# Évaluation et Interpolation rapide.

*une généralisation de la FFT*

Adrien Poteaux

*inspiré du cours d'Alin Bostan aux JNCF 2010*[1]

Univ. Lille 1 - LIFL, équipe Calcul Formel

Groupe de Travail de Calcul Formel

20 Mai 2014

1. merci à lui pour le partage de ses fichiers `.tex` !

# Algorithmes modulaires

Choix de modulos $m_i$

$\Downarrow$

Réductions du problème modulo $m_i$

$\Downarrow$

Calculs modulo $m_i$

$\Downarrow$

Reconstruction : Théorème des Restes Chinois

- Résultats corrects si modulos assez nombreux et indépendants,
- Intérêt si explosion des expressions intermédiaires
  - déterminant de matrices entières,
  - calcul de pgcd de polynômes à coefficients entiers...

# Choix des modulos ?

Dans le cadre polynomial, si

- $\xi \in \mathbb{A}$ racine de l'unité d'ordre suffisamment élevé,
- la taille des données (degré) est suffisamment grande,

un choix intéressant est :

- $m_i = X - \xi^i$

$$\Longrightarrow \quad \textit{évaluation-interpolation similaire à la FFT}$$

Remarque : parfois, choix des modulo imposé par le problème

# Évaluation multi-point (univariée)

- $\mathbb{A}$ anneau commutatif unitaire,
- $\alpha_0, \ldots, \alpha_{n-1} \in \mathbb{A}$, $\alpha_i \neq \alpha_j$.

## Le problème

*Étant donné* $P \in \mathbb{A}[X]$ *t.q.* $\deg_X(P) < n$, *calculer les valeurs*

$$P(\alpha_0), \ldots, P(\alpha_{n-1})$$

# Interpolation

- $\mathbb{A}$ anneau commutatif unitaire,
- $\alpha_0, \ldots, \alpha_{n-1} \in \mathbb{A}$.

## Le problème

*Étant donnés* $\beta_0, \ldots, \beta_{n-1} \in \mathbb{A}$ *trouver* $P = \sum_{i=0}^{n-1} p_i X^i \in \mathbb{A}[X]$ *t.q.*

$$P(\alpha_0) = \beta_0, \ldots, P(\alpha_{n-1}) = \beta_{n-1}$$

Remarque : solution unique si $\alpha_i - \alpha_j \in \mathbb{A}^\times$ pour $i \neq j$

# Lien avec la matrice de Vandermonde

$$V = \begin{pmatrix} 1 & \alpha_0 & \dots & \alpha_0^{n-1} \\ 1 & \alpha_1 & \dots & \alpha_1^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & \alpha_{n-1} & \dots & \alpha_{n-1}^{n-1} \end{pmatrix} ; \; p = \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_{n-1} \end{pmatrix} ; \; b = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{n-1} \end{pmatrix}$$

- Évaluation : calculer $V \cdot p$
- Interpolation : résoudre le système linéaire $V \cdot p = b$

Deux conséquences :

- Premières complexités en respectivement $O(n^2)$ et $O(n^\omega)$,
- Preuve de la remarque précédente ($\det(V) = \prod_{i<j}(\alpha_j - \alpha_i)$)

# Interpolation de Lagrange

- Formule d'interpolation de Lagrange :

$$P(X) = \sum_{i=0}^{n-1} \beta_i \prod_{j \neq i} \frac{X - \alpha_j}{\alpha_i - \alpha_j}$$

# Interpolation de Lagrange

- Formule d'interpolation de Lagrange :

$$P(X) = \sum_{i=0}^{n-1} \beta_i \prod_{j \neq i} \frac{X - \alpha_j}{\alpha_i - \alpha_j} = \sum_{i=0}^{n-1} \beta_i \frac{A_i(X)}{A_i(\alpha_i)}$$

# Interpolation de Lagrange

- Formule d'interpolation de Lagrange :

$$P(X) = \sum_{i=0}^{n-1} \beta_i \frac{A_i(X)}{A_i(\alpha_i)} \; ; \; A_i(X) = \prod_{j \neq i} X - \alpha_j$$

- Calcul direct :

1. Calculer les $A_i(X)$, (*naïvement*) $O(n^3)$
2. Évaluations $A_i(\alpha_i)$, $O(n^2)$
3. Combinaisons linéaires. $O(n^2)$

# Interpolation de Lagrange "améliorée"

$$P(X) = \sum_{i=0}^{n-1} \beta_i \frac{A_i(X)}{A_i(\alpha_i)} \; ; \; A_i(X) = \prod_{j \neq i} X - \alpha_j$$

1. $A \leftarrow 1; P \leftarrow 0$
2. Pour $i = 0, \cdots, n-1$ faire

   $A \leftarrow A \cdot (X - \alpha_i)$
3. Pour $i = 0, \cdots, n-1$ faire

   $A_i(X) \leftarrow A/(X - \alpha_i)$

   $q_i \leftarrow A(\alpha_i)$

   $P \leftarrow P + \beta_i A_i / q_i$

# Interpolation de Lagrange "améliorée"

$$P(X) = \sum_{i=0}^{n-1} \beta_i \frac{A_i(X)}{A_i(\alpha_i)} \; ; \; A_i(X) = \prod_{j \neq i} X - \alpha_j$$

1. $A \leftarrow 1; P \leftarrow 0$
2. Pour $i = 0, \cdots, n-1$ faire

   $A \leftarrow A \cdot (X - \alpha_i)$ $\qquad O(i)$
3. Pour $i = 0, \cdots, n-1$ faire

   $A_i(X) \leftarrow A/(X - \alpha_i)$

   $q_i \leftarrow A(\alpha_i)$

   $P \leftarrow P + \beta_i A_i / q_i$

# Interpolation de Lagrange "améliorée"

$$P(X) = \sum_{i=0}^{n-1} \beta_i \frac{A_i(X)}{A_i(\alpha_i)} \; ; \; A_i(X) = \prod_{j \neq i} X - \alpha_j$$

1. $A \leftarrow 1; P \leftarrow 0$
2. Pour $i = 0, \cdots, n-1$ faire $\qquad O(n^2)$

   $A \leftarrow A \cdot (X - \alpha_i) \qquad O(i)$
3. Pour $i = 0, \cdots, n-1$ faire

   $A_i(X) \leftarrow A/(X - \alpha_i)$

   $q_i \leftarrow A(\alpha_i)$

   $P \leftarrow P + \beta_i A_i / q_i$

# Interpolation de Lagrange "améliorée"

$$P(X) = \sum_{i=0}^{n-1} \beta_i \frac{A_i(X)}{A_i(\alpha_i)} \; ; \; A_i(X) = \prod_{j \neq i} X - \alpha_j$$

1. $A \leftarrow 1; P \leftarrow 0$
2. Pour $i = 0, \cdots, n-1$ faire $\qquad O(n^2)$
   $A \leftarrow A \cdot (X - \alpha_i) \qquad O(i)$
3. Pour $i = 0, \cdots, n-1$ faire
   $\left.\begin{array}{l} A_i(X) \leftarrow A/(X - \alpha_i) \\ q_i \leftarrow A(\alpha_i) \\ P \leftarrow P + \beta_i A_i / q_i \end{array}\right\} \quad O(n)$

# Interpolation de Lagrange "améliorée"

$$P(X) = \sum_{i=0}^{n-1} \beta_i \frac{A_i(X)}{A_i(\alpha_i)} \; ; \; A_i(X) = \prod_{j \neq i} X - \alpha_j$$

1. $A \leftarrow 1; P \leftarrow 0$
2. Pour $i = 0, \cdots, n-1$ faire — $O(n^2)$
   $A \leftarrow A \cdot (X - \alpha_i)$ — $O(i)$
3. Pour $i = 0, \cdots, n-1$ faire — $O(n^2)$
   $A_i(X) \leftarrow A/(X - \alpha_i)$
   $q_i \leftarrow A(\alpha_i)$
   $P \leftarrow P + \beta_i A_i / q_i$
   — $O(n)$

# Algorithme rapide ? Diviser pour régner

- Hypothèse dans la suite : $n = 2^k$
- Idée du processus :
  1. Calculer $P_0 = P \mod (X - \alpha_0) \cdots (X - \alpha_{n/2-1})$ et $P_1 = P \mod (X - \alpha_{n/2}) \cdots (X - \alpha_{n-1})$,
  2. Appliquer récursivement ce procédé à $P_0$ et $P_1$.
  3. Terminer quand $n = 1$ : on a $P \mod (X - \alpha_i) = P(\alpha_i)$.
- Préliminaire : calculer *rapidement* les modulo $\prod(X - \alpha_i)$

Remarque : dans la suite, $O(n)$ omis quand $\mathrm{M}(n)$ dans la complexité

# Préliminaire : propriétés pour la multiplication rapide

La fonction $\mathsf{M}(n)$ (complexité pour multiplier deux polynômes de degrés $< n$) vérifie :

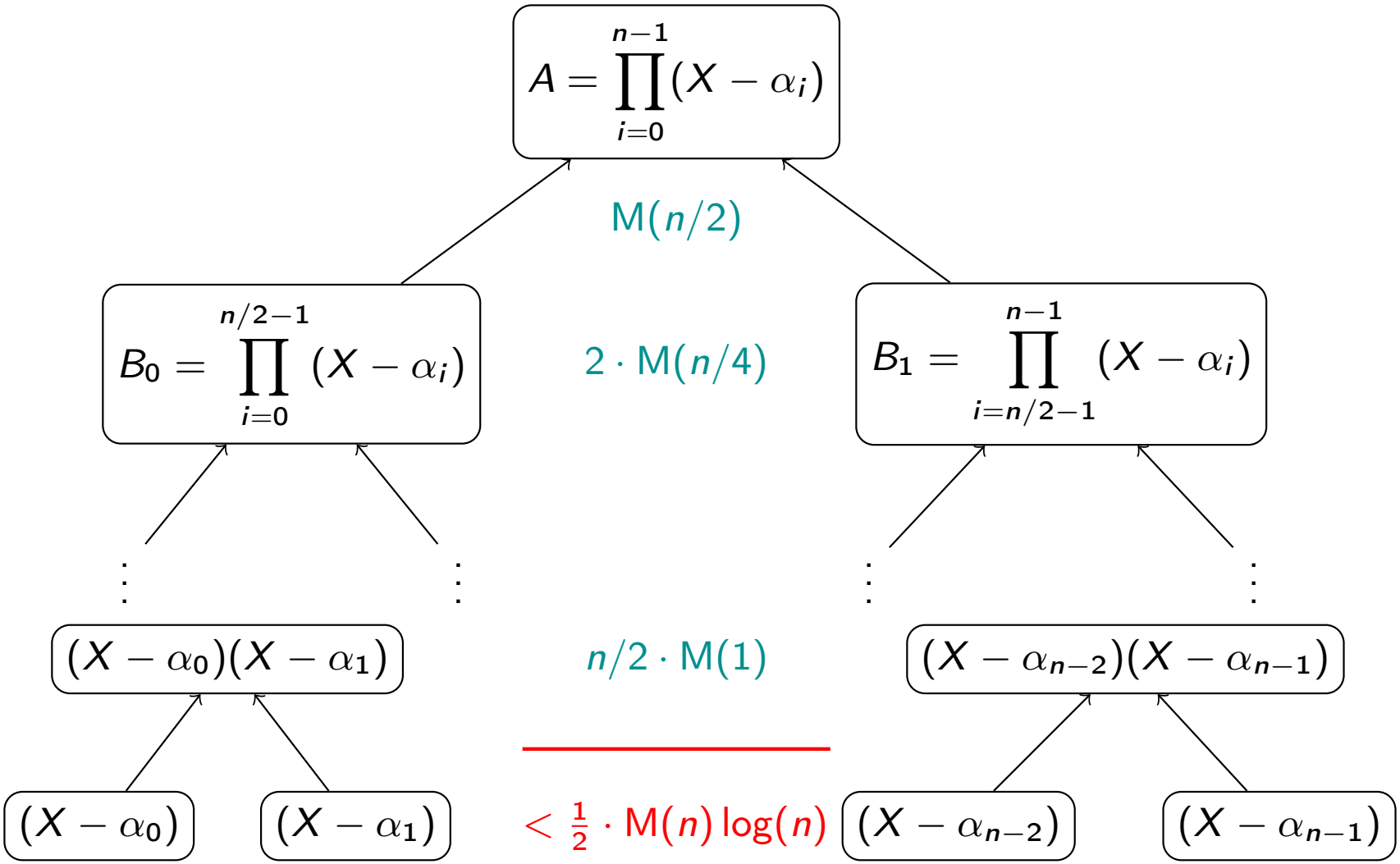
- $\mathsf{M}(n/2) + \mathsf{M}(n/4) + \cdots + \mathsf{M}(1) < \mathsf{M}(n)$
- $2\mathsf{M}(n/2) + 4\mathsf{M}(n/4) + \cdots + n\mathsf{M}(1) < \mathsf{M}(n)\log(n)$

(*cf e.g. thèse Alin Bostan, Lemma 2.2 page 49*)

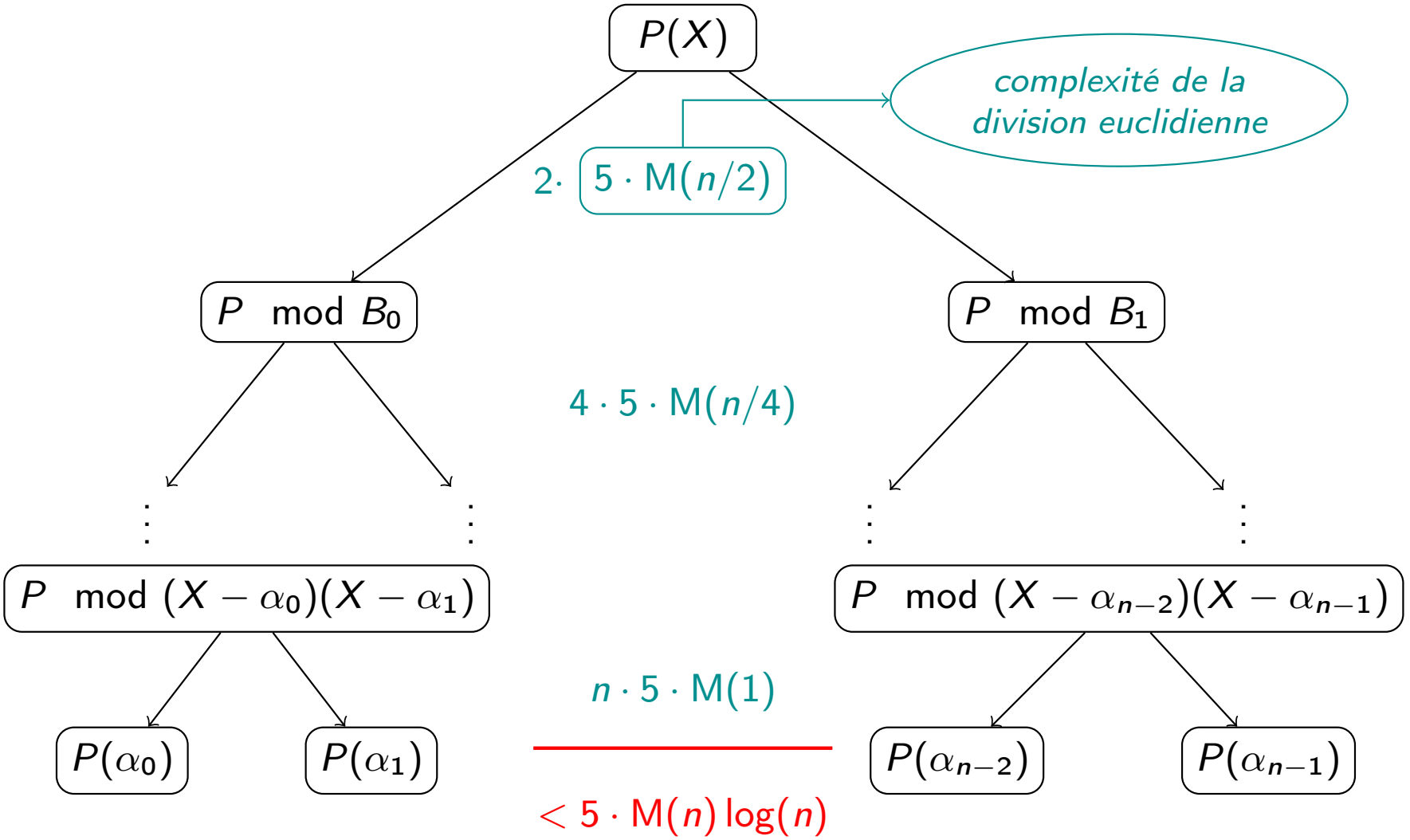
Pour rappel, $\mathsf{M}(n) =$

- $O(n^2)$ pour l'algorithme "naïf",
- $O(n^{\log_2 3}) = O(n^{1.59})$ avec l'algorithme de Karatsuba,
- $O(n\log(n))$ via FFT si racines de l'unité dans $\mathbb{A}$.
- $O(n\log(n)\log\log(n))$ sinon

# Arbre des sous-produits



# Évaluation multi-point rapide : complexité [Borodin-Moenck, 1974]




Complexité totale : $< \frac{11}{2}\cdot \mathrm{M}(n)\log(n)$ ($\frac{9}{2}$ possible)

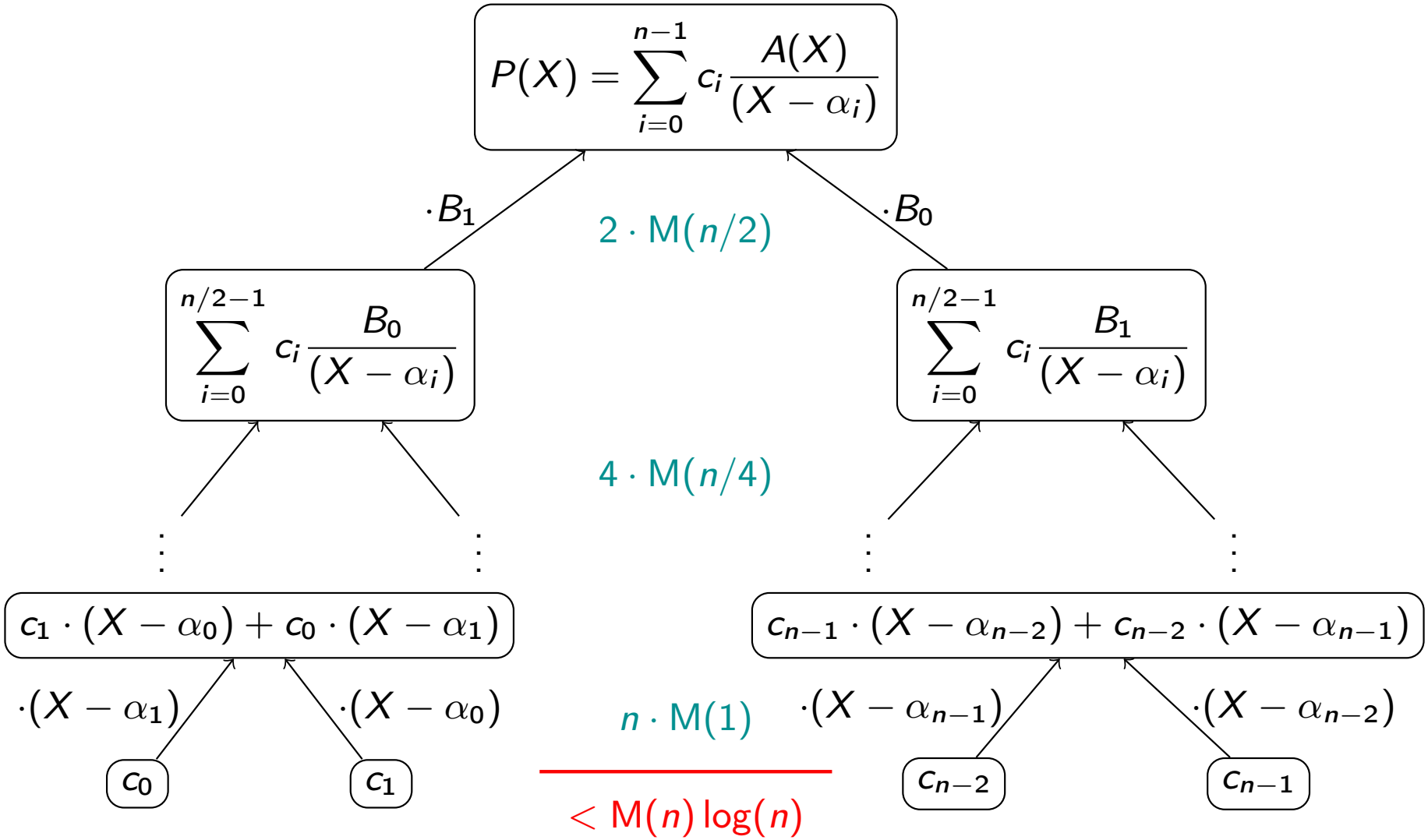
# Interpolation rapide [Borodin-Moenck, 1974]

Formule de Lagrange modifiée - $A(X) = \prod_{i=0}^{n-1}(X - \alpha_i)$

$$P(X) = A(X) \cdot \sum_{i=0}^{n-1} \frac{\beta_i / A'(\alpha_i)}{X - \alpha_i}$$

- Calculer $c_i = \beta_i / A'(\alpha_i)$ (évaluation multi-point) $\frac{11}{2} \cdot \mathrm{M}(n) \log n$
- Calculer $\sum_{i=0}^{n-1} c_i \frac{A(X)}{X - \alpha_i}$ : diviser pour régner $\mathrm{M}(n) \log n$
- Coût total : $\frac{13}{2} \cdot \mathrm{M}(n) \log n$

# Interpolation rapide : combinaisons linéaires [Borodin-Moenck, 1974]



# Arbre des produits : cas géometrique [Bostan-Schost, 2005]

Le problème : $q \in \mathbb{A}^\times$ donné, calculer $A = \prod_{i=0}^{n-1}(X - q^i)$

Idée : $B_0 = \prod_{i=0}^{n/2-1}(X - q^i) \Longrightarrow B_1 = \prod_{i=n/2}^{n-1}(X - q^i)$ par homothétie :

$$B_1(X) = B_0\left(\frac{X}{q^{n/2}}\right) \cdot q^{(n/2)^2}$$

Diviser pour régner :

- Calcul récursif de $B_0(X)$
- Déduction de $B_1(X)$ à partir de $B_0(X)$ $O(n)$
- Retourner $A(X) = B_0(X) \cdot B_1(X)$ $\mathrm{M}(n/2)$

Coût total : $\mathrm{M}(n)$ $(+O(n))$

# Évaluation multi-point rapide, cas géometrique [Bluestein, 1970]

Problème : $q \in \mathbb{A}^\times$, $P \in \mathbb{A}[X]_{<n}$ donnés, calculer $P(q^i), 0 \leq i < n$

On cherche : $$P(q^i) = \sum_{j=0}^{n-1} c_j q^{ij}, \qquad 0 \leq i < n$$

Astuce de Bluestein :

$$ij = \frac{(i+j)^2 - i^2 - j^2}{2} \implies q^{ij} = q^{(i+j)^2/2} \cdot q^{-i^2/2} \cdot q^{-j^2/2}$$

$$\text{D'où :} \quad P(q^i) = q^{-i^2/2} \cdot \underbrace{\sum_{j=0}^{n-1} c_j q^{-j^2/2} \cdot q^{(i+j)^2/2}}_{\text{convolution :}}$$

$$\left[X^{n-1+i}\right] \left(\sum_{k=0}^{n-1} c_k q^{-k^2/2} X^{n-k-1}\right) \left(\sum_{\ell=0}^{2n-2} q^{\ell^2/2} X^\ell\right)$$

Conclusion : Évaluation rapide pour $\alpha_i = q^i$ en $O(\mathrm{M}(n))$

# Interpolation rapide, cas géometrique [B-Schost, 2005]

Problème : $q \in \mathbb{A}\times$, $v_0, \ldots, v_{n-1} \in \mathbb{A}$ donnés, calculer $P \in \mathbb{K}[X]_{<n}$ t.q. $P(1) = v_0, \ldots, P(q^{n-1}) = v_{n-1}$

Algorithme rapide : Formule de Lagrange modifiée

$$P(X) = A(X) \cdot \sum_{i=0}^{n-1} \frac{v_i / A'(q^i)}{X - q^i}, \qquad A = \prod_i (X - q^i)$$

- Calculer $\prod_{i=0}^{n-1}(X - q^i)$ : diviser pour régner $O(\mathrm{M}(n))$
- Calculer $c_i = v_i / A'(q^i)$ : algorithme de Bluestein $O(\mathrm{M}(n))$
- Calculer $\sum_{i=0}^{n-1} \frac{c_i}{X - q^i}$ : diviser pour régner $O(\mathrm{M}(n))$

# Combinaisons linéaires : cas géometrique [B-Schost, 2005]

On a : $q \in \mathbb{A}\times$, $c_0, \ldots, c_{n-1} \in \mathbb{A}$

On veut calculer : $R(x) = \sum_{i=0}^{n-1} \frac{c_i}{x - q^i}$

Idée 1 : changer de représentation ; suffit pour calculer $R \mod x^n$

Idée 2 : $R \mod x^n$ = évaluation multi-point en $1, q^{-1}, \ldots, q^{-(n-1)}$

$$\sum_{i=0}^{n-1} \frac{c_i}{x - q^i} \mod x^n = -\sum_{i=0}^{n-1} \left( \sum_{j=0}^{n-1} c_i q^{-i(j+1)} x^j \right) = -\sum_{j=0}^{n-1} C(q^{-j-1}) x^j$$

Conclusion : algorithme en $O(\mathrm{M}(n))$

(*généralisation de la transformée de Fourier inverse*)

# Conclusion

- Algorithme “simple” $O(n^2)$
- Algorithme rapide $O(\mathrm{M}(n)\log n)$
- Cas géométrique $O(\mathrm{M}(n))$

*Plus de détails ?*

- cours d'Alin Bostan aux JNCF 2010,
- Modern Computer Algebra. . .