

# Algorithmique et structures de données

Diviser pour régner

---

Nicolas Audebert

Mercredi 13 février 2019



Rappels

Diviser pour régner

Quicksort

Tri fusion

Fast Fourier Transform

TP

Il existe de nombreux algorithmes de tris :

- ▶ en complexité quadratique  $O(n^2)$  (tri à bulles, tri par insertion, tri par sélection...)
- ▶ en complexité linéarithmique en moyenne  $O(n.\log(n))$  (tri rapide)
- ▶ en complexité linéarithmique dans le pire cas  $O(n.\log(n))$  (tri par tas, tri fusion)

## Complexité minimale du problème de triage

Dans le cas général, la complexité minimale d'un tri est  $O(n.\log(n))$ .

## En pratique

En pratique, on utilise le tri rapide (*QuickSort*) car il offre les meilleurs performances et on applique une contre-mesure pour éviter de se trouver dans le pire cas.

Il existe de nombreux algorithmes de tris :

- ▶ en complexité quadratique  $O(n^2)$  (tri à bulles, tri par insertion, tri par sélection...)
- ▶ en complexité linéarithmique en moyenne  $O(n.\log(n))$  (tri rapide)
- ▶ en complexité linéarithmique dans le pire cas  $O(n.\log(n))$  (tri par tas, tri fusion)

## Complexité minimale du problème de triage

Dans le cas général, la complexité minimale d'un tri est  $O(n.\log(n))$ .

## En pratique

En pratique, on utilise le tri rapide (*QuickSort*) car il offre les meilleurs performances et on applique une contre-mesure pour éviter de se trouver dans le pire cas.

Il existe de nombreux algorithmes de tris :

- ▶ en complexité quadratique  $O(n^2)$  (tri à bulles, tri par insertion, tri par sélection...)
- ▶ en complexité linéarithmique en moyenne  $O(n.\log(n))$  (tri rapide)
- ▶ en complexité linéarithmique dans le pire cas  $O(n.\log(n))$  (tri par tas, tri fusion)

## Complexité minimale du problème de triage

Dans le cas général, la complexité minimale d'un tri est  $O(n.\log(n))$ .

## En pratique

En pratique, on utilise le tri rapide (*QuickSort*) car il offre les meilleurs performances et on applique une contre-mesure pour éviter de se trouver dans le pire cas.

Rappels

Diviser pour régner

Quicksort

Tri fusion

Fast Fourier Transform

TP

*Diviser pour régner*

## Principe

Diviser un problème en sous-problèmes plus petits, plus faciles à résoudre.

Rappels

Diviser pour régner

Quicksort

Tri fusion

Fast Fourier Transform

TP

# Principe - Rappel



Le parcours du tableau implique  $N - 1$  comparaison. Puis on réitère l'opération sur chaque moitié de tableau.

En notant  $i$  la position du pivot, la complexité s'écrit :

$$C_{moy}(N) = N - 1 + Moy_i[C_{moy}(i - 1) + C_{moy}(N - i)] ,$$

car le tri du tableau de longueur  $N$  implique :

- ▶  $N - 1$  comparaisons pour placer le pivot,
- ▶ le tri d'un tableau de longueur  $i - 1$  (à gauche du pivot),
- ▶ le tri d'un tableau de longueur  $N - i$  (à droite du pivot).

Le pivot peut se retrouver à **n'importe quelle position** de façon **équiprobable** :

$$Moy_i[C_{moy}(i-1) + C_{moy}(N-i)] = \frac{1}{N} \sum_{p=1}^N C_{moy}(p-1) + C_{moy}(N-p)$$

En réinjectant dans la complexité moyenne :

$$C_{moy}(N) = N - 1 + \frac{1}{N} \sum_{p=1}^N C_{moy}(p-1) + C_{moy}(N-p) .$$

Le pivot peut se retrouver à n'importe quelle position de façon équiprobable :

$$\text{Moy}_i[C_{\text{moy}}(i-1) + C_{\text{moy}}(N-i)] = \frac{1}{N} \sum_{p=1}^N C_{\text{moy}}(p-1) + C_{\text{moy}}(N-p)$$

En réinjectant dans la complexité moyenne :

$$C_{\text{moy}}(N) = N - 1 + \frac{1}{N} \sum_{p=1}^N C_{\text{moy}}(p-1) + C_{\text{moy}}(N-p) .$$

$$C_{moy}(N) = N - 1 + \frac{1}{N} \sum_{p=1}^N C_{moy}(p - 1) + C_{moy}(N - p) .$$

Un changement de variable  $q = N - p$  dans la seconde somme donne :

$$C_{moy}(N) = N - 1 + \frac{1}{N} \sum_{p=1}^N C_{moy}(p - 1) + \frac{1}{N} \sum_{q=1}^N C_{moy}(q - 1)$$

Autrement dit, la complexité se réécrit :

$$C_{moy}(N) = N - 1 + \frac{2}{N} \sum_{p=1}^N C_{moy}(p - 1)$$

$$C_{moy}(N) = N - 1 + \frac{1}{N} \sum_{p=1}^N C_{moy}(p - 1) + C_{moy}(N - p) .$$

Un changement de variable  $q = N - p$  dans la seconde somme donne :

$$C_{moy}(N) = N - 1 + \frac{1}{N} \sum_{p=1}^N C_{moy}(p - 1) + \frac{1}{N} \sum_{q=1}^N C_{moy}(q - 1)$$

Autrement dit, la complexité se réécrit :

$$C_{moy}(N) = N - 1 + \frac{2}{N} \sum_{p=1}^N C_{moy}(p - 1)$$

$$C_{moy}(N) = N - 1 + \frac{1}{N} \sum_{p=1}^N C_{moy}(p - 1) + C_{moy}(N - p) .$$

Un changement de variable  $q = N - p$  dans la seconde somme donne :

$$C_{moy}(N) = N - 1 + \frac{1}{N} \sum_{p=1}^N C_{moy}(p - 1) + \frac{1}{N} \sum_{q=1}^N C_{moy}(q - 1)$$

Autrement dit, la complexité se réécrit :

$$C_{moy}(N) = N - 1 + \frac{2}{N} \sum_{p=1}^N C_{moy}(p - 1)$$

En multipliant par  $N$  des deux côtés :

$$NC_{moy}(N) = N(N - 1) + 2 \sum_{p=1}^N C_{moy}(p - 1) \quad (a)$$

En outre, pour un tableau de taille  $N - 1$ , la relation (a) se réécrit :

$$(N - 1)C_{moy}(N - 1) = (N - 1).(N - 2) + 2 \sum_{p=1}^{N-1} C_{moy}(p - 1) \quad (b)$$

En calculant (a) - (b), il vient :

$$NC_{moy}(N) - (N - 1)C_{moy}(N - 1) = 2N + 2C_{moy}(N - 1) \quad .$$

En multipliant par  $N$  des deux côtés :

$$NC_{moy}(N) = N(N - 1) + 2 \sum_{p=1}^N C_{moy}(p - 1) \quad (a)$$

En outre, pour un tableau de taille  $N - 1$ , la relation (a) se réécrit :

$$(N - 1)C_{moy}(N - 1) = (N - 1) \cdot (N - 2) + 2 \sum_{p=1}^{N-1} C_{moy}(p - 1) \quad (b)$$

En calculant (a) - (b), il vient :

$$NC_{moy}(N) - (N - 1)C_{moy}(N - 1) = 2N + 2C_{moy}(N - 1) \quad .$$

En multipliant par  $N$  des deux côtés :

$$NC_{moy}(N) = N(N - 1) + 2 \sum_{p=1}^N C_{moy}(p - 1) \quad (a)$$

En outre, pour un tableau de taille  $N - 1$ , la relation (a) se réécrit :

$$(N - 1)C_{moy}(N - 1) = (N - 1) \cdot (N - 2) + 2 \sum_{p=1}^{N-1} C_{moy}(p - 1) \quad (b)$$

En calculant (a) - (b), il vient :

$$NC_{moy}(N) - (N - 1)C_{moy}(N - 1) = 2N + 2C_{moy}(N - 1) \quad .$$

En simplifiant :

$$NC_{moy}(N) = 2N + (N + 1)C_{moy}(N - 1) .$$

On divise par  $N(N + 1)$  :

$$\frac{C_{moy}(N)}{N + 1} = \frac{2}{N + 1} + \frac{C_{moy}(N - 1)}{N}$$

Puis par récurrence :

$$\frac{C_{moy}(N)}{N + 1} = \sum_{k=3}^{N+1} \frac{2}{k} + \frac{C_{moy}(1)}{2}$$

En simplifiant :

$$NC_{moy}(N) = 2N + (N + 1)C_{moy}(N - 1) .$$

On divise par  $N(N + 1)$  :

$$\frac{C_{moy}(N)}{N + 1} = \frac{2}{N + 1} + \frac{C_{moy}(N - 1)}{N}$$

Puis par récurrence :

$$\frac{C_{moy}(N)}{N + 1} = \sum_{k=3}^{N+1} \frac{2}{k} + \frac{C_{moy}(1)}{2}$$

En simplifiant :

$$NC_{moy}(N) = 2N + (N + 1)C_{moy}(N - 1) .$$

On divise par  $N(N + 1)$  :

$$\frac{C_{moy}(N)}{N + 1} = \frac{2}{N + 1} + \frac{C_{moy}(N - 1)}{N}$$

Puis par récurrence :

$$\frac{C_{moy}(N)}{N + 1} = \sum_{k=3}^{N+1} \frac{2}{k} + \frac{C_{moy}(1)}{2}$$

Comme :

$$\sum_{k=1}^N \frac{1}{k} \sim_{N \rightarrow \infty} \log(N)$$

il vient :

$$C_{moy}(N) \sim_{N \rightarrow \infty} (N+1)\log(N) + (N+1)\frac{C_{moy}(1)}{2}$$

Finalement :

$$C_{moy}(N) = O(N \log(N))$$

Comme :

$$\sum_{k=1}^N \frac{1}{k} \sim_{N \rightarrow \infty} \log(N)$$

il vient :

$$C_{moy}(N) \sim_{N \rightarrow \infty} (N + 1) \log(N) + (N + 1) \frac{C_{moy}(1)}{2}$$

Finalement :

$$C_{moy}(N) = O(N \log(N))$$

Comme :

$$\sum_{k=1}^N \frac{1}{k} \sim_{N \rightarrow \infty} \log(N)$$

il vient :

$$C_{moy}(N) \sim_{N \rightarrow \infty} (N + 1) \log(N) + (N + 1) \frac{C_{moy}(1)}{2}$$

Finalement :

$$C_{moy}(N) = O(N \log(N))$$

Rappels

Diviser pour régner

Quicksort

Tri fusion

Fast Fourier Transform

TP

Le principe du tri fusion est proche du tri rapide. L'idée est de couper un tableau en deux, de trier chaque moitié du tableau, puis de remplir un nouveau tableau avec les sous-tableaux triés.

1 7 3 9 5 10 4 6 8 2

1 7 3 9 5 | 10 4 6 8 2

1 3 5 7 9 | 2 4 6 8 10

1 2 3 4 5 6 7 8 9 10

Le parcours du tableau implique  $N - 1$  comparaison. Donc :

$$C_N = (N - 1) + C_i + C_{N-i-1}$$

En moyenne,  $i \simeq \frac{N}{2}$  :

$$C_N = N + 2 * C_{\frac{N}{2}}$$

Au rang suivant :

$$C_N = 2N + 4 * C_{\frac{N}{4}}$$

Puis :

$$C_N = kN + 2^k * C_{\frac{N}{2^k}}$$

La récurrence se termine après  $k = \log_2(N)$  étapes, donc :

$$C_N = N \log N + NC_1 = O(N \log N)$$

Le parcours du tableau implique  $N - 1$  comparaison. Donc :

$$C_N = (N - 1) + C_i + C_{N-i-1}$$

En moyenne,  $i \simeq \frac{N}{2}$  :

$$C_N = N + 2 * C_{\frac{N}{2}}$$

Au rang suivant :

$$C_N = 2N + 4 * C_{\frac{N}{4}}$$

Puis :

$$C_N = kN + 2^k * C_{\frac{N}{2^k}}$$

La récurrence se termine après  $k = \log_2(N)$  étapes, donc :

$$C_N = N \log N + NC_1 = O(N \log N)$$

Le parcours du tableau implique  $N - 1$  comparaison. Donc :

$$C_N = (N - 1) + C_i + C_{N-i-1}$$

En moyenne,  $i \simeq \frac{N}{2}$  :

$$C_N = N + 2 * C_{\frac{N}{2}}$$

Au rang suivant :

$$C_N = 2N + 4 * C_{\frac{N}{4}}$$

Puis :

$$C_N = kN + 2^k * C_{\frac{N}{2^k}}$$

La récurrence se termine après  $k = \log_2(N)$  étapes, donc :

$$C_N = N \log N + NC_1 = O(N \log N)$$

Le parcours du tableau implique  $N - 1$  comparaison. Donc :

$$C_N = (N - 1) + C_i + C_{N-i-1}$$

En moyenne,  $i \simeq \frac{N}{2}$  :

$$C_N = N + 2 * C_{\frac{N}{2}}$$

Au rang suivant :

$$C_N = 2N + 4 * C_{\frac{N}{4}}$$

Puis :

$$C_N = kN + 2^k * C_{\frac{N}{2^k}}$$

La récurrence se termine après  $k = \log_2(N)$  étapes, donc :

$$C_N = N \log N + NC_1 = O(N \log N)$$

Le parcours du tableau implique  $N - 1$  comparaison. Donc :

$$C_N = (N - 1) + C_i + C_{N-i-1}$$

En moyenne,  $i \simeq \frac{N}{2}$  :

$$C_N = N + 2 * C_{\frac{N}{2}}$$

Au rang suivant :

$$C_N = 2N + 4 * C_{\frac{N}{4}}$$

Puis :

$$C_N = kN + 2^k * C_{\frac{N}{2^k}}$$

La récurrence se termine après  $k = \log_2(N)$  étapes, donc :

$$C_N = N \log N + NC_1 = O(N \log N)$$

Le tri fusion est un  $O(N \log(N))$  dans tous les cas. Cependant il est en moyenne plus lent que Quicksort, c'est pourquoi ce dernier est le plus utilisé.

Rappels

Diviser pour régner

Quicksort

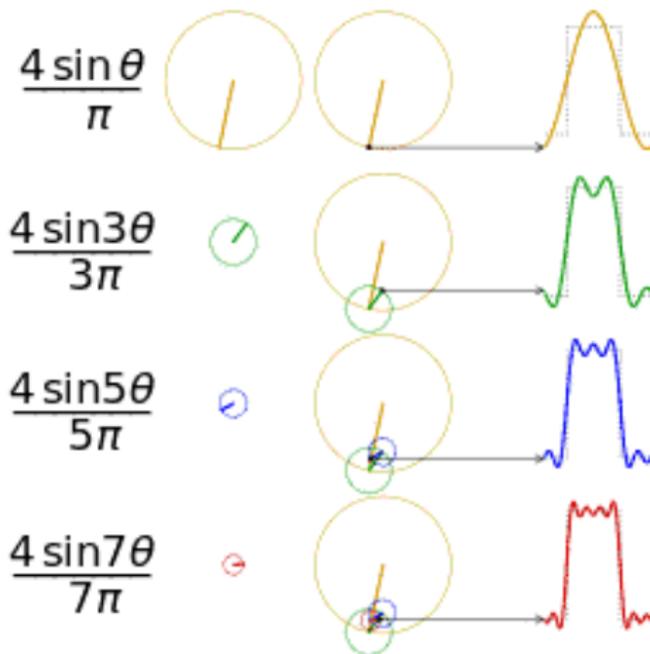
Tri fusion

Fast Fourier Transform

TP

# Discrete Fourier Transform

La Transformée de Fourier discrète est un algorithme important en traitement du signal et des images.



Soit l'espace complexe  $\mathbb{C}^N$  et la forme hermitienne :

$$\langle f, g \rangle = \sum_{j=0}^{N-1} f[j] \overline{g[j]}.$$

La famille des vecteur  $e_k$  :

$$e_k = \frac{1}{\sqrt{N}} \left( e^{\frac{2i\pi}{N} 0 \cdot k}, e^{\frac{2i\pi}{N} 1 \cdot k}, \dots, e^{\frac{2i\pi}{N} (N-1) \cdot k} \right)$$

pour  $k = 0, \dots, N - 1$  est une famille libre orthonormale (donc une base).

## Décomposition dans la base des exponentielles complexes

Soit  $f$  un tableau de  $N$  nombres complexes. Comme  $(e_0, e_1, \dots, e_{N-1})$  est une base :

$$f = \sum_{j=0}^{N-1} \langle f, e_j \rangle e_j$$

## Définition

Les valeurs de la Transformée de Fourier discrète de  $f$  sont les coordonnées de  $f$  dans cette nouvelle base.

## Expression

La transformée de Fourier discrète transforme un tableau  $f$  de  $N$  nombres complexes en un tableau  $DFT(f)$  de même taille par l'opération suivante :

$$DFT(f)[k] = \langle f, e_k \rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} f[j] e^{-\frac{2i\pi}{N}jk}.$$

## Interprétation physique

La DFT transforme un signal complexe (resp. réel)  $f$  en une somme d'exponentielles complexes (resp. de sinus). Le coefficient  $DFT(f)[k]$  représente l'énergie du signal  $f$  à la fréquence  $k$ .

Soit  $f$  un tableau de  $N$  nombres complexes. Comme  $(e_0, e_1, \dots, e_{N-1})$  est une base :

$$f = \sum_{j=0}^{N-1} \langle f, e_j \rangle e_j$$

ou encore :

$$f = \sum_{j=0}^{N-1} \text{DFT}(f)[j] e_j$$

Soit  $f$  un tableau de  $N$  nombres complexes. Comme  $(e_0, e_1, \dots, e_{N-1})$  est une base :

$$f = \sum_{j=0}^{N-1} \langle f, e_j \rangle e_j$$

ou encore :

$$f = \sum_{j=0}^{N-1} \text{DFT}(f)[j] e_j$$

$$f = \sum_{j=0}^{N-1} DFT(f)[j] e_j$$

$$f[k] = \left( \sum_{j=0}^{N-1} DFT(f)[j] e_j \right) [k]$$

$$f[k] = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} DFT(f)[j] e^{+ \frac{2i\pi}{N} jk}$$

Notant  $IDFT$  la transformée inverse ( $IDFT \circ DFT = Id$ ) :

$$f[k] = IDFT(DFT(f))[k] = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} DFT(f)[j] e^{+ \frac{2i\pi}{N} jk}$$

Pour résumer,  $f$  un tableau de  $N$  nombres complexes :

### Discrete Fourier Transform

$$DFT(f)[k] = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} f[j] e^{-\frac{2i\pi}{N}jk}.$$

### Inverse Discrete Fourier Transform

$$IDFT(g)[k] = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} g[j] e^{+\frac{2i\pi}{N}jk}.$$

## Motivation

La transformée de Fourier transforme les convolutions (= opérations de filtrage) en multiplication. Il est bien plus rapide de faire une multiplication dans l'espace de Fourier qu'une convolution dans l'espace initial.

Input image



Convolution  
Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Feature map



## Motivation

La transformée de Fourier transforme les convolutions (= opérations de filtrage) en multiplication. Il est bien plus rapide de faire une multiplication dans l'espace de Fourier qu'une convolution dans l'espace initial.



$$DFT(f)[k] = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} f[j] e^{-\frac{2i\pi}{N}jk}.$$

Calcul d'un terme de la DFT

Calculer un terme :  $O(N)$ .

Calcul de tous les termes de la DFT

Calculer tous les termes :  $O(N^2)$

$$DFT(f)[k] = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} f[j] e^{-\frac{2i\pi}{N}jk}.$$

## Calcul d'un terme de la DFT

Calculer un terme :  $O(N)$ .

## Calcul de tous les termes de la DFT

Calculer tous les termes :  $O(N^2)$

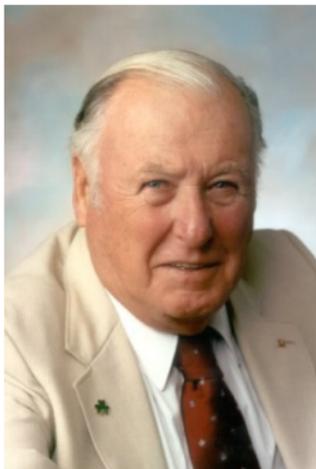
$$DFT(f)[k] = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} f[j] e^{-\frac{2i\pi}{N}jk}.$$

## Calcul d'un terme de la DFT

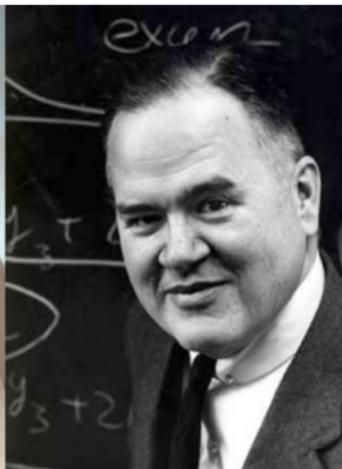
Calculer un terme :  $O(N)$ .

## Calcul de tous les termes de la DFT

Calculer tous les termes :  $O(N^2)$



James William Cooley  
(1926-)



John Wilder Tukey  
(1915-2000)

La FFT est un algorithme introduit par Cooley and Tukey en 1965. Elle permet de calculer la DFT en temps  $N \log(N)$ .

Elle utilise une approche **diviser pour régner**.

On sépare la somme dans la DFT en indices **pairs** et **impairs** :

$$\sqrt{N} \text{DFT}(f)[k] = \sum_{j=0}^{N/2-1} f[2j] e^{-\frac{2i\pi}{N}(2j)k} + \sum_{j=0}^{N/2-1} f[2j+1] e^{-\frac{2i\pi}{N}(2j+1)k},$$

dont on déduit facilement

$$\sqrt{N} \text{DFT}(f)[k] = \sum_{j=0}^{N/2-1} f[2j] e^{-\frac{2i\pi}{N/2}jk} + e^{-\frac{2i\pi}{N}k} \sum_{j=0}^{N/2-1} f[2j+1] e^{-\frac{2i\pi}{N/2}jk}.$$

On retrouve en fait le calcul de la transformée de Fourier discrète sur les deux sous-tableaux :

$$\sqrt{N}DFT(f)[k] = \sqrt{\frac{N}{2}} \left( DFT(f_{pair})[k] + e^{-\frac{2i\pi}{N}k} DFT(f_{impair})[k] \right) \quad (1)$$

où  $f_{pair}$  est le sous-tableau des indices pairs de  $f$  et  $f_{impair}$  est celui des indices impairs.

Le problème est alors de :

1. Calculer la DFT des indices pairs de  $f$ .
2. Calculer la DFT des indices impairs de  $f$ .
3. Combiner en  $O(N)$  les deux suivant la formule ci-dessus.

Comme pour le tri fusion :

- ▶ calcul sur les tableau de taille  $N/2$
- ▶ relation de récurrence  $C(N) \approx N + 2 * C(\frac{N}{2})$
- ▶ complexité en  $O(N \log(N))$

$$\sqrt{N} DFT(f)[k] = \sqrt{\frac{N}{2}} \left( DFT(f_{0:2:N-2})[k] + e^{-\frac{2i\pi}{N}k} DFT(f_{1:2:N-1})[k] \right)$$

Relation de récurrence :

1. Calculer la DFT des indices pairs de  $f$ .
2. Calculer la DFT des indices impairs de  $f$ .
3. Combiner en  $O(N)$  les deux suivant la formule ci-dessus.

$$\sqrt{N} DFT(f)[k] = \sqrt{\frac{N}{2}} \left( DFT(f_{0:2:N-2})[k] + e^{-\frac{2i\pi}{N}k} DFT(f_{1:2:N-1})[k] \right)$$

Les sous DFT sont de longueur  $N/2$ , donc définies pour  $0 \leq k \leq N/2$ .

On utilise :

$$\exp\left(-\frac{2i\pi}{N/2}j(k + N/2)\right) = \exp\left(-\frac{2i\pi}{N/2}jk\right)$$

$$\exp\left(-\frac{2i\pi}{N}(k + N/2)\right) = \exp\left(-\frac{2i\pi}{N}k\right)$$

c'est-à-dire qu'il y a périodicité de la DFT :

$$DFT_{\text{paire}}\left[k + \frac{N}{2}\right] = DFT_{\text{paire}}[k],$$

$$DFT_{\text{impaire}}\left[k + \frac{N}{2}\right] = DFT_{\text{impaire}}[k] .$$

$$\sqrt{N} \text{DFT}(f)[k] = \sqrt{\frac{N}{2}} \left( \text{DFT}(f_{0:2:N-2})[k] + e^{-\frac{2i\pi}{N}k} \text{DFT}(f_{1:2:N-1})[k] \right)$$

1. Calculer la DFT des indices pairs de  $f$ .
2. Calculer la DFT des indices impairs de  $f$ .
3. Combiner en  $O(N)$  les deux suivant la formule ci-dessus.
  - ▶ Copier  $f$  dans un tableau temporaire **buffer**. On a dans les indices pairs et impairs les résultats des sous-DFT (non normalisées).
  - ▶ Boucle de  $k = 0$  à  $N/2 - 1$  :

$$f[k] \leftarrow \text{buffer}[2 * k] + e^{-\frac{2i\pi}{N}k} \text{buffer}[2 * k + 1]$$

$$f[k + N/2] \leftarrow \text{buffer}[2 * k] - e^{-\frac{2i\pi}{N}k} \text{buffer}[2 * k + 1].$$

- ▶ Le facteur  $t_k = e^{-\frac{2i\pi}{N}k}$  est appelé *twiddle*. On en fait un calcul rapide par la relation de récurrence de suite géométrique  $t_{k+1} = r t_k$ , dont la raison  $r = e^{-\frac{2i\pi}{N}}$  est pré-calculée.
- ▶ On alloue une seule fois **buffer** et on le passe dans les arguments de la fonction récursive.

Il est possible de définir un équivalent de dérivation pour la DFT.

Soit  $e_j(x) = \frac{1}{\sqrt{N}} \exp(\frac{2i\pi}{N} jx)$ . Pour  $k \in \mathbb{N}$  :  $e_j(k) = e_j[k]$

Soit  $f$  la fonction périodique :

$$f(x) = \sum_j DFT(f)[j] e_j(x),$$

et  $f'(k) = f'[k]$  :

$$f'(x) = \sum_j DFT(f)[j] e'_j(x) = \sum_j \frac{2i\pi j}{N} DFT(f)[j] e_j(x).$$

Puis :

$$DFT(f')[j] = \frac{2i\pi j}{N} DFT(f)[j].$$

Pour les fonctions réelles : il faut que  $f'$  soit réelle.

$$DFT(f') [j] = \begin{cases} \frac{2i\pi}{N} j DFT(f) [j] & \text{pour } 0 \leq j < N/2 \\ 0 & \text{pour } j = N/2 \\ \frac{2i\pi}{N} (j - N) DFT(f) [j] & \text{pour } N/2 < j < N \end{cases}$$

Cette relation est utile pour résoudre certaines équations aux dérivées partielles. En particulier, l'équation de Poisson que nous allons résoudre en TP.

Rappels

Diviser pour régner

Quicksort

Tri fusion

Fast Fourier Transform

TP

TP long : transformée de Fourier rapide et éditeur de Poisson.

## Idée

On veut copier/coller un morceau d'image dans une autre. Pour que les transitions soient naturelles, il faut que les variations d'intensité au niveau de la frontière soient égales : c'est l'équation de Poisson.

On la résout dans l'espace de Fourier, puis on récupère l'image lissée par transformée de Fourier inverse.