TUGAS AKHIR – IF184802

DESAIN DAN ANALISIS ALGORITMA PRIME FACTORIZATION PADA STUDI KASUS SPOJ SQFFACT – SQUARE-FREE INTEGERS FACTORIZATION

RADEN TEJA KUSUMA
05111640000012

Dosen Pembimbing I :
Rully Soelaiman, S.Kom., M.Kom.

Dosen Pembimbing II :
Yudhi Purwananto, S.Kom., M.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya
2020

TUGAS AKHIR – IF184802

DESAIN DAN ANALISIS ALGORITMA PRIME FACTORIZATION PADA STUDI KASUS SPOJ SQFFACT – SQUARE-FREE INTEGERS FACTORIZATION

RADEN TEJA KUSUMA
05111640000012

Dosen Pembimbing I :
Rully Soelaiman, S.Kom., M.Kom.

Dosen Pembimbing II :
Yudhi Purwananto, S.Kom., M.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya
2020

[*Halaman ini sengaja dikosongkan*]

UNDERGRADUATE THESIS – IF184802

DESIGN AND ANALYSIS ALGORITHM PRIME FACTORIZATION ON CASE STUDY SPOJ SQFFACT – SQUARE-FREE INTEGERS FACTORIZATION

RADEN TEJA KUSUMA
05111640000012

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

Supervisor II
Yudhi Purwananto, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya
2020

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

DESAIN DAN ANALISIS ALGORITMA PRIME FACTORIZATION PADA STUDI KASUS SPOJ SQFFACT – SQUARE-FREE INTEGERS FACTORIZATION

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Algoritma Pemrograman
Program Studi S-1 Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh:
RADEN TEJA KUSUMA
NRP: 051116 40000 012

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Rully Soelaiman, S.Kom., M.Kom.
NIP. 197002131994021001 ……
(Pembimbing 1)

Yudhi Purwananto, S.Kom., M.Kom.
NIP. 197007141997031002 ……
(Pembimbing 2)

SURABAYA
Juni 2020

[*Halaman ini sengaja dikosongkan*]

DESAIN DAN ANALISIS ALGORITMA *PRIME FACTORIZATION* PADA STUDI KASUS SPOJ SQFFACT – *SQUARE-FREE INTEGERS FACTORIZATION*

Nama Mahasiswa	**:**	**Raden Teja Kusuma**
NRP	**:**	**05111640000012**
Departemen	**:**	**Teknik Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas, ITS**
Dosen Pembimbing 1	**:**	**Rully Soelaiman, S.Kom., M.Kom.**
Dosen Pembimbing 2	**:**	**Yudhi Purwananto, S.Kom., M.Kom.**

Abstrak

Keamanan adalah salah satu aspek terpenting dalam pertukaran data melalui jaringan internet. Salah satu contoh metode pengamanan data adalah membuat asymmetric key yang terdiri dari public key untuk mengenkripsi data dan private key untuk mendeskripsi data. Skema keamanan seperti RSA Algorithm membutuhkan prime factorization dalam proses yang dilakukan sehingga bisa menghasilkan public key dan private key.

Dalam tugas akhir ini, akan dibahas mengenai cara penyelesaian prime factorization sebuah bilangan jika diketahui nilai euler's totient dari bilangan tersebut dengan menggunakan Carmichael's function, Euler's theorem, dan Euler's totient function yang disusun kedalam randomized algorithm.

Hasil dari tugas akhir ini telah berhasil menyelesaikan permasalahan yang diberikan dengan cukup efisien, dengan rata-rata waktu penyelesaian dalam 10 kali uji coba adalah 0,616 detik dengan penggunaan memori 56 MB.

Kata Kunci: Carmichael's function; Euler's theorem; Euler's totient function; Prime Factorization; Randomized Algorithm.

[*Halaman ini sengaja dikosongkan*]

DESIGN AND ANALYSIS ALGORITHM PRIME FACTORIZATION ON CASE STUDY SPOJ SQFFACT – SQUARE-FREE INTEGERS FACTORIZATION

Name	**: Raden Teja Kusuma**
NRP	**: 05111640000012**
Department	**: Informatics, Faculty of Intelligent Electrical and Informatics Technology, ITS**
Supervisor I	**: Rully Soelaiman, S.Kom., M.Kom.**
Supervisor II	**: Yudhi Purwananto, S.Kom., M.Kom.**

Abstract

Security is one of most important aspect in data exchange through internet. One of data securing method is to crate asymmetric key which consists of public key for data encryption and private key for data decryption. Security scheme such as RSA Algorithm require prime factorization in the process to produce public key and private key.

In this thesis, will discuss how to solve prime factorization of a number if know the Euler's totient from that number by using Carmichael's function, Euler's theorem, and Euler's totient function arranged into randomized algorithm.

The resut of this thesis have successfully resolved problem quite efficiently,with an average running time of 10 trial is 0.616 seconds with 56 MB of memory usage.

Keywords: Carmichael's function; Euler's theorem; Euler's totient function; Prime Factorization; Randomized Algorithm.

[*Halaman ini sengaja dikosongkan*]

KATA PENGANTAR

Puji syukur penulis ucapkan kepada Tuhan Yang Maha Esa atas pimpinan, penyertaan, dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

DESAIN DAN ANALISIS ALGORITMA PRIME FACTORIZATION PADA STUDI KASUS SPOJ SQFFACT – SQUARE-FREE INTEGERS FACTORIZATION

Pengerjaan Tugas Akhir ini dilakukan untuk memenuhi salah satu syarat meraih gelar Sarjana di Departemen Teknik Informatika Fakultas Teknologi Elektro dan Informatika Cerdas Institut Teknologi Sepuluh Nopember.

Dengan selesainya Tugas Akhir ini diharapkan apa yang telah dikerjakan penulis dapat memberikan manfaat bagi perkembangan ilmu pengetahuan terutama di bidang teknologi informasi serta bagi diri penulis sendiri selaku peneliti.

Penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan baik secara langsung maupun tidak langsung selama penulis mengerjakan Tugas Akhir maupun selama menempuh masa studi antara lain:

1. Terimakasih kepada Allah SWT, di mana penulis masih diberi kesempatan, kesehatan dan umur untuk menempuh kuliah disini dan menjalani hidup dengan baik.
2. Ayah, Ibu, dan Kakak penulis yang selalu memberikan perhatian, dorongan, dan kasih sayang juga tunjangan makanan supaya lebih semangat menempuh kuliah maupun pengerjaan Tugas Akhir ini.
3. Bapak Rully Soelaiman, S.Kom., M.Kom. selaku Dosen Pembimbing yang telah membimbing saya selama masa kuliah maupun selama penyelesaian Tugas Akhir ini, Dosen yang paling perhatian kepada saya dalam memberi ilmu, nasihat, dan motivasi selama

berada menempuh kuliah di Departemen Teknik Informatika ITS.

4. Bapak Yudhi Purwananto, S.Kom., M.Kom. Selaku dosen pembimbing yang telah memberikan ilmu, dan masukan kepada penulis.
5. Yoshima Syach Putri dan Taufiq Tirtajiwangga yang tidak lelah membantu penulis selama masa studi dan dalam pengerjaan Tugas Akhir ini.
6. Teman-teman angkatan 2016 departemen Teknik Informatika ITS yang telah menemani perjuangan penulis selama empat tahun masa perkuliahan.
7. Serta pihak-pihak lain yang tidak dapat disebutkan disini yang telah banyak membantu penulis dalam penyusunan Tugas Akhir ini.

Penulis mohon maaf apabila masih ada kekurangan pada Tugas Akhir ini. Penulis juga mengharapkan kritik dan saran yang membangun untuk pembelajaran dan perbaikan di kemudian hari. Semoga melalui Tugas Akhir ini penulis dapat memberikan kontribusi dan manfaat yang sebaik-baiknya.

Surabaya, Juni 2020

Raden Teja Kusuma

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR PSEUDOCODE

[*Halaman ini sengaja dikosongkan*]

DAFTAR KODE SUMBER

[*Halaman ini sengaja dikosongkan*]

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar tugas akhir yang meliputi latar belakang, tujuan, rumusan masalah, batasan permasalahan, metodologi pembuatan tugas akhir, dan sistematika penulisan buku tugas akhir ini.

1.1 Latar Belakang

Keamanan adalah sebuah aspek penting pada proses pertukaran data. Hal ini dikarenakan terdapat berbagai faktor yang dapat mengakibatkan terjadinya pengubahan atau penyadapan data oleh pihak yang tidak memiliki hak terhadap data tersebut. Untuk itu dibuat berbagai usaha untuk melakukan penyandian data atau pemeriksaan terhadap keutuhan dan keaslian data.

Prime Factorization dari sebuah bilangan sering digunakan dalam skema pengamanan data ketika membuat *public key* maupun *private key*. Untuk mengetahui pentingnya *prime factorization*, digunakan contoh skema keamanan RSA *Algorithm*. RSA merupakan algoritma keamanan yang sangat terkenal dan masih sering dipakai karena algoritma ini bisa menghasilkan *public key* dan *private key* yang cukup panjang sehingga lebih sulit ketika ingin melakukan penyadapan data. Nama RSA itu sendiri diambil dari nama penemunya yaitu Ron Rivest, Adi Shamir, dan Len Adleman. Pada RSA, dalam pembuatan *public key* dan *private key* mengimplementasikan *prime factorization* dan dilanjutkan dengan proses-proses selanjutnya sehingga bisa tercipta *public key* dan *private key* dalam RSA[1].

Pada Tugas Akhir ini akan dilakukan desain dan analisis mengenai algoritma *prime factorization* menggunakan studi kasus *SPOJ SQFFACT – Square-free Integers Factorization*[2]. Dalam permasalahan, diketahui bilangan bulat positif *N*, *euler's*

totient function dari N yaitu M, dan T yang merupakan banyaknya uji coba yang akan dilakukan.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana desain algoritma *prime factorization* untuk menyelesaikan masalah pada SPOJ SQFFACT?
2. Bagaimana analisis algoritma *prime factorization* untuk menyelesaikan masalah pada SPOJ SQFFACT?

1.3 Batasan Masalah

Permasalahan pada pembuatan Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut.

1. Penggunaan algoritma *prime factorization* dalam menyelesaikan permasalahan dalam studi kasus SPOJ SQFFACT.
2. Algoritma *prime factorization* yang dibahas terbatas pada analisis intuitif dan logis.

Pada kasus SPOJ SQFFACT – *Square-free Integers Factorization,* diberikan batasan sebagai berikut.

1. Implementasi algoritma menggunakan bahasa pemrograman Java.
2. Batas nilai T adalah bilangan natural pada rentang 0 sampai 10^2 inklusif.
3. Batas nilai N adalah bilangan bulat positif pada rentang 1 sampai 10^{100} eksklusif.
4. Batasan waktu adalah 1,355 detik.
5. Batasan memorinya adalah 1536 MB.
6. Batasan ukuran program adalah 20000 B.

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

1. Melakukan desain dan analisis algoritma *prime factorization* untuk menemukan solusi dari permasalahan *prime factorization* secara optimal dari soal SPOJ SQFFACT – *Square-free Integers Factorization*.
2. Melakukan implementasi algoritma *prime factorization* untuk menemukan solusi dari permasalahan *prime factorization* secara optimal dari soal SPOJ SQFFACT – *Square-free Integers Factorization*.

1.5 Manfaat Tugas Akhir

Tugas Akhir ini dapat membantu memahami penggunaan algoritma *prime factorization* dan penerapannya untuk menemukan solusi dari permasalahan *prime factorization* secara optimal dari soal SPOJ SQFFACT – *Square-free Integers Factorization.*

1.6 Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini yaitu:

1.6.1 Penyusunan Proposal Tugas Akhir

Tahap awal yang dilakukan untuk memulai pengerjaan Tugas Akhir adalah dengan menyusun proposal Tugas Akhir. Dalam proposal, penulis mengajukan gagasan yang digunakan untuk menyelesaikan permasalahan *prime factorization* secara optimal dari soal SPOJ SQFFACT – *Square-free Integers Factorization.*

1.6.2 Studi Literatur

Pada tahap ini dilakukan pencarian informasi dan studi literatur yang relevan untuk dijadikan referensi dalam melakukan pengerjaan Tugas Akhir. Informasi didapatkan dari materi-materi yang berhubungan dengan struktur data. Informasi tersebut didapatkan dari buku, internet, dan materi kuliah yang berhubungan dengan metode yang akan digunakan.

1.6.3 Desain dan Analisis

Pada tahap ini akan dilakukan desain rancangan algoritma untuk memecahkan permasalahan *Prime Factorization*. Luaran dari tahap ini adalah algoritma yang digunakan sebagai solusi optimal pada permasalahan SPOJ SQFFACT – *Square-free Integers Factorization.*

1.6.4 Implementasi Algoritma

Tahapan ini merupakan tahapan untuk membangun algoritma yang akan digunakan. Pada tahap ini dilakukan implementasi dari rancangan struktur data yang akan dimodelkan sesuai dengan permasalahan. Implementasi ini dilakukan dengan menggunakan bahasa pemrograman Java.

1.6.5 Pengujian dan Evaluasi

Pada tahap ini dilakukan uji coba kebenaran dan kinerja solusi yang telah diimplementasikan dengan melakukan pengiriman sumber kode sistem ke situs penilaian *Sphere Online Judge (SPOJ)* pada permasalahan yang terkait unutk mendapatkan umpan balik. Pengujian dilakukan dengan membandingkan kompleksitas hasil uji coba dengan kompleksitas hasil analisis.

1.6.6 Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas

akhir ini serta hasil dari implementasi algoritma yang telah dibuat.

1.7 Sistematika Penulisan

Buku Tugas Akhir ini merupakan laporan secara lengkap mengenai Tugas Akhir yang telah dikerjakan baik dari sisi teori, rancangan, maupun implementasi sehingga memudahkan bagi pembaca dan juga pihak yang ingin mengembangkan lebih lanjut. Sistematika penulisan buku Tugas Akhir secara garis besar antara lain:

Bab I Pendahuluan

Bab ini berisi penjelasan latar belakang, rumusan masalah, batasan masalah dan tujuan pembuatan Tugas Akhir. Selain itu, metodologi pengerjaan dan sistematika penulisan laporan Tugas Akhir juga terdapat di dalamnya.

Bab II Dasar Teori

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

Bab III Desain dan Analisis

Bab ini berisi penjelasan tentang rancangan dari sistem yang akan dibangun.

Bab IV Implementasi

Bab ini berisi penjelasan implementasi dari rancangan yang telah dibuat pada bab sebelumnya. Implementasi disajikan dalam bentuk kode sumber disertai dengan penjelasannya.

Bab V Uji Coba dan Evaluasi

Bab ini berisi penjelasan mengenai data hasil percobaan dan pembahasan mengenai hasil percobaan yang telah dilakukan.

Bab VI Kesimpulan dan Saran

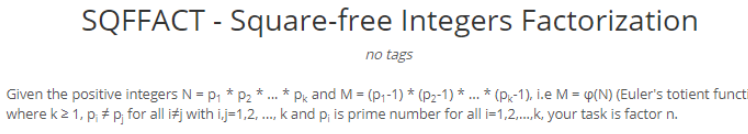
Bab ini merupakan bab terakhir yang menjelaskan kesimpulan dari hasil ujicoba yang dilakukan dan saran untuk pengembangan algoritma kedepannya.

BAB II
DASAR TEORI

Pada bab ini dijelaskan mengenai dasar-dasar teori yang akan digunakan untuk menyelesaikan permasalahan dalam tugas akhir ini. Terlebih dahulu akan dijelaskan mengenai permasalahan SPOJ SQFFACT. Dasar teori yang akan digunakan meliputi *Euler's totient function, Carmichael's function, Euler's theorm*, serta penjabaran penggunaan teori dan landasan pemilihan struktur data yang akan digunakan sebagai solusi permasalahan tersebut.

2.1 Deskripsi Permasalahan SPOJ SQFFACT

SQFFACT – *Square-Free Integers Factorization* [2], merupakan suatu permasalahan yang terdapat pada situs penilaian Sphere Online Judge (SPOJ) dengan deskripsi soal dari sumber asli menggunakan bahasa Inggris, yang dapat dilihat pada Gambar 2.1.


SQFFACT - Square-free Integers Factorization

no tags

Given the positive integers $N = p_1 * p_2 * ... * p_k$ and $M = (p_1-1) * (p_2-1) * ... * (p_k-1)$, i.e $M = \varphi(N)$ (Euler's totient function), where $k \geq 1$, $p_i \neq p_j$ for all $i \neq j$ with $i,j=1,2, ..., k$ and p_i is prime number for all $i=1,2,...,k$, your task is factor n.

Gambar 2.1 Deskripsi Soal SPOJ SQFFACT – Square-free Integers Factorization

Permasalahan SPOJ SQFFACT – *Square-free Integers Factorization* ialah permasalahan dimana kita diharuskan mencari faktor prima dari sebuah bilangan N dengan diketahui nilai *euler's totient* dari N ($\varphi(N)$).

2.1.1 Parameter Masukan

Berdasarkan deskripsi soal dari SPOJ, permasalahan yang diberikan memiliki parameter masukan dengan batasan sebagai berikut.

1. N, bilangan bulat yang berperan sebagai bilangan yang akan dicari faktor primanya.

$$N < 10^{100} \tag{2.1}$$

2. M, bilangan bulat yang berperan sebagai *euler's totient dari N* ($\varphi(N)$).
3. T, bilangan bulat yang berperan sebagai jumlah uji coba yang dilakukan dalam permasalahan ini.

$$T \leq 100 \tag{2.2}$$

Masukan dari permasalahan ini dimulai dengan T dilanjutkan dengan memasukan nilai N dan nilai M. Contoh masukan pada permasalahan ini dapat dilihat pada Gambar 2.2.

```
Input:
3
210 48
983 982
14351 14112
```

Gambar 2.2 Contoh masukan pada permasalahan SPOJ SQFFACT

2.1.2 Keluaran Permasalahan

Keluaran dari permasalahan ini adalah nilai dari variabel N dan diikuti dengan hasil faktor primanya. Berikut adalah contoh keluaran pada permasalahan ini dapat dilihat pada Gambar 2.3.

```
Output:
210 = 2 * 3 * 5 * 7
983 = 983
14351 = 113 * 127
```

Gambar 2.3 Contoh keluaran pada permasalahan SPOJ SQFFACT

2.2 Deskripsi Umum

Pada subbab ini akan memaparkan definisi, deskripsi, dan landasan yang digunakan dalam penyelesaian permasalahan yang diberikan.

2.2.1 Fermat's Little Theorem

Fermat's little theorem adalah sebuah teorema dasar dalam teori bilangan, yang mana teori ini membantu dalam menyelesaikan permasalahan dalam mencari bilangan prima.

Misalkan p adalah bilangan prima. Maka untuk sembarang bilangan bulat a berlaku Persamaan 2.3[5].

$$a^p \equiv a \ (mod\ p) \tag{2.3}$$

Untuk $gcd(a, p) = 1$, Persamaan 2.3 ekivalen dengan persamaan 2.4.

$$a^{p-1} \equiv 1 \ (mod\ p) \tag{2.4}$$

Teorema ini digunakan sebagai dasar penggunaan *Euler's theorem* (yang dijelaskan dalam subbab 2.2.2). Karena teorema ini dijadikan bentuk umum oleh *Euler's theorem.*

2.2.2 Euler's Theorem

Euler's theorem adalah sebuah teori bilangan yang menyatakan jika n dan a merupakan koprima, maka akan memenuhi Persamaan 2.4 yang mana bisa ditulis menjadi Persamaan 2.5[3].

$$a^{\varphi(n)} \equiv 1 \ (mod\ n) \tag{2.5}$$

Dimana $\varphi(n)$ adalah *Euler's totient function* (yang akan dijelaskan pada subbab 2.2.3). Teorema ini akan digunakan dalam *Carmichael's function* sebagai persamaan awal karena teorema ini dijadikan bentuk umum dari *Carmichael's function.*

2.2.3 Euler's Totient Function

Di dalam teori bilangan *euler's totient function* menghitung bilangan bulat positif hingga bilangan bulat n yang relatif prima terhadap n. Dengan kata lain bilangan bulat k dalam *range* $1 \leq k \leq n$ untuk $gcd(n,k) = 1$[6].

Berikut ini adalah formula produk dari *Euler's totient function* yaitu dalam Persamaan 2.6.

$$\varphi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right) \tag{2.6}$$

Berdasarkan Persamaan 2.6 itu terdapat dua fakta penting dalam membuktikan formula tersebut.

Fakta yang pertama ialah fungsi ini merupakan *multiplicative function* sehingga jika $gcd(m,n) = 1$, maka $\varphi(mn) = \varphi(m)\varphi(n)$.

Fakta yang kedua adalah untuk prima yang berpangkat. Jika p adalah prima dan $k \geq 1$, maka

$$\varphi(p^k) = p^{k-1}(p-1) = p^k \left(1 - \frac{1}{p}\right) \tag{2.7}$$

Buktinya karena p adalah bilangan prima maka nilai yang mungkin dari $gcd(p^k, m)$ adalah $1, p, p^2, \cdots, p^k$, dan salah satu cara agar hasil $gcd(p^k, m)$ itu tidak sama dengan 1 jadi untuk nilai m haruslah kelipatan dari p. Kelipatan dari p yang kurang dari atau sama dengan p^k adalah $p, 2p, 3p, \cdots, p^{k-1} = p^k$. Oleh karena itu angka antara $p^k - p^{k-1}$ itu relatif prima terhadap p^k.

Misalkan $\varphi(n)$ merupakan jumlah angka k, dengan $1 \leq k \leq n$, sehingga $\gcd(n,k) = 1$. Dimana ini jelas, jika $\gcd(n,k) = 1$, maka $\gcd(n-k,n) = 1$ juga. Jadi (untuk $n > 2$) semua angka yang relatif prima ke n dapat dicocokan menjadi pasangan $\{k, n-k\}$. Jadi hasil dari $\varphi(n)$ adalah genap.

Ini digunakan untuk mengetahui bagaimana sifat-sifat yang dimiliki oleh *Euler's totient function* tersebut sehingga bisa digunakan dalam memfaktorkan sebuah bilangan sehingga menemukan bilangan prima dari bilangan tersebut.

2.2.4 Group

Didefinisikan himpunan bilangan G disebut sebagai *group* dengan operasi perkalian (*multiplicative group)* jika memenuhi kondisi-kondisi berikut[1].

1. Asosiatif, untuk $g_i, g_j \in G$ berlaku $g_i * g_j = g_j * g_i$.
2. Memiliki elemen indentitas $e \in G$ sedemikian sehingga untuk setiap $e \in G$ berlaku $g * e = g$.
3. Untuk setiap elemen $g \in G$ terdapat invers $g^{-1} \in G$ sedemikian sehingga $g * g^{-1} = e$.

Dari definisi tersebut, maka sifat dari *multiplicative group* ini dapat diaplikasikan pada *multiplicative order* yang akan dibahas pada subbab 2.2.6.

2.2.5 Order

Order dari *group* G adalah banyaknya elemen $g \epsilon G$ dan dinotasikan dengan $|G|$. *Group* $\mathbb{Z}_p^*$ dengan p bilangan prima memiliki elemen $h = \{1,2,\dots,p-1\}$. Sehingga *order* dari $\mathbb{Z}_p^*$ adalah $p-1$[1]. *Group order* ini akan digunakan pada *multiplicative order* yang akan dibahas pada subbab 2.2.6.

2.2.6 Multiplicative Order

Diberikan sebuah bilangan bulat a dan sebuah bilangan bulat positif n yang mana koprima terhadap a, sehingga hasil dari

multiplicative order a modulus n adalah sebuah bilangan bulat positif terkecil k[5].

$$a^k \equiv 1 \ (mod \ n) \tag{2.8}$$

Jadi dengan kata lain, *multiplicative order* dari a modulus n adalah *order* didalam *multiplicative group.*

2.2.7 Carmichael's Function

Di dalam teori bilangan *Carmichael's function* berhubungan dengan setiap bilangan bulat positif n yang mana pada fungsi ini disimbolkan dengan $\lambda(N)$ yang didefinisikan sebagai bilangan bulat positif terkecil m sehingga memenuhi Persamaan 2.9 [4].

$$a^m \equiv 1 \ (mod \ n) \tag{2.9}$$

Dimana untuk setiap bilangan bulat a diantara 1 dan n merupakan koprima dari n. Pada fungsi ini juga menjelaskan bagaimana cara menghitung $\boldsymbol{\lambda}$ dari bilangan prima berpangkat p^r. Untuk hasil pemangkatan prima dengan prima ganjil ditambah 2 dan 4 sama dengan *Euler's totient* $\varphi(p^r)$. Sedangkan unuk pemangkatan 2 selain 2 dan 4 sama dengan setengah dari *Euler's totient* $\varphi(p^r)$. Sehingga hasil dari *Carmichael's function* selalu merupakan keliapatan dari *Euler's totient function.*

$$\boldsymbol{\lambda}(p^r) = \begin{cases} \varphi(p^r) \ if \ p^r = 2,3,4,5, \dots \\ \frac{1}{2}\varphi(p^r) \ if \ p^r = 8,16,32,64, \dots \end{cases} \tag{2.10}$$

Sehingga *Euler's function* untuk bilangan prima dengan p^r adalah sama seperti Persamaan 2.7.

Misalkan *Carmichael's function* didefinisikan jika $p_1, \dots, p_n$ merupakan prima yang berbeda maka akan memenuhi persamaan 2.11

$$\lambda(p_1^{k_1} \dots p_n^{k_n}) = lcm(\lambda(p_1^{k_1}), \dots, \lambda(p_n^{k_n})) \quad (2.11)$$

Lalu jika p adalah prima ganjil dan nilai $k \geq 1$ maka akan memenuhi persamaan 2.12.

$$\lambda(p^k) = p^{k-1}(p-1) \quad (2.12)$$

Lalu $\lambda(2) = 1$, $\lambda(4) = 2$ maka untuk $k \geq 3$ akan memenuhi persamaan 2.13.

$$\lambda(2^k) = 2^{k-2} \quad (2.13)$$

Sehingga nilai dari *Carmichael's function* akan selalu bernilai genap kecuali untuk $\lambda(2)$ dan $\lambda(1)$. Dimana semua informasi tersebut digunakan untuk merumuskan persamaan untuk memecahkan permasalahan SPOJ SQFFACT.

2.2.8 Randomized Algorithm

Randomized algorithm adalah sebuah algoritma dimana menggunakan bilangan acak atau *random x* dalam rentang angka yang sudah ditentukan dan akan memproses angka *random* tersebut dan membuat keputusan berdasarkan hasilnya. *Randomized algoritma* itu biasanya digunakan untuk mengurangi *running time*, kompleksitas waktu, dan memori yang digunakan dalam algoritma standar.

Randomized algorithm ini dipakai penulis dalam menyelesaikan permasalahan yang ada. Karena berdasarkan strategi yang digunakan penulis yang akan dijelaskan pada subbab 3.1 menerangkan dalam penyelesaian masalahnya penulis menggunakan bilangan *random* dengan rentang yang telah ditentukan. Yang mana bilangan *random* tersebut digunakan dan diproses sedemikian rupa sehingga bisa mengetahui hasil faktor prima dari sebuah bilangan yang telah dimasukkan.

[*Halaman ini sengaja dikosongkan*]

BAB III
DESAIN DAN ANALISIS

Pada bab ini akan dijelaskan mengenai stategi dalam menyelesaikan permasalahan dan desain program yang akan digunakan untuk menyelesaikan permasalahan yang diberikan pada Tugas Akhir ini.

3.1 Strategi Penyelesaian

Pada bagian ini akan membahas mengenai strategi penyelesaian permasalahan SPOJ SQFFACT – *Square-free Integers Factorization*. Strategi penyelesaian permasalahan ini adalah sebagai berikut.

Bagi N dengan 2 hingga tidak bisa dibagi lagi dengan memperhatikan langkah-langkah sebagai berikut.

1. Cek jika N adalah bilangan prima dengan cara cek selisih antara nilai N dengan M yang mana sudah diketahui M merupakan $\varphi(N)$. Jika selisihnya 1 maka N merupakan bilangan prima[3].
2. Ambil bilangan sembarang antara $[2, N-1]$. Sebut nilai sembarang itu dengan x. Lalu cek $gcd(N, x)$, jika hasilnya > 1 maka akan menemukan faktornya tetapi jika hasilnya 1 maka x dan N adalah koprima. Lalu dengan menggunakan Persamaan 3.1.

$$x^{\varphi(N)} \equiv 1 \ (mod\ N) \tag{3.1}$$

Dari Persamaan 3.1 bisa menuliskan nilai $\varphi(N)$ menjadi Persamaan 3.2.

$$\varphi(N) = 2^K * t \tag{3.2}$$

Persamaan tersebut untuk menemukan nilai terkecil dari $k \leq K$ yang memenuhi ekivalen dengan Persamaan 3.3.

$$x^{2^k * t} \equiv 1 \, mod \, N \quad (3.3)$$

Karena $\varphi(N)$ menghasilkan nilai genap untuk semua angka yang lebih dari sama dengan 2, maka nilai $K > 1$. Jika sudah menemukannya, maka akan mendapatkan nilai dari Persamaan 3.3. Maka nilai a bisa ditulisakan menjadi Persamaan 3.4.

$$a = x^{2^{k-1} * t} \quad (3.4)$$

Dari Persamaan 3.4 itu bisa mendapatkan Persamaan 3.5 – Persamaan 3.7.

$$a^2 \equiv 1 \, mod \, N \quad (3.5)$$
$$(a^2 - 1) \equiv 0 \, mod \, N \quad (3.6)$$
$$(a - 1) * (a + 1) \equiv 0 \, mod \, N \quad (3.7)$$

Setelah itu lakukan pengecekan $gcd(x^k \pm 1, N)$ dan lanjutkan proses faktorisasinya. Jika tidak memenuhi persamaannya maka selanjutnya hitunglah $X = x^M \, mod \, N$. Jika hasil nilai $X + 1 = N$ atau $X = 1$ maka coba nilai x yang lain. Jika masih tidak memenuhi maka hitung $nw = X * X \, mod \, N$. Selama nilai $nw \neq 1$ maka nilai $X = nw$ dan lalu hitung ulang nilai nw. Jika nilai $nw = 1$ maka selanjutnya cek jika $X + 1 \neq N$ maka buat variabel $fac = gcd(N, X - 1)$ dan lalu rekursikan dengan parameter fac dan rekursikan dengan parameter N/fac.

3.2 Penjelasan Strategi Penyelesaian

Pada bagian ini akan membahas mengenai penjelasan terhadap strategi penyelesaian permasalahan yang telah disampaikan pada subbab 3.1.

Dari permasalahan bisa ketahui nilai N adalah nilai yang akan dicari faktor primanya. Selain itu juga mengetahui $\varphi(N)$

dengan demikian kita juga bisa mengetahui nilai dari $k\boldsymbol{\lambda}(N)$[4]. Dimana $\boldsymbol{\lambda}(N)$ adalah merupakan *Carmichael's function.*

Dari *Euler's theorem* dapat diketahui Persamaan 2.5 yang mana dari persamaan tersebut bisa mendapatkan nilai a dan N yang koprima. Karena pada *Carmichael's function* itu akan selalu menghasilkan nilai genap berdasarkan Persamaan 2.13, maka dari Persamaan 2.9 bisa ditulis menjadi Persamaan 3.8 – 3.10.

$$a^{2k} \equiv 1 \; mod \; N \tag{3.8}$$

$$a^{2k} - 1 \equiv 0 \; mod \; N \tag{3.9}$$

$$(a^k - 1) * (a^k + 1) \equiv 0 \; mod \; N \tag{3.10}$$

Dari penjelasan *Carmichael's function* pada subbab 2.2.7 didefinisikan dengan m. Karena nilai m ini selalu genap maka bisa ditulis menjadi Persamaan 3.11.

$$m = 2^k * b \; ; \; dimana \; b \; ganjil \tag{3.11}$$

Selanjutnya mengambil nilai sembarang x yang berada diantara $[2, N-1]$. Jika x membagi N maka akan menemukan faktor dari N karena Persamaan 2.9. *Carmichael's function* adalah m yang mana setiap angka dalam *range* $[1, N-1]$ koprima dengan N akan menghasilkan nilai 1 sebagai sisa dari modulus. Tetapi untuk nilai tertentu, x dalam kasus ini bisa ada nilai k yang lebih kecil sehingga memenuhi Persamaan 2.8.

Selanjutnya akan mengeksplorasi Persamaan 3.12.

$$x^{2k} \equiv 1 \; mod \; N \tag{3.12}$$

Dimana dalam Persamaan 3.12 itu sudah merepresentasikan Persamaan 3.11. Selanjutnya akan mencari nilai terkecil dari m sehingga terbentuk Persamaan 3.13.

$$x^m \equiv 1 \; mod \; N \tag{3.13}$$

Yang mana itu bisa dilakukan dengan cara membagi $\varphi(N)$ dengan 2 hingga tidak bisa lagi dibagi. Karena sudah diketahui jika $\varphi(N)$ selalu merupakan kelipatan dari $\lambda(N)$. **S**ebut nilai tersebut dengan q. Karena nilai q selalu genap maka bisa ditulis menjadi $q = 2k$. Sehingga bisa menemukan Persamaan 3.12 dan Persamaan 3.12 itu juga bisa ditulis menjadi Persamaan 3.14.

$$(a^k - 1) * (a^k + 1) \equiv 0 \; mod \; N \tag{3.14}$$

Berdasarkan Persamaan 3.14 itu akan memunculkan dua kasus baru. Salah satu kasus ini dianggap sebagai *bad case* sedangkan satunya dianggap sebagai *good case.*

1. Antara $(x^k - 1)$ atau $(x^k + 1)$ adalah kelipatan dari N. Yang mana memenuhi Persamaan 3.15.

   $$x^k \equiv \pm 1 \; mod \; N \tag{3.15}$$

   Ini merupakan *bad case* karena tidak ada lagi informasi yang bisa didapatkan.
2. Antara $(x^k - 1)$ atau $(x^k + 1)$ adalah kelipatan dari N. Yang mana memenuhi Persamaan 3.16.

   $$x^k \not\equiv \pm 1 \; mod \; N \tag{3.16}$$

   Ini merupakan *good case* karena baik $(x^k - 1)$ dapat dibagi dengan N, begitu juga dengan $(x^k + 1)$. Yang ini berarti terdapat beberapa faktor N yang sama untuk N dan $(x^k \pm 1)$, dan untuk faktor yang direpresentasikan dalam $(x^k - 1)$, faktor komplemennya harus direpresentasikan dalam $(x^k + 1)$. Karena N dan $(x^k \pm 1)$ memiliki kesamaan faktor maka perlu melakukan $gcd(x^k \pm 1, N)$ untuk menemukan faktornya. Mari sebut hasil dari GCD itu dengan g. Selanjutnya lakukan rekursi N/g untuk mendapatkan faktor primanya. Selanjutnya hitunglah $X = x^M \; mod \; N$. Jika hasil

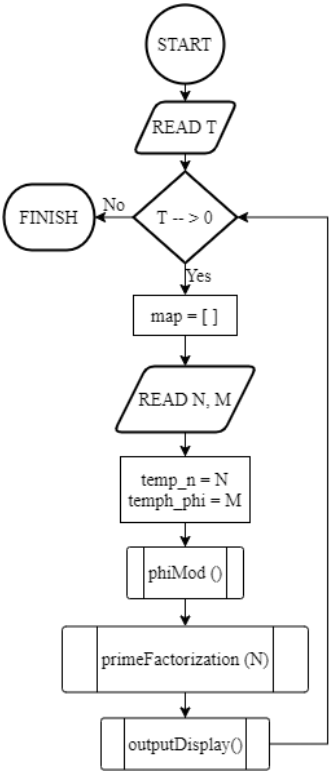
nilai $X + 1 = N$ atau $X = 1$ maka coba nilai x yang lain. Jika masih tidak memenuhi maka hitung $nw = X^2 \ mod \ N$. Selama nilai $nw \neq 1$ maka nilai $X = nw$ dan lalu hitung ulang nilai nw. Jika nilai $nw = 1$ maka selanjutnya cek jika $X + 1 \neq N$ maka buat variabel $fac = gcd(N, X - 1)$ dan lalu rekursikan dengan parameter fac dan rekursikan dengan parameter N/fac.

Dari cara yang digunakan tersebut juga terdapat beberapa *nasty case* yang muncul sebagai berikut.

1. Jika N merupakan bilangan prima. Solusinya adalah dengan melihat selisih antara nilai N dengan nilai $\varphi(N)$, jika selisihnya 1, maka N merupakan bilangan prima[3].
2. Jika N merupakan bilangan genap maka N tidak akan koprima dengan hampir setengah nilai dalam rentang $[1, N - 1]$. Ini bukan menjadi masalah karena kita memiliki banyak kesempatan dalam "menebak" nilai x sehingga $gcd(x, N) > 1$ tetapi akan lebih menghemat waktu jika kita membagi terus N dengan x.

3.3 Deskripsi Umum Sistem

Sistem menerima masukan berupa tiga jenis bilangan yaitu T, N, dan M yang telah dijelaskan fungsi dan batasannya pada subbab 2.1.1. Setelah menerima ketiga masukan tersebut sistem akan menjalankan fungsi *phiMod()*. Setelah selesai maka sistem akan menjalankan fungsi *primeFactorization()* untuk mencari faktor prima dari N. Setelah berhasil menemukan faktor prima dari N proses selanjutnya adalah menampilkan keluaran dengan menjalankan fungsi *outputDisplay()*. Visualisasi alur jalannya sistem dapat dilihat pada Gambar 3.1.



Gambar 3.1 Alur Program

3.4 Desain Program Utama

Pada subbab ini akan dijelaskan desain fungsi dalam program yang digunakan untuk menyelesaikan permasalahan.

3.4.1 Desain Tipe Data

Berdasarkan pada subbab 2.1.1 bisa diketahui batasan dalam setiap variabel yang ada. Sehingga untuk variabel N serta M harus menggunakan tipe data *BigInteger*. Karena batasan untuk N adalah 10^{100} yang mana ini tidak akan bisa ditampung jika menggunakan tipe data seperti *int* ataupun *long long* karena dalam tipe data *int* hanya mampu menampung sekitar -10^9 sampai 10^9 saja sedangkan pada tipe data *long long int* hanya mampu menampung sekitar -10^{18} sampai 10^{18}. Sedangkan untuk variabel T menggunakan tipe data *int* karena batasannya hanya $T \leq 100$ sehingga masih bisa ditangani oleh *int.*

3.4.2 Desain Struktur Data

Untuk desain struktur data sistem ini menerapkan salah satu jenis *map* yaitu *TreeMap*. *TreeMap* adalah sebuah *map* yang mengimplementasikan *SortedMap* dan *NavigableMap*. Sehingga *TreeMap* bisa menggunakan fungsionalitas dari *SortedMap* maupun *NavigableMap*. Berikut adalah beberapa poin penting mengenai *TreeMap*.

1. *TreeMap* selalu tersortir berdasarkan *key*. Urutan dalam penyortirannya mengikuti *natural ordering* dari *key*. Tetapi juga diperbolehkan untuk mengubah *comparator* pada *TreeMap* pada saat mendeklarasikan sehingga akan mengurutkan berdasarkan *comparator* tersebut.
2. *TreeMap* tidak boleh terdapat *key* yang sama.
3. *TreeMap* tidak boleh terdapat *null key*. Tetapi boleh terdapat nilai yang *null*.

3.4.3 Desain Fungsi Main

Berdasarkan deskripsi umum mengenai sistem pada subbab 3.1. berikut ini merupakan desain dari fungsi ini yang dituliskan pada *Pseudocode* 3.1

Fungsi MAIN

```
1:  Input: T
2:  rand ←RANDOM()
3:  while (T − −) > 0 do
4:      map ← [ ]
5:      N, M ←INPUT()
6:      temp_n = N
7:      temp_phi = M
8:      phiMod()
9:      primeFactorization(N)
10:     outputDisplay()
```

Pseudocode 3.1 Fungsi MAIN

3.4.4 Desain Fungsi *phiMod*

Fungsi *phiMod* digunakan untuk melihat jika nilai dari M ketika di modulus oleh 2 menghasilkan nilai 0 maka nilai M tersebut akan dibagi dengan 2. Fungsi ini diperlukan untuk membantu pencarian faktor prima dari N karena tidak semua bilangan bisa langsung dicari faktor primanya terutama untuk M yang bernilai genap atau M yang merupakan *perfect power*.

Fungsi PHIMOD

```
1:  Input: -
2:  phi_divide = 2
3:  while (M mod phi_divide = 0)
4:      M = M/phi_divide
```

Pseudocode 3.2 Fungsi PHIMOD

3.4.5 Desain Fungsi primeFactorization

Fungsi *primeFactorization* digunakan untuk memfaktorkan dari masukan sehingga bisa mendapatkan hasil yang berupa bilangan prima dari masukkan yang diberikan. Hal pertama yang akan dilakukan ialah dengan mengecek selisih antara nilai N dengan M. Selanjutnya fungsi tersebut akan melakukan beberapa kali perulangan. Didalam perulangan tersebut akan diterapkan *randomized algorithm* dimana sistem akan mengambil sebuah nilai *random* dari rentang yang sudah ditentukan. Nilai *random* tersebut lalu akan digunakan kedalam beberapa persamaan sehingga fungsi ini bisa menentukan berapa saja faktor prima dari nilai N. Berikut adalah desain dari fungsi ini yang dituliskan pada *Pseudocode* 3.3 – *Pseudocode* 3.4.

Fungsi PRIMEFACTORIZATION (bagian 1)

```
1:  Input: N
2:  prime ← N
3:  if (temp_phi + 1) == prime
4:      map[ ] ← (prime, 1)
5:      return
6:  for i ← 0 to 10
7:      x = rand
8:      x = x mod prime
9:      if (x == 0)
10:         i--
11:         continue
12:         g = gcd(x, prime)
13:         if ((! g == 1) and (! g == prime))
14:                 primeFactorization(g)
15:             primeFactorization(prime / g)
16:         return
17:         X = x^M mod prime
```

Pseudocode 3.3 Fungsi PRIMEFACTORIZATION (bagian 1)

Fungsi PRIMEFACTORIZATION (bagian 2)

```
18:     if (((X + 1) == prime) or (X == 1))
19:     continue
20:     nw = X * X mod prime
21:     while (! nw == 1)
22:         X = nw
23:         nw = nw * nw mod prime
24:     if (! (X + 1) == prime)
25:         fac = gcd(prime, X − 1)
26:         primeFactorization(fac)
27:         primeFactorization(prime/fac)
28:         return
29: map[ ] ←(prime, 1)
```

Pseudocode 3.4 Fungsi PRIMEFACTORIZATION (bagian 2)

3.4.6 Desain Fungsi o*utputDisplay*

Fungsi *outputDisplay* digunakan untuk menampilkan hasil akhir setiap uji coba yang diberikan. Berikut adalah desain dari fungsi ini yang dituliskan pada *Pseudocode* 3.5

Fungsi OUTPUTDISPLAY

```
1:  Input: -
2:  print temp_n + " = "
3:  was = false
4:  for map[ ]
5:      if (was)
6:          print " * "
7:      else was = true
8:      print " "
9:  print newline
```

Pseudocode 3.5 Fungsi OUTPUTDISPLAY

BAB IV
IMPLEMENTASI

Pada bab ini akan dijelaskan tentang implementasi yang dilakukan berdasarkan algoritma yang telah dirancang pada bab sebelumnya.

4.1 Lingkungan Implementasi

Lingkungan implementasi dan pengembangan yang dilakukan adalah sebagai berikut.

1. Perangkat Keras
   (a) Processor: Intel® Core™ i7-4720HQ CPU @ 2.60GHZ (8 CPUs), ~2.6GHz
   (b) Memory: 8192 RAM
2. Perangkat Lunak
   (a) Sistem Operasi: Windows 10 Pro 64-bit
   (b) *Text Editor:* Netbeans dan Visual Studio Code
   (c) Bahasa Pemrograman: Java

4.2 Penggunaan *Library*, Konstanta, dan Variabel Global

Subbab ini menjelaskan mengenai *library*, konstanta, dan variabel global yang digunakan dalam sistem. Implementasi algoritma membutuhkan empat buah *library*, yaitu *java.util.*, java.lang.*, java.io.*,* dan *java.math.*. Library java.util.** digunakan supaya kita bisa menggunakan *TreeMap,* mendapatkan bilangan *random,* dll. *Library java.lang.** menyediakan untuk kelas dengan nama *Class,* juga menyediakan tipe data primitif (*boolean, byte, char, int,* dsb). *Library java.io.** digunakan untuk menerima masukan dan menampilkan keluaran. *Library java.math.** digunakan untuk membantu dalam operasi numerik seperti kuadrat, akar kuadrat, dan yang paling penting ialah untuk

bisa menggunakan *BigInteger*. *Library* yang digunakan akan ditampilkan pada Kode Sumber 4.1.

```
1  import java.util.*;
2  import java.lang.*;
3  import java.io.*;
4  import java.math.*;
```

Kode Sumber 4.1 *Library* yang digunakan

Variabel global digunakan untuk memudahkan dalam mengakses data yang digunakan antar fungsi, seperti pada Kode Sumber 4.2. Variabel N merupakan nilai yang akan dicari faktor primanya, variabel M merupakan *euler's totient* dari N, variabel $temp\_phi$ digunakan untuk menyimpan nilai awal dari M, variabel $temp\_n$ digunakan untuk menyimpan nilai awal dari N, variabel phi_divide digunakan untuk menyimpan nilai tertentu dan nanti untuk membagi dari nilai M, variabel map merupakan deklarasi dari *NavigableMap*, variabel $rand$ digunakan untuk mendapatkan nilai *random*, variabel fin digunakan untuk menangkap masukkan, dan variabel $fout$ digunakan untuk mencetak keluaran.

```
1  static NavigableMap<BigInteger,Integer> map;
2  static BigInteger N, M, temp_phi,
   phi_divide,temp_n;
3  Static Random rand;
4  static BufferedReader fin;
5  static PrintWriter fout;
```

Kode Sumber 4.2 Variabel global yang digunakan

4.3 Implementasi Desain Algoritma

Pada subbab ini akan dijelaskan mengenai implementasi proses algoritma secara keseluruhan berdasarkan desain yang telah dijelaskan pada bab 3.

4.3.1 Implementasi Fungsi Main

Fungsi *main* adalah implementasi Algoritma yang dirancang pada subbab 3.2.3. Implementasi fungsi *main* dapat dilihat pada Kode Sumber 4.3.

```
 1  Public static void main(String[] args)
    throws java.lang.Exception
 2  {
 3      fin = new BufferedReader(new
        InputStreamReader(System.in));
 4      fout = new PrintWriter(new
        OutputStreamWriter(System.out), true);
 5      Rand = new Random();
 6      Int testcase =
        Integer.parseInt(fin.readLine());
 7      while(testcase-- > 0)
 8      {
 9           map = new TreeMap<BigInteger,
             Integer>();
10           String s[] = fin.readLine().split(“
             ”);
11           N = new BigInteger(s[0]);
12           M = new BigInteger(s[1]);
13           temp_n = N;
14           temp_phi = M;
15           phiMod();
16           primeFactorization(N);
17          outputDisplay();
18      }
19      fin.close();
20      fout.close();
21  }
```

Kode Sumber 4.3 Implementasi Fungsi Main

4.3.2 Implementasi Fungsi *phiMod*

Fungsi *phiMod* digunakan untuk mengecek apakah jika nilai *M* dimodulus dengan 2 jika menghasilkan nilai 0 maka nilai *M* akan dibagi menjadi 2. Implementasi fungsi *phiMod* dapat dilihat pada Kode Sumber 4.4.

```
1  public static void phiMod()
2  {
3      phi_divide = BigInteger.valueOf(2);
4      while (M.mod(phi_divide)
       .equals(BigInteger.ZERO))
5          M = M.divided(phi_divide);
6  }
```

Kode Sumber 4.4 Implementasi Fungsi *phiMod*

4.3.3 Implementasi Fungsi *primeFactorization*

Fungsi *primeFactorization* digunakan untuk mencari berapa saja faktor prima dari *N* seperti yang sudah dijelaskan pada Pseudocode 3.2.5. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 4.5 dan Kode Sumber 4.6.

```
1  private static void
   primeFactorization(BigInteger prime){
2      if (temp_phi.add
       (BigIntegerOne).equal(N)){
3      map.put(prime,1);
4          return;
5      }
6      for (int i =0; i < 10; i++){
7          BigInteger x = new BigInteger
           (20,rand);
8          x = x.mod(prime);
9          if (x.equals(BigInteger.ZERO)){
10             i--;
11             continue;
12         }
```

Kode Sumber 4.5 Implementasi Fungsi *primeFactorization* (bagian 1)

```
13            BigInteger g = x.gcd(prime);
14            if   (!g.equals(BigInteger.ONE)   &&
              !g.equals(prime)){
15                 primeFactorization(g);
16                 primeFactorization(prime
                   .divided(g));
17                 return;
18            }
19            BigInteger X = x.modPow(M, prime);
20            if (X.add(BigInteger.One)
              .equals(prime) ||
              X.equals(BigInteger.ONE))
21                 continue;
22            BigInteger   nw   =   X.multiply(X)
              .mod(prime);
23            while (!nw.equals(BigInteger.ONE)){
24                 X = nw;
25                 nw = nw.multiply(nw)
                   .mod(prime);
26            }
27            if (!X.add(BigInteger.ONE)
              .equals(prime)){
28                 BigInteger fac = prime
                   .gcd(X.subtract
                   (BigInteger.ONE));
29                 primeFactorization(fac);
30                 primeFactorization(prime
                   .divided(fac));
31                 return;
32            }
33       }
34       map.put(prime,1)
35  }
```

Kode Sumber 4.6 Implementasi Fungsi *primeFactorization* (bagian 2)

4.3.4 Implementasi Fungsi *outputDisplay*

Fungsi *outputDisplay* ini digunakan untuk menampilkan isi yang berada dalam *map*. Implementasi dari fungsi ini bisa dilihat pada Kode Sumber 4.7.

```
 1  public static void outputDisplay()
 2  {
 3      fout.print(temp_n + " =");
 4      boolean was = false;
 5      for    (Map.Entry<BigInteger,    Integer>
        entry : map.entrySet())
 6      {
 7           if (was)
 8                fout.print(" *");
 9           else was = true;
10           fout.print(" " + entry.getKey());
11      }
12      fout.println();
13  }
```

Kode Sumber 4.7 Implementasi Fungsi *outputDisplay*

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan tentang uji coba dan evaluasi dari implementasi sistem yang telah dilakukan pada Tugas Akhir. Uji coba dilakukan dengan skenario tertentu dan dilanjutkan dengan tahap analisis dari struktur data yang digunakan untuk menyelesaikan permasalahan.

5.1 Lingkungan Uji Coba

Uji coba kebenaran dan uji coba kinerja akan dilakukan pada situs penilaian SPOJ dan evaluasi kebenaran dari struktur data yang diimplementasikan akan dilakukan pada komputer pribadi penulis. Lingkungan uji coba yang dilakukan pada situs penilaian SPOJ dengan *cluster Cube* yang memiliki spesifikasi sebagai berikut.

1. Perangkat Keras
    a. Processor Intel Xeon E3-1220 v5 (5CPUs).
    b. *Random Access Memory* 1536 MB.
2. Perangkat Lunak
    a. *Compiler* Java (HotSpot 12).

Sedangkan lingkungan evaluasi kebenaran dari implementasi dilakukan pada komputer pribadi penulis dengan spesifikasi sebagai berikut.

1. Perangkat Keras
    a. Processor: Intel® Core™ i7-4720HQ CPU @ 2.60GHZ (8 CPUs), ~2.6GHz.
    b. Memory: 8192 MB
2. Perangkat Lunak
    a. Sistem Operasi: Windows 10 Pro 64-bit.
    b. *Text Editor:* Netbeans dan Visual Studio Code.
    c. Bahasa Pemrograman: Java.

5.2 Skenario Uji Coba

Pada bagian ini akan dijelaskan skenario yang akan digunakan untuk melakukan pengujian terhadap implementasi yang dibuat untuk menyelesaikan permasalahan SPOJ SQFFACT – *Square-free Integers Factorization*.

Uji coba kebenaran, akan dilakukan dengan melihat umpan balik yang diberikan oleh SPOJ setelah sumber kode dikirimkan. SPOJ akan mengecek kebenaran dari sumber kode yang dikirimkan dengan memasukan kasus uji dengan batasan yang sudah dijabarkan pada subbab 2.1.1 yaitu.

1. N sebagai nilai yang akan dicari faktor primanya. Berupa bilangan dengan tipe *Integer* dengan batasan $N < 10^{100}$.
2. M sebagai nilai *euler's totient* dari N. Berupa bilangan dengan tipe *Integer* dengan batasan $M < N$.
3. T sebagai banyaknya uji coba yang dilakukan oleh sistem, berupa bilangan dengan tipe *Integer* dengan batasan $T \leq 100$.

Uji coba kinerja akan dilakukan dengan mengirimkan kode hasil implementasi program ke situs penilaian SPOJ sebanyak 10 kali kemudian mengevaluasi kinerja dari umpan balik yang diberikan.

5.2.1 Evaluasi Kebenaran

Evaluasi dilakukan dengan mengecek masukan yang diberikan dengan keluaran yang dihasilkan dari implementasi program yang sudah dibuat apakah sama dengan contoh keluaran yang ada pada permasalahan SPOJ SQFFACT – *Square-free Integers Factorization.* Kasus uji yang akan dicoba dapat dilihat pada Tabel 5.1 yang diambil dari situs SPOJ [1].

Tabel 5.1 Tabel uji coba

Masukan	Keluaran
3	
210 48	210 = 2 *3 * 5 * 7
983 982	983 = 983
14351 14112	14351 = 113 * 127

Pada program implementasi yang dibuat, sistem akan melakukan pengecekan terhadap tiga buah nilai N dan M. Berdasarkan nilai N dan M pertama yaitu $N = 210$ dan $M = 48$ sistem akan memberikan nilai $temp\_n = N$ dan $temp\_phi = M$. Setelah itu sistem akan memanggil fungsi *phiMod* dimana dalam fungsi tersebut akan terus membagi 2 nilai M selama masih memenuhi kondisi yang ada pada fungsi tersebut. Lalu jika telah selesai maka selanjutnya akan memanggil fungsi *primeFactorization* dengan parameter N. Hal pertama yang akan dilakukan fungsi tersebut akan mengecek selisih antara nilai N dengan M dimana jika memenuhi kondisinya maka bisa langsung dipastikan bahwa N adalah bilangan prima. Jika tidak memenuhi kondisi yang ada maka akan dilanjutkan kedalam sebuah perulangan yang didalamnya menerapkan *randomized algorithm*. Ternyata untuk masukan yang pertama ini tidak memenuhi kondisi dalam pengecekan selisih antara nilai N dan M sehingga proses dilanjutkan kedalam perulangan *randomized algorithm* yang telah dibuat. Setelah memasuki perulangan tersebut akan menghasilkan berapa saja faktor prima dari N. Detail mengenai sistem dalam mengolah masukan dapat dilihat pada Lampiran A.

Setelah masukan pertama selesai diproses maka selanjutnya memproses masukan yang kedua yaitu dengan nilai $N = 983$ dan nilai $M = 982$. Untuk prosesnya sebenarnya sama jika dibandingkan dengan proses masukan yang pertama namun yang membedakan pada masukan kedua ini ternyata memenuhi kondisi dalam pengecekan selisih antara nilai N dan M sehingga tidak perlu masuk kedalam proses selanjutnya yaitu perulangan

dengan *randomized algorithm.* Berikut adalah proses sistem dalam mengolah N dan M pada Gambar 5.1.

```
Nilai N = 983 Nilai M = 982
=============PHI MOD 2==============
Hasil M = 491
===========Factorization============
=======Hasil Factorization==========
983 = 983
```

Gambar 5.1 Proses sistem dalam mengolah masukan

Sedangakan untuk masukan ketiga dengan nilai $N = 14351$ dan nilai $M = 14112$ memiliki proses yang sama dengan masukan yang pertama. Detail mengenai sistem dalam mengolah masukan dapat dilihat pada Lampiran A.

Setelah seluruh masukan selesai diproses, maka keluaran yang dihasilkan oleh sistem yang dibuat penulis sudah sesuai dengan keluaran uji kasus pada Tabel 5.1 yang mana dapat dilihat pada Tabel 5.2.

Tabel 5.2 Perbandingan keluaran pada sistem dan keluaran uji kasus

Keluaran SPOJ	Keluaran Sistem
210 = 2 * 3 * 5 * 7	210 = 2 * 3 * 5 * 7
983 = 983	983 = 983
14351 = 113 * 127	14351 = 113 * 127

5.2.2 Uji Coba Kebenaran

Uji coba kebenaran dilakukan dengan mengirimkan kode hasil implementasi program ke situs penilaian SPOJ SQFFACT-*Square-free Integers Factorization*. Setelah mengirimkan kode sumber, maka akan didapatkan umpan balik dari situs SPOJ seperti yang pada Gambar 5.2.

25955512	2020-05-11 14:56:30	rtejakusuma	**accepted**	0.53	56M	JAVA

Gambar 5.2 Hasil umpan balik dari uji kebenaran pada SPOJ

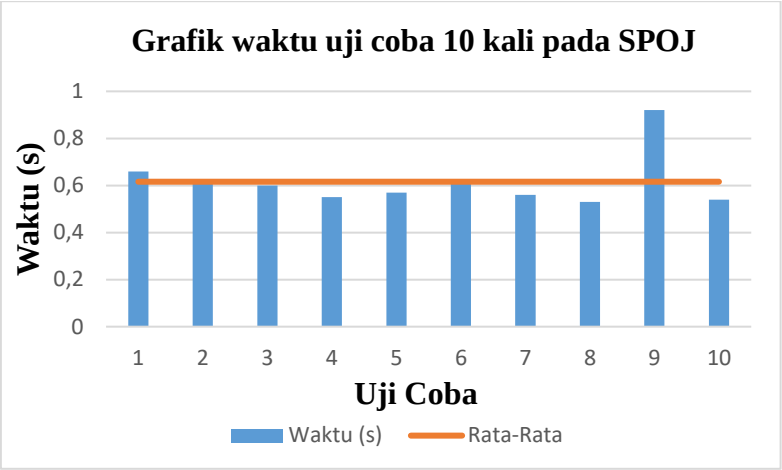
Dari hasil uji coba yang dilakukan kode sumber mendapatkan umpan balik *Accepted*. Waktu yang diperlukan program adalah 0,53 detik dan memori yang dibutuhkan program adalah 56 MB.

5.2.3 Uji Coba Kinerja

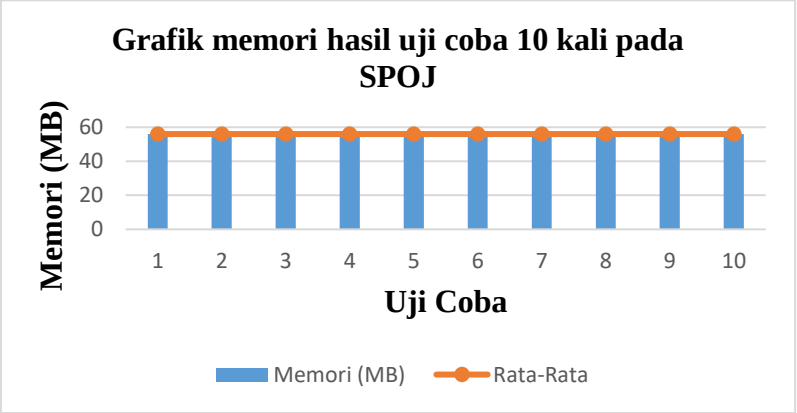
Kode sumber yang sama akan dikirimkan kembali sebanyak 10 kali untuk melihat variasi waktu dan memori yang dibutuhkan. Hasil uji coba dengan mengirimkan kode sumber sebanyak 10 kali, dapat dilihat pada Gambar 5.3 – Gambar 5.5.

25955565	2020-05-11 15:02:31	rtejakusuma	**accepted**	0.66	56M	JAVA
25955562	2020-05-11 15:01:47	rtejakusuma	**accepted**	0.61	56M	JAVA
25955558	2020-05-11 15:01:10	rtejakusuma	**accepted**	0.60	56M	JAVA
25955541	2020-05-11 14:59:33	rtejakusuma	**accepted**	0.55	56M	JAVA
25955534	2020-05-11 14:58:57	rtejakusuma	**accepted**	0.57	56M	JAVA
25955524	2020-05-11 14:57:33	rtejakusuma	**accepted**	0.62	56M	JAVA
25955518	2020-05-11 14:57:04	rtejakusuma	**accepted**	0.56	56M	JAVA
25955512	2020-05-11 14:56:30	rtejakusuma	**accepted**	0.53	56M	JAVA
25955511	2020-05-11 14:55:54	rtejakusuma	**accepted**	0.92	60M	JAVA
25955507	2020-05-11 14:55:07	rtejakusuma	**accepted**	0.54	56M	JAVA

Gambar 5.3 Hasil umpan balik dari uji kebenaran 10 kali pada SPOJ



Gambar 5.4 Grafik waktu uji coba kinerja 10 kali pada SPOJ



Gambar 5.5 Grafik memori uji coba kinerja 10 kali pada SPOJ

Dari hasil uji coba kinerja sebanyak 10 kali pada SPOJ, maka dapat dilihat bahwa rata-rata memori yang dibutuhkan oleh program adalah 56 MB, sedangkan waktu rata-rata yang dibutuhkan program adalah 0,616 detik. Detail mengenai peringkat sistem yang dibuat penulis bisa dilihat pada Lampiran

B. Berdasarkan Lampiran B diketahui jika peringkat dari implementasi yang dibuat penulis berada dalam peringkat 11 dari 16. Karena jika terdapat algoritma yang sama lalu diimplementasikan dengan menggunakan bahasa pemrograman Java dan C++ maka hasilnya tentu C++ akan lebih cepat dari Java. Tetapi jika kita melihat peringkat hanya berdasarkan bahasa pemrograman Java maka implementasi penulis berada dalam peringkat lima dari sembilan dan merupakan implementasi dengan penggunaan memori yang paling sedikit jika dibandingkan dengan yang lainnya. Dimana implementasi penulis hanya membutuhkan memori sebesar 56MB sementara untuk yang lainnya membutuhkan paling sedikit 61MB dan yang paling besar ialah 1340MB. Jika ditinjau dari waktunya yang paling tercepat 0,14 detik dan yang paling lama 0,71 detik sedangkan hasil penulis 0,53 detik. Jadi bisa dianggap jika hasil implementasi penulis dianggap cukup efisien dalam menyelesaikan permasalahan yang ada.

5.3 Analisis dan Kesimpulan Umum

Dari hasil pengujian kinerja yang telah dilakukan, dapat diambil poin-poin penting sebagai berikut.

- Permasalahan SPOJ SQFFACT dapat dimodelkan dengan menggunakan *Carmichael's function, Euler's theorem, Euler's totient function,* dan disusun kedalam *randomized algorithm.*
- Algoritma yang telah disusun pada *randomized algorithm* terbukti bisa menyelesaikan permasalahan SPOJ SQFFACT dan mampu melewati batas waktu yang diberikan.
- Rata-rata hasil *running time* dari program dalam menyelesaikan permasalahan SQFFACT adalah 0,616 detik.
- Rata-rata penggunaan memori dari sistem dalam menyelesaikan permasalahan SQFFACT adalah 56 MB.

[*Halaman ini sengaja dikosongkan*]

BAB VI
KESIMPULAN

Pada bab ini akan dijelaskan mengenai kesimpulan dari hasil uji coba serta saran-saran tentang pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan datang.

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan terhadap implementasi solusi untuk permasalahan SPOJ SQFFACT – *Square-free Integers Factorization* dapat diambil beberapa kesimpulan sebagai berikut.

1. Permasalahan SPOJ SQFFACT – *Square-free Integers Factorization* dapat dimodelkan dengan menggunakan *Carmichael's function, Euler's theorem,* dan *Euler's totient function* yang disusun kedalam *randomized algorithm*.
2. Implementasi dari algoritma yang telah dibuat terbukti benar dalam menyelesaikan permasalahan SPOJ SQFFACT – *Square-free Integers Factorization.*
3. Implementasi yang dibuat mampu menyelesaikan permasalahan dengan cukup efisien karena implementasi yang dibuat memerlukan waktu 0.53 detik dan penggunaan memori sebesar 56MB dimana berada dalam *ranking* 11 dari 16 secara umum atau berada dalam *ranking* lima dari sembilan jika dilihat dari hasil yang menggunakan bahasa pemrograman Java.

6.2 Saran

Saran yang diberikan dalam pengembangan algoritma pada permasalahan *prime factorization* adalah mencari algoritma baru ataupun pendekatan yang berbeda dari yang digunakan penulis yaitu menggunakan *Carmichael's function, Euler's theorem,* dan *Euler's totient* yang dikombinasikan dengan *randomized algorithm.*

[*Halaman ini sengaja dikosongkan*]

DAFTAR PUSTAKA

[1] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, 5th ed. CRC Press, 1996. [Online] Available: https://cacr.uwaterloo.ca/hac/, [Accessed 20 February 2020].

[2] P. R. Santos de Sousa, "Sphere Online Judge," 18 January 2010. [Online]. Available: https://www.spoj.com/problems/SQFFACT. [Accessed 11 February 2020].

[3] Keith Conrad, Euler's Theorem. [Online]. Available: https://kconrad.math.uconn.edu/blurbs/ugradnumthy/eulerthm.pdf. [Accessed 02 March 2020].

[4] P Erdos, C. Pomerance, E. Schmutz, Carmichael's Lambda Function, Acta Arithmetica LVIII.4, 1991. [Online] Available: https://math.dartmouth.edu/~carlp/PDF/lambda.pdf. [Accessed 10 March 2020].

[5] W. Stein, Elementary Number Theory: Primes, Congruences, and Secrets. Springer, 2017, [Online] Available: https://wstein.org/ent/ent.pdf. [Accessed 15 February 2020].

[6] Shashank Chorge and Juan Vargas, Proof of Euler's Function Formula, Volume 14, No. 2, Fall 2013. [Online] Available: https://scholar.rose-hulman.edu/cgi/viewcontent.cgi?article=1081&context=rhumj. [Accessed 25 March 2020]

[*Halaman ini sengaja dikosongkan*]

LAMPIRAN A: EVALUASI KEBENARAN

Berikut adalah gambar sistem dalam mengolah masukan dengan nilai $N = 210$ dan $M = 48$ dapat dilihat pada Gambar A.1 – Gambar A.9.

```
Nilai N = 210 Nilai M = 48
=============PHI MOD 2==============
Hasil PHI MOD 2 = 3
=============Randomized=============
Nilai x = 1017218
Nilai x mod prime = 188
Nilai g = 2
==========Factorization(g)==========
=============Randomized=============
Nilai x = 223573
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
=============Randomized=============
Nilai x = 932908
Nilai x mod prime = 0
=============Randomized=============
Nilai x = 851899
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
=============Randomized=============
Nilai x = 936523
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
=============Randomized=============
Nilai x = 839537
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
=============Randomized=============
Nilai x = 809213
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
```

Gambar A.1 Sistem dalam mengolah masukan dengan N = 210 dan M = 48 (bagian 1)

```
============Randomized============
Nilai x = 986582
Nilai x mod prime = 0
============Randomized============
Nilai x = 395998
Nilai x mod prime = 0
============Randomized============
Nilai x = 573844
Nilai x mod prime = 0
============Randomized============
Nilai x = 291312
Nilai x mod prime = 0
============Randomized============
Nilai x = 81789
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
============Randomized============
Nilai x = 555990
Nilai x mod prime = 0
============Randomized============
Nilai x = 13260
Nilai x mod prime = 0
============Randomized============
Nilai x = 576072
Nilai x mod prime = 0
============Randomized============
Nilai x = 595463
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
============Randomized============
Nilai x = 170938
Nilai x mod prime = 0
============Randomized============
Nilai x = 1029676
Nilai x mod prime = 0
```

Gambar A.2 Sistem dalam mengolah masukan dengan N = 210 dan M = 48 (bagian 2)

```
============Randomized============
Nilai x = 42671
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
============Randomized============
Nilai x = 663142
Nilai x mod prime = 0
============Randomized============
Nilai x = 987977
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
============Randomized============
Nilai x = 57136
Nilai x mod prime = 0
============Randomized============
Nilai x = 110280
Nilai x mod prime = 0
============Randomized============
Nilai x = 71092
Nilai x mod prime = 0
============Randomized============
Nilai x = 253610
Nilai x mod prime = 0
============Randomized============
Nilai x = 801326
Nilai x mod prime = 0
============Randomized============
Nilai x = 839079
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
```

Gambar A.3 Sistem dalam mengolah masukan dengan N = 210 dan M = 48 (bagian 3)

```
=======Factorization(prime/g)=======
=============Randomized=============
Nilai x = 935724
Nilai x mod prime = 69
Nilai g = 3
==========Factorization(g)==========
=============Randomized=============
Nilai x = 538896
Nilai x mod prime = 0
=============Randomized=============
Nilai x = 883809
Nilai x mod prime = 0
=============Randomized=============
Nilai x = 275738
Nilai x mod prime = 2
Nilai g = 1
Nilai X = 2
=============Randomized=============
Nilai x = 395729
Nilai x mod prime = 2
Nilai g = 1
Nilai X = 2
=============Randomized=============
Nilai x = 206141
Nilai x mod prime = 2
Nilai g = 1
Nilai X = 2
=============Randomized=============
Nilai x = 522746
Nilai x mod prime = 2
Nilai g = 1
Nilai X = 2
=============Randomized=============
Nilai x = 222378
Nilai x mod prime = 0
```

Gambar A.4 Sistem dalam mengolah masukan dengan N = 210 dan M = 48 (bagian 4)

```
=============Randomized=============
Nilai x = 482640
Nilai x mod prime = 0
=============Randomized=============
Nilai x = 782089
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
=============Randomized=============
Nilai x = 619951
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
=============Randomized=============
Nilai x = 876385
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
=============Randomized=============
Nilai x = 723305
Nilai x mod prime = 2
Nilai g = 1
Nilai X = 2
=============Randomized=============
Nilai x = 825376
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
=============Randomized=============
Nilai x = 1041120
Nilai x mod prime = 0
=============Randomized=============
Nilai x = 355269
Nilai x mod prime = 0
```

Gambar A.5 Sistem dalam mengolah masukan dengan N = 210 dan M = 48 (bagian 5)

```
============Randomized============
Nilai x = 53074
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
=======Factorization(prime/g)=======
============Randomized============
Nilai x = 460680
Nilai x mod prime = 10
Nilai g = 5
==========Factorization(g)==========
============Randomized============
Nilai x = 633525
Nilai x mod prime = 0
============Randomized============
Nilai x = 1977
Nilai x mod prime = 2
Nilai g = 1
Nilai X = 3
Nilai nw = 4
============Randomized============
Nilai x = 555969
Nilai x mod prime = 4
Nilai g = 1
Nilai X = 4
============Randomized============
Nilai x = 878362
Nilai x mod prime = 2
Nilai g = 1
Nilai X = 3
Nilai nw = 4
============Randomized============
Nilai x = 164012
Nilai x mod prime = 2
Nilai g = 1
Nilai X = 3
Nilai nw = 4
```

Gambar A.6 Sistem dalam mengolah masukan dengan N = 210 dan M = 48 (bagian 6)

```
=============Randomized=============
Nilai x = 520162
Nilai x mod prime = 2
Nilai g = 1
Nilai X = 3
Nilai nw = 4
=============Randomized=============
Nilai x = 980291
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
=============Randomized=============
Nilai x = 792596
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
=============Randomized=============
Nilai x = 717477
Nilai x mod prime = 2
Nilai g = 1
Nilai X = 3
Nilai nw = 4
=============Randomized=============
Nilai x = 92786
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
=============Randomized=============
Nilai x = 629309
Nilai x mod prime = 4
Nilai g = 1
Nilai X = 4
```

Gambar A.7 Sistem dalam mengolah masukan dengan N = 210 dan M = 48 (bagian 7)

```
=======Factorization(prime/g)=======
============Randomized============
Nilai x = 593597
Nilai x mod prime = 4
Nilai g = 1
Nilai X = 1
============Randomized============
Nilai x = 337212
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
============Randomized============
Nilai x = 74401
Nilai x mod prime = 5
Nilai g = 1
Nilai X = 6
============Randomized============
Nilai x = 293833
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
============Randomized============
Nilai x = 1019910
Nilai x mod prime = 3
Nilai g = 1
Nilai X = 6
============Randomized============
Nilai x = 619220
Nilai x mod prime = 0
============Randomized============
Nilai x = 806534
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
============Randomized============
Nilai x = 871612
Nilai x mod prime = 0
```

Gambar A.8 Sistem dalam mengolah masukan dengan N = 210 dan M = 48 (bagian 8)

```
=======Factorization(prime/g)=======
============Randomized============
Nilai x = 593597
Nilai x mod prime = 4
Nilai g = 1
Nilai X = 1
============Randomized============
Nilai x = 337212
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
============Randomized============
Nilai x = 74401
Nilai x mod prime = 5
Nilai g = 1
Nilai X = 6
============Randomized============
Nilai x = 293833
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
============Randomized============
Nilai x = 1019910
Nilai x mod prime = 3
Nilai g = 1
Nilai X = 6
============Randomized============
Nilai x = 619220
Nilai x mod prime = 0
============Randomized============
Nilai x = 806534
Nilai x mod prime = 1
Nilai g = 1
Nilai X = 1
============Randomized============
Nilai x = 871612
Nilai x mod prime = 0
```

Gambar A.9 Sistem dalam mengolah masukan dengan N = 210 dan M = 48 (bagian 9)

Berikut adalah gambar sistem dalam mengolah masukan dengan nilai $N = 14351$ dan $M = 14112$ dapat dilihat pada Gambar A.10 – Gambar A.12.

```
Nilai N = 14351 Nilai M = 14112
=============PHI MOD 2==============
Hasil PHI MOD 2 = 441
=============Randomized=============
Nilai x = 708418
Nilai x mod prime = 5219
Nilai g = 1
Nilai X = 7619
Nilai nw = 13717
Nilai fac = 127
=========Factorization(fac)=========
=============Randomized=============
Nilai x = 637546
Nilai x mod prime = 6
Nilai g = 1
Nilai X = 126
=============Randomized=============
Nilai x = 962777
Nilai x mod prime = 117
Nilai g = 1
Nilai X = 1
=============Randomized=============
Nilai x = 543850
Nilai x mod prime = 36
Nilai g = 1
Nilai X = 1
=============Randomized=============
Nilai x = 221875
Nilai x mod prime = 6
Nilai g = 1
Nilai X = 126
=============Randomized=============
Nilai x = 353154
Nilai x mod prime = 94
Nilai g = 1
Nilai X = 1
```

Gambar A.10 Sistem dalam mengolah masukan *N* = 14351 dan *M* = 14112 (bagian 1)

```
============Randomized============
Nilai x = 1019369
Nilai x mod prime = 67
Nilai g = 1
Nilai X = 126
============Randomized============
Nilai x = 447608
Nilai x mod prime = 60
Nilai g = 1
Nilai X = 1
============Randomized============
Nilai x = 610730
Nilai x mod prime = 114
Nilai g = 1
Nilai X = 126
============Randomized============
Nilai x = 113516
Nilai x mod prime = 105
Nilai g = 1
Nilai X = 126
============Randomized============
Nilai x = 590904
Nilai x mod prime = 100
Nilai g = 1
Nilai X = 1
======Factorization(prime/fac)======
============Randomized============
Nilai x = 105026
Nilai x mod prime = 49
Nilai g = 1
Nilai X = 1
============Randomized============
Nilai x = 840164
Nilai x mod prime = 9
Nilai g = 1
Nilai X = 44
Nilai nw = 15
```

Gambar A.11 Sistem dalam mengolah masukan N = 14351 dan M = 14112 (bagian 2)

```
============Randomized=============
Nilai x = 1008910
Nilai x mod prime = 46
Nilai g = 1
Nilai X = 71
Nilai nw = 69
============Randomized=============
Nilai x = 629453
Nilai x mod prime = 43
Nilai g = 1
Nilai X = 40
Nilai nw = 18
============Randomized=============
Nilai x = 62779
Nilai x mod prime = 64
Nilai g = 1
Nilai X = 112
============Randomized=============
Nilai x = 978456
Nilai x mod prime = 102
Nilai g = 1
Nilai X = 44
Nilai nw = 15
============Randomized=============
Nilai x = 820601
Nilai x mod prime = 108
Nilai g = 1
Nilai X = 78
Nilai nw = 95
============Randomized=============
Nilai x = 263973
Nilai x mod prime = 5
Nilai g = 1
Nilai X = 35
Nilai nw = 95
============Randomized=============
Nilai x = 346807
Nilai x mod prime = 10
Nilai g = 1
Nilai X = 40
Nilai nw = 18
============Randomized=============
Nilai x = 300247
Nilai x mod prime = 6
Nilai g = 1
Nilai X = 42
Nilai nw = 69
========Hasil Factorization=========
14351 = 113 * 127
```

Gambar A.12 Sistem dalam mengolah masukan N = 14351 dan M = 14112 (bagian 3)

LAMPIRAN B: Peringkat Sistem dalam Menyelesaikan Permasalahan

Berikut adalah gambar mengenai peringkat sistem yang telah dibuat penulis yang dapat dilihat pada Gambar B.1 dan Gambar B.2. Gambar B.1 menunjukkan peringkat secara umum yaitu peringkat 11 dari 16. Sedangkan Gambar B.2 menunjukkan peringkat jika dilihat berdasarkan bahasa pemrograman Java.

RANK	DATE	USER	RESULT	TIME	MEM	LANG
1	2019-03-21 08:22:58	liouzhou_101	accepted	0.04	17M	CPP14
2	2015-05-16 23:44:35	Min_25	accepted	0.12	11M	PYTHON3
3	2017-01-28 04:15:57	Michael Kharitonov	accepted	0.14	694M	JAVA
4	2015-12-23 16:35:54	Min_25	accepted	0.19	47M	PYPY
5	2012-04-02 21:20:34	edel	accepted	0.21	1340M	JAVA
6	2019-03-21 07:24:43	liouzhou_101	accepted	0.21	120M	PYPY
7	2014-06-05 23:44:34	Mostafa mahmoud	accepted	0.23	178M	JAVA
8	2014-06-03 04:56:55	[SPbSU ITMO] WiNGeR	accepted	0.26	1340M	JAVA
9	2011-06-28 01:14:01	Esteban Crespí de Valldaura	accepted	0.41	7.1M	C++ 4.3.2
10	2020-04-18 23:57:51	Viplov Jain	accepted	0.41	9.6M	PY_NBC
11	2020-05-11 14:56:30	rtejakusuma	accepted	0.53	56M	JAVA
12	2020-04-24 22:36:29	Rully Soelaiman	accepted	0.62	63M	JAVA
13	2019-11-24 19:46:55	Karolis Kusas	accepted	0.65	61M	JAVA
14	2010-02-05 16:49:40	[Rampage] Blue.Mary	accepted	0.69	1340M	JAVA

Gambar B.1 Peringkat pada Permasalahan SPOJ SQFFACT Secara Umum

RANK	DATE	USER	RESULT	TIME	MEM	LANG
1	2017-01-28 04:15:57	Michael Kharitonov	accepted	0.14	694M	JAVA
2	2012-04-02 21:20:34	edel	accepted	0.21	1340M	JAVA
3	2014-06-05 23:44:34	Mostafa mahmoud	accepted	0.23	178M	JAVA
4	2014-06-03 04:56:55	[SPbSU ITMO] WiNGeR	accepted	0.26	1340M	JAVA
5	2020-05-11 14:56:30	rtejakusuma	accepted	0.53	56M	JAVA
6	2020-06-16 00:00:36	Rully Soelaiman	accepted	0.58	61M	JAVA
7	2019-11-24 19:46:55	Karolis Kusas	accepted	0.65	61M	JAVA
8	2010-02-05 16:49:40	[Rampage] Blue.Mary	accepted	0.69	1340M	JAVA
9	2013-09-08 14:42:58	Georgy	accepted	0.71	1340M	JAVA

Gambar B.2 Peringkat pada Permasalahan SPOJ SQFFACT dengan Bahasa Pemrograman Java

BIODATA PENULIS

Penulis bernama Raden Teja Kusuma, putra ketiga dari tiga bersaudara yang lahir pada tanggal 19 Agustus 1997 di Madiun. Penulis telah mengenyam pendidikan di Sekolah Dasar Negeri Kawedanan 2 pada tahun 2004 hingga 2010, Sekolah Menengah Pertama Negeri 1 Kawedanan pada tahun 2010 hingga 2011, Sekolah Menengah Pertama Negeri 12 Madiun pada tahun 2011 hingga 2013, dan Sekolah Menengah Atas Negeri 3 Madiun pada tahun 2013 hingga 2016. Pada masa penulisan Tugas Akhir ini, penulis sedang menempuh studi S1 di Institut Teknologi Sepuluh Nopember, Surabaya di Departemen Teknik Informatika.

Selama masa studi, penulis memiliki ketertarikan dalam bidang matematika, rancang bangun aplikasi web, rancang bangun aplikasi sistem informasi, jaringan, UI/UX, dan manajemen informasi. Keinginan penulis dalam mengajar juga mendorong penulis menjadi asdos pada mata kuliah Sistem Basis Data dan Perancangan Perangkat Lunak.

Selama menempuh perkuliahan penulis juga aktif mengikuti kompetisi turnamen gim DOTA 2 tingkat nasional dan menjadi juara 2 pada MAGE 2018.

Selain kesibukan akademik, penulis juga berperan aktif dalam berbagai kegiatan dan kepanitiaan. Beberapa diantaranya adalah menjadi Badan Pengurus Harian 3 mengenai Perlengkapan dan Transportasi dalam kegiatan Schematics 2017, Badan pengurus Harian 1 mengenai Perlengkapan dan Transportasi dalam kegiatan Schematics 2018, staff Perlengkapan dan Transportasi dalam kegiatan FTIF Festival 2017, dan Himpunan Mahasiswa Teknik Computer (HMTC) ITS. Penulis dapat dihubungi memalui surel di tejakusuma60@gmail.com.