Chapitre 4 : Algorithmes diviser-pour-régner

INF4705 - Analyse et conception d'algorithmes

Gilles Pesant   Simon Brockbank

École Polytechnique Montréal
gilles.pesant@polymtl.ca, simon.brockbank@polymtl.ca

Hiver 2017

Plan

Principe Général

Principe

On ramène la résolution d'un exemplaire de problème à celle de plusieurs exemplaires de plus petite taille dont les solutions servent à obtenir une solution à l'exemplaire originel.

Patron de conception Diviser-pour-régner

fonction diviser-pour-regner(x {exemplaire}) : y {solution}
  si x est petit ou simple **alors retourner** algo-simple(x) ;
  décomposer x en sous-exemplaires $x_1, \ldots, x_\ell$;
  pour $i = 1$ **à** ℓ **faire**
    $y_i \leftarrow$ diviser-pour-regner(x_i) ;
  recombiner les y_i en une solution y pour x ;
  retourner y

Décroître-pour-régner

- À strictement parler, Diviser-pour-régner décompose en au moins deux sous-exemplaires.
- Sinon on parlera du patron Décroître-pour-régner.
- Ex. fouille dichotomique
- On peut habituellement les exprimer de manière itérative.

Comment appliquer le patron de conception Diviser-pour-régner ?

Trois décisions à prendre

1. Décomposition : nombre et taille des morceaux ?
2. Recombinaison : comment recoller les morceaux ?
3. Seuil de récursivité : quand est-ce que x est assez petit ?

fonction diviser-pour-regner(x {exemplaire}) : y {solution}
  si x est petit ou simple **alors retourner** algo-simple(x) ;
  décomposer x en sous-exemplaires $x_1, \ldots, x_\ell$;
  pour $i = 1$ **à** ℓ **faire**
    $y_i \leftarrow$ diviser-pour-regner(x_i) ;
  recombiner les y_i en une solution y pour x ;
  retourner y

Décision 1 : Décomposition

Une décomposition équilibrée

Considérons

$$T(n) = T(\lambda n) + T((1-\lambda)n) + cn, \quad 0 < \lambda < 1$$

si $\lambda = 1/2$ (parfaitement équilibré) :

$$T(n) = 2T(n/2) + cn \longrightarrow T(n) \in \Theta(n \lg n)$$

Décision 1 : Décomposition

Une décomposition équilibrée

Considérons

$$T(n) = T(\lambda n) + T((1-\lambda)n) + cn, \quad 0 < \lambda < 1$$

si $\lambda = 1/n$ (parfaitement déséquilibré) :

$$T(n) = T(1) + T(n-1) + cn \longrightarrow T(n) \in \Theta(n^2)$$

Décision 2 : Recombinaison

Il faut pouvoir combiner efficacement les solutions aux sous-exemplaires en une solution à notre exemplaire originel.

Ex. Plus grand facteur commun de n et m (où $m < n$)

Quels sont nos sous-exemplaires ? $(n, \lceil m/2 \rceil)$ et $(\lceil n/2 \rceil, m)$?
Comment en reconstituer une solution ?

C'est plutôt un exemple du patron *Décroître-pour-régner* :

$$pgfc(m, n) = pgfc(n \text{ modulo } m, m)$$

Décision 3 : Seuil de récursivité

À partir d'une certaine taille d'exemplaire, il est plus efficace d'arrêter les appels récursifs et de le résoudre directement avec un autre algorithme (non-récursif)
même si sa complexité asymptotique est moins bonne.

Ça ne change pas l'analyse asymptotique théorique de notre algorithme mais le choix d'un bon seuil l'accélérera en pratique.

Décision 3 : Seuil de récursivité

Exemple : Multiplication de grands entiers

Considérons

$$T(n) = \begin{cases} h(n) & \text{si } n \leq n_{\text{min}} \\ 3T(\lceil n/2 \rceil) + g(n) & \text{sinon} \end{cases}$$

Si $h(n) = n^2$ μsec. et $g(n) = 16n$ μsec.,
la multiplication d'entiers à $n = 5000$ chiffres prend :

Décision 3 : Seuil de récursivité

Exemple : Multiplication de grands entiers

Considérons

$$T(n) = \begin{cases} h(n) & \text{si } n \leq n_{\text{min}} \\ 3T(\lceil n/2 \rceil) + g(n) & \text{sinon} \end{cases}$$

Si $h(n) = n^2$ $\mu sec.$ et $g(n) = 16n$ $\mu sec.$,
la multiplication d'entiers à $n = 5000$ chiffres prend :

- ~ 25 *sec*. de façon classique (en $\Theta(h(n))$) ;

Décision 3 : Seuil de récursivité

Exemple : Multiplication de grands entiers

Considérons

$$T(n) = \begin{cases} h(n) & \text{si } n \leq n_{\text{min}} \\ 3T(\lceil n/2 \rceil) + g(n) & \text{sinon} \end{cases}$$

Si $h(n) = n^2$ $\mu sec.$ et $g(n) = 16n$ $\mu sec.$,
la multiplication d'entiers à $n = 5000$ chiffres prend :

- ~ 25 *sec*. de façon classique (en $\Theta(h(n))$) ;
- ~ 41 *sec*. en diviser-pour-régner avec $n_{\text{min}} = 1$;

Décision 3 : Seuil de récursivité

Exemple : Multiplication de grands entiers

Considérons

$$T(n) = \begin{cases} h(n) & \text{si } n \leq n_{\text{min}} \\ 3T(\lceil n/2 \rceil) + g(n) & \text{sinon} \end{cases}$$

Si $h(n) = n^2\ \mu sec.$ et $g(n) = 16n\ \mu sec.$,
la multiplication d'entiers à $n = 5000$ chiffres prend :

- $\sim 25\ sec.$ de façon classique (en $\Theta(h(n))$) ;
- $\sim 41\ sec.$ en diviser-pour-régner avec $n_{\text{min}} = 1$;
- $\sim 6\ sec.$ en diviser-pour-régner avec $n_{\text{min}} = 64$.

Décision 3 : Seuil de récursivité

Méthode hybride

Soient $h(n) = (an^2 + bn + c)\mu sec.$ et $g(n) = (dn + e)\mu sec.$

1. Évaluer empiriquement les constantes (a, b, c, d, e).
2. Déterminer un seuil $n_{\min}$ pour lequel $T(n_{\min})$ est sensiblement le même qu'on utilise "algo-simple" directement ou après un niveau de récursivité :

$$an_{\min}^2 + bn_{\min} + c \quad \simeq \quad 3(a\lceil n_{\min}/2\rceil^2 + b\lceil n_{\min}/2\rceil + c) + dn_{\min} + e$$

Méthode essai-erreur

On cherche un seuil qui donne un meilleur temps de calcul que les seuils voisins (par exemple, que sa moitié et son double).

Plan

pour un même problème…

avec un même patron de conception…

deux algorithmes très différents

Tri par fusion

Idée

On sépare les éléments en deux parties (presque) de même taille qu'on trie récursivement, puis on fusionne les suites triées en préservant l'ordre.

Si n est petit, on utilise par exemple un tri par insertion.

Exemple

Tableau initial :

4	8	1	3	2	6	7	5

1er niveau de décomposition :

4	8	1	3

2	6	7	5

2e niveau de décomposition :

4	8

1	3

2	6

7	5

Tri :

4	8

1	3

2	6

5	7

1er niveau de fusion :

1	3	4	8

2	5	6	7

2e niveau de fusion :

Exemple

Tableau initial :

4	8	1	3	2	6	7	5

1er niveau de décomposition :

4	8	1	3

2	6	7	5

2e niveau de décomposition :

4	8

1	3

2	6

7	5

Tri :

4	8

1	3

2	6

5	7

1er niveau de fusion :

1	3	4	8

2	5	6	7

2e niveau de fusion :

1							

Exemple

Tableau initial :

4	8	1	3	2	6	7	5

1er niveau de décomposition :

4	8	1	3

2	6	7	5

2e niveau de décomposition :

4	8

1	3

2	6

7	5

Tri :

4	8

1	3

2	6

5	7

1er niveau de fusion :

1	3	4	8

2	5	6	7

2e niveau de fusion :

1	2						

Exemple

Tableau initial :

4	8	1	3	2	6	7	5

1er niveau de décomposition :

4	8	1	3	2	6	7	5

2e niveau de décomposition :

4	8	1	3	2	6	7	5

Tri :

4	8	1	3	2	6	5	7

1er niveau de fusion :

1	3	4	8	2	5	6	7

2e niveau de fusion :

1	2	3					

Exemple

Tableau initial :

4	8	1	3	2	6	7	5

1er niveau de décomposition :

4	8	1	3

2	6	7	5

2e niveau de décomposition :

4	8

1	3

2	6

7	5

Tri :

4	8

1	3

2	6

5	7

1er niveau de fusion :

1	3	4	8

2	5	6	7

2e niveau de fusion :

1	2	3	4				

Exemple

Tableau initial :

4	8	1	3	2	6	7	5

1er niveau de décomposition :

4	8	1	3

2	6	7	5

2e niveau de décomposition :

4	8

1	3

2	6

7	5

Tri :

4	8

1	3

2	6

5	7

1er niveau de fusion :

1	3	4	8

2	5	6	7

2e niveau de fusion :

1	2	3	4	5	6	7	8

Tri par fusion

- décomposition triviale (et équilibrée)
- recombinaison facile

- Le gros du travail est fait *après* les appels récursifs : on applique un algorithme (glouton) de fusion de deux listes triées en temps linéaire

Tri par fusion

Analyse de la complexité temps

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + g(n), \quad g(n) \in \Theta(n)$$

Analyse de la complexité espace

La fusion utilise un espace mémoire supplémentaire de taille n ; la complexité totale est donc $\Theta(n)$.

Tri par fusion

Analyse de la complexité temps

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + g(n), \quad g(n) \in \Theta(n)$$

$$T(n) = 2T(n/2) + cn,$$

Analyse de la complexité espace

La fusion utilise un espace mémoire supplémentaire de taille n ; la complexité totale est donc $\Theta(n)$.

Tri par fusion

Analyse de la complexité temps

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + g(n), \quad g(n) \in \Theta(n)$$

$$T(n) = 2T(n/2) + cn, \qquad \ell = 2; b = 2; k = 1$$

Analyse de la complexité espace

La fusion utilise un espace mémoire supplémentaire de taille n ; la complexité totale est donc $\Theta(n)$.

Tri par fusion

Analyse de la complexité temps

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + g(n), \quad g(n) \in \Theta(n)$$

$$T(n) = 2T(n/2) + cn, \qquad \ell = 2; b = 2; k = 1$$

$T(n) \in \Theta(n \lg n)$ en pire cas

Analyse de la complexité espace

La fusion utilise un espace mémoire supplémentaire de taille n ; la complexité totale est donc $\Theta(n)$.

Tri rapide

Idée :

On sépare les entiers de part et d'autre d'un pivot,
on trie récursivement chaque partie,
puis . . . c'est tout !

Si n est petit, on utilise par exemple un tri par insertion.

Exemple

Tableau initial (pivot en jaune)

4	8	1	3	5

Placement initial des pointeurs (ascendant : vert, descendant : rouge)

4	8	1	3	5

Identification d'une paire d'éléments à déplacer

4	8	1	3	5

Échange de 8 et 3

4	3	1	8	5

On continue . . . les pointeurs se croisent

4	3	1	8	5

Échange du pivot et du dernier élément de la partie inférieure

1	3	4	8	5

On applique récursivement à chacune des deux parties

1	3	4	8	5

Tableau trié

1	3	4	5	8

Tri rapide

- décomposition facile ; décomposition *équilibrée* difficile
- recombinaison triviale
- Le gros du travail est fait *avant* les appels récursifs : on peut séparer les éléments de part et d´autre du pivot efficacement (en temps linéaire) et *in situ*.

Décomposition équilibrée

choix du pivot	décomposition équilibrée ?	coût
quelconque	aucune garantie	$\Theta(1)$
élément moyen	bon comportement moyen ?	$\Theta(n)$
élément médian	parfaite	$\Theta(n \log n)$?
médiane de k élts	un compromis	$\Theta(k \log k)$

Analyse du temps de calcul

pivot : élément médian

- en pire cas : $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil - 1) + cn$
  $\ldots \quad T(n) \in \Theta(n \log n)$
- en moyenne : la même chose

pivot : quelconque

- en pire cas : $T(n) = T(n-1) + T(0) + cn$
  $\ldots \quad T(n) \in \Theta(n^2)$
- en moyenne :
  $\ldots \quad \ldots \quad T(n) \in \Theta(n \log n)$

Analyse en moyenne, pivot quelconque

hypothèses de travail

- Les n éléments du tableau sont distincts.
- Les $n!$ permutations initiales sont équiprobables.
  ($\Rightarrow$ idem pour les permutations dans les sous-tableaux)

$$T(n) = \frac{1}{n}\sum_{\ell=1}^{n}(g(n) + T(\ell-1) + T(n-\ell)), \quad g(n) \in \Theta(n)$$

$$T(n) \leq dn + \frac{1}{n}\sum_{\ell=1}^{n}(T(\ell-1) + T(n-\ell)), \quad d > 0$$

Analyse en moyenne, pivot quelconque

Or

$$\sum_{\ell=1}^{n}(T(\ell-1)+T(n-\ell)) = (T(0)+T(n-1))+(T(1)+T(n-2))+\cdots$$

$$\cdots + (T(n-2) + T(1)) + (T(n-1) + T(0))$$

Donc

$$T(n) \leq dn + \frac{2}{n}\sum_{\ell=1}^{n-1} T(\ell), \quad d > 0 \quad (T(0) = 0)$$

Malheureusement les techniques de résolution de relations de récurrence que nous avons vues ne s'appliquent pas ici.

Analyse en moyenne, pivot quelconque

Prouvons par induction sur n que $T(n) \leq c\, n \ln n,\ \ n \geq 2$

base d'induction

$n = 2:\ T(2) \leq 2d + T(1) \overset{?}{\leq} c\, 2 \ln 2$

On choisira un $c \geq (2d + T(1))/2\ln 2$

hypothèse d'induction

$T(i) \leq c\, i \ln i, \quad$ pour $i < n$

Analyse en moyenne, pivot quelconque

pas d'induction

$$\begin{array}{rcll} T(n) & \leq & dn + \frac{2}{n}\sum_{\ell=1}^{n-1} T(\ell) & \\ & \leq & dn + \frac{2}{n}\sum_{\ell=1}^{n-1} c\ell \ln \ell & \text{par l'hypo. d'induction} \\ & \leq & dn + \frac{2c}{n}\int_1^n x \ln x \, dx & \\ & = & dn + \frac{2c}{n}\left[x^2\left(\frac{\ln x}{2} - \frac{1}{4}\right)\right]_{x=1}^{n} & \\ & = & dn + \frac{2c}{n}\left(n^2\left(\frac{\ln n}{2} - \frac{1}{4}\right) + \frac{1}{4}\right) & \\ & = & cn \ln n + \left(dn - \frac{c}{2}n + \frac{c}{2n}\right) & \end{array}$$

Analyse en moyenne, pivot quelconque

pas d'induction (suite)

Nous avons

$$T(n) \leq c\ n \ln n + \left(dn - \frac{c}{2}n + \frac{c}{2n}\right)$$

Il reste à choisir un c tel que $(dn - \frac{c}{2}n + \frac{c}{2n}) \leq 0$
c'est-à-dire tel que $c \geq 2d\frac{n^2}{n^2-1}$

Or $\frac{n^2}{n^2-1} \leq \frac{4}{3}$ lorsque $n \geq 2$

Donc il suffit de prendre $c \geq \max(\frac{8d}{3}, \frac{2d+T(1)}{2\ln 2})$ $\square$

Plan

Médiane et sélection

Problème de sélection

Trouver l'élément de rang k parmi un ensemble de n entiers.

Si $k = \lceil n/2 \rceil$, c'est le problème de l'élément *médian*.

- On pourrait d'abord trier le tableau puis le parcourir en ordre ascendant jusqu'au $k^{\text{ème}}$: $\Theta(n \lg n)$ en pire cas.
- On pourrait construire un monceau puis retirer un à un k éléments : $\Theta(n)$ puis $\Theta(k \lg n)$ en pire cas
  donc $\Theta(n)$ si $k < n/\lg n$
  mais $\Theta(n \lg n)$ si $k = \lceil n/2 \rceil$ (médiane).
- Voyons comment le faire en $\Theta(n)$ en pire cas (optimal).

Patron Diviser-pour-régner appliqué au problème de sélection

On vise à obtenir une récurrence semblable à

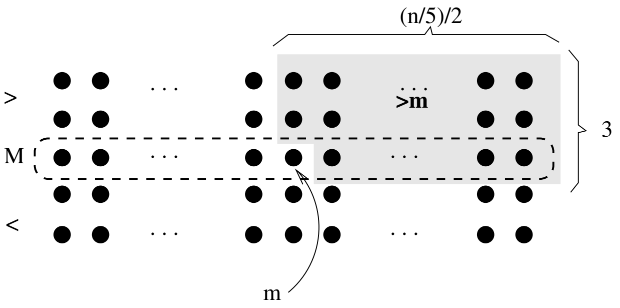
$$T(n) = \begin{cases} c_1 & \text{si } n \leq n_{\min} \\ T(9n/10) + c_2 n & \text{sinon} \end{cases},$$

qui correspond à se débarrasser d'au moins une fraction constante (ici $\frac{1}{10}$) des éléments à chaque étape.

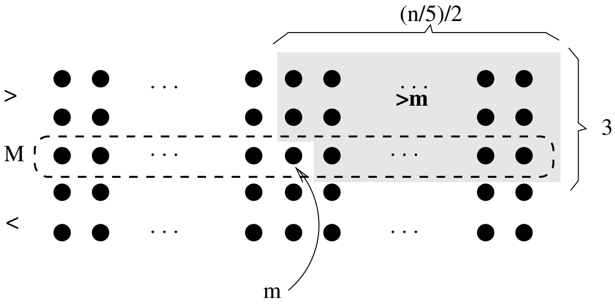
Ébauche de l'algorithme `sélection`(S, k)

1. Diviser les éléments en groupes de cinq (le dernier groupe peut contenir moins d'éléments).
2. Trier chacun des groupes puis rassembler l´élément médian de chacun (le troisième) dans M.
3. $m \leftarrow$ sélection$(M, \lceil \lceil n/5 \rceil /2 \rceil)$ (m est la médiane des médianes, où *pseudo-médiane*).
4. Séparer les éléments de part et d'autre de m en deux ensembles, $S_<$ (plus petits que m) et $S_>$ (plus grands).
5.
    1. Si $k = |S_<| + 1$, retourner m.
    2. Si $k < |S_<| + 1$, retourner sélection$(S_<, k)$.
    3. Si $k > |S_<| + 1$, retourner sélection$(S_>, k - (|S_<| + 1))$.

Cardinalité des ensembles $S_<$ et $S_>$



Cardinalité des ensembles $S_<$ et $S_>$



$$T(n) = T(\lceil n/5 \rceil) + T(7n/10) + c_2 n \Rightarrow \ldots T(n) \simeq T(9n/10) + c_2 n$$

De retour au quicksort...

Note

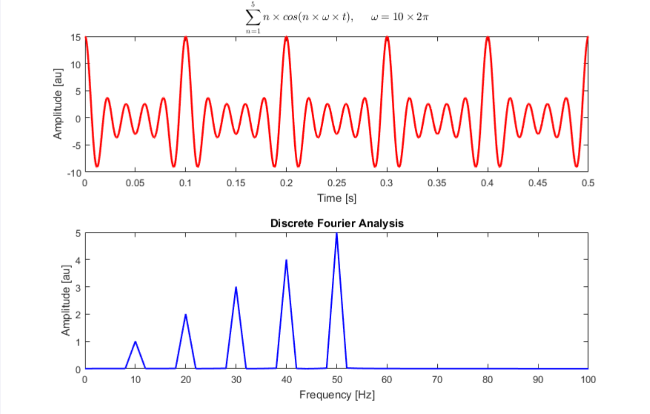
En utilisant cet algorithme pour choisir la médiane comme pivot dans le tri de Hoare, on obtient une décomposition équilibrée et donc un comportement dans $\Theta(n \lg n)$ en pire cas.

Par contre en pratique la constante multiplicative cachée associée au calcul de la médiane fait préférer un choix de pivot plus simple.

Plan

Transformée *discrète* de Fourier

Fait passer une fonction du domaine temporel à fréquenciel



source : DaveSGage CC BY-SA 4.0

Transformée *discrète* de Fourier

Applications

- **Traitement de signal**
  - imagerie computationnelle
  - communication
  - astronomie
  - géologie

    ⋮
- Multiplication de polynômes / calcul de convolutions

Représentation des polynômes

Soit le polynôme en x de degré $n-1$: $A(x) = \sum_{j=0}^{n-1} a_j x^j$.

représentation par coefficients

$\langle a_0, a_1, \ldots, a_{n-1} \rangle$

représentation par paires point-valeur

$\langle (x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1}) \rangle$ où

$$y_k = A(x_k) = \sum_{j=0}^{n-1} a_j (x_k)^j$$

pour $x_0, x_1, \ldots, x_{n-1}$ distincts.

La conversion d'une représentation à l'autre peut se faire en temps $\in \Theta(n^2)$.

Représentation des polynômes

Soit le polynôme en x de degré $n-1$: $A(x) = \sum_{j=0}^{n-1} a_j x^j$.

représentation par coefficients

$\langle a_0, a_1, \ldots, a_{n-1} \rangle$

représentation par paires point-valeur

$\langle (x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1}) \rangle$ où

$$y_k = A(x_k) = \sum_{j=0}^{n-1} a_j (x_k)^j$$

pour $x_0, x_1, \ldots, x_{n-1}$ distincts.

Par un choix judicieux des x_k,
on passe de la 1ère à la 2e en temps $\in \Theta(n \lg n)$.

Transformée *discrète* de Fourier

Racines $n^{\text{ièmes}}$ complexes de l'unité

Nombre complexe ω tel que $\omega^n = 1$

Il y en a n :

$$\omega_n^k = e^{2\pi ik/n} = \cos(2\pi k/n) + i\sin(2\pi k/n), \quad {\scriptstyle k=0,1,\ldots,n-1}.$$

Lemme de la bipartition

Si n est pair, les carrés des n racines $n^{\text{ièmes}}$ de l'unité sont les $n/2$ racines $(n/2)^{\text{ièmes}}$ de l'unité.

$$(\omega_n^{k+n/2})^2 = (\omega_n^k)^2 = \omega_{n/2}^k$$

Transformée *discrète* de Fourier

Étant donné le vecteur $\langle a_0, a_1, \ldots, a_{n-1} \rangle$
(par exemple, les coefficients d'un polynôme),

le vecteur $\langle y_0, y_1, \ldots, y_{n-1} \rangle$ calculé par

$$y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j (\omega_n^k)^j$$

est sa transformée discrète de Fourier.

Transformée *discrète* de Fourier

Étant donné le vecteur $\langle a_0, a_1, \ldots, a_{n-1} \rangle$
(par exemple, les coefficients d'un polynôme),

le vecteur $\langle y_0, y_1, \ldots, y_{n-1} \rangle$ calculé par

$$y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j (\omega_n^k)^j$$

est sa transformée discrète de Fourier.

La transformée rapide de Fourier (FFT) permet de calculer la transformée discrète de Fourier en $\Theta(n \lg n)$.

Transformée rapide de Fourier (FFT)

Remarquons d'abord que $A(x) = A^{pair}(x^2) + xA^{impair}(x^2)$ où

$$A^{pair}(x) = a_0 + a_2 x + a_4 x^2 + \cdots + a_{n-2} x^{n/2-1}$$

$$A^{impair}(x) = a_1 + a_3 x + a_5 x^2 + \cdots + a_{n-1} x^{n/2-1}$$

évaluer A aux points $\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$

$$\Downarrow$$

évaluer A^{pair} et A^{impair} aux points $(\omega_n^0)^2, (\omega_n^1)^2, \ldots, (\omega_n^{n-1})^2$

Transformée rapide de Fourier (FFT)

Remarquons d'abord que $A(x) = A^{pair}(x^2) + xA^{impair}(x^2)$ où

$$A^{pair}(x) = a_0 + a_2 x + a_4 x^2 + \cdots + a_{n-2} x^{n/2-1}$$

$$A^{impair}(x) = a_1 + a_3 x + a_5 x^2 + \cdots + a_{n-1} x^{n/2-1}$$

évaluer A aux points $\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$

$\Downarrow$

évaluer A^{pair} et A^{impair} aux points $\omega_{n/2}^0, \omega_{n/2}^1, \ldots, \omega_{n/2}^{n/2-1}$

par le lemme de la bipartition

Algorithme de Cooley-Tukey ($n = 2^\ell$)

fonction FFT($\langle a_0, a_1, \ldots, a_{n-1} \rangle$)
  si $n = 1$ **alors retourner** a_0 ;
  $\omega_n \leftarrow e^{2\pi i/n}$;
  $\omega \leftarrow 1$;
  $y^{pair} \leftarrow FFT(\langle a_0, a_2, \ldots, a_{n-2} \rangle)$;
  $y^{impair} \leftarrow FFT(\langle a_1, a_3, \ldots, a_{n-1} \rangle)$;
  pour $k = 0$ **à** $n/2 - 1$ **faire**
    $y_k \leftarrow y_k^{pair} + \omega y_k^{impair}$;
    $y_{k+n/2} \leftarrow y_k^{pair} - \omega y_k^{impair}$;
    $\omega \leftarrow \omega \omega_n$;
  retourner y

Analyse asymptotique

fonction $\text{FFT}(\langle a_0, a_1, \ldots, a_{n-1} \rangle)$
  si $n = 1$ **alors retourner** a_0 ;
  $\omega_n \leftarrow e^{2\pi i/n}$;
  $\omega \leftarrow 1$;
  $y^{pair} \leftarrow FFT(\langle a_0, a_2, \ldots, a_{n-2} \rangle)$;
  $y^{impair} \leftarrow FFT(\langle a_1, a_3, \ldots, a_{n-1} \rangle)$;
  pour $k = 0$ **à** $n/2 - 1$ **faire**
    $y_k \leftarrow y_k^{pair} + \omega y_k^{impair}$;
    $y_{k+n/2} \leftarrow y_k^{pair} - \omega y_k^{impair}$;
    $\omega \leftarrow \omega \omega_n$;
  retourner y