# Kompleksitas Algoritma Transformasi Fourier Cepat

Daniel Prihartoni - 13509088
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*13509088@std.stei.itb.ac.id*

***Abstrak* — Transformasi Fourier Cepat, dalam bahasa Inggris dikenal dengan *Fast Fourier Transform,* adalah suatu algoritma yang banyak digunakan untuk menghitung Transformasi Fourier Diskrit atau *Discrete Fourier Transform* dalam bahasa Inggris. Adapun kelebihan dari Transformasi Fourier Cepat adalah tingkat kompleksitas yang lebih rendah sehingga lebih cepat dan mangkus. Nilai tambah ini sangat berguna karena Transformasi Fourier Diskrit banyak diaplikasikan pada berbagai bidang khususnya pengolahan sinyal.**

***Kata Kunci* — Transformasi Fourier Cepat, Transformasi Fourier Diskrit, kompleksitas, pengolahan sinyal.**

## 1. PENDAHULUAN

Transformasi Fourier Diskrit merupakan bagian dari Transformasi Fourier yang digunakan dalam analisis Fourier. Dalam analisis Fourier dipelajari bagaimana cara merepresentasikan sevuah fungsi dengan penjumlahan beberapa fungsi trionometri yang lebih sederhana. Analisis Fourier dinamakan sesuai dengan penemunya yaitu Joseph Fourier (1768-1830), seorang matematikawan dan fisikawan berkebangsaan Prancis. Beliau menunjukkan bahwa dengan mengubah sebuah fungsi menjadi deret trigonometri akan mempermudah pembelajaran tentang propagasi panas.

Dalam bidang matematika, Transformasi Fourier digunakan untuk menguraikan sebuah sinyal menjadi frekuensinya. Dengan kata lain, Transformasi Fourier mengubah suatu fungsi ke daam bentuk lain. Pada Transformasi Fourier Diskrit, masukan fungsi harus dalam bentuk diskrit. Masukan Transformasi Fourier Diskrit adalah urutan terbatas bilangan riil ataupun bilangan kompleks. Hal ini menyebabkan Transformasi Fourier Diskrit ideal untuk memproses informasi di dalam komputer.

Dalam perkembangannya, para peneliti terus berupaya mengembangkan suatu algoritma yang lebih cepat dan mangkus. Pada tahun 1965, J. W. Cooley dan John Tukey mengenalkan sebuah metode baru yang lebih cepat dan mangkus dalam memproses Transformasi Fourier Diskrit. Algoritma yang diberi nama algoritma Cooley-Tukey ini sebenarnya telah ditemukan oleh Carl Friedrich Gauss pada sekitar tahun 1805. Karena lebih cepat dalam proses perhitungan, metode ini disebut dengan nama Transformasi Fourier Cepat.

Dalam penggunaan sehari-hari, istilah Transformasi Fourier Cepat dan Transformasi Fourier Diskrit sering dipertukarkan. Padahal keduanya memiliki perbedaan yang jelas, yakni Transformasi Fourier Diskrit merujuk pada transformasi matematik bebas sedangkan Transformasi Fourier Cepat merujuk pada algoritma mangkus yang digunakan untuk menghitung Transformasi Fourier Diskrit. Hal demikian terjadi karena pada umumnya Transformasi Fourier Cepat digunakan untuk menghitung Transformasi Fourier Diskrit.

Selain algoritma Cooley-Tukey, ada beberapa algoritma lain yang juga ditujukan untuk menghitung Transformasi Fourier Diskrit dengan lebih cepat dan mangkus. Beberapa diantaranya yaitu algoritma faktor prima, algoritma Radix terpisah, algoritma Bruun, algoritma Rader, dan algoritma Bluestein. Sampai dengan waktu pada saat makalah ini dibuat, algoritma Cooley-Tukey merupakan bentuk Transformasi Fourier Cepat yang paling populer digunakan.

Penemuan Transformasi Fourier Cepat membawa dampak besar pada sejarah ilmu pengetahuan. Transformasi dapat menghitung deret Fourier dengan kecepatan dibandingkan dengan Transformasi Fourier Diskrit biasa. Untuk data dengan jumlah ribuan atau bahkan jutaan, penggunaan Transformasi Fourier Cepat dapat mengurangi waktu komputasi hingga beberapa kali lipat. Peningkatan besar ini berperan penting pada berbagai macam aplikasi, contohnya pemrosesan sinyal digital, penyelesaian persamaan diferensial, dan perkalian bilangan bulat besar.

## 2. TRANSFORMASI FOURIER DISKRIT

Menurut Buku "Understanding Digital Signal Processing, Second Edition" karangan Richard G. Lyons. Transformasi Fourier Diskrit adalah prosedur yang kuat yang digunakan dalam pemrosesan sinyal digital dan filterisasi digital. Transformasi Fourier Diskrit menungkinkan seseorang untuk menganalisa, memanipulasi dan mensintesis sinyal yang tidak mungkin dapat dilakukan dalam pemrosesan sinyal analog. Sedangkan menurut buku "Handbook of Digital Signal Processing Engineering Applications", Transformasi Fourier Diskrit merupakan gambaran karakteristik spektrum periodik dari suatu sampel data. Transformasi Fourier Diskrit memiliki spectrum garis yang mewakili periode sekuensial N. Adanya istilah "*discrete fourier transform*" karena Transformasi Fourier Diskrit memberikan gambaran deret fourier untuk sekuens terbatas.

Secara umum Transformasi Fourier Diskrit adalah salah satu macam transformasi Fourier yang digunakan dalam analisis Fourier. Dengan Transformasi Fourier Diskrit, suatu fungsi diubah ke dalam domain frekuensi. Adapun rumus untuk menghitung Transformasi Fourier Diskrit sebagai berikut : Diketahui barisan N buah bilangan kompleks $h_0, \ldots, h_{k-1}$ diubah menjadi N buah barisan bilangan kompleks $H_0, \ldots, H_{n-1}$ dengan rumus :

$$H_n \equiv \sum_{k=0}^{N-1} h_k \, e^{2\pi i k n / N} \tag{1}$$

dengan k = 0, 1, … , N-1. Persamaan (1) di atas mengubah $h_k$ menjadi $H_n$.

Untuk menghitung kompleksitas algoritma dari perhitungan Transformasi Fourier Diskrit, dimisalkan W sebagai suatu bilangan kompleks seperti di bawah ini.

$$W \equiv e^{2\pi i / N} \tag{2}$$

Dengan melakukan substitusi (2) ke dalam (1), maka dihasilkan persamaan di bawah ini.

$$H_n = \sum_{k=0}^{N-1} W^{nk} h_k \tag{3}$$

Dengan kata lain, vektor $h_k$ dikalikan dengan matriks yang elemen ke (n,k)-nya adalah konstanta W untuk daya n x k. Perkalian matriks ini menghasilkan vektor yang tersusun dari komponen $H_n$. Oleh karena itu, perkalian matriks ini menggunakan N x N = $N^2$ perkalian kompleks ditambah dengan suatu nilai yang lebih kecil untuk menghasilkan W. Dapat disimpulkan bahwa Transformasi Fourier Diskrit memiliki kompleksitas waktu asimptotik $N^2$ atau $O(N^2)$.

Di bawah ini contoh algoritma Transformasi Fourier Diskrit ditulis dalam bahasa C.

```
/* Direct fourier transform */

int DFT(int dir, int m, double *x1,
double *y1) {

long i,k;
double arg;
double cosarg,sinarg;
double *x2=NULL,*y2=NULL;

x2 = malloc(m*sizeof(double));
y2 = malloc(m*sizeof(double));

if (x2 == NULL || y2 == NULL)
    return(FALSE);

for (i=0;i<m;i++) {
    x2[i] = 0;
    y2[i] = 0;
    arg = - dir * 2.0 * 3.141592654 *
(double)i / (double)m;

    for (k=0;k<m;k++) {
        cosarg = cos(k * arg);
        sinarg = sin(k * arg);
        x2[i] += (x1[k] * cosarg -
y1[k] * sinarg);
        y2[i] += (x1[k] * sinarg +
y1[k] * cosarg);
    }
}

/* Copy the data back */

if (dir == 1) {
    for (i=0;i<m;i++) {
    x1[i] = x2[i] / (double)m;
    y1[i] = y2[i] / (double)m;
    }
} else {
    for (i=0;i<m;i++) {
    x1[i] = x2[i];
    y1[i] = y2[i];
    }
}

free(x2);
free(y2);
return(TRUE);
}
```

## 3. TRANSFORMASI FOURIER CEPAT

Ada beberapa macam algoritma yang tergolong ke dalam Transformasi Fourier Cepat. Namun dalam makalah ini hanya akan membahas tiga algoritma saja.

### 3.1 Algoritma Cooley-Tukey

Algoritma Cooley Tukey adalah salah satu dari sekian algoritma yang digunakan untuk menghitung Transformasi Fourier Diskrit. Algoritma ini dipopulerkan oleh J. W. Cooley dan John Tukey pada tahun 1965. Sebenarnya algoritma ini pertama kali ditemukan oleh Carl Friedrich Gauss pada tahun 1805. Hanya saja ia tidak sampai pada pembahasan waktu perhitungan asimptotik. Sampai dengan waktu saat makalah ini dibuat, algoritma ini merupakan algoritma yang paling umum digunakan.

Ide dari Transformasi Fourier Cepat adalah mengubah suatu bentuk Transformasi Fourier Diskrit dengan panjang N menjadi bentuk penjumlahan dari dua buah bentuk Transformasi Fourier Diskrit dengan panjang masing-masing N/2, satu bagian terdiri dari elemen bernomor ganjil sementara yang lain genap. Pembuktian dari algoritma ini adalah sebagai berikut :

$$
\begin{aligned}
F_k &= \sum_{j=0}^{N-1} e^{2\pi ijk/N} f_j \\
&= \sum_{j=0}^{N/2-1} e^{2\pi ik(2j)/N} f_{2j} + \sum_{j=0}^{N/2-1} e^{2\pi ik(2j+1)/N} f_{2j+1} \\
&= \sum_{j=0}^{N/2-1} e^{2\pi ikj/(N/2)} f_{2j} + W^k \sum_{j=0}^{N/2-1} e^{2\pi ikj/(N/2)} f_{2j+1} \\
&= F_k^e + W^k F_k^o
\end{aligned}
\qquad (4)
$$

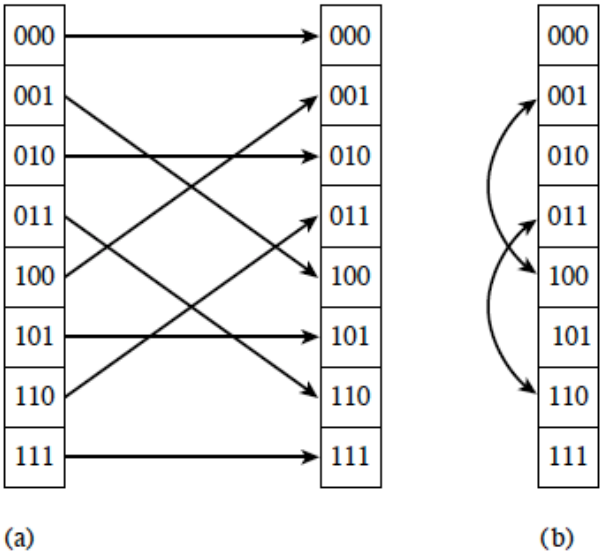
Tanda e dan o pada F menyatakan genap (even) dan ganjil (odd) sedangkan W didefinisikan seperti pada (2). Keistimewaan dari pembuktian di atas adalah dapat dilakukan secara rekursif hingga akhirnya panjang dari bentuk transformasi Fourier bernilai satu. Transformasi Fourier dengan panjang satu adalah identitas operasi yang menyalin masukan ke keluaran. Pada suatu nilai n, bentuk penjabaran seperti pada (4) dapat dinyatakan dengan :

$$F_k^{eoeeoeo\cdots oee} = f_n$$

, dengan n tertentu (5)

Langkah selanjutnya adalah menghubungkan nlai dari n dengan pola ganjil genap pada (5). Sebelum menghubung-hubungkan, nilai genap dan ganjil diubah terlebih dahulu ke dalam bentuk biner (0 dan 1). Proses menghubung-hubungkan ini menghasilkan nilai pasangan n dalam bentuk biner.

Pengulangan yang terjadi untuk setiap pola menghabiskan waktu sebanyak $\log_2 N$. Karena setiap kombinasi terdiri dari N operasi, maka keseluruhan algoritma berorde $N \log_2 N$ dengan asumsi bahwa proses menghubungkan suatu nilai dengan bentuk bit tidak lebih dari $N \log_2 N$.




Gambar 1 - Proses pengurutan bit pada sebuah array, (a) pada dua array, (b) pada satu array.

Perbedaan antara orde metode Transformasi Fourier Diskrit dan Transformasi Fourier Cepat, yaitu $N^2$ dan $N \log_2 N$, sangat mempengaruhi waktu yang dibutuhkan untuk menghitung deret Fourier. Dengan adanya Transformasi Fourier Cepat akan sangat membantu penghitungan deret Fourier dalam dua hal yaitu : jumlah data yang besar dan keterbatasan alat penghitung (komputer).

Tabel 1 – Perbandingan kecepatan operasi dengan menggunakan Transformasi Fourier Cepat (kolom “FFT” pada tabel) dan langsung dengan Transformasi Fourier Diskrit (kolom ”direct” pada tabel)

| | Number of flops | | Increase in |
|---|---|---|---|
| $K$ | FFT ($5K\log_2 K$) | direct ($8K^2$) | Speed |
| 32 | 800 | 8192 | 10.2 |
| 256 | 10 240 | 524 288 | 51.2 |
| 1024 | 51 200 | 8 388 608 | 163.8 |
| 8192 | 532 480 | 536 870 912 | 1 008.2 |

Di bawah ini contoh algoritma Cooley-Tukey ditulis dalam bahasa C

```
/* This computes an in-place complex-
to-complex FFT x and y are the real
and imaginary arrays of 2^m points.
dir =  1 gives forward transform
dir = -1 gives reverse transform  */

short FFT(short int dir,long m,double
*x,double *y) {

long n,i,i1,j,k,i2,l,l1,l2;
double c1,c2,tx,ty,t1,t2,u1,u2,z;

/* Calculate the number of points */
n = 1;

for (i=0;i<m;i++)
   n *= 2;

/* Do the bit reversal */

i2 = n >> 1;
j = 0;
for (i=0;i<n-1;i++) {
   if (i < j) {
      tx = x[i];
      ty = y[i];
      x[i] = x[j];
      y[i] = y[j];
      x[j] = tx;
      y[j] = ty;
   }

   k = i2;
   while (k <= j) {
      j -= k;
      k >>= 1;
   }

   j += k;
}

/* Compute the FFT */
c1 = -1.0;
c2 = 0.0;
l2 = 1;

for (l=0;l<m;l++) {
   l1 = l2;
   l2 <<= 1;
   u1 = 1.0;
   u2 = 0.0;

   for (j=0;j<l1;j++) {
      for (i=j;i<n;i+=l2) {
         i1 = i + l1;
         t1 = u1 * x[i1] - u2 * y[i1];
         t2 = u1 * y[i1] + u2 * x[i1];
         x[i1] = x[i] - t1;
         y[i1] = y[i] - t2;
         x[i] += t1;
         y[i] += t2;
      }

      z =  u1 * c1 - u2 * c2;

      u2 = u1 * c2 + u2 * c1;
      u1 = z;
   }

   c2 = sqrt((1.0 - c1) / 2.0);

   if (dir == 1)
      c2 = -c2;

c1 = sqrt((1.0 + c1) / 2.0);

}

/* Scaling for forward transform */

if (dir == 1) {
   for (i=0;i<n;i++) {
      x[i] /= n;
      y[i] /= n;
   }
}

return(TRUE);
}
```

3.2 Algoritma Rader

Algoritma Rader, ditemukan pada tahun 1968, adalah salah satu jenis Transformasi Fourier Cepat untuk menghitung transformasi Fourier Diskrit berukuran bilangan prima dengan mengekspresikan kembali Transformasi Fourier Diskrit sebagai sebuah siklus konvolusi. Untuk suatu kasus yang sama, algoritma Cooley-Tukey jauh lebih sederhana dan lebih praktis dalam memecahkan suatu masalah. Oleh karena itu, algoritma Rader biasanya hanya digunakan untuk kasus dasar dari dekomposisi rekursif Cooley-Tukey dari Transformasi Fourier Diskrit.

Jika Transformasi Fourier didefinisikan sebagai :

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} nk} \qquad k = 0, \ldots, N-1. \tag{6}$$

Jika N adalah bilangan prima, maka suatu set indeks tidak nol n = 1, … , N-1 membentuk kelompok terhadap perkalian modulo N. Berdasarkan teori bilangan, kelompok bilangan tersebut mempunyai sebiah pembangkit, yaitu bilangan bulat g sedemikian sehingga $n = g^q \pmod N$ untuk setiap indeks tidak nol dan setiap q yang unik paa 0, …, N-2. Maka $k = g^{-p} \pmod N$ untk setiap indeks tidak nol k dan setiap p yang unik pada 0, … , N-2. Persamaan (6) dapat ditulis ulang menjadi :

$$X_{g^{-p}} = x_0 + \sum_{q=0}^{N-2} x_{g^q} e^{-\frac{2\pi i}{N} g^{-(p-q)}} \tag{7}$$

, dengan p = 0, … , N-2 dan nilai $X_0$ :

$$X_0 = \sum_{n=0}^{N-1} x_n, \tag{8}$$

Penjumlahan akhir dari (7) adalah sebuah siklus konvolusi dari dua bagiab $a_q$ dan $b_q$ dengan panjang N-1 (q=0, …, N-2) dengan definisi :

$$a_q = x_{g^q} \tag{9}$$

$$b_q = e^{-\frac{2\pi i}{N} g^{-q}}. \tag{10}$$

Algoritma Rader membutuhkan waktu O(N) ditambah O(N log N) untuk konvolusi.

### 3.3 Algoritma Bluestein

Algoritma Bluestein, ditemukan pada tahun 1968, adalah salah satu bentuk Transformasi Fourier Cepat untuk menghitung Transformasi Fourier Diskrit dengan ukuran bebas dengan cara mengubah bentuk Transformasi Fourier Diskrit ke dalam bentuk konvolusi.

Jika produk nk pada (6) diganti dengan identitas $nk = -(k-n)^2/2 + n^2/2 + k^2/2$, didapatkan :

$$X_k = e^{-\frac{\pi i}{N} k^2} \sum_{n=0}^{N-1} \left( x_n e^{-\frac{\pi i}{N} n^2} \right) e^{\frac{\pi i}{N} (k-n)^2} \tag{11}$$

,dengan k = 0, … , N-1.

Penjumlahan pada (11) merupakan konvolusi dari dua urutan $a_n$ dan $b_n$ dengan panjang N (n = 0, … , N-1) dengan definisi :

$$a_n = x_n e^{-\frac{\pi i}{N} n^2} \tag{12}$$

$$b_n = e^{\frac{\pi i}{N} n^2}, \tag{13}$$

Keluaran dari konvolusi dikalikan dengan faktor fase N $b_k^*$ menjadi :

$$X_k = b_k^* \sum_{n=0}^{N-1} a_n b_{k-n} \tag{14}$$

Algoritma Bluestein memiliki O(N log N) untuk menghitung Transformasi Fourier Diskrit dengan ukuran prima. Namun, untuk Transformasi Fourier Diskrit dengan ukuran komposit, algoritma ini masih beberapa kali lebih lambat dari algoritma Cooley-Tukey.

## 4. KESIMPULAN

Dari sekian banyak algoritma yang digunakan dalam Transformasi Fourier Cepat, bak yang dibahas dalam makalah ini ataupun tidak, algoritma Cooley-Tukey adalah algoritma yang paling cepat dan mangkus. Algoritma ini memiliki O(N log N) yang sangat bermanfaat untuk perhitungan dengan jumlah data besar. Selain itu, algoritma Cooley-Tukey berlaku untuk semua jenis jumlah masukan (tidak terbatas pada suatu jenis saja).

## REFERENSI


[1] Douglas F. Elliott, 1987, "*Handbook of Digital Signal Processing, Engineering Applications*", Academic Press Inc.

[2] Munir, Rinaldi. Diktat Kuliah IF2091 Struktur Diskrit. Edisi keempat. Program Studi Teknik Informatika, Institut Teknologi Bandung

[3] Richards G. Lyons, 2004, "*Understanding Digital Signal Processing*", Prentice-Hall.

[4] http://www.cambridge.org/resources/0521854555/4421_Chapter%2012%20-%20Discrete%20Fourier%20transform.pdf
Waktu akses : 15 Desember 2010 pukul 10.00 WIB

[5] http://www.mathcs.org/java/programs/FFT/FFTInfo/c12-2.pdf
Waktu akses : 16 Desember 2010 pukul 15.15 WIB

[6] http://local.wasp.uwa.edu.au/~pbourke/miscellaneous/dft/
Waktu akses : 16 Desember 2010 pukul 15.00 WIB

[7] http://en.wikipedia.org/wiki/Bluestein%27s_FFT_algorithm
Waktu akses : 16 Desember 2010 pukul 20.30 WIB

[8] http://en.wikipedia.org/wiki/Cooley–Tukey_FFT_algorithm
Waktu akses : 16 Desember 2010 pukul 20.00 WIB

[9] http://en.wikipedia.org/wiki/Discrete_Fourier_transform
Waktu akses : 15 Desember 2010 pukul 18.00 WIB

[10] http://en.wikipedia.org/wiki/Fast_Fourier_transform
Waktu akses : 15 Desember 2010 pukul 18.15 WIB

[11] http://en.wikipedia.org/wiki/Rader%27s_FFT_algorithm
Waktu akses : 16 Desember 2010 pukul 20.15 WIB

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 17 Desember 2010

ttd

Daniel Prihartoni
13509088