

- L'algo diviser pour régner analysé en cours :
  $O(\max(n, m) \cdot [\min(m, n)]^{\alpha})$ où $\alpha = \log_2 3 - 1 = 0,58496$
  Donc $O(n^{1,58496})$ lorsque $n = m$.
  On peut montrer aussi $\Theta$.
  (L'algo classique donnerait $\alpha = 1$ : exercice.)

- B&B problème 7.2 (avec $m = n$) demande de calculer $a \times b$ à l'aide de sommes, de décalages et de 5 produits de nombres de $\lceil \frac{n}{3} \rceil$ chiffres.
  Coût de cet algo vu en démo ?
  $O(n^{\log_3 5}) = O(n^{1,46497})$, donc mieux que $O(n^{\log_2 3})$.

# Fouille dichotomique d'un tableau trié

B&B Section 7.3

Rappel (bien connu) :

**function** *binsearch*$(T[1..n], x)$
  **if** $n = 0$ **or** $x > T[n]$ **then return** $n + 1$
  **else return** *binrec*$(T[1..n], x)$

**function** *binrec*$(T[i..j], x)$
  {Binary search for $x$ in subarray $T[i..j]$
  with the promise that $T[i-1] < x \leq T[j]$}
  **if** $i = j$ **then return** $i$
  $k \leftarrow (i + j) \div 2$
  **if** $x \leq T[k]$ **then return** *binrec*$(T[i..k], x)$
      **else return** *binrec*$(T[k+1..j], x)$

# Fouille dichotomique : temps de calcul

```
function binsearch(T[1..n], x)
    if n = 0 or x > T[n] then return n + 1
    else return binrec(T[1..n], x)

function binrec(T[i..j], x)
    {Binary search for x in subarray T[i..j]
     with the promise that T[i − 1] < x ≤ T[j]}
    if i = j then return i
    k ← (i + j) ÷ 2
    if x ≤ T[k] then return binrec(T[i..k], x)
                else return binrec(T[k + 1..j], x)
```

- On suppose coût unitaire pour accéder à l'élément $i$
- $T(n) \in T(\lceil \frac{n}{2} \rceil) + O(1)$
- $T(n) \in T(\lfloor \frac{n}{2} \rfloor) + \Omega(1)$
- Cas 3 du transparent 17 sur l'analyse d'algos :
  $a = 1, b = 2, \varepsilon = 0 \implies T(n) \in \Theta(n^{\log_2 1} \log n) = \Theta(\log n)$.

Rappel du tri par fusion (merge sort) :

- trier deux demi-tableaux puis fusionner
- $T(n) \in T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + \Theta(n)$
- Cas 3 du transparent 17 sur l'analyse d'algos :
  $a = 2, b = 2, \varepsilon = 0 \Longrightarrow T(n) \in \Theta(n^{\log_2 2} \log n) = \Theta(n \log n)$.

Quicksort (que nous n'étudierons pas plus à fond) :

- choisir pivot, trier $\{x : x \leq \text{pivot}\}$, trier $\{x : x > \text{pivot}\}$, concaténer
- en pire cas : $T(n) \in T(n-1) + \Omega(n) \implies T(n) \in \Omega(n^2)$
- en moyenne (si tableaux équiprobables) : analyse difficile, $O(n \log n)$.

# Médiane

B&B Section 7.5

<u>MÉDIANE</u>

**DONNÉE:** tableau de $n$ éléments, non trié
**CALCULER:** l'élément qui serait le $\lceil \frac{n}{2} \rceil$ième si le tableau était trié

- Pas besoin de trier le tableau : on peut trouver la médiane en $\Theta(n)$ !
- Étonnant et à voir en détail, en cours.

EXPONENTIATION

**DONNÉE:** $a, n \in \mathbb{N}$

**CALCULER:** $\underbrace{a \times a \times a \times \cdots \times a}_{n \text{ fois}}$

Deux contextes :

- $\times$ à coût unitaire (ex : produits modulo un nombre $m$ fixé)
  nous étudierons ce cas.
- $\times$ à coût qui croît avec le nombre de chiffres
  serait pertinent à l'arithmétique exacte de très grands nombres
  le meilleur ordre s'obtient en combinant dpr pour $a \times b$ et dpr pour $a^n$

# Produit matriciel

B&B Section 7.6

<u>PRODUIT MATRICIEL</u>

**DONNÉE:** Matrices carrées $A \in \mathbb{R}^{m \times m}$ et $B \in \mathbb{R}^{m \times m}$
**CALCULER:** Matrice $A \times B$

- L'algorithme naïf utilise $\Omega(n^3)$ produits scalaires
- Supposons produits scalaires à coût unitaire
- Alors Strassen résoud PRODUIT MATRICIEL en $O(m^{\log_2 7})$ !

# Produit matriciel

L'idée géniale de Strassen (B&B page 242) :

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \text{ and } B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

be two matrices to be multiplied. Consider the following operations, each of which involves just one multiplication.

$$\left.\begin{aligned}
m_1 &= (a_{21} + a_{22} - a_{11})(b_{22} - b_{12} + b_{11}) \\
m_2 &= a_{11} b_{11} \\
m_3 &= a_{12} b_{21} \\
m_4 &= (a_{11} - a_{21})(b_{22} - b_{12}) \\
m_5 &= (a_{21} + a_{22})(b_{12} - b_{11}) \\
m_6 &= (a_{12} - a_{21} + a_{11} - a_{22}) b_{22} \\
m_7 &= a_{22}(b_{11} + b_{22} - b_{12} - b_{21})
\end{aligned}\right\} \tag{7.9}$$

We leave the reader to verify that the required product $AB$ is given by the following matrix.

$$C = \begin{pmatrix} m_2 + m_3 & m_1 + m_2 + m_5 + m_6 \\ m_1 + m_2 + m_4 - m_7 & m_1 + m_2 + m_4 + m_5 \end{pmatrix} \tag{7.10}$$

# Produit matriciel

Quelques salves dans la "guerre des décimales"

- Strassen 1969 : $\log_2 7 = 2,8074$
- Pan 1978 : $\log_{70} 143640 = 2,7951$
- Bini et al 1979 : $< 2,78$
- Schönhage 1981 : $< 2,522$
- Romani 1982 : $< 2,517$
- Coppersmith Winograd 1986 : $< 2,496$
- Strassen 1986 : $< 2,479$
- Coppersmith Winograd 1989 : $< 2,376$
- 2012 : $< 2,373$ (record mondial de Virginia Vassilevska Williams [1] )
- 2017 : est-ce que $2 + \varepsilon$ est atteignable ?

1. Multiplying matrices in $O(n^{2,373})$ time, Stanford University, juillet 2014, 73 pages.

# Cryptographie à clef publique

B&B Section 7.8

La tâche :

- Alice veut envoyer en secret un entier $a$ de 500 chiffres à Bob
- N'importe qui peut lire ce qu'enverra Alice
- Personne d'autre que Bob ne doit pouvoir décoder le message

La difficulté supplémentaire : Alice et Bob ne doivent pas supposer qu'ils possèdent au préalable un secret quelconque qu'eux seuls partagent.

Possible ? Étonnamment oui...sous hypothèse calculatoire !

# Les outils mathématiques disponibles

1. $\mathrm{pgcd}(a, b) = 1 \implies (\exists s \in \mathbb{N})[\ as \equiv 1 \mod b\ ]$

2. (Fermat) :
   $p$ premier et $0 < a < p \implies a^{p-1} \equiv 1 \mod p$

3. (fonction indicatrice d'Euler, définition) :
   $\varphi(z) = |\{a \in [1..z] : \mathrm{pgcd}(a, z) = 1\}|$

4. $0 \leq a < \underbrace{pq}_{\text{premiers}} \implies (\forall x \equiv 1 \mod \varphi(pq))\ [\ a^x \equiv a \mod pq\ ]$

# Les outils calculatoires (polynomiaux) disponibles

1. EULER

   **DONNÉE:** nombres premiers $p_1, \ldots, p_k$ (avec répétitions)
   **CALCULER:** $\varphi(p_1 \times p_2 \times \cdots \times p_k)$.
   Car $\varphi(p_1 \times \cdots \times p_k) = p_1 \times \cdots \times p_k \times \underbrace{(1 - \frac{1}{p_{i_1}}) \times \cdots \times (1 - \frac{1}{p_{i_\ell}})}_{\text{répétitions supprimées}}$
   On n'aura besoin que de $\varphi(pq) = (p-1)(q-1)$

2. PUISSANCE

   **DONNÉE:** naturels $a, n, z$
   **CALCULER:** $a^n \mod z$
   Par exponentiation rapide (modulo $z$ à chaque étape).

3. INVERSEMOD

   **DONNÉE:** naturels $a$ et $z$ tels que $\mathrm{pgcd}(a, z) = 1$
   **CALCULER:** naturel $s$ tel que $as \equiv 1 \mod z$
   Par l'algorithme d'Euclide étendu (Introduction, transparent 16).

# Le maillon faible de la crypto : hypothèses calculatoires

Fait : aucun algo polynomial pour ci-dessous n'est du domaine public.

Hypothèse : Aucun tel algorithme n'existe !

1. RACINEMOD

   **DONNÉE:** naturels $c, n, z$

   **CALCULER:** naturel $a$ tel que $c \equiv a^n \mod z$ si un tel $a$ existe.

   Doit demeurer difficile même sous la promesse qu'un $a$ existe et que $z$ est "semi-premier", i.e., $z = pq$ avec $p, q$ premiers

2. ni, a fortiori :

   FACTORISATION

   **DONNÉE:** naturel $z$

   **CALCULER:** décomposition de $z$ en produits de nombres premiers

   ("a fortiori" car découvrir $z = pq$ découvre $\varphi(z)$, qui découvre $s$ tel que $ns \equiv 1 \mod \varphi(z)$ et qui résout RACINEMOD en posant $a = c^s$, puisqu'alors $a^n = c^{ns} \equiv c \mod z$)

1. Bob
   1. choisit deux nombres premiers $p$ et $q$ de 251 chiffres chacun
   2. calcule $z = pq$ et $\phi = (p-1)(q-1)$
   3. choisit un nombre $n \in [1..z-1]$
   4. calcule $s \in [1..z-1]$ tel que $ns = 1 \mod \phi$
      (si échec alors $\text{pgcd}(n, \phi) \neq 1$ alors reprendre le choix de $n$)
   5. annonce $z$ et $n$ publiquement

1. Bob
    1. choisit deux nombres premiers $p$ et $q$ de 251 chiffres chacun
    2. calcule $z = pq$ et $\phi = (p-1)(q-1)$
    3. choisit un nombre $n \in [1..z-1]$
    4. calcule $s \in [1..z-1]$ tel que $ns = 1 \mod \phi$
       (si échec alors $\text{pgcd}(n, \phi) \neq 1$ alors reprendre le choix de $n$)
    5. annonce $z$ et $n$ publiquement
2. Alice
    1. calcule $m = a^n \mod z$
    2. envoie $m$ (que tous observent) à Bob

1. Bob
   1. choisit deux nombres premiers $p$ et $q$ de 251 chiffres chacun
   2. calcule $z = pq$ et $\phi = (p-1)(q-1)$
   3. choisit un nombre $n \in [1..z-1]$
   4. calcule $s \in [1..z-1]$ tel que $ns = 1 \mod \phi$
      (si échec alors $\mathrm{pgcd}(n, \phi) \neq 1$ alors reprendre le choix de $n$)
   5. annonce $z$ et $n$ publiquement
2. Alice
   1. calcule $m = a^n \mod z$
   2. envoie $m$ (que tous observent) à Bob
3. Bob
   1. Bob calcule $m^s \mod z = a^{ns} \mod z = a$, le secret d'Alice !

Les étapes 2.1 et 3.1 ne sont rendues possibles que par diviser-pour-régner.