TUGAS AKHIR – KI141502

DESAIN DAN ANALISIS ALGORITMA PENYELESAIAN PERSOALAN SPOJ MAXIMUM EDGE OF POWERS OF PERMUTATION DENGAN METODE PERMUTATION CYCLES FINDING DAN FFT CONVOLUTION

KARSTEN ARI AGATHON
NRP. 5112 100 110

Dosen Pembimbing 1
Victor Hariadi, S.Si., M.Kom.

Dosen Pembimbing 2
Rully Soelaiman, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

DESAIN DAN ANALISIS ALGORITMA PENYELESAIAN PERSOALAN SPOJ MAXIMUM EDGE OF POWERS OF PERMUTATION DENGAN METODE PERMUTATION CYCLES FINDING DAN FFT CONVOLUTION

KARSTEN ARI AGATHON
NRP. 5112 100 110

Dosen Pembimbing 1
Victor Hariadi, S.Si., M.Kom.

Dosen Pembimbing 2
Rully Soelaiman, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

(Halaman ini sengaja dikosongkan)

FINAL PROJECT – KI141502

DESIGN AND ANALYSIS OF ALGORITHM FOR SOLVING SPOJ MAXIMUM EDGE OF POWERS OF PERMUTATION USING PERMUTATION CYCLES FINDING AND FFT CONVOLUTION

KARSTEN ARI AGATHON
NRP. 5112 100 115

Supervisor 1
Victor Hariadi, S.Si., M.Kom.

Supervisor 2
Rully Soelaiman, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2017

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

DESAIN DAN ANALISIS ALGORITMA PENYELESAIAN PERSOALAN SPOJ MAXIMUM EDGE OF POWERS OF PERMUTATION DENGAN METODE PERMUTATION CYCLES FINDING DAN FFT CONVOLUTION

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Dasar dan Terapan Komputasi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
KARSTEN ARI AGATHON
NRP. 5112100110

Disetujui oleh Pembimbing Tugas Akhir:

1. Victor Hariadi, S.Si., M.Kom. .....
   NIP. 196912281994121001 (Pembimbing 1)

2. Rully Soelaiman, S.Kom., M.Kom. .....
   NIP. 197002131994021001 (Pembimbing 2)

SURABAYA
JANUARI, 2017

(Halaman ini sengaja dikosongkan)

DESAIN DAN ANALISIS ALGORITMA PENYELESAIAN PERSOALAN SPOJ MAXIMUM EDGE OF POWERS OF PERMUTATION DENGAN METODE PERMUTATION CYCLES FINDING DAN FFT CONVOLUTION

Nama : **Karsten Ari Agathon**
NRP : **5112100110**
Jurusan : **Teknik Informatika**
Fakultas Teknologi Informasi ITS
Dosen Pembimbing I : **Victor Hariadi, S.Si., M.Kom.**
Dosen Pembimbing II : **Rully Soelaiman, S.Kom., M.Kom.**

ABSTRAK

Permasalahan dalam buku tugas akhir ini adalah permasalahan "Maximum Edge of Powers of Permutation" yang terdapat pada situs penilaian daring Sphere Online Judge(SPOJ) dengan nomor soal 6895 dan kode soal MEPPERM. Dalam permasalahan ini, diberikan sejumlah simpul yang masing-masing memiliki bobot keluar dan masuk. Diberikan suatu permutasi terhadap sejumlah simpul tersebut. Suatu graf dibentuk dari setiap simpul yang terhubung ke simpul hasil permutasi dengan bobot jumlah bobot keluar simpul asal dan bobot masuk simpul tujuan. Dicari sisi dengan bobot terbesar pada setiap graf yang dibentuk dari hasil pangkat permutasi.

Tugas akhir ini akan mengimplementasikan metode pencarian permutasi siklus dan konvolusi menggunakan transformasi Fourier cepat(FFT) dalam menyelesaikan permasalahan Maximum Edge of Powers of Permutation. Metode transformasi Fourier cepat yang digunakan adalah algoritma Cooley-Tukey.

Implementasi dalam tugas akhir ini menggunakan bahasa pemrograman C++. Hasil uji coba menunjukkan bahwa sistem

menghasilkan bobot sisi maksimum pada setiap pangkat permutasi dengan benar dan memiliki pertumbuhan waktu dengan kompleksitas $O(NMlogNM + Q\sqrt{N})$.
Kata kunci: Permutasi Siklus, Pangkat Permutasi, bobot maksimum, FFT, Konvolusi.

DESIGN AND ANALYSIS OF ALGORITHM FOR SOLVING SPOJ MAXIMUM EDGE OF POWERS OF PERMUTATION USING PERMUTATION CYCLES FINDING AND FFT CONVOLUTION

Name	**: Karsten Ari Agathon**
NRP	**: 5112100110**
Department	**: Department of Informatics** **Faculty of Information Technology ITS**
Supervisor I	**: Victor Hariadi, S.Si., M.Kom.**
Supervisor II	**: Rully Soelaiman, S.Kom., M.Kom.**

ABSTRACT

This final project is based on "Maximum Edge of Powers of Permutation" problem on SPOJ with problem number 6895 and problem code MEPPERM. In this problem, given vertices which each have weight in and out. Given a permutation for those vertices. A graph can be built by connecting each vertices to its permutation with edge weight sum of vertex weight out and vertex weight in from its permutation. Determine edge with maximum weight from each graph which are created from the powers of permutation.

This final project implements permutation cycles finding and convolution using fast Fourier Transform(FFT) to solve the problem of Maximum Edge of Powers of Permutation. Fast Fourier transform method used is Cooley-Tukey algorithm.

The implementation of final project uses C++ programming language. The experiment result proved the system provide edge with maximum weight for each powers of permutation correctly and has growth time with complexity of $O(NMlogNM + Q\sqrt{N})$.

Keywords: Permutation Cycles, Powers of Permutation, Maximum Weight, FFT, Convolution.

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul:

"DESAIN DAN ANALISIS ALGORITMA PENYELESAIAN PERSOALAN SPOJ MAXIMUM EDGE OF POWERS OF PERMUTATION DENGAN METODE PERMUTATION CYCLES FINDING DAN FFT CONVOLUTION"

Tugas akhir ini dilakukan untuk memenuhi salah satu syarat memperoleh gelar Sarjana Komputer di Jurusan Teknik Informatika Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember.

Penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan baik secara langsung maupun tidak langsung selama proses pengerjaan tugas akhir ini hingga selesai, antara lain:

1. Tuhan Yang Maha Esa atas segala karunia dan rahmat-Nya yang telah diberikan selama ini.
2. Orang tua, saudara serta keluarga penulis yang selalu memberikan dukungan, semangat, perhatian dan doa selama perkuliahan penulis di Jurusan Teknik Informatika ini.
3. Bapak Victor Hariadi, S.Si., M.Kom. selaku dosen pembimbing I yang telah memberikan bimbingan, dukungan, motivasi, serta arahan dalam pengerjaan tugas akhir ini.
4. Bapak Rully Soelaiman, S.Kom., M.Kom. selaku dosen pembimbing II yang telah banyak memberikan ilmu, bimbingan, nasihat, motivasi, serta waktu diskusi sehingga penulis dapat menyelesaikan tugas akhir ini.

5. Seluruh pihak yang tidak bisa saya sebutkan satu persatu yang telah memberikan dukungan selama saya menyelesaikan tugas akhir ini.

Saya mohon maaf apabila terdapat kekurangan dalam penulisan buku tugas akhir ini. Kritik dan saran saya harapkan untuk perbaikan dan pembelajaran di kemudian hari. Semoga tugas akhir ini dapat memberikan manfaat yang sebaik-baiknya.

Surabaya, Januari 2017

Karsten Ari Agathon

DAFTAR ISI

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB 1
PENDAHULUAN

Pada bab ini akan dijelaskan hal-hal yang menjadi latar belakang, permasalahan yang dihadapi, batasan masalah, tujuan, metodologi dan sistematika penulisan yang digunakan dalam pembuatan buku tugas akhir ini.

1.1 Latar Belakang

Pada soal SPOJ berjudul "*Maximum Edge of Powers of Permutation*" mengangkat suatu permasalahan dimana suatu permutasi dapat membentuk suatu graf dengan sejumlah sisi yang menghubungkan setiap simpul ke simpul yang merupakan hasil permutasinya. Kemudian dari hasil graf tersebut dicari bobot sisi yang memiliki nilai terbesar. Namun pencarian bobot sisi maksimum tidak hanya dilakukan pada suatu permutasi saja melainkan juga dari sejumlah permutasi yang merupakan hasil pangkat permutasi tersebut. Penulis mengajukan solusi permasalahan ini dengan menggunakan pencarian permutasi siklus dan konvolusi menggunakan algoritma transformasi Fourier cepat(*FFT*).

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana desain algoritma efisien dalam menyelesaikan permasalahan "*Maximum Edge of Powers of Permutation*" pada situs penilaian daring SPOJ?
2. Bagaimana implementasi algoritma yang sudah didesain untuk menyelesaikan permasalahan "*Maximum Edge of Powers of Permutation*"

3. Bagaimana uji coba untuk mengetahui kebenaran dan kinerja dari implementasi yang telah dilakukan?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Permasalahan dalam tugas akhir ini adalah permasalahan "*Maximum Edge of Powers of Permutation"* pada situs penilaian daring SPOJ klasikal dengan nomor soal 6895 dan kode soal MEPPERM.
2. Implementasi dilakukan dengan bahasa pemrograman C++.
3. Batas maksimum jumlah simpul N adalah 66000 buah.
4. Batas maksimum nilai bobot A dan B pada setiap simpul adalah 16.
5. Batas maksimum nilai pangkat permutasi Q adalah 1000000.

1.4 Tujuan

Tujuan dari pengerjaan tugas akhir ini adalah sebagai berikut:

1. Melakukan desain algoritma dan struktur yang efisien dalam menyelesaikan permasalahan "*Maximum Edge of Powers of Permutation*" pada situs penilaian daring SPOJ.
2. Melakukan implementasi algoritma yang sudah didesain untuk menyelesaikan permasalahan.
3. Melakukan uji coba untuk mengetahui kebenaran dan kinerja dari implementasi yang telah dilakukan.

1.5 Metodologi

Ada beberapa tahapan dalam pengerjaan tugas akhir ini, yaitu sebagai berikut:

1. Penyusunan Proposal Tugas Akhir

Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Pada proposal ini, penulis mengajukan gagasan untuk menyelesaikan permasalahan komputasi bobot sisi maksimum pada *Maximum Edge of Powers of Permutation*.

2. Studi Literatur
   Pada tahap ini dilakukan studi literatur yang membahas lebih dalam yang berkaitan dengan permutasi, graf, konvolusi, dan algoritma transformasi Fourier cepat. Studi literatur didapatkan dari buku, internet, dan materi-materi kuliah yang berhubungan dengan metode yang akan digunakan.
3. Desain
   Pada tahap ini dilakukan desain algoritma serta struktur data dari solusi untuk permasalahan komputasi bobot sisi maksimum pada *Maximum Edge of Powers of Permutation* berdasarkan studi literatur sebelumnya.
4. Implementasi
   Pada tahap ini dilakukan implementasi solusi dari permasalahan komputasi bobot sisi maksimum pada *Maximum Edge of Powers of Permutation* berdasarkan analisis dan desain sebelumnya.
5. Pengujian dan evaluasi
   Pada tahap ini dilakukan dengan uji coba kebenaran dan kinerja solusi yang telah diimplementasi dengan melakukan pengiriman sumber kode sistem ke situs penilaian daring SPOJ pada permasalahan yang terkait dan melihat hasil umpan balik. Pengujian dilakuan dengan membandingan kompleksitas hasil uji coba dengan kompleksitas hasil analisis.
6. Penyusunan buku tugas akhir

Tahap ini merupakan tahap penyusunan laporan berupa buku sebagai dokumentasi pengerjaan tugas akhir yang mencakup seluruh dasar teori, desain, implementasi serta hasil pengujian yang telah dilakukan.

1.6 Sistematika Penulisan

Penulisan buku tugas akhir ini dibagi kedalam 6 bab yang masing-masing membahas bagian-bagian yang disusun dalam menyelesaikan persoalan yang terkait, yaitu:

1. Bab 1: Pendahuluan
   Bab ini berisi penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi, dan sistematika penulisan buku.
2. Bab 2: Dasar Teori
   Bab ini berisi penjelasan mengenai teori-teori yang digunakan sebagai dasar pengerjaan tugas akhir ini.
3. Bab 3: Desain
   Bab ini berisi rancangan pembuatan sistem penyelesaian permasalahan dalam tugas akhir ini.
4. Bab 4: Implementasi
   Bab ini berisi lingkungan serta hasil penerapan rancangan sistem penyelesaian permasalahan dalam tugas akhir ini dalam bentuk sumber kode beserta penjelasannya.
5. Bab 5: Uji Coba dan Evaluasi
   Bab ini berisi lingkungan serta hasil dari rangkaian uji coba yang dilakukan untuk menguji kebenaran serta kinerja dari sistem.
6. Bab 6: Kesimpulan dan Saran
   Bab ini berisi kesimpulan pengerjaan tugas akhir ini dan saran untuk pengembangan kedepannya.

BAB 2
DASAR TEORI

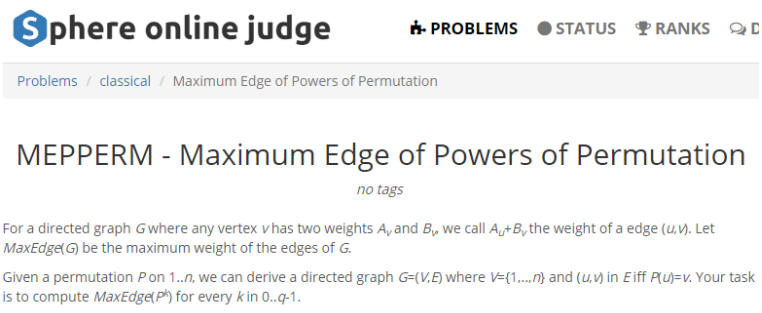
Bab ini akan membahas mengenai dasar teori dan literatur yang menjadi dasar pengerjaan tugas akhir ini. Pada subbab 2.1 membahas mengenai deskripsi permasalahan. Pada subbab 2.2 membahas mengenai contoh kasus permasalahan. Pada subbab 2.3 membahas mengenai definisi umum graf, permutasi, polinomial, konvolusi, dan *root of unity* yang menjadi dasar dari pemahaman masalah dan penyelesaian masalah ini. Pada subbab 2.4 membahas mengenai transformasi Fourier diskrit. Pada subbab 2.5 membahas mengenai transformasi Fourier cepat yang menjadi algoritma dasar dari penyelesaian soal ini. Pada subbab 2.6 membahas mengenai penyelesaian soal ini secara lengkap. Pada subbab 2.7 membahas mengenai pembuatan data *generator* untuk uji coba.

2.1 Deskripsi Permasalahan

Permasalahan yang diangkat dalam tugas akhir ini diangkat dari suatu permasalahan yang terdapat pada suatu situs penilaian daring atau *online judge* SPOJ yaitu *Maximum Edge of Powers of Permutation* dengan nomor soal 6895 dan kode soal MEPPERM [1], deskripsi soal dari sumber asli menggunakan bahasa Inggris terdapat pada Gambar 2.1.

Pada permasalahan *Maximum Edge of Powers of Permutation* diberikan N simpul yang masing-masing memiliki nilai bobot A_{V_i} dan B_{V_i}. Diberikan suatu permutasi P pada N simpul, dengan permutasi tersebut dapat dibentuk suatu graf G dimana setiap simpul memiliki sisi ke hasil permutasi simpul. Setiap sisi memiliki bobot perjumlahan bobot A_{V_i} simpul V_i dan bobot B_{V_j} hasil permutasi simpul V_i yaitu V_j. Kemudian dicari bobot sisi

maksimum pada sejumlah graf yang dibentuk dari permutasi P^0 hingga P^{Q-1} terhadap N simpul tersebut.



Gambar 2.1 Deskripsi permasalahan Maximum Edge of Powers of Permutation pada SPOJ

Format masukan pada baris pertama diberikan bilangan bulat positif N yang merupakan jumlah simpul. Pada baris kedua terdapat N bilangan bulat positif unik dari himpunan $\{1, \dots, N\}$ yang merepresentasikan permutasi P. Pada baris ketiga diberikan N bilangan bulat non-negatif yang merepresentasikan bobot A_{V_i} hingga A_{V_N}. Pada baris keempat diberikan N bilangan bulat non-negatif yang merepresentasikan bobot B_{V_1} hingga B_{V_N}. Pada baris kelima diberikan bilangan bulat positif Q yang merepresentasikan batas pangkat permutasi terbesar.

Format keluaran terdiri dari satu baris yang berisi Q bilangan bulat yang merupakan nilai bobot sisi maksimum pada graf yang dibentuk dari permutasi P^0 hingga P^{Q-1}.

Deskripsi format masukan, format keluaran beserta contoh masukan dan keluaran dapat dilihat pada Gambar 2.2.

Input

The first line contains a positive integer n.
The second line contains n integers in $\{1,...,n\}$, denoting the permutation P.
The third and the fourth line both contain n natural numbers, $A_1,...,A_n$ and $B_1,...,B_n$ respectively.
The fifth line contains a positive integer q.

Output

The only one line contains q integers $MaxEdge(P^0),...,MaxEdge(P^{q-1})$, separated by a single space.

Example

```
Input:
3
3 2 1
0 1 2
2 2 0
5

Output:
3 4 3 4 3
```

Gambar 2.2 Contoh masukan dan keluaran dari permasalahan MEPPERM

Batasan dari permasalahan *Maximum Edge of Powers of Permutation* adalah sebagai berikut:

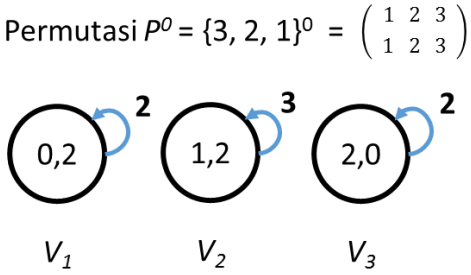
1. $N \leq 66000$.
2. $A_{V_i}, B_{V_i} \leq 16$.
3. $Q \leq 1000000$.
4. Lingkungan penilaian Intel Pentium G860 3GHz.
5. Batas Waktu: 1 – 3 detik.
6. Batas Sumber Kode: 50000 B.
7. Batas Memori: 1536 MB.

2.2 Contoh Kasus Permasalahan

Diketahui N bernilai 3 yang merupakan jumlah simpul. Setiap simpul memiliki bobot A yang secara berurutan adalah $\{0, 1, 2\}$ dan bobot B secara berurutan yaitu $\{2, 2, 0\}$. Permutasi P secara

berurutan adalah $\{3, 2, 1\}$ dan Q bernilai 3 yang merupakan batas pangkat permutasi terbesar sehingga nilai bobot sisi maksimum yang perlu dicari adalah P^0 hingga P^{Q-1}.

Pada permutasi P^0, hasil permutasi setiap simpul merupakan simpul itu sendiri, sehingga terbentuk graf dimana terdapat sejumlah sisi yang menghubungi setiap simpul ke simpul itu sendiri. Ilustrasi graf dapat dilihat pada Gambar 2.3.



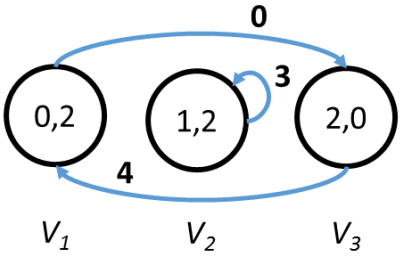
Gambar 2.3 Ilustrasi graf terhadap permutasi P^0

Pada ilustrasi diatas, setiap simpul direpresentasikan dengan nilai (A_{V_i}, B_{V_i}). Setiap sisi menghubungi simpul dengan simpul hasil permutasinya dengan bobot $A_{V_i} + B_{V_j}$. Dari graf tersebut dapat ditentukan bahwa pada permutasi P^0 memiliki bobot sisi maksimal yaitu 3 dimana sisi tersebut menghubungkan simpul V_2 ke V_2.

Pada permutasi P^1, hasil permutasi setiap simpul merupakan nilai permutasi simpul itu sendiri. Ilustrasi graf dapat dilihat pada Gambar 2.4.

Pada ilustrasi Gambar 2.4 dapat ditentukan bahwa pada permutasi P^1 memiliki bobot sisi maksimal yaitu 4 dimana sisi tersebut menghubungkan simpul V_3 ke V_1.

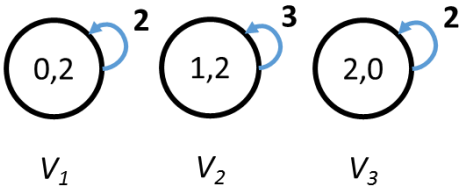
Permutasi P^1 = {3, 2, 1} = $\begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$



Gambar 2.4 Ilustrasi graf terhadap permutasi P^1

Pada permutasi P^2, hasil permutasi setiap simpul adalah simpul sendiri. Hal ini disebabkan oleh hasil permutasi dari P^2 sama dengan P^0 sehingga tentu saja bobot sisi maksimal permutasi P^2 sama dengan P^0 yaitu 3 dimana sisi tersebut menghubungkan V^2 ke V^2. Ilustrasi perkalian permutasi dan ilustrasi graf dapat dilihat pada Gambar 2.5.

Permutasi P^2 = {3,2,1}^{2} = $\begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$ x $\begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$ = $\begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$



Gambar 2.5 Ilustrasi graf terhadap permutasi P^2

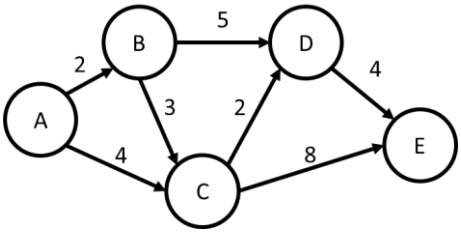
Dari hasil evaluasi bobot sisi maksimum pada permutasi P^0 hingga P^2 maka keluaran yang diharapkan dari pangkat permutasi 0 hingga $Q - 1$ secara berurutan adalah $\{3, 4, 3\}$.

2.3 Definisi Umum

Dalam subbab ini membahas definisi-definisi umum yang akan digunakan sebagai dasar untuk memahami soal dan menyelesaikan permasalahan ini.

2.3.1 Graf

Suatu graf [2] adalah suatu himpunan simpul(*vertex*) dan himpunan sisi(*edge*), dimana setiap sisi menghubungi dua buah simpul. Graf berarah adalah graf dimana setiap sisi memiliki arah dari suatu simpul ke simpul yang terhubung. Graf berbobot adalah graf dimana setiap sisi atau simpul memiliki nilai bobot. Graf yang memiliki sifat graf berarah dan graf berbobot disebut graf berbobot berarah. Ilustrasi graf berbobot terarah dapat dilihat pada Gambar 2.6.



Gambar 2.6 Ilustrasi graf berbobot terarah

2.3.2 Permutasi

Suatu permutasi [3] adalah suatu pengurutan posisi elemen-elemen pada suatu himpunan berdasarkan pola tertentu. Ilustrasi bentuk permutasi yang mungkin dibentuk dari suatu himpunan yang berisi 3 elemen ditunjukkan pada Tabel 2.1.

Tabel 2.1 Permutasi dari suatu himpunan yang berisi 3 elemen

1	$1 \rightarrow 1$	$2 \rightarrow 2$	$3 \rightarrow 3$
2	$1 \rightarrow 2$	$2 \rightarrow 1$	$3 \rightarrow 3$
3	$1 \rightarrow 3$	$2 \rightarrow 2$	$3 \rightarrow 1$
4	$1 \rightarrow 1$	$2 \rightarrow 3$	$3 \rightarrow 2$
5	$1 \rightarrow 2$	$2 \rightarrow 3$	$3 \rightarrow 1$
6	$1 \rightarrow 3$	$2 \rightarrow 1$	$3 \rightarrow 2$

Terdapat dua bentuk notasi permutasi, yaitu:

a. Notasi dua baris
   Notasi ini berisi dua baris dimana pada baris pertama berisi setiap posisi elemen pada himpunan, kemudian pada baris kedua berisi perubahan posisi setiap elemen yang sekolom. Ilustrasi permutasi dua baris ditunjukkan pada Gambar 2.7.

$$Permutasi\ P = \{3, 2, 1\} = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$$

Gambar 2.7 Ilustrasi permutasi dua baris

b. Notasi siklus(*cycle*)
   Notasi ini berisi sejumlah siklus yang merupakan perubahan posisi elemen secara siklus. Suatu permutasi dapat terdiri dari lebih dari satu siklus. Ilustrasi permutasi siklus ditunjukkan pada Gambar 2.8.

$$Permutasi\ P = \{3, 2, 1\} = (1\ 3)(2)$$

Gambar 2.8 Ilustrasi permutasi siklus

2.3.3 Polinomial

Polinomial [2] adalah suatu pernyataan matematika berbentuk fungsi yang merupakan jumlah perkalian pangkat dalam satu atau lebih variabel dengan koefisien. Bentuk persamaannya adalah sebagai berikut:

$$A(x) = \sum_{j=0}^{n-1} a_j x^j \qquad (2, 1)$$

Bentuk a_j merepresentasikan nilai koefisien pada derajat j, sedangkan bentuk x^j merepresentasikan nilai variabel x pada derajat j. Derajat(*degree*) suatu polinomial ditentukan berdasarkan nilai j terbesar dimana terdapat a_j yang bernilai tidak nol. Batas derajat(*degree-bound*) suatu polinomial adalah nilai derajat polinomial tersebut ditambah satu.

Terdapat dua bentuk yang dapat merepresentasikan polinomial, yaitu:

a. Representasi koefisien
   Polinomial direpresentasikan dengan suatu larik berupa koefisien yang diletakkan berurutan berdasarkan derajatnya.
   $$a = (a_0, a_1, \dots, a_{n-1}) \qquad (2, 2)$$

   Evaluasi polinomial dapat dilakukan dalam kompleksitas komputasi O(n) menggunakan metode Horner(*Horner's rule*).
   $$(x) = a_0 + x(a_1 + \cdots + x(a_{n-1}) \dots) \qquad (2, 3)$$

   Perjumlahan dua polinomial dapat dilakukan dalam kompleksitas komputasi $O(n)$. Sedangkan perkalian atau konvolusi dua polinomial dengan cara biasa dapat dilakukan dalam kompleksitas komputasi $O(n^2)$.

b. Representasi titik-nilai(*point-value*)
   Polinomial direpresentasikan dengan suatu larik yang berisi pasangan titik-nilai sebanyak batas derajat polinomial dimana setiap titik berbeda-beda.

Perkalian atau konvolusi dua polinomial dapat dilakukan dalam kompleksitas komputasi $O(nlogn)$ dengan menggunakan algoritma transformasi Fourier cepat(*FFT*). Proses invers dari evaluasi untuk mengubah bentuk polinomial dari representasi titik-nilai ke representasi koefisien disebut interpolasi.

2.3.4 Konvolusi

Konvolusi [2] adalah suatu operasi perkalian pada dua buah polinomial membentuk satu buah polinomial gabungan. Batas derajat hasil polinomial adalah $n + m - 1$ dimana n dan m merupakan batas derajat kedua polinomial. Bentuk persamaan dari polinomial hasil konvolusi adalah sebagai berikut:

$$C(x) = \sum_{j=0}^{2n-2} c_j x^j \qquad (2,4)$$

dimana

$$c_j = \sum_{k=0}^{j} a_k b_{j-k} \qquad (2,5)$$

Dari fungsi tersebut, dapat ditentukan bahwa dibutuhkan kompleksitas komputasi $O(n^2)$. Dengan menggunakan algoritma seperti Toom-Cook, Karatsuba, dan transformasi Fourier cepat, kompleksitas komputasi dapat berkurang hingga mencapai $O(nlogn)$. Pada Gambar 2.9 mengilustrasikan konvolusi dua buah polinomial.

A

A_0	A_1	A_2	A_3
a	0	b	0

X

B

B_0	B_1	B_2	B_3
c	d	0	e

C

C_0	C_1	C_2	C_3	C_4	C_5	C_6
ac	ad	bc	ae + bd	0	be	0

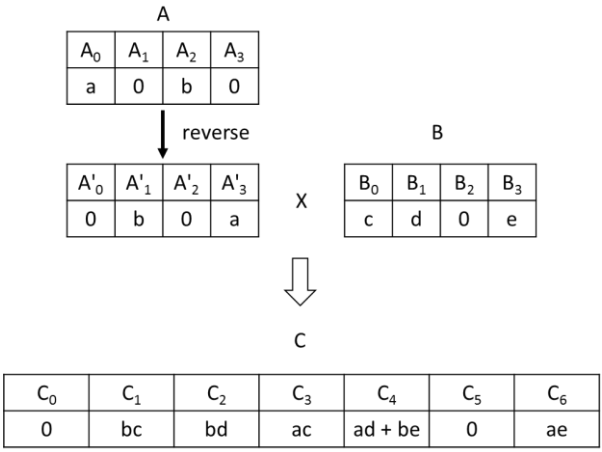
Gambar 2.9 Ilustrasi konvolusi polinomial A dan B

Dengan memahami persamaan konvolusi (2, 4) dan (2, 5) disertai dengan ilustrasi pada Gambar 2.9, dapat diketahui bahwa nilai C_{j+k} bertambah sebanyak $A_j * B_k$ apabila kedua nilai A_j dan B_k bukan nol. Misal pada Gambar 2.9, nilai C_3 adalah $A_0B_3 + A_1B_2 + A_2B_1 + A_3B_0$, dikarenakan A_1B_2 dan A_3B_0 bernilai 0 sehingga nilai C_3 disederhanakan menjadi $A_0B_3 + A_2B_1$ yaitu $ae + bd$. Hal ini juga dapat dibuktikan bahwa pada saat mencari nilai C_i, maka perjumlahan posisi koefisien A_j dan B_k yang terlibat pada setiap perkalian koefisien bernilai i.

Konvolusi memiliki sisi lain, yaitu korelasi. Korelasi merupakan konvolusi antar polinomial dimana salah satu polinomial urutan koefisiennya dibalik (*reverse*) sehingga terbentuk persamaan berikut:

$$c_j = \sum_{k=0}^{j} a_{n-1-k} b_{j-k} \qquad (2, 6)$$

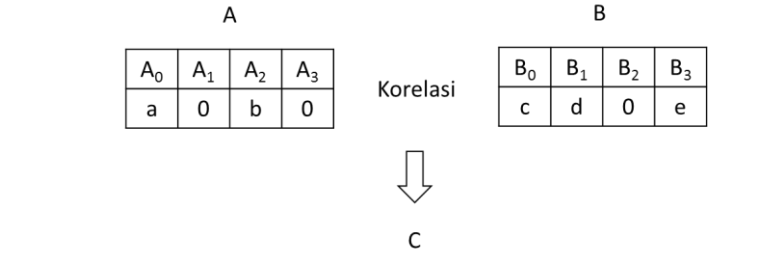
Pada persamaan (2, 6) larik A dilakukan pemindahan posisi koefisien secara *bit-reversal* sehingga pada posisi awal A_k bertukar posisi dengan A_{n-1-k}. Ilustrasi dari korelasi dua buah larik dapat dilihat pada Gambar 2.10.



Gambar 2.10 Ilustrasi korelasi polinomial A dan B menggunakan konvolusi

Dengan memahami persamaan korelasi (2, 6) disertai dengan ilustrasi pada Gambar 2.10, dapat diketahui bahwa nilai $C_{n-1-j+k}$ bertambah sebanyak $A_{n-1-j} * B_k$ apabila kedua nilai A_{n-1-j} dan B_k bukan nol. Hal ini terjadi karena larik A mengalami *bit-reverse* menjadi larik A', maka terjadi pertambahan pada C_{j+k} dari perjumlahan setiap $A'_j * B_k$. Dan apabila ditransformasi menggunakan larik A, maka nilai j berubah menjadi $n-1-j$ sehingga terjadi pertambahan pada $C_{n-1-j+k}$ dari perjumlahan setiap $A_j * B_k$. Dari pengamatan tersebut, dikarenakan larik C menyimpan dari 0 hingga $2n-2$, sehingga dapat diambil kesimpulan bahwa C_0 hingga C_{n-2} menyimpan perjumlahan setiap $A_{n-1-j} * B_k$ dimana $k-j<0$, sedangkan C_{n-1} hingga C_{2n-2}

menyimpan perjumlahan setiap $A_{n-1-j} * B_k$ dimana $k - j \geq 0$. Ilustrasi kesimpulan yang didapatkan dapat dilihat pada Gambar 2.11 menggunakan contoh sebelumnya.



A_0	A_1	A_2	A_3
a	0	b	0

B_0	B_1	B_2	B_3
c	d	0	e

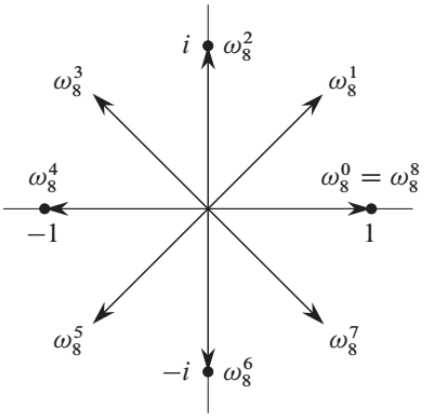
	C_0	C_1	C_2	C_3	C_4	C_5	C_6
k - j	-3	-2	-1	0	1	2	3
	0	bc	bd	ac	ad + be	0	ae

Gambar 2.11 Ilustrasi korelasi polinomial A dan B

Dari ilustrasi diatas, nilai C_4 dimana $k - j = 1$ adalah $A_0B_1 + A_1B_2 + A_2B_3$, dikarenakan nilai A_1B_2 adalah 0 sehingga nilai C_4 menjadi $A_0B_1 + A_2B_3$ yaitu $ad + be$. Dengan begitu dapat disimpulkan bahwa untuk mencari nilai korelasi larik *A* dan *B* pada posisi $k - j$ dapat dicari pada nilai $C_{n-1-j+k}$.

2.3.5 Root of Unity

Nth root of unity [2] adalah bilangan kompleks dimana $\omega^n = 1$. Terdapat *n* bilangan kompleks unik yang merupakan *nth root of unity* dimana terdiri dari nilai $e^{2\pi ik/n}$ untuk setiap $k = 0, 1, \ldots, n - 1$. Dengan menggunakan rumus Euler, dapat dinyatakan bahwa $e^{2\pi ik/n} = cos(2\pi k/n) + i\, sin(2\pi k/n)$. Ilustrasi *8th root of unity* pada bidang kompleks dapat dilihat pada Gambar 2.12.



Gambar 2.12 Ilustrasi 8th root of unity pada bidang kompleks

Root of unity memiliki beberapa sifat unik, dua diantaranya adalah:

a. Sifat Refleksi(*reflective*)
   Suatu nilai ω yang merupakan *nth roof of unity* dan nilai *n* adalah genap, maka $\omega^{k+n/2} = -\omega^k$.
b. Sifat Reduksi(*reduction*)
   Apabila ω adalah *2nth root of unity*, maka ω^2 adalah *nth root of unity*.

2.4 Tranformasi Fourier Diskrit

Transformasi Fourier Diskrit(DFT) [2] adalah perubahan bentuk polinomial dari bentuk koefisien menjadi bentuk titik-nilai dimana polinomial dievaluasi menggunakan nilai titik yang merupakan bilangan kompleks *nth root of unity*. Jumlah titik yang dievaluasi sebanyak batas derajat polinomial tersebut sehingga membutuhkan *n* bilangan kompleks *nth root of unity* untuk menghasilkan titik yang berbeda-beda. Bentuk persamaan dalam menentukan nilai pada setiap titik adalah sebagai berikut:

$$y_k = A\left(\omega_n^k\right) = \sum_{j=0}^{n-1} a_j \omega_n^{kj} \quad (2, 7)$$

Dari rumus tersebut, dapat ditentukan bahwa untuk menentukan setiap titik-nilai dibutuhkan kompleksitas waktu $O(N^2)$.

Proses interpolasi atau invers tranformasi Fourier Diskrit(IDFT) adalah perubahan bentuk polinomial dari titik-nilai menjadi bentuk koefisien dengan melakukan perkalian matriks dari invers matriks Vandermonde berisi pangkat dari ω_n dan bentuk titik-nilai. Bentuk persamaan untuk melakukan perubahan bentuk polinomial dari titik-nilai menjadi bentuk koefisien adalah sebagai berikut:

$$a_j = \frac{1}{n} \sum_{k=0}^{n-1} y_k \omega_n^{-kj} \quad (2, 8)$$

Dari persamaan diatas dapat dibandingkan dengan persamaan (2, 7) bahwa perbedaan DFT dengan IDFT terletak pada pembagi dengan n dan pangkat yang berupa minus pada ω, sehingga implementasi IDFT dapat dibuat berdasarkan implementasi DFT.

2.5 Transformasi Fourier Cepat

Transformasi Fourier cepat(FFT) [2] adalah suatu algoritma yang digunakan untuk mengomputasi DFT dengan memanfaatkan sifat unik *root of unity* sehingga kompleksitas waktu menjadi $O(nlogn)$. Terdapat beberapa algoritma FFT seperti Cooley-Tukey, Schönhage–Strassen, dan Fürer. Dalam konteks buku ini, algoritma FFT mengacu pada algoritma Cooley-Tukey [4], sehingga batas derajat diasumsi merupakan kelipatan dua.

Transformasi Fourier cepat menggunakan strategi *divide and conquer* dengan membagi koefisien indeks genap dan ganjil fungsi polinomial $A(x)$ dengan batas derajat n menjadi sebagai berikut:

$$A_{genap}(x) = a_0 + a_2x + a_4x^2 + a_{n-2}x^{n/2-1}$$
$$A_{ganjil}(x) = a_1 + a_3x + a_5x^2 + a_{n-1}x^{n/2-1} \quad (2, 9)$$

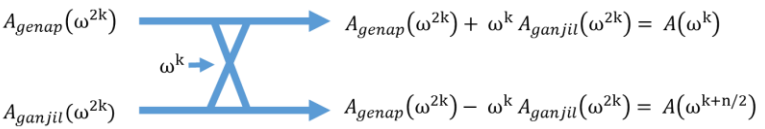
Fungsi polinomial $A_{genap}(x)$ menyimpan seluruh koefisien genap fungsi polinomial $A(x)$ sedangkan $A_{ganjil}(x)$ menyimpan seluruh koefisien ganjil fungsi polinomial $A(x)$. Maka dapat dibentuk:

$$A(x) = A_{genap}(x^2) + xA_{ganjil}(x^2) \quad (2, 10)$$

Pada komputasi DFT, nilai x merupakan *nth root of unity* yang mewakili titik evaluasi pada fungsi polinomial $A(x)$. Dengan sifat reduksi pada *root of unity*, diketahui bahwa $\left(\omega_n^k\right)^2$ merupakan $\omega_{n/2}^k$ yang juga merupakan subset dari ω_n^k. Dan juga dengan sifat refleksi pada *root of unity* dimana $\omega^{k+n/2} = -\omega^k$, dapat dibentuk persamaan sebagai berikut:

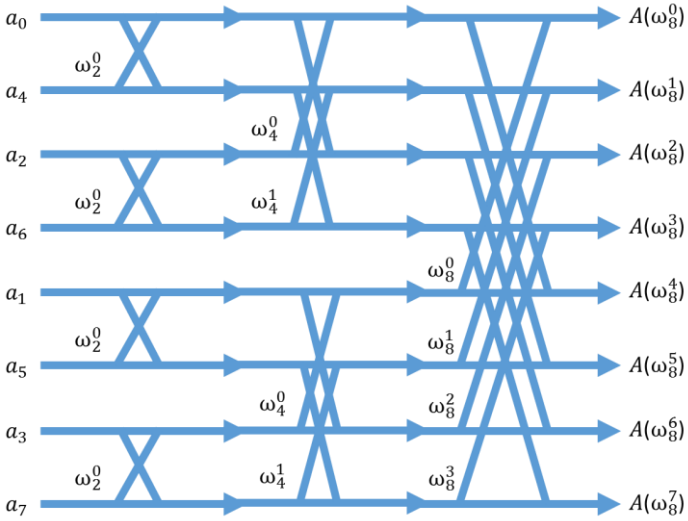
$$A\left(\omega^k\right) = A_{genap}\left(\omega^{2k}\right) + \omega^k A_{ganjil}(\omega^{2k}) \quad (2, 11)$$
$$A\left(\omega^{k+n/2}\right) = A_{genap}\left(\omega^{2k}\right) - \omega^k A_{ganjil}(\omega^{2k})$$

Persamaan (2, 11) dapat direpresentasikan seperti pada Gambar 2.13 dimana disebut sebagai operasi kupu-kupu(*butterfly operation).*



Gambar 2.13 Ilustrasi operasi kupu-kupu (Butterfly operation)

Dengan begitu cukup melakukan evaluasasi terhadap titik ω_n^0 hingga $\omega_n^{n/2-1}$ untuk mendapatkan hasil evaluasi dari titik ω_n^0 hingga ω_n^n. Kemudian untuk melakukan evaluasi A_{genap} dan A_{ganjil} dapat dilakukan dengan cara yang sama, yaitu membagi-dua berdasarkan derajat yang bernilai genap dan ganjil seperti pada persamaan (2, 11) hingga derajat maksimal yang dimiliki adalah 1. Dengan menggunakan cara tersebut, maka komputasi dapat dilakukan secara rekursif. Ilustrasi pembagian polinomial secara rekursif dapat dilihat pada Gambar 2.14.



Gambar 2.14 Ilustrasi operasi kupu-kupu pada FFT dengan polinomial memiliki batas derajat 8 yang dilakukan secara rekursif

Pada Gambar 2.14 dapat diperhatikan bahwa posisi koefisien tidak terletak secara berurutan berdasarkan batas derajat melainkan posisi koefisien bertukaran dengan posisi yang memiliki nilai posisi dengan bit terbalik(*bit-reverse)*, sehingga dalam implementasi perlu dilakukan proses penukaran posisi koefisien

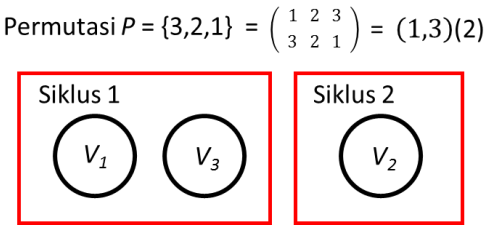
terlebih dahulu. Dengan pendekatan *divide and conquer* ini, maka komputasi transformasi Fourier cepat dapat dilakukan dengan kompleksitas komputasi $O(nlogn)$.

2.6 Penyelesaian Permasalahan Maximum Edge of Powers of Permutation

Permasalahan *Maximum Edge of Powers of Permutation* dapat diselesaikan dengan menggunakan algoritma transformasi Fourier cepat untuk menghitung bobot sisi maksimum pada setiap permutasi siklus dari permutasi P. Berikut adalah tahap-tahap dalam menyelesaikan permasalahan ini.

2.6.1 Pembentukan permutasi siklus

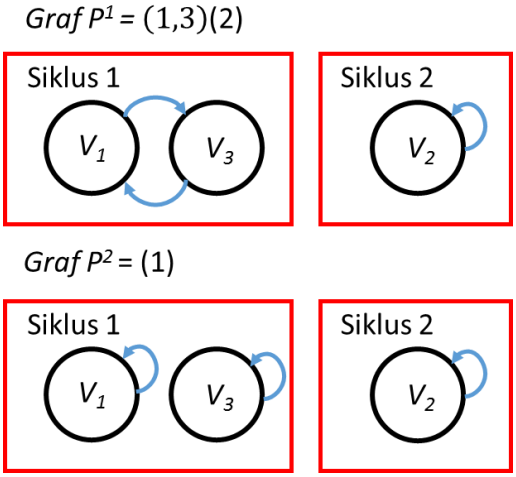
Pada tahap pertama, permutasi P diubah bentuk dari notasi dua baris menjadi notasi siklus sehingga membentuk sejumlah permutasi siklus. Hal ini dilakukan untuk mempermudah perhitungan bobot sisi maksimum pada setiap siklus. Setiap permutasi siklus terdiri dari sejumlah simpul yang berurutan berdasarkan permutasinya. Ilustrasi perubahan notasi dapat dilihat pada Gambar 2.15.



Gambar 2.15 Ilustrasi perubahan notasi permutasi P menjadi notasi siklus

Pada suatu pangkat permutasi P^q, setiap simpul pada posisi i akan memiliki sisi dengan simpul pada posisi $i + q$ pada siklus dimana simpul tersebut berada, sehingga terbentuk suatu graf dengan sisi berjumlah N yaitu banyaknya simpul. Karena sifat siklus, apabila

$i + q$ lebih besar dari pada ukuran siklus tersebut, maka nilai tersebut dimodulo dengan ukuran siklus. Hal ini terjadi karena hasil permutasi simpul pada posisi terakhir pada suatu siklus adalah simpul pada posisi awal. Selain itu, simpul pada suatu siklus tidak akan berhubungan atau memiliki sisi pada simpul dengan siklus yang berbeda pada pangkat permutasi berapapun. Ilustrasi graf yang terbentuk dari pangkat permutasi *P* dapat dilihat pada Gambar 2.16.

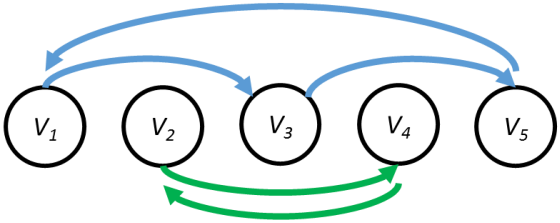


Gambar 2.16 Ilustrasi graf yang terbentuk dari pangkat permutasi P

Karena bobot sisi maksimum pada suatu pangkat permutasi dapat dihitung secara terpisah berdasarkan siklusnya, maka dilakukan perhitungan bobot sisi maksimum untuk setiap pangkat permutasi pada setiap siklus, kemudian digabungkan hasil perhitungan tersebut untuk mencari bobot sisi maksimum pada seluruh siklus. Untuk mendapatkan siklus-siklus tersebut dapat dicari dengan melakukan *traversal* berdasarkan permutasi pada simpul yang belum menjadi bagian dari siklus manapun. *Traversal* simpul dilakukan hingga bertemu simpul yang menjadi awal permulaan

traversal. Setiap simpul yang dilewati pada *traversal* membentuk suatu siklus dimana simpul diurutkan berdasarkan urutan *traversal*. Misal pada $P = \{3,4,5,2,1\}$, maka dimulai dari simpul V_1 ke V_3 ke V_5 kemudian kembali ke simpul V_1, sehingga terbentuk siklus $(1,3,5)$. Kemudian selanjutnya dimulai dari simpul V_2 ke V_4 kemudian kembali ke simpul V_1, sehingga terbentuk siklus $(2,4)$. Dengan demikian setiap simpul telah menjadi bagian dari siklus tertentu, sehingga terbentuk 2 siklus, yaitu (1,3,5) dan (2,4). Ilustrasi pencarian siklus berdasarkan permutasi P dapat dilihat pada Gambar 2.17.

$$\text{Permutasi } P = \{3, 4, 5, 2, 1\} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 2 & 1 \end{pmatrix} = (1, 3, 5)(2, 4)$$



Gambar 2.17 Ilustrasi pencarian siklus berdasarkan permutasi P

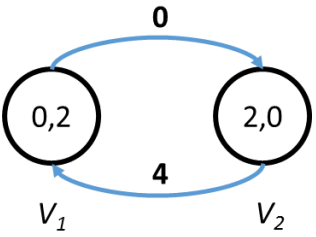
Pada ilustrasi diatas, pencarian siklus pertama dimulai dari simpul V_1 (garis biru), kemudian siklus kedua dimulai dari simpul V_2 (garis hijau). Pencarian siklus menggunakan traversal pada simpul yang belum menjadi bagian dari suatu siklus dapat dilakukan dengan kompleksitas komputasi $O(N)$.

2.6.2 Perhitungan bobot sisi maksimum tiap siklus

Perhitungan bobot sisi maksimum tiap siklus pada setiap pangkat permutasinya dapat dilakukan secara naif dengan menggunakan metode *complete search*. Metode ini dengan mudah dilakukan dengan cara mengiterasi setiap simpul pada siklus untuk mencari

bobot sisi simpul terhadap simpul yang memiliki jarak pangkat permutasi yang diinginkan. Dengan membandingkan setiap bobot sisi didapatkan bobot sisi maksimum pada pangkat permutasi tersebut. Karena pada suatu pangkat permutasi perlu dilakukan iterasi sebanyak N yang merupakan banyak simpul pada siklus tersebut, dan terdapat N permutasi unik dari setiap pangkat permutasi dari suatu siklus yang terdiri dari N simpul, maka komputasi perhitungan bobot sisi maksimum untuk setiap pangkat permutasi dari suatu siklus memiliki kompleksitas sebesar $O(N^2)$.

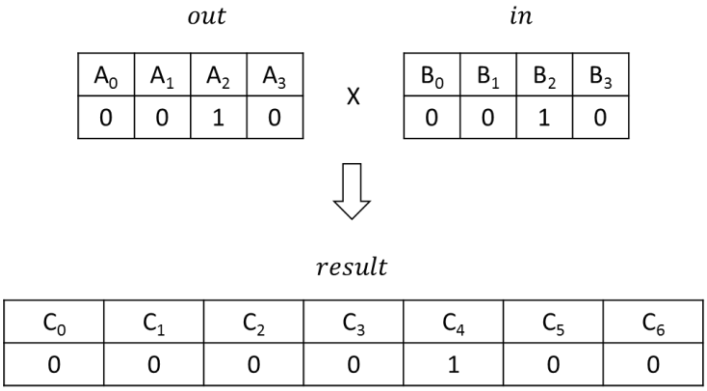
Terdapat metode yang lebih cepat menggunakan konvolusi dan korelasi dengan algoritma transformasi Fourier cepat(FFT). Konvolusi digunakan untuk mengetahui keberadaan suatu bobot sisi pada graf yg terbentuk berdasarkan suatu pangkat permutasi. Misal terdapat suatu graf yang terbentuk dari suatu pangkat permutasi terhadap suatu siklus pada Gambar 2.18 dimana setiap simpul memiliki nilai (A_{V_i}, B_{V_i}).



Gambar 2.18 Ilustrasi suatu graf yang terbentuk dari suatu permutasi terhadap suatu siklus

Perhitungan bobot sisi dari setiap simpul ke simpul lainnya bisa dilakukan dengan metode konvolusi diawali dengan membuat larik out dan in sebesar $M = A_{V_i} + B_{V_j} + 1$ dimana A_{V_i} dan B_{V_j} merupakan bobot simpul terbesar. Kemudian untuk mencari bobot sisi pada simpul V_i ke V_j bisa dilakukan dengan memberikan nilai

1 pada posisi A_{V_i} pada larik out dan juga pada posisi A_{V_j} pada larik in. Selanjutnya lakukan konvolusi pada larik out dan in. Ilustrasi pencarian bobot sisi V_2 ke V_1 pada Gambar 2.18 dapat dilihat pada Gambar 2.19.



Gambar 2.19 Ilustrasi perhitungan bobot sisi V_2 ke V_1 menggunakan konvolusi

Dari ilustrasi tersebut diketahui terdapat nilai 1 pada posisi C_4 pada larik hasil konvolusi, sehingga diketahui bobot sisi V_2 ke V_1 adalah 4. Karena metode konvolusi ini memerlukan kompleksitas sebesar $O(MlogM)$, maka perhitungan bobot sisi maksimum untuk setiap permutasi pada suatu siklus membutuhkan kompleksitas $O(N^2MlogM)$ yang tentunya lebih buruk dibanding metode naif yang sebesar $O(N^2)$. Akan tetapi terdapat cara untuk menghitung bobot sisi maksimum untuk setiap permutasi secara bersamaan, yaitu menggabungkan konvolusi dan korelasi. Karena korelasi merupakan konvolusi dimana salah satu larik mengalami *bit-reverse* sehingga kedua metode tersebut dapat dilakukan secara bersamaan untuk mendapatkan hasil yang diinginkan. Konvolusi dilakukan untuk menghitung bobot sisi antar simpul seperti pada Gambar 2.19, sedangkan korelasi digunakan untuk menentukan letak bobot simpul A_i dan B_i pada posisi simpul yang sesuai

sehingga hasil konvolusi menyimpan bobot-bobot sisi yang terletak pada pangkat permutasi yang sesuai. Dengan metode ini, maka kompleksitas komputasi menjadi $O(NMlogNM)$ yang lebih cepat dibanding metode naif $O(N^2)$ dengan diketahui batas terbesar nilai N adalah 66000 sedangkan M adalah 33. Hal ini diawali dengan membuat larik *coefA* dan *coefB* dimana masing-masing sebesar $N * M$, kemudian lakukan konvolusi terhadap kedua larik tersebut. *N* merupakan banyak simpul dalam siklus sehingga setiap kelipatan *M* merepresentasikan posisi simpul pada siklus tersebut. Dimisalkan i merupakan posisi simpul awal sedangkan j merupakan posisi simpul akhir dengan asumsi posisi simpul dimulai dari 0 hingga $N-1$. Agar hasil konvolusi menampilkan nilai pada posisi pangkat permutasi($j-i$), maka posisi simpul i dilakukan *bit-reverse* menjadi $N-1-i$, sehingga korelasi bisa dilakukan menggunakan konvolusi. Sedangkan *M* merupakan perjumlahan A_{V_i} dan B_{V_j} terbesar pada siklus tersebut ditambah 1, sehingga *M* merepresentasikan nilai bobot. Kemudian masing-masing simpul V_i, berikan nilai 1 pada larik *coefA* pada posisi $(N-1-i) * M + A_{V_i}$ dan pada larik $coefB$ pada posisi $i * M + B_{V_i}$. Hasil konvolusi larik $coefA$ dan $coefB$ akan menampilkan nilai 1 atau lebih pada posisi $(j + (N-1-i)) * M + A_{V_i} + B_{V_j}$ dimana $N-1-i$ dan A_{V_i} merepresentasikan posisi dan bobot pada larik $coefA$, sedangkan j dan B_{V_j} merepresentasikan posisi dan bobot larik $coefB$. Penjelasan singkat konvolusi larik $coefA$ dan $coefB$ dapat dilihat pada Tabel 2.2.

Dari hasil konvolusi pada larik $coefC$ dapat disimpulkan bahwa apabila terdapat nilai 1 atau lebih pada posisi k, maka terdapat bobot sisi $k\%M$ pada pangkat permutasi $\left\lfloor \frac{k}{M} \right\rfloor$. Karena nilai $\left\lfloor \frac{k}{M} \right\rfloor$ berkisaran dari 0 hingga $2N-2$ dan merepresentasikan hasil

korelasi, maka aturan korelasi berlaku dimana pada posisi 0 hingga $N-2$ merepresentasikan $j-i$ dimana $j-i<0$, sedangkan pada posisi $N-1$ hingga $2N-2$ merepresentasikan $j-i$ dimana $j-i\geq 0$. Pada $j-i$ dimana $j-i<0$ merupakan pangkat permutasi dimulai dari i hingga j dimana i lebih besar daripada j. Hal ini mungkin terjadi karena sifat siklus. Oleh karena itu nilai pangkat permutasi $\left\lfloor\frac{k}{M}\right\rfloor$ berkisar 0 hingga $N-2$ sebenarnya adalah $\left\lfloor\frac{k}{M}\right\rfloor+1$, sedangkan nilai pangkat permutasi $\left\lfloor\frac{k}{M}\right\rfloor$ berkisar $N-1$ hingga $2N-2$ sebenarnya adalah $\left\lfloor\frac{k}{M}\right\rfloor-(N-1)$. Dari hasil konvolusi larik $coefC$ tersebut dapat disimpulkan bobot sisi maksimum pada setiap pangkat permutasi pada siklus tersebut. Penjelasan singkat mengenai evaluasi nilai 1 atau lebih pada posisi k pada larik $coefC$ terdapat pada Tabel 2.3.

Tabel 2.2 Konvolusi larik coefA dan coefB menjadi coefC

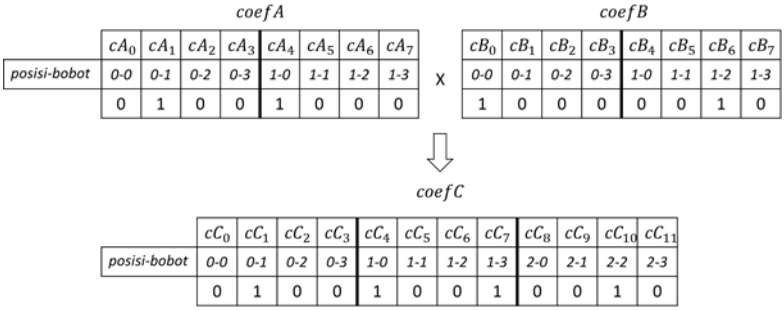
Larik	Posisi bernilai 1 atau lebih
$coefA$	$(N-1-i)*M+A_{V_i}$
$coefB$	$j*M+B_{V_j}$
$coefC$	$(j+(N-1-i))*M+A_{V_i}+B_{V_j}$

Tabel 2.3 Evaluasi nilai 1 atau lebih pada posisi k pada larik coefC

	Pangkat Permutasi		Bobot Sisi
Kondisi	$\left\lfloor\frac{k}{M}\right\rfloor\leq N-2$	$\left\lfloor\frac{k}{M}\right\rfloor\geq N-1$	
Nilai	$\left\lfloor\frac{k}{M}\right\rfloor+1$	$\left\lfloor\frac{k}{M}\right\rfloor-(N-1)$	$k\%M$

Sebagai contoh, misal terdapat siklus dengan $N=2$ dengan $A=\{0,1\}$ dan $B=\{0,2\}$. Nilai M adalah $maxA+maxB+1$ sehingga

M adalah 4. Dan ukuran larik $coefA$ dan $coefB$ adalah $N * M$ sehingga ukurannya adalah 8. Maka ukuran hasil konvolusi larik $coefC$ adalah $(2N - 1) * M$ sehingga ukurannya adalah 12. Ilustrasi konvolusi siklus tersebut ada pada Gambar 2.20.



Gambar 2.20 Ilustrasi konvolusi siklus

Pada simpul $i = 0$ dengan bobot $A_{V_0} = 0$ memberikan nilai 1 pada $coefA_4$ dan dengan bobot $B_{V_0} = 0$ memberikan nilai 1 pada $coefB_0$. Pada simpul $i = 1$ dengan bobot $A_{V_1} = 1$ memberikan nilai 1 pada $coefA_1$ dan dengan bobot $B_{V_1} = 2$ memberikan nilai 1 pada $coefB_6$. Hasil dari konvolusi $coefA$ dan $coefB$ menghasilkan $coefC$. Berdasarkan cara evaluasi pada Tabel 2.3, maka pada $coefC_1$ menyatakan terdapat bobot sisi dengan nilai 1 pada pangkat permutasi 1. Pada $coefC_4$ dan $coefC_7$ menyatakan terdapat bobot sisi dengan nilai 0 dan 3 pada pangkat permutasi 0. Sedangkan pada $coefC_{10}$ menyatakan terdapat bobot sisi dengan nilai 2 pada pangkat permutasi 1. Maka dari evaluasi tersebut dapat disimpulkan bahwa pada pangkat permutasi 0 memiliki bobot sisi maksimum sebesar 3, sedangkan pada pangkat permutasi 1 memiliki bobot sisi maksimum sebesar 2.Perlu diperhatikan bahwa hasil konvolusi $coefC$ pada posisi $(j + (N - 1 - i)) * M + A_{V_i} + B_{V_j}$ dapat bernilai lebih dari 1 karena terdapat kemungkinan

beberapa nilai i dan j yang memiliki jarak $j - i$ yang sama dan perjumlahan bobot A_{V_i} dan B_{V_j} yang sama pula. Hal ini dapat dimanfaatkan untuk melakukan optimasi dengan menggabungkan dua nilai bobot yang saling bersebelahan, dalam hal ini pada bobot genap dan ganjil, sehingga nilai M bisa menjadi $M/2$. Perubahan M menjadi $M/2$ bisa memberikan pengaruh yang cukup besar apabila N memiliki nilai yang besar, karena konvolusi menggunakan FFT yang berbasis kelipatan 2. Cara menggabungkannya adalah dengan memberikan nilai tertentu apabila bobot bernilai genap atau ganjil sehingga ketika setelah konvolusi dapat dibedakan nilai genap atau ganjil. Misalkan p dan q merupakan nilai yang dapat merepresentasikan posisi bobot pada larik $coefA$ dan $coefB$ dengan ukuran $M/2$ dan hanya merepresentasikan pada suatu pangkat permutasi V_i terhadap V_j saja. Maka posisi p merepresentasikan 2 nilai gabungan yaitu bobot $p * 2$ dan $p * 2 + 1$, sedangkan posisi q merepresentasikan bobot $q * 2$ dan $q * 2 + 1$ dalm bentuk aslinya. Maka hasil konvolusi $coefC$ dari $coefA$ dan $coefB$ pada suatu pangkat permutasi V_i terhadap V_j direpresentasikan pada $coefC$ pada posisi $p + q$ dimana merepresentasikan bobot gabungan dari $(p + q) * 2$ hingga $(p + q) * 2 + 2$. Agar dapat mengevaluasi nilai bobot yang sebenarnya, maka diperlukan nilai yang unik pada posisi p dan q pada $coefA$ dan $coefB$ yang dapat mewakili seluruh kombinasi pada posisi $p + q$ pada $coefC$. Seluruh kombinasi tersebut dapat dilihat pada Tabel 2.4.

Diketahui bahwa suatu bobot pada pada suatu pangkat permutasi dari $coefC$ dapat berakumulasi hingga N dimana setiap simpul V_i terhadap simpul V_j memiliki jumlah bobot yang sama. Sehingga dengan memberikan nilai 1 yang merepresentasikan bobot genap sedangkan nilai $N + 1$ pada bobot ganjil dapat memberikan hasil konvolusi yang unik sehingga hasil konvolusi dapat dievaluasi

dengan benar. Pada Tabel 2.5 menjelaskan pemberian nilai $coefA_p$ dan $coefB_q$ berdasarkan setiap kombinasi bobot p dan q sebenarnya disertai dengan nilai yang berakumulasi pada $coefC_{p+q}$ ketika dilakukan konvolusi.

Tabel 2.4 Seluruh kombinasi bobot p dan q pada konvolusi coefA dan coefB

No.	Bobot p sebenarnya	Bobot q sebenarnya	Nilai $coefC_{p+q}$ yang diharapkan mewakili
1	$p * 2$	$q * 2$	$(p + q) * 2$
2	$p * 2$	$q * 2 + 1$	$(p + q) * 2 + 1$
3	$p * 2 + 1$	$q * 2$	$(p + q) * 2 + 1$
4	$p * 2 + 1$	$q * 2 + 1$	$(p + q) * 2 + 2$

Tabel 2.5 Pemberian nilai pada $coefA_p$ atau $coefB_q$ berdasarkan bobot p dan q yang sebenarnya disertai akumulasi nilai $coefC_{p+q}$ setelah konvolusi

No.	Bobot p sebenarnya	Nilai $coefA_p$	Bobot q sebenarnya	Nilai $coefB_q$	Nilai $coefC_{p+q}$ berakumulasi
1	$p * 2$	1	$q * 2$	1	1
2	$p * 2$	1	$q * 2 + 1$	$N + 1$	$N + 1$
3	$p * 2 + 1$	$N + 1$	$q * 2$	1	$N + 1$
4	$p * 2 + 1$	$N + 1$	$q * 2 + 1$	$N + 1$	$(N + 1)^2$

Dengan adanya Tabel 2.5, dapat dipahami bahwa apabila hasil konvolusi $coefC$ pada posisi k bernilai 1 hingga N, maka terdapat sisi berbobot $k * 2$ pada pangkat permutasi tersebut karena terdapat $coefC_k$ sisi yang memenuhi $(p + q) * 2 = k * 2$. Sedangkan apabila $coefC_k$ bernilai $N + 1$ hingga $(N + 1) * N$, maka terdapat sisi berbobot $k * 2 + 1$ pada pangkat permutasi tersebut karena terdapat $\lfloor coefC_k/(N + 1) \rfloor$ sisi yang memenuhi $(p + q) * 2 + 1 = k * 2 + 1$. Sedangkan apabila $coefC_k$ bernilai $(N + 1)^2$ hingga $(N + 1)^2 * N$, maka terdapat sisi berbobot $k * 2 + 2$ pada pangkat permutasi tersebut karena terdapat $\lfloor coefC_k/(N + 1)^2 \rfloor$ sisi yang memenuhi $(p + q) * 2 + 2 = k * 2 + 2$. Meskipun pada $coefC_k$ bernilai $N + 1$ hingga $(N + 1) * N$ memungkinan terdapat bobot sisi yang lebih kecil seperti $k * 2$, namun hal ini bisa diabaikan karena bobot sisi yang dicari adalah yang maksimum, begitu pula dengan $coefC_k$ bernilai $(N + 1)^2$ hingga $(N + 1)^2 * N$. Sehingga evaluasi hasil konvolusi $coefC$ pada posisi k untuk mencari bobot sebenarnya pada suatu pangkat permutasi dapat disimpulkan sesuai pada Tabel 2.6.

Tabel 2.6 Evaluasi hasil konvolusi coefC pada posisi k dalam mencari bobot sebenarnya pada suatu pangkat permutasi

No.	Nilai $coefC_k$	bobot $coefC_k$ yang sebenarnya
1	$1\ hingga\ N$	$2k$
2	$N + 1\ hingga\ (N + 1) * N$	$2k + 1$
3	$(N + 1)^2\ hingga\ (N + 1)^2 * N$	$2k + 2$

Untuk menyederhanakan pemberian nilai $coefA$ dan $coefB$ dengan diketahui bobot A dan B pada suatu simpul berdasarkan Tabel 2.5, maka apabila bobot bernilai genap, beri nilai 1 pada

$coef$ yang bersangkutan pada posisi $\lfloor bobot/2 \rfloor$, sedangkan apabila ganjil, beri nilai $N+1$ pada coef yang bersangkutan pada posisi $\lfloor bobot/2 \rfloor$ seperti yang ditunjukkan pada Tabel 2.7.

Tabel 2.7 Pemberian nilai coefA atau coefB berdasarkan nilai bobot yang diketahui

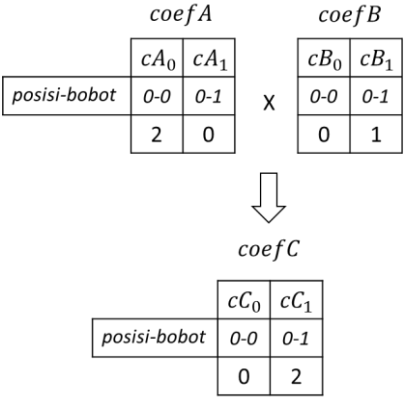
$bobot$	Nilai $coef_{\lfloor bobot/2 \rfloor}$
Genap	1
Ganjil	$N+1$

Berikut adalah contoh mencari bobot sisi V_i terhadap V_j dengan diketahui $A_{V_i} = 1$ dan $B_{V_j} = 2$. Nilai $M = \lceil (maxA + maxB + 1)/2 \rceil = 2$. Larik $coefA$ dan $coefB$ berukuran M. Karena hanya mencari 1 bobot sisi, maka $N = 1$. Ukuran larik hasil konvolusi $coefC$ adalah $(2N-1) * M$ sehingga bernilai M. Bobot $A_{V_i} = 1$, sehingga berikan nilai $N+1$ pada $coefA$ posisi $\lfloor A_{V_i}/2 \rfloor$. Sedangkan bobot $B_{V_j} = 2$, sehingga berikan nilai 1 pada $coefB$ posisi $\lfloor B_{V_j}/2 \rfloor$. Berikut pada Gambar 2.21 mengilustrasikan konvolusi dalam mencari bobot sisi V_i terhadap V_j.

Dari hasil konvolusi $coefC$ pada Gambar 2.21, berdasarkan Tabel 2.6, dengan $k = 1$ dimana $coefC_k$ bernilai 2 memenuhi persyaratan Tabel 2.6 nomor 2, maka bobot sisi V_i terhadap V_j adalah $2k + 1$ yaitu 3.

Apabila cara ini dilakukan untuk menghitung bobot sisi setiap simpul pada setiap pangkat permutasi pada suatu siklus, maka peletakan bobot simpul disesuaikan pada posisi simpul. Pemberian nilai pada $coefA$ dan $coefB$ disesuaikan dengan perpaduan metode pada Tabel 2.2 dan Tabel 2.7. Untuk simpul posisi i, maka

beri nilai berdasarkan evaluasi bobot A_{V_i} sesuai Tabel 2.7 pada $coefA$ posisi $(N-1-i)*M+\lfloor A_{V_i}/2 \rfloor$, kemudian beri nilai berdasarkan evaluasi bobot B_{V_i} pada $coefB$ posisi $i*M+\lfloor B_{V_i}/2 \rfloor$. Penjelasan ringkas pemberian nilai pada larik $coefA$ dan $coefB$ terdapat pada Tabel 2.8 dan Tabel 2.9.



Gambar 2.21 Ilustrasi konvolusi mencari bobot sisi V_i terhadap V_j

Tabel 2.8 Pemberian nilai pada larik coefA untuk setiap simpul i berdasarkan bobot A_{V_i}

Bobot A_{V_i}	Nilai $coefA_{(N-1-i)*M+\lfloor A_{V_i}/2 \rfloor}$
Genap	1
Ganjil	$N+1$

Sedangkan untuk mengevaluasi hasil konvolusi larik $coefC$ menggunakan perpaduan metode pada Tabel 2.3 dan Tabel 2.6. Untuk setiap posisi k pada larik $coefC$ dimana memiliki nilai 1

atau lebih, maka terdapat bobot sisi dengan pangkat permutasi $\left\lfloor \frac{k}{M} \right\rfloor + 1$ apabila $\left\lfloor \frac{k}{M} \right\rfloor \leq N - 2$, atau pangkat permutasi $\left\lfloor \frac{k}{M} \right\rfloor - (N - 1)$ apabila $\left\lfloor \frac{k}{M} \right\rfloor \geq N - 1$ dimana memiliki nilai bobot sisi $k\%M * 2$ apabila $1 \leq coefC_k \leq N$, atau berbobot $k\%M * 2 + 1$ apabila $N + 1 \leq coefC_k \leq (N + 1) * N$, atau berbobot $k\%M * 2 + 2$ apabila $(N + 1)^2 \leq coefC_k \leq (N + 1)^2 * N$. Dengan mengevaluasi nilai setiap elemen larik $coefC$ dimana elemen tersebut bernilai lebih dari 1, maka akan didapatkan bobot sisi maksimum pada setiap pangkat permutasi pada siklus tersebut. Penjelasan singkat evaluasi pangkat permutasi dan bobot sisi hasil konvolusi larik $coefC$ pada posisi k terdapat pada Tabel 2.10 dan Tabel 2.11.

Tabel 2.9 Pemberian nilai pada larik coefB untuk setiap simpul i berdasarkan bobot B_{V_i}

Bobot B_{V_i}	Nilai $coefB_{i*M+\lfloor B_{V_i}/2 \rfloor}$
Genap	1
Ganjil	$N + 1$

Berikut adalah contoh mencari bobot sisi maksimum setiap pangkat permutasi pada suatu siklus. Misal terdapat siklus dengan $N = 2$ dengan $A = \{0,1\}$ dan $B = \{0,2\}$. Nilai M adalah $\lceil (maxA + maxB + 1)/2 \rceil$ sehingga M adalah 2. Dan ukuran larik $coefA$ dan $coefB$ adalah $N * M$ sehingga ukurannya adalah 4. Maka ukuran hasil konvolusi larik $coefC$ adalah $(2N - 1) * M$ sehingga ukurannya adalah 8. Pemberian nilai pada larik $coefA$ dan $coefB$ dilakukan berdasarkan metode pada Tabel 2.8 dan Tabel 2.9. Pada simpul $k = 0$ dengan bobot $A_{V_0} = 0$ memberikan

nilai 1 pada $coefA_2$, sedangkan dengan bobot $B_{V_0} = 0$ memberikan nilai 1 pada $coefB_0$. Pada simpul $k = 1$ dengan bobot $A_{V_1} = 1$ memberikan nilai 3 pada $coefA_0$, sedangkan dengan bobot $B_{V_1} = 2$ memberikan nilai 1 pada $coefB_3$. Ilustrasi pemberian nilai larik $coefA$ dan $coefB$ beserta hasil konvolusi larik $coefC$ dapat dilihat pada Gambar 2.22.

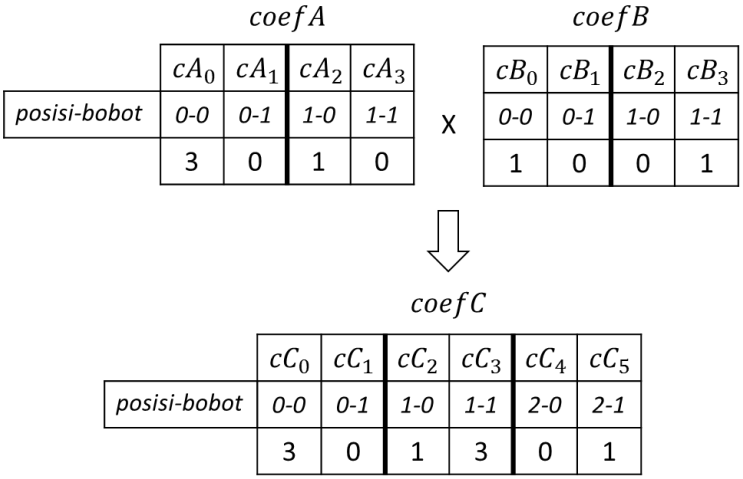
Tabel 2.10 Evaluasi pangkat permutasi larik $coefC$ pada posisi k

	Pangkat Permutasi	
Kondisi	$\left\lfloor \frac{k}{M} \right\rfloor \leq N - 2$	$\left\lfloor \frac{k}{M} \right\rfloor \geq N - 1$
Nilai	$\left\lfloor \frac{k}{M} \right\rfloor + 1$	$\left\lfloor \frac{k}{M} \right\rfloor - (N - 1)$

Tabel 2.11 Evaluasi bobot sisi larik $coefC$ pada posisi k

No.	Nilai $coefC_k$	bobot $coefC_k$ yang sebenarnya
1	$1\ hingga\ N$	$k\%M * 2$
2	$N + 1\ hingga\ (N + 1) * N$	$k\%M * 2 + 1$
3	$(N + 1)^2\ hingga\ (N + 1)^2 * N$	$k\%M * 2 + 2$

Hasil konvolusi pada Gambar 2.22 dapat dievaluasi menggunakan metode evaluasi pangkat permutasi dan bobot pada Tabel 2.10 dan Tabel 2.11.



Gambar 2.22 Ilustrasi konvolusi siklus dengan optimasi

Pada $coefC_0 = 3$ menyatakan terdapat bobot sisi bernilai 1 pada pangkat permutasi 1. Pada $coefC_2 = 1$ menyatakan terdapat bobot sisi bernilai 0 pada pangkat permutasi 0. Pada $coefC_3 = 3$ menyatakan terdapat bobot sisi bernilai 3 pada pangkat permutasi 0. Pada $coefC_5 = 1$ menyatakan terdapat bobot sisi bernilai 2 pada pangkat permutasi 1. Dari evaluasi tersebut dapat diambil kesimpulan bahwa siklus ini memiliki bobot sisi maksimum 3 pada pangkat permutasi 0 dan bobot sisi maksimum 2 pada pangkat permutasi 1. Hasil kesimpulan ini akan disimpan pada suatu larik $result$ pada posisi N dimana pada $result[N][k]$ menyimpan bobot sisi maksimum siklus tersebut pada pangkat permutasi k. Apabila terdapat evaluasi siklus sebelumnya dengan jumlah simpul yang sama, maka pemberian nilai $result[N][k]$ harus dibandingkan dengan nilai $result[N][k]$ sebelumnya untuk mendapatkan bobot yang terbesar. Untuk mengetahui terdapat siklus dengan jumlah simpul apa saja, maka dibentuk larik $cycleIndex$ yang menyimpan jumlah simpul pada setiap siklus. Sehingga apabila terdapat

evaluasi siklus dengan jumlah simpul yang belum dievaluasi sebelumnya, maka tambahkan nilai N pada larik $cycleIndex$. Hal ini dilakukan untuk memudahkan penggabungan bobot sisi maksimum pada setiap siklus pada langkah selanjutnya.

2.6.3 Penggabungan bobot sisi maksimum pada setiap siklus

Pada tahap ini, seluruh larik pada hasil evaluasi larik $result$ pada posisi $cycleIndex_j$ digabungkan untuk menampilkan bobot sisi maksimum setiap pangkat permutasi dari 0 hingga Q – 1. Penggabungan larik dapat dilakukan dengan mencari bobot terbesar dari setiap larik pada pangkat permutasi tertentu. Karena pangkat permutasi yang ingin dicari dapat melebihi jumlah simpul pada suatu siklus, maka pangkat permutasi dimodulo dengan jumlah simpul pada siklus tersebut. Misal terdapat siklus berukuran N dan ingin dicari bobot pada posisi k, maka nilai yang diambil adalah $result[N][k\%N]$. Dengan cara tersebut didapatkan nilai bobot sisi maksimum pada setiap pangkat permutasi dari seluruh siklus. Misal diketahui larik $cycleIndex = \{4, 1, 2\}$ maka terdapat sejumlah siklus yang berukuran 1, 2 dan 4. Diketahui larik $result[1] = \{2\}$, $result[2] = \{3,0\}$, dan $result[4] = \{5,1,2,1\}$. Kemudian diminta $Q = 8$ maka bobot sisi maksimum yang ingin diketahui dari pangkat permutasi 0 hingga 7. Ilustrasi pengambilan bobot sisi terbesar dapat dilihat pada Gambar 2.23.

Dari ilustrasi Gambar 2.23 didapatkan bobot sisi maksimum pada pangkat permutasi k dengan membandingkan setiap bobot pada hasil evaluasi larik $result$ untuk setiap ukuran siklus $cycleIndex_i$ pada posisi $k\%cycleIndex_i$. Misal pada $k = 3$, bobot sisi maksimum diambil dari nilai terbesar antara $result[1][0]$, $result[2][1]$, dan $result[4][3]$, sehingga bobot sisi maksimum untuk pangkat permutasi $k = 3$ adalah 2.

$$cycleIndex = \{4,1,2\}$$

Q	0	1	2	3	4	5	6	7
$result[1]$	2	2	2	2	2	2	2	2
$result[2]$	3	0	3	0	3	0	3	0
$result[4]$	5	1	2	1	5	1	2	1
ans	5	2	3	2	5	2	3	2

Gambar 2.23 Ilutrasi pengambilan bobot sisi terbesar dari larik result untuk pangkat permutasi 0 hingga $Q - 1$.

Pencarian bobot sisi maksimum untuk pangkat permutasi 0 hingga $Q - 1$ dari setiap siklus dengan jumlah simpul dari setiap siklus sebanyak N membutuhkan kompleksitas komputasi $O(Q\sqrt{N})$ dimana terdapat kisaran $\sqrt{2N}$ siklus dengan jumlah simpul yang berbeda-berbeda dimulai dari 1 hingga kisaran $\sqrt{2N}$ simpul. Banyak siklus(S) terbanyak yang didapatkan dengan $2N$ simpul dapat dihitung dengan memenuhi persamaan (2, 11) yang juga merupakan persamaan jumlah deret aritmatika sehingga apabila diambil nilai kisaran menjadi $\sqrt{2N}$ siklus.

$$N = \frac{S*(S+1)}{2} \qquad (2, 11)$$

2.7 Pembuatan Data *Generator* Untuk Uji Coba

Pembuatan data *generator* dilakukan untuk melakukan uji coba selain uji yang dilakukan pada penilaian daring SPOJ kode soal MEPPERM. Data *generator* yang dibuat disesuaikan dengan format masukan yang telah dijelaskan pada subbab 2.1.

Terdapat dua skenario yang akan dilakukan dalam melakukan pengujian, yaitu pengujian pengaruh banyak simpul dengan satu siklus terhadap waktu berjalannya *program* dan pengujian pengaruh banyak simpul dengan siklus sebanyak mungkin terhadap waktu berjalannya *program*.

Pada skenario pertama akan dibuat sejumlah uji coba dimana banyak simpul berkisaran antara 1 hingga 66000 sesuai dengan batasan jumlah simpul terbesar pada soal. Permutasi akan dibentuk sedemikian rupa agar menghasilkan satu siklus dengan seluruh simpul. Bobot A_{V_i} dan B_{V_i} diberikan secara acak berkisaran antara 0 hingga 16. Pangkat permutasi bernilai tetap yaitu 1.000.000.

Pada skenario kedua akan dibuat sejumlah uji coba dimana jumlah simpul berkisaran antara 1 hingga 66000 sesuai dengan batasan banyak simpul terbesar pada soal. Permutasi akan dibentuk sedemikian rupa agar menghasil siklus terbanyak dengan setiap siklus memiliki banyak simpul yang berbeda, yaitu kisaran $\sqrt{2N}$ siklus. Bobot A_{V_i} dan B_{V_i} diberikan secara acak berkisaran antara 0 hingga 16. Pangkat permutasi bernilai tetap yaitu 1.000.000.

(Halaman ini sengaja dikosongkan)

BAB 3
DESAIN

Pada bab ini dijelaskan desain algoritma yang digunakan dalam menyelesaikan permasalahan pada Tugas Akhir ini.

3.1 Definisi Umum Sistem

Sistem akan menerima masukan berupa banyak simpul(N), permutasi($permutation$), bobot A_{V_i} dan B_{V_i}, dan batas pangkat permutasi(Q). Kemudian sistem akan membentuk sejumlah siklus berdasarkan bentuk permutasi yang dimasukan dengan memanggil fungsi CREATE-CYCLES. Sistem melakukan perhitungan bobot sisi maksimum untuk setiap pangkat permutasi pada setiap siklus dengan memanggil fungsi EVALUATE-CYCLES. Sistem melakukan perbandingan bobot sisi maksimum dari hasil perhitungan bobot sisi maksimum setiap siklus sebelumnya untuk mendapatkan bobot sisi terbesar pada pangkat permutasi 0 hingga pangkat permutasi yang dimasukan dengan memanggil fungsi CREATE-ANSWER. Terakhir, sistem menampilkan bobot sisi terbesar dari pangkat permutasi 0 hingga pangkat permutasi yang dimasukkan. *Pseudocode* fungsi MAIN ditunjukkan pada Gambar 3.1.

3.2 Desain Algoritma

Sistem terdiri dari beberapa fungsi, yaitu fungsi INIT-FFT, FFT, INVERSE-FFT, MULTIPLY, dan CONVOLUTE yang digunakan untuk melakukan komputasi konvolusi menggunakan FFT, serta fungsi CREATE-CYCLES, EVALUATE-CYCLES, dan CREATE-ANSWER. Sistem juga memiliki satu kelas yaitu kelas $Complex$ yang digunakan untuk merepresentasikan bilangan kompleks. Pada subbab ini dijelaskan desain algoritma dari masing-masing kelas dan fungsi tersebut.

```
MAIN()
 1  input N
 2  let permutation[1..N] be a new array
 3  for i = 1 to N
 4     input permutation[i]
 5  let A[1..N] and B[1..N] be new arrays
 6  for i = 1 to N
 7     input A[i]
 8  for i = 1 to N
 9     input B[i]
10  input Q
11  (cycles,cycleN) = CREATE-CYCLES(N,permutation)
12  (result,cycleIndex) = EVALUATE-
    CYCLES(cycles,cycleN,A,B)
13  ans = CREATE-ANSWER(result,cycleIndex,Q)
14  for i = 0 to Q − 1
15     print ans[i]
16  return
```

Gambar 3.1 Pseudocode fungsi MAIN

3.2.1 Desain Kelas *Complex*

Kelas *Complex* digunakan sebagai representasi bilangan kompleks yang digunakan saat melakukan komputasi konvolusi. *Pseudocode* kelas *Complex* ditunjukkan pada Gambar 3.2.

```
Complex
 1  R   // real number
 2  I   // imaginary number
```

Gambar 3.2 Pseudocode kelas Complex

3.2.2 Desain Fungsi INIT-FFT

Fungsi INIT-FFT digunakan untuk membentuk *length* bilangan kompleks *Nth root of unity* dengan menggunakan persamaan Euler seperti yang telah dibahas pada subbab 2.3.5, yang kemudian disimpan pada larik *wFFT*. Fungsi INIT-FFT digunakan oleh

fungsi CONVOLUTE untuk menghasilkan $wFFT$ yang digunakan oleh fungsi FFT dan INVERSE-FFT. Variabel $length$ memiliki nilai kelipatan 2 karena algoritma FFT yang digunakan berbasis kelipatan 2. *Pseudocode* dari fungsi INIT-FFT ditunjukkan pada Gambar 3.3.

```
INIT-FFT(length)
1  step = 2π/length
2  let wFFT[0..length/2 − 1] be a new array of
   Complex
3  for i = 0 to length/2 − 1
4      wFFT[i].R = cos(i ∗ step)
5      wFFT[i].I = sin(i ∗ step)
6  return wFFT
```

Gambar 3.3 Pseudocode fungsi INIT-FFT

3.2.3 Desain Fungsi FFT

Fungsi FFT digunakan untuk melakukan perubahan representasi koefisien menjadi representasi titik-nilai dan sebaliknya menggunakan algoritma FFT berbasis kelipatan 2 seperti yang telah dibahas pada subbab 2.4 dan 2.5. Fungsi FFT digunakan oleh fungsi CONVOLUTE untuk mengubah dua buah larik dengan representasi koefisien menjadi representasi titik-nilai, dan juga digunakan oleh fungsi INVERSE-FFT untuk mengubah hasil perkalian larik dengan representasi titik-nilai menjadi representasi koefisien. Algoritma FFT pada *pseudocode* ini dilakukan menggunakan metode *divide and conquer* secara rekursif. Larik $coef$ yang merupakan bentuk awal diubah menjadi bentuk representasi yang diinginkan pada larik fft. Variabel $coefPos$ merupakan posisi koefisien pertama terhadap larik $coef$.Variabel $length$ merupakan ukuran larik $coef$ bernilai kelipatan 2. Variabel $step$ merupakan langkah *root of unity*. Variabel $sign$ merupakan penanda fungsi melakukan FFT atau invers FFT. Larik $wFFT$ merupakan N bilangan *Nth root of unity* dimana N adalah ukuran

larik $coef$ secara keseluruhan. *Pseudocode* dari fungsi FFT ditunjukkan pada Gambar 3.4.

```
FFT(coef, coefPos,length,step,sign,wFFT)
 1  let fft[0..length − 1] be a new array of Complex
 2  if length == 2
 3      fft[0] = coef[coefPos] + coef[coefPos + step]
 4      fft[1] = coef[coefPos] − coef[coefPos + step]
 5      return fft
 6  fft0 = FFT(coefPos,length/2,step ∗ 2,sign)
 7  fft1 = FFT(coefPos + step,length/2,step ∗ 2,sign)
 8  for i = 0 to length/2 − 1
 9      fft[i] = fft0[i] + fft1[i] ∗ wFFT[i ∗ step]^sign
10      fft[i + step] = fft0[i] − fft1[i] ∗ wFFT[i ∗ step]^sign
11  return fft
```

Gambar 3.4 Pseudocode fungsi FFT

Fungsi FFT akan dijalankan dengan metode *divide and conquer* secara rekursif dengan membagi dua larik $coef$ yang direpresentasikan menggunakan variabel $coefPos$ hingga hanya terdapat 2 nilai koefisien dimana $length = 2$. Pada fungsi FFT dengan $length = 2$, maka perhitungan FFT dapat dikomputasi secara langsung berdasarkan nilai $coef[coefPos]$ dan $coef[coefPos + step]$ dengan diketahui bahwa nilai $coefPos$ merupakan posisi larik $coef$ yang telah dilakukan *bit-reverse* secara rekursif menggunakan nilai $step$, dan diketahui bahwa pada fungsi FFT dengan $length = 2$ terdapat dua nilai koefisien dimana kedua posisi koefisien pada larik $coef$ memiliki jarak sebesar $step$ atau $lengthCoef/2$ dimana $lengthCoef$ adalah ukuran larik $coef$. Hasil fungsi FFT dengan $length = 2$ disimpan pada larik fft yang akan digunakan untuk mengomputasi FFT dengan $length$ yang lebih besar. Pada fungsi FFT dengan $length > 2$, komputasi FFT dilakukan mengunakan operasi kupu-kupu(*butterfly operation)* dari dua hasil komputasi FFT yang dilakukan secara rekursif sebelumnya.

3.2.4 Desain Fungsi INVERSE-FFT

Fungsi INVERSE-FFT digunakan untuk melakukan komputasi inverse FFT dengan memanggil fungsi FFT dengan parameter $sign = -1$. Fungsi INVERSE-FFT digunakan oleh fungsi CONVOLUTE untuk mengubah larik dengan representasi titik-nilai menjadi representasi koefisien. Nilai setiap elemen dari larik $coef$ hasil fungsi FFT dibagi dengan ukuran larik koefisiennya sesuai dengan rumus invers FFT pada persamaan (2, 8) pada subbab 2.4. Nilai hasil tiap koefisien juga dilakukan pembulatan untuk mengoreksi kesalahan pada presisi. Larik fft merupakan larik yang ingin diubah menjadi representasi koefisien. Larik *wFFT* merupakan *N* bilangan *Nth root of unity* dimana *N* adalah ukuran larik. Larik $coef$ merupakan hasil dari perubahan tersebut. Variabel $length$ merupakan ukuran larik yang ingin diubah. *Pseudocode* fungsi INVERSE-FFT ditunjukkan pada Gambar 3.5.

```
INVERSE-FFT(fft,length,wFFT)
1  coef = FFT(fft,length,1,-1,wFFT)
2  for i = 0 to length − 1
3      coef[i].R = round(coef[i].R/length)
4  return coef
```

Gambar 3.5 Pseudocode fungsi INVERSE-FFT

3.2.5 Desain Fungsi MULTIPLY

Fungsi MULTIPLY digunakan untuk melakukan perkalian antar dua larik dengan representasi titik-nilai. Fungsi MULTIPLY digunakan oleh fungsi CONVOLUTE untuk melakukan perkalian dua buah larik dengan representasi titik-nilai. Larik $fftA$ dikalikan dengan larik $fftB$ menghasilkan larik $fftC$. Variabel $length$ merupakan ukuran larik. *Pseudocode* fungsi MULTIPLY ditunjukkan pada Gambar 3.6.

```
MULTIPLY(fftA,fftB,length)
1  let fftC[0..length − 1] be a new array of Complex
2  for i = 0 to length − 1
       fftA[i] = fftA[i] ∗ fftB[i]
4  return fftC
```

Gambar 3.6 Pseudocode fungsi MULTIPLY

3.2.6 Desain Fungsi CONVOLUTE

Fungsi CONVOLUTE digunakan untuk melakukan konvolusi antar dua larik dengan representasi koefisien seperti yang telah dibahas pada subbab 2.3.4. Fungsi ini mengonvolusi larik $coefA$ dan $coefB$ dengan ukuran $length$ dengan memanggil fungsi INIT-FFT, FFT, MULTIPLY, dan INVERSE-FFT. Fungsi CONVOLUTE digunakan oleh fungsi EVALUATE-CYCLES dalam mengomputasi bobot sisi maksimum pada setiap pangkat permutasi pada suatu siklus. *Pseudocode* fungsi CONVOLUTE ditunjukkan pada Gambar 3.7.

```
CONVOLUTE(coefA,coefB,length)
1  wFFT = INIT-FFT(length)
2  fftA = FFT(coefA,length,1,1)
3  fftB = FFT(coefB,length,1,1)
4  fftC = MULTIPLY(fftA,fftB,length)
5  coefC = INVERSE-FFT(fftC,length,wFFT)
6  return coefC
```

Gambar 3.7 Pseudocode fungsi CONVOLUTE

3.2.7 Desain Fungsi CREATE-CYCLES

Fugnsi CREATE-CYCLES digunakan untuk membentuk sejumlah siklus yang merupakan permutasi siklus berdasarkan permutasi dua baris yang dimasukkan seperti yang telah dibahas pada subbab 2.6.1. Larik $permutation$ merupakan larik permutasi dengan notasi dua baris. Larik $cycles$ merupakan larik yang setiap elemennya terdiri dari larik dinamis(*vector*) yang digunakan untuk

menyimpan sejumlah siklus dengan sejumlah simpul dengan urutan tertentu. Variabel N merupakan banyak simpul. Variabel $cycleN$ merupakan banyak siklus yang terbentuk. Pseudocode fungsi CREATE-CYCLES ditunjukkan pada Gambar 3.8.

```
CREATE-CYCLES(permutation,N)
 1  let used[1..N] be a new array of boolean which
    each element is false
 2  let cycles[0..N − 1] be a new array of dynamic
    array(vector)
 3  cycleN = 0
 4  for i = 1 to N
 5     if used[i] == false
 6        now = i
 7        while used[now] == false
 8           cycles[cycleN].push(now)
 9           used[now] = true
10           now = permutation[now]
11           cycleN = cycleN + 1
12  return (cycles,cycleN)
```

Gambar 3.8 Pseudocode fungsi CREATE-CYCLES

Pertama untuk mengetahui apakah simpul telah menjadi bagian dari siklus yang ada, maka digunakan larik $used$. Kemudian lakukan iterasi dari simpul 1 hingga N dimana pada simpul i dilakukan pengecekan nilai $used[i]$. Apabila $used[i] = false$, maka lakukan *traversal* dimulai dari simpul i menuju $permutation[i]$ hingga bertemu dengan simpul i kembali untuk membentuk siklus baru. Setiap simpul yang dikunjungi secara berurutan saat melakukan *traversal* menjadi bagian dari siklus baru.

3.2.8 Desain Fungsi EVALUATE-CYCLES

Fungsi EVALUATE-CYCLES digunakan untuk menghitung bobot sisi maksimum pada setiap pangkat permutasi pada suatu

siklus seperti yang telah dibahas pada subbab 2.6.2. Larik $cycles$ merupakan larik yang menyimpan siklus sebanyak $cycleN$. Larik A dan B menyimpan bobot setiap simpul. Hasil fungsi EVALUTE-CYCLES adalah larik $cycleIndex$ yang menyimpan seluruh ukuran siklus dan larik $result$ yang menyimpan $cycleIndex.length$ siklus dimana pada $result[cycleIndex[i]]$ menyimpan bobot sisi maksimum untuk setiap pangkat permutasi 0 hingga $cycleIndex[i]-1$ pada siklus dengan ukuran $cycleIndex[i]$. *Pseudocode* fungsi EVALUATE-CYCLES ditunjukkan pada Gambar 3.9 dan Gambar 3.10.

```
EVALUATE-CYCLES(cycles,cycleN,A,B)
 1  let result[0..N − 1] be a new array of dynamic
    array(vector)
 2  let cycleIndex be a dynamic array(vector)
 3  for i = 0 to cycleN
 4      cycle = cycles[i]
 5      outMax = 0
 6      inMax = 0
 7      for j = 0 to cycle.length − 1
 8          outMax = max(outMax, A[cycle[j]])
 9          inMax = max(inMax, B[cycle[j]])
10      edgeLength = ceil((outMax + inMax + 1)/2)
11      convolutionSize = (2 ∗ cycle.length − 1) ∗
    edgeLength
12      lengthFFT = 1
13      while lengthFFT < convolutionSize
14          lengthFFT = lengthFFT ∗ 2
15      let coefA[0..lengthFFT − 1] and
    coefB[0..lengthFFT − 1] be new arrays of Complex
    which each elements is zero
16      for j = 0 to cycle.length − 1
17          outValue = A[cycle[j]]
18          outPos = (cycle.length − 1 − j) ∗ edgeLength +
    floor(outValue/2)
```

Gambar 3.9 Pseudocode fungsi EVALUATE-CYCLES(1)

```
19          inValue = B[cycle[j]]
20          inPos = j * edgeLength + floor(inValue/2)
21          if outValue mod 2 == 1
22              coefA[outPos].R = cycle.length + 1
23          else coefA[outPos].R = 1
24          if inValue mod 2 == 1
25              coefB[inPos].R = cycle.length + 1
26          else coefB[inPos].R = 1
27      coefC = CONVOLUTE(coefA,coefB,lengthFFT)
28      if result[cycle.length].length == 0
29          let result[cycle.length] be a new array with
    cycle.length elements which each elements is 0
30          cycleIndex.push(cycle.length)
31      boundary1 = cycle.length + 1
32      boundary2 = boundary1 * boundary1
33      for j = 0 to cycle.length - 1
34          now1 = j * edgeLength - 1
35          end1 = (j - 1) * edgeLength
36          now2 = (cycle.length + j) * edgeLength - 1
37          end2 = (cycle.length + j - 1) * edgeLength
38          value = edgeLength * 2 - 2
39          while now1 ≥ end and value + 2 >
    result[cycle.length][j]
40              if (j and coefC[now1].R) or
    coefA[now2].R
41                  if (j and coefC[now1].R ≥ boundary2)
    or coefA[now2].R ≥ boundary2
42                      value = value + 2
43                  else if (j and coefC[now1].R ≥
    boundary1) or coefA[now2].R ≥ boundary1
44                      value = value + 1
45                  break
46              value = value - 2
47              now1 = now1 - 1
48              now2 = now2 - 1
49  return (result,cycleIndex)
```

Gambar 3.10 Pseudocode fungsi EVALUATE-CYCLES(2)

Fungsi EVALUATE-CYCLES mengevaluasi setiap siklus pada satu waktu. Pada awalnya dicari bobot A dan B terbesar dari setiap simpul pada siklus tersebut dan disimpan pada $outMax$ dan $inMax$. Dari kedua variabel tersebut dapat diketahui perjumlahan bobot terbesar sehingga dapat ditentukan ukuran bobot sisi pada setiap simpul untuk pada saat dilakukan konvolusi, yaitu $(outMax + inMax + 2)/2$ disimpan pada $edgeLength$ dan dapat ditentukan ukuran larik untuk konvolusi sebesar $(2 * cycle.length - 1) * edgeLength$ disimpan pada $convolutionSize$. Karena algoritma FFT yang dipakai berbasis kelipatan 2, maka dicari ukuran larik kelipatan 2 yang lebih besar daripada $convolutionSize$ dan disimpan pada $lengthFFT$. Kemudian bentuk larik $coefA$ dan $coefB$ dengan ukuran $lengthFFT$ dan beri nilai yang merepresentasikan bobot setiap simpul sesuai dengan aturan pada Tabel 2.8 dan Tabel 2.9. Lakukan konvolusi antara $coefA$ dan $coefB$ dengan memanggil fungi CONVOLUTE yang menghasilkan $coefC$ yang kemudian dievaluasi untuk mendapatkan bobot sisi maksimum pada setiap pangkat permutasi sesuai dengan aturan pada Tabel 2.10 dan Tabel 2.11 dimulai dari bobot yang lebih besar. Pada saat evaluasi juga perlu dilakukan perbandingan nilai bobot sisi pada setiap pangkat permutasi dengan hasil evaluasi siklus apabila terdapat evaluasi siklus dengan jumlah simpul yang sama yang telah dilakukan sebelumnya.

3.2.9 Desain Fungsi CREATE-ANSWER

Fungsi CREATE-ANSWER digunakan untuk mencari bobot sisi maksimum pada pangkat permutasi 0 hingga $Q - 1$ dengan membandingkan hasil evaluasi setiap siklus. Larik $result$ merupakan larik yang menyimpan hasil evaluasi siklus pada fungsi EVALUATE-CYCLES. Larik $cycleIndex$ menyimpan seluruh ukuran siklus yang telah dievaluasi. Variabel Q merupakan batas pangkat permutasi. Larik ans merupakan hasil dari fungsi ini yang

menyimpan bobot sisi maksimum dari seluruh siklus pada pangkat permutasi 0 hingga $Q-1$. *Pseudocode* fungsi CREATE-ANSWER ditunjukkan pada Gambar 3.11.

```
CREATE-ANSWER(result,cycleIndex,Q)
 1  let ans[0..Q − 1] be a new array which each
    element is 0
 2  for i = 0 to cycleIndex.length
 3     pos = 0
 4     cycleLength = cycleIndex[i]
 5     for j = 0 to Q − 1
 6        ans[j] = max(ans[j], result[cycleLength][pos]
 7        pos = pos + 1
 8        if pos ≥ cycleLength
 9           pos = 0
10  return ans
```

Gambar 3.11 Pseudocode fungsi CREATE-ANSWER

Fungsi CREATE-ANSWER mendapatkan bobot sisi maksimum pada pangkat permutasi 0 hingga $Q-1$ dengan membandingkan setiap bobot pada pangkat permutasi 0 hingga $Q-1$ pada setiap siklus dengan bobot terbesar sementara yang disimpan pada larik ans.

3.3 Desain Data *Generator*

Data *generator* digunakan untuk membuat kasus uji coba sesuai dengan format masukkan untuk menguji pengaruh pertumbuhan variabel tertentu terhadap pertumbuhan waktu sesuai dengan skenario yang telah dijelaskan pada subbab 2.7. Pada skenario pertama, uji coba dilakukan untuk mengetahui pengaruh jumlah simpul pada satu siklus. Pada skenario kedua, uji coba dilakukan utuk mengetahui pengaruh jumlah simpul dengan membuat jumlah siklus sebanyak mungkin dengan banyak simpul yang berbeda-beda. *Pseudocode* dari data *generator* skenario 1 ditunjukkan pada Gambar 3.12 dan Gambar 3.13.

```
GENERATE-1()
 1  input N
 2  print N
 3  Q = 1000000
 4  for i = 1 to N
 5     if i < N
 6        print i + 1
 7     else print 1
 8  for i = 1 to N
 9     print random() mod 17
10  for i = 1 to N
11     print random() mod 17
12  print Q
```

Gambar 3.12 Pseudocode data generator skenario 1

```
GENERATE-2()
 1  input N
 2  print N
 3  Q = 1000000
 4  cycleLength = 1
 5  startPos = 1
 6  pos = 1
 7  for i = 1 to N
 8     if pos == cycleLength
 9        print startPos
10        pos = 1
11        startPos = i + 1
12     else
13        if i == N
14           print startPos
15        else print i + 1
16        pos = pos + 1
17  for i = 1 to N
18     print random() mod 17
19  for i = 1 to N
20     print random() mod 17
21  print Q
```

Gambar 3.13 Pseudocode data generator skenario 2

BAB 4
IMPLEMENTASI

Pada bab ini dijelaskan implementasi sesuai dengan desain algoritma yang telah ditentukan sebelumnya.

4.1 Lingkungan Implementasi

Lingkungan uji coba yang digunakan adalah sebagai berikut:

1. Perangkat Keras:
   Processor: Intel® Core™ i5 2430M @ 2.40 GHz x 2
   RAM: 4.00 GB
2. Perangkat Lunak:
   Operating System: Windows 10 Pro 64 -bit
   IDE: Orwell Bloodshed Dev-C++ 5.11
   Compiler : g++ (TDM-GCC 4.9.2 32-bit)

4.2 Penggunaan *Library*, Konstanta, dan Variabel Global

Pada subbab ini akan dibahas penggunaan *library*, konstanta dan variabel global yang digunakan dalam sistem. Potongan kode yang menyatakan penggunaan *library* ditunjukkan pada Kode Sumber 4.1.

```
1  #include <cstdio>
2  #include <cmath>
3  #include <vector>
4  using namespace std;
5  #define MAXFFT 1 << 22
6  #define MAXN 66000
7  #define MAXQ 1000000
```

Kode Sumber 4.1 Potongan kode penggunaan library

Pada Kode Sumber 4.1, terdapat tiga *library* yang digunakan yaitu *cstdio*, *cmath*, dan *vector*. Dan definisikan konstanta MAXFFT yaitu 2^{22} yang menjadi batas ukuran larik terbesar untuk

melakukan algoritma FFT dan konstanta MAXN dan MAXQ yaitu 66000 dan 1000000 yang menjadi banyak simpul dan batas pangkat permutasi terbesar sesuai dengan batasan pada subbab 2.1.

Pada Tabel 4.1 dan Tabel 4.2, terdapat daftar variabel global yang digunakan serta penjelasan pengunaan pada implementasi program.

Tabel 4.1 Daftar variabel global(1)

No	Nama Variabel	Tipe	Penjelasan
1	$cycleN$	int	Jumlah siklus
2	$permutation$	$int[]$	Permutasi dua baris dari masukkan
3	A	$int[]$	Menyimpan bobot A pada setiap simpul dari masukkan
4	B	$int[]$	Menyimpan bobot B pada setiap simpul dari masukkan
5	ans	$int[]$	Menyimpan bobot sisi maksimum dari pangkat permutasi 0 hingga batas pangkat permutasi yang dimasukkan
6	$used$	$bool[]$	Digunakan untuk mengetahui apakah suatu simpul telah menjadi bagian dari suatu siklus
7	$cycleIndex$	$vector < int >$	Menyimpan setiap ukuran siklus yang berbeda
8	$cycles$	$vector < int > []$	Menyimpan seluruh siklus dimana setiap siklus menyimpan simpul secara berurutan

Tabel 4.2 Daftar variabel global(2)

No	Nama Variabel	Tipe	Penjelasan
9	$result$	$vector < int >$ []	Menyimpan bobot sisi maksimum pada setiap pangkat permutasi pada setiap siklus
10	$coefA$	$Complex$[]	Menyimpan hasil pemberian nilai berdasarkan bobot A pada setiap simpul pada suatu siklus
11	$coefB$	$Complex$[]	Menyimpan hasil pemberian nilai berdasarkan bobot B pada setiap simpul pada suatu siklus
12	$wFFT$	$Complex$[]	Menyimpan *Nth root of unity* untuk digunakan saat menjalankan algoritma FFT
13	$fftA$	$Complex$[]	Menyimpan hasil FFT larik $coefA$
14	$fftB$	$Complex$[]	Menyimpan hasil FFT larik $coefB$

Potongan kode yang menyatakan pendeklarasian variable global sesuai yang ditentukan pada Tabel 4.1 ditunjukkan pada Kode Sumber 4.2.

4.3 Implementasi Fungsi MAIN

Fungsi MAIN diimplementasikan sesuai dengan desain fungsi MAIN pada subbab Gambar 3.1 yang ditunjukkan pada Kode Sumber 4.3.

```
1  int cycleN;
2  int permutation[MAXN + 1], A[MAXN + 1], B[MAXN
   + 1], ans[MAXQ + 1];
3  bool used[MAXN + 1];
4  vector<int> cycleIndex;
5  vector<int> cycles[MAXN + 1];
6  vector<int> result[MAXN + 1];
7  Complex coefA[MAXFFT], coefB[MAXFFT],
   wFFT[MAXFFT], fftA[MAXFFT], fftB[MAXFFT];
```

Kode Sumber 4.2 Potongan kode deklarasi variabel global

```
 1  int main(){
 2     int N, Q, input;
 3     scanf("%d", &N);
 4     for(int i = 1; i <= N; i++){
 5        scanf("%d", &permutation[i]);
 6     }
 7     for(int i = 1; i <= N; i++){
 8        scanf("%d", &A[i]);
 9     }
10     for(int i = 1; i <= N; i++){
11        scanf("%d", &B[i]);
12     }
13     scanf("%d", &Q);
14     createCycles(N);
15     evaluateCycles();
16     createAnswer(Q);
17     for(int i = 0; i < Q; i++){
18        if(i){
19           printf(" ");
20        }
21        printf("%d", ans[i]);
22     }
23     printf("\n");
24     return 0;
25  }
```

Kode Sumber 4.3 Potongan kode fungsi MAIN

4.4 Implementasi Struct *Complex*

Struct *Complex* merupakan implementasi dari desain kelas *Complex* pada subbab 3.2.1 yang ditunjukkan pada Kode Sumber 4.4.

```
1  typedef struct{
2     double R, I;
3  } Complex;
```

Kode Sumber 4.4 Potongan kode struct Complex

4.5 Implementasi Fungsi INIT-FFT

Fungsi INIT-FFT diimplementasikan sesuai dengan desain fungsi INIT-FFT pada subbab 3.2.2 yang ditunjukkan pada Kode Sumber 4.5.

```
1  void initFFT(int length){
2     double step = 2.0 * M_PI / length;
3     for(int i = 0; i * 2 < length; i++){
4        wFFT[i].R = cos(step * i);
5        wFFT[i].I = sin(step * i);
6     }
7  }
```

Kode Sumber 4.5 Potongan kode fungsi INIT-FFT

4.6 Implementasi Fungsi FFT

Fungsi FFT diimplementasikan sesuai desain fungsi FFT pada subbab 3.2.3 berdasarkan pembahasan pada subbab 2.4 dan 2.5 yang ditunjukkan pada Kode Sumber 4.6.

4.7 Implementasi Fungsi INVERSE-FFT

Fungsi INVERSE-FFT diimplementasikan sesusai dengan desain fungsi INVERSE-FFT pada subbab 3.2.4 yang ditunjukkan pada Kode Sumber 4.7.

```
 1  void  FFT(Complex  *coef,  Complex  *fft,  int
    length, int step, int sign){
 2     if(length == 2){
 3        fft[0].R = coef[0].R + coef[step].R;
 4        fft[0].I = coef[0].I + coef[step].I;
 5        fft[1].R = coef[0].R - coef[step].R;
 6        fft[1].I = coef[0].I - coef[step].I;
 7        return;
 8     }
 9     Complex *fft0 = fft, *fft1 = fft + (length
    >> 1);
10     FFT(coef, fft0, length >> 1, step << 1,
    sign);
11     FFT(coef + step, fft1, length >> 1, step <<
    1, sign);
12     for(int i = 0; i < length >> 1; i++){
13        Complex tmp;
14        tmp.R = wFFT[i * step].R * fft1[i].R -
    sign * wFFT[i * step].I * fft1[i].I;
15        tmp.I = wFFT[i * step].R * fft1[i].I +
    sign * wFFT[i * step].I * fft1[i].R
16        fft1[i].R = fft0[i].R - tmp.R;
17        fft1[i].I = fft0[i].I - tmp.I;
18        fft0[i].R += tmp.R;
19        fft0[i].I += tmp.I;
20     }
21  }
```

Kode Sumber 4.6 Potongan kode fungsi FFT

```
 1  void inverseFFT(Complex *fft, Complex *coef, int
    length){
 2     double inFFT = 1.0 / length;
 3     FFT(fft, coef, length, 1, -1);
 4     for(int i = 0; i < length; i++){
 5        coef[i].R = floor(coef[i].R * inFFT +
    0.5);
 6     }
 7  }
```

Kode Sumber 4.7 Potongan kode fungsi INVERSE-FFT

4.8 Implementasi Fungsi MULTIPLY

Fungsi MULTIPLY dimplementasikan sesuai dengan desain fungsi MULTIPLY pada subbab 3.2.5 yang ditunjukkan pada Kode Sumber 4.8.

```
1  void multiply(Complex *fftA, Complex *fftB, int
   length){
2     Complex temp;
3     for(int i = 0; i < length; i++){
4        temp.R = fftA[i].R * fftB[i].R -
   fftA[i].I * fftB[i].I;
5        temp.I = fftA[i].R * fftB[i].I +
   fftA[i].I * fftB[i].R;
6        fftA[i].R = temp.R;
7        fftA[i].I = temp.I;
8     }
9  }
```

Kode Sumber 4.8 Potongan kode fungsi MULTIPLY

4.9 Implementasi Fungsi CONVOLUTE

Fungsi CONVOLUTE diimplementasikan sesuai dengan desain fungsi CONVOLUTE pada subbab 3.2.6 yang ditunjukkan pada Kode Sumber 4.9.

```
1  void convolute(Complex *coefA, Complex *coefB,
   int length){
2     initFFT(length);
3     FFT(coefA, fftA, length, 1, 1);
4     FFT(coefB, fftB, length, 1, 1);
5     multiply(fftA, fftB, length);
6     inverseFFT(fftA, coefA, length);
7  }
```

Kode Sumber 4.9 Potongan kode fungsi CONVOLUTE

4.10 Implementasi Fungsi CREATE-CYCLES

Fungsi CREATE-CYCLES diimplementasikan sesusai dengan desain fungsi CREATE-CYCLES pada subbab 3.2.7 berdasarkan pembahasan pada subbab 2.6.1 yang ditunjukkan pada Kode Sumber 4.10.

```
 1  void createCycles(int N){
 2     for(int i = 1; i <= N; i++){
 3        if(used[i]){
 4           continue;
 5        }
 6        int now = i;
 7        while(!used[now]){
 8           cycles[cycleN].push_back(now);
 9           used[now] = true;
10           now = permutation[now];
11        }
12        cycleN++;
13     }
14  }
```

Kode Sumber 4.10 Potongan kode Fungsi CREATE-CYCLES

4.11 Implementasi Fungsi EVALUATE-CYCLES

Fungsi EVALUATE-CYCLES diimplementasikan sesuai dengan desain fungsi EVALUATE-CYCLES pada subbab 3.2.8 berdasarkan pembahasan pada subbab 2.6.2 yang ditunjukkan pada Kode Sumber 4.11, Kode Sumber 4.12, dan Kode Sumber 4.13.

```
 1  void evaluateCycles(){
 2     for(int i = 0;i < cycleN; i++){
 3        int outMax = 0, inMax = 0;
 4        for(int j = 0; j < cycles[i].size(); j++){
 5           outMax = max(outMax, A[cycles[i][j]]);
 6           inMax = max(inMax, B[cycles[i][j]]);
 7        }
```

Kode Sumber 4.11 Potongan kode fungsi EVALUATE-CYCLES(1)

```
 8        int edgeLength = (outMax + inMax + 2) >>
    1;
 9        int convolutionSize = ((cycles[i].size()
    << 1) - 1) * edgeLength;
10        int lengthFFT = 1;
11        while(lengthFFT < convolutionSize){
12           lengthFFT <<= 1;
13        }
14        for(int j = 0; j < lengthFFT; j++){
15           coefA[j].R = coefA[j].I = coefB[j].R =
    coefB[j].I = 0.0;
16        }
17        for(int j = 0; j < cycles[i].size();
    j++){
18           int outValue = A[cycles[i][j]],
    inValue = B[cycles[i][j]];
19           int outPos = (cycles[i].size() - 1 -
    j) * edgeLength + (outValue >> 1);
20           int inPos = j * edgeLength + (inValue
    >> 1);
21           if(outValue & 1){
22              coefA[outPos].R = cycles[i].size()
    + 1;
23           } else {
24              coefA[outPos].R = 1;
25           }
26           if(inValue & 1){
27              coefB[inPos].R = cycles[i].size() +
    1;
28           } else {
29              coefB[inPos].R = 1;
30           }
31        }
32        convolute(coefA, coefB, lengthFFT);
33        if(result[cycles[i].size()].empty()){
34           result[cycles[i].size()] =
    vector<int>(cycles[i].size(), 0);
```

Kode Sumber 4.12 Potongan kode fungsi EVALUATE-CYCLES(2)

```
35          cycleIndex.push_back(
    cycles[i].size());
36       }
37       int boundary1 = cycles[i].size() + 1;
38       int boundary2 = boundary1 * boundary1;
39       for(int j = 0;j < cycles[i].size(); j++){
40          int now1 = j * edgeLength - 1;
41          int end1 = (j - 1) * edgeLength;
42          int now2 = (cycles[i].size() + j) *
    edgeLength - 1;
43          int end2 = (cycles[i].size() + j - 1)
    * edgeLength;
44          int value = (edgeLength << 1) - 2;
45          while(now1 >= end1 && value + 2 >
    result[cycles[i].size()][j]){
46             if((j && coefA[now1].R) ||
    coefA[now2].R){
47                if((j && coefA[now1].R >=
    boundary2) || coefA[now2].R >= boundary2 ){
48                   value += 2;
49                } else if((j && coefA[now1].R >=
    boundary1) || coefA[now2].R >= boundary1){
50                   value += 1;
51                }
52                result[cycles[i].size()][j] =
    max(result[cycles[i].size()][j], value);
53                break;
54             }
55             value -= 2;
56             now1--;
57             now2--;
58          }
59       }
60    }
61 }
```

Kode Sumber 4.13 Potongan kode fungsi EVALUATE-CYCLES(3)

4.12 Implementasi Fungsi CREATE-ANSWER

Fungsi CREATE-ANSWER diimplementasikan sesuai dengan desain fungsi CREATE-ANSWER pada subbab 3.2.9 berdasarkan pembahasan pada subbab 2.6.3 yang ditunjukkan pada Kode Sumber 4.14.

```
 1  void createAnswer(int Q){
 2     for(int i = 0; i < cycleIndex.size(); i++){
 3        int pos = 0;
 4        for(int j = 0; j < Q; j++){
 5           ans[j] = max(ans[j],
    result[cycleIndex[i]][pos++]);
 6           if(pos >= cycleIndex[i]){
 7              pos = 0;
 8           }
 9        }
10     }
11  }
```

Kode Sumber 4.14 Potongan kode fungsi CREATE-ANSWER

4.13 Implementasi Data *Generator*

Terdapat dua skenario pada desain data *generator* pada subbab 3.3. Pada skenario pertama mengukur pertumbuhan banyak simpul terhadap pertumbuhan waktu dimana hanya terdapat satu siklus. Implementasi data *generator* skenario pertama berdasarkan *pseudocode* pada Gambar 3.12 ditunjukkan pada Kode Sumber 4.15.

Pada skenario kedua mengukur pertumbuhan banyak simpul terhadap pertumbuhan waktu dimana dibentuk siklus sebanyak mungkin dengan ukuran siklus yang berbeda-beda. Implementasi data *generator* skenario kedua berdasarkan *pseudocode* pada Gambar 3.13 ditunjukan pada Kode Sumber 4.16.

```
 1  #include <cstdio>
 2  #include <cstdlib>
 3  #include <ctime>
 4  int main(){
 5     srand(time(NULL));
 6     int n;
 7     int q = 1000000;
 8     scanf("%d", &n);
 9     printf("%d\n", n);
10     for(int i = 1; i <= n; i++){
11        if(i < n){
12           printf("%d ", i + 1);
13        } else {
14           printf("1\n");
15        }
16     }
17     for(int i = 0; i < n; i++){
18        printf("%d ", rand() % 17);
19     }
20     printf("\n");
21     for(int i = 0; i < n; i++){
22        printf("%d ", rand() % 17);
23     }
24     printf("\n%d\n", q);
25     return 0;
26  }
```

Kode Sumber 4.15 Implementasi data generator skenario pertama

Pada implementasi data *generator* scenario pertama dan kedua menggunakan empat *library* yaitu *cstdio*, *cstdlib*, dan *ctime* yang merupakan *library* standar bahasa *C*. Fungsi *srand* yang terdapat pada *library ctime* digunakan untuk menghasilkan nilai-nilai acak dari fungsi *rand* yang berbeda-berbeda setiap kali program dijalankan. Fungsi *time* digunakan sebagai nilai *seed* untuk fungsi *srand* karena setiap kali memanggil fungsi *time* akan menghasilkan nilai yang berbeda-beda.

```
 1  #include <cstdio>
 2  #include <cstdlib>
 3  #include <ctime>
 4  int main(){
 5     srand(time(NULL));
 6     int N;
 7     int Q = 1000000;
 8     scanf("%d", &N);
 9     printf("%d\n", N);
10     int cycleSize = 1, startPos = 1, pos = 1;
11     for(int i = 1; i <= N; i++){
12        if(pos == cycleSize){
13           printf("%d ", startPos);
14           cycleSize++;
15           pos = 1;
16           startPos = i + 1;
17        } else {
18           if(i == N){
19              printf("%d ", startPos);
20           } else {
21              printf("%d ", i + 1);
22           }
23           pos++;
24        }
25     }
26     printf("\n");
27     for(int i = 0; i < N; i++){
28        printf("%d ", rand() % 17);
29     }
30     printf("\n");
31     for(int i = 0; i < N; i++){
32        printf("%d ", rand() % 17);
33     }
34     printf("\n%d\n", Q);
35     return 0;
36  }
```

Kode Sumber 4.16 Implementasi data generator skenario kedua

(Halaman ini sengaja dikosongkan)

BAB 5
UJI COBA DAN EVALUASI

Pada bab ini dijelaskan tentang uji coba dan evaluasi dari implementasi yang telah dilakukan pada BAB 4.

5.1 Lingkungan Uji Coba

Lingkungan uji coba yang digunakan adalah salah satu sistem yang digunakan situs penilaian daring SPOJ, yaitu kluster *Cube* dengan spesifikasi sebagai berikut:

1. Perangkat Keras:
   Processor : Intel Pentium G860 3GHz
   RAM: 1536 MB
2. Perangkat Lunak:
   Compiler: g++ 4.3.2

5.2 Skenario Uji Coba

Pada subbab ini akan dijelaskan uji coba yang dilakukan. Terdapat dua uji coba yang dilakukan yaitu uji coba kebenaran dan uji coba kinerja.

5.2.1 Uji Coba Kebenaran

Uji coba kebenaran yang dilakukan adalah menganalisis kasus uji coba sederhana dengan langkah-langkah yang telah dijelaskan pada subbab 2.6 untuk memvalidasi kebenaran hasil keluaran program sesuai dengan hasil keluaran yang diharapkan, dan mengirimkan kode sumber ke situs penilaian daring SPOJ. Permasalahan yang diselesaikan adalah Maximum Edge of Powers of Permutation dengan kode soal MEPPERM.

Kasus uji coba yang digunakan sebagai masukkan uji coba sederhana dalam menganalisis uji coba kebenaran adalah data yang dihasilkan oleh data *generator* skenario 2 yang telah dibahas pada

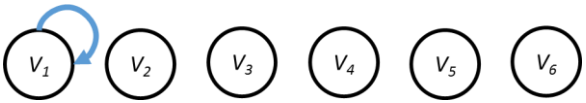
subbab 2.7. Kasus uji coba yang dibentuk terdiri dari 6 simpul dengan bobot *A* dan *B* acak. Pada uji coba skenario 2 menghasilkan siklus sebanyak mungkin dengan ukuran simpul berbeda-beda sehingga pada kasus uji coba ini membentuk 3 siklus yang masing-masing terdiri dari 1, 2, dan 3 simpul. Sebagai kasus uji coba, beberapa nilai pada data *generator* dibatasi seperti nilai bobot *A* dan *B* dibatasi hingga maksimal 4, sedangkan batas pangkat permutasi dibatasi sehingga hanya bernilai 10 untuk menghindari hasil yang berulang. Contoh kasus uji coba dari data *generator* skenario 2 yang telah dibatasi ditunjukkan pada Gambar 5.1.

```
1  6
2  1 3 2 5 6 4
3  3 0 4 1 4 2
4  2 4 0 0 2 3
5  10
```

Gambar 5.1 Contoh kasus uji coba dari data generator skenario 2 yang dibatasi

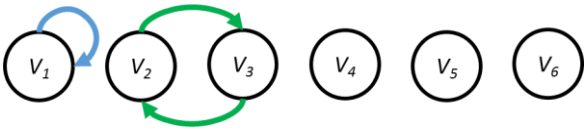
Pertama-tama permutasi yang berupa notasi dua baris diubah menjadi permutasi notasi siklus sehingga membentuk sejumlah siklus seperti yang telah dijelaskan pada subbab 2.6.1. Diketahui bahwa permutasi $P = \{1, 3, 2, 5, 6, 4\}$ yang dalam notasi dua baris adalah $P = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 3 & 2 & 5 & 6 & 4 \end{pmatrix}$. Setiap simpul yang belum menjadi bagian suatu siklus melakukan *traversal* terhadap permutasinya hingga kembali ke simpul awal.

Dimulai dari simpul V_1 yang belum menjadi bagian dari suatu siklus sehingga dilakukan *traversal*. Karena pada saat *traversal* pertama langsung kembali ke simpul V_1, maka hasil *traversal* pada simpul V_1 secara berurutan adalah {1} yang menjadi siklus pertama. Ilustrasi pembentukan siklus pertama ditunjukkan pada Gambar 5.2.



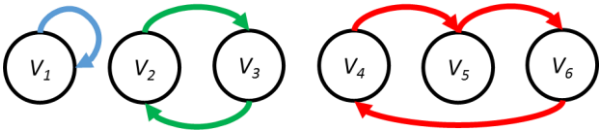
Gambar 5.2 Ilustrasi pembentukan siklus pertama(biru)

Selanjutnya, simpul V_2 yang juga belum menjadi bagian dari suatu siklus sehingga dilakukan *traversal*. *Traversal* dimulai dari V_2 kemudian V_3 dan kembali ke V_2 sehingga hasil *traversal* pada simpul V_2 secara berurutan adalah {2, 3} yang menjadi siklus kedua. Ilustrasi pembentukan siklus kedua ditunjukkan pada Gambar 5.3.



Gambar 5.3 Ilustrasi pembentukan siklus kedua(hijau)

Kemudian pada simpul V_3, karena V_3 sudah menjadi bagian dari suatu siklus sehingga tidak perlu dilakukan *traversal*. Pada simpul V_4 belum menjadi bagian dari suatu siklus sehingga dilakukan *traversal*. *Traversal* dimulai dari V_4 ke V_5 kemudian V_6 dan kembali ke V_4 sehingga hasil *traversal* pada simpul V_4 secara berurutan adalah {4, 5, 6} yang menjadi siklus ketiga. Ilustrasi pembentukan siklus ketiga ditunjukkan pada Gambar 5.4.



Gambar 5.4 Ilustrasi pembentukan siklus ketiga(merah)

Selanjutnya pada simpul V_5 dan V_6 telah menjadi bagian dari suatu siklus. Dengan begitu setiap simpul telah menjadi bagian dari

siklus dan telah didapatkan semua siklus yang ada yaitu 3 siklus yang terdiri dari {1}, {2, 3}, dan {4, 5, 6}. Dan apabila dibandingkan dengan bentuk permutasi notasi siklus dari permutasi notasi dua baris $P = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 3 & 2 & 5 & 6 & 4 \end{pmatrix}$, maka didapatkan $P = (1)(2,3)(4,5,6)$ yang dimana identik dengan siklus yang didapatkan.

Setelah pembentuk siklus maka dilakukan perhitungan bobot sisi maksimum pada setiap pangkat permutasi pada setiap siklus seperti yang telah dibahas pada subbab 2.6.2.

Pada siklus pertama dengan banyak simpul $N = 1$ yaitu {1} dicari bobot A_{V_i} dan B_{V_j} terbesar dari simpul yang pada siklus untuk mendapatkan ukuran larik setiap simpul yang diilustrasikan pada Gambar 5.5.

Bobot	V_1
A	3
B	2

Gambar 5.5 Ilustrasi pengambilan bobot A dan B terbesar pada siklus pertama

Karena siklus hanya terdiri atas satu simpul maka bobot A dan B terbesar adalah simpul itu sendiri sehingga ukuran larik setiap simpul adalah $M = \lceil (A_{V_1} + B_{V_1} + 1)/2 \rceil = 3$. Maka ukuran larik $coefA$ dan $coefB$ yang masing-masing merepresentasikan bobot A dan B pada setiap simpul pada siklus tersebut adalah $N * M$ yaitu 3. Kemudian pada setiap simpul pada siklus tersebut melakukan pemberian nilai pada larik $coefA$ dan $coefB$ sesuai dengan aturan pada Tabel 2.8 dan Tabel 2.9.

Pada simpul V_1 yang merupakan simpul pada posisi ke-0 pada siklus dengan bobot $A_{V_1} = 3$ bernilai ganjil, maka memberikan nilai $coefA_{(N-1-i)*M+\lfloor A_{V_1}/2 \rfloor} = N + 1$ sehingga $coefA_1 = 2$, dan dengan bobot $B_{V_1} = 2$ bernilai genap, maka memberikan nilai $coefB_{i*M+\lfloor B_{V_1}/2 \rfloor} = 1$ sehingga $coefB_1 = 1$.

Ringkasan pemberian nilai pada $coefA$ dan $coefB$ pada siklus ini ditunjukkan pada Tabel 5.1 dan Tabel 5.2.

Tabel 5.1 Pemberian nilai coefA pada siklus pertama

$coefA$	Posisi	Nilai
$A_{V_1} = 3$	$(N-1-i)*M + \lfloor A_{V_1}/2 \rfloor = 1$	$N + 1 = 2$

Tabel 5.2 Pemberian nilai coefB pada siklus pertama

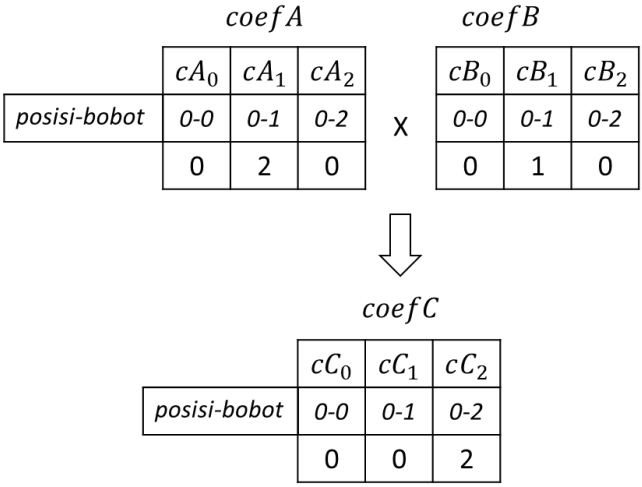
$coefB$	Posisi	Nilai
$B_{V_1} = 2$	$i * M + \lfloor B_{V_1}/2 \rfloor = 1$	1

Kemudian larik $coefA$ dan $coefB$ dikonvolusi menggunakan algoritma FFT menghasilkan larik $coefC$ berukuran $(2N-1) * M = 3$. Ilustrasi pemberian nilai $coefA$ dan $coefB$ serta hasil konvolusi $coefC$ ditunjukkan pada Gambar 5.6

Dari hasil konvolusi $coefC$ pada Gambar 5.6, dilakukan evaluasi untuk mendapatkan nilai bobot sisi maksimum pada setiap pangkat permutasi sesuai dengan aturan pada Tabel 2.10 dan Tabel 2.11.

Posisi $coefC_2$ memenuhi kondisi $\left\lfloor \frac{k}{M} \right\rfloor \geq N - 1$ sehingga merepresentasikan pangkat permutasi $\left\lfloor \frac{k}{M} \right\rfloor - (N-1) = 0$ dan juga nilai $coefC_2 = 2$ berada di antara $N + 1$ hingga $(N+1) * N$ sehingga merepresentasikan bobot $k\%M * 2 + 1 = 5$. Dengan

begitu nilai dan posisi $coefC_2$ menyatakan terdapat bobot sisi sebesar 5 pada pangkat permutasi 0.



Gambar 5.6 Ilustrasi pemberian nilai coefA dan coefB serta hasil konvolusi coefC pada siklus pertama

Ringkasan evaluasi larik $coefC$ pada siklus pertama dengan diberikan tanda(warna kuning) pada setiap $coefC_k$ yang merepresentasikan bobot sisi maksimum pada setiap pangkat permutasi ditunjukkan pada Tabel 5.3.

Tabel 5.3 Evaluasi hasil konvolusi $coefC$ pada siklus pertama

$coefC_k$	Pangkat permutasi	Bobot
$coefC_2 = 2$	$\left\lfloor \frac{k}{M} \right\rfloor - (N-1) = 0$	$k\%M * 2 + 1 = 5$

Karena hanya $coefC_2$ yang bernilai lebih dari 0 maka evaluasi $coefC$ berakhir dan dapat ditentukan pada pangkat permutasi 0 memiliki bobot sisi maksimum sebesar 5 yang direpresentasikan oleh $coefC_2$. Sehingga didapatkan hasil evaluasi siklus pertama, yaitu {5}.

Kemudian pada siklus kedua dengan banyak simpul $N = 2$ yaitu {2, 3} dicari bobot A_{V_i} dan B_{V_j} terbesar dari simpul yang pada siklus untuk mendapatkan ukuran larik setiap simpul yang diilustrasikan pada Gambar 5.7.

Bobot	V_2	V_3
A	0	4
B	4	0

Gambar 5.7 Ilustrasi pengambilan bobot A dan B terbesar pada siklus kedua

Pada siklus ini bobot A dan B terbesar adalah A_{V_3} dan B_{V_2} sehingga ukuran larik setiap simpul adalah $M = \lceil (A_{V_3} + B_{V_2} + 1)/2 \rceil = 5$. Maka ukuran larik $coefA$ dan $coefB$ yang masing-masing merepresentasikan bobot A dan B pada setiap simpul pada siklus tersebut adalah $N * M$ yaitu 10. Kemudian pada setiap simpul pada siklus tersebut melakukan pemberian nilai pada larik $coefA$ dan $coefB$ sesuai dengan aturan pada Tabel 2.8 dan Tabel 2.9.

Pertama pada simpul V_2 yang merupakan simpul pada posisi ke-0 pada siklus kedua dengan bobot $A_{V_2} = 0$ bernilai genap, maka memberikan nilai $coefA_{(N-1-i)*M+\lfloor A_{V_2}/2 \rfloor} = 1$ sehingga $coefA_5 = 1$, dan dengan bobot $B_{V_2} = 4$ bernilai genap, maka memberikan nilai $coefB_{i*M+\lfloor B_{V_2}/2 \rfloor} = 1$ sehingga $coefB_2 = 1$.

Kedua pada simpul V_3 yang merupakan simpul pada posisi ke-1 pada siklus kedua dengan bobot $A_{V_3} = 4$ bernilai genap, maka memberikan nilai $coefA_{(N-1-i)*M+\lfloor A_{V_3}/2 \rfloor} = 1$ sehingga $coefA_2 = 1$, dan dengan bobot $B_{V_3} = 0$ bernilai genap, maka memberikan nilai $coefB_{i*M+\lfloor B_{V_3}/2 \rfloor} = 1$ sehingga $coefB_5 = 1$.

Ringkasan pemberian nilai pada $coefA$ dan $coefB$ pada siklus ini ditunjukkan pada Tabel 5.4 dan Tabel 5.5.

Tabel 5.4 Pemberian nilai coefA pada siklus kedua

$coefA$	Posisi	Nilai
$A_{V_2} = 0$	$(N-1-i)*M + \lfloor A_{V_2}/2 \rfloor = 5$	1
$A_{V_3} = 4$	$(N-1-i)*M + \lfloor A_{V_3}/2 \rfloor = 2$	1

Tabel 5.5 Pemberian nilai coefB pada siklus kedua

$coefB$	Posisi	Nilai
$B_{V_2} = 4$	$i*M + \lfloor B_{V_2}/2 \rfloor = 2$	1
$B_{V_3} = 0$	$i*M + \lfloor B_{V_3}/2 \rfloor = 5$	1

Kemudian larik $coefA$ dan $coefB$ dikonvolusi menggunakan algoritma FFT menghasilkan larik $coefC$ yang berukuran $(2N-1)*M = 10$. Ilustrasi pemberian nilai $coefA$ dan $coefB$ serta hasil konvolusi $coefC$ ditunjukkan pada Gambar 5.8, Gambar 5.9, dan Gambar 5.10.

Dari hasil konvolusi $coefC$ pada Gambar 5.10, dilakukan evaluasi untuk mendapatkan nilai bobot sisi maksimum pada setiap pangkat permutasi sesuai dengan aturan pada Tabel 2.10 dan Tabel 2.11.

$coefA$

	cA_0	cA_1	cA_2	cA_3	cA_4	cA_5	cA_6	cA_7	cA_8	cA_9
posisi-bobot	*0-0*	*0-1*	*0-2*	*0-3*	*0-4*	*1-0*	*1-1*	*1-2*	*1-3*	*1-4*
	0	0	1	0	0	1	0	0	0	0

Gambar 5.8 Ilustrasi pemberian nilai coefA pada siklus kedua

$coefB$

	cB_0	cB_1	cB_2	cB_3	cB_4	cB_5	cB_6	cB_7	cB_8	cB_9
posisi-bobot	*0-0*	*0-1*	*0-2*	*0-3*	*0-4*	*1-0*	*1-1*	*1-2*	*1-3*	*1-4*
	0	0	1	0	0	1	0	0	0	0

Gambar 5.9 Ilustrasi pemberian nilai coefB pada siklus kedua

$coefC$

	cC_0	cC_1	cC_2	cC_3	cC_4	cC_5	cC_6	cC_7	cC_8	cC_9	cC_{10}	cC_{11}	cC_{12}	cC_{13}	cC_{14}
posisi-bobot	*0-0*	*0-1*	*0-2*	*0-3*	*0-4*	*1-0*	*1-1*	*1-2*	*1-3*	*1-4*	*2-0*	*2-1*	*2-2*	*2-3*	*2-4*
	0	0	0	0	1	0	0	2	0	0	1	0	0	0	0

Gambar 5.10 Ilustrasi hasil konvolusi coefC antara coefA dan coefB pada siklus kedua

Pertama dilakukan evaluasi terhadap $coefC_4$. Posisi $coefC_4$ memenuhi kondisi $\left\lfloor \frac{k}{M} \right\rfloor \leq N-2$ sehingga merepresentasikan pangkat permutasi $\left\lfloor \frac{k}{M} \right\rfloor + 1 = 1$ dan juga nilai $coefC_4 = 1$ berada di antara 1 hingga N sehingga merepresentasikan bobot $k\%M * 2 = 8$. Maka nilai dan posisi $coefC_4$ menyatakan terdapat bobot sisi sebesar 8 pada pangkat permutasi 1.

Kedua dilakukan evaluasi terhadap $coefC_7$. Posisi $coefC_7$ memenuhi kondisi $\left\lfloor \frac{k}{M} \right\rfloor \geq N-1$ sehingga merepresentasikan

pangkat permutasi $\left\lfloor \frac{k}{M} \right\rfloor - (N-1) = 0$ dan juga nilai $coefC_7 = 2$ berada di antara 1 hingga N sehingga merepresentasikan bobot $k\%M * 2 = 4$. Maka nilai dan posisi $coefC_7$ menyatakan terdapat bobot sisi sebesar 4 pada pangkat permutasi 0.

Ketiga dilakukan evaluasi terhadap $coefC_{10}$. Posisi $coefC_{10}$ memenuhi kondisi $\left\lfloor \frac{k}{M} \right\rfloor \geq N-1$ sehingga merepresentasikan pangkat permutasi $\left\lfloor \frac{k}{M} \right\rfloor - (N-1) = 1$ dan juga nilai $coefC_{10} = 1$ berada di antara 1 hingga N sehingga merepresentasikan bobot $k\%M * 2 = 0$. Maka nilai dan posisi $coefC_{10}$ menyatakan terdapat bobot sisi sebesar 0 pada pangkat permutasi 1.

Ringkasan evaluasi larik $coefC$ pada siklus kedua dengan diberikan tanda(warna kuning) pada setiap $coefC_k$ yang merepresentasikan bobot sisi maksimum pada setiap pangkat permutasi ditunjukkan pada Tabel 5.6.

Tabel 5.6 Evaluasi hasil konvolusi coefC pada siklus kedua

$coefC_k$	Pangkat permutasi	Bobot
$coefC_4 = 1$	$\left\lfloor \frac{k}{M} \right\rfloor + 1 = 1$	$k\%M * 2 = 8$
$coefC_7 = 2$	$\left\lfloor \frac{k}{M} \right\rfloor - (N-1) = 0$	$k\%M * 2 = 4$
$coefC_{11} = 1$	$\left\lfloor \frac{k}{M} \right\rfloor - (N-1) = 1$	$k\%M * 2 = 0$

Dari evaluasi hasil konvolusi $coefC$ dapat ditentukan bahwa pada pangkat permutasi 0 memiliki bobot sisi maksimum sebesar 4 yang direpresentasikan oleh $coefC_7$, dan pada pangkat permutasi 1 memiliki bobot sisi maksimum sebesar 8 yang direpresentasikan

oleh $coefC_4$. Dengan begitu dapat dibentuk hasil evaluasi siklus kedua, yaitu {4, 8}.

Kemudian pada siklus ketiga dengan banyak simpul $N = 3$ yaitu {4, 5, 6} dicari bobot A_{V_i} dan B_{V_j} terbesar dari simpul yang pada siklus untuk mendapatkan ukuran larik setiap simpul yang diilustrasikan pada Gambar 5.11.

Bobot	V_4	V_5	V_6
A	1	4	2
B	0	2	3

Gambar 5.11 Ilustrasi pengambilan bobot A dan B terbesar pada siklus ketiga

Pada siklus ini bobot A dan B terbesar adalah A_{V_5} dan B_{V_6} sehingga ukuran larik setiap simpul adalah $M = \lceil (A_{V_5} + B_{V_6} + 1)/2 \rceil = 4$. Maka ukuran larik $coefA$ dan $coefB$ yang masing-masing merepresentasikan bobot A dan B pada setiap simpul pada siklus tersebut adalah $N * M$ yaitu 12. Kemudian pada setiap simpul pada siklus tersebut melakukan pemberian nilai pada larik $coefA$ dan $coefB$ sesuai dengan aturan pada Tabel 2.8 dan Tabel 2.9.

Pertama pada simpul V_4 yang merupakan simpul pada posisi ke-0 pada siklus ketiga dengan bobot $A_{V_4} = 1$ bernilai ganjil, maka memberikan nilai $coefA_{(N-1-i)*M+\lfloor A_{V_2}/2 \rfloor} = N + 1$ sehingga $coefA_8 = N + 1$, dan dengan bobot $B_{V_4} = 0$ bernilai genap, maka memberikan nilai $coefB_{i*M+\lfloor B_{V_4}/2 \rfloor} = 1$ sehingga $coefB_0 = 1$.

Kedua pada simpul V_5 yang merupakan simpul pada posisi ke-1 pada siklus ketiga dengan bobot $A_{V_5} = 4$ bernilai genap, maka memberikan nilai $coefA_{(N-1-i)*M+\lfloor A_{V_5}/2 \rfloor} = 1$ sehingga

$coefA_6 = 1$, dan dengan bobot $B_{V_5} = 2$ bernilai genap, maka memberikan nilai $coefB_{i*M+\lfloor B_{V_5}/2 \rfloor} = 1$ sehingga $coefB_5 = 1$.

Ketiga pada simpul V_6 yang merupakan simpul pada posisi ke-2 pada siklus ketiga dengan bobot $A_{V_6} = 2$ bernilai genap, maka memberikan nilai $coefA_{(N-1-i)*M+\lfloor A_{V_6}/2 \rfloor} = 1$ sehingga $coefA_1 = 1$, dan dengan bobot $B_{V_6} = 3$ bernilai ganjil, maka memberikan nilai $coefB_{i*M+\lfloor B_{V_6}/2 \rfloor} = N + 1$ sehingga $coefB_9 = N + 1$. Ringkasan pemberian nilai pada $coefA$ dan $coefB$ pada siklus ini ditunjukkan padaTabel 5.7 dan Tabel 5.8.

Tabel 5.7 Pemberian nilai coefA pada siklus ketiga

$coefA$	Posisi	Nilai
$A_{V_4} = 1$	$(N-1-i)*M + \lfloor A_{V_4}/2 \rfloor = 8$	$N + 1 = 4$
$A_{V_5} = 4$	$(N-1-i)*M + \lfloor A_{V_5}/2 \rfloor = 6$	1
$A_{V_6} = 2$	$(N-1-i)*M + \lfloor A_{V_6}/2 \rfloor = 1$	1

Tabel 5.8 Pemberian nilai coefB pada siklus ketiga

$coefB$	Posisi	Nilai
$B_{V_4} = 0$	$i*M + \lfloor B_{V_4}/2 \rfloor = 0$	1
$B_{V_5} = 2$	$i*M + \lfloor B_{V_5}/2 \rfloor = 5$	1
$B_{V_6} = 3$	$i*M + \lfloor B_{V_6}/2 \rfloor = 9$	$N + 1 = 4$

Kemudian larik $coefA$ dan $coefB$ dikonvolusi menggunakan algoritma FFT menghasilkan larik $coefC$ yang berukuran $(2N-1)*M = 20$. Ilustrasi pemberian nilai $coefA$ dan $coefB$

serta hasil konvolusi $coefC$ ditunjukkan pada Gambar 5.12, Gambar 5.13, dan Gambar 5.14.

$coefA$

	cA_0	cA_1	cA_2	cA_3	cA_4	cA_5	cA_6	cA_7	cA_8	cA_9	cA_{10}	cA_{11}
posisi-bobot	*0-0*	*0-1*	*0-2*	*0-3*	*1-0*	*1-1*	*1-2*	*1-3*	*2-0*	*2-1*	*2-2*	*2-3*
	0	1	0	0	0	0	1	0	4	0	0	0

Gambar 5.12 Ilustrasi pemberian nilai coefA pada siklus ketiga

$coefB$

	cB_0	cB_1	cB_2	cB_3	cB_4	cB_5	cB_6	cB_7	cB_8	cB_9	cB_{10}	cB_{11}
posisi-bobot	*0-0*	*0-1*	*0-2*	*0-3*	*1-0*	*1-1*	*1-2*	*1-3*	*2-0*	*2-1*	*2-2*	*2-3*
	1	0	0	0	0	1	0	0	0	4	0	0

Gambar 5.13 Ilustrasi pemberian nilai coefB pada siklus ketiga

$coefC$

	cC_0	cC_1	cC_2	cC_3	cC_4	cC_5	cC_6	cC_7	cC_8	cC_9	cC_{10}	cC_{11}
posisi-bobot	*0-0*	*0-1*	*0-2*	*0-3*	*1-0*	*1-1*	*1-2*	*1-3*	*2-0*	*2-1*	*2-2*	*2-3*
	0	1	0	0	0	0	2	0	4	0	4	1

	cC_{12}	cC_{13}	cC_{14}	cC_{15}	cC_{16}	cC_{17}	cC_{18}	cC_{19}
posisi-bobot	*3-0*	*3-1*	*3-2*	*3-3*	*4-0*	*4-1*	*4-2*	*4-3*
	0	4	0	4	0	16	0	0

Gambar 5.14 Ilustrasi hasil konvolusi coefC antara coefA dan coefB pada siklus ketiga

Dari hasil konvolusi $coefC$ pada Gambar 5.14, dilakukan evaluasi untuk mendapatkan nilai bobot sisi maksimum pada setiap pangkat permutasi sesuai dengan aturan pada Tabel 2.10 dan Tabel 2.11.

Pertama dilakukan evaluasi terhadap $coefC_1$. Posisi $coefC_1$ memenuhi kondisi $\left\lfloor \frac{k}{M} \right\rfloor \leq N-2$ sehingga merepresentasikan pangkat permutasi $\left\lfloor \frac{k}{M} \right\rfloor + 1 = 1$ dan juga nilai $coefC_1 = 1$ berada di antara 1 hingga N sehingga merepresentasikan bobot $k\%M * 2 = 2$. Maka nilai dan posisi $coefC_1$ menyatakan terdapat bobot sisi sebesar 2 pada pangkat permutasi 1.

Kedua dilakukan evaluasi terhadap $coefC_6$. Posisi $coefC_6$ memenuhi kondisi $\left\lfloor \frac{k}{M} \right\rfloor \leq N-2$ sehingga merepresentasikan pangkat permutasi $\left\lfloor \frac{k}{M} \right\rfloor + 1 = 2$ dan juga nilai $coefC_6 = 2$ berada di antara 1 hingga N sehingga merepresentasikan bobot $k\%M * 2 = 4$. Maka nilai dan posisi $coefC_5$ menyatakan terdapat bobot sisi sebesar 4 pada pangkat permutasi 2.

Ketiga dilakukan evaluasi terhadap $coefC_8$. Posisi $coefC_8$ memenuhi kondisi $\left\lfloor \frac{k}{M} \right\rfloor \geq N-1$ sehingga merepresentasikan pangkat permutasi $\left\lfloor \frac{k}{M} \right\rfloor - (N-1) = 0$ dan juga nilai $coefC_8 = 4$ berada di antara $N+1$ hingga $(N+1) * N$ sehingga merepresentasikan bobot $k\%M * 2 + 1 = 1$. Maka nilai dan posisi $coefC_8$ menyatakan terdapat bobot sisi sebesar 1 pada pangkat permutasi 0.

Keempat dilakukan evaluasi terhadap $coefC_{10}$. Posisi $coefC_{10}$ memenuhi kondisi $\left\lfloor \frac{k}{M} \right\rfloor \geq N-1$ sehingga merepresentasikan pangkat permutasi $\left\lfloor \frac{k}{M} \right\rfloor - (N-1) = 0$ dan juga nilai $coefC_{10} = 4$ berada di antara $N+1$ hingga $(N+1) * N$ sehingga merepresentasikan bobot $k\%M * 2 + 1 = 5$. Maka nilai dan posisi $coefC_{10}$ menyatakan terdapat bobot sisi sebesar 5 pada pangkat permutasi 0.

Kelima dilakukan evaluasi terhadap $coefC_{11}$. Posisi $coefC_{11}$ memenuhi kondisi $\left\lfloor \frac{k}{M} \right\rfloor \geq N-1$ sehingga merepresentasikan pangkat permutasi $\left\lfloor \frac{k}{M} \right\rfloor - (N-1) = 0$ dan juga nilai $coefC_{11} = 1$ berada di antara 1 hingga N sehingga merepresentasikan bobot $k\%M * 2 = 6$. Maka nilai dan posisi $coefC_{11}$ menyatakan terdapat bobot sisi sebesar 6 pada pangkat permutasi 0.

Keenam dilakukan evaluasi terhadap $coefC_{13}$. Posisi $coefC_{13}$ memenuhi kondisi $\left\lfloor \frac{k}{M} \right\rfloor \geq N-1$ sehingga merepresentasikan pangkat permutasi $\left\lfloor \frac{k}{M} \right\rfloor - (N-1) = 1$ dan juga nilai $coefC_{13} = 4$ berada di antara $N+1$ hingga $(N+1) * N$ sehingga merepresentasikan bobot $k\%M * 2 + 1 = 3$. Maka nilai dan posisi $coefC_{13}$ menyatakan terdapat bobot sisi sebesar 3 pada pangkat permutasi 1.

Ketujuh dilakukan evaluasi terhadap $coefC_{15}$. Posisi $coefC_{15}$ memenuhi kondisi $\left\lfloor \frac{k}{M} \right\rfloor \geq N-1$ sehingga merepresentasikan pangkat permutasi $\left\lfloor \frac{k}{M} \right\rfloor - (N-1) = 1$ dan juga nilai $coefC_{15} = 4$ berada di antara $N+1$ hingga $(N+1) * N$ sehingga merepresentasikan bobot $k\%M * 2 + 1 = 7$. Maka nilai dan posisi $coefC_{15}$ menyatakan terdapat bobot sisi sebesar 7 pada pangkat permutasi 1.

Kedelapan dilakukan evaluasi terhadap $coefC_{17}$. Posisi $coefC_{17}$ memenuhi kondisi $\left\lfloor \frac{k}{M} \right\rfloor \geq N-1$ sehingga merepresentasikan pangkat permutasi $\left\lfloor \frac{k}{M} \right\rfloor - (N-1) = 2$ dan juga nilai $coefC_{17} = 16$ berada di antara $(N+1)^2$ hingga $(N+1)^2 * N$ sehingga merepresentasikan bobo $k\%M * 2 + 2 = 4$. Maka nilai dan posisi $coefC_{17}$ menyatakan terdapat bobot sisi sebesar 4 pada pangkat permutasi 2.

Ringkasan evaluasi larik $coefC$ pada siklus kedua dengan diberikan tanda(warna kuning) pada setiap $coefC_k$ yang merepresentasikan bobot sisi maksimum pada setiap pangkat permutasi ditunjukkan pada Tabel 5.9.

Tabel 5.9 Evaluasi hasil konvolusi coefC pada siklus ketiga

$coefC_k$	Pangkat permutasi	Bobot
$coefC_1 = 1$	$\left\lfloor \frac{k}{M} \right\rfloor + 1 = 1$	$k\%M * 2 = 2$
$coefC_6 = 2$	$\left\lfloor \frac{k}{M} \right\rfloor + 1 = 2$	$k\%M * 2 = 4$
$coefC_8 = 4$	$\left\lfloor \frac{k}{M} \right\rfloor - (N-1) = 0$	$k\%M * 2 + 1 = 1$
$coefC_{10} = 4$	$\left\lfloor \frac{k}{M} \right\rfloor - (N-1) = 0$	$k\%M * 2 + 1 = 5$
$coefC_{11} = 1$	$\left\lfloor \frac{k}{M} \right\rfloor - (N-1) = 0$	$k\%M * 2 = 6$
$coefC_{13} = 4$	$\left\lfloor \frac{k}{M} \right\rfloor - (N-1) = 1$	$k\%M * 2 + 1 = 3$
$coefC_{15} = 4$	$\left\lfloor \frac{k}{M} \right\rfloor - (N-1) = 1$	$k\%M * 2 + 1 = 7$
$coefC_{17} = 16$	$\left\lfloor \frac{k}{M} \right\rfloor - (N-1) = 2$	$k\%M * 2 + 2 = 4$

Dari evaluasi hasil konvolusi $coefC$ dapat ditentukan bahwa pada pangkat permutasi 0 memiliki bobot sisi maksimum sebesar 6 yang direpresentasikan oleh $coefC_{11}$, dan pada pangkat permutasi 1 memiliki bobot sisi maksimum sebesar 7 yang direpresentasikan

oleh $coefC_{15}$, dan pada pangkat permutasi 2 memiliki bobot sisi maksimum sebesar 4 yang direpresentasikan oleh $coefC_6$ dan $coefC_{17}$. Dengan begitu dapat dibentuk hasil evaluasi siklus ketiga, yaitu {6, 7, 4}.

Setelah melakukan evaluasi setiap siklus, maka hasil evaluasi tersebut digabungkan untuk mendapatkan bobot sisi maksimum pada pangkat permutasi 0 hingga $Q - 1$. Dari evaluasi setiap siklus diketahui bahwa terdapat 3 ukuran siklus yaitu $cycleIndex = \{1, 2, 3\}$ dan diketahui ketiga hasil evaluasi siklus tersebut adalah $result[1] = \{5\}$, $result[2] = \{4, 8\}$, dan $result[3] = \{6, 7, 4\}$ Dari hasil evaluasi siklus tersebut dapat diambil bobot sisi terbesar pada pangkat permutasi 0 hingga $Q - 1$ dari setiap siklus yang ada, dapat dilihat pada Gambar 5.15.

$cycleIndex = \{1,2,3\}$

Q	0	1	2	3	4	5	6	7	8	9
$result[1]$	5	5	5	5	5	5	5	5	5	5
$result[2]$	4	8	4	8	4	8	4	8	4	8
$result[3]$	6	7	4	6	7	4	6	7	4	6
ans	6	8	5	8	7	8	6	8	5	8

Gambar 5.15 Ilustrasi pengambilan bobot sisi terbesar dari hasil evaluasi siklus pada pangkat permutasi 0 hingga $Q - 1$

Dari Gambar 5.15 dapat ditentukan bobot sisi maksimum pada pangkat permutasi 0 hingga $Q - 1$ secara berurutan yaitu {6, 8, 5, 8, 7, 8, 6 ,8, 5, 8}. Kemudian sistem penyelesaian yang diimplementasi pada BAB 4 dijalankan dengan masukkan pada contoh kasus uji kebenaran pada Gambar 5.1, dan hasil keluaran sesuai seperti hasil uji coba kebenaran, yaitu tercetak “6 8 5 8 7 8 6 8 5 8” seperti pada Gambar 5.16.

```
6
1 3 2 5 6 4
3 0 4 1 4 2
2 4 0 0 2 3
10
6 8 5 8 7 8 6 8 5 8

--------------------------------
Process exited after 0.3628 seconds with return value 0
Press any key to continue . . .
```

Gambar 5.16 Masukkan dan keluaran pada program berdasarkan contoh kasus uji kebenaran

Selanjutnya dilakukan juga uji coba kebenaran beserta penilaian dengan mengumpulkan berkas kode sumber ke situs penilaian daring SPOJ pada kode soal MEPPERM. Hasil uji kebenaran, jumlah waktu eksekusi, dan pengunaan memori program berdasarkan sejumlah kasus uji coba yang disediakan oleh *problemsetter* ditunjukkan pada Gambar 5.17.

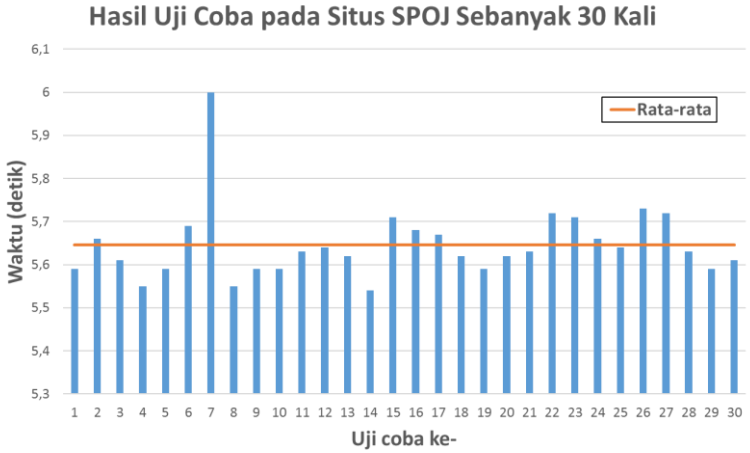
18428364	☐	2016-12-21 17:19:41	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.54	3.1M	C++ 4.3.2

Gambar 5.17 Hasil uji coba kebenaran pada situs penilaian daring SPOJ kode soal MEPPERM

Dari hasil uji coba kebenaran pada Gambar 5.17 didapatkan umpan balik *Accepted.* Jumlah waktu yang dibutuhkan program untuk menjalankan sejumlah kasus uji coba yang disediakan oleh *problemsetter* adalah 5,54 detik dan memori yang dibutuhkan program adalah 3,1 MB.

Berdasarkan hasil uji coba kebenaran ini telah membuktikan bahwa implementasi kode program pada BAB 4 berhasil menyelesaikan permasalahan dalam mencari bobot sisi maksimum pada graf yang terbentuk dari pangkat permutasi sesuai dengan batasan-batasan yang diberikan pada subbab 2.1.

Kemudian dilakukan pengiriman kode sumber sebanyak 30 kali untuk melihat variasi jumlah waktu yang dibutuhkan program untuk menjalankan sejumlah kasus uji coba yang disediakan oleh *problemsetter*. Hasil uji coba sebanyak 30 kali dapat dilihat pada Gambar A.1, Gambar A.2, Gambar A.3, Tabel A.1, dan Tabel A.2 yang direpresentasikan oleh grafik pada Gambar 5.18.



Gambar 5.18 Grafik hasil uji coba pada situs SPOJ sebanyak 30 kali

Dari hasil uji coba sebanyak 30 kali, didapatkan rata-rata waktu eksekusi program adalah 5,646 detik dan penggunaan memori program sebesar 3,1 MB.

5.2.2 Uji Coba Kinerja

Proses pembentukan siklus dari permutasi yang terdiri dari N simpul yang telah dibahas pada subbab 2.6.1 membutuhkan komputasi dengan kompleksitas $O(N)$. Kemudian proses perhitungan bobot sisi maksimum pada setiap siklus dimana M merupakan perjumlahan bobot A dan B terbesar, yang telah dibahas

pada subbab 2.6.2 membutuhkan komputasi dengan kompleksitas $O(NMlog(NM))$. Terakhir proses penggabungan bobot sisi maksimum pada setiap siklus untuk mendapatkan bobot sisi maksimum pada pangkat permutasi 0 hingga $Q-1$ dimana Q merupakan batas pangkat permutasi, yang telah dibahas pada subbab 2.6.3 membutuhkan komputasi dengan kompleksitas $O\left(Q\sqrt{N}\right)$. Dengan begitu untuk menjalankan seluruh proses membutuhkan kompleksitas komputasi $O(NMlog(NM)+Q\sqrt{N})$. Dengan demikian dibutuhkan dua skenario uji coba kinerja seperti yang telah dibahas pada subbab 3.3.

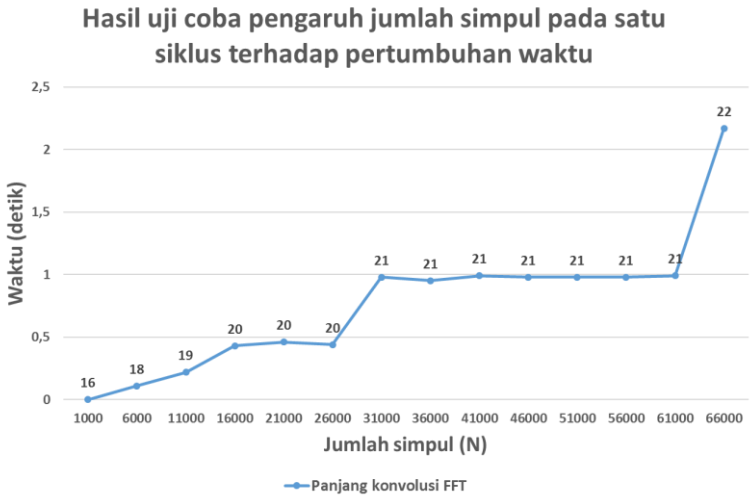
Uji coba skenario pertama dilakukan untuk melakukan uji coba pengaruh jumlah simpul pada satu siklus terhadap pertumbuhan waktu. Banyak simpul berkisaran antara 1000 hingga 66000 dengan rentang 5000. Bobot A dan B setiap simpul dibuat acak berkisaran antara 0 hingga 16 sesuai dengan batasan soal sehingga diasumsi banyak nilai antara 0 hingga perjumlahan A dan B terbesar adalah $M=33$. Permutasi dibuat sedemikian rupa sehingga hanya membentuk 1 siklus. Batas pangkat permutasi bernilai tetap yaitu 1000000. Kasus uji coba yang dibuat menggunakan implementasi data *generator* skenario pertama pada Kode Sumber 4.15. Hasil uji coba skenario pertama ditunjukkan pada Tabel 5.10 dan Gambar 5.19.

Dari hasil uji coba pada Tabel 5.10 dan grafik Gambar 5.19, dapat dilihat bahwa setiap nilai *N* hampir atau melewati nilai kelipatan 2, maka waktu eksekusi program menjadi kira-kira 2 kali lipat dari sebelumnya. Hal ini terjadi karena konvolusi menggunakan algoritma FFT memiliki kompleksitas yang berbasis kelipatan dua, sehingga apabila nilai $N*\lceil M/2\rceil *2$ melewati batas kelipatan 2 sebelumnya, maka panjang konvolusi FFT menjadi 2 kali lipat sehingga kompleksitas menjadi dua kali lipat pula dari sebelumnya.

Tabel 5.10 Hasil uji coba pengaruh jumlah simpul pada satu siklus terhadap pertumbuhan waktu

No	Jumlah simpul	Panjang konvolusi FFT	Waktu (detik)	Memori (MB)
1	1000	16	0.07	2,8
2	6000	18	0,11	2,8
3	11000	19	0,22	2,8
4	16000	20	0,43	2,8
5	21000	20	0,46	3,1
6	26000	20	0,44	3,2
7	31000	21	0,98	3,3
8	36000	21	0,95	3,2
9	41000	21	0,99	3,3
10	46000	21	0,98	2,8
11	51000	21	0,98	2,8
12	56000	21	0,98	2,8
13	61000	21	0,99	2,8
14	66000	22	2,17	3,1

Contoh pada hasil uji coba dengan $N = 26000$ memiliki kompleksitas konvolusi menggunakan FFT sebesar $N * \lceil M/2 \rceil * 2 = 884000$ sehingga nilai kelipatan 2 terdekat dan lebih besar adalah 2^{20} memiliki waktu eksekusi 0,44 detik, dibandingkan dengan $N = 31000$ memiliki nilai $N * \lceil M/2 \rceil * 2 = 1054000$ dengan nilai kelipatan 2 terdekat yaitu 2^{21} memiliki waktu eksekusi 0,98 detik. Dan apabila nilai $N * \lceil M/2 \rceil * 2$ tidak melewati batas kelipatan 2 sebelumnya, maka kompleksitas tetap sama. Dengan begitu dapat disimpulkan bahwa pengaruh penerapan konvolusi menggunakan algoritma FFT berbasis kelipatan 2 memiliki pengaruh yang besar terhadap kompleksitas waktu eksekusi program menggunakan solusi yang diterapkan.



Gambar 5.19 Grafik hasil uji coba pengaruh jumlah simpul pada satu siklus terhadap pertumbuhan waktu

Kemudian uji coba skenario kedua dilakukan untuk melakukan uji coba pengaruh jumlah simpul yang membentuk siklus sebanyak mungkin terhadap pertumbuhan waktu. Banyak simpul berkisaran antara 1000 hingga 66000 dengan rentang 5000. Bobot A dan B setiap simpul dibuat acak berkisaran antara 0 hingga 16 sesuai dengan batasan soal sehingga diasumsi banyak nilai antara 0 hingga perjumlahan A dan B terbesar adalah $M = 33$. Permutasi dibuat sedemikian rupa agar menghasil siklus sebanyak mungkin dengan setiap siklus memiliki banyak simpul yang berbeda, yaitu kisaran $\sqrt{2N}$ siklus. Batas pangkat permutasi bernilai tetap yaitu 1000000. Kasus uji coba yang dibuat menggunakan implementasi data *generator* skenario kedua pada Kode Sumber 4.16. Hasil uji coba skenario kedua ditunjukkan pada Tabel 5.11 dan Gambar 5.20.

Tabel 5.11 Hasil uji coba pengaruh jumlah simpul yang membentuk siklus sebanyak mungkin terhadap pertumbuhan waktu

No	Jumlah simpul	Jumlah siklus	Panjang konvolusi FFT terbesar	Waktu (detik)	Memori (MB)
1	1000	45	3	0,13	2,8
2	6000	110	4	0,24	2,8
3	11000	149	4	0,32	2,8
4	16000	179	4	0,38	2,8
5	21000	205	4	0,45	2,9
6	26000	229	5	0,5	2,9
7	31000	249	5	0,55	2,9
8	36000	269	5	0,61	3,0
9	41000	287	5	0,69	3,3
10	46000	304	5	0,74	3,0
11	51000	320	5	0,8	3,0
12	56000	335	5	0,87	3,1
13	61000	350	5	0,9	3,1
14	66000	364	6	0,95	3,1

Dari hasil uji coba pada Tabel 5.11 dan grafik Gambar 5.20, dapat dilihat pengaruh jumlah siklus yang disebabkan oleh perbedaan jumlah simpul. Dengan jumlah siklus yang berbeda namun memiliki panjang konvolusi FFT terbesar yang sama menghasilkan perbedaan waktu eksekusi yang cukup signifikan dibanding dengan uji coba skenario pertama dimana hanya terdiri dari satu siklus dengan jumlah simpul yang berbeda namun memiliki panjang konvolusi FFT yang sama, tidak memiliki perbedaan waktu eksekusi yang cukup signifikan. Dengan begitu dapat disimpulkan bahwa pengaruh banyak siklus yang terbentuk mempengaruhi panjang konvolusi FFT terbesar yang terbentuk

sekaligus memberikan perbedaan waktu yang cukup signifikan apabila jumlah simpul berbeda meskipun panjang konvolusi FFT terbesar bernilai sama.

Gambar 5.20 Grafik hasil uji coba pengaruh jumlah simpul yang membentuk siklus sebanyak mungkin terhadap pertumbuhan waktu

BAB 6
KESIMPULAN DAN SARAN

Pada bab ini dijelaskan mengenai kesimpulan dari hasil uji coba yang telah dilakukan dan saran mengenai hal-hal yang masih bisa untuk dikembangkan dari tugas akhir ini.

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan terhadap implementasi penyelesaian permasalahan *Maximum Edge of Powers of Permutation* dapat diambil kesimpulan sebagai berikut:

1. Implementasi metode pencarian permutasi siklus dan konvolusi menggunakan algoritma FFT dapat menyelesaikan permasalahan *Maximum Edge of Powers of Permutation* dengan benar.
2. Waktu yang dibutuhkan oleh sistem untuk melakukan penyelesaian permasalahan mengalami pertumbuhan secara *quasilinear* terhadap pertumbuhan jumlah simpul dikali perjumlahan bobot A dan B terbesar.
3. Waktu yang dibutuhkan oleh sistem untuk melakukan penyelesaian permasalahan mengalami pertumbuhan secara *sublinear* terhadap pertumbuhan jumlah siklus.
4. Kompleksitas waktu yang dibutuhkan untuk seluruh proses sistem adalah $O(NMlogNM + Q\sqrt{N})$ pada N buah simpul dengan perjumlahan bobot A dan B terbesar yaitu M dan batas pangkat permutasi sebesar Q.

6.2 Saran

Saran yang diberikan adalah menggabungkan operasi FFT dua buah larik koefisien pada saat konvolusi. Kemudian menggunakan algoritma FFT berbasis kelipatan 4 atau 8 sehingga waktu eksekusi lebih cepat dengan penggunaan memori lebih besar dibanding

algoritma FFT berbasis kelipatan 2 atau menggunakan algoritma FFT yang tidak berbasis kelipatan 2.

LAMPIRAN A
HASIL UJI COBA PADA SITUS SPOJ SEBANYAK 30 KALI

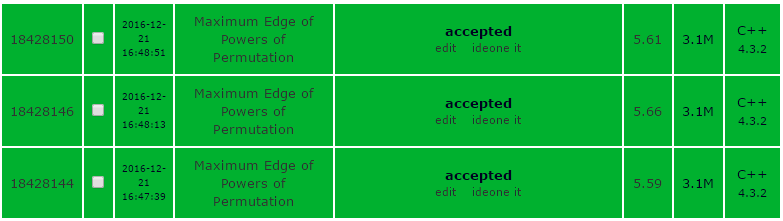
Berikut merupakan lampiran hasil uji coba pengumpulan berkas kode sumber yang merupakan solusi pada situs SPOJ sebanyak 30 kali pada subbab 5.2.1 yang direpresentasikan oleh grafik pada Gambar 5.18.

18428800	☐	2016-12-21 18:32:35	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.61	3.1M	C++ 4.3.2
18428797	☐	2016-12-21 18:32:11	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.59	3.1M	C++ 4.3.2
18428642	☐	2016-12-21 18:01:40	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.63	3.1M	C++ 4.3.2
18428639	☐	2016-12-21 18:01:15	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.72	3.1M	C++ 4.3.2
18428614	☐	2016-12-21 17:57:32	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.73	3.1M	C++ 4.3.2
18428592	☐	2016-12-21 17:55:26	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.64	3.1M	C++ 4.3.2
18428575	☐	2016-12-21 17:52:50	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.66	3.1M	C++ 4.3.2
18428569	☐	2016-12-21 17:52:25	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.71	3.1M	C++ 4.3.2
18428537	☐	2016-12-21 17:49:06	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.72	3.1M	C++ 4.3.2
18428531	☐	2016-12-21 17:48:24	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.63	3.1M	C++ 4.3.2
18428458	☐	2016-12-21 17:34:06	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.62	3.1M	C++ 4.3.2
18428451	☐	2016-12-21 17:33:15	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.59	3.1M	C++ 4.3.2

Gambar A.1 Hasil uji coba pada situs SPOJ sebanyak 30 kali (1)

18428416		2016-12-21 17:28:12	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.62	3.1M	C++ 4.3.2
18428410		2016-12-21 17:27:23	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.67	3.1M	C++ 4.3.2
18428395		2016-12-21 17:23:45	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.68	3.1M	C++ 4.3.2
18428390		2016-12-21 17:23:01	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.71	3.1M	C++ 4.3.2
18428364		2016-12-21 17:19:41	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.54	3.1M	C++ 4.3.2
18428327		2016-12-21 17:11:50	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.62	3.1M	C++ 4.3.2
18428324		2016-12-21 17:10:57	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.64	3.1M	C++ 4.3.2
18428296		2016-12-21 17:06:32	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.63	3.1M	C++ 4.3.2
18428282		2016-12-21 17:04:23	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.59	3.1M	C++ 4.3.2
18428264		2016-12-21 17:01:53	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.59	3.1M	C++ 4.3.2
18428246		2016-12-21 16:59:49	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.55	3.1M	C++ 4.3.2
18428242		2016-12-21 16:58:53	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	6.00	3.1M	C++ 4.3.2
18428220		2016-12-21 16:57:12	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.69	3.1M	C++ 4.3.2
18428186		2016-12-21 16:53:54	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.59	3.1M	C++ 4.3.2
18428171		2016-12-21 16:51:52	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.55	3.1M	C++ 4.3.2

Gambar A.2 Hasil uji coba pada situs SPOJ sebanyak 30 kali (2)



18428150		2016-12-21 16:48:51	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.61	3.1M	C++ 4.3.2
18428146		2016-12-21 16:48:13	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.66	3.1M	C++ 4.3.2
18428144		2016-12-21 16:47:39	Maximum Edge of Powers of Permutation	**accepted** edit ideone it	5.59	3.1M	C++ 4.3.2

Gambar A.3 Hasil uji coba pada situs SPOJ sebanyak 30 kali (3)

Tabel A.1 Hasil uji coba pada situs SPOJ sebanyak 30 kali (1)

No	Hasil	Waktu (detik)	Memori (MB)
1	*Accepted*	5,59	3,1
2	*Accepted*	5,66	3,1
3	*Accepted*	5,61	3,1
4	*Accepted*	5,55	3,1
5	*Accepted*	5,59	3,1
6	*Accepted*	5,69	3,1
7	*Accepted*	6,00	3,1
8	*Accepted*	5,55	3,1
9	*Accepted*	5,59	3,1
10	*Accepted*	5,59	3,1
11	*Accepted*	5,63	3,1
12	*Accepted*	5,64	3,1
13	*Accepted*	5,62	3,1
14	*Accepted*	5,54	3,1
15	*Accepted*	5,71	3,1
16	*Accepted*	5,68	3,1
17	*Accepted*	5,67	3,1
18	*Accepted*	5,62	3,1
19	*Accepted*	5,59	3,1
20	*Accepted*	5,62	3,1
21	*Accepted*	5,63	3,1
22	*Accepted*	5,72	3,1

Tabel A.2 Hasil uji coba pada situs SPOJ sebanyak 30 kali (2)

No	Hasil	Waktu (detik)	Memori (MB)
23	*Accepted*	5,71	3,1
24	*Accepted*	5,66	3,1
25	*Accepted*	5,64	3,1
26	*Accepted*	5,73	3,1
27	*Accepted*	5,72	3,1
28	*Accepted*	5,63	3,1
29	*Accepted*	5,59	3,1
30	*Accepted*	5,61	3,1

DAFTAR PUSTAKA

[1] SPOJ, “MEPPERM - Maximum Edge of Powers of Permutation,” 2010. [Online]. Available: http://www.spoj.com/problems/MEPPERM/. [Diakses 23 Desember 2016].

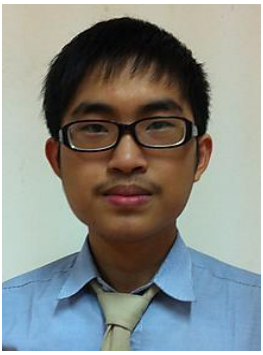
[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest dan C. Stein, Introduction to Algorithms 3rd Edition, Cambridge: MIT Press, 2009.

[3] T. Davis, “Permutation Groups and Rubik's Cube,” Berkeley Math Circle, 2000.

[4] A. Emerencia, “Multiplying Huge Integers Using Fourier Transforms,” 2007.

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS



Karsten Ari Agathon, lahir pada tanggal 03 Mei 1994 di Denpasar. Saat ini sedang menempuh pendidikan perguruan tinggi di Institut Teknologi Sepuluh Nopember Surabaya di jurusan Teknik Informatika Fakultas Teknologi Informasi angkatan tahun 2012. Penulis pernah menjadi asisten mata kuliah Dasar Pemrograman dan Algoritma dan Struktur Data. Penulis terlibat aktif dalam organisasi kemahasiswaan sebagai staff Riset dan Teknologi di Himpunan Mahasiswa Teknik Computer-Informatika(HMTC) ITS. Penulis juga aktif dalam kegiatan kepanitiaan Schematics sebagai staff National Programming Contest(NPC) Schematics 2013 dan wakil ketua NPC Schematics 2014. Penulis juga aktif menjadi asisten Lab Pemrograman(LP) Teknik Informatika ITS.

Selain itu penulis juga mengikut kompetisi pemrograman tingkat nasional dan menjadi finalis kategori pemrograman pada lomba Compfest 2013, 2014, 2015 yang diadakan oleh Universitas Indonesia, INC(Indonesia National Contest) dan ACM ICPC(ACM International Collegiate Programming Contest) 2013, 2014, 2015, 2016 yang diadakan oleh ACM di Universitas Bina Nusantara, dan mendapatkan juara 5 pada Bukalapak Programming Contest 5 yang diadakan oleh Bukalapak di Surabaya.