TUGAS AKHIR – IF184802

Desain dan Analisis Algoritma Dynamic Programming dengan Optimasi Symmetrical State dan Pruning Heuristic pada Studi Kasus UVa Online Judge Guess The Fake Coin II

FADLI WILDAN FIRJATULLAH
05111640000126

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

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

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

TUGAS AKHIR – IF184802

DESAIN DAN ANALISIS ALGORITMA SIMULASI PENIMBANGAN DENGAN PENDEKATAN DIVIDE AND CONQUER PADA STUDI KASUS CODEFORCES 306 - BALANCE

FADLI WILDAN FIRJATULLAH
05111640000126

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

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

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

TUGAS AKHIR – IF184802

DESAIN DAN ANALISIS ALGORITMA SIMULASI PENIMBANGAN DENGAN PENDEKATAN DIVIDE AND CONQUER PADA STUDI KASUS CODEFORCES 306 – BALANCE

FADLI WILDAN FIRJATULLAH
05111640000126

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

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

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

[*Halaman ini sengaja dikosongkan*]

UNDERGRADUATE THESIS – IF184802

DESIGN AND ANALYSIS WEIGHING SIMULATION ALGORITHM USING DIVIDE AND CONQUER APPROACH WITH CASE STUDY CODEFORCES 306 - BALANCE

FADLI WILDAN FIRJATULLAH
05111640000126

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

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

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

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

DESAIN DAN ANALISIS ALGORITMA SIMULASI PENIMBANGAN DENGAN PENDEKATAN DIVIDE AND CONQUER PADA STUDI KASUS CODEFORCES 306 - BALANCE

TUGAS AKHIR

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

Oleh:
Fadli Wildan Firjatullah
NRP: 051116 40000 126

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
2020

[*Halaman ini sengaja dikosongkan*]

DESAIN DAN ANALISIS ALGORITMA SIMULASI PENIMBANGAN DENGAN PENDEKATAN DIVIDE AND CONQUER PADA STUDI KASUS CODEFORCES 306 - BALANCE

Nama Mahasiswa	:	Fadli Wildan Firjatullah
NRP	:	051116 40000 126
Departemen	:	Teknik Informatika Fakultas Teknologi Elektro dan Informatika Cerdas - ITS
Dosen Pembimbing 1	:	Rully Soelaiman, S.Kom., M.Kom.
Dosen Pembimbing 2	:	Yudhi Purwananto, S.Kom, M.Kom.

Abstrak

Teka-teki keseimbangan adalah sebuah permasalahan klasik yang memiliki banyak variasi teka-teki. Teka-teki dilakukan dengan menimbang kelompok koin pada skala keseimbangan. Tujuan dari teka-teki adalah mencari koin palsu dimana memiliki bobot yang berbeda dibanding koin lain.

Pada tugas akhir ini, akan dirancang penyelesaian permasalahan teka-teki keseimbangan dengan menghasilkan semua rute koin palsu yang mungkin dengan penimbangan yang dibutuhkan seminimum mungkin. Permasalahan ini dapat diselesaikan dengan menggunakan simulasi penimbangan dengan pendekatan divide and conquer.

Hasil dari tugas akhir ini telah berhasil menyelesaikan permasalahan diatas dengan cukup efisien, dengan rata-rata waktu eksekusi 31 milidetik dengan rata-rata penggunaan memori 140 KB.

[*Halaman ini sengaja dikosongkan*]

DESIGN AND ANALYSIS WEIGHING SIMULATION ALGORITHM USING DIVIDE AND CONQUER APPROACH WITH CASE STUDY CODEFORCES 306 – BALANCE

Student Name	: Fadli Wildan Firjatullah
Registration Number	: 051116 40000 126
Department	: Informatics Engineering Department Faculty of Intelligent Electrical and Informatics Technology
First Supervisor	: Rully Soelaiman, S.Kom., M.Kom.
Second Supervisor	: Yudhi Purwananto, S.Kom, M.Kom.

Abstract

Balance puzzle is a classic problem that has many puzzle variations. The puzzle is done by weighing groups of coins on a balance scale. The aim of the puzzle is to find a fake coin which has a different weight compared to other coins.

In this thesis, a solution to the balance puzzle problem will be designed to produce all the possible fake coin routes with the minimum required weighing. This problem can be solved by using a weighing simulation with a divide and conquer approach.

The results of this thesis have successfully resolved the problems quite efficiently, with an average execution time of 31 milliseconds with an average memory usage of 140KB.

[*Halaman ini sengaja dikosongkan*]

KATA PENGANTAR

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

DESAIN DAN ANALISIS ALGORITMA SIMULASI PENIMBANGAN DENGAN PENDEKATAN DIVIDE AND CONQUER PADA STUDI KASUS CODEFORCES 306 - BALANCE

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 Yudhi Purwananto, S.Kom, M.Kom. selaku dosen pembimbing yang telah memberikan ilmu, dan masukan kepada penulis.
3. Bapak, Ibu, dan kerabat terdekat penulis yang selalu memberikan dukungan, perhatian, dan kasih sayang bagi penulis yang menjadi semangat selama perkuliahan maupun pengerjaan tugas akhir.

4. Yolanda Hertita Pratama yang tidak lelah membantu penulis selama masa studi dan dalam pengerjaan tugas akhir ini.
5. Ferdinand Jason yang tidak lelah membantu penulis selama masa studi dan dalam pengerjaan tugas akhir ini.
6. Yoshima Syach Putri yang tidak lelah membantu penulis selama masa studi dan dalam pengerjaan tugas akhir ini.
7. Teman-teman angkatan 2016 Departemen Teknik Informatika ITS yang telah menemani perjuangan penulis selama 4 tahun masa perkuliahan.
8. 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, 23 Juni 2020

Fadli Wildan Firjatullah

DAFTAR ISI

LEMBAR PENGESAHAN ..... **v**
Abstrak ..... **vii**
Abstract ..... **ix**
KATA PENGANTAR ..... **xi**
DAFTAR ISI ..... **xiii**
DAFTAR GAMBAR ..... **xv**
DAFTAR TABEL ..... **xvii**
DAFTAR PSEUDOCODE ..... **xix**
DAFTAR KODE SUMBER ..... **xxi**
BAB I PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Rumusan Permasalahan 2
1.3 Batasan Permasalahan 2
1.4 Tujuan Pembuatan Tugas Akhir 3
1.5 Manfaat Tugas Akhir 3
1.6 Metodologi 4
1.6.1 Penyusunan Proposal Tugas Akhir 4
1.6.2 Studi Literatur 4
1.6.3 Implementasi Algoritma 4
1.6.4 Pengujian dan Evaluasi 5
1.6.5 Penyusunan Buku Tugas Akhir 5
1.7 Sistematika Penulisan 5
BAB II DASAR TEORI **7**
2.1 Deskripsi Umum Permasalahan 7
2.2 Deskripsi Umum Teori 9
2.2.1 Balance Puzzle 9
2.2.2 Divide and Conquer 10
BAB III DESAIN DAN ANALISIS **13**
3.1 Desain dan Analisis Penyelesaian Masalah 13
3.2 Solusi Menggunakan Divide and Conquer 13
3.2.1 Tahap Divide 13
3.2.1.1 Tahap Divide pada Solve1 14

3.2.1.2 Tahap Divide pada Solve2 14
3.2.2 Tahap Conquer 15
3.2.1.1 Tahap Conquer pada Solve1..... 15
3.2.1.2 Tahap Conquer pada Solve2..... 15
3.2.3 Tahap Combine 16
3.3 Deskripsi Umum Sistem..... 16
3.3.1 Fungsi Solve1 17
3.3.2 Fungsi Solve2 18
BAB IV IMPLEMENTASI.....21
4.1 Lingkungan Implementasi..... 21
4.2 Pendefinisian Library, Preprocessor Directives, Fungsi, dan Variabel Global 21
4.3 Fungsi Input dan Output..... 22
4.4 Fungsi Divide 23
4.5 Fungsi Main 24
4.6 Fungsi Solve1 25
4.7 Fungsi Solve2 26
BAB V UJI COBA DAN EVALUASI.....31
5.1 Lingkungan Uji Coba 31
5.2 Uji Coba Kebenaran 31
5.2.1 Uji Coba Kebenaran Lokal..... 31
5.2.2 Uji Coba dengan Algoritma Pembanding 40
5.2.3 Uji Coba Kebenaran pada Codeforces 41
BAB VI KESIMPULAN DAN SARAN.....45
6.1 Kesimpulan 45
6.2 Saran..... 45
DAFTAR PUSTAKA 47
LAMPIRAN A: HASIL UJI COBA LOKAL PADA 38 KOIN 49
LAMPIRAN B: HASIL UJI COBA LOKAL PADA 100 KOIN 58
BIODATA PENULIS.....85

DAFTAR GAMBAR

Gambar 2.1 Soal 306 – Balance pada Codeforces 7
Gambar 5.1 Hasil uji coba lokal pada 4 koin 32
Gambar 5.2.1 Hasil uji coba lokal pada 12 koin bagian 1.....35
Gambar 5.2.2 Hasil uji coba lokal pada 12 koin bagian 2.....36
Gambar 5.2.3 Hasil uji coba lokal pada 12 koin bagian 3.....37
Gambar 5.3 Ilustrasi hasil uji coba pada 4 koin 38
Gambar 5.4 Ilustrasi hasil uji coba pada 12 koin 39
Gambar 5.5 Grafik waktu eksekusi dan memori dari hasil uji coba pada situs Codeforces.....42
Gambar 5.6 Uji coba kebenaran pada Codeforces 43

[*Halaman ini sengaja dikosongkan*]

DAFTAR TABEL

Tabel 2.1 Sampel *input* dan *output*.....8
Tabel 2.2 Variasi Balance Puzzle.....10
Tabel 4.1 Spesifikasi Lingkungan Implementasi 21
Tabel 5.1 Penjabaran rute koin palsu. 33
Tabel 5.2 Hasil pengujian dengan algoritma pembanding.....40

[*Halaman ini sengaja dikosongkan*]

DAFTAR PSEUDOCODE

Pseudocode 3.1 Fungsi main 17
Pseudocode 3.2 Fungsi solve1 18
Pseudocode 3.3 Fungsi solve2 20

[*Halaman ini sengaja dikosongkan*]

DAFTAR KODE SUMBER

Kode Sumber 4.1 Pendefinisian library, preprocessor directives, fungsi, dan variabel global 22
Kode Sumber 4.2 Fungsi input dan output..... 23
Kode Sumber 4.3 Fungsi divide..... 23
Kode Sumber 4.4 Fungsi main..... 24
Kode Sumber 4.5 Fungsi solve1 26
Kode Sumber 4.6 Fungsi solve2 29

[*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 akhir yang diajukan ini bermula dari adanya permasalahan 306 - Balance pada *Codeforces*. Permasalahan tersebut merupakan salah satu variasi permasalahan klasik teka-teki keseimbangan. Permasalahan bermula ketika diberikan koin sebanyak n. Terdapat sebuah koin palsu $\{x \mid x \in n\}$ yang diidentifikasi karena memiliki bobot berbeda dengan koin asli. Bobot koin palsu bisa lebih ringan atau lebih berat dibanding koin asli. Untuk dapat mengidentifikasi koin palsu diberikan sebuah alat bantu yaitu skala keseimbangan untuk menimbang koin yang telah diberikan. Dari skala keseimbangan tersebut didapatkan informasi apakah tumpukan kiri lebih ringan dari tumpukan kanan, tumpukan kiri lebih berat dari tumpukan kanan, atau tumpukan kiri sama berat dengan tumpukan kanan. Masalah yang harus diselesaikan pada kasus ini adalah bagaimana cara menghasilkan semua rute koin palsu yang mungkin untuk setiap koin dengan menggunakan skala keseimbangan yang mana penimbangan maksimal dapat dilakukan seminimal mungkin untuk setiap n koin dan setiap rute dapat mengidentifikasi koin palsu.

Hasil dari tugas akhir ini adalah implementasi solusi dari pencarian koin palsu dan mampu menyelesaikan permasalahan dari 306 - Balance.

1.2 Rumusan Permasalahan

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

1. Bagaimana cara menghasilkan semua rute koin palsu yang mungkin untuk setiap koin dengan penimbangan maksimal dapat dilakukan seminimal mungkin.
2. Apakah algoritma simulasi penimbangan dengan pendekatan *divide and conquer* dapat menghasilkan semua rute koin palsu?
3. Bagaimana implementasi algoritma simulasi penimbangan dengan pendekatan *divide and conquer* yang efisien dan optimal untuk menghasilkan semua rute koin palsu?
4. Apakah solusi menggunakan algoritma simulasi penimbangan dengan pendekatan *divide and conquer* dapat diterima pada pengujian *online Codeforces*?

1.3 Batasan Permasalahan

Permasalahan pada pembuatan Tugas Akhir ini memiliki beberapa batasan. Batasan-batasan dari *Codeforces* adalah sebagai berikut.

- Implementasi algoritma menggunakan bahasa pemrograman C, C++, C#, D, Go, Haskell, Java, Kotlin, OCaml, Delphi, Pascal, Perl, PHP Python, Ruby, Rust, Scala, JavaScript.
- Batas koin yang diujikan adalah $3 \leq n \leq 100$.
- Tidak boleh mencetak kasus yang tidak mungkin pada simulasi penimbangan.
- Banyaknya koin yang akan ditimbang pada sisi kiri sama dengan pada sisi kanan.

- Tidak ada koin yang sama diletakkan pada kedua sisi.
- Format *input* dan *output* seperti pada sampel *input* dan *output* yang disediakan pada soal.
- Batasan waktu untuk setiap tes adalah 0,5 detik.
- Batasan memori adalah 65536 kilobytes.

1.4 Tujuan Pembuatan Tugas Akhir

Tujuan dari Tugas Akhir ini antara lain,

1. Merancang sebuah solusi yang dapat menghasilkan semua rute koin palsu yang mungkin untuk setiap koin dengan penimbangan maksimal dapat dilakukan seminimal mungkin.
2. Melakukan desain dan analisis algoritma dengan pendekatan *divide and conquer* untuk menghasilkan semua rute koin palsu dalam studi kasus permasalahan dari 306 - Balance.
3. Mengimplementasikan algoritma dengan pendekatan *divide and conquer* untuk menghasilkan semua rute koin palsu dalam studi kasus permasalahan dari 306 - Balance.
4. Mengevaluasi hasil dan kinerja algoritma yang telah dirancang untuk permasalahan dari 306 - Balance.

1.5 Manfaat Tugas Akhir

Permasalahan menghasilkan semua rute koin palsu dapat menjadi penting dalam beberapa aplikasi praktis. Sebagai contoh rute koin palsu dapat digunakan untuk mendeteksi letak dari koin palsu. Dan deteksi koin palsu dapat dianalogikan sebagai deteksi kesalahan yang mana teknik yang digunakan dapat memungkinkan pengiriman data digital yang andal melalui saluran komunikasi yang tidak andal. Banyak saluran

komunikasi terhalang oleh kebisingan saluran, dan dengan demikian kesalahan dapat terjadi selama transmisi dari sumber ke penerima. Teknik pendeteksi kesalahan memungkinkan mendeteksi kesalahan seperti itu.

1.6 Metodologi

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

1.6.1 Penyusunan Proposal Tugas Akhir

Tahap awal memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Pada proposal ini, penulis mengajukan gagasan untuk menyelesaikan permasalahan pada studi kasus *Codeforces* 306 - Balance dengan algoritma simulasi penimbangan dengan pendekatan *divide and conquer*.

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 algoritma dan struktur data. Informasi tersebut didapatkan dari buku, internet, dan materi kuliah yang berhubungan dengan metode yang akan digunakan.

1.6.3 Implementasi Algoritma

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

1.6.4 Pengujian dan Evaluasi

Pada tahap ini dilakukan uji coba kebenaran dan kinerja solusi yang telah diimplementasikan dengan melakukan pengujian secara lokal dan melakukan pengiriman sumber kode sistem ke situs penilaian *Codeforces* pada permasalahan yang terkait untuk mendapatkan umpan balik. Pengujian dilakukan dengan membandingkan kompleksitas hasil uji coba dengan kompleksitas hasil analisis.

1.6.5 Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi algoritma yang telah dibuat.

1.7 Sistematika Penulisan

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

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

Bab IV Implementasi

Bab ini berisi penjelasan implementasi dari rancangan yang telah dibuat pada bab III. Implementasi disajikan dalam bentuk *pseudocode* disertai dengan penjelasannya.

Bab V Uji Coba dan Evaluasi

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

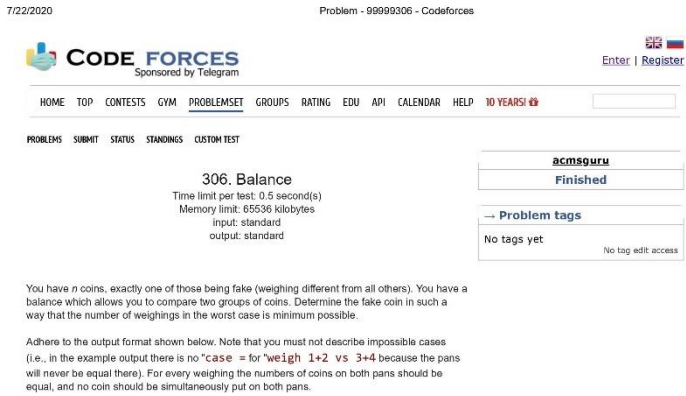
Bab VI Kesimpulan dan Saran

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

BAB II
DASAR TEORI

Bab ini membahas mengenai permasalahan *Codeforces* 306 - Balance dan landasan teori yang digunakan untuk menyelesaikan pengerjaan Tugas Akhir ini. Teori yang digunakan diantaranya adalah *Balance Puzzle* dan *Divide and Conquer*.

2.1 Deskripsi Umum Permasalahan



Gambar 2.1 Soal 306 – Balance pada Codeforces

Pada permasalahan *Codeforces* 306 – Balance, diberikan *input* n yang berarti banyaknya koin yang akan diujikan. Dari n koin yang diberikan, terdapat sebuah koin palsu di dalamnya dengan ciri-ciri memiliki bobot berbeda dengan koin asli. Bobot koin palsu ini bisa lebih ringan atau lebih berat dibanding koin asli. Untuk mempermudah mencari koin palsu tersebut, diberikan sebuah skala keseimbangan yang bisa digunakan untuk membandingkan bobot dari dua kelompok koin. Tugas dari soal ini adalah menentukan semua

koin palsu yang mungkin dengan jumlah penimbangan yang dibutuhkan pada kasus terburuk seminimal mungkin.

Tabel 2.1 Sampel *input* dan *output*

sample input	sample output

4	need 2 weighings
	weigh 1 vs 2
	case <:
	  weigh 1+2 vs 3+4
	  case <:
	  fake 1
	  case >:
	  fake 2
	  end
	case =:
	  weigh 1 vs 3
	  case =:
	  fake 4
	  case <:
	  fake 3
	  case >:
	  fake 3
	  end
	case >:
	  weigh 1+2 vs 4+3
	  case >:
	  fake 1
	  case <:
	  fake 2
	  end
	end

Batasan n yang diberikan pada soal yaitu $3 \leq n \leq 100$. Sampel dari *input* dan *output* dapat dilihat pada Tabel 2.1. Terdapat catatan bahwa tidak boleh menampilkan kasus penimbangan yang tidak mungkin. Sebagai contoh pada Tabel

2.1 tidak ada “`case =`” untuk “`weigh 1+2 vs 3+4`” karena kedua sisi tidak mungkin seimbang. Pada setiap penimbangan banyaknya koin pada kedua sisi skala keseimbangan harus sama banyak dan tidak boleh ada koin yang sama diletakkan pada kedua sisi skala keseimbangan yang mana mustahil untuk dilakukan.

2.2 Deskripsi Umum Teori

Pada subbab ini akan dijelaskan landasan teori secara umum yang digunakan untuk melakukan pendekatan terhadap penyelesaian permasalahan.

2.2.1 Balance Puzzle

Balance Puzzle adalah teka-teki logika tentang menyeimbangkan barang untuk mencari tahu manakah barang yang menyimpan nilai yang berbeda menggunakan skala keseimbangan dengan banyaknya penimbangan yang terbatas. Nilai bobot setiap barang tidak diberikan dan bobot setiap barang relatif satu sama lain. Variasi dari *Balance Puzzle* ditunjukkan pada Tabel 2.2.

Sebagai contoh untuk mendeteksi koin palsu dengan maksimal tiga kali penimbangan, koin maksimum yang dapat dianalisis adalah (3^3-1)/2=13. Terdapat catatan bahwa dengan 13 koin dan 3 penimbangan hanya bisa didapatkan koin palsu, namun tidak bisa untuk mendapatkan bobot dari koin palsu pada koin terakhir. Untuk mendapatkan koin palsu beserta bobot koin palsu, maka dalam tiga kali penimbangan koin yang dapat dievaluasi adalah (3^3-1)/2-1=12.

Tabel 2.2 Variasi Balance Puzzle

Diketahui	Target	Koin Maksimum untuk n Penimbangan	Banyaknya Penimbangan untuk c Koin
Koin palsu pasti lebih ringan atau pasti lebih berat dibanding koin lain	Mencari koin palsu	3^n	$\lceil \log_3(c) \rceil$
Koin palsu memiliki bobot berbeda dengan koin lain	Mencari koin palsu	$\frac{3^n - 1}{2}$	$\lceil \log_3(2c + 1) \rceil$
Koin palsu memiliki bobot yang berbeda dengan koin lain	Mencari koin palsu dan mencari bobot dari koin palsu	$\frac{3^n - 1}{2} - 1$	$\lceil \log_3(2c + 3) \rceil$

2.2.2 Divide and Conquer

Divide and Conquer adalah sebuah paradigma desain algoritma yang menyelesaikan sebuah permasalahan dengan rekursif bercabang banyak. Terdapat tiga tahapan penting pada *Divide and Conquer* yaitu:

- *Divide*: Membagi sebuah permasalahan ke dalam beberapa subpermasalahan yang lebih kecil.

- *Conquer*: Subpermasalahan yang akan diselesaikan dengan rekursif. Jika subpermasalahan sudah cukup kecil maka akan diselesaikan secara langsung.
- *Combine*: Menggabungkan solusi dari subpermasalahan menjadi sebuah solusi utuh dari permasalahan.

Untuk menyelesaikan permasalahan *Codeforces* 306-Balance, akan digunakan metode *recursion-tree* untuk menyelesaikan rekurensi. Pada *recursion-tree*, setiap *node* merepresentasikan biaya dari sebuah submasalah di suatu tempat di set pemanggilan fungsi rekursif. Biaya dalam setiap tingkat pada pohon dijumlahkan untuk mendapatkan satu set biaya per-tingkat, dan kemudian semua biaya per-tingkat dijumlahkan untuk menentukan biaya total semua tingkat rekursi.

Recursion-tree paling baik digunakan untuk menghasilkan tebakan yang baik. Saat menggunakan *recursion-tree* untuk menghasilkan tebakan yang baik, beberapa kali "kecerobohan" kecil dapat ditolerir karena di akhir akan dilakukan verifikasi. Jika ketika menggambar *recursion-tree* dilakukan secara hati-hati, maka *recursion-tree* dapat menjadi bukti langsung dari solusi rekurensi.

[*Halaman ini sengaja dikosongkan*]

BAB III
DESAIN DAN ANALISIS

Pada bab ini akan dijelaskan mengenai desain dan analisis algoritma yang digunakan dalam strategi penyelesaian permasalahan *Codeforces* 306 – Balance.

3.1 Desain dan Analisis Penyelesaian Masalah

Berdasarkan deskripsi umum permasalahan yang telah dijelaskan sebelumnya, maka proses analisis algoritma menggunakan metode *divide and conquer* yang akan digunakan untuk mensimulasikan proses penimbangan pada koin untuk mendapatkan koin palsu.

3.2 Solusi Menggunakan Divide and Conquer

Penyelesaian suatu permasalahan menggunakan pendekatan *divide and conquer* dilakukan dengan cara rekursif dengan mengaplikasikan tiga tahapan pada setiap level rekursi, yaitu *divide*, *conquer*, dan *combine*.

3.2.1 Tahap Divide

Pertama-tama, n koin dengan penimbangan k dengan $k = \lceil \log_3(2n+1) \rceil$ akan dibagi menjadi tiga bagian. Jika $n \, mod \, 3 = 0$, maka dibagi menjadi tiga bagian sama banyak yaitu x, x, x. Jika $n \, mod \, 3 = 1$ maka dibagi menjadi $x, x, x + 1$. Jika $n \, mod \, 3 = 2$ maka akan dibagi menjadi $x, x + 1, x + 1$. Tiga bagian tersebut kemudian diberi nama A, B, dan C dengan ketentuan banyaknya koin pada bagian A sama dengan banyaknya koin pada bagian B. A dan B dievaluasi pada skala penimbangan, dan C berada di luar skala penimbangan. Jika bobot A dan B sama, maka koin palsu berada pada C. Karena itu, C perlu dievaluasi lebih lanjut pada fungsi `Solve1`. Jika

bobot A dan B berbeda, maka koin palsu berada di antara keduanya. Karena itu, A dan B pada fungsi `Solve2`.

3.2.1.1 Tahap Divide pada Solve1

Pada bagian ini, n koin akan dibagi menjadi tiga bagian yaitu A, B, dan C dengan banyaknya A sama dengan B. Jika $n\ mod\ 3 = 0$ maka dibagi menjadi x, x, x. Jika $n\ mod\ 3 > 0$ maka pada bagian A akan ditambah sebuah koin asli yang didapat dari hasil penimbangan sebelumnya dan pada bagian B akan ditambahkan koin yang berada paling belakang dari tumpukan n koin. Jika $n\ mod\ 3 = 2$ maka pada bagian C akan ditambah koin yang berada di kedua paling belakang dari tumpukan n koin. A dan B dievaluasi pada skala penimbangan. Jika bobot A dan B sama maka dilanjutkan dengan mengevaluasi C ke fungsi `Solve1` secara rekursif. Jika bobot A dan B berbeda maka lanjut ke fungsi `Solve2` secara rekursif.

3.2.1.2 Tahap Divide pada Solve2

Pada bagian ini, m koin dan n koin akan dibagi menjadi enam bagian yaitu A_1, A_2, A_3, B_1, B_2, dan B_3. Proses pembagian m koin menjadi A_1, A_2, A_3 dan n koin menjadi B_1, B_2, B_3 seperti pada tahap *divide* yang pertama. Jika $m\ mod\ 3 = 1$ dan $n\ mod\ 3 = 1$ maka koin yang berada di tumpukan paling belakang pada B_3 dipindah ke B_1 dan tumpukan kedua paling belakang dari B_3 dipindah ke B_2. Lalu dilakukan evaluasi terhadap $A_1 + B_1$ dengan $A_2 + B_2$ pada skala keseimbangan. $A_3 + B_3$ berada di luar skala keseimbangan. Jika $A_1 + B_1 < A_2 + B_2$ maka dilanjutkan dengan mengevaluasi A_2 dan B_1 ke fungsi `Solve2` secara rekurtif. Jika $A_1 + B_1 = A_2 + B_2$ maka dilanjutkan dengan

mengevaluasi A_3 dan B_3 ke fungsi `Solve2` secara rekursif. Jika $A_1 + B_1 > A_2 + B_2$ maka dilanjutkan dengan mengevaluasi A_1 dan B_2 ke fungsi `Solve2` secara rekursif. Jika terdapat kasus dimana ada m koin dengan $m > 2$ dan tidak ada koin n, maka m dipindah ke fungsi `Solve1`. Begitu pula jika terdapat kasus dimana ada n koin dengan $n > 2$ dan tidak ada koin m, maka n dipindah ke fungsi `Solve1`.

3.2.2 Tahap Conquer

Pada tahapan ini dijelaskan mengenai penanganan kasus ketika ukuran dari kasus sudah cukup kecil untuk dievaluasi sehingga koin palsu bisa ditemukan. Terdapat beberapa tahap *conquer* yang berada di dalam fungsi `Solve1` dan `Solve2`.

3.2.1.1 Tahap Conquer pada Solve1

Jika terdapat n koin dengan $n = 1$, maka koin tersebut adalah koin palsu. Jika n koin dengan $n = 2$, maka koin tersebut dibagi menjadi dua bagian yaitu A dan B. Lalu A dievaluasi dengan koin yang telah diketahui merupakan koin asli yang disebut id. Jika bobot A dan id tidak sama, maka koin palsu adalah A. Sedangkan jika bobot A dan id sama, maka koin palsu adalah B.

3.2.1.2 Tahap Conquer pada Solve2

Jika terdapat m dan n koin dengan $m = 1$ dan $n = 1$, maka koin m masuk ke bagian A dan koin n masuk ke bagian B. Lalu A dievaluasi dengan id. Jika $A > id$ maka koin palsu adalah A. Jika $A = id$ maka koin palsu adalah B. Jika terdapat m koin dengan $m = 2$ dan tidak ada n koin, maka m dibagi menjadi dua bagian yaitu A dan B. Kemudian A dievaluasi

dengan B. Jika $A > B$ maka koin palsu adalah A. Jika $A < B$ maka koin palsu adalah B. Jika terdapat n koin dengan $n = 2$ dan tidak ada m koin, maka n dibagi menjadi dua bagian yaitu A dan B. Kemudian A dievaluasi dengan B. Jika $A > B$ maka koin palsu adalah B. Jika $A < B$ maka koin palsu adalah A.

3.2.3 Tahap Combine

Pada tahap ini, solusi dari submasalah akan digabungkan menjadi solusi dari masalah. Penggabungan solusi submasalah dilakukan dengan mencetak hasil dari submasalah dengan mengikuti aturan sesuai dengan contoh *output* yang diberikan pada soal. Kemudian *output* yang dicetak menjadi sebuah kesatuan solusi berbentuk *script* yang menjadi solusi dari soal.

3.3 Deskripsi Umum Sistem

Pada subbab ini akan dijelaskan tentang gambaran secara umum dari sistem yang akan dibangun. Pertama sistem akan menerima *input* n yang merepresentasikan banyaknya koin yang akan dievaluasi. Untuk mengetahui banyaknya k penimbangan yang diperlukan, dilakukan komputasi dengan $k = \lceil \log_3(2n + 1) \rceil$.

Setelah mendapat nilai dari n dan k, set koin n $\{1, \dots, n\}$ dibagi menjadi tiga bagian. Tiga bagian tersebut kemudian dievaluasi dengan skala penimbangan. Setelah didapatkan hasil evaluasi, dilanjutkan dengan memanggil fungsi `Solve1` atau `Solve2` sesuai dengan hasil evaluasi.

Function *main*()	
1.	input *n*
2.	*k* = ceil(log3(2*n*+1))
3.	*sub_coin* = divide(*n*, 3)
4.	print *sub_coin*[0] vs *sub_coin*[1]
5.	print case <
6.	solve2(*sub_coin*[1], *sub_coin*[0], *depth* + 1)
7.	print case =
8.	solve1(*sub_coin*[2], *depth*)
9.	print case >
10.	solve2(*sub_coin*[0], *sub_coin*[1], *depth* + 1)

Pseudocode 3.1 Fungsi main

3.3.1 Fungsi Solve1

Fungsi `Solve1` digunakan untuk mengevaluasi lebih lanjut pada sebuah kelompok koin. Fungsi `Solve1` bersifat rekursif yang berarti fungsi akan dipanggil lagi pada tahap selanjutnya ketika diperlukan. Di dalam fungsi `Solve1` terdapat aturan *divide*, *conquer*, dan *combine* yang diperlukan untuk mengevaluasi sebuah kelompok koin. *Divide* dilakukan ketika kelompok koin belum cukup kecil untuk dilakukan pengecekan koin palsu sehingga kelompok koin tersebut dibagi menjadi bagian yang lebih kecil. *Conquer* dilakukan ketika kelompok koin sudah cukup kecil untuk dilakukan pengecekan sehingga koin palsu dapat ditemukan. *Combine* dilakukan pada setiap level rekursi untuk mencetak *output* sesuai dengan aturan yang diberikan pada soal.

Function *solve1*(*coin*, *depth*)	
1.	**if** size(*coin*) == 1
2.	*fake_coin* = *coin*[0]
3.	print *fake_coin*
4.	**if** size(coin) == 2
5.	print genuine coin vs *coin*[0]
6.	print case <
7.	*fake_coin* = *coin*[0]
8.	print *fake_coin*
9.	print case =
10.	*fake_coin* = *coin*[1]
11.	print *fake_coin*
12.	print case >
13.	*fake_coin* = *coin*[0]
14.	print *fake_coin*
15.	*sub_coin* = divide(*coin*, 3)
16.	print *sub_coin*[0] vs *sub_coin*[1]
17.	print case <
18.	solve2(*sub_coin*[1], *sub_coin*[0], *depth*+1)
19.	print case =
20.	solve1(*sub_coin*[2], *depth*+1)
21.	print case >
22.	solve2(*sub_coin*[0], *sub_coin*[1], *depth*+1)

Pseudocode 3.2 Fungsi solve1

3.3.2 Fungsi Solve2

Fungsi `solve2` digunakan untuk mengevaluasi lebih lanjut pada dua kelompok koin. Fungsi `solve2` bersifat rekursif yang berarti fungsi akan dipanggil lagi pada tahap selanjutnya ketika diperlukan. Di dalam fungsi `Solve2`

terdapat aturan *divide*, *conquer*, dan *combine* yang diperlukan untuk mengevaluasi dua kelompok koin. *Divide* dilakukan ketika kelompok koin belum cukup kecil untuk dilakukan pengecekan koin palsu sehingga kelompok koin tersebut dibagi menjadi bagian yang lebih kecil. *Conquer* dilakukan ketika kelompok koin sudah cukup kecil untuk dilakukan pengecekan sehingga koin palsu dapat ditemukan. *Combine* dilakukan pada setiap level rekursi untuk mencetak *output* sesuai dengan aturan yang diberikan pada soal.

Function *solve2*(*coinA*, *coinB*, *depth*)	
1.	**if** size(*coinA*) == 1 and size(*coinB*) == 1
2.	print *coinA*[0] vs genuine coin
3.	print case >
4.	*fake_coin* = *coinA*[0]
5.	print *fake_coin*
6.	print case =
7.	*fake_coin* = *coinB*[0]
8.	print *fake_coin*
9.	**if** size(*coinA*) == 2 and !size(*coinB*)
10.	print *coinA*[0] vs *coinA*[1]
11.	print case >
12.	*fake_coin* = *coinA*[0]
13.	print *fake_coin*
14.	print case <
15.	*fake_coin* = *coinA*[1]
16.	print *fake_coin*
17.	**if** !size(*coinA*) and size(*coinB*) == 2
18.	print *coinB*[0] vs *coinB*[1]
19.	print case >
20.	*fake_coin* = *coinB*[1]

21.	print *fake_coin*
22.	print case <
23.	*fake_coin* = *coinB*[0]
24.	print *fake_coin*
25.	**if** !size(*coinA*)
26.	solve1(*coinB*, *depth*)
27.	**if** !size(*coinB*)
28.	solve1(*coinA*, *depth*)
29.	*sub_coinA* = divide(*coinA*, 3)
30.	*sub_coinB* = divide(*coinB*, 3)
31.	print *sub_coinA*[0] + *sub_coinA*[1] vs *sub_coinB*[0] + *sub_coinB*[1]
32.	print case <
33.	solve2(*sub_coinA*[1], *sub_coinB*[0], *depth*+1)
34.	**if** size(*sub_coinA*[2]) or size(*sub_coinB*[2])
35.	print case =
36.	solve2(*sub_coinA*[2], *sub_coinB*[2], *depth*+1)
37.	print case >
38.	solve2(*sub_coinA*[0], *sub_coinB*[1], *depth*+1)

Pseudocode 3.3 Fungsi solve2

BAB IV
IMPLEMENTASI

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

4.1 Lingkungan Implementasi

Lingkungan implementasi menggunakan sebuah *personal computer* (PC) dengan spesifikasi perangkat lunak dan perangkat keras seperti tercantum pada Tabel 4.1.

Tabel 4.1 Spesifikasi Lingkungan Implementasi

No.	Jenis Perangkat	Spesifikasi
1.	Perangkat Keras	• *Processor* Intel Core i5-7200U CPU @3,1 GHz • *Memory* 8192 RAM
2.	Perangkat Lunak	• Sistem operasi Windows 10 Home Single Language 64-bit • Editor Teks Visual Studio Code 1.45.1 • *Compiler* g++ 7.4.0 • *Windows Subsystem for Linux* Ubuntu 18.04 LTS

4.2 Pendefinisian Library, Preprocessor Directives, Fungsi, dan Variabel Global

Tujuan pendefinisian *preprocessor directives*, fungsi, dan variabel global adalah untuk mempermudah dan

mempersingkat implementasi. Implementasi *library*, *preprocessor directive*, fungsi, dan variabel global ditunjukkan pada Kode Sumber 4.1.

```
1.  #include <iostream>
2.  #include <cstdio>
3.  #include <vector>
4.
5.  #define pb(x) push_back(x)
6.  #define SZ(x) x.size()
7.  #define FOR0(i, x) for (i = 0; i < x; i++)
8.  #define FOR1(i, x) for (i = 1; i <= x; i++)
9.  #define FOR(i, a, b) for (i = a; i <= b; i++)
10. #define type vector<int>
11. using namespace std;
12.
13. void solve1(type, int);
14. void solve2(type, type, int);
15.
16. int n, id;
```

Kode Sumber 4.1 Pendefinisian library, preprocessor directives, fungsi, dan variabel global

4.3 Fungsi Input dan Output

Fungsi ini mengatur *input* dan *output* kode sumber. Fungsi *input* dan *output* ditunjukkan pada Kode Sumber 4.2.

```
1.  void RD(int &x) { scanf("%d", &x); }
2.
3.  void printSet(type p)
4.  {
5.      printf("%d", p[0]);
6.      int i;
7.      FOR1(i, SZ(p) - 1)
8.      printf("+%d", p[i]);
9.  }
10.
11. void print(int cnt)
12. {
13.     while (cnt--)
14.         putchar(' ');
15. }
```

Kode Sumber 4.2 Fungsi input dan output

4.4 Fungsi Divide

Fungsi `divide` dideklarasikan untuk mempermudah operasi pembagian kelompok koin menjadi kelompok yang lebih kecil. Fungsi `divide` ditunjukkan pada Kode Sumber 4.3.

```
1.  void divide(type a, type b[3])
2.  {
3.      int n = SZ(a), k = n / 3, i, j;
4.      FOR0(i, 3)
5.      FOR0(j, k) b[i].pb(a[i * k + j]);
6.      if (n % 3 == 1)
7.          b[2].pb(a[n - 1]);
8.      else if (n % 3 == 2)
9.          b[0].pb(a[n - 2]), b[1].pb(a[n - 1]);
10. }
```

Kode Sumber 4.3 Fungsi divide

4.5 Fungsi Main

Fungsi `main` adalah fungsi utama dari kode sumber. Implementasi fungsi main ditunjukkan pada Kode Sumber 4.4.

```
1.  int main()
2.  {
3.      RD(n);
4.      int ans = 0, i, j, k = n / 3;
5.      for (i = 1; i < n * 2; i *= 3, ans++);
6.      printf("need %d weighings\n", ans);
7.      type s[3];
8.      FOR0(i, 3)
9.      FOR1(j, k) s[i].pb(i * k + j);
10.     if (n % 3 == 1)
11.         s[2].pb(n);
12.     else if (n % 3 == 2)
13.         s[0].pb(n - 1), s[1].pb(n);
14.     printf("weigh ");
15.     printSet(s[0]);
16.     printf(" vs ");
17.     printSet(s[1]);
18.     printf("\n");
19.     printf("case <:\n");
20.     id = s[2][0];
21.     solve2(s[1], s[0], 1);
22.     printf("case =:\n");
23.     id = s[0][0];
24.     solve1(s[2], 1);
25.     printf("case >:\n");
26.     id = s[2][0]
27.     solve2(s[0], s[1], 1);
28.     printf("end\n");
29.     return 0;
30. }
```

Kode Sumber 4.4 Fungsi main

4.6 Fungsi Solve1

Fungsi `solve1` adalah fungsi yang digunakan untuk mengevaluasi sebuah kelompok koin. Implementasi fungsi `solve1` ditunjukkan pada Kode Sumber 4.5.

```
1.  void solve1(type a, int dep)
2.  {
3.      if (SZ(a) == 1)
4.      {
5.          print(dep);
6.          printf("fake %d\n", a[0]);
7.          return
8.      }
9.      if (SZ(a) == 2)
10.     {
11.         print(dep);
12.         printf("weigh %d vs %d\n", id, a[0]);
13.         print(dep);
14.         print("case <:\n");
15.         print(dep);
16.         printf(" fake %d\n", a[0]);
17.         print(dep);
18.         printf("case =:\n");
19.         print(dep);
20.         printf(" fake %d\n", a[1]);
21.         print(dep);
22.         printf("case >:\n");
23.         print(dep);
24.         printf(" fake %d\n", a[0]);
25.         print(dep);
26.         print("end\n");
27.         return;
28.     }
29.     type son[3];
30.     int n = SZ(a), k = n / 3, i, j;
31.     FOR0(i, 3)
32.     FOR0(j, k) son[i].pb(a[i * k + j]);
33.     if (n % 3)
34.         son[0].pb(id), son[1].pb(a[n - 1]);
35.     if (n % 3 == 2)
```

```
36.             son[2].pb(a[n – 2]);
37.         print(dep);
38.         printf(“weigh ”);
39.         printSet(son[0]);
40.         printf(“ vs ”);
41.         printSet(son[1]);
42.         printf(“\n”);
43.         if (n % 3)
44.             son[0].pop_back();
45.         print(dep);
46.         printf(“case <:\n”);
47.         solve2(son[1], son[0], dep + 1);
48.         print(dep);
49.         printf(“case =:\n”);
50.         solve1(son[2], dep + 1);
51.         print(dep);
52.         printf(“case >:\n”);
53.         solve2(son[0], son[1], dep + 1);
54.         print(dep);
55.         printf(“end\n”);
56. }
```

Kode Sumber 4.5 Fungsi solve1

4.7 Fungsi Solve2

Fungsi solve2 adalah fungsi yang digunakan untuk mengevaluasi dua kelompok koin. Implementasi fungsi solve2 ditunjukkan pada Kode Sumber 4.6.

```
void solve2(type a, type b, int dep)
{
     if (SZ(a) == 1 && SZ(b) == 1)
     {
          print(dep);
          printf("weigh %d vs %d\n", a[0], id);
          print(dep);
          print("case >:\n");
          print(dep);
          printf(" fake %d\n", a[0]);
          print(dep);
          printf("case =:\n");
          print(dep)
          printf(" fake %d\n", b[0]);
          print(dep);
          printf("end\n");
          return;
     }
     if (SZ(a) == 2 && !SZ(b))
     {
          print(dep);
          printf("weigh %d vs %d\n", a[0], a[1]);
          print(dep);
          printf("case >:\n");
          print(dep);
          printf(" fake %d\n", a[0]);
          print(dep);
          printf("case <:\n");
          print(dep);
          printf(" fake %d\n", a[1]);
          print(dep);
          printf("end\n");
          return;
     }
     if (!SZ(a) && SZ(b) == 2)
     {
          print(dep);
          printf("weigh %d vs %d\n", b[0], b[1]);
          print(dep);
          printf("case >:\n");
```

```
41.             print(dep);
42.             printf(“ fake %d\n”, b[1]);
43.             print(dep);
44.             printf(“case <:\n”);
45.             print(dep);
46.             printf(“ fake %d\n”, b[0]);
47.             print(dep);
48.             printf(“end\n”);
49.             return;
50.         }
51.         if (!SZ(a))
52.         {
53.             solve1(b, dep);
54.             return;
55.         }
56.         if (!SZ(b))
57.         {
58.             solve1(a, dep);
59.             return;
60.         }
61.         type sonA[3], sonB[3];
62.         divide(a, sonA);
63.         divide(b, sonB);
64.         if (SZ(a) % 3 == 1 && SZ(b) % 3 == 1)
65.         {
66.             sonB[0].pb(*(sonB[2].end() - 1));
67.             sonB[2].pop_back();
68.             sonB[1].pb(*(sonB[2].end() - 1));
69.             sonB[2].pop_back();
70.         }
71.         print(dep);
72.         printf(“weigh ”);
73.         if (SZ(sonA[0]))
74.         {
75.             printSet(sonA[0]);
76.             if (SZ(sonB[0]))
77.                 printf(“+”);
78.         }
79.         if (SZ(sonB[0]))
80.             printSet(sonB[0]);
81.         printf(“ vs ”);
82.         if (SZ(sonA[1]))
```

```
83.         {
84.              printSet(sonA[1]);
85.              if (SZ(sonB[1]))
86.                   printf("+");
87.         }
88.         if (SZ(sonB[1]))
89.              printSet(sonB[1]);
90.         printf("\n");
91.         print(dep);
92.         printf("case <:\n");
93.         solve2(sonA[1], sonB[0], dep + 1);
94.         if (SZ(sonA[2]) || SZ(sonB[2]))
95.         {
96.              print(dep);
97.              printf("case =:\n");
98.              solve2(sonA[2], sonB[2], dep + 1);
99.         }
100.        print(dep);
101.        printf("case >:\n");
102.        solve2(sonA[0], sonB[1], dep + 1);
103.        print(dep);
104.        printf("end\n");
105.   }
```

Kode Sumber 4.6 Fungsi solve2

[*Halaman ini sengaja dikosongkan*]

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan mengenai hasil uji coba program yang telah dirancang pada tugas akhir ini. Uji coba dilakukan untuk mengetahui kinerja algoritma dengan lingkungan uji coba yang telah ditentukan.

5.1 Lingkungan Uji Coba

Lingkungan uji coba pada tugas akhir ini adalah sebuah *personal computer* (PC) dengan sistem operasi Windows 10 Home Single Language 64-bit. PC yang digunakan memiliki spesifikasi *Processor* Intel Core i5-7200U dengan kecepatan 3,1 GHz, *Random Access Memory* (RAM) sebesar 8 GB.

Pada sisi perangkat lunak, uji coba pada Tugas Akhir ini dilakukan dengan menggunakan bahasa pemrograman C++ dilengkapi dengan *library* dan menggunakan editor teks Visual Studio Code 1.45.1 dengan *compiler* g++ 7.4.0 yang berjalan di dalam *Windows Subsystem for Linux* (WSL) Ubuntu 18.04 LTS.

5.2 Uji Coba Kebenaran

Pada subbab ini akan dijelaskan uji coba kebenaran yang dilakukan untuk menguji implementasi yang telah dibuat untuk menyelesaikan Tugas Akhir ini.

5.2.1 Uji Coba Kebenaran Lokal

Uji coba kebenaran lokal dilakukan dengan menguji implementasi pada lingkungan uji coba lokal. Kasus yang akan digunakan sebagai bahan uji coba kebenaran lokal adalah

contoh kasus uji yang diberikan *Codeforces*. Hasil uji coba lokal dapat dilihat pada Gambar 5.1.

```
Input:
4

Output:
need 2 weighings
weigh 1 vs 2
case <:
 weigh 2 vs 3
 case >:
   fake 2
 case =:
   fake 1
 end
case =:
 weigh 1 vs 3
 case <:
   fake 3
 case =:
   fake 4
 case >:
   fake 3
 end
case >:
 weigh 1 vs 3
 case >:
   fake 1
 case =:
   fake 2
 end
end
```

Gambar 5.1 Hasil uji coba lokal pada 4 koin

Dari hasil uji coba lokal dapat dilihat pada gambar 5.1 bahwa *input* sebuah *integer* mengeluarkan *output* berbentuk *script*. *Input* yang diberikan adalah 4 koin. Dari *output* didapatkan informasi bahwa penimbangan maksimal yang diperlukan untuk mengidentifikasi koin palsu di dalam 4 koin adalah 2 kali penimbangan. Penjabaran rute koin palsu yang dibentuk dapat dilihat pada Tabel 5.1.

Tabel 5.1 Penjabaran rute koin palsu.

W1	Case1	W2	Case2	Fake
1 vs 2	<	2 vs 3	>	2
1 vs 2	<	2 vs 3	=	1
1 vs 2	=	1 vs 3	<	3
1 vs 2	=	1 vs 3	=	4
1 vs 2	=	1 vs 3	>	3
1 vs 2	>	1 vs 3	>	1
1 vs 2	>	1 vs 3	=	2

Dari Tabel 5.1 didapatkan informasi sebagai berikut:

1. Jika penimbangan pertama (1 vs 2) didapati bahwa 1 < 2 dan pada penimbangan kedua (2 vs 3) didapati bahwa 2 > 3, maka koin palsu adalah 2.
2. Jika penimbangan pertama (1 vs 2) didapati bahwa 1 < 2 dan pada penimbangan kedua (2 vs 3) didapati bahwa 2 = 3, maka koin palsu adalah 1.
3. Jika penimbangan pertama (1 vs 2) didapati bahwa 1 = 2 dan pada penimbangan kedua (1 vs 3) didapati bahwa 1 < 3, maka koin palsu adalah 3.
4. Jika penimbangan pertama (1 vs 2) didapati bahwa 1 = 2 dan pada penimbangan kedua (1 vs 3) didapati bahwa 1 = 3, maka koin palsu adalah 4.

5. Jika penimbangan pertama (1 vs 2) didapat bahwa 1 = 2 dan pada penimbangan kedua (1 vs 3) didapati bahwa 1 > 3, maka koin palsu adalah 3.
6. Jika penimbangan pertama (1 vs 2) didapati bahwa 1 > 2 dan pada penimbangan kedua (1 vs 3) didapati bahwa 1 > 3, maka koin palsu adalah 1.
7. Jika penimbangan pertama (1 vs 2) didapati bahwa 1 > 2 dan pada penimbangan kedua (1 vs 3) didapati bahwa 1 = 3, maka koin palsu adalah 2.
8. Terdapat 2 rute yang tidak mungkin pada hasil pengujian. Rute pertama yang tidak mungkin yaitu ketika penimbangan pertama (1 vs 2) didapati bahwa 1 < 2 dan pada penimbangan kedua (2 vs 3) didapati bahwa 2 > 3, dan rute kedua yang tidak mungkin yaitu ketika penimbangan pertama (1 vs 2) didapati bahwa 1 > 2 dan pada penimbangan kedua (2 vs 3) didapati bahwa 2 < 3.

Selain pengujian menggunakan kasus uji yang diberikan di dalam soal dimana dari kasus uji tersebut diketahui $n\ mod\ 3 == 1$ dan penimbangan yang dibutuhkan yaitu 2, dilakukan pengujian lain untuk mengetahui apakah program dapat menghasilkan rute koin palsu dimana penimbangan yang dibutuhkan sebanyak 3, 4 dan 5 dan pada kondisi $n\ mod\ 3 == 0$ dan $n\ mod\ 3 == 2$. Koin yang akan digunakan sebagai *input* adalah 12 pada kasus uji kedua dimana $n\ mod\ 3 == 0$ dan penimbangan sebanyak 3, 38 pada kasus uji ketiga dimana $n\ mod\ 3 == 2$ dan penimbangan sebanyak 4, dan 100 pada kasus uji keempat dimana penimbangan sebanyak 5 dan merupakan batas atas dari koin pada soal. Hasil pengujian dengan kasus uji kedua, ketiga, dan keempat dapat dilihat pada Gambar 5.2.

```
need 3 weighings
weigh 1+2+3+4 vs 5+6+7+8
case <:
 weigh 5+1+4 vs 6+2+3
 case <:
  weigh 1 vs 4
  case <:
   fake 1
  case =:
   fake 6
  case >:
   fake 4
  end
 case =:
  weigh 7 vs 8
  case >:
    fake 7
  case <:
    fake 8
  end
 case >:
  weigh 2 vs 3
  case <:
   fake 2
  case =:
   fake 5
  case >:
   fake 3
  end
 end
```

Gambar 5.2.1 Hasil uji coba lokal pada 12 koin bagian 1

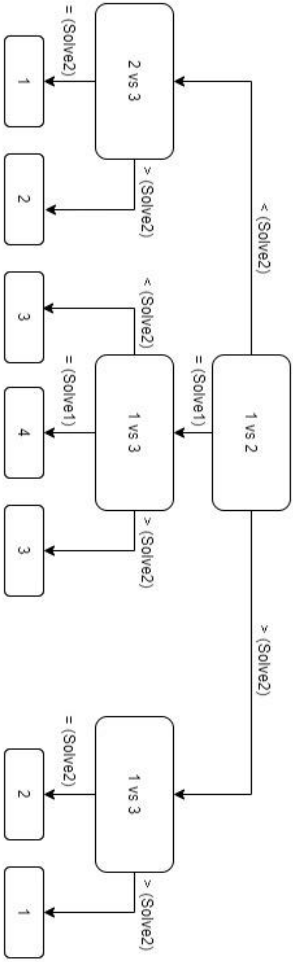
```
case =:
 weigh 9+1 vs 10+12
 case <:
  weigh 10 vs 12
  case <:
   fake 12
  case =:
   fake 9
  case >:
   fake 10
  end
 case =:
  fake 11
 case >:
  weigh 10 vs 12
  case <:
   fake 10
  case =:
   fake 9
  case >:
   fake 12
  end
 end
case >:
 weigh 1+5+8 vs 2+6+7
 case <:
  weigh 5 vs 8
  case <:
   fake 5
  case =:
```

Gambar 5.2.2 Hasil uji coba lokal pada 12 koin bagian 2

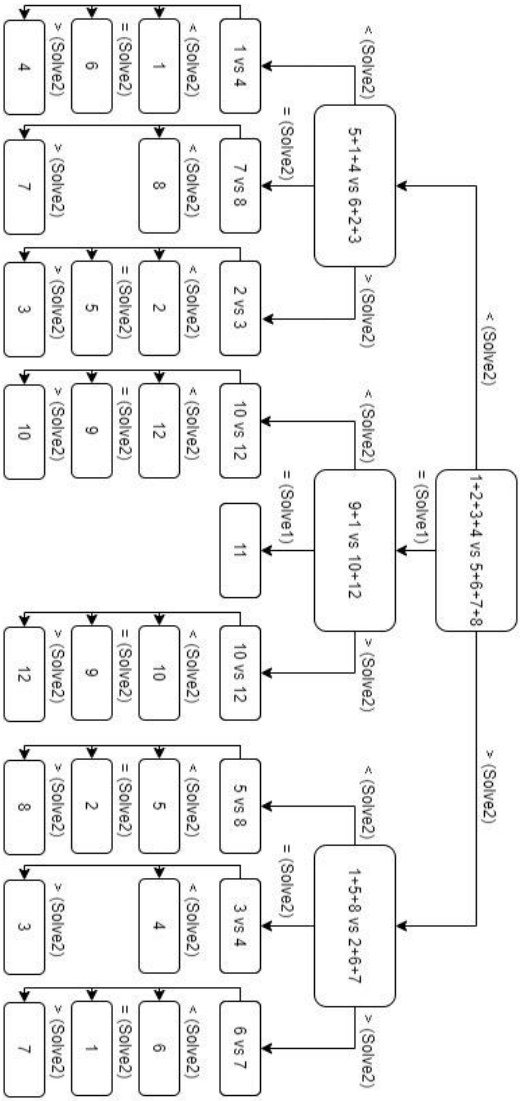
```
  fake 2
 case >:
  fake 8
 end
case =:
 weigh 3 vs 4
 case >:
   fake 3
 case <:
   fake 4
 end
case >:
 weigh 6 vs 7
 case <:
  fake 6
 case =:
  fake 1
 case >:
  fake 7
 end
end
end
```

Gambar 5.2.3 Hasil uji coba lokal pada 12 koin bagian 3

Ilustrasi hasil uji coba lokal pada 4 koin dan 12 koin dapat dilihat seperti pada Gambar 5.3 dan Gambar 5.4.



Gambar 5.3 Ilustrasi hasil uji coba pada 4 koin



Gambar 5.4 Ilustrasi hasil uji coba pada 12 koin

5.2.2 Uji Coba dengan Algoritma Pembanding

Uji coba dengan algoritma pembanding dilakukan dengan membandingkan algoritma *divide and conquer* dengan algoritma sistem penomoran terner. Algoritma sistem penomoran terner menyelesaikan permasalahan dengan membentuk matriks yang tersusun atas bilangan basis tiga. Setiap vektor kolom tersusun atas susunan bilangan basis tiga yang unik. Banyaknya baris pada matriks yaitu banyaknya penimbangan yang dibutuhkan dan banyaknya kolom pada matriks yaitu banyaknya koin yang akan diujikan. Setiap vektor kolom mewakili sebuah koin dan invers dari vektor kolom tersebut juga mewakili koin yang sama namun dengan bobot yang berbeda. Untuk mengetahui rute koin palsu perlu dilakukan *precompute* untuk membentuk matriks pada setiap banyaknya koin yang diujikan. Pada Tabel 5.2 ditunjukkan hasil perbandingan waktu eksekusi program dalam satuan detik antara algoritma *divide and conquer* dan algoritma sistem penomoran terner.

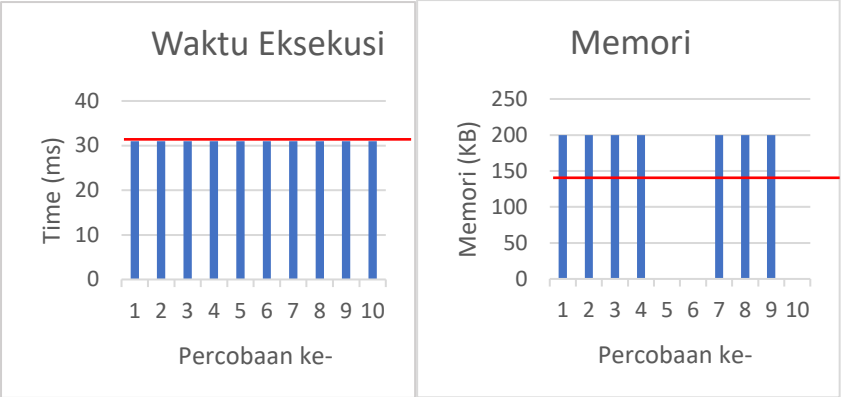
Tabel 5.2 Hasil pengujian dengan algoritma pembanding

Jumlah Koin	Divide and Conquer	Sistem Penomoran Terner
4	0,000351	0,182429
12	0,000970	0,182889
38	0,003561	0,197450
100	0,008087	0,246165

Pada tabel hasil pengujian dengan algoritma pembanding dapat dilihat bahwa algoritma *divide and conquer* lebih cepat dibanding dengan algoritma sistem penomoran terner. Hal ini karena proses *precompute* pada algoritma sistem penomoran terner membutuhkan waktu yang lama.

5.2.3 Uji Coba Kebenaran pada Codeforces

Uji coba kebenaran program dilakukan dengan cara mengirimkan kode sumber ke situs penilaian *Codeforces*. Sistem pada situs penilaian tersebut akan menjalankan program yang dikirim oleh pengguna dan hasilnya kemudian akan diuji di dalam server situs dengan sistem juri yang telah disediakan oleh pembuat soal. Selain itu juga dilakukan percobaan pengujian 10 kali untuk mendapatkan rata-rata waktu eksekusi dan memori. Grafik waktu eksekusi dan memori hasil uji coba pada situs *Codeforces* dapat dilihat pada Gambar 5.5. Garis merah menandakan rata-rata dari waktu eksekusi dan memori. Dari grafik waktu eksekusi dapat dilihat bahwa untuk waktu eksekusi selama 10 kali percobaan selalu stabil di angka 31ms. Sehingga rata-rata waktu eksekusi adalah 31 ms. Sedangkan di grafik memori dapat dilihat hasil uji coba menggunakan memori sebesar 200 KB kecuali pada uji coba kelima, keenam, dan kesepuluh yaitu sebesar 0 KB. Sehinga rata-rata memori adalah 140KB. Hasil memori yang didapatkan yaitu berasal dari pembulatan yang didapatkan pada situs *Codeforces* dan tidak dapat diketahui detail memori yang digunakan. Sedangkan hasil dari uji coba kebenaran pada situs daring *Codeforces* dapat dilihat pada Gambar 5.6.



Gambar 5.5 Grafik waktu eksekusi dan memori dari hasil uji coba pada situs Codeforces

#	When	Who	Problem	Lang	Verdict	Time	Memory
81870869	May/29/2020 16:48 UTC+7	fwildanf	306 - Balance	GNU C++11	Accepted	31 ms	0 KB
81870827	May/29/2020 16:48 UTC+7	fwildanf	306 - Balance	GNU C++11	Accepted	31 ms	200 KB
81870795	May/29/2020 16:48 UTC+7	fwildanf	306 - Balance	GNU C++11	Accepted	31 ms	200 KB
81870755	May/29/2020 16:47 UTC+7	fwildanf	306 - Balance	GNU C++11	Accepted	31 ms	200 KB
81870703	May/29/2020 16:47 UTC+7	fwildanf	306 - Balance	GNU C++11	Accepted	31 ms	0 KB
81870666	May/29/2020 16:46 UTC+7	fwildanf	306 - Balance	GNU C++11	Accepted	31 ms	0 KB
81870620	May/29/2020 16:45 UTC+7	fwildanf	306 - Balance	GNU C++11	Accepted	31 ms	200 KB
81866271	May/29/2020 15:52 UTC+7	fwildanf	306 - Balance	GNU C++11	Accepted	31 ms	200 KB
81866195	May/29/2020 15:51 UTC+7	fwildanf	306 - Balance	GNU C++11	Accepted	31 ms	200 KB
81865953	May/29/2020 15:48 UTC+7	fwildanf	306 - Balance	GNU C++11	Accepted	31 ms	200 KB

Gambar 5.6 Uji coba kebenaran pada Codeforces

[*Halaman ini sengaja dikosongkan*]

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Dalam pengerjaan Tugas Akhir ini setelah melalui tahap desain, implementasi, dan uji coba, diperoleh kesimpulan sebagai berikut:

1. Desain yang digunakan adalah simulasi penimbangan menggunakan pendekatan *Divide and Conquer*.
2. Pada implementasi algoritma simulasi penimbangan dengan pendekatan *Divide and Conquer*, seluruh kemungkinan koin palsu dapat dipetakan.
3. Dari hasil uji coba kebenaran pada situs *Codeforces* yang dilakukan sebanyak 10 kali, disimpulkan bahwa rata-rata waktu eksekusi yang dibutuhkan adalah 31 milidetik dan rata-rata memori yang dibutuhkan adalah 140KB. Dengan begitu, metode *Divide* and *Conquer* terbukti dapat menyelesaikan permasalahan *Codeforces* 306 – Balance dengan baik dan efisien.

6.2 Saran

Pada Tugas Akhir ini tidak lepas dari kekurangan serta hal-hal penting yang bisa menjadi pertimbangan nantinya. Berikut saran-saran yang dapat diambil dari tugas akhir ini:

1. Implementasi yang telah dilakukan dapat dikembangkan untuk menampilkan bobot dari koin palsu dengan menyimpan informasi bobot. Namun dalam beberapa kasus terdapat koin yang menyimpan dua bobot sehingga perlu dibandingkan dengan koin asli agar didapatkan bobot sebenarnya dari koin palsu. Perbandingan baru ini mempengaruhi jumlah

penimbangan sehingga jika dibandingkan dengan implementasi saat ini jumlah penimbangan yang dibutuhkan bisa berbeda. Jumlah penimbangan yang berbeda ini didapat pada koin n dengan penimbangan k dimana $n = 3^k - 1$.

2. Metode *Divide and Conquer* dapat menjadi bahan riset untuk mencari optimasi lebih lanjut.

DAFTAR PUSTAKA

[1] Mitrichev, Petr. “Problem - 99999306.” Codeforces, ITMO University, 2018, codeforces.com/problemsets/acmsguru/problem/99999/306.

[2] Cormen, Thomas H., and Charles E. Leiserson. Introduction to Algorithms, 3rd Edition. The MIT Press, 2009.

[3] Chudnov, Alexander M. “Weighing Algorithms of Classification and Identification of Situations.” Discrete Mathematics and Applications, vol. 25, no. 2, 2015, doi:10.1515/dma-2015-0007.

[4] Dominguez-Montes, Juan. “Solution to the Counterfeit Coin Problem and its Generalization.” arXiv:1005.1391 [cs, math], Mei 2010. arXiv.org, http://arxiv.org/abs/1005.1391.

[5] Halbeisen, Lorenz, dan Norbert Hungerbühler. “The General Counterfeit Coin Problem.” Discrete Mathematics, vol. 147, no. 1–3, Desember 1995, hlm. 139–50. DOI.org (Crossref), doi:10.1016/0012-365X(94)00232-8.

[*Halaman ini sengaja dikosongkan*]

LAMPIRAN A: HASIL UJI COBA LOKAL PADA 38 KOIN

```
need 4 weighings
weigh 1+2+3+4+5+6+7+8+9+10+11+12+37 vs
13+14+15+16+17+18+19+20+21+22+23+24+38
case <:
 weigh 13+14+15+16+1+2+3+4+37 vs
17+18+19+20+5+6+7+8+12
 case <:
  weigh 17+1+4 vs 18+2+37
  case <:
   weigh 1 vs 4
   case <:
    fake 1
   case =:
    fake 18
   case >:
    fake 4
   end
  case =:
   weigh 19 vs 20
   case <:
    fake 20
   case =:
    fake 3
   case >:
    fake 19
   end
  case >:
   weigh 2 vs 37
   case <:
    fake 2
```

```
  case =:
   fake 17
  case >:
   fake 37
  end
 end
case =:
 weigh 21+24+9 vs 22+38+10
 case <:
  weigh 22 vs 38
  case <:
   fake 38
  case =:
   fake 9
  case >:
   fake 22
  end
 case =:
  weigh 23 vs 25
  case >:
    fake 23
  case =:
    fake 11
  end
 case >:
  weigh 21 vs 24
  case <:
   fake 24
  case =:
   fake 10
```

```
  case >:
   fake 21
  end
 end
case >:
 weigh 13+5+8 vs 14+6+12
 case <:
  weigh 5 vs 8
  case <:
   fake 5
  case =:
   fake 14
  case >:
   fake 8
  end
 case =:
  weigh 15 vs 16
  case <:
   fake 16
  case =:
   fake 7
  case >:
   fake 15
  end
 case >:
  weigh 6 vs 12
  case <:
   fake 6
  case =:
   fake 13
```

```
    case >:
     fake 12
    end
   end
  end
 case =:
  weigh 25+26+27+28 vs 29+30+31+32
  case <:
   weigh 29+25+28 vs 30+26+27
   case <:
    weigh 25 vs 28
    case <:
     fake 25
    case =:
     fake 30
    case >:
     fake 28
    end
   case =:
    weigh 31 vs 32
    case >:
      fake 31
    case <:
      fake 32
    end
   case >:
    weigh 26 vs 27
    case <:
     fake 26
    case =:
```

```
    fake 29
   case >:
    fake 27
   end
  end
 case =:
  weigh 33+1 vs 34+36
  case <:
   weigh 34 vs 36
   case <:
    fake 36
   case =:
    fake 33
   case >:
    fake 34
   end
  case =:
   fake 35
  case >:
   weigh 34 vs 36
   case <:
    fake 34
   case =:
    fake 33
   case >:
    fake 36
   end
  end
 case >:
  weigh 25+29+32 vs 26+30+31
```

```
  case <:
   weigh 29 vs 32
   case <:
    fake 29
   case =:
    fake 26
   case >:
    fake 32
   end
  case =:
   weigh 27 vs 28
   case >:
     fake 27
   case <:
     fake 28
   end
  case >:
   weigh 30 vs 31
   case <:
    fake 30
   case =:
    fake 25
   case >:
    fake 31
   end
  end
 end
case >:
 weigh 1+2+3+4+13+14+15+16+38 vs
5+6+7+8+17+18+19+20+24
```

```
case <:
  weigh 5+13+16 vs 6+14+38
  case <:
   weigh 13 vs 16
   case <:
    fake 13
   case =:
    fake 6
   case >:
    fake 16
   end
  case =:
   weigh 7 vs 8
   case <:
    fake 8
   case =:
    fake 15
   case >:
    fake 7
   end
  case >:
   weigh 14 vs 38
   case <:
    fake 14
   case =:
    fake 5
   case >:
    fake 38
   end
  end
```

```
case =:
  weigh 9+12+21 vs 10+37+22
  case <:
   weigh 10 vs 37
   case <:
    fake 37
   case =:
    fake 21
   case >:
    fake 10
   end
  case =:
   weigh 11 vs 25
   case >:
     fake 11
   case =:
     fake 23
   end
  case >:
   weigh 9 vs 12
   case <:
    fake 12
   case =:
    fake 22
   case >:
    fake 9
   end
  end
 case >:
  weigh 1+17+20 vs 2+18+24
```

```
  case <:
   weigh 17 vs 20
   case <:
    fake 17
   case =:
    fake 2
   case >:
    fake 20
   end
  case =:
   weigh 3 vs 4
   case <:
    fake 4
   case =:
    fake 19
   case >:
    fake 3
   end
  case >:
   weigh 18 vs 24
   case <:
    fake 18
   case =:
    fake 1
   case >:
    fake 24
   end
  end
 end
end
```

LAMPIRAN B: HASIL UJI COBA LOKAL PADA 100 KOIN

```
need 5 weighings
weigh
1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16+17+18+19+2
0+21+22+23+24+25+26+27+28+29+30+31+32+33 vs
34+35+36+37+38+39+40+41+42+43+44+45+46+47+48+49+5
0+51+52+53+54+55+56+57+58+59+60+61+62+63+64+65+66
case <:
 weigh
34+35+36+37+38+39+40+41+42+43+44+1+2+3+4+5+6+7+8+
9+10+11 vs
45+46+47+48+49+50+51+52+53+54+55+12+13+14+15+16+1
7+18+19+20+21+22
 case <:
  weigh 45+46+47+54+1+2+3+10 vs 48+49+50+55+4+5+6+11
  case <:
   weigh 48+1+10 vs 49+2+3
   case <:
    weigh 1 vs 10
    case <:
     fake 1
    case =:
     fake 49
    case >:
     fake 10
    end
   case =:
    weigh 50 vs 55
    case >:
      fake 50
    case <:
```

```
    fake 55
   end
  case >:
   weigh 2 vs 3
   case <:
    fake 2
   case =:
    fake 48
   case >:
    fake 3
   end
  end
 case =:
  weigh 51+7 vs 52+8
  case <:
   weigh 52 vs 67
   case >:
    fake 52
   case =:
    fake 7
   end
  case =:
   weigh 53 vs 67
   case >:
    fake 53
   case =:
    fake 9
   end
  case >:
   weigh 51 vs 67
```

```
    case >:
      fake 51
    case =:
      fake 8
    end
  end
 case >:
  weigh 45+4+11 vs 46+5+6
  case <:
   weigh 4 vs 11
   case <:
    fake 4
   case =:
    fake 46
   case >:
    fake 11
   end
  case =:
   weigh 47 vs 54
   case >:
     fake 47
   case <:
     fake 54
   end
  case >:
   weigh 5 vs 6
   case <:
    fake 5
   case =:
    fake 45
```

```
    case >:
     fake 6
    end
   end
  end
 case =:
  weigh 56+57+58+65+23+24+25+32 vs
59+60+61+66+26+27+28+33
  case <:
   weigh 59+23+32 vs 60+24+25
   case <:
    weigh 23 vs 32
    case <:
     fake 23
    case =:
     fake 60
    case >:
     fake 32
    end
   case =:
    weigh 61 vs 66
    case >:
      fake 61
    case <:
      fake 66
    end
   case >:
    weigh 24 vs 25
    case <:
     fake 24
```

```
    case =:
     fake 59
    case >:
     fake 25
    end
   end
  case =:
   weigh 62+29 vs 63+30
   case <:
    weigh 63 vs 67
    case >:
      fake 63
    case =:
      fake 29
    end
   case =:
    weigh 64 vs 67
    case >:
      fake 64
    case =:
      fake 31
    end
   case >:
    weigh 62 vs 67
    case >:
      fake 62
    case =:
      fake 30
    end
   end
```

```
 case >:
  weigh 56+26+33 vs 57+27+28
  case <:
   weigh 26 vs 33
   case <:
    fake 26
   case =:
    fake 57
   case >:
    fake 33
   end
  case =:
   weigh 58 vs 65
   case >:
     fake 58
   case <:
     fake 65
   end
  case >:
   weigh 27 vs 28
   case <:
    fake 27
   case =:
    fake 56
   case >:
    fake 28
   end
  end
 end
case >:
```

```
 weigh 34+35+36+43+12+13+14+21 vs
37+38+39+44+15+16+17+22
 case <:
  weigh 37+12+21 vs 38+13+14
  case <:
   weigh 12 vs 21
   case <:
    fake 12
   case =:
    fake 38
   case >:
    fake 21
   end
  case =:
   weigh 39 vs 44
   case >:
     fake 39
   case <:
     fake 44
   end
  case >:
   weigh 13 vs 14
   case <:
    fake 13
   case =:
    fake 37
   case >:
    fake 14
   end
  end
```

```
case =:
 weigh 40+18 vs 41+19
 case <:
  weigh 41 vs 67
  case >:
    fake 41
  case =:
    fake 18
  end
 case =:
  weigh 42 vs 67
  case >:
    fake 42
  case =:
    fake 20
  end
 case >:
  weigh 40 vs 67
  case >:
    fake 40
  case =:
    fake 19
  end
 end
case >:
 weigh 34+15+22 vs 35+16+17
 case <:
  weigh 15 vs 22
  case <:
   fake 15
```

```
    case =:
     fake 35
    case >:
     fake 22
    end
   case =:
    weigh 36 vs 43
    case >:
      fake 36
    case <:
      fake 43
    end
   case >:
    weigh 16 vs 17
    case <:
     fake 16
    case =:
     fake 34
    case >:
     fake 17
    end
   end
  end
 end
case =:
 weigh 67+68+69+70+71+72+73+74+75+76+77+1 vs
78+79+80+81+82+83+84+85+86+87+88+100
 case <:
  weigh 78+79+80+81+67+68+69+76 vs
82+83+84+85+70+71+72+77
```

```
case <:
 weigh 82+67+76 vs 83+68+69
 case <:
  weigh 67 vs 76
  case <:
   fake 67
  case =:
   fake 83
  case >:
   fake 76
  end
 case =:
  weigh 84 vs 85
  case >:
    fake 84
  case <:
    fake 85
  end
 case >:
  weigh 68 vs 69
  case <:
   fake 68
  case =:
   fake 82
  case >:
   fake 69
  end
 end
case =:
 weigh 86+73 vs 87+74
```

```
  case <:
   weigh 87 vs 1
   case >:
     fake 87
   case =:
     fake 73
   end
  case =:
   weigh 88 vs 100
   case <:
    fake 100
   case =:
    fake 75
   case >:
    fake 88
   end
  case >:
   weigh 86 vs 1
   case >:
     fake 86
   case =:
     fake 74
   end
  end
 case >:
  weigh 78+70+77 vs 79+71+72
  case <:
   weigh 70 vs 77
   case <:
    fake 70
```

```
    case =:
     fake 79
    case >:
     fake 77
    end
   case =:
    weigh 80 vs 81
    case >:
      fake 80
    case <:
      fake 81
    end
   case >:
    weigh 71 vs 72
    case <:
     fake 71
    case =:
     fake 78
    case >:
     fake 72
    end
   end
  end
 case =:
  weigh 89+90+91+1 vs 92+93+94+99
  case <:
   weigh 92+89 vs 93+90
   case <:
    weigh 93 vs 1
    case >:
```

```
    fake 93
   case =:
    fake 89
   end
  case =:
   weigh 94 vs 99
   case <:
    fake 99
   case =:
    fake 91
   case >:
    fake 94
   end
  case >:
   weigh 92 vs 1
   case >:
    fake 92
   case =:
    fake 90
   end
  end
 case =:
  weigh 95+1 vs 96+98
  case <:
   weigh 96 vs 98
   case <:
    fake 98
   case =:
    fake 95
   case >:
```

```
  fake 96
 end
case =:
 fake 97
case >:
 weigh 96 vs 98
 case <:
  fake 96
 case =:
  fake 95
 case >:
  fake 98
 end
end
case >:
 weigh 89+92 vs 90+93
 case <:
  weigh 90 vs 1
  case >:
   fake 90
  case =:
   fake 92
  end
 case =:
  weigh 94 vs 99
  case <:
   fake 94
  case =:
   fake 91
  case >:
```

```
    fake 99
   end
  case >:
   weigh 89 vs 1
   case >:
     fake 89
   case =:
     fake 93
   end
  end
 end
case >:
 weigh 67+68+69+76+78+79+80+81 vs
70+71+72+77+82+83+84+85
 case <:
  weigh 70+78+81 vs 71+79+80
  case <:
   weigh 78 vs 81
   case <:
    fake 78
   case =:
    fake 71
   case >:
    fake 81
   end
  case =:
   weigh 72 vs 77
   case >:
     fake 72
   case <:
```

```
    fake 77
   end
  case >:
   weigh 79 vs 80
   case <:
    fake 79
   case =:
    fake 70
   case >:
    fake 80
   end
  end
 case =:
  weigh 73+86 vs 74+87
  case <:
   weigh 74 vs 1
   case >:
    fake 74
   case =:
    fake 86
   end
  case =:
   weigh 88 vs 100
   case <:
    fake 88
   case =:
    fake 75
   case >:
    fake 100
   end
```

```
  case >:
   weigh 73 vs 1
   case >:
     fake 73
   case =:
     fake 87
   end
  end
 case >:
  weigh 67+82+85 vs 68+83+84
  case <:
   weigh 82 vs 85
   case <:
    fake 82
   case =:
    fake 68
   case >:
    fake 85
   end
  case =:
   weigh 69 vs 76
   case >:
     fake 69
   case <:
     fake 76
   end
  case >:
   weigh 83 vs 84
   case <:
    fake 83
```

```
    case =:
     fake 67
    case >:
     fake 84
    end
   end
  end
 end
case >:
 weigh
1+2+3+4+5+6+7+8+9+10+11+34+35+36+37+38+39+40+41+4
2+43+44 vs
12+13+14+15+16+17+18+19+20+21+22+45+46+47+48+49+5
0+51+52+53+54+55
 case <:
  weigh 12+13+14+21+34+35+36+43 vs
15+16+17+22+37+38+39+44
  case <:
   weigh 15+34+43 vs 16+35+36
   case <:
    weigh 34 vs 43
    case <:
     fake 34
    case =:
     fake 16
    case >:
     fake 43
    end
   case =:
    weigh 17 vs 22
```

```
    case >:
      fake 17
    case <:
      fake 22
    end
   case >:
    weigh 35 vs 36
    case <:
     fake 35
    case =:
     fake 15
    case >:
     fake 36
    end
   end
  case =:
   weigh 18+40 vs 19+41
   case <:
    weigh 19 vs 67
    case >:
      fake 19
    case =:
      fake 40
    end
   case =:
    weigh 20 vs 67
    case >:
      fake 20
    case =:
      fake 42
```

```
 end
case >:
 weigh 18 vs 67
 case >:
  fake 18
 case =:
  fake 41
 end
end
case >:
weigh 12+37+44 vs 13+38+39
case <:
 weigh 37 vs 44
 case <:
  fake 37
 case =:
  fake 13
 case >:
  fake 44
 end
case =:
 weigh 14 vs 21
 case >:
  fake 14
 case <:
  fake 21
 end
case >:
 weigh 38 vs 39
 case <:
```

```
     fake 38
    case =:
     fake 12
    case >:
     fake 39
    end
   end
  end
 case =:
  weigh 23+24+25+32+56+57+58+65 vs
26+27+28+33+59+60+61+66
  case <:
   weigh 26+56+65 vs 27+57+58
   case <:
    weigh 56 vs 65
    case <:
     fake 56
    case =:
     fake 27
    case >:
     fake 65
    end
   case =:
    weigh 28 vs 33
    case >:
      fake 28
    case <:
      fake 33
    end
   case >:
```

```
  weigh 57 vs 58
  case <:
   fake 57
  case =:
   fake 26
  case >:
   fake 58
  end
 end
case =:
 weigh 29+62 vs 30+63
 case <:
  weigh 30 vs 67
  case >:
    fake 30
  case =:
    fake 62
  end
 case =:
  weigh 31 vs 67
  case >:
    fake 31
  case =:
    fake 64
  end
 case >:
  weigh 29 vs 67
  case >:
    fake 29
  case =:
```

```
    fake 63
   end
  end
 case >:
  weigh 23+59+66 vs 24+60+61
  case <:
   weigh 59 vs 66
   case <:
    fake 59
   case =:
    fake 24
   case >:
    fake 66
   end
  case =:
   weigh 25 vs 32
   case >:
    fake 25
   case <:
    fake 32
   end
  case >:
   weigh 60 vs 61
   case <:
    fake 60
   case =:
    fake 23
   case >:
    fake 61
   end
```

```
 end
end
case >:
weigh 1+2+3+10+45+46+47+54 vs 4+5+6+11+48+49+50+55
case <:
 weigh 4+45+54 vs 5+46+47
 case <:
  weigh 45 vs 54
  case <:
   fake 45
  case =:
   fake 5
  case >:
   fake 54
  end
 case =:
  weigh 6 vs 11
  case >:
    fake 6
  case <:
    fake 11
  end
 case >:
  weigh 46 vs 47
  case <:
   fake 46
  case =:
   fake 4
  case >:
   fake 47
```

```
  end
 end
case =:
 weigh 7+51 vs 8+52
 case <:
  weigh 8 vs 67
  case >:
   fake 8
  case =:
   fake 51
  end
 case =:
  weigh 9 vs 67
  case >:
   fake 9
  case =:
   fake 53
  end
 case >:
  weigh 7 vs 67
  case >:
   fake 7
  case =:
   fake 52
  end
 end
case >:
 weigh 1+48+55 vs 2+49+50
 case <:
  weigh 48 vs 55
```

```
    case <:
     fake 48
    case =:
     fake 2
    case >:
     fake 55
    end
   case =:
    weigh 3 vs 10
    case >:
      fake 3
    case <:
      fake 10
    end
   case >:
    weigh 49 vs 50
    case <:
     fake 49
    case =:
     fake 1
    case >:
     fake 50
    end
   end
  end
 end
end
```

[*Halaman ini sengaja dikosongkan*]

BIODATA PENULIS

Penulis bernama Fadli Wildan Firjatullah, putra pertama dari tiga bersaudara yang lahir pada tanggal 28 Oktober 1997 di Kuta. Penulis telah mengenyam pendidikan di SD Muhammadiyah 06 Genteng pada tahun 2004 hingga 2010, MTs Negeri Genteng pada tahun 2010 hingga 2013, dan MA Negeri Insan Cendekia Serpong 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 menempuh perkuliahan penulis aktif mengikuti kompetisi di bidang pemrograman dan penambangan data. Beberapa hasil yang diperoleh penulis adalah Finalis Data Mining Gemastik 10 (2017), Finalis ACM-ICPC Asia Jakarta Regional Contest (2017), 2nd Best Participant Data Science Academy Compfest (2018), dan Finalis Data Mining Gemastik 11 (2018).

Selain itu, penulis juga menerima penghargaan sebagai penerima Beasiswa Unggulan oleh Kementerian Pendidikan dan Kebudayaan. Penulis dapat dihubungi melalui surel di fwildanf@gmail.com.