TUGAS AKHIR - IF184802

DESAIN DAN ANALISIS PENYELESAIAN PERMASALAHAN *LINEAR RECURSIVE SEQUENCE* PADA STUDI KASUS : SPOJ - HAL9000

FANDY PUTRA MOHAMMAD
NRP 05111640000080

Dosen Pembimbing 1
Rully Soelaiman, S.Kom., M.Kom.

Dosen Pembimbing 2
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*]

TUGAS AKHIR - IF184802

DESAIN DAN ANALISIS PENYELESAIAN PERMASALAHAN *LINEAR RECURSIVE SEQUENCE* PADA STUDI KASUS : SPOJ - HAL9000

FANDY PUTRA MOHAMMAD
NRP 05111640000080

Dosen Pembimbing 1
Rully Soelaiman, S.Kom., M.Kom.

Dosen Pembimbing 2
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 OF SOLVING LINEAR RECURSIVE SEQUENCE PROBLEM ON STUDY CASE : SPOJ - HAL9000

FANDY PUTRA MOHAMMAD
NRP 05111640000080

Supervisor 1
Rully Soelaiman, S.Kom., M.Kom.

Supervisor 2
Yudhi Purwananto, S.Kom, M.Kom.

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

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

DESAIN DAN ANALISIS PENYELESAIAN PERMASALAHAN *LINEAR RECURSIVE SEQUENCE* PADA STUDI KASUS : SPOJ - HAL9000

TUGAS AKHIR

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

Oleh:
Fandy Putra Mohammad
NRP. 05111640000080

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

DESAIN DAN ANALISIS PENYELESAIAN PERMASALAHAN *LINEAR RECURSIVE SEQUENCE* PADA STUDI KASUS : SPOJ - HAL9000

Nama	:	Fandy Putra Mohammad
NRP	:	05111640000080
Departemen	:	Departemen Teknik Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas, ITS
Pembimbing I	:	Rully Soelaiman, S.Kom., M.Kom.
Pembimbing II	:	Yudhi Purwananto, S.Kom, M.Kom.

Abstrak

Linear recursive sequence merupakan salah satu bagian penting dalam bidang matematika. Berbagai contohnya antara lain: faktorial, bilangan fibonacci, bilangan lucas, dan lain-lain. Selain di bidang matematika, linear recursive sequence memiliki banyak manfaat di bidang lainnya seperti, ekonomi makro, bidang digital, dan lain-lain. Penelitian Tugas Akhir ini dapat direpresentasikan sebagai persoalan Linear Recursive Sequence dan terdapat pada situs Sphere Online Judge 100pct failure in 72 hours . Secara umum persoalan yang akan diselesaikan pada penelitian Tugas Akhir ini adalah bagaimana cara melakukan komputasi dari n suku pertama atau deret dari sebuah Linear Recursive Sequence dengan efisien.

Tugas akhir ini mengulas pendekatan Matrix Exponentiation dan penggunaan algoritma Gaussian Elimination, yang digunakan untuk menyelesaikan permasalahan mencari jumlah n suku pertama dari Linear Recursive Sequence. Lalu, dengan memanfaatkan salah satu generalisasi dari deret geometri yaitu deret Neumann, permasalahan ini dapat diselesaikan dengan mengkomputasi

invers sebuah matriks. Modifikasi Gaussian Elimination dapat menyelesaikan penghitungan invers dari matriks ini dengan cukup efisien, yaitu dengan kompleksitas $\mathcal{O}(n)$.

Dari hasil uji coba pada studi kasus yang diberikan, didapat hasil bahwa algoritma Gaussian Elimination memiliki efisiensi kinerja yang lebih baik dibandingkan dengan pendekatan naive. Rata-rata waktu hasil uji coba pada Linear Recursive Sequence menggunakan Gaussian Elimination adalah $0,012$ *detik. Sementara rata-rata memori hasil uji coba adalah* $4,86$ *MB.*

Kata Kunci: deret, gaussian elimination, linear recursive sequence, matrix exponentiation

[*Halaman ini sengaja dikosongkan*]

DESIGN AND ANALYSIS OF SOLVING LINEAR RECURSIVE SEQUENCE PROBLEM ON STUDY CASE : SPOJ - HAL9000

Name	:	Fandy Putra Mohammad
Student ID	:	05111640000080
Department	:	Department of Informatics Engineering, 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

Linear recursive sequence is one of the important parts of the mathematics field. Various examples include factorial, Fibonacci sequence, Lucas sequence, etc. Besides in the field of mathematics, linear recursive sequence has many benefits in other fields like in, macroeconomics, digital field, etc. This thesis can be represented as a Linear Recursive Sequence problem and found on SPOJ site as 100pct failure in 72 hours problem. By general the problem to be solved is about how to compute the first n term or sum of a sequence of a linear recursive sequence efficiently.

This thesis reviewed the Matrix Exponentiation approach and Gaussian Elimination algorithm that used to solve computation of the first n term of a Linear Recursive Sequence. Then, using one of the generalization of geometric series, Neumann-Series, this problem can be solved by computing the inverse of a matrix. Modified Gaussian Elimination can solve the computation of the inverse of this matrix efficiently, with complexity $\mathcal{O}(n)$.

From the result of testing from given case studies, it turns that the Gaussian Elimination algorithm has a better performance compared to the naive approach to solve computation of the first n *term of a Linear Recursive Sequence. The average time result of testing on Linear Recursive Sequence using the Gaussian Elimination algorithm is* 0.012 *second. While the average memory usage result is* 4.86 *MB.*

Keywords: gaussian elimination, linear recursive sequence, matrix exponentiation, series

KATA PENGANTAR

Puji syukur penulis ucapkan kepada Tuhan Yang Maha Esa. Atas rahmat dan karuniaNya, penulis dapat menyelesaikan tugas akhir dan laporan akhir dalam bentuk buku ini.

Pengerjaan buku ini penulis tujukan untuk mengeksplorasi lebih mendalam topik-topik yang tidak diwadahi oleh kampus, namun banyak menarik perhatian penulis. Selain itu besar harapan penulis bahwa pengerjaan Tugas Akhir (TA) sekaligus pengerjaan buku ini dapat menjadi batu loncatan penulis dalam menimba ilmu yang bermanfaat.

Penulis ingin menyampaikan rasa terima kasih kepada banyak pihak yang telah membimbing, menemani dan membantu penulis selama masa pengerjaan TA maupun masa studi.

1. Terimakasih yang sebesar-besarnya kepada Allah SWT, di mana penulis masih diberi berbagai nikmat dan kelancaran sehingga penulis dapat menempuh kuliah di Teknik Informatika ITS dengan baik hingga saat pengerjaan Tugas Akhir ini.
2. Ayah, Ibu, dan keluarga penulis yang selalu memberikan dukungan, perhatian, kasih sayang serta do'a dan restu bagi penulis yang menjadi semangat selama perkuliahan maupun pengerjaan Tugas Akhir.
3. Bapak Rully Soelaiman S.Kom.,M.Kom., selaku pembimbing penulis. Ucapan terima kasih penulis sampaikan atas segala kebaikan dalam bentuk perhatian, didikan, ilmu, dan nasihat yang telah diberikan oleh beliau sejak awal perkuliahan hingga saat pengerjaan Tugas Akhir.
4. Bapak Yudhi Purnawanto S.Kom.,M.Kom., selaku pembimbing penulis. Ucapan terima kasih juga penulis sampaikan atas bantuan dan masukannya. Berkat beliau buku TA ini dapat diselesaikan dengan baik.

5. Yoshima Syach Putri, Ferdinand Jason, Ivanda Zevi Amalia, dan Yolanda Hertita yang telah membantu dan memberikan saran terkait pengerjaan dan penyusunan buku tugas akhir.
6. Tamtam, Mail, Deka, Alam dan Vincent yang telah menemani dan membantu penulis selama menempuh masa studi di Departemen Informatika, Institut Teknologi Sepuluh Nopember
7. Teman-teman *administrator* Laboratorium Algoritma dan Pemrograman, Departemen Informatika, Institut Teknologi Sepuluh Nopember, yang telah menjadi keluarga dan telah membantu penulis serta menjadi penyemangat selama menjalani masa studi di kampus.
8. Teman-teman angkatan 2016 Departemen Teknik Informatika ITS, khususnya teman-teman bidang minat Algoritma dan Pemrograman, yang senantiasa membantu, menemani, memberi semangat, dan meninggalkan kenangan selama kurang lebih 4 tahun masa studi penulis.
9. Seluruh pihak yang tidak bisa penulis sebutkan satu-persatu yang telah memberikan dukungan selama penulis menyelesaikan TA ini.

Penulis menyadari bahwa buku ini jauh dari kata sempurna. Maka dari itu, penulis memohon maaf apabila terdapat salah kata maupun makna pada buku ini. Akhir kata, penulis berharap dapat memberikan kontribusi dan manfaat melalui buku ini.

Surabaya, Juni 2020

Fandy Putra Mohammad

DAFTAR ISI

KATA PENGANTAR . . . . . **xiii**
DAFTAR ISI . . . . . **xv**
DAFTAR GAMBAR . . . . . **xix**
DAFTAR TABEL . . . . . **xxi**
DAFTAR PSEUDOCODE . . . . . **xxiii**
DAFTAR KODE SUMBER . . . . . **xxv**
DAFTAR NOTASI . . . . . **xxvii**
BAB I PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 3
1.4 Tujuan 3
1.5 Manfaat 4
1.6 Metodologi 4
1.6.1 Penyusunan proposal 4
1.6.2 Studi literatur 4
1.6.3 Implementasi algoritma 5
1.6.4 Pengujian dan evaluasi 5
1.6.5 Penyusunan buku 5
1.7 Sistematika Penulisan 5
BAB II DASAR TEORI **7**
2.1 Deskripsi Permasalahan 7
2.2 Matrix Exponentiation 8
2.3 Neumann-Series 10
2.4 Gaussian Elimination 11
2.5 Modular Arithmetic 14

BAB III DESAIN DAN ANALISIS **17**
3.1 Analisis Strategi Penyelesaian Permasalahan . . . 17
3.2 Desain Umum Sistem 18
3.2.1 Desain Fungsi Main 19
3.2.2 Desain Class Linear Recursive Sequence . 20
3.2.3 Desain Fungsi Mod 23
3.2.4 Desain Fungsi Find MMI 24
3.2.5 Desain Fungsi Fast Pow 25
3.2.6 Desain Fungsi Solve 26
BAB IV IMPLEMENTASI **29**
4.1 Lingkungan Implementasi 29
4.2 Implementasi Program Utama 29
4.2.1 Header yang Diperlukan 30
4.2.2 Preprocessor 30
4.2.3 Variabel Global 31
4.2.4 Implementasi Fungsi Main 31
4.2.5 Implementasi Struct Linear Recursive Sequence 32
4.2.6 Implementasi Fungsi Mod 33
4.2.7 Implementasi Fungsi FindMMI 34
4.2.8 Implementasi Fungsi FastPow 34
4.2.9 Implementasi Fungsi Solve 35
BAB V UJI COBA DAN ANALISIS **37**
5.1 Lingkungan Uji Coba 37
5.2 Skenario Uji Coba 38
5.3 Uji Coba Kebenaran 38
5.4 Uji Coba Kinerja Lokal 39
5.5 Uji Coba Kinerja Luar 42
BAB VI KESIMPULAN DAN SARAN **45**

6.1 Kesimpulan 45
6.2 Saran 45
DAFTAR PUSTAKA **47**
LAMPIRAN A: DATA UJI **49**
LAMPIRAN B: Hasil Uji Coba pada Situs SPOJ Sebanyak 10 Kali **59**
LAMPIRAN C: Kode Sumber Uji Coba pada Situs SPOJ **61**
BIODATA PENULIS **63**

[*Halaman ini sengaja dikosongkan*]

DAFTAR GAMBAR

Gambar 5.1 Hasil Uji Coba Kebenaran Situs SPOJ dengan pendekatan *naive* 38

Gambar 5.2 Hasil Uji Coba Kebeneran Situs SPOJ dengan algoritma *Gaussian Elimination* 39

Gambar 5.3 Peringkat pada HAL9000 - *100pct failure in 72 hours* 39

Gambar 5.4 Rata-rata *Running Time* untuk Setiap Kasus Uji dengan nilai d tetap 40

Gambar 5.5 Rata-rata *Running Time* untuk Setiap Kasus Uji dengan nilai n tetap 41

Gambar 5.6 Grafik Waktu Hasil Pengumpulan Kode Berkas Sebanyak 10 Kali 42

Gambar 5.7 Grafik Memori Hasil Pengumpulan Kode Berkas Sebanyak 10 Kali 43

Gambar B.1 Hasil Pengumpulan Kode Program Utama . . 59

Gambar C.1 Kode Sumber Uji Coba pada Situs SPOJ . . . 61

[*Halaman ini sengaja dikosongkan*]

DAFTAR TABEL

Tabel 3.1 Nama dan Fungsi Variabel dalam *class* SEQ . . . 20
Tabel 3.2 Masukan, Proses, dan Keluaran dari Fungsi SETBETA *Class* SEQ 22
Tabel 3.3 Masukan, Proses, dan Keluaran dari Fungsi SETALPHA *Class* SEQ 23
Tabel 3.4 Masukan, Proses, dan Keluaran dari Fungsi SETDELTA *Class* SEQ 23
Tabel 3.5 Masukan, Proses, dan Keluaran dari Fungsi MOD 24
Tabel 3.6 Masukan, Proses, dan Keluaran dari Fungsi FINDMMI 25
Tabel 3.7 Masukan, Proses, dan Keluaran dari Fungsi FAST POW 26
Tabel 3.8 Masukan, Proses, dan Keluaran dari Fungsi SOLVE 26

Tabel A.1 Data Uji dengan Nilai $d = 2(1)$ 49
Tabel A.2 Data Uji dengan Nilai $d = 2(2)$ 50
Tabel A.3 Data Uji dengan Nilai $d = 2(3)$ 51
Tabel A.4 Data Uji dengan Nilai $d = 2(4)$ 52
Tabel A.5 Data Uji dengan Nilai $d = 2(5)$ 53
Tabel A.6 Data Uji dengan Nilai $n = 100(1)$ 54
Tabel A.7 Data Uji dengan Nilai $n = 100(2)$ 55
Tabel A.8 Data Uji dengan Nilai $n = 100(3)$ 56
Tabel A.9 Data Uji dengan Nilai $n = 100(4)$ 57
Tabel A.10 Data Uji dengan Nilai $n = 100(5)$ 58

[*Halaman ini sengaja dikosongkan*]

DAFTAR PSEUDOCODE

Pseudocode 2.1 Pendekatan *naive* 9
Pseudocode 2.2 Menghitung hasil modulo 15
Pseudocode 3.1 Fungsi MAIN 19
Pseudocode 3.2 *Class* SEQ 21
Pseudocode 3.3 Prosedur CONSTRUCTOR pada *class* SEQ 21
Pseudocode 3.4 Prosedur SETBETA pada *class* SEQ . . . 22
Pseudocode 3.5 Prosedur SETALPHA pada *class* SEQ . . 22
Pseudocode 3.6 Prosedur SETDELTA pada *class* SEQ . . 23
Pseudocode 3.7 Menghitung hasil modulo 24
Pseudocode 3.8 Mencari nilai MMI 24
Pseudocode 3.9 Melakukan operasi pemangkatan 25
Pseudocode 3.10 Fungsi Solve 27

[*Halaman ini sengaja dikosongkan*]

DAFTAR KODE SUMBER

Kode Sumber 4.1 *Header* yang diperlukan 30
Kode Sumber 4.2 *Preprocessor* yang diperlukan 30
Kode Sumber 4.3 Variabel global yang didefinisikan untuk Program 31
Kode Sumber 4.4 Fungsi main 32
Kode Sumber 4.5 Struct Seq 33
Kode Sumber 4.6 Fungsi Mod 34
Kode Sumber 4.7 Fungsi FindMMI 34
Kode Sumber 4.8 Fungsi FastPow 34
Kode Sumber 4.9 Fungsi Solve 35

[*Halaman ini sengaja dikosongkan*]

DAFTAR NOTASI

$\sum$ Notasi yang digunakan untuk menjumlahkan sejumlah bilangan terurut dengan aturan tertentu.

Δ Notasi yang digunakan untuk menghitung selisih antara dua nilai dengan aturan tertentu.

[*Halaman ini sengaja dikosongkan*]

BAB I
PENDAHULUAN

Pada bab ini, akan dijelaskan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi pengerjaan, dan sistematika penulisan buku Tugas Akhir.

1.1 Latar Belakang

Permasalahan HAL9000 - *100pct failure in 72 hours* merupakan permasalahan *linear recursive sequence*. *Linear recursive sequence* merupakan salah satu bagian matematika yang umum ditemui. *Linear recurrence relation* merupakan sebuah persamaan yang mendefinisikan sebuah *linear recursive sequence*, jika satu atau lebih suku sudah diberi suatu nilai; maka suku berikutnya dapat didefinisikan sesuai fungsi dari suku sebelumnya. Beberapa contoh dari sebuah *linear recursive sequence* seperti faktorial, deret *fibonacci*, deret *lucas*, dan lain-lain. Secara umum *linear recurrence* terbagi dua jenis, yaitu *homogeneous* dan *non-homogeneous*.

Meskipun *linear recursive sequence* merupakan salah satu bagian dari bidang matematika, hal ini juga memiliki banyak manfaat di berbagai bidang lainnya, seperti di bawah ini.

- Di bidang biologi, *linear recursive sequence (logistic map)* digunakan untuk memodelkan pertumbuhan dari suatu populasi[4],
- *Recurence relation*, terutama *linear recurrence relation*, digunakan di ekonomi makro untuk memodelkan berbagai sektor ekonomi[3],
- Di bidang digital *signal processing*, *reccurence relation* dapat digunakan untuk memodelkan *feedback* dari sebuah sistem, dimana keluaran di suatu waktu akan menjadi masukan di waktu yang akan datang[7],

- *Recurrence relation* juga merupakan dasar penting dalam analisis algoritma. Jika suatu algoritma didesain dalam pemecahan *subproblem* yang lebih kecil *(divide and conquer)*,

Dalam studi kasus SPOJ dengan kode HAL9000[6]. Permasalahan yang diberikan merupakan pencarian jumlah semua suku atau deret dari sebuah *linear recursive sequence* yang beberapa elemennya sudah diketahui. Pencarian deret ini dapat diselesaikan melalui berbagai jenis pendekatan. Pendekatan secara naif dapat menyebabkan penyelesaian dengan kompleksitas $O(n)$. Sehingga dibutuhkan pendekatan dan algoritma yang optimal agar dapat menyelesaikan permasalahan ini dengan efektif dan efisien.

Terdapat beberapa algoritma dan metode yang dapat digunakan untuk menyelesaikan permasalahan dalam tugas akhir ini. Dengan menggunakan metode *Matrix Exponentiation* yang sudah umum digunakan dalam *competitive programming* untuk menyelesaikan *linear recursive sequence* permasalahan ini dapat diubah ke bentuk persamaan matriks. Selanjutnya dengan menggunakan pembuktian dari deret-*Neumann*, bisa dihitung nilai deret matriks dengan sangat efisien. Lalu, dengan menggunakan modifikasi dari algoritma *Gaussian Elimination*, penghitungan nilai invers bisa dilakukan dengan waktu dan penggunaan memori yang sangat efisien. Di tahap implementasi, dengan menerapkan *modular arithmetic*, bisa didapatkan perhitungan nilai yang akurat walaupun nilai hasil dapat melewati batasan yang bisa disimpan dalam suatu variabel dalam bahasa pemrogaraman C++.

1.2 Rumusan Masalah

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

1. Bagaimana cara menentukan algoritma yang tepat untuk menyelesaikan permasalahan *linear recursive sequence* pada soal

HAL9000 - *100pct failure in 72 hours* di situs SPOJ ?

2. Bagaimana cara mengimplementasikan algoritma yang tepat dan optimal pada permasalahan SPOJ : HAL9000 - *100pct failure in 72 hours* agar didapatkan hasil yang akurat dan efisien?

1.3 Batasan Masalah

Permasalahan yang dibahas pada Tugas Akhir ini memiliki beberapa batasan, seperti di bawah ini.

1. Permasalahan ini merupakan permasalahan *linear recursive sequence* yang akan diselesaikan dengan algoritma *Gaussian Elimination*
2. Implementasi algoritma menggunakan bahasa pemrograman C++.

Adapun batasan dalam SPOJ untuk Tugas Akhir ini adalah sebagai berikut:

1. Batas nilai T di antara 1 hingga 100 di mana T merupakan banyaknya data uji.
2. Batas nilai d di antara 1 hingga 1000 merupakan *order* dari *linear recursive sequence*.
3. Batas nilai n di antara 1 hingga 10^9.
4. Batas nilai a_i dan b_i di antara 0 dan 10^6 dan nilai b_0 lebih besar daripada 0.
5. Batas nilai a_{n+i} di antara 0 hingga $10^9 + 7$.
6. Batas waktu yang diberikan 3 detik.
7. Batas memori yang diberikan 1536MB.
8. Batas memori sumber kode yang diberikan 1024B.

1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah sebagai berikut:

1. Melakukan desain dan analisis algoritma yang tepat untuk menyelesaikan permasalahan *linear recursive sequence* pada soal HAL9000 - *100pct failure in 72 hours* di situs SPOJ
2. Melakukan implementasi algoritma yang tepat dan optimal pada permasalahan SPOJ : HAL9000 - *100pct failure in 72 hours* agar didapatkan hasil yang akurat dan efisien

1.5 Manfaat

Tugas Akhir ini diharapkan dapat membantu untuk memberikan pemahaman penggunaan dan implementasi algoritma yang tepat dan optimal dalam penyelesaian permasalahan pencarian summation dari *linear recursive sequence* pada studi kasus *Sphere Online Judge* : HAL900 - *100pct failure in 72 hours* .

1.6 Metodologi

Metodologi pengerjaan yang digunakan pada Tugas Akhir ini memiliki beberapa tahapan. Tahapan-tahapan tersebut adalah sebagai berikut.

1.6.1 Penyusunan proposal

Pada tahapan ini penulis memberikan penjelasan mengenai apa yang penulis akan lakukan dan mengapa Tugas Akhir ini dilakukan. Penjelasan tersebut dituliskan dalam bentuk proposal Tugas Akhir.

1.6.2 Studi literatur

Pada tahapan ini penulis mengumpulkan referensi yang diperlukan guna mendukung pengerjaan Tugas Akhir. Referensi yang digunakan dapat berupa hasil penelitian yang sudah pernah dilakukan, buku, artikel internet, atau sumber lain yang bisa dipertanggungjawabkan.

1.6.3 Implementasi algoritma

Pada tahapan ini penulis mulai mengembangkan algoritma yang digunakan untuk menyelesaikan permasalahan pencarian deret dari sebuah *linear recursive sequence*.

1.6.4 Pengujian dan evaluasi

Pada tahapan ini penulis menguji kinerja algoritma yang digunakan. Hasil pengujian kemudian dievaluasi untuk kemudian dipertimbangkan apakah algoritma masih bisa ditingkatkan kinerja lagi atau tidak.

1.6.5 Penyusunan buku

Pada tahapan ini penulis menyusun hasil pengerjaan Tugas Akhir mengikuti format penulisan Tugas Akhir.

1.7 Sistematika Penulisan

Sistematika laporan Tugas Akhir yang akan digunakan adalah sebagai berikut :

1. BAB I : Pendahuluan
   Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.
2. BAB II : Dasar Teori
   Bab ini berisi kajian teori dari metode dan algoritma yang digunakan dalam penyusunan Tugas Akhir ini.
3. BAB III : Desain
   Bab ini menjelaskan desain algoritma yang akan dibangun berdasarkan dasar teori dan digunakan dalam penyelesaian.
4. BAB IV : Implementasi
   Bab ini membahas implementasi dari perancangan yang telah di-

buat pada bab sebelumnya. Terdapat juga penjelasan berupa kode yang digunakan untuk proses implementasi.

5. BAB V : Uji Coba dan Analisis
   Bab ini membahas tahapan uji coba, kemudian hasil uji coba dievaluasi dan dianalisis terhadap kinerja dari algoritma yang diimplementasikan.
6. BAB VI : Kesimpulan dan Saran
   Bab ini merupakan bab yang menjelaskan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses dan tertulis saat pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.

BAB II
DASAR TEORI

Pada bab ini, akan dijelaskan dasar teori yang digunakan sebagai landasaan pengerjaan Tugas Akhir ini. Pertama akan dijelaskan mengenai deskripsi permasalahan, dilanjutkan dengan penyelesaian secara naif. Kemudian, pembahasan mengenai analisis soal beserta dengan beberapa dasar teori yang dapat digunakan akan dijabarkan.

2.1 Deskripsi Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini adalah perhitungan nilai dari jumlah n suku atau deret dalam sebuah *linear recursive sequence* yang didefinisikan pada persamaan 2.1.

$$a_n = a_{n-1}b_{d-1} + a_{n-2}b_{d-2} + \cdots + a_{n-d}b_0. \tag{2.1}$$

Di mana d koefisien b_i merupakan konstanta dan $b_0 \neq 0$. Sehingga nilai dari jumlah n suku atau deret dari sebuah *linear recursive sequence* dapat didefinisikan seperti pada persamaan 2.2

$$S_n = \sum_{i=0}^{n} a_i \tag{2.2}$$

Pada studi kasus SPOJ : HAL9000 - *100pct failure in 72 hours* ada beberapa data berupa masukan yang diberikan, yaitu:

- T berupa jumlah data uji.
- n merupakan jumlah suku yang akan dicari.
- d merupakan nilai $order$ dari persamaan *linear recursive sequence* yang diberikan.

- $a_i[0...d]$ merupakan d nilai suku pertama dari *linear recursive sequence* yang diberikan.
- $b_i[0...d]$ merupakan d konstanta untuk persamaan dari *linear recursive sequence* yang diberikan.
- $a_{n+i}[0...d]$ merupakan d suku mulai a_n.

Selain itu hasil keluaran dioperasikan dengan operasi modulus terhadap bilangan 10^9+7 terlebih dahulu dikarenakan hasil yang didapatkan bisa bernilai sangat besar.

Berdasarkan persamaan 2.2 dan *pseudocode* 2.1, pendekatan ini kurang efisien untuk permasalahan pencarian jumlah n suku *linear recursive sequence* dikarenakan masih memiliki kompleksitas $\mathcal{O}(n)$. Hal ini belum termasuk kompleksitas dari perhitungan nilai untuk tiap a_i hingga n. Dikarenakan nilai batasan n pada deskripsi soal cukup besar yaitu 10^9, sehingga algoritma naif akan memberikan *verdict Time Limit Excedeed* karena melampaui batas waktu yang telah ditentukan.

2.2 Matrix Exponentiation

Matrix Exponentiation merupakan salah satu teknik yang paling umum digunakan dalam bidang *competitive programming*[5]. Teknik ini biasa digunakan untuk mencari nilai suku ke n dari deret *Fibonacci* yang merupakan salah satu bentuk dari persamaan *linear recursive sequence*. Teknik ini mengubah bentuk persamaan *linear recursive sequence* ke dalam bentuk matriks. Menggunakan persamaan 2.2, pencarian nilai a_{n+1} dapat direpresentasikan dalam bentuk matriks seperti berikut:

$$\begin{bmatrix} a_{n+1} & a_{n+2} & \cdots & a_{n+d} \end{bmatrix} = \begin{bmatrix} a_n & a_{n+1} & \cdots & a_{n+d-1} \end{bmatrix} \times Z \tag{2.3}$$

Pseudocode 2.1 Pendekatan *naive*

Input: $n,\ d,\ ai[\,],\ bi[\,]$
Output: Sn

```
Sn ← 0
for i ← 0, d do
    Sn ← Sn + ai[i]
end for
for i ← d, N do
    f ← (f * i) mod p
    ai[i] ← 0
    for j ← 0, d do
        ai[i] ← ai[i] + ai[i − d + j] * bi[j]
    end for
    Sn ← Sn + ai[i]
end for
return Sn
```

Dari persamaan 2.3 terbentuk dua matriks berukuran $1 \times D$. Agar aturan perkalian matriks terpenuhi, maka matriks Z harus berukuran $D \times D$. Berdasarkan persamaan 2.1 matriks Z dapat dibentuk seperti berikut:

$$Z = \begin{bmatrix} 0 & 0 & \cdots & 0 & b_0 \\ 1 & 0 & \cdots & 0 & b_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & b_{d-1} \end{bmatrix} \tag{2.4}$$

Dari persamaan 2.3 didapat persamaan 2.5 untuk $n \geq d$

$$\begin{bmatrix} a_n & a_{n+1} & \cdots & a_{n+d-1} \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & \cdots & a_{d-1} \end{bmatrix} \times Z^n \tag{2.5}$$

Sehingga persamaan untuk mencari jumlah n suku pertama

atau deret dari sebuah *linear recursive sequence* dapat dinyatakan seperti dibawah ini:

$$S_n = A_0 + A_1 + \cdots + A_n \tag{2.6}$$
$$S_n = A \times Z^0 + A \times Z^1 + \cdots + A \times Z^n \tag{2.7}$$
$$S_n = A \times (Z^0 + Z^1 + \cdots + Z^n) \tag{2.8}$$

Di mana A_i merupakan bentuk matriks dari suku a_i, seperti pada persamaan 2.5.

2.3 Neumann-Series

Neumann-Series atau deret *Neumann* merupakan salah satu generalisasi dari deret geometri. Deret ini didefinisikan pada persamaan 2.9. Dengan menggunakan pembuktian deret-Neumann bisa didapat nilai deret matriks dengan lebih efisien dibanding dengan penghitungan dengan pendekatan *naive*

$$\sum_{k=0}^{\infty} T^k \tag{2.9}$$

Di mana

$$T^k = T^{k-1} \times T \tag{2.10}$$

Jika $(I - T)$ dapat *invertible*, kita dapat mencari nilai S_n untuk $S_n = \sum_{k=0}^{n} T^k$.

$$(I - T)\sum_{k=0}^{n} T^k = \sum_{k=0}^{n} T^k - \sum_{k=0}^{n} T^{k+1} \tag{2.11}$$

Dikarenakan $(I - T)$ *invertible*, rumus S_n bisa didefinisikan pada persamaan 2.12.

$$S_n = (I - T)^{-1}(I - T^{n+1}) \tag{2.12}$$

$$S_n = (T^{n+1} - I)(T - I)^{-1} \tag{2.13}$$

2.4 Gaussian Elimination

Gaussian elimination atau *row reduction* adalah salah satu algoritma yang umum digunakan untuk menyelesaikan sebuah sistem persamaan linear[2]. Selain itu algoritma ini juga bisa digunakan untuk menyelesaikan berbagai persoalan matriks seperti perhitungan determinan dan juga pencarian invers. Dengan menggunakan algoritma *Gaussian Elimination* yang sudah dimodifikasi, bisa dihitung hasil invers sebuah matriks dengan efisien. Untuk melakukan perhitungan tersebut, terdapat tiga jenis operasi yang bisa digunakan, yaitu:

- Melakukan pertukaran antara dua baris.
- Mengalikan sebuah baris dengan bilangan bukan 0.
- Menambahkan hasil kali sebuah baris ke baris lainnya.

Berdasarkan persamaan 2.13, algoritma *Gaussian Eliminiation* dapat digunakan untuk menghitung nilai invers dari $(T-I)^{-1}$. Berdasarkan persamaan 2.4, jika matriks T merupakan matriks Z, matriks tersebut dapat digambarkan seperti berikut

$$\begin{bmatrix} -1 & 0 & 0 & \cdots & 0 & b_0 \\ 1 & -1 & 0 & \cdots & 0 & b_1 \\ 0 & 1 & -1 & \cdots & 0 & b_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & b_{d-2} \\ 0 & 0 & 0 & \cdots & 1 & b_{d-1} - 1 \end{bmatrix} \tag{2.14}$$

Pencarian nilai jumlah n suku sebuah *linear recursive sequence* tidak membutuhkan hasil invers secara keseluruhan. Nilai invers yang diperlukan hanya pada kolom pertama sehingga proses

pencarian bisa lebih efisien. Proses ini ditunjukkan pada persamaan 2.15

$$\begin{bmatrix} -1 & 0 & 0 & \cdots & 0 & b_0 \\ 1 & -1 & 0 & \cdots & 0 & b_1 \\ 0 & 1 & -1 & \cdots & 0 & b_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & b_{d-2} \\ 0 & 0 & 0 & \cdots & 1 & b_{d-1}-1 \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{d-2} \\ x_{d-1} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \quad (2.15)$$

x_i untuk $i = 0...d-1$ merupakan hasil invers yang dibutuhkan. Hasil invers dapat dihitung dengan mengaplikasikan operasi yang terdapat pada algoritma *Gaussian Elimination*. Dengan mengaplikasikan operasi $R_{i+1} \leftarrow R_{i+1} + R_i$ untuik $i = 0, \cdots, d-2$ didapat persamaan 2.16

$$\begin{bmatrix} -1 & 0 & 0 & \cdots & 0 & \beta_0 \\ 0 & -1 & 0 & \cdots & 0 & \beta_1 \\ 0 & 0 & -1 & \cdots & 0 & \beta_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & \beta_{d-2} \\ 0 & 0 & 0 & \cdots & 0 & \beta_{d-1}-1 \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{d-2} \\ x_{d-1} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix} \quad (2.16)$$

di mana nilai β_i bisa dihitung dengan persamaan berikut

$$\beta_k = \sum_{i=0}^{k} b_i \quad (2.17)$$

Nilai β dapat dihitung secara linier sehingga hanya membutuhkan n operasi. Setelah nilai β dihitung, dengan melakukan *back subtitution* didapat persemaan 2.18.

$$x_{d-1} = \frac{1}{\beta_{d-1} - 1} \quad (2.18)$$

Selanjutnya nilai x_i untuk $i = 0, \cdots, d-2$ bisa dihitung menggunakan persamaan 2.19

$$x_i = \beta_i \times x_{d-1} - 1 \tag{2.19}$$

Dengan demikian, kolom pertama hasil invers dari matriks pada persamaan 2.14 adalah sebagai berikut

$$\begin{bmatrix} \beta_0 x_{d-1} - 1 \\ \beta_1 x_{d-1} - 1 \\ \beta_2 x_{d-1} - 1 \\ \vdots \\ \beta_{d-2} x_{d-1} - 1 \\ x_{d-1} \end{bmatrix} \tag{2.20}$$

2.5 Modular Arithmetic

Modular arithmetic merupakan sebuah sistem operasi aritmatika pada bilangan bulat yang memperhitungkan sisa dari operasi pembagian. Hal ini cukup penting dikarenakan pada proses komputasi yang mengukur efisiensi sebuah algoritma, batas nilai yang dibutuhkan cukup besar. Operasi modulo dibutuhkan agar didapat hasil yang lebih akurat dan dapat disimpan pada suatu tipe data dalam bahasa pemrograman. Pada bahasa pemrograman C/C++ tipe data yang dapat menyimpan bilangan bulat terbesar adalah *unsigned long long int* yang dapat menyimpan nilai hingga $2^64 - 1$, namun di beberapa studi kasus keluaran yang dihasilkan dapat jauh lebih besar sehingga dapat menyebabkan *integers overflow*. Dikarenakan alasan tersebut, maka hasil komputasi yang dibutuhkan harus dimodulo dengan sebuah bilangan terlebih dahulu. Bilangan $10^9 - 7$ merupakan salah satu bilangan yang dapat digunakan karena merupakan bilangan prima dan dapat disimpan pada tipe data *integer*. Ada beberapa sifat distributif dari operasi modulo seperti berikut:

- $(a + b) \bmod c = ((a \bmod c) + (b \bmod c)) \bmod c$

- $(a \times b) \bmod c = ((a \bmod c) \times (b \bmod c)) \bmod c$
- $(a - b) \bmod c = ((a \bmod c) - (b \bmod c)) \bmod c$
- $(a \,/\, b) \bmod c \neq ((a \bmod c) \,/\, (b \bmod c)) \bmod c$

Operasi modulo hendaknya dilakukan pada setiap operasi aritmatika yang dilakukan agar *overflow* tidak terjadi. Pada beberapa bahasa pemrograman seperti C/C++ operasi modulo pada bilangan negatif menghasilkan nilai yang tidak akurat sehingga diperlukan operasi tambahan. Hal ini dapat dilihat pada *pseudocode* 2.2

Pseudocode 2.2 Menghitung hasil modulo

Input: $a,\ m$
Output: a
1: $a \leftarrow (a\%m + m)\%m$

Pada operasi pembagian, sifat distributif tidak dapat diterapkan. Untuk melakukan operasi pembagian pada *modular arithmetic* diperlukan pehamaman mengenai *modulo multiplicative inverse (MMI)*[1]. Operasi pembagian dapat diselesaikan dengan mencari invers dari bilangan pembagi. Jika bilangan pembagi merupakan bilangan prima, *Fermats's little theorem* dapat digunakan untuk mencari nilai inversnya.

[*Halaman ini sengaja dikosongkan*]

BAB III
DESAIN DAN ANALISIS

Pada bab ini akan dijelaskan desain dan analisis algoritma yang akan digunakan untuk menyelesaikan permasalahan.

3.1 Analisis Strategi Penyelesaian Permasalahan

Pada subbab ini akan dipaparkan mengenai strategi penyelesaian permasalahan *linear recursive sequence* pada studi kasus SPOJ dengan kode HAL9000 dengan judul soal: *100pct failure in 72 hours* . Berdasarkan teori-teori pada bab 2 didapat persamaan 3.1 untuk menyelesaikan permasalahan ini.

$$S_n = A \times (Z^n - I) \times (Z - I)^{-1} + A_n \tag{3.1}$$

$$S_n = A_n - A \times (Z - I)^{-1} + A_n \tag{3.2}$$

$$S_n = \begin{bmatrix} a_n - a_0 & \cdots & a_{n+d-1} - a_{d-1} \end{bmatrix} \begin{bmatrix} \beta_0 x_{d-1} - 1 \\ \beta_1 x_{d-1} - 1 \\ \beta_2 x_{d-1} - 1 \\ \vdots \\ \beta_{d-2} x_{d-1} - 1 \\ x_{d-1} \end{bmatrix} + a_n \tag{3.3}$$

Selanjutnya nilai $a_{n+i} - a_i$ untuk $i = 0, \cdots, d$ akan disimbolkan dengan Δ_i. Sehingga bisa didapat persamaan akhir untuk mencari nilai jumlah n suku dari *linear recursive sequence* seperti berikut

$$S_n = \Delta_0 \beta_0 x_{d-1} - \Delta_0 + \cdots + \Delta_{d-2} \beta_{d-2} x_{d-1} - \Delta_{d-2} + \Delta_{d-1} x_{d-1} + a_n \tag{3.4}$$

Berdasarkan persamaan 2.18, nilai x_{d-1} dapat bernilai pecahan, sehingga persamaan 3.4 dapat bernilai tidak akurat dalam komputasi dengan pemrograman komputer. Untuk meningkatkan akurasi perhitungan dan meminimalisir proses perhitungan, persamaan tersebut bisa diubah menjadi persamaan 3.5.

$$S_n = (\sum_{i=0}^{d-2} \Delta_i \beta_i + \Delta_{d-1})(x_{d-1}) - \sum_{i=0}^{d-2} \Delta_i + a_n \tag{3.5}$$

Persamaan 3.5 dapat digunakan untuk mencari jumlah n suku pertama atau deret dari sebuah *linear recursive sequence* jika $(Z - I)^{-1}$ *invertible*. Berdasarkan pengamatan penulis, nilai $(Z - I)^{-1}$ tidak *invertible* jika nilai x_{d-1} tidak dapat didefinisikan. Berdasarkan persamaan 2.18, nilai x_{d-1} tidak dapat didefinisikan jika nilai pembagi bernilai 0 atau nilai β_{d-1} bernilai 1. Pada kasus tersebut untuk mencari jumlah n suku pertama atau deret dari sebuah *linear recursive sequence* dapat digunakan persamaan 3.6

$$S_n = \left\lfloor \frac{n}{d} \right\rfloor \times \alpha_{d-1} + \alpha_{n\%d} \tag{3.6}$$

di mana nilai α_k dapat dihitung dengan persamaan 3.7.

$$\alpha_k = \sum_{i=0}^{k} a_i \text{ untuk } i = 0, \cdots, d-1 \tag{3.7}$$

3.2 Desain Umum Sistem

Pada subbab ini akan dijelaskan mengenai gambaran secara umum dari algoritma yang dirancang. Sistem akan diawali dengan menerima masukan berupa nilai T yang menyatakan banyaknya kasus uji. T baris selanjutnya berisi nilai dari n dan d. Nilai masukan ini mengikuti batasan pada subbab 1.3. Baris berikutnya berisikan d bilangan bulat a_i. Baris berikutnya berisikan d bilangan bulat b_i.

Baris berikutnya berisikan d bilangan bulat a_{n+i}. Setelah menerima masukan, maka masukan tersebut akan diolah untuk menghitung nilai dari S_n dalam modulo $10^9 + 7$, dikarenakan keluaran yang dihasilkan dapat cukup besar.

3.2.1 Desain Fungsi Main

Pseudocode fungsi MAIN ditunjukkan pada *pseudocode* 3.1. Pada *pseudocode* 3.1, fungsi MAIN merupakan bagian fungsi utama yang menerima masukan, memproses dan menampilkan masukan tersebut. Fungsi INPUT merupakan fungsi untuk menerima masukan, dan fungsi PRINT merupakan fungsi untuk menampilkan hasil.

Pseudocode 3.1 Fungsi MAIN

```
T ← INPUT()
for i ← 1, T do
    n, d ← INPUT()
    for j ← 0 to d − 1 do
        ai[j] ← INPUT()
    end for
    for j ← 0 to d − 1 do
        bi[j] ← INPUT()
    end for
    for j ← 0 to d − 1 do
        an[j] ← INPUT()
    end for
    sequence ← SEQ(n, d, ai[ ], bi[ ], an[ ])
    ans ← SOLVE(sequence)
    PRINT(ANS)
end for
```

3.2.2 Desain Class Linear Recursive Sequence

Class SEQ adalah *class* untuk menyimpan data-data dari sebuah *linear recursive sequence*. Pseudocode 3.2 merupakan *pseudocode* dari *class* SEQ.

Fungsi-fungsi yang terdapat dalam *class* ini adalah SETDELTA, dan SETBETA. Tabel 3.1 menjelaskan variabel dan kegunaannya dalam *class* SEQ. *Pseudocode* 3.2 memberikan gambaran mengenai desain *class* SEQ.

Tabel 3.1 Nama dan Fungsi Variabel dalam *class* SEQ

Nama Variabel	Fungsi Variabel
n	Menyimpan nilai n untuk nilai S_n yang akan dihitung
d	Menyimpan nilai d yaitu orde dari *linear recursive sequence*
ai	Menyimpan nilai bilangan a_i untuk $i = 0, \cdots, d-1$
bi	Menyimpan nilai bilangan b_i untuk $i = 0, \cdots, d-1$, yaitu konstanta dari *linear recursive sequence*
an	Menyimpan nilai bilangan a_{n+i} untuk $i = 0, \cdots, d-1$
$delta$	Menyimpan nilai bilangan Δ_i, dimana $\Delta_i = a_{n+i} - a_i$
$beta$	Menyimpan nilai bilangan β_i pada persamaan 2.17
$alpha$	Menyimpan nilai bilangan α_i pada persamaan 3.7

Pseudocode 3.2 *Class* SEQ

```
n, d ← Integer
ai, bi, an, delta, beta ← Array of Integer
constructor SEQ(_n, _d, _ai[ ], _bi[ ], _an[ ])
procedure SETBETA()
procedure SETDELTA()
procedure SETALPHA()
```

Fungsi *constructor* dari *class* ini hanya terdiri dari satu jenis. Fungsi *constructor* adalah fungsi dengan parameter $_n, _d, _ai[\,], _bi[\,], _an[\,]$, menyatakan nilai $n, d, ai[\,], bi[\,], an[\,]$, secara berurutan. *Pseudocode* fungsi ini dapat dilihat pada *pseudocode* 3.3.

Pseudocode 3.3 Prosedur CONSTRUCTOR pada *class* SEQ

```
Input: _n, _d, _ai[ ], _bi[ ], _an[ ]
n ← _n
d ← _d
ai ← _ai[ ]
bi ← _bi[ ]
an ← _an[ ]
SETBETA()
SETDELTA()
SETALPHA()
```

Fungsi SETBETA, fungsi ini bertanggung jawab untuk melakukan komputasi untuk nilai dari β_i pada persamaan 2.17 dengan menggunakan nilai dari variabel bi. Masukan, proses dan keluaran dari fungsi ini tercantum pada tabel 3.2. *Pseudocode* fungsi ini dapat dilihat pada *pseudocode* 3.4.

Pseudocode 3.4 Prosedur SETBETA pada *class* SEQ

```
1: beta[0] ← bi[0]
2: for i ← 1 to d − 1 do
3:     beta[i] ← beta[i − 1] + bi[i]
4: end for
```

Tabel 3.2 Masukan, Proses, dan Keluaran dari Fungsi SETBETA *Class* SEQ

Masukan	Proses	Keluaran
-	Melakukan komputasi untuk nilai β	-

Fungsi SETALPHA, fungsi ini bertanggung jawab untuk melakukan komputasi untuk nilai dari α_i pada persamaan 3.7 dengan menggunakan nilai dari variabel ai. Masukan, proses dan keluaran dari fungsi ini tercantum pada tabel 3.3. *Pseudocode* fungsi ini dapat dilihat pada *pseudocode* 3.5.

Pseudocode 3.5 Prosedur SETALPHA pada *class* SEQ

```
1: alpha[0] ← ai[0]
2: for i ← 1 to d − 1 do
3:     alpha[i] ← alpha[i − 1] + ai[i]
4: end for
```

Tabel 3.3 Masukan, Proses, dan Keluaran dari Fungsi SETALPHA *Class* SEQ

Masukan	Proses	Keluaran
-	Melakukan komputasi untuk nilai α	-

Fungsi SETDELTA, fungsi ini bertanggung jawab untuk melakukan komputasi untuk nilai dari $\Delta_i = a_{n+i} - a_i$ dengan menggunakan nilai dari variabel an dan ai. Masukan, proses dan keluaran dari fungsi ini tercantum pada tabel 3.4. *Pseudocode* fungsi ini dapat dilihat pada *pseudocode* 3.6.

Pseudocode 3.6 Prosedur SETDELTA pada *class* SEQ

```
for i ← 0 to d − 1 do
    delta[i] ← an[i] − ai[i]
end for
```

Tabel 3.4 Masukan, Proses, dan Keluaran dari Fungsi SETDELTA *Class* SEQ

Masukan	Proses	Keluaran
-	Melakukan komputasi untuk nilai Δ	-

3.2.3 Desain Fungsi Mod

Fungsi MOD bertanggung jawab untuk melakukan operasi modular suatu bilangan terhadap bilangan prima $10^9 + 7$, hal ini perlu dilakukan untuk mencegah hasil yang tidak akurat pada operasi modular pada bilangan negatif. *Pseudocode* dari fungsi MOD dapat dilihat pada *pseudocode* 3.7. Masukan, proses, dan keluaran dari fungsi ini tercantum pada tabel 3.5.

Pseudocode 3.7 Menghitung hasil modulo

Input: a
Output: a
1: $m \leftarrow 10^9 + 7$
2: $a \leftarrow (a\%m + m)\%m$
3: **return** a

Tabel 3.5 Masukan, Proses, dan Keluaran dari Fungsi MOD

Masukan	Proses	Keluaran
Bilangan a yang akan dioperasikan	Melakukan operasi modulo terhadap bilangan $10^9 + 7$	Nilai hasil modulo terhadap bilangan $10^9 + 7$

3.2.4 Desain Fungsi Find MMI

Fungsi FINDMMI bertanggung jawab untuk mencari nilai MMI dari variabel y jika $M = 10^9 + 7$. *Pseudocode* dari fungsi FINDMMI dapat dilihat pada *pseudocode* 3.8. Masukan, proses, dan keluaran dari fungsi ini tercantum pada tabel 3.6.

Pseudocode 3.8 Mencari nilai MMI

Input: y
1: $M \leftarrow 10^9 + 7$
2: **return** FASTPOW$(y, M - 2)$

Tabel 3.6 Masukan, Proses, dan Keluaran dari Fungsi FINDMMI

Masukan	Proses	Keluaran
Bilangan y yang akan dicari nilai MMInya	Mencari nilai MMI pada bilangan y jika $M = 10^9 + 7$	Nilai MMI dari y

3.2.5 Desain Fungsi Fast Pow

Fungsi FASTPOW bertanggung jawab untuk mencari hasil operasi pemangkatan suatu bilangan secara efisien. Fungsi ini bersifat rekursif. *Pseudocode* dari fungsi FASTPOW dapat dilihat pada *pseudocode* 3.9. Masukan, proses, dan keluaran dari fungsi ini tercantum pada tabel 3.7.

Pseudocode 3.9 Melakukan operasi pemangkatan

Input: x, y
1: $M \leftarrow 10^9 + 7$
2: **if** $y == 0$ **then**
3: **return** 1
4: **end if**
5: **if** $y == 1$ **then**
6: **return** x
7: **end if**
8: $halfn \leftarrow$ FASTPOW$(x, y/2)$
9: **if** $n\%2 == 0$ **then**
10: **return** $(halfn * halfn)\%M$
11: **else**
12: **return** $(((halfn * halfn)\%M) * x)\%M$
13: **end if**

Tabel 3.7 Masukan, Proses, dan Keluaran dari Fungsi FAST POW

Masukan	Proses	Keluaran
Menghitung nilai x^y dalam modulo M	Melakukan operasi modulo terhadap bilangan $10^9 + 7$	Nilai hasil modulo terhadap bilangan $10^9 + 7$

3.2.6 Desain Fungsi Solve

Pada subbab ini akan dijelaskan mengenai fungsi SOLVE untuk menyelesaikan permasalahan. Masukkan, proses, dan keluaran dari fungsi ini tercantum pada tabel 3.8. *Pseudocode* dari fungsi ini dapat dilihat pada *pseudocode* 3.10.

Tabel 3.8 Masukan, Proses, dan Keluaran dari Fungsi SOLVE

Masukan	Proses	Keluaran
Sebuah *linear recursive sequence*	Melakukan proses pencarian nilai n suku pertama atau deret dengan menggunakan persamaan 3.5 dan 3.6	Nilai n suku pertama atau deret dari *linear recursive sequence*

Pseudocode 3.10 Fungsi Solve

```
Input: sequence, n, d
sn ← 0
xn ← sequence.beta[d − 1] − 1
if xn == 0 then
    sn ← MOD(FLOOR(n/d) ∗ sequence.alpha[d − 1])
    sn ← MOD(sn + alpha[n%d)
else
    for i ← 0 to d − 2 do
        temp ← MOD(sequence.delta[i] ∗ sequence.beta[i])
        sn ← MOD(sn + temp)
    end for
    sn ← MOD(sn + sequence.delta[d − 1])
    xn ← FINDMMI(xn)
    sn ← MOD(sn ∗ xn)
    temp ← 0
    for i ← 0 to d − 2 do
        temp ← MOD(temp + sequence.delta[i])
    end for
    sn ← MOD(sn − temp)
    sn ← MOD(sn + sequence.ai[n])
end if
return sn
```

[*Halaman ini sengaja dikosongkan*]

BAB IV
IMPLEMENTASI

Pada bab ini akan dijelaskan mengenai implementasi dari desain algoritma penyelesaian permasalahan pencarian jumlah n suku pertama atau deret dari *linear recursive sequence* pada soal HAL9000 - *100pct failure in 72 hours* di situs SPOJ. Pertama akan dijelaskan implementasi secara umum, dilanjutkan dengan implementasi dari kedua metode yang digunakan.

4.1 Lingkungan Implementasi

Lingkungan implementasi dan pengembangan yang dilakukan adalah sebagai berikut.

1. Perangkat Keras
   (a) Processor Intel® Core™ i7-6700HQ CPU @ 2.60GHz (8 CPUs), 2.6GHz
   (b) *Random Access Memory* 8192MB
2. Perangkat Lunak
   (a) Sistem Operasi Windows 10 Education 64-bit
   (b) Visual Studio Code
   (c) Bahasa Pemrograman C++
   (d) *Compiler* GCC 7.4.0 (Ubuntu 7.4.0-1ubuntu1 18.04.1) untuk Windows Subsystem Linux

4.2 Implementasi Program Utama

Subbab ini menjelaskan implementasi proses algoritma secara keseluruhan berdasarkan desain yang telah dijelaskan pada Bab 3. Program ini merupakan program yang digunakan untuk menyelesaikan permasalahan *Linear Recursive Sequence* pada studi kasus SPOJ : HAL9000 - *100pct failure in 72 hours* .

Dikarenakan batas sumber kode yang ketat pada soal di situs SPOJ, dilakukan modifikasi pada kode program agar dapat dilakukan uji coba. Hasil modifikasi dapat dilihat pada Lampiran C.

4.2.1 Header yang Diperlukan

Implementasi algoritma ini membutuhkan satu buah *header* yaitu `cstdio` seperti yang terlihat pada kode sumber 4.1.

```
1 #pragma GCC optimize ("O3")
2 #pragma GCC target ("avx")
3
4 #include <cstdio>
```

Kode Sumber 4.1 *Header* yang diperlukan

Selain header, terdapat juga preprocessor *pragma*, digunakan untuk mengganti flag *compiler* yang digunakan pada daring SPOJ. *Header* `iostream` berisi fungsi standar input output operasi yang digunakan oleh bahasa C.

4.2.2 Preprocessor

Pre-processor seperti `define` digunakan untuk membuat alias dari tipe data atau fungsi sesungguhnya. Terdapat dua alias yang digunakan yaitu `long long` sebagai `ll` dan `for(int i = x; i < n; i++)` sebagai `FOR(x,n)`. Pre-processor dapat dilihat pada kode sumber 4.2.

```
1 using namespace std;
2
3 #define ll long long
4 #define FOR(x,n) for(int i = x; i < n; i++)
```

Kode Sumber 4.2 *Preprocessor* yang diperlukan

4.2.3 Variabel Global

Variabel global digunakan untuk memudahkan dalam mengakses data yang digunakan lintas fungsi/*struct*. Kode sumber implementasi variabel global dapat dilihat pada kode sumber 4.3.

```
1 const int M = 1000000007;
```

Kode Sumber 4.3 Variabel global yang didefinisikan untuk Program

4.2.4 Implementasi Fungsi Main

Fungsi main merupakan implementasi algoritma yang dirancang pada *pseudocode* 3.1. Implementasi fungsi main dapat dilihat pada kode sumber 4.4.

```
1 int main() {
2   int T;
3   scanf("%d",&T);
4   while(T--){
5       ll ans;
6       int n, d, ai[999], bi[999], an[999];
7       sequence Seq;
8       scanf("%d%d",&n,&d);
9       FOR(0,d-1){
10          scanf("%d",&ai[i]);
11      }
12      FOR(0,d-1){
13          scanf("%d",&bi[i]);
14      }
15      FOR(0,d-1){
16          scanf("%d",&an[i]);
17      }
18      sequence = Seq(n,d,ai,bi,an, 999);
19      ans = solve(sequence);
20    printf("%llu\n", ans);
21  }
22  return 0;
23 }
```

Kode Sumber 4.4 Fungsi main

4.2.5 Implementasi Struct Linear Recursive Sequence

Pada subbab ini akan dijelaskan mengenai implementasi dari *Linear Recursive Sequence* pada subbab 3.2.2 dan *pseudocode* 3.2. Implementasi dari *struct* `SEQ` dapat dilihat pada kode sumber 4.5.

```
1 struct Seq {
2   int n, d, ai[999], bi[999], an[999],
        delta[999], beta[999],alpha[999];
3   Seq(int _n, int _d, int *_ai, int *_bi, int
        *_an, int size){
4       n = _n;
5       d = _d;
6       ai = memset(_ai,0,size * sizeof(*array));
7       bi = memset(_bi,0,size * sizeof(*array));
8       an = memset(_an,0,size * sizeof(*array));
9   }
10  void setBeta(){
11      beta[0] = bi[0];
12      FOR(1, d-1){
13          beta[i] = beta[i-1] + bi[i];
14      }
15  }
16  void setDelta(){
17      FOR(0, d-1){
18          delta[i] = an[i] - ai[i];
19      }
20  }
21  void setAlpha(){
22      alpha[0] = ai[0];
23      FOR(1, d-1){
24          alpha[i] = alpha[i-1] + ai[i];
25      }
26  }
27 };
```

Kode Sumber 4.5 Struct Seq

4.2.6 Implementasi Fungsi Mod

Pada subbab ini akan dijelaskan mengenai implementasi dari fungsi `Mod` pada *pseudocode* 3.7. Implementasi dapat dilihat pada kode sumber 4.6.

```
1 ll mod(int a){
2     a = (a%M + M)%M;
3     return a;
4 };
```

Kode Sumber 4.6 Fungsi Mod

4.2.7 Implementasi Fungsi FindMMI

Pada subbab ini akan dijelaskan mengenai implementasi dari fungsi `FindMMI` pada *pseudocode* 3.8. Implementasi dapat dilihat pada kode sumber 4.7.

```
1 int findMMI(int y){
2   return fastPow(y, M-2);
3 };
```

Kode Sumber 4.7 Fungsi FindMMI

4.2.8 Implementasi Fungsi FastPow

Pada subbab ini akan dijelaskan mengenai implementasi dari fungsi `FastPow` pada *pseudocode* 3.9. Implementasi dapat dilihat pada kode sumber 4.8.

```
 1 int fastpow(x, y){
 2   if(y == 0) return 1;
 3   if(y == 1) return x;
 4   ll halfn = fastPow(x, y/2);
 5   if (n\%2 == 0){
 6       return (halfn * halfn)\%M;
 7   }
 8   else{
 9       return ((halfn * halfn)\%M) * x)\%M;
10   }
11 };
```

Kode Sumber 4.8 Fungsi FastPow

4.2.9 Implementasi Fungsi Solve

Pada subbab ini akan dijelaskan mengenai implementasi dari fungsi `Solve` pada *pseudocode* 3.10. Implementasi dapat dilihat pada kode sumber 4.9.

```
 1 int solve(Seq sequence) {
 2     int temp, sn = 0;
 3     int xn = sequence.beta[d-1] - 1;
 4     if (xn == 0){
 5         sn = MOD(n / d * sequence.alpha[d-1];
 6         sn = MOD(sn + alpha[n\%d];
 7     }
 8     else {
 9         FOR(0,d-2){
10            temp = MOD(sequence.delta[i] *
                  sequence.beta[i]);
11            sn = MOD(sn + temp);
12         }
13         sn = MOD(sn + sequence.delta[d-1];
14         xn = findMMI(xn);
15         sn = MOD(sn * xn);
16         temp = 0;
17         FOR(0,d-2){
18             temp = MOD(temp + sequence.delta[i]);
19         }
20         sn = MOD(sn - temp);
21         sn = MOD(sn + sequence.ai[n]);
22     }
23     return sn;
24 };
```

Kode Sumber 4.9 Fungsi Solve

[*Halaman ini sengaja dikosongkan*]

BAB V
UJI COBA DAN ANALISIS

Pada bab ini dijelaskan tentang uji coba dan analisis dari implementasi yang telah dilakukan pada Tugas Akhir ini. Uji coba dilakukan dengan skenario yang tertentu dan dilanjutkan dengan tahap evaluasi dari metode penyelesaian yang digunakan.

5.1 Lingkungan Uji Coba

Lingkungan uji coba yang digunakan untuk uji kebenaran pada situs penilaian daring SPOJ, yaitu kluster *Cube* dengan spesifikasi sebagai berikut.

1. Perangkat Keras
   (a) Processor Intel Xeon E3-1220 v5 (5 CPUs)
   (b) *Random Access Memory* 1536MB
2. Perangkat Lunak
   (a) Compiler GCC 6.3.0

Lingkungan uji coba yang digunakan untuk melakukan uji coba kinerja menggunakan komputer pribadi penulis yang memiliki spesifikasi sebagai berikut

1. Perangkat Keras
   (a) Processor Intel® Core™ i7-6700HQ CPU @ 2.60GHz (8 CPUs), 2.6GHz
   (b) *Random Access Memory* 8192MB
2. Perangkat Lunak
   (a) Sistem Operasi Windows 10 Education 64-bit
   (b) Visual Studio Code
   (c) Bahasa Pemrograman C++

(d) Kompiler GCC 7.4.0 (Ubuntu 7.4.01ubuntu1 18.04.1) untuk Windows Subsystem Linux

5.2 Skenario Uji Coba

Subbab ini akan menjelaskan hasil pengujian program untuk menyelesaikan permasalahan HAL9000 - *100pct failure in 72 hours* . Metode pengujian yang dilakukan adalah sebagai berikut.

1. Pengujian luar. Pengujian ini menggunakan *Online Judge* untuk menguji kebenaran dan kinerja program.
2. Pengujian lokal. Pengujian ini menggunakan mesin yang digunakan dalam pengembangan untuk mengukur kinerja program.

Dalam pengujian lokal, dibuat beberapa kasus uji berdasarkan batasan pada subbab 1.3 dengan menggunakan nilai acak.

Untuk pengujian luar, uji coba dilakukan dengan mengirimkan kode program ke situs penilaian daring SPOJ masing masing sebanyak 10 kali.

5.3 Uji Coba Kebenaran

Pada subbab ini akan dibahas mengenai uji coba kebenaran yang dilakukan dengan mengirim kode sumber terkait ke dalam situs penilaian daring SPOJ. Uji coba dengan mengirimkan sumber kode dengan pendekatan naive menghasilkan *verdict Time Limit Exceeded* seperti yang dapat dilihat pada gambar 5.1. Hal ini dikarenakan kompleksitas waktu menggunakan metode naive yaitu $O(n)$, dengan nilai n dapat mencapai hingga 10^9, kurang efisien.

25909250	☐	2020-05-03 22:57:40	100pct failure in 72 hours	time limit exceeded edit ideone it	-	17M	C++ 4.3.2

Gambar 5.1 Hasil Uji Coba Kebenaran Situs SPOJ dengan pendekatan *naive*

Berikut bukti hasil pengujian sumber kode menggunakan algoritma *Gaussian Elimination*.

1. Hasil uji coba kebenaran kode sumber (Gambar 5.2)

26059695	☐	2020-05-29 03:15:12	100pct failure in 72 hours	**accepted** edit ideone it	0.02	4.5M	C++ 4.3.2

Gambar 5.2 Hasil Uji Coba Kebeneran Situs SPOJ dengan algoritma *Gaussian Elimination*

2. Peringkat pada HAL9000 - *100pct failure in 72 hours* (Gambar 5.3)

RANK	DATE	USER	RESULT	TIME	MEM	LANG
1	2014-06-13 14:48:05	Min_25	**accepted**	0.00	2.6M	C++ 4.3.2
2	2016-01-13 09:17:25	[Rampage] Blue.Mary	**accepted**	0.02	2.6M	C++ 4.3.2
3	2020-05-29 03:15:12	Fandy Putra	**accepted**	0.02	4.5M	C++ 4.3.2
4	2020-05-29 04:02:29	Rully Soelaiman	**accepted**	0.02	4.5M	CPP
5	2020-05-29 12:15:10	suhash	**accepted**	0.02	4.6M	CPP14
6	2014-06-10 15:27:55	Viplov Jain	**accepted**	0.14	4.0M	PYTHON
7	2014-06-10 15:24:30	Viplov Jain	**accepted**	0.19	12M	PYTHON3
8	2014-06-07 06:49:29	Mitch Schwartz	**accepted**	0.36	4.0M	PYTHON

Gambar 5.3 Peringkat pada HAL9000 - *100pct failure in 72 hours*

5.4 Uji Coba Kinerja Lokal

Pada subbab ini akan ditampilkan hasil uji coba kinerja dari kode program untuk menyelesaikan permasalahan HAL9000 - *100pct failure in 72 hours* . Pengujian dilakukan terhadap kelompok masukan yang telah dijelaskan pada subbab 5.2. Uji coba lokal dilakukan terhadap dua skenario, yaitu dengan nilai variabel d tetap dan yang kedua dengan nilai variabel n tetap. Hal ini dilakukan untuk menguji pengaruh nilai variabel terhadap pendekatan *naive* dan algoritma *Gaussian Elimination*. Detil masukan dapat dilihat pada

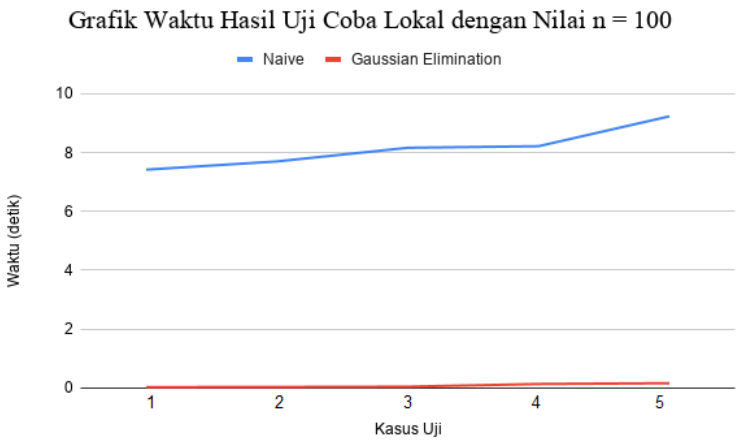
Lampiran A. Langkah pengujian kinerja dilakukan dengan langkah sebagai berikut.

1. Rekam waktu tepat sebelum komputasi penyelesaian masalah dilakukan.
2. Melakukan komputasi penyelesaian masalah untuk masukan kasus uji yang sama sebanyak 10 kali secara berturut-turut.
3. Rekam waktu tepat setelah komputasi penyelesaian masalah dilakukan.
4. Menghitung durasi waktu komputasi dengan mengurangi waktu selesai komputasi dengan waktu sebelum komputasi.
5. Ulangi untuk seluruh kasus uji.

Gambar 5.4 Rata-rata *Running Time* untuk Setiap Kasus Uji dengan nilai d tetap

Grafik 5.4 menampilkan rata-rata kinerja antara metode *Gaussian Elimination* dan metode *naive* dengan nilai d tetap. Pada grafik tersebut dapat dilihat bahwa rata-rata *running time* algoritma *Gauss Elimination* dan *Naive* berbeda cukup jauh, peningkatan

waktu terhadap nilai variabel n sangat berpengaruh terhadap pendekatan *naive*.



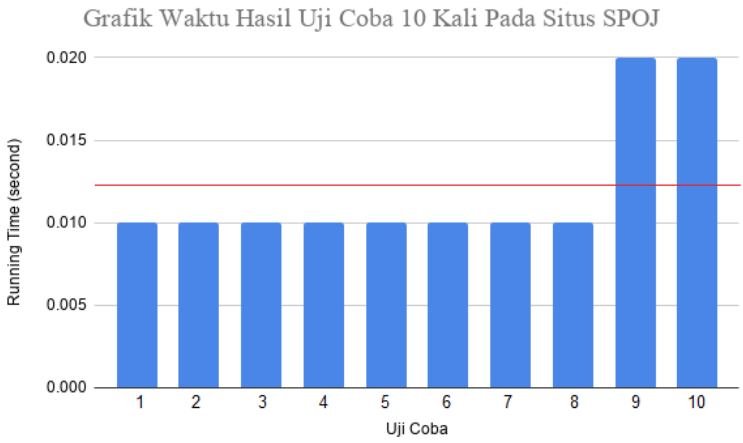
Gambar 5.5 Rata-rata *Running Time* untuk Setiap Kasus Uji dengan nilai n tetap

Grafik 5.5 menampilkan rata-rata kinerja antara metode *Gaussian Elimination* dan metode *naive* dengan nilai n tetap. Pada grafik tersebut dapat dilihat bahwa rata-rata *running time* algoritma *Gauss Elimination* dan *Naive* berbeda cukup jauh, namun peningkatan waktu terhadap nilai variabel d tidak terlalu berpengaruh besar.

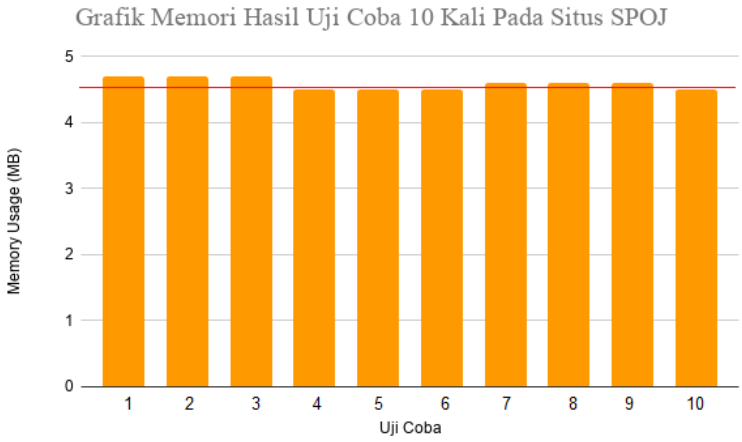
Pengujian lokal tidak dapat dilakukan hingga nilai n mencapai batas atas yaitu 10^9, dikarenakan dengan menggunakan pendekatan *naive* waktu yang dibutuhkan sangat lama. Dapat dilihat dari grafik uji coba, peningkatan waktu uji coba pendekatan *naive* dengan nilai n yang semakin bertambah cukup drastis.

5.5 Uji Coba Kinerja Luar

Pada subbab ini akan ditampilkan hasil uji coba kinerja dari algoritma *Gaussian Elimination* untuk menyelesaikan permasalahan HAL9000 - *100pct failure in 72 hours* . Pengujian dilakukan dengan cara mengirimkan kode program ke situs penilaian daring SPOJ. Detil mengenai hasil uji kinerja dapat dilihat pada Lampiran B. Rata rata hasil pengumpulan kode berkas dengan algoritma *Gauss Elimination* adalah $0,012$ detik dengan memori $4,86$ MB. Hasil uji coba pada situs SPOJ dapat dilihat pada gambar 5.6 dan 5.7



Gambar 5.6 Grafik Waktu Hasil Pengumpulan Kode Berkas Sebanyak 10 Kali



Gambar 5.7 Grafik Memori Hasil Pengumpulan Kode Berkas Sebanyak 10 Kali

[*Halaman ini sengaja dikosongkan*]

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini dijelaskan mengenai 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

Berdasarkan penjabaran di subbab-subbab sebelumnya dan dari hasil uji coba yang dilakukan, dapat disimpulkan beberapa poin terkait penyelesaian permasalahan HAL9000 - *100pct failure in 72 hours* .

1. Permasalahan HAL9000 - *100pct failure in 72 hours* dapat diselesaikan dengan memodelkan persamaan *linear recursive sequence* kedalam bentuk matriks menggunakan metode *Matrix Exponentiation*.
2. Permasalahan HAL9000 - *100pct failure in 72 hours* dengan batasan pada soal dapat diselesaikan menggunakan algoritma *Gaussian Elimination* dengan waktu minimal $0,01$ detik, waktu maksimal $0,02$ detik, memori minimal $4,5$ MB, dan memori maksimal $4,7$ MB.
3. Algoritma *Gaussian Elimination* yang sudah dimodifikasi terbukti efektif untuk mencari hasil invers yang dibutuhkan untuk mencari nilai jumlah n suku atau deret dari sebuah *linar recursive sequence*

6.2 Saran

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

1. Optimasi yang mungkin dilakukan yaitu penyederhanaan persamaan dan optimasi komputasi proses operasi untuk mempercepat waktu uji.
2. Optimasi untuk memperkecil penggunaan memori.

DAFTAR PUSTAKA

[1] Ankur. (2018). Modular Multiplicative Inverse. Accessed on: 2020-06-18, **url**: https : / / www . geeksforgeeks . org / multiplicative-inverse-under-modulo-m/.

[2] N. J. Higham, "Gaussian Elimination", 2011, Accessed on: 2020-06-21. **url**: https://doi.org/10.1002/wics.164.

[3] T. J. S. Lars Hansen, "Recursive Models of Dynamic Linear Economies", 2005.

[4] E. R. Pianka, "Exponential population growth", 2007.

[5] A. Smit, "Matrix Exponentiation", 2018, Accessed on: 2020-06-18. **url**: https : / / www . geeksforgeeks . org / matrix - exponentiation/.

[6] SPOJ, "100pct failure in 72 hours", 2014, Accessed on: 2020-06-18. **url**: https://www.spoj.com/problems/HAL9000/.

[7] S. Sutradhar, N. Sayadat, A. Rahman, S. Munira, A. F. Haque **and** S. Sakib, "IIR based digital filter design and performance analysis", 2017.

[*Halaman ini sengaja dikosongkan*]

LAMPIRAN A: DATA UJI

Tabel A.1 Data Uji dengan Nilai d = 2(1)

No	n	No	n
1	9	26	6
2	8	27	8
3	9	28	6
4	3	29	6
5	6	30	9
6	5	31	6
7	9	32	6
8	7	33	9
9	3	34	6
10	8	35	5
11	3	36	5
12	5	37	5
13	9	38	3
14	6	39	6
15	6	40	8
16	4	41	4
17	4	42	3
18	7	43	5
19	9	44	5
20	6	45	3
21	9	46	5
22	6	47	6
23	3	48	5
24	5	49	3
25	9	50	8

Tabel A.2 Data Uji dengan Nilai d = 2(2)

No	n	No	n
1	14	26	16
2	20	27	14
3	16	28	20
4	11	29	20
5	17	30	14
6	16	31	19
7	16	32	16
8	13	33	12
9	19	34	16
10	19	35	14
11	17	36	12
12	13	37	16
13	19	38	14
14	18	39	17
15	13	40	16
16	17	41	12
17	11	42	16
18	20	43	13
19	11	44	18
20	17	45	17
21	12	46	12
22	12	47	18
23	15	48	16
24	17	49	17
25	17	50	15

Tabel A.3 Data Uji dengan Nilai d = 2(3)

No	n	No	n
1	40	26	30
2	36	27	26
3	25	28	30
4	21	29	21
5	38	30	32
6	33	31	26
7	36	32	25
8	32	33	32
9	27	34	37
10	23	35	38
11	24	36	40
12	32	37	38
13	40	38	22
14	29	39	34
15	39	40	28
16	38	41	35
17	30	42	28
18	25	43	25
19	37	44	34
20	21	45	37
21	32	46	29
22	34	47	35
23	35	48	35
24	26	49	31
25	23	50	33

Tabel A.4 Data Uji dengan Nilai d = 2(4)

No	n	No	n
1	53	26	69
2	48	27	53
3	41	28	49
4	61	29	59
5	55	30	50
6	60	31	49
7	66	32	54
8	59	33	61
9	58	34	70
10	54	35	70
11	63	36	52
12	50	37	45
13	52	38	56
14	66	39	55
15	45	40	62
16	70	41	45
17	63	42	46
18	55	43	49
19	54	44	69
20	45	45	59
21	49	46	52
22	47	47	41
23	53	48	62
24	67	49	56
25	52	50	42

Tabel A.5 Data Uji dengan Nilai $d = 2(5)$

No	n	No	n
1	73	26	100
2	94	27	73
3	81	28	109
4	86	29	106
5	103	30	75
6	87	31	107
7	90	32	76
8	89	33	75
9	82	34	103
10	72	35	108
11	75	36	80
12	92	37	100
13	76	38	109
14	91	39	107
15	104	40	74
16	82	41	77
17	82	42	106
18	70	43	109
19	77	44	90
20	105	45	82
21	106	46	79
22	94	47	76
23	85	48	101
24	88	49	73
25	105	50	71

Tabel A.6 Data Uji dengan Nilai $n = 100(1)$

No	d	No	n
1	2	26	2
2	2	27	1
3	1	28	1
4	1	29	2
5	2	30	2
6	2	31	1
7	2	32	1
8	1	33	2
9	2	34	1
10	1	35	1
11	2	36	1
12	1	37	2
13	1	38	2
14	1	39	1
15	2	40	1
16	1	41	1
17	1	42	1
18	2	43	2
19	1	44	2
20	2	45	2
21	1	46	1
22	1	47	1
23	1	48	2
24	1	49	2
25	2	50	1

Tabel A.7 Data Uji dengan Nilai $n = 100(2)$

No	d	No	n
1	1	26	1
2	1	27	4
3	4	28	3
4	4	29	3
5	4	30	1
6	3	31	2
7	3	32	3
8	2	33	4
9	1	34	2
10	4	35	4
11	4	36	1
12	3	37	2
13	4	38	4
14	1	39	3
15	3	40	3
16	4	41	4
17	1	42	1
18	2	43	2
19	3	44	3
20	1	45	1
21	4	46	3
22	2	47	4
23	3	48	3
24	3	49	3
25	1	50	4

Tabel A.8 Data Uji dengan Nilai $n = 100(3)$

No	d	No	n
1	3	26	9
2	7	27	3
3	8	28	6
4	5	29	5
5	7	30	3
6	2	31	4
7	1	32	9
8	4	33	7
9	5	34	1
10	8	35	5
11	7	36	2
12	6	37	8
13	5	38	4
14	8	39	2
15	6	40	8
16	2	41	3
17	2	42	3
18	4	43	1
19	4	44	7
20	6	45	4
21	1	46	8
22	9	47	6
23	7	48	1
24	4	49	8
25	9	50	9

Tabel A.9 Data Uji dengan Nilai $n = 100(4)$

No	d	No	n
1	13	26	13
2	11	27	1
3	11	28	9
4	19	29	6
5	4	30	9
6	11	31	8
7	16	32	11
8	2	33	13
9	15	34	6
10	4	35	10
11	2	36	4
12	10	37	6
13	2	38	11
14	11	39	12
15	19	40	12
16	6	41	15
17	3	42	9
18	1	43	17
19	10	44	17
20	10	45	10
21	9	46	10
22	9	47	13
23	11	48	15
24	19	49	10
25	8	50	8

Tabel A.10 Data Uji dengan Nilai $n = 100$(5)

No	d	No	n
1	98	26	96
2	97	27	99
3	93	28	94
4	99	29	90
5	95	30	90
6	94	31	95
7	97	32	92
8	97	33	99
9	99	34	98
10	94	35	96
11	95	36	91
12	95	37	95
13	96	38	99
14	95	39	95
15	93	40	98
16	98	41	90
17	99	42	93
18	93	43	96
19	97	44	90
20	96	45	90
21	94	46	95
22	94	47	95
23	96	48	99
24	92	49	96
25	91	50	98

LAMPIRAN B: Hasil Uji Coba pada Situs SPOJ Sebanyak 10 Kali

Berikut merupakan lampiran hasil uji coba kode program sebanyak 10 kali.

26151614	☐	2020-06-16 07:16:44	100pct failure in 72 hours	**accepted** edit ideone it	0.02	4.5M	CPP14
26151609	☐	2020-06-16 07:16:26	100pct failure in 72 hours	**accepted** edit ideone it	0.02	4.6M	CPP14
26151606	☐	2020-06-16 07:16:12	100pct failure in 72 hours	**accepted** edit ideone it	0.01	4.6M	CPP14
26151605	☐	2020-06-16 07:16:00	100pct failure in 72 hours	**accepted** edit ideone it	0.01	4.6M	CPP14
26151601	☐	2020-06-16 07:15:48	100pct failure in 72 hours	**accepted** edit ideone it	0.01	4.5M	CPP14
26151600	☐	2020-06-16 07:15:38	100pct failure in 72 hours	**accepted** edit ideone it	0.01	4.5M	CPP14
26151597	☐	2020-06-16 07:15:26	100pct failure in 72 hours	**accepted** edit ideone it	0.01	4.5M	CPP14
26151596	☐	2020-06-16 07:15:15	100pct failure in 72 hours	**accepted** edit ideone it	0.01	4.7M	CPP14
26151595	☐	2020-06-16 07:15:04	100pct failure in 72 hours	**accepted** edit ideone it	0.01	4.7M	CPP14
26151593	☐	2020-06-16 07:14:41	100pct failure in 72 hours	**accepted** edit ideone it	0.01	4.7M	CPP14

Gambar B.1 Hasil Pengumpulan Kode Program Utama

[*Halaman ini sengaja dikosongkan*]

LAMPIRAN C: Kode Sumber Uji Coba pada Situs SPOJ

Berikut merupakan lampiran kode sumber uji coba kode program.

```
#include <stdio.h>
#define M 1000000007
#define ll long long
#define lp(x,n) for(i=x;i<n;i++)
#define re(n) scanf("%lld",&n)
#define W 1001
int w(ll b, ll n)
{
    if(n==0) return 1;
    if(n==1) return b;
    ll h=w(b,n/2);
    if(n%2==0) return (h*h)%M;
    return (((h*h)%M)*b)%M;
}
int v(int n){return w(n,M-2);}
int o(ll a){return (a%M+M)%M;}
int main(){
    ll tm,a[W],b[W],h[W],B[W],A[W],D[W],x,y,t,d,n,i,z;
    re(t);
    while(t--){
        re(d);re(n);
        lp(0,d) re(a[i]);
        lp(0,d) re(b[i]);
        lp(0,d) re(h[i]);
        B[0]=b[0];A[0]=a[0];D[0]=h[0]-a[0];
        lp(1,d) {B[i]=B[i-1]+b[i];A[i]=A[i-1]+a[i];D[i]=h[i]-a[i];}
        z=B[d-1]-1;
        if(z==0) x=(((n/d*A[d-1])%M)+(A[n%d]%M))%M;
        else{
            x=0,y=0;
            lp(0,d-1){tm=((D[i]%M)*(B[i]%M))%M;x=((x%M)+(tm%M))%M;y=(y%M+D[i]%M)%M;}
            x=o(o(x)+o(D[d-1]));x=o(x*v(z));x=o(x-o(y));x=(x%M+h[0]%M)%M;
        }
        printf("%lli\n",x);
    }
}
```

Gambar C.1 Kode Sumber Uji Coba pada Situs SPOJ

[*Halaman ini sengaja dikosongkan*]

BIODATA PENULIS

Penulis bernama Fandy Putra Mohammad, putra kedua dari dua bersaudara yang lahir pada tanggal 23 Februari 1999 di Padang. Penulis telah mengenyam pendidikan di Sekolah Dasar Dian Andalas pada tahun 2004 hingga 2010, Sekolah Menengah Pertama Negeri 8 Padang pada tahun 2010 hingga 2013, dan Sekolah Menengah Atas Negeri 10 Padang pada tahun 2013 hingga 2016. Pada masa penulisan, penulis sedang menempuh masa studi S1 di Institut Teknologi Sepuluh Nopember, Surabaya di Departemen Teknik Informatika.

Selama masa studi, penulis memiliki ketertarikan di bidang *data mining* dan rancang bangun aplikasi. Keinginan penulis dalam membantu sesama juga mendorong penulis menjadi asisten dosen pada mata kuliah Dasar Pemrograman dan Struktur Data. Selain itu penulis juga aktif menjadi *administrator* di lab Algoritma dan Pemrograman. Karya penulis semasa perkuliahan diantaranya adalah pembangunan aplikasi Sistem Informasi Pelajar(SIMPEL) untuk acara Hackathon. Di luar kesibukan akademik, penulis juga berkontribusi dalam berbagai kepanitiaan, baik dalam skala kecil (yaitu dalam kampus) maupun skala nasional. Organisasi yang penulis ikuti adalah HMTC (2017 dan 2018). Selain aktif mengikuti organisasi, penulis juga aktif dalam Schematics sebagai staff pada tahun 2017 dan wakil bph NLC pada tahun 2018. Penulis dapat dihubungi melalui surel di fandyputram@gmail.com.