TUGAS AKHIR – IF184802

PENERAPAN *EULER TOTIENT* PADA PENYELESAIAN PERMASALAHAN SPOJ-ADAHW: ADA AND HOMEWORK

GILBERT LIJAYA THERRY
05111640000051

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

Dosen Pembimbing II:
Abdul Munif, S.Kom., M.Sc.

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

[*Halaman ini sengaja dikosongkan*]

TUGAS AKHIR – IF184802

PENERAPAN *EULER TOTIENT* PADA PENYELESAIAN PERMASALAHAN SPOJ-ADAHW: ADA AND HOMEWORK

GILBERT LIJAYA THERRY
05111640000051

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

Dosen Pembimbing II:
Abdul Munif, S.Kom., M.Sc.

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

[*Halaman ini sengaja dikosongkan*]

FINAL PROJECT – IF184802

IMPLEMENTATION OF EULER TOTIENT IN SOLVING SPOJ PROBLEM ADAHW: ADA AND HOMEWORK

GILBERT LIJAYA THERRY
05111640000051

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

Supervisor II
Abdul Munif, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS ENGINEERING
Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya
2020

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

PENERAPAN *EULER TOTIENT* PADA PENYELESAIAN PERMASALAHAN SPOJ-ADAHW: ADA AND HOMEWORK

TUGAS AKHIR

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

Oleh:
GILBERT LIJAYA THERRY
NRP: 051116 40000 051

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Rully Soelaiman, S.Kom., M.Kom. .....
   (NIP. 19700213 199402 1 001) (Pembimbing 1)

2. Abdul Munif, S.Kom., M.Sc. .....
   (NIP. 19860823 201504 1 004) (Pembimbing 2)

SURABAYA
Januari, 2020

[*Halaman ini sengaja dikosongkan*]

PENERAPAN *EULER TOTIENT* PADA PENYELESAIAN PERMASALAHAN SPOJ-ADAHW: ADA AND HOMEWORK

Nama Mahasiswa : Gilbert Lijaya Therry
NRP : 051116 40000 051
Departemen : Teknik Informatika - ITS
DosenPembimbing 1 : Rully Soelaiman, S.Kom., M.Kom.
DosenPembimbing 2 : Abdul Munif, S.Kom., M.Sc.

ABSTRAK

Permasalahan Tugas Akhir ini bermula dari adanya permasalahan Ada and Homework (ADAHW) pada SPOJ. Permasalahan tersebut menggambarkan sebuah bilangan N dan yang akan dicari adalah jumlah dari faktor persekutuan terbesar antara K-1 dan N, dimana K adalah semua bilangan bulat positif dimulai dari 2 sampai N.

Syarat yang harus dikerjakan dalam permasalahan di atas menimbulkan hal-hal baru yang membutuhkan metode penyelesaian yang tepat untuk menyelesaikan permasalahan dari ADAHW. Hal pertama adalah batasan dari nilai N yang cukup besar yaitu 2 hingga 10^{18}. Sehingga tentunya tidak dapat dilakukan iterasi satu persatu angka karena akan memakan waktu yang sangat lama dengan bilangan sebesar 10^{18}. Karena itu dibutuhkan sebuah solusi yang cukup optimal untuk mengatasi rentang N yang cukup besar ini.

Pada Tugas Akhir ini, awalnya permasalahan ADAHW ini akan dimodelkan dulu untuk memunculkan Euler Totient pada permasalahannya dengan menggunakan Menon Identity. Setelah itu akan diimplementasikan algoritma Pollard Rho Brent dan Miller Rabin yang dapat menangani faktorisasi prima dari angka-angka besar dengan cepat. Solusi yang dibuat cukup efisien dengan rata-rata waktu penyelesaian 0,53 detik dengan penggunaan memori 4,7 MB.

Kata kunci: Euler Totient, Pollard Rho Brent, Miller Rabin.

[*Halaman ini sengaja dikosongkan*]

IMPLEMENTATION OF EULER TOTIENT IN SOLVING SPOJ PROBLEMS ADAHW: ADA AND HOMEWORK

Name	: Gilbert Lijaya Therry
NRP	: 051116 40000 051
Department	: Informatics - ITS
Supervisor 1	: Rully Soelaiman, S.Kom., M.Kom.
Supervisor 2	: Abdul Munif, S.Kom., M.Sc.

ABSTRACT

This Thesis starts with Ada and Homework problems on SPOJ. This problem describes a number N and we will try to find the sum of the greatest common divisor between K-1 and N, where K are all positive integers starting from 2 to N.

This conditions creates things that need a correct method to solve problems from ADAHW. The first problem is the range of N is from 2 to 10^{18}. So of course iterations cannot be done one by one number because it will take a very long time with 10^{18}. So we need a better solution to overcome a fairly large range of N.

In this Thesis, initially this problem will be modelled first to bring Euler Totient to the problem using Menon Identity. After that, Pollard Rho Brent and Miller Rabin algorithms will be implemented which can handle prime factorization of large numbers quickly. The solution made is quite efficient with an average completion time of 0.53 seconds with 4.7 MB of memory usage.

Keywords: Euler Totient, Pollard Rho Brent, Miller Rabin.

[*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:

PENERAPAN *EULER TOTIENT* PADA PENYELESAIAN PERMASALAHAN SPOJ-ADAHW: ADA AND HOMEWORK

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 memeberikan 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. 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.
2. Bapak Abdul Munif, S.Kom., M.Sc. selaku dosen pembimbing yang telah memberikan ilmu dan masukan kepada penulis.
3. Bapak, Ibu, dan keluarga penulis yang selalu memberikan dukungan, perhatian, dan kasih sayang bagi penulis yang menjadi semangat selama perkuliahan maupun pengerjaan Tugas Akhir.
4. Teman-teman sekontrakan saya yang sudah membantu penulis selama kuliah di Teknik Informatika.

5. Ferdinand Jason yang sudah membantu penulis menyelesaikan
6. Teman-teman angkatan 2016 Departemen Teknik Informatika ITS yang telah menemani perjuangan penulis selama 4 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, Januari 2020

Gilbert Lijaya Therry

DAFTAR ISI

[*Halaman ini sengaja dikosongkan*]

DAFTAR PSEUDOCODE

[*Halaman ini sengaja dikosongkan*]

DAFTAR GAMBAR

[*Halaman ini sengaja dikosongkan*]

DAFTAR TABEL

DAFTAR KODE SUMBER

[*Halaman ini sengaja dikosongkan*]

DAFTAR NOTASI

$\sum_{i=1}^{N} i$	Jumlah dari *i* dimana *i* adalah bilangan bulat dari 1 sampai *N*
$\varphi(N)$	*Euler Totient Function*, menotasikan banyaknya nilai yang koprima dengan *N*
$d(N)$	Banyaknya faktor positif bilangan bulat dari *N*
$\gcd(a, b)$	Faktor Persekutuan Terbesar dari *a* dan *b*

[*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

Latar belakang dari Tugas ini bermula dari permasalahan *Ada and Homework* pada SPOJ [1]. Permasalahan tersebut adalah bagaimana cara mengecek apakah bilangan *K* saling prima dengan bilangan *N* dan jika saling prima maka akan dimasukkan ke dalam Persamaan 1.1.

$$\sum_{K=2,gcd(K,N)=1}^{N} gcd(K-1,N) \tag{1.1}$$

Problem di SPOJ-ADAHW adalah permasalahan dengan *T* adalah banyaknya kasus uji coba, *N* adalah angka yang menjadi masukan dari jawaban yang diinginkan dan *K* adalah semua bilangan bulat positif 2 sampai *N*. Hal pertama yang menjadi masalah adalah karena rentang dari nilai *N* yang cukup besar yaitu dari 2 sampai 10^{18}. Untuk mengecek apakah *K* saling prima dengan *N* atau tidak dan untuk menyelesaikan permasalahan pada Persamaan 1.1 diperlukan teknik yang efisien.

1.2 Rumusan Permasalahan

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

1. Bagaimana desain algoritma yang sesuai dan efisien untuk permasalahan mengecek *K* (bilangan bulat positif dari 2 sampai *N*) saling prima dengan *N* ?
2. Bagaimana penerapan *Euler Totient* pada Studi Kasus SPOJ-ADAHW?

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

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Pengaplikasian *Euler Totient* pada problem ADAHW.
2. Pembuktian kebenaran didasarkan pada hasil *submission* di sistem penilaian daring SPOJ.
3. Implementasi menggunakan bahasa pemrograman C++.

Batasan pada SPOJ:

1. *N* sebagai angka yang akan dimasukkan berupa bilangan bulat dengan rentang antara 2 sampai 10^{18}.
2. *T* sebagai banyaknya kasus uji berupa bilangan bulat dengan rentang antara 1 sampai 1.000.
3. *K* sebagai himpunan bilangan bulat dari 2 sampai *N*.
4. Batasan waktu adalah 10 detik.
5. Batasan memori adalah 1.536 MB.
6. Batasan ukuran program adalah 50 KB.

1.4 Tujuan Pembuatan Tugas Akhir

Tujuan dari Tugas Akhir ini antara lain:

1. Melakukan analisis untuk mencari desain algoritma yang sesuai dan efisien sehingga dapat menyelesaikan permasalahan batasan dari *N* yang cukup besar dalam waktu 10 detik.
2. Melakukan analisis dan penerapan *Euler Totient* untuk menyelesaikan permasalahan SPOJ-ADAHW.
3. Melakukan uji coba dan implementasi *Euler Totient* untuk menyelesaikan permasalahan SPOJ-ADAHW.

1.5 Manfaat Tugas Akhir

Manfaat dari pembuatan Tugas Akhir ini adalah sebagai berikut:

1. Mampu memberikan pemahaman dan penjelasan mencari desain algoritma yang efisien untuk penyelesaian kasus SPOJ-ADAHW.
2. Mampu memberikan pemahaman dan penjelasan analisis *Euler Totient* pada permasalahan pada kasus SPOJ-ADAHW.
3. Memberikan hasil uji coba dari implementasi *Euler Totient* yang akan dilakukan.

1.6 Metodologi

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

1.6.1 Penyusunan Proposal Tugas Akhir

Pada tahap ini dilakukan penyusunan proposal Tugas Akhir yang berisi permasalahan pada permasalahan klasik SPOJ-ADAHW serta gagasan solusi yang akan dibahas pada Tugas Akhir ini.

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 teori bilangan. Informasi tersebut didapatkan dari buku, internet, dan materi kuliah yang berhubungan dengan metode yang akan digunakan. Studi literatur yang digunakan adalah studi *Menon Identity*, algoritma *Pollard Rho Brent*, algoritma *Miller Rabin*, dan *Euler Totient*.

1.6.3 Desain

Pada tahap ini dilakukan desain rancangan algoritma yang digunakan dalam solusi untuk pemecahan masalah klasik SPOJ-ADAHW.

1.6.4 Implementasi Perangkat Lunak

Pada tahap ini dilakukan implementasi dari rancangan desain ke dalam bentuk program dalam bahasa pemrograman C++.

1.6.5 Uji Coba Kebenaran

Pada tahap ini dilakukan uji coba kebenaran implementasi yang dilakukan pada sistem penilaian daring SPOJ.

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 aplikasi perangkat lunak 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 dapat dilihat seperti dibawah ini.

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 dijelaskan di dalamnya.

Bab II Dasar Teori

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

Bab III Desain

Bab ini berisi desain algoritma dan struktur data yang digunakan dalam penyelesaian permasalahan.

Bab IV Implementasi

Bab ini berisi penjelasan implementasi dalam bahasa pemrograman C++ berdasarkan desain algoritma yang telah dilakukan pada tahap desain.

Bab V Uji Coba dan Evaluasi

Bab ini berisi uji coba kebenaran implementasi yang dilakukan pada sistem penilaian daring SPOJ.

Bab VI Kesimpulan dan Saran

Bab ini merupakan bab terakhir yang menjelaskan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

[*Halaman ini sengaja dikosongkan*]

BAB II
DASAR TEORI

Pada bab ini akan dibahas mengenai dasar teori yang menjadi dasar pengerjaan buku Tugas Akhir ini.

2.1 Deskripsi Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini adalah perhitungan nilai *N* dan *K* dari Persamaan 2.1[1].

$$\sum_{K=2,gcd(K,N)=1}^{N} gcd(K-1,N) \tag{2.1}$$

Selain *N* dan *K*, ada juga variabel *T* (banyak kasus uji coba) yang akan menjadi masukan bersama variabel *N*. Berdasarkan Persamaan 2.1 dan *Pseudocode* 2.1, pendekatan ini kurang efisien untuk permasalahan *Ada and Homework* (ADAHW) dikarenakan masih memiliki kompleksitas *O(N)*.

Pseudocode 2.1: Penyelesaian dengan Algoritma Naif

Input : T, N
Output : $\sum_{2}^{N} gcd(K-1,N)$
1: $f \leftarrow 1$
2: **for** $\boldsymbol{i} \leftarrow 1$ ***to*** T **do**
3: **for** $\boldsymbol{k} \leftarrow 2$ ***to*** N **do**
4: **if** $(gcd(N,K))$ **then**
5: $f \leftarrow f + gcd(N,K-1)$
6: **end for**
7: **end for**
8: **return** f

Dikarenakan nilai batasan *N* pada deskripsi soal cukup besar yaitu 10^{18}, sehingga algoritma naif akan memberikan *verdict Time Limit Excedeed* karena melampaui batas waktu yang telah ditentukan.

2.2 Deskripsi Umum

Pada subbab ini akan dibahas beberapa teori yang akan digunakan pada penyelesaian permasalahan SPOJ-ADAHW.

2.2.1 Faktorisasi Prima

Faktorisasi prima adalah pecahan bilangan komposit yang terdiri dari bilangan-bilangan pembagi yang lebih kecil dan hasil perkalian dari bilangan-bilangan tersebut sama dengan bilangan komposit yang disebutkan [2]. Dalam permasalahan ini konsep dari faktorisasi prima nantinya akan digunakan untuk memfaktorkan bilangan masukan (*N*).

2.2.2 C++ Boost Library

Ada banyak jenis dari *Boost library* pada C++, salah satunya adalah *Multiprecision library*. *Multiprecision library* mempunyai berbagai kegunaan, salah satu diantaranya adalah dapat mengatasi angka yang sangat besar yang melebihi tipe data *long long* (2^{64}) [3]. Dalam permasalahan ini C++ *boost library* akan digunakan untuk mengatasi angka-angka yang melebihi tipe data *long long* (2^{64}) seperti masukan dari *N* yang batasannya sampai 10^{18}.

2.2.3 Bilangan Saling Prima

Dalam teori bilangan dua bilangan A dan B dikatakan saling prima jika $gcd(a, b) = 1$, dimana tidak terdapat sebuah bilangan asli lebih dari satu yang dapat membagi habis A dan B [4]. Dalam permasalahan ini konsep bilangan saling prima akan digunakan sebagai pemahaman awal dalam penyelesaian menggunakan penerapan *Euler Totient.*

2.2.4 Pemangkatan Modular

Pemangkatan modular merupakan teknik untuk memangkatkan suatu bilangan *x* pangkat *y* dalam modulus *p* atau dinyatakan dalam $x^y mod\, p$ [4]. Dalam permasalahan ini konsep

pemangkatan modular akan diaplikasikan pada algoritma *Miller Rabin*.

2.2.5 Perkalian Modular

Perkalian modular merupakan perkalian antara suatu bilangan *a* dikalikan dengan bilangan *b* dalam modulus *p* atau dinyatakan dalam $X = (a)(b) mod\ p$, dimana *X* adalah hasil dari perkalian modular [2]. Dalam permasalahan ini konsep perkalian modular akan diaplikasikan pada algoritma *Pollard Rho Brent* dan *Miller Rabin.*

2.2.6 Faktor Persekutuan Terbesar

Faktor persekutuan terbesar (GCD) dari dua bilangan (*a* dan *b*) bulat positif adalah bilangan bulat positif terbesar yang dapat membagi habis kedua bilangan tersebut (*a* dan *b*) [4]. Dalam permasalahan ini konsep GCD akan diaplikasikan pada algoritma *Pollard Rho Brent*.

2.2.7 Menon Identity

Menon Identity adalah sebuah identitas matematika yang menyatakan bahwa untuk setiap bilangan bulat *n*, berlaku Persamaan 2.2.

$$\sum_{i=1,gcd(i,n)=1}^{n} gcd(i-1,n) = d(n)\varphi(n) \qquad (2.2)$$

Dimana $\varphi(n)$ menyatakan *Euler Totient* dari *n* dan $d(n)$ menyatakan banyak faktor positif dari *n* [5]. Dalam permasalahan ini *Menon Identity* akan digunakan untuk memodelkan permasalahan awal sehingga dapat dipecah menjadi masalah-masalah yang lebih kecil dan lebih rinci. Penjabarannya akan dijelaskan pada subbab 2.3.

2.2.8 Algoritma Pollard Rho Brent

Pollard Rho Brent adalah optimasi dari algoritma *Pollard Rho* yang digunakan untuk mencari faktor *non-trivial* dari sebuah bilangan komposit. Didefinisikan bilangan bulat komposit *x*, bilangan bulat positif *c*, dan bilangan bulat positif $x_{i+1} = g(x_i) = x_i^2 + c$ (dimana *c*

adalah bilangan bulat yang dapat ditentukan secara acak) [6]. Faktor *non-trivial* bisa didapatkan dengan melakukan perhitungan pada Persamaan 2.3.

$$d = gcd(x_i - x_j, x) \tag{2.3}$$

Dimana Persamaan 2.3 ekivalen dengan Persamaan 2.4.

$$d \mid x \tag{2.4}$$

Nilai awal untuk $i = 0$ untuk x_i dan x_j dapat ditentukan dengan nilai acak. Persamaan 2.4 menunjukkan bahwa *d* adalah faktor dari *x*. Jika nilai *d* adalah faktor *trivial* ($d = 1$ atau $d = x$), lakukan pengecekan apakah ada untuk bilangan bulat positif *y* dimana memenuhi $2^y = i$. Jika ditemukan maka tetapkan nilai baru untuk x_j yaitu $x_j = x_i$ dan jika tidak ditemukan maka ulangi lagi dari Persamaan 2.3 dengan nilai *c* yang berbeda. Dengan nilai *c* yang berbeda ini diharapkan bisa mendapatkan faktor baru dari bilangan *d*. Dalam permasalahan ini konsep dari algoritma *Pollard Rho Brent* ini akan digunakan untuk mendapatkan faktor-faktor dari angka masukan *N*. Pada algoritma ini juga akan diaplikasikan GCD dan perkalian modular yang akan dimodelkan pada *Pseudocode* 2.2.

Pseudocode 2.2: Algoritma Pollard Rho Brent

```
1: int brent(N)
2: if (N %2= 0) then
3: return 2
4: y,c,m ← rand (1 , N-1)
5: g,r,q ← 1
6: do
7:     x ← y
8:     for i ← 0 to r do
9:         y ← (MULMOD(y, y, N) + c)
10:        y ≥ N ? y −= N : y
```

```
11:       k ← 0
12:    end for
13:    do
14:       ys = y
15:       k ← k + m
16:          for i ← 0 to min(m ,r − k) do
17.             y ← (MULMOD(y, y, N) + c)
18:          y ≥ N ? y −= N : y
19:          q ← (MULMOD(q,abs(x − y),N))
20:          end for
21:       g ← gcd (q,N)
22:       k ← k + m
23:    while (k < r & g ← 1)
24:       r <<= 1
25: while (g ← 1)
26:if (g ← n) then
27:do
28:    cs ← (MULMOD(ys, ys, N))
29:    ys ← cs +c
30:    ys ≥ N ? ys −= N : y
31:    g ← gcd(abs(x−ys ),N)
32:    if (g > 1) then break
33:while (true)
34:return g
```

2.2.9 Algoritma Miller Rabin

Algoritma *Miller Rabin* didasari pada teori Fermat yang menyatakan bahwa jika *n* adalah bilangan prima dan $gcd(a, n)$ adalah 1, maka $a^{n-1} \equiv 1 \, mod \, n$ [7]. Algoritma ini digunakan untuk mengetes sebuah bilangan apakah bilangan tersebut prima atau tidak. Menggunakan bilangan bulat ganjil positif *N*, ditentukan bilangan bulat positif *d* dan *r* sedemikian sehingga $N - 1 = d \cdot 2^r$. Metode ini menyatakan bahwa untuk bilangan prima *N* dan suatu bilangan bulat

positif a dengan $gcd(a, N) = 1$ setidaknya salah satu Persamaan 2.5 dan 2.6 bernilai benar.

$$a^d \equiv 1(mod\ N) \tag{2.5}$$

$$a^{d \cdot 2^j} \equiv -1(mod\ N), 0 \leq j < r \tag{2.6}$$

Metode ini merupakan metode probabilistik yang memiliki persentase galat $(\frac{1}{4})^t$ dengan t adalah banyaknya nilai a yang berbeda yang digunakan dalam pengecekan.

Terdapat varian deterministik dari metode ini untuk nilai N lebih kecil dari batas tertentu. Untuk nilai N < 3215031751 hanya diperlukan pengecekan untuk nilai a = {2,3,5,7}. Desain algoritma ini memiliki kompleksitas $O(k(log_2(n) + a))$ dengan $k = |a|$. Dalam permasalahan ini algoritma *Miller Rabin* akan digunakan untuk mengecek keprimaan dari hasil faktor dari algoritma *Pollard Rho Brent*. Pada algoritma ini juga akan diaplikasikan perkalian modular dan pemangkatan modular yang akan ditunjukkan pada *Pseudocode* 2.3.

Pseudocode 2.3: Algoritma Miller Rabin

```
1: bool millerTest(d,N)
2: a ← 2 + RANDOM() % (N - 4)
3:     x ← (a^d) % N
4:     if (x = 1 or x =N - 1) then
5:     return true
6:     while (d != N -1) do
7:         x ← (x * x) % N
8:         d *= 2
9:         if (x = 1) then
10:        return false
11:        if (x = N -1) then
```

```
12:         return true
13:     end while
14:return false
15:bool isPrime(N, K)
16:     if (N = 2) then
17:     return true
18:     else if (N = 1 or N % 2 = 0) then
19:     return false
20:     d = N – 1
21:     while ( d % 2 = 0) do
22:         d /= 2
23:     end while
24:     for i ← 0 to K do
25:         if (!millerTest(d,N)) then
26:         return false
27:     end for
28:return true
```

2.2.10 Euler Totient Function

Ada 3 definisi dari *Euler Totient Function*:

- Definisi 1:
  Untuk $n \geq 1$, maka *Euler Totient Function* dari n ($\varphi(n)$) adalah banyaknya bilangan bulat positif yang tidak lebih dari n dan saling prima dengan n.
- Definisi 2:
  Dua bilangan bulat positif a dan b dapat dikatakan saling prima jika $gcd(a, b) = 1$.
- Definisi 3:
  Untuk $n \geq 1$, $\varphi(n)$ menyatakan banyaknya bilangan bulat positif r dimana $r \leq n$ dan $gcd(n, r) = 1$ [8].

Pada permasalahan ini *Euler Totient* dibutuhkan untuk menyelesaikan persamaan *Menon Identity* yaitu Persamaan 2.2. Untuk

pengerjaan Tugas Akhir ini akan digunakan 3 lemma dari *Euler Totient* antara lain:

- Lemma 1:
  Misalkan $G_k = \{r \in \mathbb{N} \mid 0 < r < kn \; and \; gcd(n,r) = 1\}$.
  Maka $|G_k| = k\varphi(n)$.
  Pembuktian: Misalkan H adalah kumpulan dari angka yang relatif prima dengan *n* dan tidak lebih dari *n.* Sehingga

  $$H = \{h \in \mathbb{N} \mid 0 < h < n \; and \; gcd(n,h) = 1\} \quad (2.7)$$

  Dari Definisi 3 dapat ditulis

  $$|H| = \varphi(n) \quad (2.8)$$

  Kelipatan berapapun dari *n* ditambah dengan *h* (yang akan berbentuk *kn* + *h*), akan terdapat faktor bukan prima *p* dari *n* karena jika $p|kn + h$, maka $p|kn$ dan $p|h$. Tapi kita tahu bahwa $p|kn$ dan $p \nmid h$ karena *p* adalah faktor dari *n* dan $gcd(n,h) = 1$. Sehingga $gcd(kn + h, n) = 1$. Karena hasil terakhir berlaku untuk semua $k \in \mathbb{N}$ dan semua $h \in H$, sehingga untuk setiap *k* pasti ada $|H| = \varphi(n)$ saling prima dengan *n* pada $G = \{r \in \mathbb{N} \mid (k-1)n < r < kn \; and \; gcd(n,r) = 1\}$.Artinya dalam interval apapun antara dua kelipatan *n* berturut-turut pasti ada $\varphi(n)$ yang saling prima dengan *n*. Karena itu $|G_k|$ sebanding dengan banyak interval dari *n* * $\varphi(n)$, atau $|G_k| = k\varphi(n)$.

- Lemma 2:
  Jika *p* adalah bilangan prima dan $p|n$, maka $\varphi(pn) = p\varphi(n)$.
  Pembuktian:
  Dapat dilihat bahwa semua angka yang saling prima dengan *pn* juga saling prima dengan *n.* Karena $gcd(pn,n) = n$ dan $p|n$ sehingga $gcd(n,r) = 1$ jika dan hanya jika $gcd(pn,r) = 1$ untuk semua bilangan asli *r*. Ada *p* interval, masing-masing dengan $\varphi(n)$ angka saling prima dengan *pn*, karena itu menurut Lemma 1dapat ditarik kesimpulan
  $G_p = \{r \in \mathbb{N} \mid 0 < r < pn \; and \; gcd(n,r) = 1\}, |G_p| = p\varphi(n)$

- Lemma 3:
  Jika p adalah bilangan prima dan $p \nmid n$, maka
  $\varphi(pn) = (p-1)\varphi(n)$.

  Pembuktian:
  Dari Lemma 1 dapat dilihat bahwa $p\varphi(n)$ adalah jumlah angka yang saling prima dengan *n* dan kurang dari *pn*. Dapat dilihat bahwa semua kelipatan *p* yang faktornya saling prima dengan *n* dapat dihitung, karena $gcd(pn, r) = 1$, jika dan hanya jika $gcd(n, r) = 1$ dan $gcd(p, r) = 1$. Misalkan daftar dari kelipatannya adalah $\{r_1p, r_2p, r_3p, r_4p, \ldots, r_{\varphi(n)}p\}$, dimana semua *r* saling prima dengan *n*. Himpunan tersebut memiliki $\varphi(n)$ angka saling prima dengan *n* dan 0 saling prima dengan *p*. Karena semuanya adalah kelipatan dari *p*. Selanjutnya akan dikurangi dengan $\varphi(n)$, sehingga menjadi $p\varphi(n) - \varphi(n) = (p-1)\varphi(n)$.

2.3 Strategi Penyelesaian dengan Menon Identity

Berikut penjelasan dari penyelesaian masalah dengan menggunakan penjabaran *Menon Identity*. *Menon Identity* dipilih karena identik dengan persamaan awal soal. Dapat kita lihat dari Persamaan 2.1 yang merupakan persamaan awal soal dan Persamaan 2.2 yang merupakan persamaan dari *Menon Identity*, dapat dilihat bahwa berdasarkan Persamaan 2.1 hal yang akan dicari adalah $\sum_{K=2,gcd(K,N)=1}^{N} gcd(K-1, N)$. Sedangkan Persamaan 2.2 atau persamaan dari *Menon Identity* hal yang dicari adalah $\sum_{i=1,gcd(i,n)=1}^{n} gcd(i-1, n)$. Dapat dilihat bahwa yang membedakan Persamaan 2.1 dan 2.2 hanyalah batas bawah dari sigma. Dimana pada Persamaan 2.1 batas bawah sigma adalah 2 dan pada Persamaan 2.2 batas bawah sigma adalah 1. Sehingga akan dilakukan pendekatan dari Persamaan 2.1 sehingga dapat dimunculkan Persamaan 2.2.

Untuk memodelkan Persamaan 2.1 sehingga dapat memunculkan *Menon Identity* di dalam persamaannya, maka Persamaan 2.1 akan diubah batas nilai *K*-nya menjadi 1 sehingga dapat dituliskan menjadi Persamaan 2.9.

$$\sum_{K=1,(K,N)=1}^{N} gcd(K-1,N) - N \quad (2.9)$$

Dimana (*K*,*N*) = *gcd*(*K*,*N*). Setelah diubah menjadi Persamaan 2.9, persamaan ini (2.9) dapat kita hubungkan dengan persamaan yang ada pada *Menon Identity* (2.2), dengan mengganti bagian $gcd(K-1,N)$ karena terdapat pada kedua Persamaan 2.9 dan 2.2. Sehingga dapat ditulis menjadi

$$\sum_{K=1,(K,N)=1}^{N} gcd(K-1,N) - N = d(N)\varphi(N) - N \quad (2.10)$$

Atau dapat dituliskan sebagai

$$\sum_{K=2,(K,N)=1}^{N} gcd(K-1,N) = d(N)\varphi(N) - N \quad (2.11)$$

2.4 Strategi Penyelesaian dengan Pollard Rho Brent dan Miller Rabin

Pada subbab ini akan dijelaskan tahap-tahap penyelesaian dengan *Pollard Rho Brent dan Miller Rabin* yang akan diaplikasikan untuk mencari *Menon Identity*.

2.4.1 Pemodelan Euler Totient

Berikut penjelasan dari model *Euler Totient* yang akan digunakan. Berangkat dari lemma-lemma yang telah dibuat pada subbab 2.2.10 akan dibuat sebuah model baru *Euler Totient* agar *Pollard Rho Brent* dan *Miller Rabin* dapat diaplikasikan.

Aplikasikan Lemma 2 ke semua faktor prima dari *N* $({p_1}^{k_1}, {p_2}^{k_2}, {p_3}^{k_3}, \ldots, {p_r}^{k_r})$ sehingga dapat ditulis menjadi:

$$\varphi(p^{k_1}\ldots p_r^{k_r}) = p_1^{k_1-1}p_2^{k_2-1}\ldots p_r^{k_r-1}\varphi(p_1p_2\ldots p_r) \quad (2.12)$$

Selanjutnya aplikasikan Lemma 3 kita mendapatkan

$$\varphi(p_1 p_2 \dots p_r\ ) = (p_1 - 1)(p_2 - 1)\dots(p_r - 1) \quad (2.13)$$

Jika Persamaan 2.13 diaplikasikan pada Persamaan 2.12 kita akan mendapatkan persamaan baru yaitu:

$$\varphi(n) = p_1^{k_1-1} \dots p_r^{k_r-1}(p_1 - 1)\dots(p_r - 1) \quad (2.14)$$

Selanjutnya semua ruas akan kita kalikan dengan angka 1 dalam bentuk $\frac{p_s}{p_s}$ untuk semua $1 \leq s \leq r$ [8].

$$\varphi(n) = (\frac{p_1}{p_1})\dots(\frac{p_r}{p_r})p_1^{k_1-1} \dots p_r^{k_r-1}(p_1 - 1)\dots(p_r - 1) \quad (2.15)$$

Selanjutnya akan kita uraikan Persamaan 2.15 menjadi Persamaan 2.16 - 2.18.

$$\varphi(n) = p^{k1} \dots p^{kr}(\frac{p_1-1}{p_1})\dots(\frac{p_r-1}{p_r}) \quad (2.16)$$

$$\varphi(n) = p^{k1} \dots p^{kr}(1 - \frac{1}{p_1})\dots(1 - \frac{1}{p_r}) \quad (2.17)$$

$$\varphi(n) = n\ (1 - \frac{1}{p_1})\dots(1 - \frac{1}{p_r}) \quad (2.18)$$

2.4.2 Aplikasi Pollard Rho Brent dan Miller Rabin untuk Mencari Euler Totient

Berikut penjelasan penyelesaian pencarian *Euler Totient* dengan menggunakan algoritma *Pollard Rho Brent* dan *Miller Rabin*. Dari Persamaan 2.18 yang telah dihasilkan, akan dibuat persamaan baru yaitu Persamaan 2.19 yang lebih jelas.

$$\varphi(N)\ =\ N\ *\ (1 - \frac{1}{p_1}) * (1 - \frac{1}{p_2}) *\dots.(1 - \frac{1}{p_x}) \quad (2.19)$$

dimana $p_1, p_2, p_3, \dots, p_x$ adalah semua faktor prima dari *N*. Selanjutnya dari Persamaan 2.19 dapat dilihat bahwa dibutuhkan untuk mencari faktor-faktor prima dari *N*, di sinilah penggunaan *Pollard Rho Brent* dan *Miller Rabin*.

Awalnya dengan algoritma *Pollard Rho Brent* akan dicari faktorisasi prima dari *N* dengan cara memasukannya ke dalam *Pseudocode* 2.2, selanjutnya setiap faktor tersebut akan di cek ke-primaannya (agar bisa menjamin semua faktor-faktornya adalah bilangan prima dan faktorisasi prima dari *N* bisa didapatkan) dengan menggunakan algoritma *Miller Rabin* dengan cara akan dimasukkan ke dalam *Pseudocode* 2.3.

2.4.3 Aplikasi Pollard Rho Brent untuk Mencari Banyak Faktor Positif

Jika hasil dari faktorisasi prima dari bilangan *N* sudah didapatkan dengan menggunakan algoritma *Pollard Rho Brent* dan *Miller Rabin*, maka hasil dari *Euler Totient* dari bilangan masukan *N* juga bisa didapatkan. Masalah selanjutnya adalah untuk mencari banyak faktor positif dari bilangan masukan *N*. Untuk mencari banyaknya faktor positif dari bilangan masukan *N* awalnya dapat dilakukan pemisalan dari faktorisasi prima dari *N*. Misalkan pemisalan tersebut adalah:

$$N = p_1^{a_1} p_2^{a_2} p_3^{a_3} \ldots p_n^{a_n} \tag{2.20}$$

dengan $p_1 p_2 p_3 \ldots p_n$ merupakan bilangan prima dan $a_1 a_2 a_3 \ldots a_n$ merupakan eksponen dari bilangan prima $p_1 p_2 p_3 \ldots p_n$, maka banyak faktor positif dari bilangan masukan *N* dapat dituliskan menjadi:

$$d(N) = (a_1 + 1)(a_2 + 1)(a_3 + 1)\ldots(a_n + 1) \tag{2.21}$$

BAB III
DESAIN

Pada bab ini akan dibahas tentang desain algoritma yang akan digunakan untuk menyelesaikan permasalahan yang diberikan.

3.1 Desain Umum Sistem

Pada subbab ini akan dijelaskan mengenai gambaran secara umum dari sistem. Sistem akan diawali dengan menerima masukan berupa nilai *T* yang menyatakan banyaknya kasus uji. *T* baris selanjutnya berisi nilai dari N. Nilai masukan ini mengikuti batasan pada subbab 1.3.

3.1.1 Desain Fungsi Perkalian Modular

Perkalian modular pada penjelasan subbab 2.2.5 dituliskan dalam fungsi MULMOD pada *Pseudocode* 3.1. Fungsi ini menerima masukan bilangan bulat *bil1* sebagai bilangan pertama, bilangan bulat *bil2* sebagai pengali dari bilangan pertama, dan bilangan bulat *mod* sebagai modulus perkalian modular. Keluaran dari fungsi ini adalah hasil perkalian modular.

Pseudocode 3.1: Fungsi MULMOD

Input : *bil1, bil2, mod*
Output : *res*
1: $res \leftarrow 1$
2: $res \leftarrow bil1 * (bil2 / mod)$
3: $res \leftarrow bil1 * bil2 - res * mod$
4: **if** $(res \geq mod)$ **then**

5: $res -= mod$
6: **if** $(res < 0)$ **then**
7: $res += mod$
8: **return** res

3.1.2 Desain Fungsi Pemangkatan Modular

Pemangkatan modular pada penjelasan subbab 2.2.4 dituliskan dalam fungsi POWER pada *Pseudocode* 3.2. Fungsi ini menerima masukan bilangan bulat *base* sebagai basis pemangkatan, bilangan bulat *exp* sebagai eksponen dari basis, dan bilangan bulat *mod* sebagai modulus pemangkatan modular. Keluaran dari fungsi ini adalah hasil pemangkatan modular.

Pseudocode 3.2: Fungsi POWER

Input : *base, exp, mod*
Output : *result*
1: $power \leftarrow base$
2: $result \leftarrow 1$
3: **do**
4: **if** (*exp* & 1)
5: $result$ =**mulmod**$(result, power, mod)$
6: $power$ =**mulmod**$(power, power, mod)$
7: $exp >>= 1$
8: **while** (*exp*)
9: **return** $result$

3.1.3 Desain Fungsi Faktor Persekutuan Terbesar

Fungsi ini digunakan untuk menemukan faktor persekutuan terbesar dari dua parameter masukan *u* dan *v*. Keluaran dari fungsi ini adalah *gcd* yang merupakan faktor persekutuan terbesar dari *u* dan *v*.

Fungsi ini sesuai dengan penjelasan pada subbab 2.2.6 dituliskan dalam fungsi GCD pada *Pseudocode* 3.3.

Pseudocode 3.3: Fungsi GCD

```
Input : u, v
Output : gcd
1: if (u = 0 or v = 0) then
2:     return u or v
3: for shift ← 0; (( u|| v) & 1) = 0; ++shift do
4:     u >>= 1
5:     v >>= 1
6: end for
7: while ((u & 1)= 0) do
8:     u >>= 1
9: end while
10:do
11:    while ((u & 1)= 0) do
12:        v >>= 1
13:    end while
14:    if (u < v) then
15         v−= u
16:    else
17:        diff ← u − v
18:        u ← v
19:        v ← diff
20:    v >>= 1
21:while (v ! = 0)
22:return u << shift
```

3.2 Desain Algoritma Pollard Rho Brent

Berdasarkan penjelasan pada subbab 2.2.8 telah dijelaskan bahwa algoritma *Pollard Rho Brent* ini digunakan untuk menemukan faktor *non-trivial* dari parameter masukan *N*. Fungsi akan terus berjalan selama faktor *g* yang ditemukan adalah faktor *trivial* dan akan digunakan pasangan bilangan *c* ,*y*, dan *m* yang berbeda jika nilai dari *g* dan *N* ternyata sama. Desain fungsi ini akan dituliskan dalam fungsi POLLARD_BRENT pada *Pseudocode* 3.4.

Pseudocode 3.4: Fungsi POLLARD_BRENT

```
Input : N
Output : g
1:if (!N &1) then
2:     return 2
3: y,c,m ← RANDOM()%( N−1)+1
4: g,r,q  ← 1
5: do
6:     x ← y
7:     for i ← 0 tor  do
8:         y ← (MULMOD(y, y, N) + c)
9:         y ≥ N ? y −= N : y
10:        k←0
11:    end for
12:    do
13:        ys = y
14:        k ← k + m
15:        for i ← 0 to min(m ,r − k) do
16.            y ← (MULMOD(y, y, N) + c)
17:            y ≥ N ? y −= N : y
18:            q← (MULMOD(q,abs(x − y),N))
19:        end for
```

20: $g \leftarrow$ **gcd** (q,N)
21: $k \leftarrow k + m$
22: **while** ($k < r$ & $g \leftarrow 1$)
23: $r <<= 1$
24: **while** ($g \leftarrow 1$)
25:**if** ($g \leftarrow n$) **then**
26: **do**
27: $cs \leftarrow$ MULMOD(ys, ys, N)
28:: $ys \leftarrow cs + c$
29: $ys \geq N$ **?** $ys -= N : y$
30: $g \leftarrow$ **gcd**(**abs**($x - ys$),N)
31: **if** ($g > 1$) **then break**
32: **while (**true)
33:**return** g

3.3 Desain Algoritma Miller Rabin

Desain dari algoritma ini akan dibagi menjadi 2 fungsi yaitu fungsi IS_PRIME yang akan dijelaskan pada subbab 3.3.1 dan fungsi WITNESS yang akan dijelaskan pada subbab 3.3.2.

3.3.1 Desain Fungsi IS_PRIME

Berdasarkan penjelasan pada subbab 2.2.9, fungsi yang satu ini akan digunakan untuk menentukan keprimaan suatu bilangan masukan *N*. Keluaran dari fungsi ini adalah **True** jika *N* adalah bilangan prima dan **False** jika *N* adalah bilangan komposit. Fungsi ini juga akan memanggil fungsi WITNESS untuk membantu mengecek apakah bilangan *N* kemungkinan prima atau tidak. Jika dari semua angka *a* yang diteruskan ke fungsi WITNESS memberikan *return* **True** maka keluaran dari fungsi IS_PRIME juga **True**, sebaliknya jika salah satu angka *a* yang diteruskan ke fungsi WITNESS ada yang bernilai **False** maka *return* dari fungsi IS_PRIME adalah **False.**

Desain fungsi ini dituliskan dalam fungsi IS_PRIME pada *Pseudocode* 3.5.

Pseudocode 3.5: Fungsi IS_PRIME

```
Input : N
Output : True or False
1: if ((((!(N & 1)) & N!=2 )||(N < 2) || (N % 3 = 0 & N!= 3)) then
2:      return False
3: if (N ≤ 3)
4:      return True
5: d ← N >> 1
6: s ← 1
7: while (!(d & 1)) do
8:      d >>= 1
9:      ++s
10:if (N < 1373653) then
11:     return WITNESS(N, s, d, 2) & WITNESS(N, s,
                d, 3)
12:if (N < 9080191) then
13:     return WITNESS(N, s, d, 31) & WITNESS(N, s,
                d, 73)
14:if (N < 4759123141) then
15:         return WITNESS(N, s, d, 2) & WITNESS(N, s,
                d, 7)& WITNESS(N, s, d, 61)
16:if (N < 1122004669633LL) then
17:     return WITNESS(N, s, d, 2) & WITNESS(N, s,
                d, 13) & WITNESS(N, s, d, 23) & WITNESS(N,
                s, d, 1662803)
18:if (N < 2152302898747LL) then
19:     return WITNESS(N, s, d, 2) & WITNESS(N, s,
                d, 3) & WITNESS(N, s, d, 5) & WITNESS(N, s,
                d, 7) & WITNESS(N, s, d, 11)
20:if (N < 3474749660383LL) then
```

21: **return** WITNESS(*N, s, d,* 2) & WITNESS(*N, s, d,* 3) & WITNESS(*N, s, d,* 5) & WITNESS(*N, s, d,* 7) & WITNESS(*N, s, d,* 11) &WITNESS(*N, s, d,* 13)
22: **return** WITNESS(*N, s, d,* 2) & WITNESS(*N, s, d,* 3) & WITNESS(*N, s, d,* 5) & WITNESS(*N, s, d,* 7) & WITNESS(*N, s, d,* 11) & WITNESS(*N, s, d,* 13) & WITNESS(*N, s, d,* 17)

3.3.2 Desain Fungsi WITNESS

Masukan dari fungsi WITNESS berasal dari fungsi IS_PRIME, dimana fungsi WITNESS ini berfungsi untuk mengembalikan nilai **True** atau **False**. Fungsi WITNESS akan menerima 4 masukan yaitu *N, s, d,* dan *a.* Masukan ini mengikuti Persamaan 2.6, dimana pada Persamaan 2.6 masukan *N, s, d,* dan *a* masing-masing mewakili variabel *N, j, d,* dan *a.* Keluaran dari fungsi ini adalah **True** jika *N* mungkin merupakan bilangan prima dan **False** jika *N* bukan bilangan prima. Desain fungsi ini dituliskan dalam fungsi WITNESS pada *Pseudocode* 3.6.

Pseudocode 3.6: Fungsi WITNESS

Input : *N, s, d, a*
Output : ***True*** *or* ***False***
1: $x \leftarrow$ POWER(a, d, N)
2: **while** (*s*) **do**
3: $y \leftarrow$ MULMOD(*x,x,N)*
4: **if** $(y = 1 \;\&\; x! = 1 \;\&\; x! = N - 1)$ **then**
5: **return False**
6: $x \leftarrow y$

7: --s
8: **end while**
9: **if** ($y! = 1$)
10: **return False**
11:**return True**

3.4 Desain Fungsi Faktorisasi Prima

Fungsi ini digunakan untuk melakukan faktorisasi prima dari parameter masukan *N*. Fungsi POLLARD_BRENT digunakan untuk mendapatkan faktor *non-trivial* dari *N*. Fungsi IS_PRIME digunakan untuk menentukan keprimaan dari bilangan hasil dari pemfaktoran POLLARD_BRENT dan untuk mengecek apakah angka *N* yang dimasukkan pada fungsi FACTOR adalah bilangan prima atau bukan. Jika *N* adalah bilangan prima maka akan langsung di*return* karena faktor dari bilangan prima adalah dirinya sendiri dan 1. Keluaran dari program ini adalah hasil faktorisasi prima yang disimpan di dalam *map m1*. Dimana nilai kunci dari *map m1* adalah basis dari faktorisasi prima *N* dan nilai dari nilai kunci tersebut adalah eksponen. Desain fungsi ini dituliskan dalam fungsi FACTOR dalam *Pseudocode* 3.7.

Pseudocode 3.7: Fungsi FACTOR

Input : *N*
Output : *m1*
1: **if** (*N*=1)
2: **return**
3: **if** (IS_PRIME(*N*)) **then**
4: *m1*[*N*]+= 1
5: **return**
6: d = POLLARD_BRENT(*N*)
7: FACTOR(*d*)
8: FACTOR(*N* / *d*)

3.5 Desain Fungsi Main

Fungsi ini adalah fungsi utama yang akan dipanggil pertama ketika mengeksekusi program. Fungsi ini bertugas untuk menerima masukan dan juga melakukan pemanggilan fungsi FACTOR. Setelah menerima masukan, maka masukan tersebut akan dimasukkan ke dalam fungsi FACTOR. Selain itu akan ada 2 *map* (*m1* dan *it1*), *m1* akan digunakan untuk menampung nilai dari faktor-faktor *N* dan *it1* akan digunakan untuk perulangan yang ada di fungsi MAIN. Karena *map m1* adalah keluaran dari fungsi FACTOR yang berisi faktorisasi prima dari *N*. Maka selanjutnya fungsi MAIN akan menjalankan Persamaan 2.19 dan Persamaan 2.21 untuk mendapatkan nilai *Euler Totient* dan banyaknya faktor positif dari *N*. Desain dari fungsi ini akan ditulis dalam fungsi MAIN pada *Pseudocode* 3.8.

Pseudocode 3.8: Fungsi MAIN

Input : *T*,*N*
Output : *answer*
1: $T =$ INPUT()
2: **for** ($i \leftarrow 1$ *to* T) **do**
3: $N =$ INPUT()
4: $res \leftarrow 1$
5: $map::it1$
6: $res \leftarrow res * N$
7: FACTOR(*N*)
8: $x \leftarrow 1$
9: $flag \leftarrow$**False**
10: **for** ($it1 \rightarrow m1$.BEGIN() ***to*** $m1$.END()) **do**
11: $it \leftarrow it1 -> first$
12: $res /= it$
13: $res *= (it - 1)$
14: $x *= (it -> second + 1)$

15: **end for**
16: $answer \leftarrow ((x * res) - N)$
17: $m1$.CLEAR()
18: **end for**

Variabel *res* dan *x* masing-masing melambangkan nilai dari *Euler Totient* dari *N* dan banyak faktor positif dari *N*. *Map m1* akan dikosongkan setiap selesai perulangan agar dapat diisi lagi dengan faktorisasi prima dari angka *N* yang baru nantinya.

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas implementasi dari *pseudocode* bab 3 dalam bahasa pemrogaman C++.

4.1 Lingkungan Implementasi

Lingkungan implementasi dalam pembuatan Tugas Akhir ini meliputi perangkat keras dan perangkat lunak yang digunakan adalah sebagai berikut:

1. Perangkat Keras:
   - Processor Intel ® Core i5-M480 CPU @ 2.67GHz (4 CPUs), 2.7GHz
   - Random Access Memory 4096MB
2. Perangkat Lunak:
   - Sistem Operasi Windows 7 64-bit
   - *Text editor* Notepad ++
   - IDE Ideone.com
   - Kompiler GCC version 8.3.0 (Ideone.com)

4.2 Implementasi Program Utama

Subbab ini menjelaskan implementasi program utama. Program ini digunakan untuk menyelesaikan permasalahan SPOJ-ADAHW (*Ada and Homework*). Hal yang perlu diperhatikan pada tahap implementasi ini adalah bahasa pemrograman C++ tidak mendukung bilangan bulat yang melebihi 2^{64}. Pada beberapa kode sumber akan digunakan C++ *boost library* seperti yang dijelaskan pada subbab 2.2.2 untuk menyelesaikan permasalahan ini.

```
1. #include <cmath>
2. #include <iostream>
3. #include <map>
```

```
4. #include <limits.h>
5. #include<boost/multiprecision/cpp_int.hpp>
```

Kode Sumber 4.1: *Header* yang diperlukan

Header cmath berisi fungsi-fungsi untuk operasi matematika seperti fungsi sqrt. *Header iostream* berisi fungsi standar input output operasi yang digunakan dalam bahasa C++. *Header map* digunakan agar dapat mengakses struktur data *map* pada bahasa C++. *Header limits* digunakan agar dapat mengetahui batas-batas dari jenis bilangan bulat yang ada pada bahasa C++ seperti *long long* adalah 2^{64}. *Header boost/multiprecision/cpp_int.hpp* digunakan untuk mengatasi bilangan bulat yang lebih dari *long long* (2^{64}).

```
1. inline int64 scan(){
2.   int64 z=0;
3.   char c;
4.   do{ c=getchar_unlocked(); } while(c<'0');
5.   for(;c>='0';c=getchar_unlocked()) z =
   (z<<3) + (z<<1) + (c&15);
6.   return z;
7. }
8. void put_uint64(int64 n) {
9.   char stack[20];
10.    int top = 0;
11.    if(n == 0) {
12.      putchar_unlocked('0');
13.    } else {
14.      while(n > 0) {
15.        stack[top++] = n % 10 + '0';
16.        n /= 10;
17.      }
18.      while(top > 0) {
19.        putchar_unlocked(stack[--top]);
20.      }
21.    }
22.    putchar_unlocked('\n');
23.  }
```

Kode Sumber 4.2: Fungsi SCAN dan PUT_UINT64

Fungsi SCAN digunakan untuk mengambil masukan bilangan dengan tipe data *int64* dan fungsi PUT_UINT64 digunakan untuk menampilkan bilangan dengan tipe data *int64*.

4.2.1 Preprocessor

Preprocessor yang diperlukan untuk masalah kali ini dapat dilihat pada Kode Sumber 4.3.

```
1. namespace mp=boost::multiprecision;
2. using namespace std;
3. #define u128 __int128
4. typedef long long int int64;
5. map<int64,int>m1;
```

Kode Sumber 4.3: *Preprocessor* yang diperlukan

namespace mp=boost::multiprecision digunakan untuk mempersingkat penulisan *boost::multiprecision* menjadi *mp*. *using namespace std* digunakan agar tidak perlu menulis *std::* di awal setiap baris. *#define u128 _int128* digunakan untuk membuat tipe data *_int128* dapat dituliskan dengan *u128*. *typedef long long int int64* digunakan untuk membuat *long long int* dapat juga dituliskan sebagai *int64*. *map<int64,int>m1* digunakan sebagai *map* yang bersifat global yang nantinya akan diisi dengan faktorisasi prima dari suatu angka masukan.

4.3 Implementasi Fungsi

Pada subbab ini akan dijelaskan implementasi dari fungsi-fungsi yang sesuai dengan desainnya pada bab 3.

4.3.1 Fungsi Perkalian Modular

Berdasarkan penjelasan pada subbab 3.1.1 fungsi ini digunakan untuk menentukan hasil perkalian dari *a* dan *b* dan akan dimodulus *mod*. Implementasi fungsi ini dituliskan pada Kode Sumber 4.4.

```
1. int64 mulmod(int64 a, int64 b, int64 mod) {
2.     int64 res = (a * ((long double) b / (long
   double) mod));
3.     res = a * b - res * mod;
4.     if (res >= mod) res -= mod;
5.     if (res < 0) res += mod;
6.     return res;
7. }
```

Kode Sumber 4.4: Fungsi MULMOD

4.3.2 Fungsi Pemangkatan Modular

Berdasarkan penjelasan pada subbab 3.1.2 fungsi ini digunakan untuk menentukan hasil pemangkatan dari *a* terhadap eksponen *n* dan akan dimodulus *mod*. Implementasi fungsi ini dituliskan pada Kode Sumber 4.5.

```
1. inline int64 power(int64 a, int64 n,int64
   mod)
2. {
3.     int64 power = a;
4.     int64 result = 1;
5.
6.     do{
7.         if (n & 1)
8.             result = mulmod(result , power
   ,mod);
9.         power = mulmod(power , power, mod);
10.          n >>= 1;
11.      }while (n);
12.      return result;
13.  }
```

Kode Sumber 4.5: Fungsi POWER

4.3.3 Fungsi Faktor Persekutuan Terbesar

Berdasarkan penjelasan pada subbab 3.1.3 fungsi ini digunakan untuk mencari faktor persekutuan terbesar dari *u* dan *v*. Implementasi fungsi ini dituliskan pada Kode Sumber 4.6.

```
1. inline int64 gcd(int64 u,int64 v)
2. {
3.     int64 shift, diff;
4.     if (u == 0 || v == 0)
5.         return u | v;
6.     for (shift = 0; ((u | v) & 1) == 0;
   ++shift)
7.     {
8.         u >>= 1;
9.         v >>= 1;
10.      }
11.      while ((u & 1) == 0)
12.          u >>= 1;
13.
14.
15.      do {
16.          while ((v & 1) == 0)
17.              v >>= 1;
18.          if (u < v)
19.              v -= u;
20.          else
21.          {
22.              diff = u - v;
23.              u = v;
24.              v = diff;
25.          }
26.          v >>= 1;
27.      } while (v != 0);
28.      return u << shift;
29.  }
```

Kode Sumber 4.6: Fungsi GCD

4.3.4 Fungsi Pollard Rho Brent

Berdasarkan penjelasan pada subbab 3.2 fungsi ini digunakan untuk mencari faktor *non-trivial* dari bilangan *N*. Keluaran dari fungsi ini adalah bilangan *g* yang merupakan faktor *non-trivial* dari *N*. Implementasi fungsi ini dituliskan pada Kode Sumber 4.7.

```
1. inline int64 pollard_brent(int64 N)
2. {
3.   int64 g,r,q,x,ys,i,k,cs,xx;
4.   if(!(N&1)) return 2;
5.   int64 y = rand()%(N-1)+1;
6.   int64 c = rand()%(N-1)+1;
7.   int64 m = rand()%(N-1)+1;
8.   g = 1;
9.   r = 1;
10.    q = 1;
11.    do{
12.      x = y;
13.      for(i=0; i < r; i++)
14.      y = (mulmod(y,y,N)+c);
15.      y>=N?y-=N:y;
16.      k = 0;
17.      do{
18.        ys = y;
19.        for(i=0;i<min(m,r-k);i++){
20.          y = (mulmod(y,y,N)+c);
21.          y>=N?y-=N:y;
22.          q = mulmod(q,abs(x-y),N);
23.        }
24.        g = gcd(q,N);
25.        k = k+m;
26.      }while(k < r && g==1);
27.      r<<= 1;
28.    }while(g==1);
29.    if(g==N){
30.      do{
31.        cs = mulmod(ys,ys,N);
32.        ys = (cs+c);
```

```
33.          ys>=N?ys-=N:y;
34.          g = gcd(abs(x-ys),N);
35.          if(g>1)break;
36.        }while(true);
37.     }
38.     return g;
39.  }
```

Kode Sumber 4.7: Fungsi POLLARD_BRENT

4.3.5 Fungsi IS_PRIME

Berdasarkan penjelasan pada subbab 3.3.1 fungsi ini digunakan untuk menentukan keprimaan dari *n*. Keluaran dari fungsi ini adalah **True** jika *n* adalah bilangan prima dan **False** jika *n* bukan bilangan prima. Implementasi dari fungsi ini dituliskan pada Kode Sumber 4.8.

```
1. bool is_prime(int64 n)
2. {
3.     if ((((!(n & 1)) && n != 2 ) || (n < 2) ||
   (n % 3 == 0 && n != 3))
4.         return false;
5.     if (n <= 3)
6.         return true;
7.
8.     int64 d = n>>1;
9.     int64 s = 1;
10.      while (!(d & 1)) {
11.          d>>=1;
12.          ++s;
13.      }
14.
15.      if (n < 1373653)
16.          return witness(n, s, d, 2) &&
   witness(n, s, d, 3);
17.      if (n < 9080191)
18.          return witness(n, s, d, 31) &&
   witness(n, s, d, 73);
```

```
19.      if (n < 4759123141LL)
20.          return witness(n, s, d, 2) &&
   witness(n, s, d, 7) && witness(n, s, d, 61);
21.      if (n < 1122004669633LL)
22.          return witness(n, s, d, 2) &&
   witness(n, s, d, 13) && witness(n, s, d, 23)
   && witness(n, s, d, 1662803);
23.      if (n < 2152302898747LL)
24.          return witness(n, s, d, 2) &&
   witness(n, s, d, 3) && witness(n, s, d, 5) &&
   witness(n, s, d, 7) && witness(n, s, d, 11);
25.      if (n < 3474749660383LL)
26.          return witness(n, s, d, 2) &&
   witness(n, s, d, 3) && witness(n, s, d, 5) &&
   witness(n, s, d, 7) && witness(n, s, d, 11)
   && witness(n, s, d, 13);
27.      return witness(n, s, d, 2) &&
   witness(n, s, d, 3) && witness(n, s, d, 5) &&
   witness(n, s, d, 7) && witness(n, s, d, 11)
   && witness(n, s, d, 13) && witness(n, s, d,
   17);
28.  }
```

Kode Sumber 4.8: Fungsi IS_PRIME

4.3.6 Fungsi WITNESS

Berdasarkan penjelasan pada subbab 3.3.2 fungsi ini digunakan untuk menentukan keprimaan dari *n* dengan menggunakan Persamaan 2.6, sehingga untuk menentukan keprimaan dari *n* ini juga diperlukan variabel *s*, *d*, dan *a*. Keluaran dari fungsi ini adalah **True** jika *n* mungkin merupakan bilangan prima dan **False** jika *n* bukan bilangan prima. Implementasi dari fungsi ini dituliskan pada Kode Sumber 4.9.

```
1. bool witness(int64 n, int64 s, int64 d, int64
   a)
2. {
3.      int64 x = power(a, d, n);
4.      int64 y;
```

```
5.
6.      while (s) {
7.          y = mulmod(x , x, n);
8.          if (y == 1 && x != 1 && x != n-1)
9.              return false;
10.           x = y;
11.           --s;
12.       }
13.       if (y != 1)
14.           return false;
15.       return true;
16.   }
```

Kode Sumber 4.9: Fungsi WITNESS

4.3.7 Fungsi Faktorisasi Prima

Berdasarkan penjelasan pada subbab 3.4 fungsi ini digunakan untuk melakukan faktorisasi prima bilangan *n* dan disimpan ke dalam map *m1*. Implementasi dari fungsi ini dituliskan pada Kode Sumber 4.10.

```
1. void factor(int64 n){
2.    if(n==1)
3.       return;
4.    if(is_prime(n)){
5.       m1[n]+=1;
6.       return;
7.    }
8.       int64 d=pollard_brent(n);
9.    factor(d);
10.      factor(n/d);
11.   }
```

Kode Sumber 4.10: Fungsi FACTOR

4.4 Implementasi Fungsi Main

Berdasarkan penjelasan pada subbab 3.5 fungsi ini adalah fungsi utama yang dieksekusi saat program dijalankan, fungsi ini meliputi proses masukan data dan proses akhir dari semua fungsi-fungsi yang lain. Masukan yang ada pada fungsi ini adalah *t* sebagai banyaknya kasus uji dan *n* adalah angka uji. Implementasi dari fungsi ini dituliskan pada Kode Sumber 4.11.

```
1. int main()
2. {
3.   int t;
4.   long long n,m;
5.   cin >> t;
6.   for(;t--;){
7.     scanf("%lld",&n);
8.     m = n;
9.     map<long long,int>::iterator it1;
10.      mp::cpp_int  res=1;
11.      res = res*n;
12.      factor(n);
13.      mp::cpp_int x=1;
14.       bool flag=false;
15.
    for(it1=m1.begin();it1!=m1.end();it1++){
16.        int64 it = it1->first;
17.        res/=it;
18.        res*=(it-1);
19.        cout<<"it="<<it<<endl;
20.        cout<<"res="<<res<<endl;
21.        cout<<"itsecond="<<it1-
  >second<<endl;
22.        x*=(it1->second + 1);
23.        cout<<"x="<<x<<endl;
24.      }
25.      cout<<((x*res)-m)<<endl;;
26.      m1.clear();
27.    }
28.    return 0;
```

```
29.  }
```

Kode Sumber 4.11: Fungsi MAIN

[*Halaman ini sengaja dikosongkan*]

BAB V
UJICOBA DAN EVALUASI

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

5.1 Lingkungan Uji Coba

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

1. Perangkat Keras:
   - *Processor* Intel Xeon E3-1220 v5 (5CPUs)
   - Random Access Memory 1536 MB

2. Perangkat Lunak:
   - Kompiler GCC 6.3.0

Lingkungan uji coba yang digunakan untuk uji coba kinerja menggunakan komputer pribadi milik penulis dengan spesifikasi sebagai berikut:

1. Perangkat Keras:
   - Processor Intel ® Core i5-M480 CPU @ 2.67GHz (4 CPUs), 2.7GHz
   - Random Access Memory 4096MB
2. Perangkat Lunak:
   - Sistem Operasi Windows 7 64-bit
   - *Text Editor* Notepad++
   - IDE Ideone.com
   - Kompiler GCC Version 8.3.0 (Ideone.com)

5.2 Skenario Uji Coba

Pada bagian ini akan dijelaskan skenario yang akan digunakan untuk melakukan pengujian terhadap implementasi yang dibuat untuk permasalahan *Ada and Homework*.

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 dikirim dengan memasukkan kasus uji dengan batasan yang sudah dijabarkan pada subbab 1.3 yaitu:

1. *T* sebagai banyaknya kasus uji berupa bilangan bulat dengan rentang antara antara 1 sampai 1000.
2. *N* sebagai angka yang akan dimasukkan berupa bilangan bulat dengan rentang antara antara 2 sampai 10^{18}.
3. *K* sebagai himpunan bilangan bulat dari 2 sampai N.

Uji coba kinerja akan dilakukan dengan mengirimkan kode sumber hasil implementasi program ke situs penilaian SPOJ sebanyak 10 kali kemudian menganalisa performa 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 sama dengan contoh keluaran yang ada pada permasalahan SPOJ Ada and Homework. Kasus uji dapat dilihat pada Tabel 5.1.

Tabel 5.1A: Tabel Uji Coba

Masukan	Keluaran
11 2 5	 0 3

Tabel 5.1 B: Tabel Uji Coba

6	2
7	5
8	8
10	6
50	70
100	260
1000	5400
524288	4718592
945406969379503350	1381966975399059833610

Pada program implementasi yang dibuat, sistem akan membaca angka pertama dari masukan yaitu 11 (*T*) sebagai banyaknya kasus uji coba. Setelah itu akan masuk ke dalam perulangan sebanyak 11 kali untuk mendapatkan 11 angka masukan lagi yang nantinya akan diproses sebagai *N* atau angka yang akan dicari keluarannya. *N* pertama yang dimasukkan adalah 2. Setelah angka 2 dimasukkan sistem akan membuat map baru yaitu map *it1* dan map *m1*, setelah itu angka 2 akan dimasukkan ke dalam fungsi FACTOR untuk dicari faktorisasi primanya. Di fungsi FACTOR angka 2 akan dicek dulu keprimaannya, karena 2 adalah bilangan prima maka akan langsung di *return*. Setelah didapatkan faktorisasi prima dari angka 2, maka akan disimpan di dalam map *m1*. Kondisi *m1* setelah menyelesaikan fungsi FACTOR akan ditunjukkan pada Tabel 5.2.

Tabel 5.2: Kondisi map *m1* setelah 2 menyelesaikan fungsi FACTOR

Nilai kunci	2
Nilai	1

m1 kemudian akan dimasukkan ke dalam perulangan untuk menghitung nilai *Euler Totient* dan banyaknya faktor positif dari 2.

$$N = p_1^{a_1} p_2^{a_2} p_3^{a_3} \ldots p_n^{a_n} \quad (5.1)$$

Jika faktorisasi prima dari *n* dituliskan seperti Persamaan 5.1, maka akan ada 4 variabel yang digunakan untuk menghitung nilai *Euler Totient* dari 2, yaitu *it->first* yang merupakan nilai dari basis-basis dari faktorisasi prima *N* $(p_1, p_2, p_3, \ldots, p_n)$, *it->second* adalah nilai dari eksponen dari faktorisasi prima *N* $(a_1, a_2, a_3, \ldots, a_n)$, *res* adalah nilai dari *Euler Totient* dari *N* sesuai dengan penjelasan pada subbab 2.4.2 pada Persamaan 2.19, dan *x* adalah banyaknya faktor positif dari *N* sesuai dengan penjelasan pada subbab 2.4.3 pada Persamaan 2.21. Nilai dari *it->first*, *it->second*, *res*, dan *x* dari masukan pertama yaitu angka 2 akan ditunjukkan pada Tabel 5.3.

Tabel 5.3: Nilai *it->first*, *it->second*, *res*, dan *x* dari 2

Iterasi ke-	*it->first*	*it->second*	*res*	*x*
1	2	1	1	2

Setelah ke-4 variabel di atas sudah terisi dan sudah keluar dari perulangan maka map *m1* akan dikosongkan untuk angka masukan selanjutnya. Kemudian sistem akan memberikan keluaran sesuai dengan pembahasan pada subbab 2.3 maka sistem akan memberikan keluaran sesuai dengan Persamaan 2.11 yang akan dimodelkan oleh Persamaan 5.2.

$$answer = (res * x) - N \quad (5.2)$$

Sehingga sistem akan memberikan keluaran 0 untuk angka masukan 2. Setelah sistem memberikan keluaran pada angka masukan pertama, sistem akan kembali menerima masukan lagi dengan angka ke-2 yaitu 5. Masukan 5 juga akan diproses sama dengan angka pertama yaitu 2 begitu juga dengan sisa angka masukan. Nilai dari map *m1* dan nilai dari *it->first*, *it->second*, *res*, dan *x* dari masukan akan ditunjukkan pada Tabel 5.4 dan Tabel 5.5.

Tabel 5.4: Kondisi map *m1* setelah 5 menyelesaikan fungsi FACTOR

Nilai kunci	5
Nilai	1

Tabel 5.5: Nilai *it->first*, *it->second*, *res*, dan *x* dari 5

Iterasi ke-	*it->first*	*it->second*	*res*	*x*
1	5	1	4	2

Berdasarkan Persamaan 5.2 keluaran dari masukan 5 adalah 3. Masukan 6 adalah masukan yang akan diproses selanjutnya. Nilai dari map *m1* dan nilai dari *it->first*, *it->second*, *res*, dan *x* dari masukan akan ditunjukkan pada Tabel 5.6 dan Tabel 5.7.

Tabel 5.6: Kondisi map *m1* setelah 6 menyelesaikan fungsi FACTOR

Nilai kunci	2	3
Nilai	1	1

Tabel 5.7: Nilai *it->first*, *it->second*, *res*, dan *x* dari 6

Iterasi ke-	*it->first*	*it->second*	*res*	*x*
1	2	1	3	2
2	3	1	2	4

Berdasarkan Persamaan 5.2 keluaran dari masukan 6 adalah 2. Masukan 7 adalah masukan yang akan diproses selanjutnya. Nilai dari map *m1* dan nilai dari *it->first*, *it->second*, *res*, dan *x* dari masukan akan ditunjukkan pada Tabel 5.8 dan Tabel 5.9.

Tabel 5.8: Kondisi map *m1* setelah 7 menyelesaikan fungsi FACTOR

Nilai kunci	7
Nilai	1

Tabel 5.9: Nilai *it->first*, *it->second*, *res*, dan *x* dari 7

Iterasi ke-	*it->first*	*it->second*	*res*	*x*
1	7	1	6	2

Berdasarkan Persamaan 5.2 keluaran dari masukan 7 adalah 5. Masukan 8 adalah masukan yang akan diproses selanjutnya. Nilai dari map *m1* dan nilai dari *it->first*, *it->second*, *res*, dan *x* dari masukan akan ditunjukkan pada Tabel 5.10 dan Tabel 5.11.

Tabel 5.10: Kondisi map *m1* setelah 8 menyelesaikan fungsi FACTOR

Nilai kunci	2
Nilai	3

Tabel 5.11: Nilai *it->first*, *it->second*, *res*, dan *x* dari 8

Iterasi ke-	*it->first*	*it->second*	*res*	*x*
1	2	3	4	4

Berdasarkan Persamaan 5.2 keluaran dari masukan 8 adalah 8. Masukan 10 adalah masukan yang akan diproses selanjutnya. Nilai dari map *m1* dan nilai dari *it->first*, *it->second*, *res*, dan *x* dari masukan akan ditunjukkan pada Tabel 5.12 dan Tabel 5.13.

Tabel 5.12: Kondisi map *m1* setelah 10 menyelesaikan fungsi FACTOR

Nilai kunci	2	5
Nilai	1	1

Tabel 5.13: Nilai *it->first*, *it->second*, *res*, dan *x* dari 10

Iterasi ke-	*it->first*	*it->second*	*res*	*x*
1	2	1	5	2
2	5	1	4	4

Berdasarkan Persamaan 5.2 keluaran dari masukan 10 adalah 6. Masukan 50 adalah masukan yang akan diproses selanjutnya. Nilai dari map *m1* dan nilai dari *it->first, it->second, res,* dan *x* dari masukan akan ditunjukkan pada Tabel 5.14 dan Tabel 5.15.

Tabel 5.14: Kondisi map *m1* setelah 50 menyelesaikan fungsi FACTOR

Nilai kunci	2	5
Nilai	1	2

Tabel 5.15: Nilai *it->first, it->second, res,* dan *x* dari 50

Iterasi ke-	*it->first*	*it->second*	*res*	*x*
1	2	1	25	2
2	5	2	20	6

Berdasarkan Persamaan 5.2 keluaran dari masukan 50 adalah 70. Masukan 100 adalah masukan yang akan diproses selanjutnya. Nilai dari map *m1* dan nilai dari *it->first, it->second, res,* dan *x* dari masukan akan ditunjukkan pada Tabel 5.16 dan Tabel 5.17.

Tabel 5.16: Kondisi map *m1* setelah 100 menyelesaikan fungsi FACTOR

Nilai kunci	2	5
Nilai	2	2

Tabel 5.17: Nilai *it->first, it->second, res,* dan *x* dari 100

Iterasi ke-	*it->first*	*it->second*	*res*	*x*
1	2	2	50	2
2	5	2	40	9

Berdasarkan Persamaan 5.2 keluaran dari masukan 100 adalah 260. Masukan 1000 adalah masukan yang akan diproses selanjutnya. Nilai dari map *m1* dan nilai dari *it->first, it->second, res,* dan *x* dari masukan akan ditunjukkan pada Tabel 5.18 dan Tabel 5.19.

Tabel 5.18: Kondisi map *m1* setelah 1000 menyelesaikan fungsi FACTOR

Nilai kunci	2	5
Nilai	3	3

Tabel 5.19: Nilai *it->first, it->second, res,* dan *x* dari 1000

Iterasi ke-	*it->first*	*it->second*	*res*	*x*
1	2	3	500	2
2	5	3	400	16

Berdasarkan Persamaan 5.2 keluaran dari masukan 1000 adalah 5400. Masukan 524288 adalah masukan yang akan diproses selanjutnya. Nilai dari map *m1* dan nilai dari *it->first, it->second, res,* dan *x* dari masukan akan ditunjukkan pada Tabel 5.20 dan Tabel 5.21.

Tabel 5.20: Kondisi map *m1* setelah 524288 menyelesaikan fungsi FACTOR

Nilai kunci	2
Nilai	19

Tabel 5.21: Nilai *it->first*, *it->second*, *res*, dan *x* dari 524288

Iterasi ke-	*it->first*	*it->second*	*res*	*X*
1	2	19	262144	20

Berdasarkan Persamaan 5.2 keluaran dari masukan 524288 adalah 4718592. Masukan 945406969379503350 adalah masukan yang akan diproses selanjutnya. Nilai dari map *m1* dan nilai dari *it->first*, *it->second*, *res*, dan *x* dari masukan akan ditunjukkan pada Tabel 5.22 dan Tabel 5.23.

Tabel 5.22: Kondisi map *m1* setelah 945406969379503350 menyelesaikan fungsi FACTOR

Nilai kunci	2	3	5	17	19	29	37	59	73	79	97
Nilai	1	1	2	1	2	2	1	1	1	1	1

Tabel 5.23A: Nilai *it->first*, *it->second*, *res*, dan *x* dari 945406969379503350

Iterasi ke-	*it->first*	*it->second*	*res*	*x*
1	2	1	47270348468975 1675	2
2	3	1	31513565645983 4450	4
3	5	2	25210852516786 7560	12
4	17	1	23727861192269 8880	24
5	19	2	22479026392676 7360	72
6	29	2	21703887551549 9520	216
7	37	1	21117295996102 6560	432
8	59	1	20759375724982 2720	864
9	73	1	20475000715051 0080	1728
10	79	1	20215823490809 8560	3456

Tabel 5.23B: Nilai *it->first*, *it->second*, *res*, dan *x* dari 945406969379503350

11	97	1	2000741293 93582080	6912

Berdasarkan Persamaan 5.2 keluaran dari masukan 945406969379503350 adalah 1381966975399059833610.

5.2.2 Uji Coba Kebenaran

Uji coba kebenaran dilakukan dengan mengirimkan kode hasil implementasi program ke situs SPOJ. Permasalahan yang diselesaikan adalah *Ada and Homework*. Setelah mengirimkan kode sumber maka akan mendapatkan umpan balik dari SPOJ seperti yang ada pada Gambar 5.1.

Gambar 5.1: Hasil umpan balik dari hasil uji kebenaran di SPOJ

Dari hasil uji coba yang dilakukan kode sumber mendapatkan umpan balik *Accepted*. Waktu yang diperlukan program adalah 0,52 detik dan memori yang dibutuhkan program adalah 4,6MB.

5.2.3 Uji Coba Kinerja

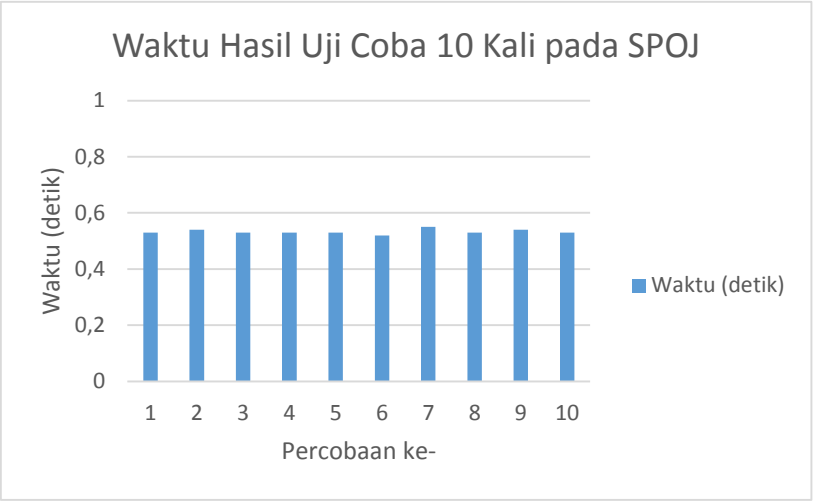
Setelah itu kode sumber yang sama akan dikirimkan sampai 10 kali untuk melihat variasi waktu dan memori yang dibutuhkan. Hasil uji coba dengan mengirimkan kode sumber sebanyak 10 kali, dapat dilihat pada Gambar 5.2, Tabel 5.24, Gambar 5.3, dan Gambar 5.4.

ID		DATE	PROBLEM	RESULT	TIME	MEM	LANG
25084287		2019-12-17 12:24:43	Ada and Homework	accepted edit ideone it	0.53	4.6M	CPP14
25084285		2019-12-17 12:24:15	Ada and Homework	accepted edit ideone it	0.54	4.7M	CPP14
25084284		2019-12-17 12:23:57	Ada and Homework	accepted edit ideone it	0.53	4.7M	CPP14
25084281		2019-12-17 12:23:33	Ada and Homework	accepted edit ideone it	0.53	4.6M	CPP14
25084279		2019-12-17 12:23:08	Ada and Homework	accepted edit ideone it	0.53	4.7M	CPP14
25084276		2019-12-17 12:22:44	Ada and Homework	accepted edit ideone it	0.52	4.7M	CPP14
25084270		2019-12-17 12:21:55	Ada and Homework	accepted edit ideone it	0.55	4.7M	CPP14
25084252		2019-12-17 12:19:28	Ada and Homework	accepted edit ideone it	0.53	4.7M	CPP14
25084248		2019-12-17 12:18:52	Ada and Homework	accepted edit ideone it	0.54	4.6M	CPP14
25084243		2019-12-17 12:18:17	Ada and Homework	accepted edit ideone it	0.53	4.7M	CPP14

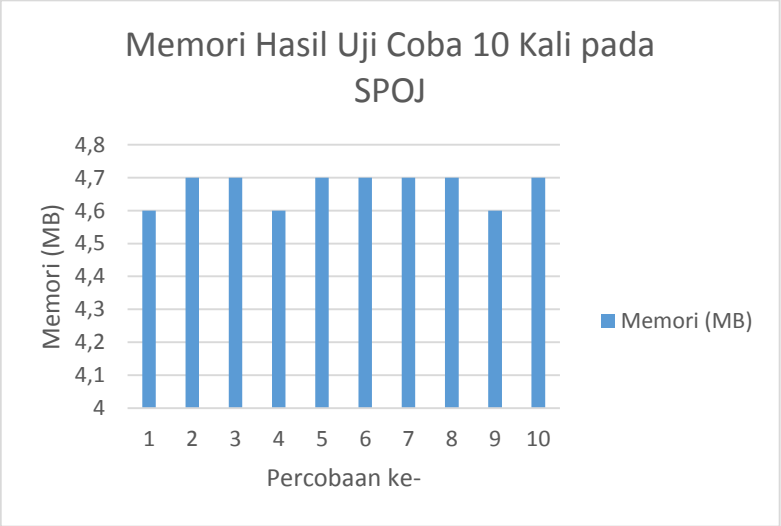
Gambar 5.2: Hasil umpan balik dari uji kebenaran 10 kali pada SPOJ

Tabel 5.24: Tabel waktu dan memori dari uji coba kebenaran 10 kali pada SPOJ

Percobaan ke-	Waktu(detik)	Memori(MB)
1	0,53	4,6
2	0,54	4,7
3	0,53	4,7
4	0,53	4,6
5	0,53	4,7
6	0,52	4,7
7	0,55	4,7
8	0,53	4,7
9	0,54	4,6
10	0,53	4,7



Gambar 5.3: Grafik waktu hasil uji kebenaran sebanyak 10 kali pada SPOJ



Gambar 5.4: Grafik memori hasil uji kebenaran sebanyak 10 kali pada SPOJ

Dari hasil uji coba kebenaran sebanyak 10 kali pada SPOJ, maka dapat dilihat bahwa rata-rata memori yang dibutuhkan oleh program adalah 4,7 MB, sedangkan waktu rata-rata yang dibutuhkan program adalah 0,53 detik.

[*Halaman ini sengaja dikosongkan*]

BAB VI
KESIMPULAN

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

6.1 Kesimpulan

Dari analisis dan uji coba yang dilakukan terhadap implementasi algoritma untuk menyelesaikan permasalahan *Ada and Homework* (ADAHW), dapat diambil kesimpulan sebagai berikut:

1. Permasalahan *Ada and Homework* pada situs SPOJ dapat diselesaikan dengan memodelkan permasalahan tersebut dengan *Euler Totient* yang ada pada *Menon Identity.*
2. Algoritma *Pollard Rho Brent* dan *Miller Rabin* dapat menyelesaikan model dari permasalahan *Ada and Homework* yang telah dimodelkan dengan *Menon Identy*.
3. Rata-rata waktu dari implementasi dari algoritma *Pollard Rho Brent* dan *Miller Rabin* kurang dari batas waktu yang diberikan oleh soal dengan waktu rata-rata 0,53 detik dan memori rata-rata yang digunakan 4,7MB.

6.2 Saran

Pada Tugas Akhir ini tentunya terdapat kekurangan serta nilai-nilai yang dapat penulis ambil. Berikut adalah saran-saran yang dapat diambil melalui Tugas Akhir ini:

- Untuk ke depannya, materi yang ada pada Tugas Akhir ini dapat menjadi bahan riset untuk mencari optimasi yang lebih lanjut.
- Algoritma *Pollard Rho Brent* dapat menjadi bahan riset untuk mencari optimasi lebih lanjut. Optimasi yang dilakukan mungkin dapat menghilangkan operasi *random* untuk mencari variabel *c*.

[*Halaman ini sengaja dikosongkan*]

DAFTAR PUSTAKA

[1] "ADAHW-Ada And Homework," SPOJ, 10 02 2017. [Online]. Available: https://www.spoj.com/problems/ADAHW/. [Diakses 20 March 2019].

[2] M. Robinson, "Prime Factorization of Large Integer Values," December 2018. [Online]. Available: https://www.researchgate.net/publication/331938989_Prime_Factorisation_of_Large_Integer_Values/link/5c93c9fe299bf111693e25c5/download. [Diakses 11 December 2019].

[3] J. Maddock dan C. Kormanyos, "Boost.Multiprecision," 2016. [Online]. Available: https://www.boost.org/doc/libs/master/libs/multiprecision/doc/html/index.html. [Diakses 26 December 2019].

[4] K. H. Rosen, dalam *Discrete Mathematics and Its Applications 7th Edition*, New York, McGraw-Hill, 2012, pp. 237-306.

[5] L. Tóth, "Menon's Identity and Arithmetical Sums Representing Functions of Several Variables," arXiv preprint arXiv:1103.5861, 2011.

[6] C. Barnes, "Integer Factorization Algorithms," 2004. [Online]. Available: http://www.connelybarnes.com/documents/factoring.pdf . [Diakses 18 November 2019].

[7] C. Pomerance, S. S. Wagstaff, Jr dan J. L. Selfridge, "The pseudoprimes to 25·10^9," July 1980. [Online]. Available: https://math.dartmouth.edu/~carlp/PDF/paper25.pdf. [Diakses 10 December 2019].

[8] J. Vargas dan C. Shasank, "Proof of Euler's Phi Function Formula," 2013. [Online]. Available: https://scholar.rose-hulman.edu/rhumj/vol14/iss/2/6. [Diakses 2 Desember 2019].

[*Halaman ini sengaja dikosongkan*]

LAMPIRAN A: DATA UJI

Berikut merupakan data uji coba yang digunakan untuk uji coba kebenaran.

Input	stdin 11 2 5 6 7 8 10 50 100 1000 524288 945406969379503350
Output	stdout 0 3 2 5 8 6 70 260 5400 4718592 1381966975399059833610

[*Halaman ini sengaja dikosongkan*]

BIODATA PENULIS

Penulis bernama Gilbert Lijaya Therry, putra pertama dari 2 bersaudara yang lahir pada tanggal 30 Maret 1999 di Palu. Penulis melaksanakan pendidikan dasar di Sekolah Dasar Katolik 1 Santo Xaverius pada tahun 2004 hingga 2010, Sekolah Menengah Pertama Negeri 1 Palu pada tahun 2010 hingga 2013, dan Sekolah Menengah Atas Negeri Model Terpadu Madani Palu 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 rancang bangun aplikasi web, sistem temu kembali informasi, evolusi perangkat lunak, dan interaksi manusia dan komputer. Penulis juga pernah menjadi asisten dosen pada mata kuliah Struktur Data, Jaringan Komputer, Matematika Diskrit, Komputasi Numerik, dan Sistem Operasi.

Selain kesibukan akademik, penulis juga berperan aktif dalam beberapa kegiatan dan kepanitiaan. Beberapa diantaranya adalah staf dari divisi perlengkapan dan transportasi pada kegiatan *National Logic Competition* pada tahun 2017 dan menjadi *liaison officer* pada kegiatan *Revulutionary Entertaiments and Expo with Various Arts* tahun 2018.