TUGAS AKHIR – IF184802

STUDI BANDING KINERJA ALGORITMA OPTIMASI LINIER MODEL PRIMAL DAN DUAL PADA PERSOALAN SPOJ 27099 FN16ROAD - ROAD TIMES

IVANDA ZEVI AMALIA
05111640000041

Dosen Pembimbing
Rully Soelaiman, S.Kom., M.Kom.
M. M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil.

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

TUGAS AKHIR – IF184802

STUDI BANDING KINERJA ALGORITMA OPTIMASI LINIER MODEL PRIMAL DAN DUAL PADA PERSOALAN SPOJ 27099 FN16ROAD – ROAD TIMES

IVANDA ZEVI AMALIA
05111640000041

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

Dosen Pembimbing II
M. M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil.

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

[*Halaman ini sengaja dikosongkan*]

FINAL PROJECT – IF184802

COMPARATIVE STUDY LINEAR OPTIMIZATION'S ALGORITHM PERFORMANCE OF PRIMAL AND DUAL MODEL FOR SOLVING SPOJ'S 27099 FN16ROAD – ROAD TIMES PROBLEM

IVANDA ZEVI AMALIA
05111640000041

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

Supervisor II
M. M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil.

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

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

STUDI BANDING KINERJA ALGORITMA OPTIMASI LINIER MODEL PRIMAL DAN DUAL PADA PERSOALAN SPOJ 27099 FN16ROAD – ROAD TIMES

TUGAS AKHIR

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

Oleh:
Ivanda Zevi Amalia
NRP: 051116 40000 041

Disetujui oleh Dosen Pembimbing Tugas Akhir:

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

M. M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil.
NIP. 197402092002121001 ..... (Pembimbing 2)

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN INSTITUT TEKNOLOGI SEPULUH NOPEMBER FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS DEPARTEMEN TEKNIK INFORMATIKA

SURABAYA
JANUARI 2020

[*Halaman ini sengaja dikosongkan*]

STUDI BANDING KINERJA ALGORITMA OPTIMASI LINIER MODEL PRIMAL DAN DUAL PADA PERSOALAN SPOJ 27099 FN16ROAD – ROAD TIMES

Nama Mahasiswa	: Ivanda Zevi Amalia
NRP	: 051116 40000 041
Departemen	: Departemen Teknik Informatika Fakultas Teknologi Elektro dan Informatika Cerdas – ITS
Dosen Pembimbing 1	: Rully Soelaiman, S.Kom., M.Kom.
Dosen Pembimbing 2	: M. M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil.

Abstrak

Permasalahan dalam Tugas Akhir ini diambil dari dunia nyata namun sudah disederhanakan ke dalam bentuk soal yang terdapat pada situs Sphere Online Judge "FN16ROAD – Road Times". Secara umum persoalan yang akan diselesaikan pada penelitian Tugas Akhir ini, dapat direpresentasikan sebagai persoalan optimasi linier. Metode penyelesaian yang dipilih adalah algoritma Simpleks.

Algoritma Simpleks diimplementasikan pada dua model, yaitu model primal dan model dual. Dilakukan uji kebenaran pada hasil implementasi algoritma Simpleks pada model primal dan model dual. Kedua model tersebut kemudian dibandingkan kinerjanya baik dari segi waktu maupun memori.

Algoritma Simpleks diimplementasikan dengan menggunakan bahasa pemrograman C++. Dari uji coba yang telah dilakukan, didapatkan kesimpulan bahwa algoritma yang dirancang telah sesuai dengan permasalahan ini. Model dual memiliki rata-rata waktu lebih baik dibandingkan model primal dengan selisih sebesar 2,007 detik. Sedangkan untuk rata-rata memori, model primal memiliki memori lebih kecil dibandingkan model dual dengan selisih sebesar 0,02 M.

Kata kunci: Model Primal, Model Dual, Optimasi Linier, Algoritma Simpleks

COMPARATIVE STUDY LINEAR OPTIMIZATION'S ALGORITHM PERFORMANCE OF PRIMAL AND DUAL MODEL FOR SOLVING SPOJ'S 27099 FN16ROAD – ROAD TIMES PROBLEM

Student Name	: Ivanda Zevi Amalia
Registration Number	: 051116 40000 041
Department	: Department of Informatics Faculty of Intelligent Electrical and Informatics Technology – ITS
First Supervisor	: Rully Soelaiman, S.Kom., M.Kom.
Second Supervisor	: M. M. Irfan Subakti, S.Kom. M.Sc.Eng., M.Phil.

Abstract

The problems in this Final Project are taken from the real world but have been simplified into the form of questions contained on the Sphere Online Judge site "FN16ROAD - Road Times". In general, the problems that will be solved in this Final Project research can be represented as linear optimization problems. The chosen settlement method is the Simplex algorithm.

The Simplex algorithm is implemented in two models, namely the primal model and the dual model. The true test was conducted on the results of the implementation of the Simplex algorithm on the primal model and the dual model. The two models are then compared to their performance in terms of both time and memory.

The Simplex algorithm is implemented using the C ++ programming language. From the trials that have been carried out, it was concluded that the algorithm was designed under this problem. The dual model has an average time better than the primal model with a difference of 2,007 seconds. As for the average

memory, the primal model has a smaller memory than the dual model with a difference of 0.02 M.

Keywords: Primal Model, Dual Model, Linear Optimization, Simplex Algorithm

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:

STUDI BANDING KINERJA ALGORITMA OPTIMASI LINIER MODEL PRIMAL DAN DUAL PADA PERSOALAN SPOJ 27099 FN16ROAD – ROAD TIMES

Pengerjaan Tugas Akhir ini dilakukan untuk memenuhi salah satu syarat meraih gelar Sarjana di Departemen Informatika Fakultas Teknologi Informasi dan Komunikasi 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. Terimakasih kepada Allah SWT, di mana penulis masih diberi kesempatan, kesehatan dan umur untuk menempuh kuliah disini dan menjalani hidup dengan baik.
2. Ibu dan Bapak penulis yang selalu memberikan perhatian, do'a, dorongan, dan juga kasih sayang agar lebih semangat menempuh kuliah dan segera menyelesaikan Tugas Akhir ini.
3. Bapak Rully Soelaiman, S.Kom., M.Kom. selaku Dosen Pembimbing yang telah membimbing penulis selama masa kuliah maupun selama penyelesaian Tugas Akhir ini, Dosen yang paling perhatian kepada penulis dalam memberi ilmu, nasihat, dan motivasi selama menempuh kuliah di Departemen Teknik Informatika ITS.

4. Bapak M. M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil. Selaku dosen pembimbing yang telah memberikan ilmu, dan masukan kepada penulis.
5. Teman-teman administrator Laboratorium MIS (*Mobile Innovation Studio*), sebagai teman dan keluarga penulis.
6. Teman-teman BPH Schematics 2018 yang sudah memberikan banyak pengalaman kepada penulis.
7. Teman-teman PH HMTC Garang yang sudah memberikan kepercayaan kepada penulis.
8. Teman-teman angkatan 2016 jurusan Teknik Informatika ITS yang telah menemani perjuangan penulis selama masa perkuliahan.
9. Serta pihak-pihak lain yang tidak dapat disebutkan di sini yang telah banyak membantu penulis dalam penyusunan Tugas Akhir ini.

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

Surabaya, Januari 2020

Ivanda Zevi Amalia

DAFTAR ISI

[*Halaman ini sengaja dikosongkan*]

DAFTAR GAMBAR

DAFTAR TABEL

[*Halaman ini sengaja dikosongkan*]

DAFTAR KODE SUMBER

[*Halaman ini sengaja dikosongkan*]

BAB I
PENDAHULUAN

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

1.1 Latar Belakang

Terdapat banyak permasalahan dunia nyata yang berkaitan dengan pemrograman linier. Pemrograman linier berguna untuk meningkatkan kualitas keputusan dalam proses pemecahan suatu masalah yang kompleks. Pemrograman linier dapat dimanfaatkan untuk menyelesaikan berbagai macam permasalahan dunia nyata, mulai dari permasalahan ekonomi, sosial, militer, industri, dan masih banyak lagi. Sehingga pemrograman linier sangat bermanfaat untuk dipelajari lebih dalam. Salah satu algoritma populer untuk menyelesaikan pemrograman linier adalah algoritma Simpleks.

Pengaplikasian algoritma Simpleks terhadap kehidupan nyata sangatlah beragam salah satunya terdapat pada permasalahan *FN16ROAD – Road Times* yaitu penentuan perkiraan waktu minimal dan maksimal untuk melakukan perjalanan antar rumah sakit. Untuk menyelesaikan permasalahan *FN16ROAD – Road Times* akan diimplementasikan algoritma Simpleks yang telah ada pada model *primal* dan model *dual*.

Hasil dari Tugas Akhir ini diharapkan dapat menentukan implementasi algoritma yang tepat dalam memecahkan permasalahan *FN16ROAD – Road Times* secara optimal dan diharapkan dapat memberikan kontribusi pada perkembangan ilmu pengetahuan dan teknologi informasi.

1.2 Rumusan Permasalahan

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

1. Bagaimana menganalisis serta menentukan desain dan algoritma yang tepat dan optimal untuk menyelesaikan permasalahan *SPOJ FN16ROAD – Road Times*?
2. Bagaimana mengimplementasi algoritma yang tepat dan optimal pada model *primal* dan model *dual* untuk menyelesaikan permasalahan *SPOJ FN16ROAD – Road Times?*
3. Bagaimana perbandingan kinerja dari algoritma yang sudah diimplementasikan pada model *primal* dan model *dual* pada penyelesaian permasalahan *SPOJ FN16ROAD – Road Times?*

1.3 Batasan Permasalahan

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

- Permasalahan ini merupakan permasalahan optimasi linier yang akan diselesaikan dengan algoritma Simpleks. Algoritma Simpleks akan diimplementasikan pada dua model, yaitu model *primal* dan model *dual*.

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

- Batas jumlah kota n yaitu dengan rentang 1 sampai 30.
- Jarak maksimal antar kota adalah 1000 dan bernilai positif.
- Batas jumlah jalan adalah 100.
- Jalan yang diberikan merupakan jalan satu arah.
- Batas kecepatan pada setiap jalan yang ada yaitu berada pada rentang 30 sampai 60 km/h.
- *Test case* yang digunakan adalah *test case* yang terdapat pada permasalahan SPOJ *FN16ROAD – Road Times*.
- Batas maksimal waktu untuk mendapatkan waktu minimal dan maksimal adalah 90 detik.
- Batas maksimal ukuran file *source code* yang dihasilkan adalah 50.000 B.

- Batas maksimal memori *RAM* yang digunakan untuk pemrosesan adalah 1.536 MB.

1.4 Tujuan Pembuatan Tugas Akhir

Tujuan dari pembuatan Tugas Akhir ini adalah sebagai berikut:

1. Melakukan desain dan analisis algoritma untuk menyelesaikan permasalahan *SPOJ FN16ROAD – Road Times*.
2. Melakukan implementasi algoritma pada model *primal* dan model *dual* untuk menyelesaikan permasalahan *SPOJ FN16ROAD – Road Times*.
3. Melakukan perbandingan kinerja terhadap algoritma yang sudah diimplementasikan pada model *primal* dan model *dual* pada penyelesaian permasalahan *SPOJ FN16ROAD – Road Times*.

1.5 Manfaat Tugas Akhir

Tugas Akhir ini diharapkan dapat membantu untuk memahami penyesuaian Metode Simpleks untuk menyelesaikan permasalahan *SPOJ FN16ROAD – Road Times* serta dapat mengimplementasikan solusi yang ada pada permasalahan lain yang ada pada dunia nyata sehingga dapat memberikan kontribusi pada pengembangan ilmu pengetahuan dan teknologi informasi.

1.6 Metodologi

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

1.6.1 Penyusunan Proposal Tugas Akhir

Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Pada proposal ini, penulis mengajukan gagasan untuk menyelesaikan permasalahan dalam menentukan perkiraan waktu minimal dan maksimal pada studi kasus *SPOJ FN16ROAD – Road Times*.

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 optimasi linier dan algoritma-algoritma yang ada dalam optimasi linier. Informasi tersebut didapatkan dari buku, internet, dan materi kuliah yang berhubungan dengan metode yang akan digunakan.

1.6.3 Implementasi Perangkat Lunak

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

1.6.4 Pengujian dan Evaluasi

Pada tahap ini dilakukan ujicoba kebenaran dan kinerja solusi yang telah diimplementasikan dengan melakukan pengiriman sumber kode sistem ke situs penilaian *Sphere Online Judge (SPOJ)* pada permasalahan yang terkait dan melihat hasil umpan baliknya.

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 aplikasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan

Buku Tugas Akhir ini merupakan laporan secara lengkap mengenai Tugas Akhir yang telah dikerjakan baik dari sisi teori, rancangan, maupun implementasi sehingga memudahkan bagi pembaca dan juga pihak yang ingin mengembangkannya lebih

lanjut. Sistematika penulisan buku Tugas Akhir secara garis besar antara lain:

Bab I Pendahuluan

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

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

Bab IV Implementasi

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

Bab V Pengujian dan Evaluasi

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

Bab VI Kesimpulan dan Saran

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

[*Halaman ini sengaja dikosongkan*]

BAB II
DASAR TEORI

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

2.1 Deskripsi Umum Permasalahan

Permasalahan yang diangkat pada Tugas Akhir ini adalah permasalahan pencarian perkiraan waktu minimal dan maksimal yang dibutuhkan untuk melakukan perjalanan antar rumah sakit dari kota asal ke kota tujuan. Untuk mendapatkan perkiraan waktu minimal dan maksimal tersebut diberikan beberapa data masukan. Data tersebut antara lain adalah jumlah kota, jarak antar kota, riwayat perjalanan dan rincian perjalanan yang akan dilakukan. Tidak semua kota memiliki jalan penghubung, dan jalan penghubung tersebut bersifat satu arah. Pada setiap jalan penghubung tersebut memiliki batas kecepatan yang ada pada rentang 30 sampai 60 km/h, dan truk pengantar akan mengambil jarak minimal dan menggunakan kecepatan yang konstan. Akan diberikan beberapa riwayat perjalanan sebagai pertimbangan dalam menentukan perkiraan waktu minimal dan maksimal yang diperlukan untuk melakukan perjalanan selanjutnya.

2.2 Permasalahan FN16ROAD pada SPOJ

FN16ROAD – Road Times merupakan suatu permasalahan yang terdapat pada situs penilaian *Sphere Online Judge* (SPOJ) dengan deskripsi soal dari sumber asli menggunakan bahasa Inggris, yang dapat dilihat pada Gambar 2.1.

Ubol Narongdid is the founder of a brash new startup company called Special D-Liver-E. She wants to corner the market on overnight deliveries of organs between hospitals in the Phuket area. For scheduling purposes it is important to have accurate estimates for the times to perform such deliveries. Several trips between various hospitals have already been performed, so delivery times between those pairs of hospitals are known. The company currently has software to estimate times for other (as yet untraveled) trips, but so far all the estimates have been woefully inaccurate.

You have been asked to come up with a method to improve these estimates. You have at your disposal the following information: 1) the length (in kilometers) of the roads connecting each pair of cities in the Phuket area, and 2) a set of times (in minutes) for various previously executed deliveries.

You know that roads are one-way, and each road has a fixed speed limit that lies between **30** and **60** kilometers per hour. Speed limits are real-valued and need not be integers. You also know that delivery trucks always take the route that minimizes distance traveled, and on each road will always travel at a constant speed equal to that road's speed limit. Thus you know, for example, that if a given trip is **50** kilometers, the time it will take is between **50** and **100** minutes inclusive, in the absence of any other information. Ah, but you *do* have other information, namely the times of previous deliveries. It is up to you to use it to produce the best possible estimates.

Gambar 2.1 Deskripsi Permasalahan FN16ROAD

Pada permasalahan FN16ROAD diceritakan bahwa Ubol Narongdid seorang pendiri perusahaan *startup* ingin melakukan pengiriman organ antar rumah sakit yang ada pada area Phuket. Untuk melakukan penjadwalan, penting untuk mengetahui perkiraan waktu yang akurat untuk melakukan pengiriman. Terdapat beberapa riwayat perjalanan antar rumah sakit yang sudah pernah dilakukan sebelumnya, sehingga waktu pengiriman antar rumah sakit tersebut sudah diketahui. *Software* yang sudah dimiliki oleh perusahaan tersebut tidak akurat dalam mendapatkan perkiraan waktu pengiriman.

Pada permasalahan ini, diharuskan untuk mendapatkan metode yang dapat meningkatkan hasil perkiraan tersebut dari beberapa informasi berikut: 1) Panjang jalan (dalam km) dari jalan yang menghubungkan setiap kota pada area Phuket, dan 2) informasi waktu (dalam menit) dari riwayat perjalanan yang sudah dilakukan sebelumnya.

Diketahui jika jalan yang ada merupakan jalan satu arah, dan setiap jalan memiliki batas kecepatan tetap antara 30 dan 60 km/h. Batas kecepatan merupakan bilangan *real* dan tidak harus bilangan bulat. Truk pengantar akan selalu mengambil rute yang meminimalkan jarak tempuh, dan di setiap jalan akan selalu bergerak dengan kecepatan konstan yang sama dengan batas

kecepatan jalan tersebut. Diberikan juga informasi waktu dari riwayat perjalanan yang sudah pernah dilakukan sebelumnya. Informasi tersebut dapat dimanfaatkan untuk menentukan perkiraan waktu perjalanan yang akan datang.

Input

The input starts with a line containing an integer n ($1 \leq n \leq 30$) indicating the number of cities, numbered 0 to $n-1$. After that are n lines each containing n integers specifying the distance in kilometers between cities: the j^{th} value on the i^{th} line indicates the distance when traveling directly from city i to city j. A value of -1 indicates there is no road directly connecting the two cities, and the distance from any city to itself is always 0; all other distances are positive and at most 1 000. There are at most 100 roads.

Following this is a line with a single integer r ($1 \leq r \leq 100$) indicating the number of previously executed routes. The next r lines each contain three integers s, d, and t, where s and d are the source and destination cities and t is how long the delivery from s to d took, in minutes.

Finally there is a line containing a single integer q ($1 \leq q \leq 100$) indicating the number of future delivery queries. The next q lines each contain two integers s and d giving the source and destination cities for the query.

You may assume that for each of the $r+q$ source/destination pairs in the input there is a unique minimum-distance route.

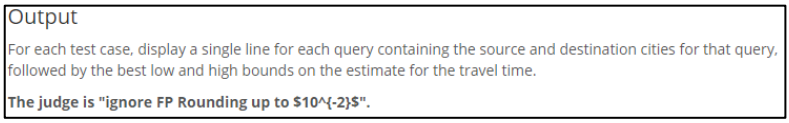
Gambar 2.2 Deskripsi Masukan Permasalahan FN16ROAD

Pada Gambar 2.2, diberikan masukan pada baris pertama satu bilangan bulat n dengan rentang $1 \leq n \leq 30$ yang merepresntasikan jumlah kota, dengan penomoran kota dimulai dari 0 hingga $n-1$. Kemudian, terdapat n baris yang masing masing berisi n bilangan bulat yang menentukan jarak (dalam km) antar kota: nilai ke-j pada baris ke-i merepresentasikan jarak ketika melakukan perjalanan dari kota i menuju kota j. Nilai -1 merepresentasikan tidak ada jalan yang menghubungkan ke dua kota tersebut, dan jarak dari kota ke kota itu sendiri selalu bernilai 0; semua jarak lainnya bernilai positif dan maksimal 1000. Maksimal terdapat 100 jalan.

Baris selanjutnya terdapat satu bilangan bulat r dengan rentang $1 \leq r \leq 100$ yang merepresentasikan jumlah dari riwayat perjalanan yang ada. Selanjutnya terdapat r baris selanjutnya terdapat 3 bilangan bulat s, d, dan t, dimana s dan d merupakan kota asal dan kota tujuan dan t adalah waktu yang diperlukan untuk melakukan perjalanan dari s menuju d (dalam menit).

Kemudian terdapat satu bilangan bulat q dengan rentang $1 \leq q \leq 100$ merepresentasikan jumlah dari perjalanan yang akan dilakukan. Selanjutnya terdapat q baris yang berisi 2 bilangan bulat s dan d yang merupakan kota asal dan tujuan yang akan dilakukan. Dapat diasumsikan bahwa untuk setiap pasangan kota asal dan tujuan $r + q$ dalam masukan terdapat rute jarak minimal yang unik.

Dari permasalahan di atas akan dihasilkan keluaran berupa satu baris untuk setiap *query* yang berisi kota asal dan tujuan dari *query* tersebut, diikuti oleh perkiraan waktu minimal dan waktu maksimal untuk melakukan perjalanan dari kota asal ke kota tujuan pada *query* tersebut (Gambar 2.3).


Output

For each test case, display a single line for each query containing the source and destination cities for that query, followed by the best low and high bounds on the estimate for the travel time.

The judge is "ignore FP Rounding up to 10^{-2}".

Gambar 2.3 Deskripsi Keluaran Permasalahan FN16ROAD


Example

```
Input:
3
0 50 -1
55 0 40
-1 40 0
1
0 2 120
3
0 1
1 2
1 0

Output:
0 1 50.0 80.0
1 2 40.0 70.0
1 0 55.0 110.0
```

Gambar 2.4 Contoh Masukan dan Keluaran Permasalahan FN16ROAD

Gambar 2.4 menjelaskan contoh masukan dan keluaran pada permasalahan FN16ROAD. Pada contoh permasalahan tersebut diberikan 3 kota dengan keterangan jarak seperti yang ada pada Tabel 2.1 di mana jarak tersebut dalam satuan km. Jarak dari kota 1 ke kota 0 adalah sejauh 55 km, jarak dari kota 1 ke kota 2 adalah

40 km, dan seterusnya. Jika jarak bernilai -1 berarti tidak ada jalan yang menghubungkan kedua kota tersebut.

Tabel 2.1 Data Jarak Antar Kota Contoh Permasalahan Road Times

	Kota 0	Kota 1	Kota 2
Kota 0	0	50	-1
Kota 1	55	0	40
Kota 2	-1	40	0

Hanya diberikan 1 riwayat perjalanan yaitu perjalanan dari kota 0 ke kota 2 yang membutuhkan waktu sebanyak 120 menit. Sehingga dari riwayat perjalanan tersebut terdapat 2 kemungkinan yang terjadi. Tabel 2.2 merupakan data riwayat waktu perjalanan jika pada proses perjalanan dari kota 0 menuju kota 1 menggunakan kecepatan maksimal 60 km/h. Sedangkan Tabel 2.3 merupakan data riwayat perjalanan jika pada proses perjalanan dari kota 1 menuju kota 2 menggunakan kecepatan maksimal.

Tabel 2.2 Kemungkinan Perjalanan Pertama

Kota	Waktu
0 - 1	50 menit
1 - 2	70 menit

Tabel 2.3 Kemungkinan Perjalanan Kedua

Kota	Waktu
0 - 1	80 menit
1 - 2	40 menit

Pada contoh permasalahan ini terdapat 3 perjalanan yang akan dilakukan, yaitu perjalanan dari kota 0 ke kota 1, kota 1 ke kota 2, dan kota 1 ke kota 0. Untuk perjalanan dari kota 0 ke kota 1 dengan jarak 50 km didapatkan hasil waktu minimal 50 menit dan waktu maksimal 80 menit. Hasil tersebut diperoleh dari mempertimbangkan waktu pada riwayat perjalanan yang diberikan. Untuk waktu minimal sudah pasti menggunakan

kecepatan maksimal, yaitu 60 km/jam sehingga hanya memerlukan waktu 50 menit. Sedangkan waktu maksimal diperoleh dari riwayat perjalanan yang ada.

Hal tersebut berlaku juga untuk perjalanan dari kota 1 ke kota 2, akan tetapi untuk perjalanan dari kota 1 ke kota 0 dikarenakan tidak memiliki data riwayat perjalanan sehingga untuk waktu maksimal didapatkan dari penggunaan kecepatan minimal yaitu 30 km/jam, dan diperoleh hasil 2*jarak. Sehingga perkiraan waktu minimal dan maksimal dari 3 perjalanan yang akan dilakukan dapat dilihat pada Tabel 2.4.

Tabel 2.4 Perkiraan Waktu Minimal dan Maksimal Perjalanan Selanjutnya

Kota	Waktu Minimal	Waktu Maksimal
0 - 1	50 menit	80 menit
1 - 2	40 menit	70 menit
1 - 0	55 menit	110 menit

2.3 Analisis dan Strategi Penyelesaian

Dari contoh permasalahan yang sudah dijelaskan pada subbab sebelumnya, dapat diperoleh beberapa kesimpulan pada permasalahan *SPOJ FN16ROAD – Road Times*. Kesimpulan tersebut antara lain:

1. Jika pada masukan tidak terdapat riwayat perjalanan dari kota asal menuju kota tujuan, maka untuk melakukan perjalanan dari kota asal menuju kota tujuan akan menggunakan kecepatan 30 km/h. Dan diperoleh perkiraan waktu maksimal adalah 2*jarak / panjang jalan.
2. Perkiraan waktu minimal akan selalu menggunakan perhitungan dengan menggunakan kecepatan 60 km/h. Sehingga diperoleh perkiraan waktu minimal adalah panjang jalan atau jarak dari kota asal menuju kota tujuan.
3. Jika pada masukan terdapat data riwayat perjalanan dari kota asal menuju kota tujuan, maka keluaran dari perkiraan waktu maksimal akan menggunakan data riwayat tersebut.

Untuk menyelesaikan permasalahan *SPOJ FN16ROAD – Road Times* harus dibuat sebuah model yang dapat merepresentasikan permasalahan tersebut, sehingga dapat diselesaikan dengan pendekatan optimasi linier. Pada pemodelan pemrograman linier, terdapat tiga komponen dasar [1] yaitu:

1. Variabel keputusan yang merupakan kumpulan variabel yang akan dicari nilainya.
2. Fungsi objektif yang merupakan fungsi tujuan yang harus dilakukan optimasi (maksimalisasi atau minimalisasi).
3. *Constraint* atau kendala yang merupakan kumpulan fungsi yang harus terpenuhi oleh solusi yang ada.

2.4 Bentuk Standar dan Bentuk *Slack*

Pemrograman linier merupakan landasan penting dalam teori optimasi. Banyak masalah realistis dapat dirumuskan dengan menggunakan model matematika linier [2]. Terdapat 2 format yang digunakan dalam pembuatan sebuah model pemrograman linier. Pada bentuk standar, semua kendala merupakan pertidaksamaan. Sedangkan pada bentuk *slack*, semua kendala merupakan persamaan (kecuali pada kendala yang menyatakan variabel harus bernilai positif) [3].

2.4.1 Bentuk Standar

Pada bentuk standar, diberikan n bilangan *real* $c_1, c_2, \ldots, c_n$; m bilangan *real* $b_1, b_2, \ldots, b_m$; dan $m\,n$ bilangan *real* a_{ij} untuk $i = 1,2,\ldots,m$ dan $j = 1,2,\ldots,n$. Akan dicari n bilangan *real* dimana

$$Maximize\ z = \sum_{j=1}^{n} c_j\, x_j \tag{2.1}$$

$Subject\ to$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i\ for\ i = 1,2, \ldots, m \tag{2.2}$$

$$x_j \geq 0\ for\ j = 1,2, \ldots, n \tag{2.3}$$

Pada model di atas, fungsi 2.1 merupakan fungsi objektif. Pertidaksamaan 2.2 dan 2.3 merupakan *constraint*/kendala, untuk pertidaksamaan 2.3 merupakan kendala nonnegativitas.

2.4.2 Bentuk *Slack*

Agar lebih efisien dalam menyelesaikan permasalahan pemrograman linier dengan menggunakan metode Simpleks, maka akan dilakukan perubahan bentuk dari bentuk standar menjadi bentuk *slack* [3]. Model yang ada harus diubah menjadi bentuk dimana kendala nonnegativitas menjadi satu-satunya pertidaksamaan, dan kendala lainnya akan diubah menjadi persamaan.

Jika pertidaksamaan 2.2 akan diubah menjadi persamaan, akan ditambahkan variabel s yang merupakan variabel *slack* yang berguna untuk mengukur perbedaan antara sisi kiri dan sisi kanan dari persamaan 2.2. Setelah ditambahkan variabel s akan menjadi persamaan 2.4 di bawah.

$$s = b_i - \sum_{j=1}^{n} a_{ij}\, x_j \,, s \geq 0 \qquad (2.4)$$

Ketika bentuk standar diubah ke bentuk *slack*, akan digunakan x_{n+i} bukannya menggunakan s untuk menunjukkan variabel *slack* yang terkait dengan pertidaksamaan ke-i. Sehingga akan menjadi:

$$x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij}\, x_j \qquad (2.5)$$

Pada pemrograman linier semua kendala kecuali untuk kendala nonnegativitas merupakan sebuah persamaan. Dengan format penulisan, untuk setiap kendala (persamaan) memiliki satu variabel pada sisi kiri (*left-hand side*) dan variabel sisanya berada pada sisi kanan (*right-hand side*). Setiap persamaan memiliki himpunan variabel yang sama pada sisi kanannya, dan variabel tersebut juga merupakan variabel yang muncul pada fungsi

objektif. Variabel yang berada pada sisi kiri disebut ***basic variables*** dan variabel yang berada pada sisi kanan disebut ***nonbasic variables***.

Seperti pada bentuk standar, akan lebih mudah jika memiliki notasi yang lebih ringkas untuk menggambarkan bentuk *slack*. Sehingga akan digunakan N untuk menunjukkan himpunan indeks dari *nonbasic variables* dan B untuk menunjukkan himpunan indeks dari *basic variables*. Sedangkan variabel v untuk menunjukkan nilai konstan opsional pada fungsi objektif dan semua variabel x harus bernilai positif. Jadi pertidaksamaan 2.1 dan 2.2 ketika dijadikan bentuk *slack* akan menjadi:

$$z = v + \sum_{j \in N} c_j x_j \tag{2.6}$$

$$x_i = b_i - \sum_{j \in N} a_{ij} x_j \quad for\ i \in B \tag{2.7}$$

2.5 Pemodelan Permasalahan FN16ROAD

Untuk membuat pemodelan permasalahan FN16ROAD dapat melihat beberapa kesimpulan dari permasalahan FN16ROAD yang sudah dijelaskan pada subbab sebelumnya. Dari kesimpulan tersebut dapat dibuat model yang dapat merepresentasikan kesimpulan-kesimpulan tersebut. Variabel-variabel yang akan digunakan dijelaskan pada Tabel 2.5.

1. Perkiraan waktu yang akan ditempuh untuk melakukan perjalanan dari kota asal s menuju kota tujuan d adalah

$$l_r \leq t_r \leq 2l_r$$

Dari pertidaksamaan di atas, dapat dituliskan menjadi:

$$t_r \leq 2l_r$$

$$-t_r \leq -l_r$$

2. Jika diketahui waktu yang dibutuhkan dari s_i ke d_i adalah a_i jam. Maka dapat ditulis:

$$\sum_{r \in R(s_i, d_i)} t_r = a_i$$

Dari pertidaksamaan di atas dapat dituliskan menjadi:

$$-\sum_{r \in R(s_i, d_i)} t_r \leq -a_i$$

$$\sum_{r \in R(s_i, d_i)} t_r \leq a_i$$

3. Ditanyakan perkiraan waktu minimal dan maksimal yang diperlukan untuk melakukan perjalanan dari kota asal s menuju kota tujuan d. Sehingga fungsi objektif dari permasalahan ini adalah

$$minimize \sum_{r \in R(s,d)} t_r$$

atau

$$maximize \sum_{r \in R(s,d)} t_r$$

Untuk mencari waktu minimal dapat dituliskan menjadi:

$$maximize \sum_{r \in R(s,d)} -t_r$$

Dari pertidaksamaan di atas, dapat dirangkum menjadi:

$$\boldsymbol{Maximize} \sum_{r \in R(s,d)} t_r \ \ dan \ \ \boldsymbol{Maximize} \sum_{r \in R(s,d)} -t_r$$

Subject to

$$-t_r \leq -l_r$$

$$t_r \leq 2l_r$$

$$-\sum_{r \in R(s_i,d_i)} t_r \leq -a_i$$

$$\sum_{r \in R(s_i,d_i)} t_r \leq a_i$$

Tabel 2.5 Variabel pada Pemodelan FN16ROAD

No	Variabel	Penjelasan
1	s	Kota asal
2	d	Kota tujuan
3	r	Jalan yang menghubungkan s menuju d
4	C	Kota
5	t_r	Waktu yang diperlukan untuk melakukan perjalanan dari s ke d
6	l_r	Panjang jalan atau jarak dari s menuju d
7	$R(s,d)$	Rute terpendek untuk sekumpulan pasangan (s,d)
8	e	Indeks dari *entering variable*

2.6 Metode Simpleks

Metode Simpleks merupakan metode klasik untuk menyelesaikan pemrograman linier [2]. Untuk melakukan perhitungan menggunakan metode Simpleks diharuskan memiliki dua persyaratan berikut pada kendala yang dimiliki [4][5]:

1. Semua kendala (kecuali kendala nonnegativitas) merupakan sebuah persamaan dengan ruas kanan bernilai positif.
2. Semua variabel bernilai positif.

Untuk mencapai *optimality condition*, yang akan dijadikan *entering variable* pada permasalahan maksimalisasi adalah *nonbasic variable* dengan nilai koefisien paling negatif pada

$z - row$. Sedangkan pada permasalahan minimalisasi, *entering variable* yang dipilih adalah *nonbasic variable* dengan nilai koefisien paling positif pada $z - row$. Solusi optimal sudah tercapai ketika semua koefisien pada $z - row$ dari *nonbasic variable* tidak ada yang bernilai negatif untuk kasus maksimalisasi dan sebaliknya tidak ada yang bernilai positif untuk kasus minimalisasi [1].

Sedangkan untuk mencapai *feasibility condition* baik pada kasus maksimalisasi maupun minimalisasi, yang dipilih menjadi *leaving variable* adalah *basic variable* dengan nilai rasio nonnegatif paling kecil dengan nilai pembagi harus bernilai positif [1].

Untuk melakukan pertukaran fungsi (*pivot*) antara *nonbasic variable* dan *basic variable* dilakukan operasi Gauss-Jordan, operasi yang akan dilakukan adalah sebagai berikut:

1. Baris *Pivot*
   b. Lakukan pergantian pada *leaving variable* yang ada pada *Basic column* dengan *entering variable*.
   c. Baris *pivot* baru = baris *pivot* sekarang ÷ *pivot element*
2. Baris lainnya, termasuk baris z
   Baris baru = (baris sekarang) – (koefisien kolom *pivot*) × (baris *pivot* baru)

Sehingga untuk menggunakan metode Simpleks, semua kendala harus diubah ke dalam bentuk *slack*. Berikut adalah langkah-langkah penggunaan metode Simpleks:

1. Tentukan solusi awal (*intial basic feasible solution*) dengan menetapkan $n - m$ variabel nonbasis sama dengan nol dimana n jumlah variabel dan m banyaknya kendala.
2. Kemudian dipilih sebuah *entering variable* (variabel yang masuk) menggunakan *optimality condition*. Berhenti jika sudah tidak ada *entering variable*, solusi terakhir merupakan solusi optimal. Selain itu, maka lanjutkan ke langkah 3.

3. Selanjutnya pilih sebuah *leaving variable* (variabel yang keluar) menggunakan *feasibility condition.*
4. Tentukan solusi yang baru (*new basic solution*) dengan menggunakan perhitungan Gauss-Jordan. Selanjutnya kembali ke langkah 2.

Setelah pertidaksamaan 2.1 dan 2.2 diubah menjadi bentuk *slack* seperti pada persamaan 2.6 dan 2.7, persamaan-persamaan tersebut dapat dibentuk menjadi sebuah kerangka tabel Simpleks seperti pada Tabel 2.6 untuk mempermudah melakukan perhitungan.

Tabel 2.6 Implementasi Tabel Simpleks

Basic Variable	z	x_1	x_2	...	x_n	s_1	s_2	...	s_n	v
z	1	$-c_1$	$-c_2$	...	$-c_n$	0	0	0	0	0
s_1	0	a_{11}	a_{12}	...	a_{13}	1	0	0	0	b_1
s_2	0	a_{21}	a_{22}	...	a_{2n}	0	1	0	0	b_2
...	...	...	...	...	...	...	...	...	...	...
s_n	0	a_{m1}	a_{m2}	...	a_{mn}	0	0	0	1	b_m

2.7 Model *Dual*

Model *dual* adalah model yang didefinisikan secara langsung dan sistematis dari model *primal* (*original*). Kedua model tersebut sangat erat hubungannya sehingga solusi optimal dari satu model secara otomatis memberikan solusi optimal untuk yang lainnya [1]. Cara membangun model *dual* dari model *primal* adalah sebagai berikut:

1. Variabel model *dual* didefinisikan untuk setiap persamaan (kendala) model *primal*.
2. Kendala model *dual* didefinisikan untuk setiap variabel model *primal*.

3. Koefisien pada variabel dari kendala model *primal* mendefinisikan koefisien pada sisi kiri (*left-hand-side*) dari kendala model *dual*.
4. Koefisien pada fungsi objektif dari model *primal* mendefinisikan sisi kanan (*right-hand-side*) dari model *dual*.
5. Koefisien pada fungsi objektif dari model *dual* mendefinisikan sisi kanan (*right-hand-side*) dari kendala model *primal*.

Cara membangun model *dual* tersebut dirangkum dalam Tabel 2.7. Dalam membangun model *dual* terdapat beberapa aturan. Aturan tersebut dapat dilihat pada Tabel 2.8.

Tabel 2.7 Konstruksi Model *Dual* dari Model *Primal*

	Variabel *Primal*						
	x_1	x_2	...	x_j	...	x_n	
Variabel *Dual*	c_1	c_2	...	c_j	...	c_n	*Right-hind side*
y_1	a_{11}	a_{12}	...	a_{1j}	...	a_{1n}	b_1
y_2	a_{21}	a_{22}	...	a_{2j}	...	a_{2n}	b_2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
y_m	a_{m1}	a_{m2}	...	a_{mj}	...	a_{mn}	b_m
				↑			↑
				j^{th} kendala model *dual*			Koefisien fungsi objektif model *dual*

Tabel 2.8 Aturan Konstruksi Model *Dual*

Maximization Problem		*Minimization Problem*
Kendala		Variabel
$\geq$	$\leftrightarrow$	≤ 0
$\leq$	$\leftrightarrow$	≥ 0
$=$	$\leftrightarrow$	*Unrestricted*
Variabel		Kendala
≤ 0	$\leftrightarrow$	$\geq$
≥ 0	$\leftrightarrow$	$\leq$
Unrestricted	$\leftrightarrow$	$=$

Tabel 2.8 menunjukkan jika pada model *primal* merupakan kasus maksimalisasi, maka pada model *dual* merupakan kasus minimalisasi, dan sebaliknya.

2.8 Penggunaan Simpleks pada Pemrograman Linier

Metode Simpleks dapat diterapkan ke dalam bentuk kode seperti *pseudocode* metode Simpleks yang ada pada Gambar 2.6 [3]. Dalam fungsi Simplex terdapat fungsi Pivot yang digunakan untuk melakukan pertukaran peran antara *entering variable* dengan *leaving variable*. Tahapan dalam melakukan pivot terdapat pada Gambar 2.5 [3].

Pivot (N, B, A, b, c, v, l, e)

1. //Compute the coefficients of the equation for new basic variable x_e
2. Let $\hat{A}$ be a new $m \times n$ matrix
3. $\widehat{b_e} = b_l / a_{le}$
4. **for** each $j \in N - \{e\}$
5. $\quad \widehat{a_{ej}} = a_{lj} / a_{le}$
6. $\widehat{a_{el}} = 1 / a_{le}$
7. //Compute coefficients remaining constraints
8. **for** each $i \in B - \{l\}$
9. $\quad \widehat{b_i} = b_i - a_{ie}\widehat{b_e}$
10. $\quad$ **for** each $j \in N - \{e\}$
11. $\quad\quad \widehat{a_{ij}} = a_{ij} - a_{ie}\widehat{a_{ej}}$
12. $\quad \widehat{a_{il}} \leftarrow -a_{ie}\widehat{a_{el}}$
13. //Compute the objective function
14. $\hat{v} = v + c_e\widehat{b_e}$
15. **for** each $j \in N - \{e\}$
16. $\quad \widehat{c_j} = c_j - c_e\widehat{a_{ej}}$
17. $\widehat{c_l} = -c_e\widehat{a_{el}}$
18. //Compute new sets of basic and nonbasic variables
19. $\hat{N} = N - \{e\} \cup \{l\}$
20. $\hat{B} = B - \{l\} \cup \{e\}$
21. **return** $(\hat{N}, \hat{B}, \hat{A}, \hat{b}, \hat{c}, \hat{v})$

Gambar 2.5 *Pseudocode* Pivot

Pada *pseudocode* Pivot di atas, terdapat 8 parameter yang digunakan. Parameter-parameter tersebut sama dengan variabel yang ada pada persamaan 2.6 dan 2.7 yang sudah dijelaskan pada subbab 2.4.2, untuk lebih jelasnya dapat dilihat pada Tabel 2.9.

Tabel 2.9 Parameter pada *Pseudocode*

No	Parameter	Penjelasan
1	N	Himpunan indeks dari variabel yang berada pada sisi kanan persamaan (*nonbasic variables*)
2	B	Himpunan indeks dari variabel yang berada pada sisi kiri persamaan (*basic variables*)
3	A	$(m \times n)$ matriks yang merepresentasikan koefisien dari sisi kanan persamaan
4	b	m vektor yang merepresentasikan nilai konstan pada *constraint*
5	c	n vektor yang merepresentasikan koefisien dari fungsi objektif
6	v	Nilai konstan opsional yang ada pada fungsi objektif
7	l	Indeks dari leaving variable
8	e	Indeks dari entering variable

Proses kerja dari *pseudocode* Pivot di atas adalah sebagai berikut:

1. Pada baris 3-6 akan dilakukan pertukaran peran antara *leaving variable* x_l dan *entering variable* x_e. Dilakukan perhitungan nilai koefisien pada persamaan baru untuk nilai x_e dengan cara menulis ulang persamaan yang memiliki x_l pada sisi kiri dan menggantinya dengan x_e pada sisi kiri.
2. Pada baris 8-12 dilakukan *update* pada semua persamaan yang tersisa dengan cara melakukan substitusi pada sisi kanan dari persamaan baru untuk setiap x_e.
3. Pada baris 14-17 dilakukan substitusi yang sama pada fungsi objektif.
4. Pada baris 19-20 dilakukan *update* nilai dari N dan B.

5. Pada baris 21 akan mengembalikan nilai dari bentuk *slack* yang baru.

Simplex (A, b, c)

```
1.  (N, B, A, b, c, v) =Initialize-Simplex(A, b, c)
2.  let Δ be a new vector of length n
3.  while some index j ∈ N has c_j > 0
4.       choose an index e ∈ N for which c_e > 0
5.       for each index i ∈ B
6.            if a_ie > 0
7.                 Δ_i = b_i / a_ie
8.            else Δ_i = ∞
9.       choose an index l ∈ B that minimizes Δ_i
10.      if Δ_l == ∞
11.           return "unbounded"
12.      else (N, B, A, b, c, v) = Pivot (N, B, A, b, c, v, l, e)
13. for i = 1 to n
14.      if i ∈ B
15.           x̄_i = b_i
16.      else x̄_i = 0
17. return (x̄_1, x̄_2, ..., x̄_n)
```

Gambar 2.6 *Pseudocode* Simpleks

Parameter yang digunakan pada fungsi *Simplex* yang ada pada Gambar 2.6 sama dengan parameter pada fungsi *Pivot* yang sudah dijelaskan pada Tabel 2.9. Untuk proses kerja dari *pseudocode* Simpleks di atas adalah sebagai berikut:

1. Pada baris 1 dilakukan pemanggilan fungsi *Initialize-Simplex* seperti pada Gambar 2.7. Akan dilakukan pengecekan apakah permasalahan yang ada memiliki solusi yang layak atau tidak. Jika tidak memiliki solusi yang layak maka iterasi akan dihentikan, dan sebaliknya jika memiliki solusi yang layak maka fungsi *Initialize-Simplex* tersebut akan mengembalikan nilai persamaan ke dalam bentuk *slack*.

2. Pada baris 3-12 terdapat perulangan yang merupakan inti dari algoritma Simpleks. Perulangan ini akan berhenti jika semua koefisian pada fungsi objektif sudah bernilai negatif. Isi dari perulangan tersebut adalah:
   a. Pada baris 4 dilakukan pemilihan *entering variable*.
   b. Pada baris 5-9 dilakukan pengecekan batas pada semua kendala dengan cara menaikkan nilai dari *entering variable* yang dipilih. Kemudian dipilih salah satu variabel yang ada untuk dijadikan *leaving variable*.
   c. Jika tidak ada variabel yang memenuhi batas yang ada, maka pada baris 11 akan mengembalikan nilai "*unbounded*"
   d. Setelah berhasil mendapatkan *leaving variable*, pada baris 12 akan dilakukan pergantian peran dengan memanggil fungsi *Pivot*.
3. Pada baris 13-16 dilakukan perhitungan hasil dari persoalan yang ada dengan cara membuat semua nilai dari *nonbasic variables* = 0 dan setiap *basic variables* = b_i
4. Pada baris 17, hasil akhir dari perhitungan Simpleks yang ada pada tahap nomor 3 akan dikembalikan.

Seperti yang sudah dijelaskan sebelumnya, sebelum melakukan metode Simpleks, harus dipastikan terlebih dahulu apakah program linier tersebut memiliki solusi yang layak. Untuk menentukan hal tersebut, akan dirumuskan sebuah ***auxiliary linear program.*** Dengan menggunakan *auxiliary linear program* ini, akan didapatkan sebuah *slack form* dengan solusi yang layak. Kemudian hasil dari *auxiliary linear program* ini akan menjadi program linier awalan dengan solusi layak untuk dilakukan proses Simpleks. Proses tersebut dapat dilihat pada *pseudocode Initialize-Simplex* di bawah [3].

Initialize-Simplex (A, b, c)

1. let k be the index of the minimum b_i
2. **if** $b_k \geq 0$ //is the initial basic solution feasible?
3. **return** $(\{1,2,\dots,n\},\{n+1,n+2,\dots,n+m\},A,b,c,0)$
4. form L_{aux} by adding $-x_0$ to the left-hand side of each constraint and setting the objective function to $-x_0$
5. let (N,B,A,b,c,v) be the resulting slack form for L_{aux}
6. $l = n + k$
7. //L_{aux} has $n+1$ nonbasic variables and m basic variables
8. $(N,B,A,b,c,v) = Pivot\,(N,B,A,b,c,v,l,0)$
9. // The basic solution is now feasible for L_{aux}
10. iterate the **while** loop of lines 3-12 of Simplex until an optimal solution to L_{aux} is found
11. **if** the optimal solution to L_{aux} sets $\overline{x_0}$ to 0
12. **if** $\overline{x_0}$ is basic
13. perform one (degenerate) pivot to make it nonbasic
14. from the final slack form of L_{aux}, remove x_0 from the constraints and restore the original objective function of L, but replace each basic variable in this objective function by the right-hand side of its associated constraints
15. **return** the modified final slack form
16. **else return** "infeasible"

Gambar 2.7 *Pseudocode* Initialize-Simplex

Untuk proses kerja dari *pseudocode* Intialize-Simplex di atas adalah sebagai berikut:

1. Pada baris 2, jika $b_k \geq 0$ dan k merupakan indeks dari nilai minimal b_i maka fungsi Initialize-Simplex akan mengembalikan nilai $(N, B, A, b, c, 0)$.
2. Dari bentuk L_{aux} akan ditambahkan $-x_0$ pada sisi kiri dari seiap kendala dan mengganti fungsi objektif menjadi $-x_0$
3. Nilai dari (N, B, A, b, c, v) akan menjadi hasil dari bentuk *slack* untuk L_{aux}.
4. Dengan nilai $l = n + k$, akan dilakukan pemanggilan fungsi Pivot dengan *leaving variable* = l dan *entering variable* = 0.
5. Lakukan perulangan pada baris 3-12 fungsi Simpleks (Gambar 2.6) hingga diperoleh solusi optimal dari L_{aux}.
6. Jika solusi optimal dari L_{aux} sudah diperoleh, substitusi nilai $x_0 = 0$
7. Jika x_0 merupakan *basic variable* lakukan proses Pivot satu kali lagi agar x_0 menjadi *nonbasic variable*.
8. Dari bentuk *slack* terakhir L_{aux}, hilangkan x_0 dari kendala, dan kembalikan fungsi objektif yang asli. Akan tetapi ubah setiap *basic variable* yang ada pada fungsi objektif dengan sisi kanan dari kendala terkait.

2.9 Penyelesaian Permasalahan FN16ROAD

Dalam menyelesaikan permasalahan FN16ROAD akan diimplementasikan 2 model, yaitu model *primal* dan model *dual*.

2.9.1 Model *Primal*

Pada *test case* pertama yang ada pada Gambar 2.4, ditanyakan perkiraan waktu minimal dan maksimal yang diperlukan untuk menempuh perjalanan dari kota 0 menuju kota 1. Untuk pengerjaan selanjutnya, penomoran kota dimulai dari angka 1, sehingga model dari *test case* tersebut untuk mencari waktu maksimal dapat dituliskan menjadi:

$$\boldsymbol{Maximize}\ z_{original} = x_1$$

Subject to

(1) $-x_1 \leq -50$
(2) $x_1 \leq 100$
(3) $-x_2 \leq -55$
(4) $x_2 \leq 110$
(5) $-x_3 \leq -40$
(6) $x_3 \leq 80$
(7) $-x_4 \leq -40$
(8) $x_4 \leq 80$
(9) $x_1 + x_3 \leq 120$
(10) $-x_1 - x_3 \leq -120$

Sebelum melakukan metode Simpleks, harus dipastikan apakah program linier tersebut memiliki solusi yang layak. Untuk menentukan hal tersebut, akan dirumuskan sebuah *auxiliary linear program*. Dengan menggunakan *auxiliary linear program* ini, akan didapatkan sebuah bentuk *slack* dengan solusi yang layak. Kemudian hasil dari *auxiliary linear program* ini akan menjadi program linier awalan dengan solusi layak untuk dilakukan proses Simpleks. Proses tersebut dapat dilihat pada *pseudocode* Initialize-Simplex (Gambar 2.7).

Untuk tahap pertama akan ditambahkan $-x_0$ pada sisi kiri semua kendala yang ada, dan mengganti fungsi objektif menjadi $-x_0$.

$$\boldsymbol{Maximize}\ z_{aux} = -x_0$$

Subject to

(1) $-x_1 - x_0 \leq -50$
(2) $x_1 - x_0 \leq 100$
(3) $-x_2 - x_0 \leq -55$
(4) $x_2 - x_0 \leq 110$
(5) $-x_3 - x_0 \leq -40$
(6) $x_3 - x_0 \leq 80$
(7) $-x_4 - x_0 \leq -40$
(8) $x_4 - x_0 \leq 80$
(9) $x_1 + x_3 - x_0 \leq 120$
(10) $-x_1 - x_3 - x_0 \leq -120$

Dari pemodelan di atas, akan dituliskan ke dalam bentuk *slack*.

$$\boldsymbol{z_{aux} = -x_0}$$
$$\boldsymbol{z_{original} = x_1}$$

$x_5 = -50 + x_1 + x_0$
$x_6 = 100 - x_1 + x_0$
$x_7 = -55 + x_2 + x_0$
$x_8 = 110 - x_2 + x_0$
$x_9 = -40 + x_3 + x_0$
$x_{10} = 80 - x_3 + x_0$
$x_{11} = -40 + x_4 + x_0$
$x_{12} = 80 - x_4 + x_0$
$x_{13} = 120 - x_1 - x_3 + x_0$
$x_{14} = -120 + x_1 + x_3 + x_0$

Karena masih berada dalam tahapan *Initialize-Simplex*, sehingga fungsi objektif yang akan dijadikan acuan adalah fungsi objektif $\boldsymbol{Maximize}\ z_{aux} = -x_0$. Jika dituliskan ke dalam bentuk tabel Simpleks dapat dilihat pada Tabel 2.10.

$N = (x_1, x_2, x_3, x_4, x_0)$

$B = (x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14})$

Tabel 2.10 Tabel Inisialisasi Simpleks Waktu Maksimal

$Basic$	x_1	x_2	x_3	x_4	x_0	v
$\boldsymbol{z_{aux}}$	0	0	0	0	1	0
$z_{original}$	-1	0	0	0	0	0
x_5	-1	0	0	0	-1	-50
x_6	1	0	0	0	-1	100
x_7	0	-1	0	0	-1	-55
x_8	0	1	0	0	-1	110
x_9	0	0	-1	0	-1	-40
x_{10}	0	0	1	0	-1	80
x_{11}	0	0	0	-1	-1	-40
x_{12}	0	0	0	1	-1	80
x_{13}	1	0	1	0	-1	120
x_{14}	-1	0	-1	0	-1	-120

Untuk tahap pertama akan dilakukan *pivoting* dengan *entering variable* = 0 dan *leaving variable* $= n + k = 4 + 10 = 14$. Hasil dari *pivoting* dapat dilihat pada Tabel 2.11.

$N = (x_1, x_2, x_3, x_4, x_{14})$

$B = (x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_0)$

Tabel 2.11 Tabel Inisialisasi Simpleks Hasil *Pivoting* Pertama

Basic	x_1	x_2	x_3	x_4	x_{14}	v
$\boldsymbol{z_{aux}}$	-1	0	-1	0	1	-120
$z_{original}$	-1	0	0	0	0	0
x_5	0	0	1	0	-1	70
x_6	2	0	1	0	-1	220
x_7	1	-1	1	0	-1	65
x_8	1	1	1	0	-1	230
x_9	1	0	0	0	-1	80
x_{10}	1	0	2	0	-1	200
x_{11}	1	0	1	-1	-1	80
x_{12}	1	0	1	1	-1	200
x_{13}	2	0	2	0	-1	240
x_0	1	0	1	0	-1	120

Setelah melakukan *pivoting* di atas, akan dilanjutkan dengan perulangan pada baris 3-12 metode Simpleks hingga didapatkan hasil yang optimal dari L_{aux}.

Iterasi 0

Dari semua koefisien yang ada pada fungsi objektif $\boldsymbol{z_{aux}}$ yang memiliki nilai minimal. Sehingga akan dipilih x_1 sebagai *entering variable*. Kemudian akan dipilih *leaving variable* yang memiliki nilai Δ_i paling minimal.

Tabel 2.12 Pemilihan *leaving variable* iterasi 0

Basic	x_1	v	Δ_i
x_5	0	70	-
x_6	2	220	110
x_7	1	65	65
x_8	1	230	230
x_9	1	80	80
x_{10}	1	200	200
x_{11}	1	80	80
x_{12}	1	200	200
x_{13}	2	240	120
x_0	1	120	120

Dari Tabel 2.12 dapat dilihat jika nilai Δ_7 paling minimal, sehingga x_7 dipilih sebagai *leaving variable*. Kemudian akan dilakukan *pivoting* antara x_1 dan x_7. Sehingga hasilnya dapat dilihat pada Tabel 2.13.

$N = (x_7, x_2, x_3, x_4, x_{14})$

$B = (x_5, x_6, x_1, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_0)$

Tabel 2.13 Hasil *pivoting* iterasi 0

Basic	x_7	x_2	x_3	x_4	x_{14}	v
$\boldsymbol{z_{aux}}$	1	-1	0	0	0	-55
$z_{original}$	1	-1	1	0	-1	65
x_5	0	0	1	0	-1	70
x_6	-2	2	-1	0	1	90
x_1	1	-1	1	0	-1	65
x_8	-1	2	0	0	0	165

Tabel 2.13 Hasil *pivoting* iterasi 0

$Basic$	x_7	x_2	x_3	x_4	x_{14}	v
x_9	-1	1	-1	0	0	15
x_{10}	-1	1	1	0	0	135
x_{11}	-1	1	0	-1	0	15
x_{12}	-1	1	0	1	0	135
x_{13}	-2	2	0	0	1	110
x_0	-1	1	0	0	0	55

Iterasi 1

Kemudian dipilih x_2 sebagai *entering variable* karena memiliki nilai minimal.

Tabel 2.14 Pemilihan *leaving variable* iterasi 1

$Basic$	x_2	v	Δ_i
x_5	0	70	–
x_6	2	90	45
x_1	-1	65	–
x_8	2	165	82.5
x_9	1	15	15
x_{10}	1	135	135
x_{11}	1	15	15
x_{12}	1	135	135
x_{13}	2	110	55
x_0	1	55	55

Dipilih x_9 sebagai *leaving variable* karena memiliki nilai minimal. Kemudian akan dilakukan *pivoting* antara x_2 dan x_9. Sehingga hasilnya dapat dilihat pada Tabel 2.15.

$N = (x_7, x_9, x_3, x_4, x_{14})$

$B = (x_5, x_6, x_1, x_8, x_2, x_{10}, x_{11}, x_{12}, x_{13}, x_0)$

Tabel 2.15 Hasil *pivoting* iterasi 1

Basic	x_7	x_9	x_3	x_4	x_{14}	v
$\boldsymbol{z_{aux}}$	0	1	−1	0	0	−40
$z_{original}$	0	1	0	0	−1	80
x_5	0	0	1	0	−1	70
x_6	0	−2	1	0	1	60
x_1	0	1	0	0	−1	80
x_8	1	−2	2	0	0	135
x_2	−1	1	−1	0	0	15
x_{10}	0	−1	2	0	0	120
x_{11}	0	−1	1	−1	0	0
x_{12}	0	−1	1	1	0	120
x_{13}	0	−2	2	0	1	80
x_0	0	−1	1	0	0	40

Iterasi 2

Kemudian dipilih x_3 sebagai *entering variable* karena memiliki nilai minimal.

Tabel 2.16 Pemilihan *leaving variable* iterasi 2

Basic	x_3	v	Δ_i
x_5	1	70	70
x_6	1	60	60
x_1	0	80	–
x_8	2	135	82.5
x_2	−1	15	–

Tabel 2.16 Pemilihan *leaving variable* iterasi 2

Basic	x_3	v	Δ_i
x_{10}	2	120	60
x_{11}	1	0	0
x_{12}	1	120	120
x_{13}	2	80	40
x_0	1	40	40

Dipilih x_{11} sebagai *leaving variable* karena memiliki nilai minimal. Kemudian akan dilakukan *pivoting* antara x_3 dan x_{11}. Sehingga hasilnya dapat dilihat pada Tabel 2.17.

$N = (x_7, x_9, x_{11}, x_4, x_{14})$

$B = (x_5, x_6, x_1, x_8, x_2, x_{10}, x_3, x_{12}, x_{13}, x_0)$

Tabel 2.17 Hasil *pivoting* iterasi 2

Basic	x_7	x_9	x_{11}	x_4	x_{14}	v
$\boldsymbol{z_{aux}}$	0	0	1	−1	0	−40
$z_{original}$	0	1	0	0	−1	80
x_5	0	1	−1	1	−1	70
x_6	0	−1	−1	1	1	60
x_1	0	1	0	0	−1	80
x_8	1	0	−2	2	0	135
x_2	−1	0	1	−1	0	15
x_{10}	0	1	−2	2	0	120
x_3	0	−1	1	−1	0	0
x_{12}	0	0	−1	2	0	120
x_{13}	0	0	−2	2	1	80
x_0	0	0	−1	1	0	40

Iterasi 3

Kemudian dipilih x_4 sebagai *entering variable* karena memiliki nilai minimal.

Tabel 2.18 Pemilihan *leaving variable* iterasi 3

Basic	x_4	v	Δ_i
x_5	1	70	70
x_6	1	60	60
x_1	0	80	–
x_8	2	135	82.5
x_2	−1	15	–
x_{10}	2	120	60
x_3	−1	0	–
x_{12}	2	120	60
x_{13}	2	80	40
x_0	1	40	40

Dipilih x_{13} sebagai *leaving variable* karena memiliki nilai minimal. Kemudian akan dilakukan *pivoting* antara x_4 dan x_{13}. Sehingga hasilnya dapat dilihat pada Tabel 2.19.

$N = (x_7, x_9, x_{11}, x_{13}, x_{14})$

$B = (x_5, x_6, x_1, x_8, x_2, x_{10}, x_3, x_{12}, x_4, x_0)$

Tabel 2.19 Hasil *pivoting* iterasi 3

Basic	x_7	x_9	x_{11}	x_{13}	x_{14}	v
$\boldsymbol{z_{aux}}$	0	0	0	0.5	0.5	0
$z_{original}$	0	1	0	0	−1	80
x_5	0	1	0	−0.5	−1.5	30
x_6	0	−1	0	−0.5	0.5	20
x_1	0	1	0	0	−1	80

Tabel 2.19 Hasil *pivoting* iterasi 3

$Basic$	x_7	x_9	x_{11}	x_{13}	x_{14}	v
x_8	1	0	0	-1	-1	55
x_2	-1	0	0	0.5	0.5	55
x_{10}	0	1	0	-1	-1	40
x_3	0	-1	0	0.5	0.5	40
x_{12}	0	0	1	-1	-1	40
x_4	0	0	-1	0.5	0.5	40
x_0	0	0	0	-0.5	-0.5	0

Hasil optimal dari L_{aux} sudah didapatkan, karena semua fungsi objektif sudah bernilai positif. Akan tetapi, karena x_0 masih berada pada *basic variable*, sehingga harus dilakukan *pivoting* satu kali lagi untuk menghilangkan x_0 dari basis.

Iterasi 4

Dipilih $\boldsymbol{x_{13}}$ sebagai ***entering variable*** dan $\boldsymbol{x_0}$ sebagai ***leaving variable***. Kemudian akan dilakukan *pivoting* antara x_{13} dan x_0. Sehingga hasilnya dapat dilihat pada Tabel 2.20.

$N = (x_7, x_9, x_{11}, x_0, x_{14})$

$B = (x_5, x_6, x_1, x_8, x_2, x_{10}, x_3, x_{12}, x_4, x_{13})$

Tabel 2.20 Hasil *pivoting* iterasi 4

$Basic$	x_7	x_9	x_{11}	x_0	x_{14}	v
$\boldsymbol{z_{aux}}$	0	0	0	1	0	0
$z_{original}$	0	1	0	0	-1	80
x_5	0	1	0	-1	-1	30
x_6	0	-1	0	-1	1	20
x_1	0	1	0	0	-1	80
x_8	1	0	0	-2	0	55

Tabel 2.20 Hasil *pivoting* iterasi 4

$Basic$	x_7	x_9	x_{11}	x_0	x_{14}	v
x_2	-1	0	0	1	0	55
x_{10}	0	1	0	-2	0	40
x_3	0	-1	0	1	0	40
x_{12}	0	0	1	-2	0	40
x_4	0	0	-1	1	0	40
x_{13}	0	0	0	-2	1	0

Karena semua koefisien pada fungsi objektif sudah bernilai positif dan x_0 sudah menjadi *nonbasic variable*, maka iterasi selesai dengan hasil seperti Tabel 2.20. Solusi optimal dari L_{aux} sudah didapatkan, sehingga akan dilanjutkan dengan menjalankan metode Simpleks. Dan perulangan akan terus dilakukan selama masih terdapat koefisien pada fungsi objektif yang bernilai negatif. Fungsi objektif yang akan digunakan sebagai acuan adalah $z_{original}$. Dan nilai dari x_0 akan dihilangkan.

$N = (x_7, x_9, x_{11}, x_{14})$

$B = (x_5, x_6, x_1, x_8, x_2, x_{10}, x_3, x_{12}, x_4, x_{13})$

Tabel 2.21 Menghilangkan x_0 dari tabel

$Basic$	x_7	x_9	x_{11}	x_{14}	v
$\boldsymbol{z_{original}}$	0	1	0	-1	80
x_5	0	1	0	-1	30
x_6	0	-1	0	1	20
x_1	0	1	0	-1	80
x_8	1	0	0	0	55
x_2	-1	0	0	0	55
x_{10}	0	1	0	0	40

Tabel 2.21 Menghilangkan x_0 dari tabel

$Basic$	x_7	x_9	x_{11}	x_{14}	v
x_3	0	-1	0	0	40
x_{12}	0	0	1	0	40
x_4	0	0	-1	0	40
x_{13}	0	0	0	1	0

Iterasi 0

Kemudian dipilih x_{14} sebagai *entering variable* karena memiliki nilai minimal.

Tabel 2.22 Tahap Simpleks pemilihan *leaving variable* iterasi 0

$Basic$	x_{14}	v	Δ_i
x_5	-1	30	–
x_6	1	20	20
x_1	-1	80	–
x_8	0	55	–
x_2	0	55	–
x_{10}	0	40	–
x_3	0	40	–
x_{12}	0	40	–
x_4	0	40	–
x_{13}	1	0	0

Dipilih x_{13} sebagai *leaving variable* karena memiliki nilai minimal. Kemudian akan dilakukan *pivoting* antara x_{14} dan x_{13}. Sehingga hasilnya dapat dilihat pada Tabel 2.23.

$N = (x_7, x_9, x_{11}, x_{13})$

$B = (x_5, x_6, x_1, x_8, x_2, x_{10}, x_3, x_{12}, x_4, x_{14})$

Tabel 2.23 Tahap Simpleks iterasi 0

$Basic$	x_7	x_9	x_{11}	x_{13}	v
$z_{original}$	0	1	0	1	80
x_5	0	1	0	1	30
x_6	0	−1	0	−1	20
x_1	0	1	0	1	80
x_8	1	0	0	0	55
x_2	−1	0	0	0	55
x_{10}	0	1	0	0	40
x_3	0	−1	0	0	40
x_{12}	0	0	1	0	40
x_4	0	0	−1	0	40
x_{14}	0	0	0	1	0

Karena semua koefisien pada fungsi objektif sudah bernilai positif, sehingga iterasi selesai. Dan diperoleh solusi optimal seperti berikut:

$$\textbf{\textit{If}}\ i \in B \quad \bar{x_i} = b_i \qquad (i = \{1,2,3,4,5,6,8,10,12,14\})$$

(1) $\bar{x_1} = 80$
(2) $\bar{x_2} = 55$
(3) $\bar{x_3} = 40$
(4) $\bar{x_4} = 40$
(5) $\bar{x_5} = 30$
(6) $\bar{x_6} = 20$
(7) $\bar{x_8} = 55$
(8) $\bar{x_{10}} = 40$
(9) $\bar{x_{12}} = 40$
(10) $\bar{x_{14}} = 0$

Else $\overline{x_i} = 0$	(11) $\overline{x_7} = 0$ (12) $\overline{x_9} = 0$ (13) $\overline{x_{11}} = 0$ (14) $\overline{x_{13}} = 0$

Dari hasil perhitungan di atas, didapatkan nilai maksimal dengan $z = 80$. Sehingga dapat diambil kesimpulan jika perkiraan waktu maksimal untuk menempuh perjalanan dari kota 0 ke kota 1 adalah 80 menit.

Sedangkan model dari *test case* untuk mencari waktu minimal tersebut dapat dituliskan menjadi:

$$\boldsymbol{Maximize}\ z_{original} = -x_1$$

Subject to

(1) $-x_1 \leq -50$	(6) $x_3 \leq 80$
(2) $x_1 \leq 100$	(7) $-x_4 \leq -40$
(3) $-x_2 \leq -55$	(8) $x_4 \leq 80$
(4) $x_2 \leq 110$	(9) $x_1 + x_3 \leq 120$
(5) $-x_3 \leq -40$	(10) $-x_1 - x_3 \leq -120$

Sama seperti sebelumnya, sebelum melakukan metode Simpleks, harus dipastikan terlebih dahulu apakah program linier tersebut memiliki solusi yang layak. Sehingga akan ditambahkan $-x_0$ pada sisi kiri semua kendala yang ada, dan mengganti fungsi objektif menjadi $-x_0$.

$$\boldsymbol{Maximize}\ z_{aux} = -x_0$$

Subject to

(1) $-x_1 - x_0 \leq -50$	(6) $x_3 - x_0 \leq 80$
(2) $x_1 - x_0 \leq 100$	(7) $-x_4 - x_0 \leq -40$
(3) $-x_2 - x_0 \leq -55$	(8) $x_4 - x_0 \leq 80$
(4) $x_2 - x_0 \leq 110$	(9) $x_1 + x_3 - x_0 \leq 120$
(5) $-x_3 - x_0 \leq -40$	(10) $-x_1 - x_3 - x_0 \leq -120$

Dari pemodelan di atas, akan dituliskan ke dalam bentuk *slack*.

$$\boldsymbol{z_{aux} = -x_0}$$
$$\boldsymbol{z_{original} = -x_1}$$

$x_5 = -50 + x_1 + x_0$ $\quad$ $x_{10} = 80 - x_3 + x_0$
$x_6 = 100 - x_1 + x_0$ $\quad$ $x_{11} = -40 + x_4 + x_0$
$x_7 = -55 + x_2 + x_0$ $\quad$ $x_{12} = 80 - x_4 + x_0$
$x_8 = 110 - x_2 + x_0$ $\quad$ $x_{13} = 120 - x_1 - x_3 + x_0$
$x_9 = -40 + x_3 + x_0$ $\quad$ $x_{14} = -120 + x_1 + x_3 + x_0$

Karena masih berada dalam tahapan *Initialize-Simplex*, sehingga fungsi objektif yang akan dijadikan acuan adalah fungsi objektif $\boldsymbol{Maximize}\ z_{aux} = -x_0$. Jika dituliskan ke dalam bentuk tabel dapat dilihat pada Tabel 2.24.

$N = (x_1, x_2, x_3, x_4, x_0)$

$B = (x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14})$

Tabel 2.24 Tabel Inisialisasi Simpleks Waktu Minimal

Basic	x_1	x_2	x_3	x_4	x_0	v
$\boldsymbol{z_{aux}}$	0	0	0	0	1	0
$z_{original}$	1	0	0	0	0	0
x_5	-1	0	0	0	-1	-50
x_6	1	0	0	0	-1	100
x_7	0	-1	0	0	-1	-55
x_8	0	1	0	0	-1	110
x_9	0	0	-1	0	-1	-40
x_{10}	0	0	1	0	-1	80
x_{11}	0	0	0	-1	-1	-40
x_{12}	0	0	0	1	-1	80
x_{13}	1	0	1	0	-1	120
x_{14}	-1	0	-1	0	-1	-120

Untuk tahap pertama akan dilakukan *pivoting* dengan *entering variable* = 0 dan *leaving variable* $= n + k = 4 + 10 = 14$. Hasil dari *pivoting* dapat dilihat pada Tabel 2.25.

$N = (x_1, x_2, x_3, x_4, x_{14})$

$B = (x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_0)$

Tabel 2.25 Tabel Inisialisasi Simpleks Hasil *Pivoting* Pertama

Basic	x_1	x_2	x_3	x_4	x_{14}	v
$\boldsymbol{z_{aux}}$	-1	0	-1	0	1	-120
$z_{original}$	1	0	0	0	0	0
x_5	0	0	1	0	-1	70
x_6	2	0	1	0	-1	220
x_7	1	-1	1	0	-1	65
x_8	1	1	1	0	-1	230
x_9	1	0	0	0	-1	80
x_{10}	1	0	2	0	-1	200
x_{11}	1	0	1	-1	-1	80
x_{12}	1	0	1	1	-1	200
x_{13}	2	0	2	0	-1	240
x_0	1	0	1	0	-1	120

Setelah melakukan *pivoting* di atas, akan dilanjutkan dengan perulangan pada baris 3-12 metode Simpleks hingga didapatkan hasil yang optimal dari L_{aux}.

Iterasi 0

Dari semua koefisien yang ada pada fungsi objektif $\boldsymbol{z_{aux}}$ yang memiliki nilai minimal. Sehingga akan dipilih x_1 sebagai *entering variable*. Kemudian akan dipilih *leaving variable* yang memiliki nilai Δ_i paling minimal.

Tabel 2.26 Pemilihan *leaving variable* iterasi 0

$Basic$	x_1	v	Δ_i
x_5	0	70	-
x_6	2	220	110
x_7	1	65	65
x_8	1	230	230
x_9	1	80	80
x_{10}	1	200	200
x_{11}	1	80	80
x_{12}	1	200	200
x_{13}	2	240	120
x_0	1	120	120

Dari Tabel 2.26 dapat dilihat jika nilai Δ_7 paling minimal, sehingga x_7 dipilih sebagai *leaving variable*. Kemudian akan dilakukan *pivoting* antara x_1 dan x_7. Sehingga hasilnya dapat dilihat pada Tabel 2.27.

$N = (x_7, x_2, x_3, x_4, x_{14})$

$B = (x_5, x_6, x_1, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_0)$

Tabel 2.27 Hasil *pivoting* iterasi 0

$Basic$	x_7	x_2	x_3	x_4	x_{14}	v
$\boldsymbol{z_{aux}}$	1	-1	0	0	0	-55
$z_{original}$	-1	1	-1	0	1	-65
x_5	0	0	1	0	-1	70
x_6	-2	2	-1	0	1	90
x_1	1	-1	1	0	-1	65
x_8	-1	2	0	0	0	165
x_9	-1	1	-1	0	0	15

Tabel 2.27 Hasil *pivoting* iterasi 0

$Basic$	x_7	x_2	x_3	x_4	x_{14}	v
x_{10}	-1	1	1	0	0	135
x_{11}	-1	1	0	-1	0	15
x_{12}	-1	1	0	1	0	135
x_{13}	-2	2	0	0	1	110
x_0	-1	1	0	0	0	55

Iterasi 1

Kemudian dipilih x_2 sebagai *entering variable* karena memiliki nilai minimal.

Tabel 2.28 Pemilihan *leaving variable* iterasi 1

$Basic$	x_2	v	Δ_i
x_5	0	70	–
x_6	2	90	45
x_1	-1	65	–
x_8	2	165	82.5
x_9	1	15	15
x_{10}	1	135	135
x_{11}	1	15	15
x_{12}	1	135	135
x_{13}	2	110	55
x_0	1	55	55

Dipilih x_9 sebagai *leaving variable* karena memiliki nilai minimal. Kemudian akan dilakukan *pivoting* antara x_2 dan x_9. Sehingga hasilnya dapat dilihat pada Tabel 2.29.

$N = (x_7, x_9, x_3, x_4, x_{14})$

$B = (x_5, x_6, x_1, x_8, x_2, x_{10}, x_{11}, x_{12}, x_{13}, x_0)$

Tabel 2.29 Hasil *pivoting* iterasi 1

$Basic$	x_7	x_9	x_3	x_4	x_{14}	v
$\boldsymbol{z_{aux}}$	0	1	-1	0	0	-40
$z_{original}$	0	-1	0	0	1	-80
x_5	0	0	1	0	-1	70
x_6	0	-2	1	0	1	60
x_1	0	1	0	0	-1	80
x_8	1	-2	2	0	0	135
x_2	-1	1	-1	0	0	15
x_{10}	0	-1	2	0	0	120
x_{11}	0	-1	1	-1	0	0
x_{12}	0	-1	1	1	0	120
x_{13}	0	-2	2	0	1	80
x_0	0	-1	1	0	0	40

Iterasi 2

Kemudian dipilih x_3 sebagai *entering variable* karena memiliki nilai minimal.

Tabel 2.30 Pemilihan *leaving variable* iterasi 2

$Basic$	x_3	v	Δ_i
x_5	1	70	70
x_6	1	60	60
x_1	0	80	–
x_8	2	135	82.5
x_2	-1	15	–
x_{10}	2	120	60

Tabel 2.30 Pemilihan *leaving variable* iterasi 2

Basic	x_3	v	Δ_i
x_{11}	1	0	0
x_{12}	1	120	120
x_{13}	2	80	40
x_0	1	40	40

Dipilih x_{11} sebagai *leaving variable* karena memiliki nilai minimal. Kemudian akan dilakukan *pivoting* antara x_3 dan x_{11}. Sehingga hasilnya dapat dilihat pada Tabel 2.31.

$N = (x_7, x_9, x_{11}, x_4, x_{14})$

$B = (x_5, x_6, x_1, x_8, x_2, x_{10}, x_3, x_{12}, x_{13}, x_0)$

Tabel 2.31 Hasil *pivoting* iterasi 2

Basic	x_7	x_9	x_{11}	x_4	x_{14}	v
z_{aux}	0	0	1	−1	0	−40
$z_{original}$	0	−1	0	0	1	−80
x_5	0	1	−1	1	−1	70
x_6	0	−1	−1	1	1	60
x_1	0	1	0	0	−1	80
x_8	1	0	−2	2	0	135
x_2	−1	0	1	−1	0	15
x_{10}	0	1	−2	2	0	120
x_3	0	−1	1	−1	0	0
x_{12}	0	0	−1	2	0	120
x_{13}	0	0	−2	2	1	80
x_0	0	0	−1	1	0	40

Iterasi 3

Kemudian dipilih x_4 sebagai *entering variable* karena memiliki nilai minimal.

Tabel 2.32 Pemilihan *leaving variable* iterasi 3

Basic	x_4	v	Δ_i
x_5	1	70	70
x_6	1	60	60
x_1	0	80	–
x_8	2	135	82.5
x_2	−1	15	–
x_{10}	2	120	60
x_3	−1	0	–
x_{12}	2	120	60
x_{13}	2	80	40
x_0	1	40	40

Dipilih x_{13} sebagai *leaving variable* karena memiliki nilai minimal. Kemudian akan dilakukan *pivoting* antara x_4 dan x_{13}. Sehingga hasilnya dapat dilihat pada Tabel 2.33.

$N = (x_7, x_9, x_{11}, x_{13}, x_{14})$

$B = (x_5, x_6, x_1, x_8, x_2, x_{10}, x_3, x_{12}, x_4, x_0)$

Tabel 2.33 Hasil *pivoting* iterasi 3

Basic	x_7	x_9	x_{11}	x_{13}	x_{14}	v
$\boldsymbol{z_{aux}}$	0	0	0	0.5	0.5	0
$z_{original}$	0	−1	0	0	1	−80
x_5	0	1	0	−0.5	−1.5	30
x_6	0	−1	0	−0.5	0.5	20
x_1	0	1	0	0	−1	80

Tabel 2.33 Hasil *pivoting* iterasi 3

$Basic$	x_7	x_9	x_{11}	x_{13}	x_{14}	v
x_8	1	0	0	-1	-1	55
x_2	-1	0	0	0.5	0.5	55
x_{10}	0	1	0	-1	-1	40
x_3	0	-1	0	0.5	0.5	40
x_{12}	0	0	1	-1	-1	40
x_4	0	0	-1	0.5	0.5	40
x_0	0	0	0	-0.5	-0.5	0

Hasil optimal dari L_{aux} sudah didapatkan, karena semua fungsi objektif sudah bernilai positif. Akan tetapi, karena x_0 masih berada pada *basic variable*, sehingga harus dilakukan *pivoting* satu kali lagi untuk menghilangkan x_0 dari basis.

Iterasi 4

Dipilih $\boldsymbol{x_{13}}$ sebagai ***entering variable*** dan $\boldsymbol{x_0}$ sebagai ***leaving variable***. Kemudian akan dilakukan *pivoting* antara x_{13} dan x_0. Sehingga hasilnya dapat dilihat pada Tabel 2.34.

$N = (x_7, x_9, x_{11}, x_0, x_{14})$

$B = (x_5, x_6, x_1, x_8, x_2, x_{10}, x_3, x_{12}, x_4, x_{13})$

Tabel 2.34 Hasil *pivoting* iterasi 4

$Basic$	x_7	x_9	x_{11}	x_0	x_{14}	v
$\boldsymbol{z_{aux}}$	0	0	0	1	0	0
$z_{original}$	0	-1	0	0	1	-80
x_5	0	1	0	-1	-1	30
x_6	0	-1	0	-1	1	20
x_1	0	1	0	0	-1	80
x_8	1	0	0	-2	0	55

Tabel 2.34 Hasil *pivoting* iterasi 4

$Basic$	x_7	x_9	x_{11}	x_0	x_{14}	v
x_2	-1	0	0	1	0	55
x_{10}	0	1	0	-2	0	40
x_3	0	-1	0	1	0	40
x_{12}	0	0	1	-2	0	40
x_4	0	0	-1	1	0	40
x_{13}	0	0	0	-2	1	0

Karena semua koefisien pada fungsi objektif sudah bernilai positif dan x_0 sudah menjadi *nonbasic variable*, sehingga iterasi selesai dengan hasil seperti Tabel 2.34. Solusi optimal dari L_{aux} sudah didapatkan, sehingga akan dilanjutkan dengan menjalankan metode Simpleks. Dan perulangan akan terus dilakukan selama masih terdapat koefisien pada fungsi objektif yang bernilai negatif. Fungsi objektif yang akan digunakan sebagai acuan adalah $z_{original}$. Dan nilai dari x_0 akan dihilangkan.

$N = (x_7, x_9, x_{11}, x_{14})$

$B = (x_5, x_6, x_1, x_8, x_2, x_{10}, x_3, x_{12}, x_4, x_{13})$

Tabel 2.35 Menghilangkan x_0 dari tabel

$Basic$	x_7	x_9	x_{11}	x_{14}	v
$\boldsymbol{z_{original}}$	0	-1	0	1	-80
x_5	0	1	0	-1	30
x_6	0	-1	0	1	20
x_1	0	1	0	-1	80
x_8	1	0	0	0	55
x_2	-1	0	0	0	55
x_{10}	0	1	0	0	40

Tabel 2.35 Menghilangkan x_0 dari tabel

$Basic$	x_7	x_9	x_{11}	x_{14}	v
x_3	0	-1	0	0	40
x_{12}	0	0	1	0	40
x_4	0	0	-1	0	40
x_{13}	0	0	0	1	0

Iterasi 0

Kemudian dipilih x_9 sebagai *entering variable* karena memiliki nilai minimal.

Tabel 2.36 Tahap Simpleks pemilihan *leaving variable* iterasi 3

$Basic$	x_9	v	Δ_i
x_5	1	30	30
x_6	-1	20	–
x_1	1	80	80
x_8	0	55	–
x_2	0	55	–
x_{10}	1	40	40
x_3	-1	40	–
x_{12}	0	40	–
x_4	0	40	–
x_{13}	0	0	–

Dipilih x_5 sebagai *leaving variable* karena memiliki nilai minimal. Kemudian akan dilakukan *pivoting* antara x_9 dan x_5. Sehingga hasilnya dapat dilihat pada Tabel 2.37.

$N = (x_7, x_5, x_{11}, x_{14})$

$B = (x_9, x_6, x_1, x_8, x_2, x_{10}, x_3, x_{12}, x_4, x_{13})$

Tabel 2.37 Tahap Simpleks iterasi 0

Basic	x_7	x_5	x_{11}	x_{14}	v
$z_{original}$	0	1	0	0	-50
x_9	0	1	0	-1	30
x_6	0	1	0	0	50
x_1	0	-1	0	0	50
x_8	1	0	0	0	55
x_2	-1	0	0	0	55
x_{10}	0	-1	0	1	10
x_3	0	1	0	-1	70
x_{12}	0	0	1	0	40
x_4	0	0	-1	0	40
x_{13}	0	0	0	1	0

Karena semua koefisien pada fungsi objektif sudah bernilai positif, sehingga iterasi selesai. Dan diperoleh solusi optimal seperti berikut:

If $\boldsymbol{i \in B}$ $\quad \boldsymbol{\bar{x_i} = b_i}$ $\quad (i = \{1,2,3,4,6,8,9,10,12,13\})$

(1) $\bar{x_1} = 50$
(2) $\bar{x_2} = 55$
(3) $\bar{x_3} = 70$
(4) $\bar{x_4} = 40$
(5) $\bar{x_6} = 50$
(6) $\bar{x_8} = 55$
(7) $\bar{x_9} = 30$
(8) $\bar{x_{10}} = 10$
(9) $\bar{x_{12}} = 40$
(15) $\bar{x_{13}} = 0$

Else $\overline{x_i} = \mathbf{0}$	(10) $\overline{x_5} = 0$ (11) $\overline{x_7} = 0$ (12) $\overline{x_{11}} = 0$ (13) $\overline{x_{14}} = 0$

Dari hasil perhitungan di atas, didapatkan nilai maksimal dari $z = -50$. Sehingga dapat diambil kesimpulan jika perkiraan waktu minimal untuk menempuh perjalanan dari kota 0 ke kota 1 adalah 50 menit. Keluaran dari *test case* pertama adalah 0 1 50 80, yang berarti perkiraan waktu minimal untuk menempuh perjalanan dari kota 0 ke kota 1 adalah 50 menit dan waktu maksimal 80 menit.

2.9.2 Model *Dual*

Model *dual* dibangun dari model *primal* yang sudah dibahas pada subbab 2.9.2. Sehingga model *dual* untuk mencari waktu maksimal pada *test case* pertama dapat dituliskan menjadi:

$$\textbf{\textit{Minimize}}\ z_{original} = -50y_1 + 100y_2 - 55y_3 + 110y_4 - 40y_5 + 80y_6 - 40y_7 + 80y_8 + 120y_9 - 120y_{10}$$

Subject to

(1) $-y_1 + y_2 + y_9 - y_{10} \geq 1$
(2) $-y_3 + y_4 \geq 0$
(3) $-y_5 + y_6 + y_9 - y_{10} \geq 0$
(4) $-y_7 + y_8 \geq 0$

Model *dual* di atas merupakan kasus minimalisasi, agar menjadi kasus maksimalisasi maka model *dual* tersebut akan menjadi sebagai berikut:

$$\boldsymbol{Maximize}\ z_{original} = 50y_1 - 100y_2 + 55y_3 - 110y_4 + 40y_5 - 80y_6 + 40y_7 - 80y_8 - 120y_9 + 120y_{10}$$

Subject to

(1) $y_1 - y_2 - y_9 + y_{l0} \leq -1$
(2) $y_3 - y_4 \leq 0$
(3) $y_5 - y_6 - y_9 + y_{l0} \leq 0$
(4) $y_7 - y_8 \leq 0$

Seperti sebelumnya, harus dipastikan apakah program linier tersebut memiliki solusi yang layak. Sehingga akan ditambahkan $-y_0$ pada sisi kiri semua kendala yang ada, dan mengganti fungsi objektif menjadi $-y_0$.

$$\boldsymbol{Maximize}\ z_{aux} = -y_0$$

Subject to

(1) $y_1 - y_2 - y_9 + y_{l0} - y_0 \leq -1$
(2) $y_3 - y_4 - y_0 \leq 0$
(3) $y_5 - y_6 - y_9 + y_{l0} - y_0 \leq 0$
(4) $y_7 - y_8 - y_0 \leq 0$

Dari pemodelan di atas, akan dituliskan ke dalam bentuk *slack*.

$$\boldsymbol{z_{aux} = -y_0}$$

$$\boldsymbol{z_{original} = -50y_1 + 100y_2 - 55y_3 + 110y_4 - 40y_5 + 80y_6 - 40y_7 + 80y_8 + 120y_9 - 120y_{10}}$$

$y_{11} = -1 - y_1 + y_2 + y_9 - y_{l0} + y_0$
$y_{12} = 0 - y_3 + y_4 + y_0$
$y_{13} = 0 - y_5 + y_6 + y_9 - y_{l0} + y_0$
$y_{14} = 0 - y_7 + y_8 + y_0$

Karena masih berada dalam tahapan *Initialize-Simplex*, sehingga fungsi objektif yang akan dijadikan acuan adalah fungsi

objektif $\boldsymbol{Maximize}\ z_{aux} = -y_0$. Jika dituliskan ke dalam bentuk tabel Simpleks dapat dilihat pada Tabel 2.38.

$N = (y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{10}, y_0)$

$B = (y_{11}, y_{12}, y_{13}, y_{14})$

Tabel 2.38 Tabel Inisialisasi Simpleks Waktu Maksimal Model *Dual*

B	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_0	v
$\boldsymbol{z_{aux}}$	0	0	0	0	0	0	0	0	0	0	1	0
z_{ori}	−50	100	−55	110	−40	80	−40	80	120	−120	0	0
y_{11}	1	−1	0	0	0	0	0	0	−1	1	−1	−1
y_{12}	0	0	1	−1	0	0	0	0	0	0	−1	0
y_{13}	0	0	0	0	1	−1	0	0	−1	1	−1	0
y_{14}	0	0	0	0	0	0	1	−1	0	0	−1	0

Untuk tahap pertama akan dilakukan *pivoting* dengan *entering variable* = 0 dan *leaving variable* $= n + k = 10 + 1 = 11$. Hasil dari *pivoting* dapat dilihat pada Tabel 2.39.

$N = (y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{10}, y_{11})$

$B = (y_0, y_{12}, y_{13}, y_{14})$

Tabel 2.39 Tabel Inisialisasi Simpleks Hasil *Pivoting* Pertama Model *Dual*

B	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	v
$\boldsymbol{z_{aux}}$	1	−1	0	0	0	0	0	0	−1	1	1	−1
z_{ori}	−50	100	−55	110	−40	80	−40	80	120	−120	0	0
y_0	−1	1	0	0	0	0	0	0	1	−1	−1	1
y_{12}	−1	1	1	−1	0	0	0	0	1	−1	−1	1
y_{13}	−1	1	0	0	1	−1	0	0	0	0	−1	1
y_{14}	−1	1	0	0	0	0	1	−1	1	−1	−1	1

Setelah melakukan *pivoting* di atas, akan dilanjutkan dengan perulangan pada baris 3-12 metode Simpleks hingga didapatkan hasil yang optimal dari L_{aux}.

Iterasi 0

Dari semua koefisien yang ada pada fungsi objektif $\boldsymbol{z_{aux}}$ yang memiliki nilai minimal. Sehingga akan dipilih y_2 sebagai *entering variable*. Kemudian akan dipilih *leaving variable* yang memiliki nilai Δ_i paling minimal.

Tabel 2.40 Pemilihan *leaving variable* iterasi 0 model *dual*

$Basic$	y_2	v	Δ_i
y_0	1	1	1
y_{12}	1	1	1
y_{13}	1	1	1
y_{14}	1	1	1

Dari Tabel 2.40 dapat dilihat jika nilai Δ_0 paling minimal, sehingga y_0 dipilih sebagai *leaving variable*. Kemudian akan dilakukan *pivoting* antara y_2 dan y_0. Sehingga hasilnya dapat dilihat pada Tabel 2.41.

$N = (y_1, y_0, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{10}, y_{11})$

$B = (y_2, y_{12}, y_{13}, y_{14})$

Tabel 2.41 Hasil *pivoting* iterasi 0 model *dual*

B	y_1	y_0	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	v
$\boldsymbol{z_{aux}}$	0	1	0	0	0	0	0	0	0	0	0	0
z_{ori}	50	−100	−55	110	−40	80	−40	80	20	−20	100	−100
y_2	−1	1	0	0	0	0	0	0	1	−1	−1	1
y_{12}	0	−1	1	−1	0	0	0	0	0	0	0	0
y_{13}	0	−1	0	0	1	−1	0	0	−1	1	0	0
y_{14}	0	−1	0	0	0	0	1	−1	0	0	0	0

Karena semua koefisien pada fungsi objektif sudah bernilai positif dan y_0 sudah menjadi *nonbasic variable*, maka iterasi selesai dengan hasil seperti Tabel 2.41. Solusi optimal dari L_{aux} sudah didapatkan, sehingga akan dilanjutkan dengan menjalankan metode Simpleks. Dan perulangan akan terus dilakukan selama masih terdapat koefisien pada fungsi objektif yang bernilai negatif. Fungsi objektif yang akan digunakan sebagai acuan adalah $z_{original}$. Dan nilai dari y_0 akan dihilangkan.

$N = (y_1, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{10}, y_{11})$

$B = (y_2, y_{12}, y_{13}, y_{14})$

Tabel 2.42 Menghilangkan x_0 dari tabel model *dual*

B	y_1	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	v
$\boldsymbol{z_{ori}}$	50	-55	110	-40	80	-40	80	20	-20	100	-100
y_2	-1	0	0	0	0	0	0	1	-1	-1	1
y_{12}	0	1	-1	0	0	0	0	0	0	0	0
y_{13}	0	0	0	1	-1	0	0	-1	1	0	0
y_{14}	0	0	0	0	0	1	-1	0	0	0	0

Iterasi 0

Kemudian dipilih y_3 sebagai *entering variable* karena memiliki nilai minimal.

Tabel 2.43 Tahap Simpleks pemilihan *leaving variable* iterasi 0 model *dual*

$Basic$	y_3	v	Δ_i
y_2	0	1	–
y_{12}	1	0	0
y_{13}	0	0	–
y_{14}	0	0	–

Dipilih y_{12} sebagai *leaving variable* karena memiliki nilai minimal. Kemudian akan dilakukan *pivoting* antara y_3 dan y_{12}. Sehingga hasilnya dapat dilihat pada Tabel 2.44.

$N = (y_1, y_{12}, y_4, y_5, y_6, y_7, y_8, y_9, y_{10}, y_{11})$

$B = (y_2, y_3, y_{13}, y_{14})$

Tabel 2.44 Tahap Simpleks iterasi 0 model *dual*

B	y_1	y_{12}	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	v
z_{ori}	50	55	55	−40	80	−40	80	20	−20	100	−100
y_2	−1	0	0	0	0	0	0	1	−1	−1	1
y_3	0	1	−1	0	0	0	0	0	0	0	0
y_{13}	0	0	0	1	−1	0	0	−1	1	0	0
y_{14}	0	0	0	0	0	1	−1	0	0	0	0

Iterasi 1

Kemudian dipilih y_5 sebagai *entering variable* karena memiliki nilai minimal.

Tabel 2.45 Tahap Simpleks pemilihan *leaving variable* iterasi 1 model *dual*

$Basic$	y_5	v	Δ_i
y_2	0	1	–
y_3	0	0	–
y_{13}	1	0	0
y_{14}	0	0	–

Dipilih y_{13} sebagai *leaving variable* karena memiliki nilai minimal. Kemudian akan dilakukan *pivoting* antara y_5 dan y_{13}. Sehingga hasilnya dapat dilihat pada Tabel 2.46.

$N = (y_1, y_{12}, y_4, y_{13}, y_6, y_7, y_8, y_9, y_{10}, y_{11})$

$B = (y_2, y_3, y_5, y_{14})$

Tabel 2.46 Tahap Simpleks iterasi 1 model *dual*

B	y_1	y_{12}	y_4	y_{13}	y_6	y_7	y_8	y_9	y_{10}	y_{11}	v
z_{ori}	50	55	55	40	40	−40	80	−20	20	100	−100
y_2	−1	0	0	0	0	0	0	1	−1	−1	1
y_3	0	1	−1	0	0	0	0	0	0	0	0
y_5	0	0	0	1	−1	0	0	−1	1	0	0
y_{14}	0	0	0	0	0	1	−1	0	0	0	0

Iterasi 2

Kemudian dipilih y_7 sebagai *entering variable* karena memiliki nilai minimal.

Tabel 2.47 Tahap Simpleks pemilihan *leaving variable* iterasi 2 model *dual*

$Basic$	y_7	v	Δ_i
y_2	0	1	–
y_3	0	0	–
y_5	0	0	–
y_{14}	1	0	0

Dipilih y_{14} sebagai *leaving variable* karena memiliki nilai minimal. Kemudian akan dilakukan *pivoting* antara y_7 dan y_{14}. Sehingga hasilnya dapat dilihat pada Tabel 2.48.

$N = (y_1, y_{12}, y_4, y_{13}, y_6, y_{14}, y_8, y_9, y_{10}, y_{11})$

$B = (y_2, y_3, y_5, y_7)$

Tabel 2.48 Tahap Simpleks iterasi 2 model *dual*

B	y_1	y_{12}	y_4	y_{13}	y_6	y_{14}	y_8	y_9	y_{10}	y_{11}	v
z_{ori}	50	55	55	40	40	40	40	−20	20	100	−100
y_2	−1	0	0	0	0	0	0	1	−1	−1	1
y_3	0	1	−1	0	0	0	0	0	0	0	0

Tabel 2.48 Tahap Simpleks iterasi 2 model *dual*

B	y_1	y_{12}	y_4	y_{13}	y_6	y_{14}	y_8	y_9	y_{10}	y_{11}	v
y_5	0	0	0	1	-1	0	0	-1	1	0	0
y_7	0	0	0	0	0	1	-1	0	0	0	0

Iterasi 3

Kemudian dipilih y_9 sebagai *entering variable* karena memiliki nilai minimal.

Tabel 2.49 Tahap Simpleks pemilihan *leaving variable* iterasi 3 model *dual*

$Basic$	y_9	v	Δ_i
y_2	1	1	1
y_3	0	0	–
y_5	-1	0	–
y_7	0	0	–

Dipilih y_2 sebagai *leaving variable* karena memiliki nilai minimal. Kemudian akan dilakukan *pivoting* antara y_9 dan y_2. Sehingga hasilnya dapat dilihat pada Tabel 2.50.

$N = (y_1, y_{12}, y_4, y_{13}, y_6, y_{14}, y_8, y_2, y_{10}, y_{11})$

$B = (y_9, y_3, y_5, y_7)$

Tabel 2.50 Tahap Simpleks iterasi 3 model *dual*

B	y_1	y_{12}	y_4	y_{13}	y_6	y_{14}	y_8	y_2	y_{10}	y_{11}	v
z_{ori}	30	55	55	40	40	40	40	20	0	80	-80
y_9	-1	0	0	0	0	0	0	1	-1	-1	1
y_3	0	1	-1	0	0	0	0	0	0	0	0
y_5	-1	0	0	1	-1	0	0	1	0	-1	1
y_7	0	0	0	0	0	1	-1	0	0	0	0

Karena semua koefisien pada fungsi objektif sudah bernilai positif, sehingga iterasi selesai. Dari hasil perhitungan di atas, didapatkan nilai maksimal dengan $z = 80$. Sehingga dapat diambil kesimpulan jika perkiraan waktu maksimal untuk menempuh perjalanan dari kota 0 ke kota 1 adalah 80 menit.

Sedangkan model *dual* dari *test case* untuk mencari waktu minimal tersebut dapat dituliskan menjadi:

$$\boldsymbol{Minimize}\ z_{original} = -50y_1 + 100y_2 - 55y_3 + 110y_4 - 40y_5 + 80y_6 - 40y_7 + 80y_8 + 120y_9 - 120y_{10}$$

Subject to

(1) $-y_1 + y_2 + y_9 - y_{10} \geq -1$
(2) $-y_3 + y_4 \geq 0$
(3) $-y_5 + y_6 + y_9 - y_{10} \geq 0$
(4) $-y_7 + y_8 \geq 0$

Model *dual* di atas merupakan kasus minimalisasi, agar menjadi kasus maksimalisasi maka model *dual* tersebut akan menjadi sebagai berikut:

$$\boldsymbol{Maximize}\ z_{original} = 50y_1 - 100y_2 + 55y_3 - 110y_4 + 40y_5 - 80y_6 + 40y_7 - 80y_8 - 120y_9 + 120y_{10}$$

Subject to

(1) $y_1 - y_2 - y_9 + y_{10} \leq 1$
(2) $y_3 - y_4 \leq 0$
(3) $y_5 - y_6 - y_9 + y_{10} \leq 0$
(4) $y_7 - y_8 \leq 0$

Karena semua nilai $b \geq 0$, maka proses *Initialize-Simplex* akan mengembalikan nilai $(N, B, A, b, c, 0)$ dan dilanjutkan dengan proses Simpleks. Model tersebut jika dituliskan ke dalam bentuk tabel Simpleks dapat dilihat pada Tabel 2.51.

$N = (y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{10})$

$B = (y_{11}, y_{12}, y_{13}, y_{14})$

Tabel 2.51 Tabel Simpleks Waktu Minimal Model *Dual*

B	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	v
z_{ori}	-50	100	-55	110	-40	80	-40	80	120	-120	0
y_{11}	1	-1	0	0	0	0	0	0	-1	1	1
y_{12}	0	0	1	-1	0	0	0	0	0	0	0
y_{13}	0	0	0	0	1	-1	0	0	-1	1	0
y_{14}	0	0	0	0	0	0	1	-1	0	0	0

Iterasi 0

Kemudian dipilih y_{10} sebagai *entering variable* karena memiliki nilai minimal.

Tabel 2.52 Tahap Simpleks pemilihan *leaving variable* iterasi 0 model *dual*

$Basic$	y_{10}	v	Δ_i
y_{11}	1	1	1
y_{12}	0	0	–
y_{13}	1	0	0
y_{14}	0	0	–

Dipilih y_{13} sebagai *leaving variable* karena memiliki nilai minimal. Kemudian akan dilakukan *pivoting* antara y_{10} dan y_{13}. Sehingga hasilnya dapat dilihat pada Tabel 2.53.

$N = (y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{13})$

$B = (y_{11}, y_{12}, y_{10}, y_{14})$

Tabel 2.53 Tahap Simpleks iterasi 0 model *dual*

B	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{13}	v
z_{ori}	-50	100	-55	110	80	-40	-40	80	0	120	0
y_{11}	1	-1	0	0	-1	1	0	0	0	-1	1

Tabel 2.53 Tahap Simpleks iterasi 0 model *dual*

B	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{13}	v
y_{12}	0	0	1	−1	0	0	0	0	0	0	0
y_{10}	0	0	0	0	1	−1	0	0	−1	1	0
y_{14}	0	0	0	0	0	0	1	−1	0	0	0

Iterasi 1

Kemudian dipilih y_3 sebagai *entering variable* karena memiliki nilai minimal.

Tabel 2.54 Tahap Simpleks pemilihan *leaving variable* iterasi 1 model *dual*

$Basic$	y_3	v	Δ_i
y_{11}	0	1	–
y_{12}	1	0	0
y_{10}	0	0	–
y_{14}	0	0	–

Dipilih y_{12} sebagai *leaving variable* karena memiliki nilai minimal. Kemudian akan dilakukan *pivoting* antara y_3 dan y_{12}. Sehingga hasilnya dapat dilihat pada Tabel 2.55.

$N = (y_1, y_2, y_{12}, y_4, y_5, y_6, y_7, y_8, y_9, y_{13})$

$B = (y_{11}, y_3, y_{10}, y_{14})$

Tabel 2.55 Tahap Simpleks iterasi 1 model *dual*

B	y_1	y_2	y_{12}	y_4	y_5	y_6	y_7	y_8	y_9	y_{13}	v
$\boldsymbol{z_{ori}}$	−50	100	55	55	80	−40	−40	80	0	120	0
y_{11}	1	−1	0	0	−1	1	0	0	0	−1	1
y_3	0	0	1	−1	0	0	0	0	0	0	0
y_{10}	0	0	0	0	1	−1	0	0	−1	1	0
y_{14}	0	0	0	0	0	0	1	−1	0	0	0

Iterasi 2

Kemudian dipilih y_1 sebagai *entering variable* karena memiliki nilai minimal.

Tabel 2.56 Tahap Simpleks pemilihan *leaving variable* iterasi 2 model *dual*

$Basic$	y_1	v	Δ_i
y_{11}	1	1	1
y_3	0	0	–
y_{10}	0	0	–
y_{14}	0	0	–

Dipilih y_{11} sebagai *leaving variable* karena memiliki nilai minimal. Kemudian akan dilakukan *pivoting* antara y_1 dan y_{11}. Sehingga hasilnya dapat dilihat pada Tabel 2.57.

$N = (y_{11}, y_2, y_{12}, y_4, y_5, y_6, y_7, y_8, y_9, y_{13})$

$B = (y_1, y_3, y_{10}, y_{14})$

Tabel 2.57 Tahap Simpleks iterasi 2 model *dual*

B	y_{11}	y_2	y_{12}	y_4	y_5	y_6	y_7	y_8	y_9	y_{13}	v
$\boldsymbol{z_{ori}}$	50	50	55	55	30	10	−40	80	0	70	50
y_1	1	−1	0	0	−1	1	0	0	0	−1	1
y_3	0	0	1	−1	0	0	0	0	0	0	0
y_{10}	0	0	0	0	1	−1	0	0	−1	1	0
y_{14}	0	0	0	0	0	0	1	−1	0	0	0

Iterasi 3

Kemudian dipilih y_7 sebagai *entering variable* karena memiliki nilai minimal.

Tabel 2.58 Tahap Simpleks pemilihan *leaving variable* iterasi 3 model *dual*

$Basic$	y_7	v	Δ_i
y_1	0	1	–
y_3	0	0	–
y_{10}	0	0	–
y_{14}	1	0	0

Dipilih y_{11} sebagai *leaving variable* karena memiliki nilai minimal. Kemudian akan dilakukan *pivoting* antara y_7 dan y_{14}. Sehingga hasilnya dapat dilihat pada Tabel 2.59.

$N = (y_{11}, y_2, y_{12}, y_4, y_5, y_6, y_{14}, y_8, y_9, y_{13})$

$B = (y_1, y_3, y_{10}, y_7)$

Tabel 2.59 Tahap Simpleks iterasi 3 model *dual*

B	y_{11}	y_2	y_{12}	y_4	y_5	y_6	y_{14}	y_8	y_9	y_{13}	v
$\boldsymbol{z_{ori}}$	50	50	55	55	30	10	40	40	0	70	50
y_1	1	−1	0	0	−1	1	0	0	0	−1	1
y_3	0	0	1	−1	0	0	0	0	0	0	0
y_{10}	0	0	0	0	1	−1	0	0	−1	1	0
y_7	0	0	0	0	0	0	1	−1	0	0	0

Karena semua koefisien pada fungsi objektif sudah bernilai positif, sehingga iterasi selesai. Dari hasil perhitungan di atas, didapatkan nilai maksimal dari $z = 50$. Sehingga dapat diambil kesimpulan jika perkiraan waktu minimal untuk menempuh perjalanan dari kota 0 ke kota 1 adalah 50 menit. Keluaran dari *test case* pertama adalah 0 1 50 80, yang berarti perkiraan waktu minimal untuk menempuh perjalanan dari kota 0 ke kota 1 adalah 50 menit dan waktu maksimal 80 menit.

[*Halaman ini sengaja dikosongkan*]

BAB III
PERANCANGAN DAN ANALISIS

Pada bagian ini akan dijelaskan analisis dan desain sistem yang digunakan untuk menyelesaikan permasalahan pada Tugas Akhir ini.

3.1 Definisi Umum Sistem

Dalam menyelesaikan permasalahan FN16ROAD akan diimplementasikan 2 model, yaitu model *primal* dan model *dual*. Sehingga akan terdapat 2 program yang berbeda.

3.1.1 Model *Primal*

Pada model *primal* sistem akan menerima masukan berupa satu buah bilangan bulat n yang merepresentasikan jumlah kota, dengan penomoran kota dimulai dari 0 hingga $n - 1$. Kemudian sistem akan menerima masukan sejumlah n baris dan setiap baris akan berisi n bilangan bulat sehingga membentuk sebuah matriks persegi $n \times n$. Matriks $n \times n$ tersebut merepresentasikan jarak (dalam km) ketika melakukan perjalanan dari kota i menuju kota j. Jarak tersebut akan bernilai -1 jika tidak ada jalan yang menghubungkan kedua kota tersebut, dan akan bernilai 0 jika merupakan jarak dari kota i menuju kota i itu sendiri.

Setelah menerima masukan matriks $n \times n$, sistem akan menyediakan 3 matriks baru. Matriks yang pertama untuk memberikan indeks pada perjalanan antar kota yang dapat dilakukan berdasarkan masukan dari pengguna. Sedangkan matriks yang kedua berisi jarak antar kota untuk membantu dalam mencari jalur alternatif. Matriks ketiga berfungsi untuk menampung indeks dari kota yang bisa dilewati untuk mendapatkan jalur altematif.

Selanjutnya sistem akan menyimpan masukan dari pengguna berupa data perjalanan antar kota yang dapat dilakukan pada matriks A dan data waktu yang diperlukan pada vektor B. Pada saat melakukan penyimpanan data, sistem akan menyesuaikan dengan pemodelan permasalahan FN16ROAD yang

sudah dibahas pada subbab 2.5 dan 2.9.1. Data-data tersebut akan menjadi kendala awal dari permasalahan FN16ROAD.

Kemudian sistem akan menerima masukan berupa satu buah bilangan bulat q yang merepresentasikan jumlah riwayat perjalanan yang sudah pernah dilakukan. Pada q baris selanjutnya, terdapat 3 bilangan bulat s, e, dan x dimana s dan e merupakan kota asal dan kota tujuan. Sedangkan x adalah waktu yang diperlukan untuk melakukan perjalanan dari s menuju e (dalam menit). Masukan tersebut juga akan disimpan dalam matriks A dan vektor B sesuai dengan pemodelan FN16ROAD.

Setelah semua data kendala tersimpan, sistem akan menerima masukan berupa satu buah bilangan bulat q yang merepresentasikan jumlah perjalanan yang akan dilakukan. Kemudian pada q baris selanjutnya, sistem akan menerima masukan 2 buah bilangan bulat s dan e yang merupakan kota asal dan kota tujuan. Kemudian sistem akan memanggil fungsi solve yang ada pada Simplex.

Dari permasalahan di atas akan dihasilkan keluaran berupa satu baris untuk setiap q berisi 4 bilangan. Bilangan tersebut merepresentasikan kota asal, kota tujuan, perkiraan waktu minimal dan perkiraan waktu maksimal yang diperlukan pada setiap perjalanan yang dilakukan. *Pseudocode* dari fungsi *main* model *primal* ditunjukan pada Gambar 3.1.

main ( )

```
1.  while n ← input
2.        m ← 0
3.        for i ← 0 to n
4.              for j ← 0 to n
5.                    a[i][j] ← input
6.                    if a[i][j] > 0
7.                          idx[i][j] ← m++
8.                          adj[i][j] ← a[i][j]
9.                    else
10.                         idx[i][j] ← −1
```

Gambar 3.1 *Pseudocode* Fungsi *Main* Model *Primal* FN16ROAD

```
11.                         adj[i][j] ← 1e9
12.       memset(trk, −1, sizeof(trk))
13.       for i ← 0 to n
14.             for j ← 0 to n
15.                   for k ← 0 to n
16.                         if adj[j][k] > adj[j][i] +
                             adj[i][k]
17.                               adj[j][k] ← adj[j][i] +
                                   adj[i][k]
18.                               trk[j][k] ← i
19.       for i ← 0 to n
20.             for j ← 0 to n
21.                   if idx[i][j] ≠ −1
22.                         p1[idx[i][j]] ← −1
23.                         A ← p1
24.                         B ← −a[i][j]
25.                         p1[idx[i][j]] ← 1
26.                         A ← p1
27.                         B ← 2 ∗ a[i][j]
28.       q ← input
29.             while q − −
30.                   s, e, x ← input
31.                   if s ← e then continue
32.                   l ← spath(s, e)
33.                   for auto i : l do p1[i] ← 1
34.                   A ← p1
35.                   B ← x
36.                   for auto i : l do p1[i] ← −1
37.                   A ← p1
38.                   B ← −x
39.       q ← input
40.             while q − −
41.                   s, e ← input
42.                   if s ← e
43.                         s, e, 0,0 → output
```

Gambar 3.1 *Pseudocode* Fungsi *Main* Model *Primal* FN16ROAD

```
44.                 continue
45.                 l ← spath(s, e)
46.                 s, e → output
47.                 for auto i : l do x[i] ← −1
48.                 −Simplex(A, B, x). solve(aux) →
                    output
49.                 for auto i : l do x[i] ← 1
50.                 Simplex(A, B, x). solve(aux) → output
51.                       return 0
```

Gambar 3.1 *Pseudocode* Fungsi *Main* Model *Primal* FN16ROAD

3.1.2 Model *Dual*

Masukan dan keluaran sistem pada model *dual* sama dengan masukan dan keluaran pada model *primal*. Hanya saja pada model *dual* terdapat perbedaan pada saat melakukan penyimpanan data masukan. Penyimpanan tersebut menyesuaikan aturan dalam membangun model *dual* dari model *primal* seperti yang sudah dibahas pada subbab 2.7 dan 2.9.2.

Pada fungsi *main* model *dual* akan ditambahkan satu matriks baru *ctnnew* untuk menyimpan koefisien pada sisi kiri (*left-hand*-side) dari kendala model *dual* yang didefinisikan untuk setiap variabel model *primal*. Vektor *B* menyimpan koefisien pada fungsi objektif dari model *dual* yang meruapakan sisi kanan (*right-hand-side*) dari kendala model *primal*. Sedangkan vektor x menyimpan koefisien pada sisi kanan (*right-hand-side*) dari model *dual* yang merupakan koefisien pada fungsi objektif dari model *primal*. *Pseudocode* dari fungsi *main* model *dual* ditunjukan pada Gambar 3.2.

```
main ( )
1.  while n ← input
2.        m ← 0
3.        for i ← 0 to n
4.              for j ← 0 to n
5.                    a[i][j] ← input
```

Gambar 3.2 *Pseudocode* Fungsi *Main* Model *Dual* FN16ROAD

```
6.                  if a[i][j] > 0
7.                        idx[i][j] ← m + +
8.                        adj[i][j] ← a[i][j]
9.                  else
10.                       idx[i][j] ← −1
11.                       adj[i][j] ← 1e9
12.       memset(trk, −1, sizeof(trk))
13.       for i ← 0 to n
14.            for j ← 0 to n
15.                 for k ← 0 to n
16.                       if adj[j][k] > adj[j][i] +
                           adj[i][k]
17.                             adj[j][k] ← adj[j][i] +
                                 adj[i][k]
18.                             trk[j][k] ← i
19.       for i ← 0 to n
20.            for j ← 0 to n
21.                 if idx[i][j] ≠ −1
22.                       p1[idx[i][j]] ← −1
23.                       A ← p1
24.                       B ← a[i][j]
25.                       p1[idx[i][j]] ← 1
26.                       A ← p1
27.                       B ← −2 ∗ a[i][j]
28.       q ← input
29.            while q − −
30.                 s, e, x ← input
31.                 if s ← e then continue
32.                 l ← spath(s, e)
33.                 for auto i : l do p1[i] ← 1
34.                 A ← p1
35.                 B ← −x
36.                 for auto i : l do p1[i] ← −1
37.                 A ← p1
38.                 B ← x
39.       q ← input
```

Gambar 3.2 *Pseudocode* Fungsi *Main* Model *Dual* FN16ROAD

```
40.          while q − −
41.                s, e ← input
42.                if s ← e
43.                      s, e, 0,0 → output
44.                      continue
45.                l ← spath(s, e)
46.                for i ← 0 to m
47.                      for j ← 0 to A.size()
48.                            tmpnew[j] ← −1 ∗ A[j][i]
49.                      ctnnew ← tmpnew
50.                s, e → output
51.                for auto i : l do x[i] ← 1
52.                Simplex(ctnnew, x, B). solve(aux) →
                   output
53.                for auto i : l do x[i] ← −1
54.                −Simplex(ctnnew, x, B).solve (aux) →
                   output
55. return 0
```

Gambar 3.2 *Pseudocode* Fungsi *Main* Model *Dual* FN16ROAD

3.2 Desain Algoritma

Sistem akan menggunakan algoritma Simpleks seperti yang sudah dijelaskan pada subbab 2.8. Hanya saja untuk diimplementasikan ke dalam sebuah kode harus dilakukan penyesuaian. Algoritma yang ada pada sistem merupakan hasil implementasi dari *pseudocode* yang ada pada Gambar 2.5-2.7.

3.2.1 Desain *Struct Simplex*

Struct Simplex digunakan untuk merepresentasikan algoritma Simpleks. Dimana pada *struct Simplex* terdapat atribut, konstruktor, dan fungsi-fungsi yang diperlukan oleh algoritma Simpleks. Desain dari *Struct Simplex* dapat dilihat pada Gambar 3.3.

Simplex ()

```
1.  Simplex(A, b, c) : m(b.size()),
    n(c.size()), N(n+1), B(m), D(m+2, VD(n+2))
2.      for i ← 0 to m
3.          for j ← 0 to n
4.              D[i][j] ← A[i][j]
5.      for i ← 0 to m
6.          B[i] ← n + i
7.          D[i][n] ← −1
8.          D[i][n + 1] ← b[i]
9.      for j ← 0 to n
10.         N[j] ← j
11.         D[m][j] ← −c[j]
12.     N[n] ← −1
13.     D[m + 1][n] ← 1
14.
15.     Pivot(r, s)
16.         inv ← 1/D[r][s]
17.         for i ← 0 to m + 2
18.             for j ← 0 to n + 2
19.                 if i ≠ r and j ≠ s
20.                     D[i][j] ← D[i][j] − D[r][j] *
                        D[i][s] * inv
21.         for i ← 0 to m + 2
22.             if i ≠ r
23.                 D[i][s] ← D[i][s] * −inv
24.         for j ← 0 to n + 2
25.             if j ≠ s
26.                 D[r][j] ← D[r][j] * inv
27.         D[r][s] ← inv
28.         swap(r, s)
29.
30.     Phase(p)
31.         x ← m + p
32.         while true
33.             s ← −1
```

Gambar 3.3 *Pseudocode Struct Simplex* FN16ROAD

```
34.             for j ← 0  to n
35.                 if !p  and N[j] ← −1
36.                     continue
37.                 if s ← −1 or (D[x][j] < D[x][s])
                    then s ← j
38.             if D[x][s] > −(1e − 9)
39.                 return true
40.             r ← −1
41.             for i ← 0  to m
42.                 if D[i][s] ≤ (1e − 9)
43.                     continue
44.                 if r ← −1 or
                    (D[i][n + 1]/D[i][s] <
                     D[r][n + 1]/D[r][s])
45.                     r ← i
46.             if r ← −1 return false
47.             Pivot(r, s)
48.
49.     solve(x)
50.         r ← 0
51.         for i ← 1 to m
52.             if D[i][n + 1] < D[r][n + 1]
53.                 r ← i
54.         if D[r][n + 1] < −(1e − 9)
55.             Pivot(r, n)
56.             if !Phase(1) or D[m + 1][n + 1] <
                −(1e − 9) then return − 1/0.0
57.             for i ← 0 to m
58.                 if B[i] ← −1
59.                     s ←
                        min_element(D[i].begin,
                        D[i].end −1)−D[i].begin
60.                     Pivot(i, s)
61.         if (!Phase(0)) then return 1/0.0
62.         x ← n
```

Gambar 3.3 *Pseudocode Struct Simplex* FN16ROAD

```
63.         for i ← 0 to m
64.             if B[i] < n
65.                 x[B[i]] ← D[i][n + 1]
66.         return D[m][n + 1]
```

Gambar 3.3 *Pseudocode Struct Simplex* FN16ROAD

Pada *Struct Simplex* di atas, terdapat konstruktor yang berfungsi untuk menciptakan sebuah matriks besar yang akan digunakan selama *Struct Simplex* dijalankan. Matriks tersebut merupakan gabungan dari kendala, fungsi objektif, konstanta, dan juga fungsi objektif z_{aux} yang akan digunakan untuk melakukan pengecekan apakah model yang ada layak untuk dikerjakan.

Pada baris 2-4 dilakukan duplikasi matriks A yang berisi jarak antar kota. Pada baris 5-8 akan dilakukan penambahan variabel x_0 yang bernilai -1 pada sisi kanan, dan kemudian ditambahkan juga nilai konstanta semua kendala dan fungsi objektif pada sisi kanan x_0. Sedangkan pada baris 9-11 akan ditambahkan fungsi objektif asli yang menjadi permasalahan utama pada sisi bawah matriks. Kemudian pada baris 12-13 akan ditambahkan fungsi objektif dari z_{aux} di bawahnya. Setelah semua digabungkan, akan terbentuk matriks D yang akan digunakan selama proses algoritma Simpleks berjalan.

Setelah konstruktor, terdapat 3 fungsi yang juga merupakan bagian dari *Struct Simplex*. Untuk lebih jelasnya, akan dibahas pada beberapa subbab berikut.

3.2.1.1 Desain Fungsi *Pivot*

Fungsi *Pivot* adalah fungsi yang digunakan untuk melakukan pertukaran peran antara *entering variable* dengan *leaving variable*. Fungsi ini merupakan hasil implementasi dari Gambar 2.5, akan tetapi dilakukan modifikasi mengikuti keperluan sistem. Pada sistem baik kendala, fungsi objektif, maupun nilai konstan di simpan ke dalam satu buah matriks besar. Sehingga proses yang dijalankan oleh fungsi menjadi lebih sederhana.

Gambar 3.4 memperlihatkan proses perhitungan sebagai berikut. Pada baris ke 2-8 akan dihitung nilai dari koefisien pada persamaan yang bukan merupakan *basic variable* baru seperti pada baris ke 7-17 Gambar 2.5. Sedangkan pada baris ke 9-12 akan dihitung nilai koefisien pada persamaan yang merupakan *basic variable* baru seperti pada baris 1-6 Gambar 2.5. Kemudian setelah semua koefisien baru selesai dihitung, posisi dari *entering variable* dan *leaving variable* pada *basic variable* dan *nonbasic variable* akan ditukar, seperti pada baris 19-20 Gambar 2.5. *Pseudocode* dari fungsi *pivot* ditunjukan pada Gambar 3.4.

Pivot ()

Input : *leaving variable (r), entering variable (s)*

```
1.  inv ← 1/D[r][s]
2.  for i ← 0  to m + 2
3.        for j ← 0  to n + 2
4.              if i ≠ r and j ≠ s
5.                    D[i][j] ← D[i][j] − D[r][j] * D[i][s] *
                      inv
6.  for i ← 0  to m + 2
7.        if i ≠ r
8.              D[i][s] ← D[i][s] * −inv
9.  for j ← 0  to n + 2
10.       if j ≠ s
11.             D[r][j] ← D[r][j] * inv
12. D[r][s] ← inv
13. swap(r, s)
```

Gambar 3.4 *Pseudocode* Fungsi *Pivot* FN16ROAD

3.2.1.2 Desain Fungsi *Phase*

Pengecekan apakah kondisi optimal dari permasalahan sudah tercapai atau belum akan dilakukan pada fungsi *Phase*. Fungsi ini juga akan menentukan fungsi objektif mana yang akan dijadikan sebagai acuan. Apakah masih dalam tahap inisialisasi Simpleks atau sudah pada tahap Simpleks.

Baris 1 digunakan untuk menentukan fungsi objektif mana yang akan dijadikan acuan. Perulangan pada baris 2 akan terus dilakukan hingga kondisi optimal didapatkan. Pada baris 4-7 akan dicari nilai minimal pada fungsi objektif yang menjadi acuan. Baris 8-9 akan melakukan pengecekan apakah nilai minimal tersebut masih memenuhi syarat untuk menjadi *entering variable*. Jika masih memenuhi syarat, maka akan dilanjutkan untuk mencari *leaving variable* pada baris 11-14. Setelah berhasil mendapatkan *entering* dan *leaving variable* akan dilakukan pemanggilan fungsi Pivot. *Pseudocode* dari fungsi *Phase* ditunjukan pada Gambar 3.5.

Phase ()

Input : *boolean p*

```
1.  x ← m + p
2.  while true
3.        s ← −1
4.        for j ← 0  to n
5.              if !p and N[j] ← −1
6.                    continue
7.              if s ← −1 or (D[x][j] < D[x][s]) then
                s ← j
8.        if D[x][s] > −(1e − 9)
9.              return true
10.       r ← −1
11.       for i ← 0  to m
12.             if D[i][s] ≤ (1e − 9)
13.                   continue
14.             if   r ← −1   or   (D[i][n + 1]/D[i][s] <
                D[r][n + 1]/D[r][s])
15.                   r ← i
16.       if r ← −1
17.             return false
18.       Pivot(r, s)
```

Gambar 3.5 *Pseudocode* Fungsi *Phase* FN16ROAD

3.2.1.3 Desain Fungsi *Solve*

Fungsi *Solve* ini merupakan inti dari algoritma Simpleks yang diterapkan pada sistem. Fungsi ini mencakup tahapan yang ada pada *pseudocode Intialize-Simplex* pada Gambar 2.7 dan *pseudocode Simplex* pada Gambar 2.6. Pada baris 2-4 akan dicari *leaving variable* pertama seperti pada baris 6 *pseudocode Intialize-Simplex*. Sedangkan yang menjadi *entering variable* adalah indeks matriks dimana x_0 berada, yaitu pada n. Setelah *leaving* dan *entering variable* didapatkan, akan dilakukan pemanggilan fungsi *Pivot*.

Kemudian pada baris 7 dilakukan pengecekan apakah model yang ada layak untuk dikerjakan. Pada baris 7 juga dilakukan pemanggilan fungsi *Phase*. *Phase*(1) menunjukkan jika saat ini masih berada pada tahap inisialisasi Simpleks. Fungsi *Phase* tersebut akan mengembalikan nilai *true* jika hasil optimal sudah didapatkan. Hal tersebut berarti proses inisialisasi Simpleks selesai dilakukan.

Baris 8-10 akan mengeluarkan x_0 dari *basic variable* apabila x_0 masih berada pada *basic variable*. Baris 12 akan memanggil fungsi *Phase* kembali. *Phase*(0) menunjukkan jika saat ini sudah memasuki tahap Simpleks. Fungsi *Phase* tersebut akan mengembalikan nilai *true* jika hasil optimal sudah didapatkan. Kemudian pada baris 14-16 akan mencari hasil optimal dari semua *nonbasic variable*. Baris 17 akan mengembalikan nilai optimal dari fungsi objektif utama yang merupakan keluaran dari sistem ini, yaitu perkiraan waktu minimal dan perkiraan waktu maksimal. *Pseudocode* dari fungsi *Solve* ditunjukan pada Gambar 3.6.

Solve ()

```
Input : vector x
1.  r ← 0
2.  for i ← 1 to m
3.      if D[i][n + 1] < D[r][n + 1]
4.          r ← i
```

Gambar 3.6 *Pseudocode* Fungsi *Solve* FN16ROAD

```
5.  if D[r][n + 1] < −(1e − 9)
6.        Pivot(r, n)
7.        if  !Phase(1)  or  D[m + 1][n + 1] < −(1e − 9)
          then return −1/0.0
8.        for i ← 0 to m
9.              if B[i] ← −1
10.                   s ← min_element(D[i].begin,
                      D[i].end −1)−D[i].begin
11.                   Pivot(i, s)
12. if (!Phase(0)) then return 1/0.0
13. x ← n
14. for i ← 0 to m
15.       if B[i] < n
16.             x[B[i]] ← D[i][n + 1]
17. return D[m][n + 1]
```

Gambar 3.6 *Pseudocode* Fungsi *Solve* FN16ROAD

3.2.2 Desain Fungsi *Spath*

Fungsi *Spath* ini akan mengecek apakah dari kota s menuju kota e terdapat jalan yang menghubungkannya secara langsung. Jika tidak ada, maka akan dicari jalan mana yang harus dilewati agar bisa melakukan perjalanan dari kota s menuju kota e. *Pseudocode* dari fungsi *Spath* ditunjukan pada Gambar 3.7.

```
spath()
Input : kota asal (s), kota tujuan (e)
1.  if trk[s][e] ← −1
2.        ret ← idx[s][e]
3.        return ret
4.  l ← spath(s, trk[s][e])
5.  r ← spath(trk[s][e], e)
6.  for auto i : r
7.        l ← i
8.  return l
```

Gambar 3.7 *Pseudocode* Fungsi *Spath* FN16ROAD

[*Halaman ini sengaja dikosongkan*]

BAB IV
IMPLEMENTASI

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

4.1 Lingkungan Implementasi

Digunakan lingkungan implementasi dengan spesifikasi perangkat lunak dan perangkat keras seperti terlihat pada Tabel 4.1.

Tabel 4.1 Spesifikasi Lingkungan Implementasi

No.	Jenis Perangkat	Spesifikasi
1	Perangkat Keras	• *Processor* Intel Core i7-6700HQ CPU @ 2.60GHz • *Memory* 12GB DDR4
2	Perangkat Lunak	• Sistem operasi Windows 10 Pro 64-bit • Bahasa pemrograman C++ • *Integrated Development Environment* Dev-C++ 5.11

4.2 Rancangan Data

Pada subbab ini dijelaskan mengenai desain data masukan yang diperlukan untuk melakukan proses algoritma serta data keluaran yang dihasilkan oleh program.

4.2.1 Data Masukan

Data masukan adalah data yang akan diproses oleh program sebagai masukan dengan menggunakan algoritma dan struktur data yang telah dirancang dalam penyelesaian

permasalahan FN16ROAD ini. Data masukan dari persoalan ini dapat dilihat di bawah ini.

1. Masukan berupa n dengan rentang $1 \leq n \leq 30$ yang merepresentasikan jumlah kota, dengan penomoran kota dimulai dari 0 hingga $n - 1$.
2. Terdapat n baris yang masing-masing berisi n bilangan bulat yang menentukan jarak (dalam km) antar kota: nilai ke-j pada baris ke-i merepresentasikan jarak ketika melakukan perjalanan dari kota i menuju kota j.
3. Masukan selanjutnya adalah satu bilangan bulat r dengan rentang $1 \leq r \leq 100$ yang merepresentasikan jumlah dari riwayat perjalanan yang ada.
4. Pada r baris selanjutnya terdapat 3 bilangan bulat s, d, dan t, di mana s dan d merupakan kota asal dan kota tujuan dan t adalah waktu yang diperlukan untuk melakukan perjalanan dari s menuju d (dalam menit).
5. Kemudian terdapat satu bilangan bulat q dengan rentang $1 \leq q \leq 100$ yang merepresentasikan jumlah dari perjalanan yang akan dilakukan.
6. Selanjutnya terdapat q baris yang berisi 2 bilangan bulat s dan d yang merupakan kota asal dan tujuan yang akan dilakukan.

Berikut contoh masukan pada permasalahan (Gambar 4.1).

```
3
0 50 -1
55 0 40
-1 40 0
1
0 2 120
3
0 1
1 2
1 0
```

Gambar 4.1 Format Masukan Permasalahan FN16ROAD

4.2.2 Data Keluaran

Data keluaran yang dihasilkan oleh program berupa satu baris untuk setiap *query* yang berisi kota asal dan tujuan dari *query* tersebut, diikuti oleh perkiraan waktu minimal dan waktu maksimal untuk melakukan perjalanan dari kota asal ke kota tujuan pada *query* tersebut. Berikut contoh keluaran pada permasalahan (Gambar 4.2).

```
0 1 50.0 80.0
1 2 40.0 70.0
1 0 55.0 110.0
```

Gambar 4.2 Format Keluaran Permasalahan FN16ROAD

4.3 Penggunaan *Library*, Konstanta, dan Variabel Global

Pada subbab ini dijelaskan mengenai *library*, *template*, konstanta, dan variabel global yang digunakan dalam sistem. Pada Kode Sumber 4.1, terdapat 1 *library* yang digunakan yaitu *bits/stdc++.h*. Didefinisikan $real\_t$ dan ld sebagai tipe data *double*. Sedangkan VD didefinisikan sebagai *vector<real_t>* atau bisa juga ditulis sebagai *vector<double>*. VVD didefinisikan sebagai *vector<VD>* atau bisa juga ditulis sebagai *vector<vector<double>>*. Selanjutnya didefinisikan konstanta *real_t EPS* yang bernilai $1e^{-9}$. Tabel 4.2 menampilkan variabel-variabel global yang akan digunakan dalam implementasi program.

```
1.  #include<bits/stdc++.h>
2.
3.  using namespace std;
4.  using real_t = double;
5.  using ld = double;
6.  using VD = vector<real_t>;
7.  using VVD = vector<VD>;
8.  const real_t EPS = 1e-9;
9.  int n, m, a[33][33];
```

Kode Sumber 4.1 Potongan Kode Implementasi *Library*, Konstanta, dan Variabel Global

```
10. int adj[33][33], trk[33][33];
11. int idx[33][33];
```

Kode Sumber 4.1 Potongan Kode Implementasi *Library*, Konstanta, dan Variabel Global

Tabel 4.2 Daftar Variabel Global Permasalahan FN16ROAD

No	Nama Variabel	Tipe	Penjelasam
1	*n*	int	Digunakan untuk menyimpan jumlah kota
2	*m*	int	Digunakan untuk menyimpan jumlah kota yang memiliki jalan penghubung
3	*a*	int[]	Digunakan untuk menyimpan data jarak antar kota
4	*adj*	int[]	Digunakan untuk menyimpan jarak antar kota untuk membantu mencari jalur alternatif
5	*trk*	int[]	Digunakan untuk menampung indeks dari kota yang bisa dilewati untuk mendapatkan jalur alternatif
6	*idx*	int[]	Digunakan untuk memberikan indeks pada perjalanan antar kota yang dapat dilakukan berdasarkan pada masukan dari pengguna

4.4 Implementasi Fungsi Main

Dalam menyelesaikan permasalahan FN16ROAD akan diimplementasikan 2 model, yaitu model *primal* dan model *dual*. Sehingga akan terdapat 2 program yang berbeda. Perbedaan antara kedua program tersebut terdapat pada fungsi *main*.

4.4.1 Model *Primal*

Fungsi *Main* pada model *primal* diimplementasikan sesuai *pseudocode* pada Subbab 3.1.1. Kode sumber fungsi *Main* model *primal* ditunjukan pada Kode Sumber 4.2.

```
1.  int main() {
2.      cin.sync_with_stdio(0);
3.      cin.tie(0);
4.      while(cin >> n){
5.          m = 0;
6.          for(int i=0; i<n; i++){
7.              for(int j=0; j<n; j++){
8.                  cin >> a[i][j];
9.                  if(a[i][j] > 0){
10.                     idx[i][j] = m++;
11.                     adj[i][j] = a[i][j];
12.                 }
13.                 else{
14.                     idx[i][j] = -1;
15.                     adj[i][j] = 1e9;
16.                 }
17.             }
18.         }
19.         memset(trk, -1, sizeof(trk));
20.         for(int i=0; i<n; i++){
21.             for(int j=0; j<n; j++){
22.                 for(int k=0; k<n; k++){
23.                     if(adj[j][k] > adj[j][i]
                        + adj[i][k]){
24.                         adj[j][k] =
                            adj[j][i] +
                            adj[i][k];
25.                         trk[j][k] = i;
26.                     }
```

Kode Sumber 4.2 Potongan Kode Implementasi Fungsi *Main* Model *Primal* Permasalahan FN16ROAD

```
27.                 }
28.             }
29.         }
30.         vector<vector<real_t>> A;
31.         vector<real_t> B;
32.         for(int i=0; i<n; i++){
33.             for(int j=0; j<n; j++){
34.                 if(idx[i][j] != -1){
35.                     vector<real_t> p1(m);
36.                     p1[idx[i][j]] = -1;
37.                     A.push_back(p1);
38.                     B.push_back(-a[i][j]);
39.                     p1[idx[i][j]] = 1;
40.                     A.push_back(p1);
41.                     B.push_back(2 * a[i][j]);
42.                 }
43.             }
44.         }
45.         int q; cin >> q;
46.         while(q--){
47.             int s, e, x;
48.             cin >> s >> e >> x;
49.             if(s == e) continue;
50.             auto l = spath(s, e);
51.             vector<real_t> p1(m);
52.             for(auto &i : l) p1[i] = 1;
53.             A.push_back(p1);
54.             B.push_back(x);
55.             for(auto &i : l) p1[i] = -1;
56.             A.push_back(p1);
57.             B.push_back(-x);
58.         }
59.         vector<real_t> aux;
60.         cin >> q;
61.         while(q--){
```

Kode Sumber 4.2 Potongan Kode Implementasi Fungsi *Main* Model *Primal* Permasalahan FN16ROAD

```
62.             int s, e;
63.             cin >> s >> e;
64.             if(s == e){
65.                 printf("%d %d 0 0\n", s, e);
66.                 continue;
67.             }
68.             auto l = spath(s, e);
69.             printf("%d %d ", s, e);
70.             vector<real_t> x(m);
71.             for(auto &i : l) x[i] = -1;
72.             printf("%.10f ", -Simplex(A, B,
                x).solve(aux));
73.             for(auto &i : l) x[i] = 1;
74.             printf("%.10f\n", Simplex(A, B,
                x).solve(aux));
75.         }
76.     }
77.     return 0;
78. }
```

Kode Sumber 4.2 Potongan Kode Implementasi Fungsi *Main* Model *Primal* Permasalahan FN16ROAD

4.4.2 Model *Dual*

Fungsi *Main* pada model *dual* diimplementasikan sesuai *pseudocode* pada Subbab 3.1.2. Kode sumber fungsi *Main* model *dual* ditunjukan pada Kode Sumber 4.3.

```
1.  int main() {
2.      cin.sync_with_stdio(0);
3.      cin.tie(0);
4.      while(cin >> n){
5.          m = 0;
6.          for(int i=0; i<n; i++){
7.              for(int j=0; j<n; j++){
8.                  cin >> a[i][j];
```

Kode Sumber 4.3 Potongan Kode Implementasi Fungsi *Main* Model *Dual* Permasalahan FN16ROAD

```
9.                  if(a[i][j] > 0){
10.                     idx[i][j] = m++;
11.                     adj[i][j] = a[i][j];
12.                 }
13.                 else{
14.                     idx[i][j] = -1;
15.                     adj[i][j] = 1e9;
16.                 }
17.             }
18.         }
19.         memset(trk, -1, sizeof(trk));
20.         for(int i=0; i<n; i++){
21.             for(int j=0; j<n; j++){
22.                 for(int k=0; k<n; k++){
23.                     if(adj[j][k] > adj[j][i] +
                        adj[i][k]){
24.                         adj[j][k] = adj[j][i]
                            + adj[i][k];
25.                         trk[j][k] = i;
26.                     }
27.                 }
28.             }
29.         }
30.         vector<vector<real_t>> A;
31.         vector<real_t> B;
32.         for(int i=0; i<n; i++){
33.             for(int j=0; j<n; j++){
34.                 if(idx[i][j] != -1){
35.                     vector<real_t> p1(m);
36.                     p1[idx[i][j]] = -1;
37.                     A.push_back(p1);
38.                     B.push_back(a[i][j]);
39.                     p1[idx[i][j]] = 1;
40.                     A.push_back(p1);
41.                     B.push_back(-2 * a[i][j]);
```

Kode Sumber 4.3 Potongan Kode Implementasi Fungsi *Main* Model *Dual* Permasalahan FN16ROAD

```
42.                 }
43.             }
44.         }
45.         int q; cin >> q;
46.         while(q--){
47.             int s, e, x;
48.             cin >> s >> e >> x;
49.             if(s == e) continue;
50.             auto l = spath(s, e);
51.             vector<real_t> p1(m);
52.             for(auto &i : l) p1[i] = 1;
53.             A.push_back(p1);
54.             B.push_back(-x);
55.
56.             for(auto &i : l) p1[i] = -1;
57.             A.push_back(p1);
58.             B.push_back(x);
59.         }
60.         vector<real_t> aux;
61.         cin >> q;
62.         while(q--){
63.             int s, e;
64.             cin >> s >> e;
65.             if(s == e){
66.                 printf("%d %d 0 0\n", s, e);
67.                 continue;
68.         }
69.         auto l = spath(s, e);
70.         vector<real_t> x(m);
71.         vector<vector<real_t>> ctnnew;
72.         vector<real_t> tmpnew(A.size());
73.         for(int i=0; i<m; i++){
74.             for(int j=0; j<A.size(); j++){
75.                 tmpnew[j]=(-1)*A[j][i];
76.             }
```

Kode Sumber 4.3 Potongan Kode Implementasi Fungsi *Main* Model *Dual* Permasalahan FN16ROAD

```
77.             ctnnew.push_back(tmpnew);
78.         }
79.         printf("%d %d ", s, e);
80.         for(auto &i : l) x[i] = 1;
81.         printf("%.10f ", Simplex(ctnnew, x,
                B).solve(aux));
82.         for(auto &i : l) x[i] = -1;
83.         printf("%.10f\n", -Simplex(ctnnew, x,
                B).solve(aux));
84.         }
85.     }
86.     return 0;
87. }
```

Kode Sumber 4.3 Potongan Kode Implementasi Fungsi *Main* Model *Dual* Permasalahan FN16ROAD

4.5 Implementasi Algoritma

Pada bagian ini akan dijelaskan mengenai implementasi dari subbab 3.2 yang dibagi menjadi beberapa subbab sebagai berikut.

4.5.1 Implementasi *Struct Simplex*

Struct Simplex digunakan untuk merepresentasikan algoritma Smpleks. Dimana pada *struct simplex* terdapat atribut, konstruktor, dan fungsi-fungsi yang diperlukan oleh algoritma Simpleks seperti yang sudah dijelaskan pada *pseudocode* Gambar 3.3. Implementasi dari *Struct Simplex* dapat dilihat pada Kode Sumber 4.4.

```
1.  struct Simplex{
2.      int m, n;
3.      vector<int> B, N;
4.      VVD D;
```

Kode Sumber 4.4 Potongan Kode Implementasi *Struct Simplex* Permasalahan FN16ROAD

```
5.    Simplex(const VVD& A, const VD& b, const
      VD &c)
6.        : m(b.size()), n(c.size()), N(n+1),
          B(m), D(m+2, VD(n+2)){
7.            for(int i=0; i<m; ++i) for(int
              j=0; j<n; ++j) D[i][j] = A[i][j];
8.            for(int i=0; i<m; ++i) B[i] =
              n+i, D[i][n] = -1, D[i][n+1] =
              b[i];
9.            for(int j=0; j<n; ++j) N[j] = j,
              D[m][j] = -c[j];
10.           N[n] = -1; D[m+1][n] = 1;
11.       }
12.   void Pivot(int r, int s) {
13.       real_t inv = 1/D[r][s];
14.       for(int i=0; i<m+2; ++i){
15.           for(int j=0; j<n+2; ++j){
16.               if(i != r && j != s) D[i][j]
                  -= D[r][j] * D[i][s] * inv;
17.           }
18.       }
19.       for(int i=0; i<m+2; ++i) if(i != r)
          D[i][s] *= -inv;
20.       for(int j=0; j<n+2; ++j) if(j != s)
          D[r][j] *= inv;
21.       D[r][s] = inv; swap(B[r], N[s]);
22.   }
23.   bool Phase(bool p) {
24.       int x = m + p;
25.       while(true) {
26.           int s = -1;
27.           for(int j=0; j<=n; ++j){
28.              if(!p && N[j] == -1) continue;
29.              if(s == -1 || D[x][j] <
                 D[x][s]) s = j;
```

Kode Sumber 4.4 Potongan Kode Implementasi *Struct Simplex* Permasalahan FN16ROAD

```
30.             }
31.             if(D[x][s] > -EPS) return true;
32.             int r = -1;
33.             for(int i=0; i<m; ++i){
34.                 if(D[i][s] <= EPS) continue;
35.                 if(r == -1 || D[i][n+1] /
                    D[i][s] < D[r][n+1] / D[r][s])
                    r = i;
36.             }
37.             if(r == -1) return false;
38.             Pivot(r, s);
39.         }
40.     }
41.     real_t solve(VD &x) {
42.         int r = 0;
43.         for(int i=1; i<m; ++i) if(D[i][n+1] <
            D[r][n+1]) r=i;
44.         if(D[r][n+1] < -EPS) {
45.             Pivot(r, n);
46.             if(!Phase(1) || D[m+1][n+1] < -
                EPS) return -1/0.0;
47.             for(int i=0; i<m; ++i) if(B[i] ==
                -1) {
48.                  int s =
                     min_element(D[i].begin(),
                     D[i].end() - 1) -
                     D[i].begin();
49.                  Pivot(i, s);
50.             }}
51.         if(!Phase(0)) return 1/0.0;
52.         x = VD(n);
53.         for(int i=0; i<m; ++i) if(B[i] < n)
            x[B[i]] = D[i][n+1];
54.         return D[m][n+1];
55.     }};
```

Kode Sumber 4.4 Potongan Kode Implementasi *Struct Simplex* Permasalahan FN16ROAD

4.5.1.1 Implementasi Fungsi *Pivot*

Fungsi *Pivot* adalah fungsi yang digunakan untuk melakukan pertukaran peran antara *entering variable* dengan *leaving variable* seperti yang sudah dijelaskan pada *pseudocode* Gambar 3.4. Implementasi dari fungsi ini terdapat di dalam *Struct Simplex* pada Kode Sumber 4.4. Untuk lebih jelasnya dapat dilihat pada Kode Sumber 4.5.

```
1.  void Pivot(int r, int s) {
2.     real_t inv = 1/D[r][s];
3.     for(int i=0; i<m+2; ++i){
4.         for(int j=0; j<n+2; ++j){
5.           if(i != r && j != s) D[i][j] -=
             D[r][j] * D[i][s] * inv;
6.         }
7.     }
8.     for(int i=0; i<m+2; ++i) if(i != r)
       D[i][s] *= -inv;
9.     for(int j=0; j<n+2; ++j) if(j != s)
       D[r][j] *= inv;
10.    D[r][s] = inv; swap(B[r], N[s]);
11. }
```

Kode Sumber 4.5 Implementasi Fungsi *Pivot* Permasalahan FN16ROAD

4.5.1.2 Implementasi Fungsi *Phase*

Fungsi *Phase* ini yang akan melakukan pengecekan apakah kondisi optimal dari permasalahan sudah tercapai atau belum. Fungsi ini juga akan menentukan fungsi objektif mana yang akan dijadikan sebagai acuan. Apakah masih dalam tahap inisialisasi Simpleks atau sudah pada tahap Simpleks seperti yang dijelaskan pada *pseudocode* pada Gambar 3.5. Implementasi dari fungsi ini terdapat di dalam *Struct Simplex* pada Kode Sumber 4.4. Untuk lebih jelasnya dapat dilihat pada Kode Sumber 4.6.

```
1.  bool Phase(bool p) {
2.     int x = m + p;
3.     while(true) {
4.         int s = -1;
5.         for(int j=0; j<=n; ++j){
6.             if(!p && N[j] == -1) continue;
7.             if(s == -1 || D[x][j] < D[x][s])
               s = j;
8.         }
9.         if(D[x][s] > -EPS) return true;
10.        int r = -1;
11.        for(int i=0; i<m; ++i){
12.            if(D[i][s] <= EPS) continue;
13.            if(r == -1 || D[i][n+1] / D[i][s]
               < D[r][n+1] / D[r][s]) r = i;
14.        }
15.        if(r == -1) return false;
16.        Pivot(r, s);
17.    }
18. }
```

Kode Sumber 4.6 Implementasi Fungsi *Phase* Permasalahan FN16ROAD

4.5.1.3 Implementasi Fungsi *Solve*

Fungsi *Solve* ini merupakan inti dari algoritma Simpleks yang diterapkan pada sistem. Fungsi ini mencakup tahapan yang ada pada *pseudocode Intialize-Simplex* pada Gambar 2.7 dan *pseudocode Simplex* pada Gambar 2.6 seperti yang dijelaskan pada *pseudocode* pada Gambar 3.6. Implementasi dari fungsi ini terdapat di dalam *Struct Simplex* pada Kode Sumber 4.4. Untuk lebih jelasnya dapat dilihat pada Kode Sumber 4.7.

```
1.  real_t solve(VD &x) {
2.     int r = 0;
3.     for(int i=1; i<m; ++i) if(D[i][n+1] <
       D[r][n+1]) r=i;
4.     if(D[r][n+1] < -EPS) {
5.         Pivot(r, n);
6.         if(!Phase(1) || D[m+1][n+1] < -EPS)
           return -1/0.0;
7.         for(int i=0; i<m; ++i) if(B[i] == -1)
           {
8.             int s = min_element(D[i].begin(),
               D[i].end() - 1) - D[i].begin();
9.         Pivot(i, s);
10.        }
11.    }
12.    if(!Phase(0)) return 1/0.0;
13.    x = VD(n);
14.    for(int i=0; i<m; ++i) if(B[i] < n)
         x[B[i]] = D[i][n+1];
15.    return D[m][n+1];
16. }
```

Kode Sumber 4.7 Implementasi Fungsi *Solve* Permasalahan FN16ROAD

4.5.2 Implementasi Fungsi *Spath*

Fungsi *Spath* ini akan mengecek apakah dari kota *s* menuju kota *e* terdapat jalan yang menghubungkan secara langsung. Jika tidak ada, maka akan dicari jalan mana yang harus dilewati agar bisa melakukan perjalanan dari kota *s* menuju kota *e* seperti yang dijelaskan pada *pseudocode* pada Gambar 3.7. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 4.8.

```
1.  vector<int> spath(int s, int e){
2.     if(trk[s][e] == -1){
3.         vector<int> ret = {idx[s][e]};
4.         return ret;
5.     }
6.     auto l = spath(s, trk[s][e]);
7.     auto r = spath(trk[s][e], e);
8.     for(auto &i : r) l.push_back(i),
9.     return l;
10. }
```

Kode Sumber 4.8 Implementasi Fungsi *Spath* Permasalahan FN16ROAD

BAB V
UJICOBA DAN EVALUASI

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

5.1 Lingkungan Uji Coba

Digunakan lingkungan uji coba dengan spesifikasi perangkat lunak dan perangkat keras seperti terlihat pada Tabel 5.1.

Tabel 5.1 Spesifikasi Lingkungan Ujicoba

No.	Jenis Perangkat	Spesifikasi
1	Perangkat Keras	• *Processor* Intel Core i7-6700HQ CPU @ 2.60GHz • *Memory* 12GB DDR4
2	Perangkat Lunak	• Sistem operasi Windows 10 Pro 64-bit • Bahasa pemrograman C++ • *Integrated Development Environment* Dev-C++ 5.11

5.2 Skenario Uji Coba

Pada subbab ini akan dijelaskan skenario yang akan digunakan untuk melakukan pengujian terhadap implementasi yang dibuat untuk menyelesaikan permasalahan FN16ROAD. Skenario uji coba terdiri dari uji coba kebenaran dan uji coba kinerja.

5.2.1 Uji Coba Kebenaran

Uji coba kebenaran dibagi menjadi 2 tahap, tahap yang pertama yaitu akan dilakukan pengiriman kode sumber ke dalam situs penilaian *online* SPOJ. Sedangkan tahap kedua yaitu akan dilakukan perbandingan antara hasil keluaran sistem dengan hasil uji coba kasus sederhana yang sudah dibahas pada subbab 2.9.

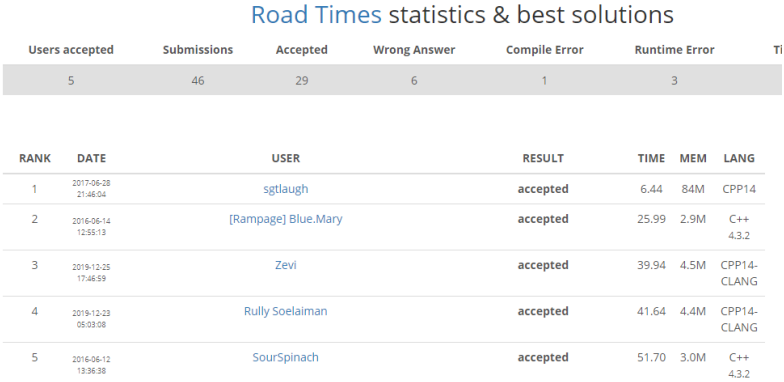
5.2.1.1 Mengirimkan Kode Sumber

Uji coba kebenaran yang pertama dilakukan dengan mengirimkan kode sumber ke dalam situs penilaian *online* SPOJ. Hasil uji coba dengan waktu terbaik pada situs penilaian *online* SPOJ ditunjukkan pada Gambar 5.1 untuk model *primal* dan Gambar 5.2 untuk model *dual*.

Dari hasil uji coba kebenaran yang dilakukan pada situs penilaian online SPOJ, dapat dilihat bahwa kode sumber mendapat keluaran *Accepted*. Waktu terbaik yang dibutuhkan oleh program adalah 41.68 detik untuk model *primal* dan 39.94 untuk model *dual* dengan memori yang dibutuhkan sebesar 4.5 M baik untuk model *primal* maupun model *dual*. Peringkat akhir dari hasil uji coba kebenaran pertama pada model *primal* dan model *dual* dapat dilihat pada Gambar 5.3.

Gambar 5.1 Hasil Uji Coba Menggunakan SPOJ Model *Primal*

Gambar 5.2 Hasil Uji Coba Menggunakan SPOJ Model *Dual*



Gambar 5.3 Peringkat Hasil Uji Coba Menggunakan SPOJ

5.2.1.2 Membandingkan Hasil Uji Coba

Uji coba selanjutnya adalah akan dibandingkan antara hasil uji coba dengan kasus sederhana yang sudah dibahas pada Subbab 2.9 dengan hasil uji coba keluaran sistem. Perbandingan akan dilakukan baik pada model *primal* maupun model *dual*. Untuk melakukan pengecekan pada setiap tahapan yang dilakukan oleh sistem, maka sistem akan sedikit dimodifikasi agar dapat mengeluarkan hasil setiap iterasi yang dilakukan. Kemudian hasil dari setiap iterasi tersebut akan dibandingkan dengan hasil dari setiap iterasi hasil uji coba pada Subbab 2.9.

Kasus sederhana yang digunakan pada uji coba adalah kasus yang sama seperti yang sudah dibahas pada Subbab 2.9. Kasus sederhana tersebut dapat dilihat pada Gambar 5.4.

```
3
0 50 -1
55 0 40
-1 40 0
1
0 2 120
1
0 1
```

Gambar 5.4 Format Masukan Uji Kebenaran

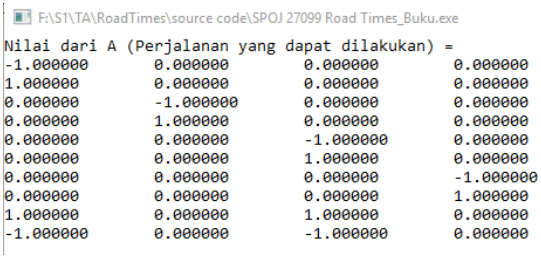
Pada kasus sederhana tersebut terdapat 3 kota yang terhubung oleh sebuah jalan satu arah, dengan jarak antar kota seperti yang ada pada Tabel 5.2. Diberikan juga 1 riwayat perjalanan, di mana untuk menempuh perjalanan dari kota 0 menuju kota 2 membutuhkan waktu 120 menit. Kemudian ditanyakan berapa perkiraan waktu minimal dan waktu maksimal yang dibutuhkan untuk menempuh perjalanan dari kota 0 menuju kota 1.

Tabel 5.2 Data Jarak Antar Kota Uji Coba Kebenaran

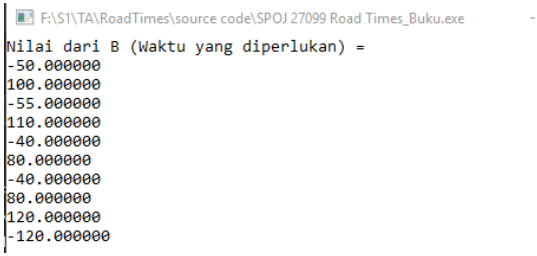
	Kota 0	Kota 1	Kota 2
Kota 0	0	50	-1
Kota 1	55	0	40
Kota 2	-1	40	0

5.2.1.2.1 Model *Primal*

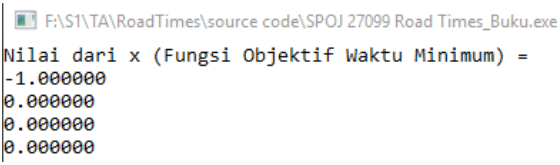
Pada model *primal* masukan dari pengguna tersebut dapat dimodelkan menjadi beberapa pertidaksamaan linier seperti yang sudah dijelaskan pada Subbab 2.9.1. Setelah menerima masukan dari pengguna, sistem akan memodelkan masukan tersebut menjadi beberapa kendala dan fungsi objektif. Setiap kendala terdiri dari beberapa koefisien dan satu buah konstanta. Pada permasalahan ini, koefisien tersebut merepresentasikan perjalanan yang dapat dilakukan. Sedangkan konstanta merepresentasikan waktu yang diperlukan untuk melakukan perjalanan. Hasil dari pemodelan yang dibuat oleh sistem dapat dilihat pada Gambar 5.5-5.8.



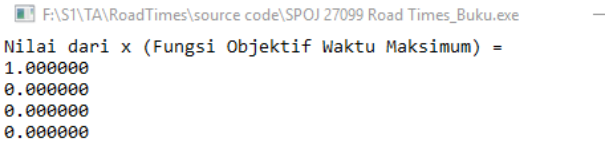
Gambar 5.5 Hasil dari *A* pada pemodelan sistem model *primal*



Gambar 5.6 Hasil dari *B* pada pemodelan sistem model *primal*



Gambar 5.7 Hasil dari x kasus minimalisasi pada pemodelan sistem model *primal*



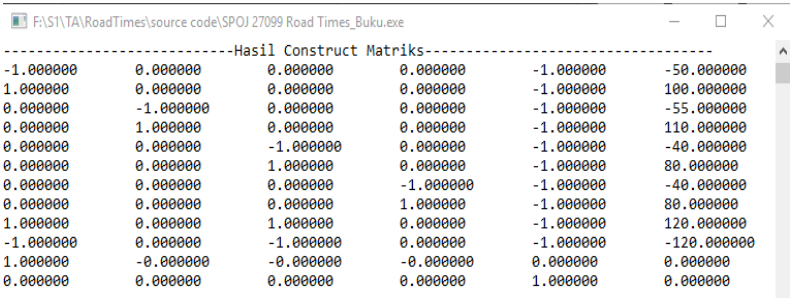
Gambar 5.8 Hasil dari x kasus maksimalisasi pada pemodelan sistem model *primal*

Model yang dihasilkan sistem sama dengan model yang dihasilkan pada uji coba yang dilakukan secara manual pada subbab 2.9.1. Kemudian sistem akan menjalankan algoritma Simpleks yang sudah diimplementasikan untuk mencari perkiraan waktu minimal dan maksimal.

Proses pertama yang dilakukan oleh sistem adalah membuat sebuah matriks baru yang menggabungkan kendala dan fungsi objektif pada Gambar 5.5-5.8. Kemudian akan ditambahkan fungsi objektif pembantu ($z_{aux} = -x_0$) dan $-x_0$ yang bernilai -1 pada semua kendala yang ada pada matriks tersebut untuk melakukan tahap inisialisasi Simpleks.

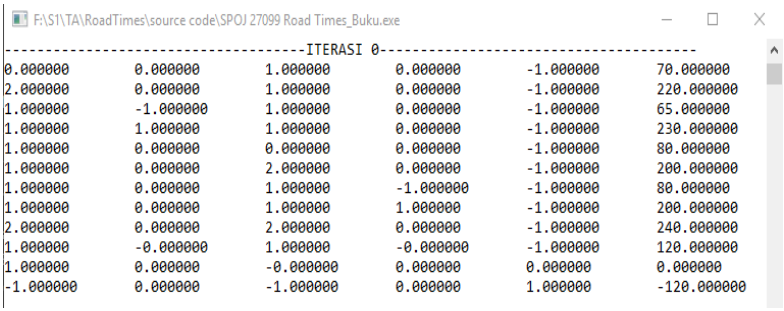
1. Mencari Waktu Minimal

Matriks yang dibuat oleh sistem pada kasus mencari waktu minimal dapat dilihat pada Gambar 5.9. Hasil dari matriks tersebut sama dengan tabel Simpleks yang ada pada Subbab 2.9.1 Tabel 2.24. Hanya saja, pada matriks tersebut fungsi objektif diletakkan pada 2 baris terbawah. Fungsi objektif utama ada pada baris 11, sedangkan fungsi objektif pembantu (z_{aux}) ada pada baris 12.



Gambar 5.9 *Construct* matriks kasus minimalisasi pada pemodelan sistem model *primal*

Kemudian sistem akan melakukan *pivoting* pertama. Hasil dari *pivoting* tersebut dapat dilihat pada Gambar 5.10. Hasil tersebut sama dengan tabel hasil dari *pivoting* pertama yang sudah dilakukan pada Subbab 2.9.1 Tabel 2.25.

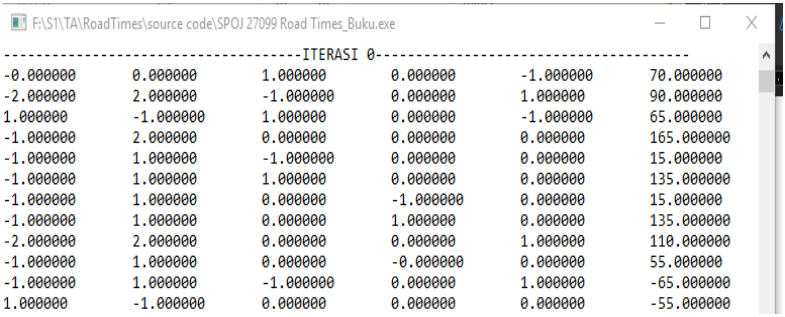


Gambar 5.10 Hasil *pivoting* pertama kasus minimalisasi pada pemodelan sistem model *primal*

Tahap selanjutnya adalah akan dilakukan beberapa iterasi hingga didapatkan hasil optimal dari fungsi objektif z_{aux}. Hasil dari beberapa iterasi yang dilakukan oleh sistem adalah sebagai berikut:

a. Iterasi 0 tahap inisialisasi Simpleks

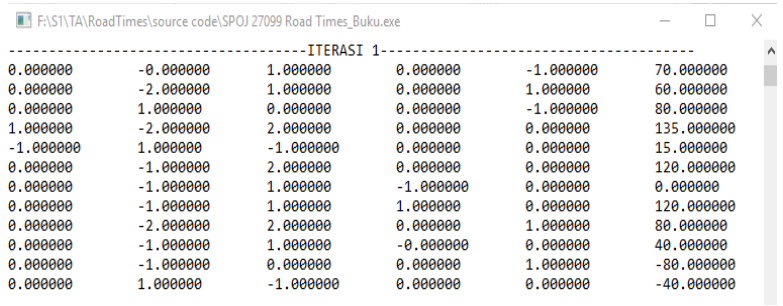
Hasil iterasi 0 pada sistem dapat dilihat pada Gambar 5.11. Dimana hasil tersebut sama dengan hasil iterasi 0 yang ada pada Subbab 2.9.1 Tabel 2.27.



Gambar 5.11 Iterasi 0 kasus minimalisasi pada pemodelan sistem model *primal*

b. Iterasi 1 tahap inisialisasi Simpleks

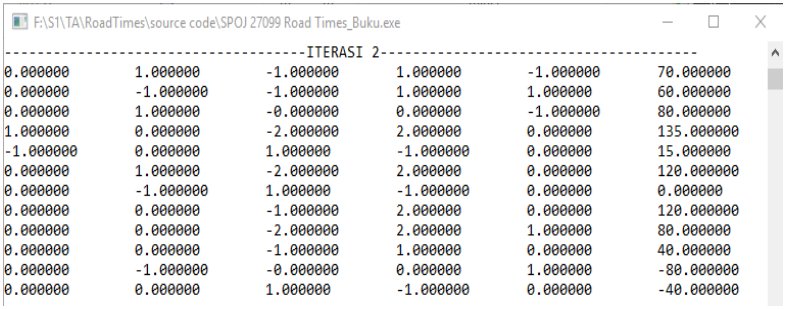
Hasil iterasi 1 pada sistem dapat dilihat pada Gambar 5.12. Dimana hasil tersebut sama dengan hasil iterasi 1 yang ada pada Subbab 2.9.1 Tabel 2.29.



Gambar 5.12 Iterasi 1 kasus minimalisasi pada pemodelan sistem model *primal*

c. Iterasi 2 tahap inisialisasi Simpleks

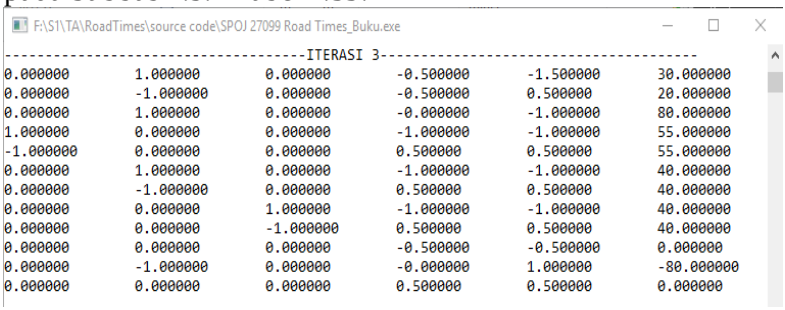
Hasil iterasi 2 pada sistem dapat dilihat pada Gambar 5.13. Dimana hasil tersebut sama dengan hasil iterasi 2 yang ada pada Subbab 2.9.1 Tabel 2.31.



Gambar 5.13 Iterasi 2 kasus minimalisasi pada pemodelan sistem model *primal*

d. Iterasi 3 tahap inisialisasi Simpleks

Hasil iterasi 3 pada sistem dapat dilihat pada Gambar 5.14. Dimana hasil tersebut sama dengan hasil iterasi 3 pada yang ada pada Subbab 2.9.1 Tabel 2.33.

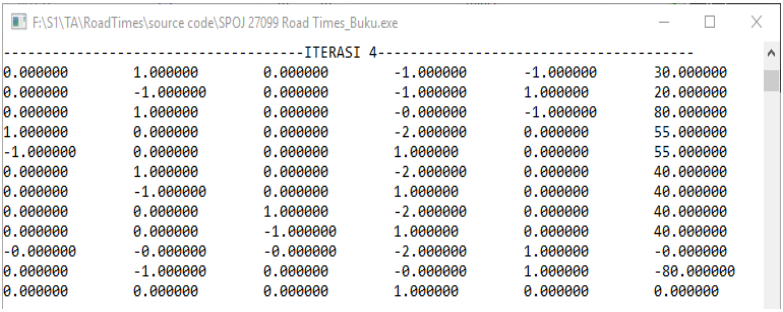


Gambar 5.14 Iterasi 3 kasus minimalisasi pada pemodelan sistem model *primal*

Pada iterasi ini, hasil optimal dari fungsi objektif z_{aux} sudah diperoleh. Akan tetapi variabel x_0 masih berada dalam *basic variable*, sehingga akan dilakukan satu kali lagi iterasi.

e. Hasil iterasi 4 tahap inisialisasi Simpleks

Hasil iterasi 4 pada sistem dapat dilihat pada Gambar 5.15. Dimana hasil tersebut sama dengan hasil iterasi 4 yang ada pada Subbab 2.9.1 Tabel 2.34.

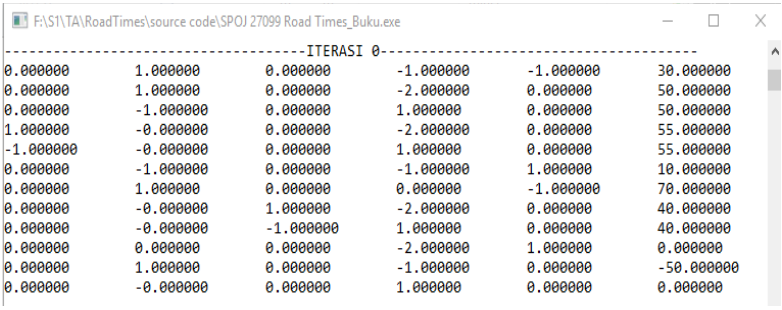


Gambar 5.15 Iterasi 4 kasus minimalisasi pada pemodelan sistem model *primal*

Setelah didapatkan hasil optimal dari fungsi objektif z_{aux}, tahap inisialisasi Simpleks selesai. Kemudian sistem akan mengganti fungsi objektif acuan menjadi fungsi objektif utama (z_{ori}), yang berarti tahap utama Simpleks dilakukan. Sistem akan kembali melakukan beberapa iterasi hingga hasil optimal dari fungsi objektif utama (z_{ori}) bisa didapatkan. Hasil dari iterasi yang dilakukan oleh sistem adalah sebagai berikut:

a. Hasil iterasi 0 tahap Simpleks

Hasil iterasi 0 pada sistem memiliki hasil yang sama dengan hasil iterasi 0 pada uji coba subbab 2.9.1. Gambar 5.16 memiliki hasil yang sama dengan Tabel 2.37. Hanya saja hasil iterasi pada sistem tidak menghilangkan fungsi objektif z_{aux} dan variabel x_0.

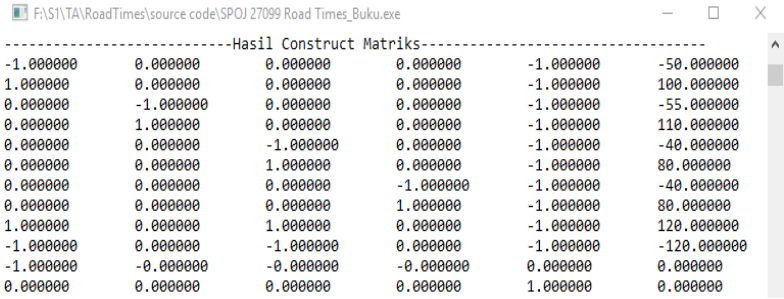


Gambar 5.16 Iterasi 0 Simpleks kasus minimalisasi pada pemodelan sistem model *primal*

Pada iterasi di atas, hasil optimal dari z_{ori} untuk mendapatkan waktu minimal sudah didapatkan. Waktu minimal yang didapatkan adalah 50. Kemudian sistem akan melanjutkan proses untuk mencari waktu maksimal.

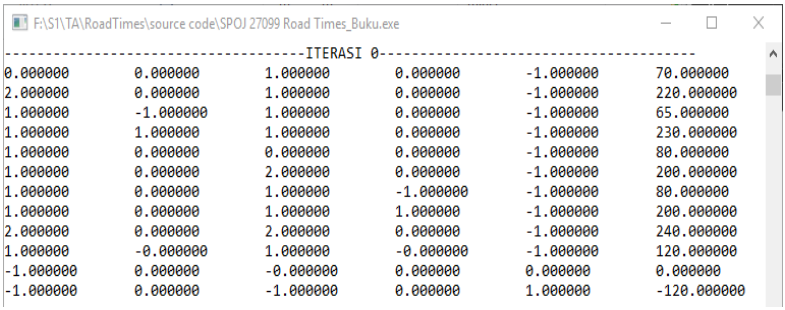
2. Mencari Waktu Maksimal

Matriks yang dibuat oleh sistem pada kasus mencari waktu maksimal dapat dilihat pada Gambar 5.17. Hasil dari matriks tersebut sama dengan tabel Simpleks pada subbab 2.9.1 Tabel 2.10.



Gambar 5.17 *Construct* matriks kasus maksimalisasi pada pemodelan sistem model *primal*

Kemudian sistem akan melakukan *pivoting* pertama. Hasil dari *pivoting* tersebut dapat dilihat pada Gambar 5.18. Hasil tersebut sama dengan tabel hasil dari *pivoting* pertama yang sudah dilakukan pada Subbab 2.9.1 Tabel 2.11.

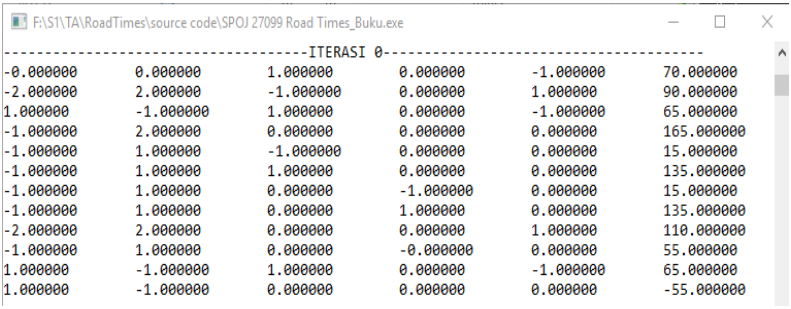


Gambar 5.18 Hasil *pivoting* pertama kasus maksimalisasi pada pemodelan sistem model *primal*

Tahap selanjutnya adalah akan dilakukan beberapa iterasi hingga didapatkan hasil optimal dari fungsi objektif z_{aux}. Hasil dari beberapa iterasi yang dilakukan oleh sistem adalah sebagai berikut:

a. Iterasi 0 tahap inisialisasi Simpleks

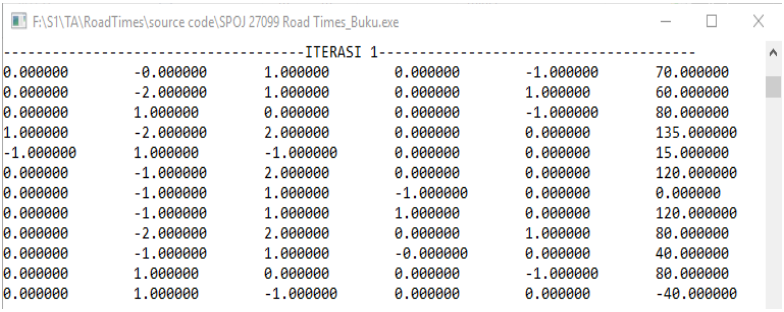
Hasil iterasi 0 pada sistem dapat dilihat pada Gambar 5.19. Dimana hasil tersebut sama dengan hasil iterasi 0 pada Subbab 2.9.1 Tabel 2.13.



Gambar 5.19 Iterasi 0 kasus maksimalisasi pada pemodelan sistem model *primal*

b. Iterasi 1 tahap inisialisasi Simpleks

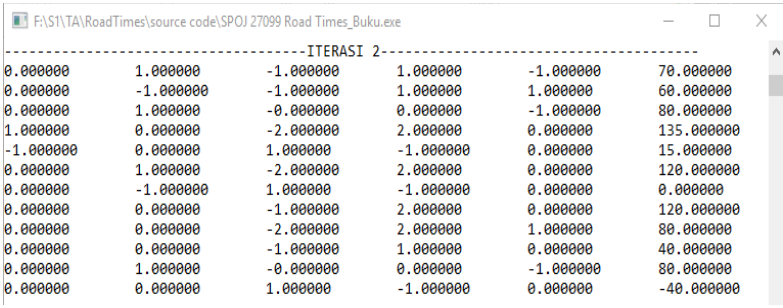
Hasil iterasi 1 pada sistem dapat dilihat pada Gambar 5.20. Dimana hasil tersebut sama dengan hasil iterasi 1 pada Subbab 2.9.1 Tabel 2.15.



Gambar 5.20 Iterasi 1 kasus maksimalisasi pada pemodelan sistem model *primal*

c. Iterasi 2 tahap inisialisasi Simpleks

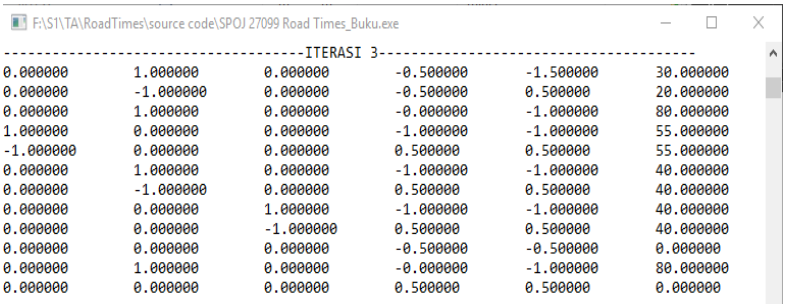
Hasil iterasi 2 pada sistem dapat dilihat pada Gambar 5.21. Dimana hasil tersebut sama dengan hasil iterasi 2 pada Subbab 2.9.1 Tabel 2.17.



Gambar 5.21 Iterasi 2 kasus maksimalisasi pada pemodelan sistem model *primal*

d. Iterasi 3 tahap inisialisasi Simpleks

Hasil iterasi 3 pada sistem dapat dilihat pada Gambar 5.22. Dimana hasil tersebut sama dengan hasil iterasi 3 pada Subbab 2.9.1 Tabel 2.19.

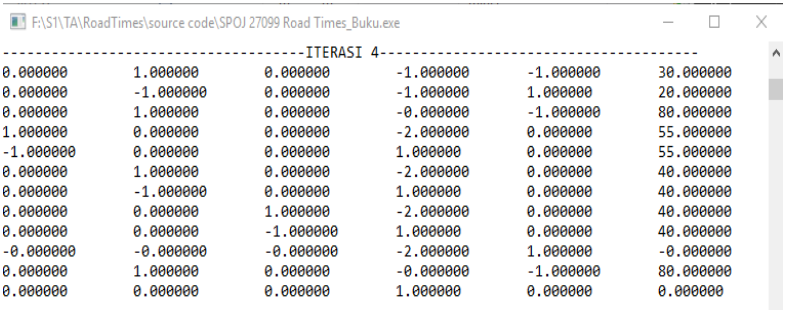


Gambar 5.22 Iterasi 3 kasus maksimalisasi pada pemodelan sistem model *primal*

Pada iterasi ini, hasil optimal dari fungsi objektif z_{aux} sudah diperoleh. Akan tetapi variabel x_0 masih berada dalam *basic variable*, sehingga akan dilakukan satu kali lagi iterasi.

e. Hasil iterasi 4 tahap inisialisasi Simpleks

Hasil iterasi 4 pada sistem dapat dilihat pada Gambar 5.23. Dimana hasil tersebut sama dengan hasil iterasi 4 pada Subbab 2.9.1 Tabel 2.20.

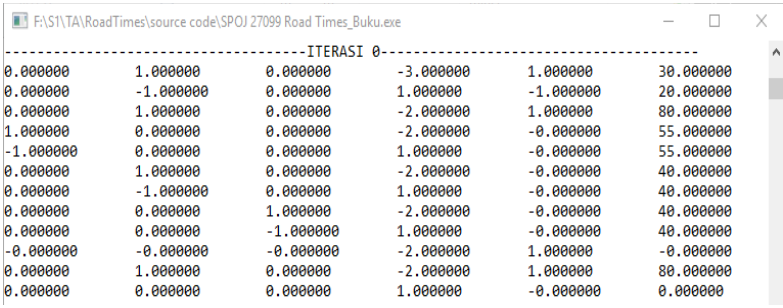


Gambar 5.23 Iterasi 4 kasus maksimalisasi pada pemodelan sistem model *primal*

Setelah didapatkan hasil optimal dari fungsi objektif z_{aux}, tahap inisialisasi Simpleks selesai. Hasil dari iterasi yang dilakukan oleh sistem pada tahap Simpleks adalah sebagai berikut:

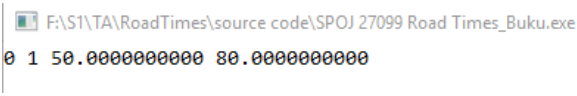
a. Hasil iterasi 0 tahap Simpleks

Hasil iterasi 0 pada sistem dapat dilhat pada Gambar 5.24. Dimana hasil tersebut sama dengan Tabel 2.23 pada Subbab 2.9.1.



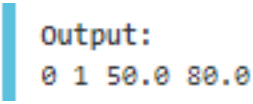
Gambar 5.24 Iterasi 0 Simpleks kasus maksimalisasi pada pemodelan sistem model *primal*

Pada iterasi di atas, hasil optimal dari z_{ori} untuk mendapatkan waktu maksimal sudah didapatkan. Waktu maksimal yang didapatkan adalah 80. Karena waktu minimal dan maksimal sudah didapatkan, sistem akan mengeluarkan hasil dari perkiraan waktu minimal dan maksimal sesuai dengan format keluaran yang diminta. Hasil keluaran dari sistem dapat dilihat pada Gambar 5.25.



Gambar 5.25 Hasil keluaran sistem model *primal*

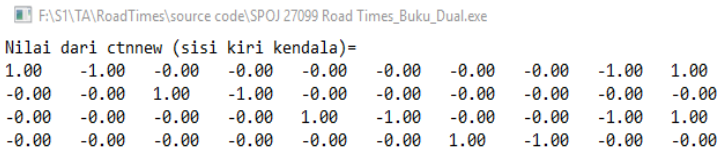
Hasil keluaran yang ada pada Gambar 5.25 menunjukkan bahwa untuk melakukan perjalanan dari kota 0 menuju kota 1, membutuhkan waktu minimal 50 menit dan waktu maksimal 80 menit. Dan hasil tersebut sesuai dengan uji coba pada Subbab 2.9.1. Hasil keluaran tersebut juga sesuai dengan contoh keluaran dalam permasalahan FN16ROAD yang ada pada SPOJ. Contoh keluaran yang ada pada SPOJ dapat dilihat pada Gambar 5.26.



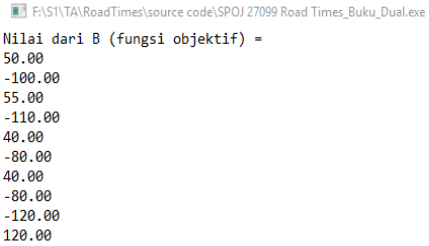
Gambar 5.26 Hasil keluaran SPOJ

5.2.1.2.2 Model *Dual*

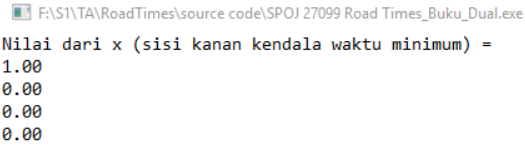
Masukan dan keluaran pada model *dual* sama dengan masukan dan keluaran pada model *primal*. Hanya saja model *dual* dibangun dari model *primal* dengan beberapa aturan yang sudah dibahas pada Subbab 2.7 dan sudah dicontohkan pada Subbab 2.9.2. Hasil dari pemodelan yang dibuat oleh sistem dapat dilihat pada Gambar 5.27-5.30.



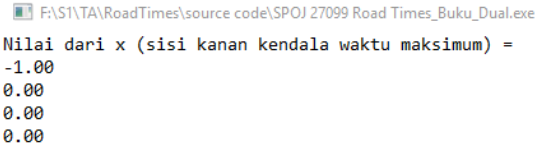
Gambar 5.27 Hasil dari *ctnnew* pada pemodelan sistem model *dual*



Gambar 5.28 Hasil dari *B* pada pemodelan sistem model *dual*



Gambar 5.29 Hasil dari x kasus minimalisasi pada pemodelan sistem model *dual*



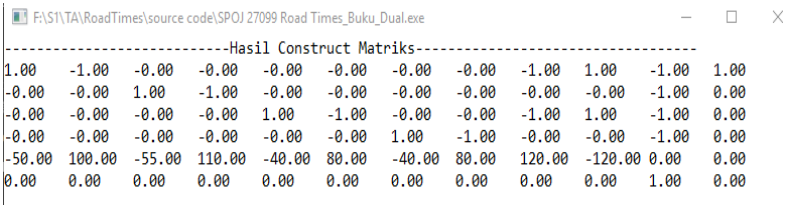
Gambar 5.30 Hasil dari x kasus maksimalisasi pada pemodelan sistem model *dual*

Hasil dari model yang dihasilkan sistem sama dengan model yang dihasilkan pada uji coba yang dilakukan secara manual pada Subbab 2.9.2. Kemudian sistem akan menjalankan algoritma Simpleks yang sudah diimplementasikan untuk mencari perkiraan

waktu minimal dan maksimal. Urutan proses yang dilakukan sama dengan proses pada model *primal*.

1. Mencari Waktu Minimal

Hasil pembuatan matriks untuk kasus mencari waktu minimal model *dual* pada sistem dapat dilihat pada Gambar 5.31. Dimana hasil tersebut sama dengan hasil pada Subbab 2.9.2 Tabel 2.51.

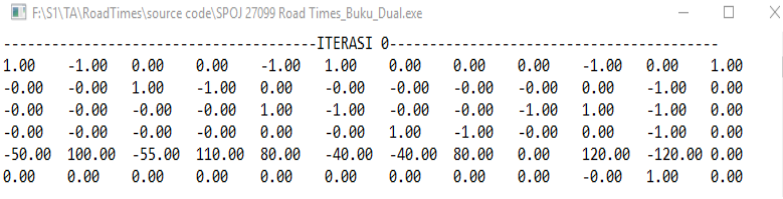


Gambar 5.31 *Construct* matriks kasus minimalisasi pada pemodelan sistem model *dual*

Karena semua koefisien pada sisi kanan kendala model *dual* sudah bernilai ≥ 0, maka sistem akan mengganti fungsi objektif acuan menjadi fungsi objektif utama (z_{ori}). Hasil dari iterasi yang dilakukan oleh sistem adalah sebagai berikut:

a. Iterasi 0 tahap Simpleks

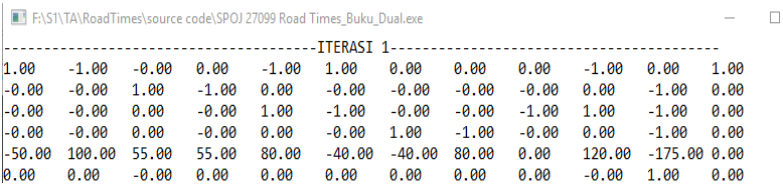
Hasil iterasi 0 pada sistem dapat dilihat pada Gambar 5.32. Dimana hasil tersebut sama dengan hasil iterasi 0 tahap Simpleks pada Subbab 2.9.2 Tabel 2.53.



Gambar 5.32 Iterasi 0 Simpleks kasus minimalisasi pada pemodelan sistem model *dual*

b. Iterasi 1 tahap Simpleks

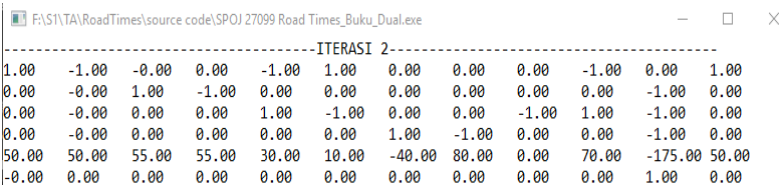
Hasil iterasi 1 pada sistem dapat dilihat pada Gambar 5.33. Dimana hasil tersebut sama dengan hasil iterasi 1 tahap Simpleks pada Subbab 2.9.2 Tabel 2.55.



Gambar 5.33 Iterasi 1 Simpleks kasus minimalisasi pada pemodelan sistem model *dual*

c. Iterasi 2 tahap Simpleks

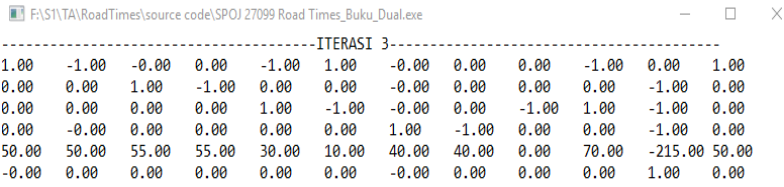
Hasil iterasi 2 pada sistem dapat dilihat pada Gambar 5.34. Dimana hasil tersebut sama dengan hasil iterasi 2 tahap Simpleks pada Subbab 2.9.2 Tabel 2.57.



Gambar 5.34 Iterasi 2 Simpleks kasus minimalisasi pada pemodelan sistem model *dual*

d. Iterasi 3 tahap Simpleks

Hasil iterasi 3 pada sistem dapat dilihat pada Gambar 5.35. Dimana hasil tersebut sama dengan hasil iterasi 3 tahap Simpleks pada Subbab 2.9.2 Tabel 2.59.

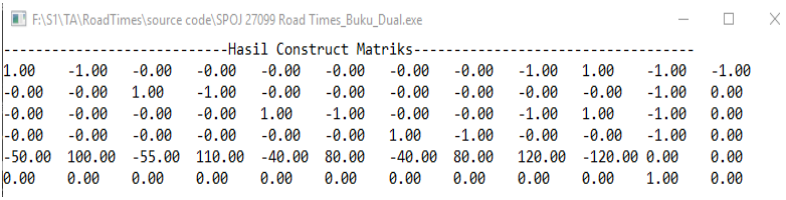


Gambar 5.35 Iterasi 3 Simpleks kasus minimalisasi pada pemodelan sistem model *dual*

Pada iterasi di atas, hasil optimal dari z_{ori} untuk mendapatkan waktu minimal sudah didapatkan. Waktu minimal yang didapatkan adalah 50. Kemudian sistem akan melanjutkan proses untuk mencari waktu maksimal.

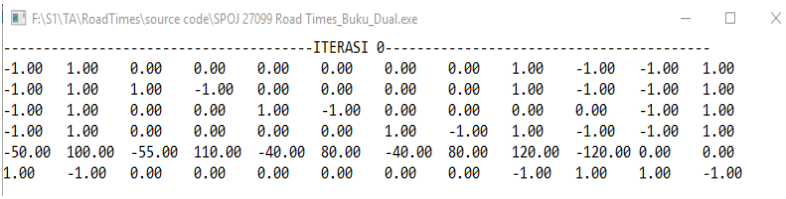
2. Mencari Waktu Maksimal

Matriks yang dibuat oleh sistem pada kasus mencari waktu maksimal untuk model *dual* dapat dilihat pada Gambar 5.36. Hasil dari matriks tersebut sama dengan tabel Simpleks yang ada pada Subbab 2.9.2 Tabel 2.38.



Gambar 5.36 *Construct* matriks kasus maksimalisasi pada pemodelan sistem model *dual*

Kemudian sistem akan melakukan *pivoting* pertama. Hasil dari *pivoting* tersebut dapat dilihat pada Gambar 5.37. Hasil tersebut sama dengan tabel hasil dari *pivoting* pertama yang sudah dilakukan pada Subbab 2.9.2 Tabel 2.39.

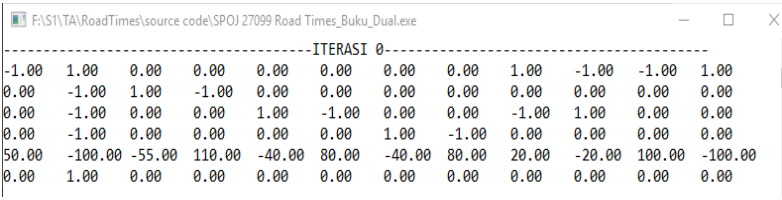


Gambar 5.37 Hasil *pivoting* pertama kasus maksimalisasi pada pemodelan sistem model *dual*

Tahap selanjutnya adalah akan dilakukan beberapa iterasi hingga didapatkan hasil optimal dari fungsi objektif z_{aux}. Hasil dari beberapa iterasi yang dilakukan oleh sistem adalah sebagai berikut:

a. Iterasi 0 tahap inisialisasi Simpleks

Hasil iterasi 0 pada sistem dapat dilihat pada Gambar 5.38. Dimana hasil tersebut sama dengan hasil iterasi 0 pada Subbab 2.9.2 Tabel 2.41.

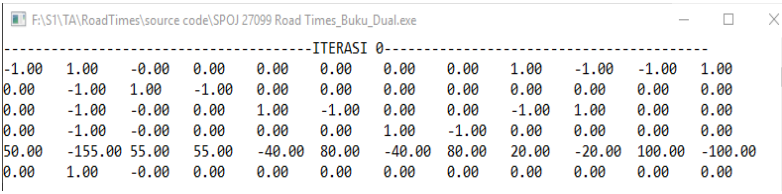


Gambar 5.38 Iterasi 0 kasus maksimalisasi pada pemodelan sistem model *dual*

Setelah didapatkan hasil optimal dari fungsi objektif z_{aux}, tahap inisialisasi Simpleks selesai. Hasil dari iterasi yang dilakukan oleh sistem pada tahap Simpleks adalah sebagai berikut:

a. Hasil iterasi 0 tahap Simpleks

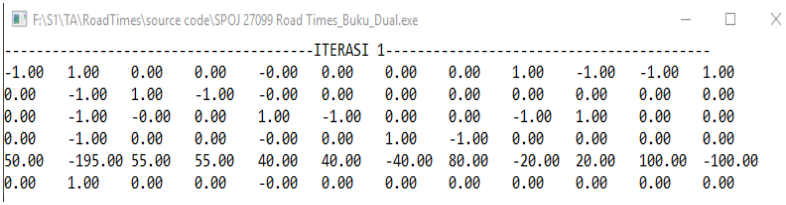
Hasil iterasi 0 pada sistem dapat dilhat pada Gambar 5.39. Dimana hasil tersebut sama dengan hasil iterasi 0 pada Subbab 2.9.2 Tabel 2.44.



Gambar 5.39 Iterasi 0 Simpleks kasus maksimalisasi pada pemodelan sistem model *dual*

b. Hasil iterasi 1 tahap Simpleks

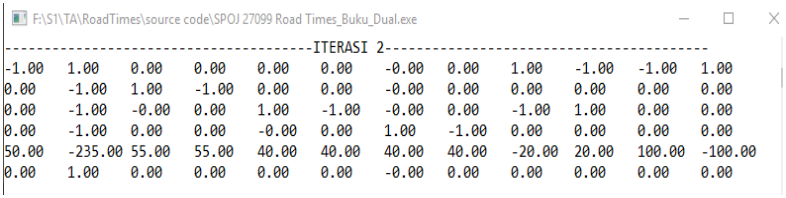
Hasil iterasi 1 pada sistem dapat dilhat pada Gambar 5.40. Dimana hasil tersebut sama dengan hasil iterasi 1 pada Subbab 2.9.2 Tabel 2.46.



Gambar 5.40 Iterasi 1 Simpleks kasus maksimalisasi pada pemodelan sistem model *dual*

c. Hasil iterasi 2 tahap Simpleks

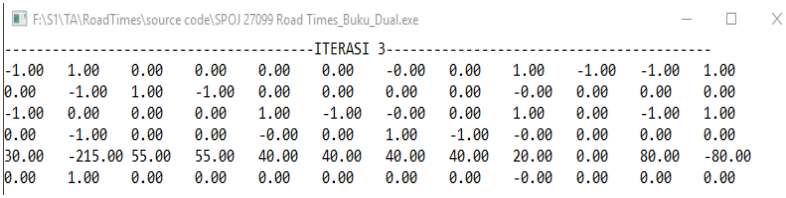
Hasil iterasi 2 pada sistem dapat dilhat pada Gambar 5.41. Dimana hasil tersebut sama dengan hasil iterasi 2 pada Subbab 2.9.2 Tabel 2.48.



Gambar 5.41 Iterasi 2 Simpleks kasus maksimalisasi pada pemodelan sistem model *dual*

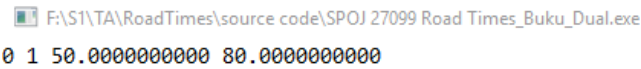
d. Hasil iterasi 3 tahap Simpleks

Hasil iterasi 3 pada sistem dapat dilhat pada Gambar 5.42. Dimana hasil tersebut sama dengan hasil iterasi 3 pada Subbab 2.9.2 Tabel 2.50.



Gambar 5.42 Iterasi 3 Simpleks kasus maksimalisasi pada pemodelan sistem model *dual*

Pada iterasi di atas, hasil optimal dari z_{ori} untuk mendapatkan waktu maksimal sudah didapatkan. Waktu maksimal yang didapatkan adalah 80. Karena waktu minimal dan maksimal sudah didapatkan, sistem akan mengeluarkan hasil dari perkiraan waktu minimal dan maksimal sesuai dengan format keluaran yang diminta. Hasil keluaran dari sistem dapat dilihat pada Gambar 5.43.



Gambar 5.43 Hasil keluaran sistem model *dual*

Hasil keluaran yang ada pada Gambar 5.43 sama dengan hasil keluaran model *primal* yang ada pada Gambar 5.25.

5.2.2 Uji Coba Kinerja

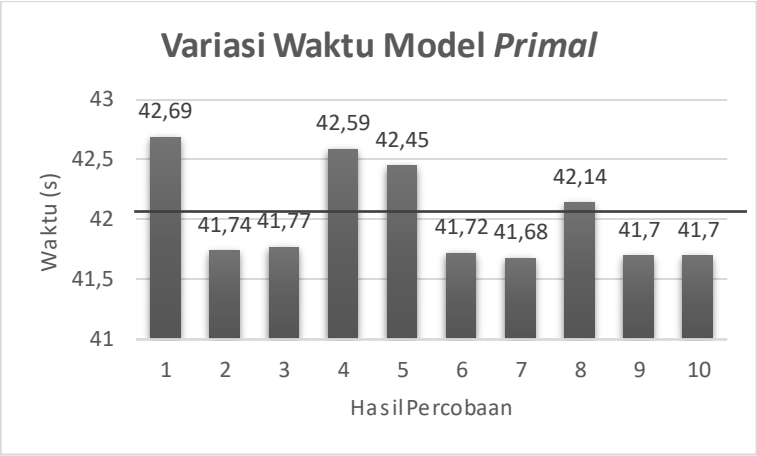
Uji coba kinerja dilakukan dengan cara mengirimkan kode sumber ke dalam situs penilaian *online* SPOJ sebanyak 10 kali baik pada model *primal* maupun model *dual*. Hal tersebut bertujuan untuk melihat variasi waktu dan variasi memori yang dibutuhkan program. Hasil uji coba sebanyak 10 kali pada SPOJ dapat dilihat pada Gambar 5.44 untuk model *primal* dan Gambar 5.45 untuk model *dual*. Sedangkan untuk diagram variasi waktu dapat dilihat pada Gambar 5.46 untuk model *primal* dan Gambar 5.47 untuk model *dual*. Diagram variasi memori dapat dilihat pada Gambar 5.48 untuk model *primal* dan Gambar 5.49 untuk model *dual*.

25037065		2019-12-09 23:43:07	Road Times	accepted edit ideone it	41.70	4.5M	CPP14-CLANG
25037062		2019-12-09 23:41:11	Road Times	accepted edit ideone it	41.70	4.5M	CPP14-CLANG
25037052		2019-12-09 23:37:44	Road Times	accepted edit ideone it	42.14	4.5M	CPP14-CLANG
25037026		2019-12-09 23:23:32	Road Times	accepted edit ideone it	41.68	4.5M	CPP14-CLANG
25037014		2019-12-09 23:20:29	Road Times	accepted edit ideone it	41.72	4.5M	CPP14-CLANG
25037011		2019-12-09 23:14:25	Road Times	accepted edit ideone it	42.45	4.5M	CPP14-CLANG
25036890		2019-12-09 22:23:53	Road Times	accepted edit ideone it	42.59	4.7M	CPP14-CLANG
25036786		2019-12-09 21:59:48	Road Times	accepted edit ideone it	41.77	4.5M	CPP14-CLANG
25036605		2019-12-09 21:54:39	Road Times	accepted edit ideone it	41.74	4.5M	CPP14-CLANG
25036411		2019-12-09 20:13:48	Road Times	accepted edit ideone it	42.69	4.5M	CPP14-CLANG

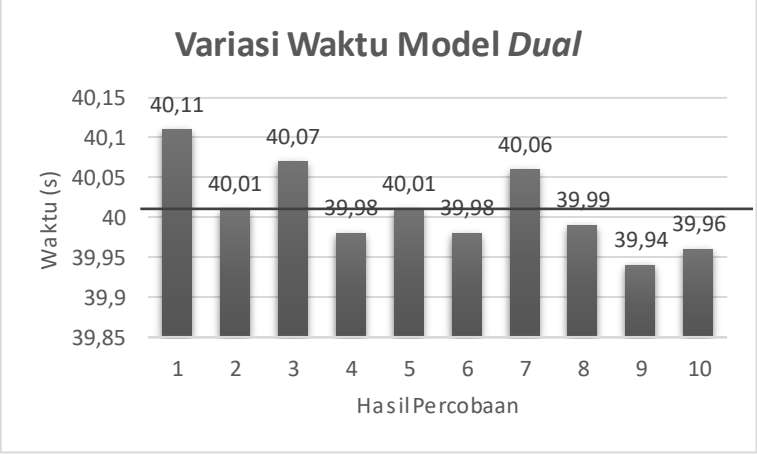
Gambar 5.44 Hasil Uji Coba Kinerja Program Model *Primal*

25132152		2019-12-25 21:58:22	Road Times	accepted edit ideone it	40.11	4.5M	CPP14-CLANG
25132103		2019-12-25 21:34:10	Road Times	accepted edit ideone it	40.01	4.5M	CPP14-CLANG
25132092		2019-12-25 21:18:25	Road Times	accepted edit ideone it	40.07	4.5M	CPP14-CLANG
25132071		2019-12-25 21:07:17	Road Times	accepted edit ideone it	39.98	4.4M	CPP14-CLANG
25132065		2019-12-25 21:05:25	Road Times	accepted edit ideone it	40.01	4.5M	CPP14-CLANG
25132007		2019-12-25 20:42:22	Road Times	accepted edit ideone it	39.98	4.5M	CPP14-CLANG
25131968		2019-12-25 20:31:02	Road Times	accepted edit ideone it	40.06	4.7M	CPP14-CLANG
25131588		2019-12-25 18:39:11	Road Times	accepted edit ideone it	39.99	4.7M	CPP14-CLANG
25131358		2019-12-25 17:46:59	Road Times	accepted edit ideone it	39.94	4.5M	CPP14-CLANG
25116548		2019-12-22 21:17:26	Road Times	accepted edit ideone it	39.96	4.6M	CPP14-CLANG

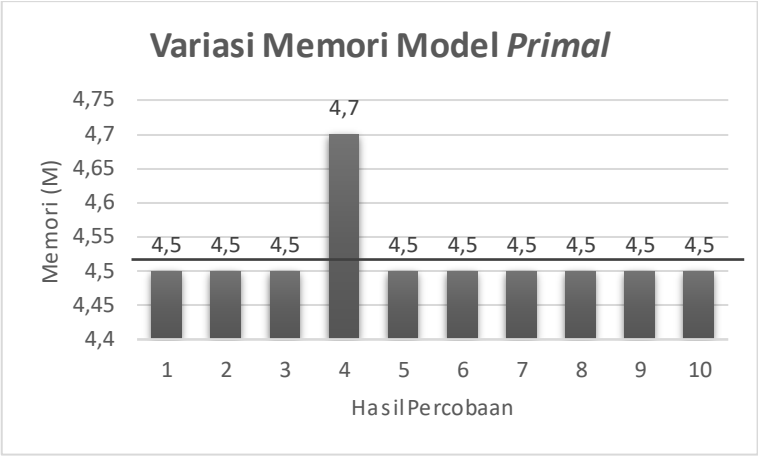
Gambar 5.45 Hasil Uji Coba Kinerja Program Model *Dual*



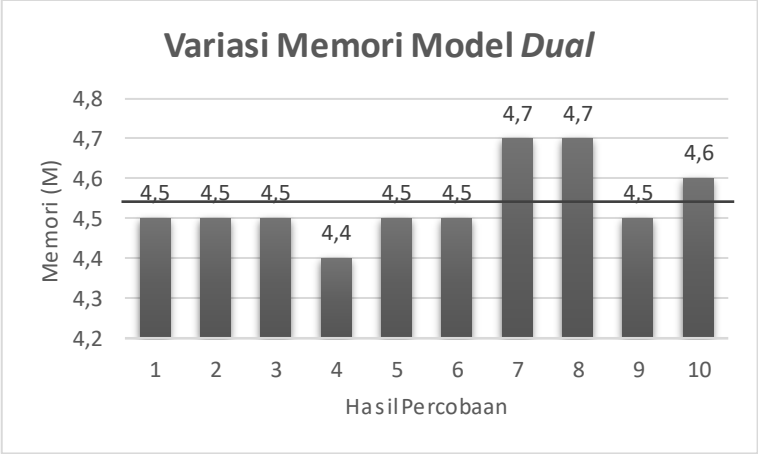
Gambar 5.46 Variasi Waktu Uji Coba Kinerja Program Model *Primal*



Gambar 5.47 Variasi Waktu Uji Coba Kinerja Program Model *Dual*



Gambar 5.48 Variasi Memori Uji Coba Kinerja Program Model *Primal*



Gambar 5.49 Variasi Memori Uji Coba Kinerja Program Model *Dual*

Gambar 5.44 memperlihatkan 10 kali percobaan pengiriman kode sumber ke dalam situs penilaian *online* SPOJ pada model *primal* dan Gambar 5.45 pada model *dual*. Semua mendapatkan

hasil *Accepted* dengan waktu dan memori yang bervariasi. Variasi waktu dan memori kemudian dirangkum dalam sebuah diagram.

Dari diagram variasi waktu dan memori yang dihasilkan oleh Gambar 5.46-5.49, dapat dilakukan pengambilan nilai rata-rata dari 10 percobaan untuk menarik kesimpulan berapa waktu dan memori yang dibutuhkan oleh program untuk menyelesaikan permasalahan FN16ROAD. Rata-rata waktu yang diperlukan program adalah 42,018 detik untuk model *primal* dan 40,011 detik untuk model *dual*. Sedangkan rata-rata memori yang diperlukan adalah 4,52 M untuk model *primal* dan 4,54 M untuk model *dual*.

5.3 Analisis dan Kesimpulan Umum

Dari uji coba yang telah dilakukan, didapatkan beberapa analisis sebagai berikut:

- Uji coba kebenaran yang dilakukan dengan mengirimkan kode sumber ke dalam situs penilaian *online* SPOJ didapatkan hasil *Accepted* dengan waktu terbaik yang dibutuhkan oleh program adalah 41,68 detik untuk model *primal* dan 39,94 untuk model *dual* dengan memori yang dibutuhkan sebesar 4,5 M baik untuk model *primal* maupun model *dual*.
- Uji coba kebenaran yang dilakukan dengan membandingkan hasil uji coba kasus sederhana yang sudah dibahas pada Subbab 2.9 dengan hasil uji coba keluaran sistem didapatkan bahwa kedua uji coba tersebut mengeluarkan hasil yang sama baik pada model *primal* maupun model *dual*.
- Uji coba kinerja yang dilakukan dengan cara mengirimkan kode sumber ke dalam situs penilaian *online* SPOJ sebanyak 10 kali didapatkan hasil *Accepted* pada semua percobaan dengan rata-rata waktu yang diperlukan adalah 42,018 detik untuk model *primal* dan 40,011 detik untuk model *dual*. Sedangkan rata-rata memori yang diperlukan adalah 4,52 M untuk model *primal* dan 4,54 M untuk model *dual*.

[*Halaman ini sengaja dikosongkan*]

BAB VI
KESIMPULAN

Pada bab ini akan dijelaskan kesimpulan dari hasil ujicoba yang telah dilakukan.

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan terhadap implementasi solusi untuk permasalahan *FN16ROAD-Road Times* dalam menemukan perkiraan waktu minimal dan waktu maksimal, serta telah disarikan pada Subbab 5.3, dapat diambil kesimpulan sebagai berikut:

1. Permasalahan untuk mencari perkiraan waktu minimal dan perkiraan waktu maksimal dengan mempertimbangkan beberapa batasan masalah telah berhasil diselesaikan dengan teori-teori yang telah dicantumkan pada Bab 2.
2. Implementasi algoritma Simpleks pada permasalahan *FN16ROAD-Road Times* menghasilkan solusi yang benar serta dapat digunakan untuk mencari nilai perkiraan waktu minimal dan maksimal yang diperlukan untuk menempuh perjalanan antar kota, baik dengan menggunakan model *primal* maupun model *dual*.
3. Model *dual* menghasilkan rata-rata waktu lebih baik dibandingkan dengan model *primal* dengan selisih rata-rata waktu yang diperlukan untuk menyelesaikan permasalahan *FN16ROAD-Road Times* adalah sebesar 2,007 detik.
4. Model *primal* menghasilkan rata-rata memori lebih baik dibandingkan dengan model *dual* dengan selisih rata-rata memori yang diperlukan untuk menyelesaikan permasalahan *FN16ROAD-Road Times* adalah sebesar 0,02 M.

[*Halaman ini sengaja dikosongkan*]

DAFTAR PUSTAKA

[1] H. A. Taha, Operations Research An Introduction Eight Edition, United States of America: Pearson Education, 2007.

[2] H. Nabli, "An overview on the simplex algorithm," *Applied Mathematics and Computation,* vol. 210, no. 2, p. 479–489, 2009.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to Algorithms Third Edition, London, England: The MIT Press, 2009.

[4] P.-Q. Pan, "A primal deficient-basis simplex algorithm," *Applied Mathematics and Computation,* vol. 196, no. 2, p. 898–912, 2008.

[5] S. Mizuno, N. Sukegawa and A. Deza, "A primal-simplex based Tardos' algorithm," *Operations Research Letters,* vol. 43, no. 6, p. 625–628, 2015.

[*Halaman ini sengaja dikosongkan*]

BIODATA PENULIS

Ivanda Zevi Amalia, lahir di Bojonegoro tanggal 23 Agustus 1998. Penulis merupakan anak pertama dari dua bersaudara. Penulis telah menempuh pendidikan formal di TK Pertiwi Bojonegoro, SD Negeri Kadipaten 1 Bojonegoro (2004-2010), SMP Negeri Model Terpadu Bojonegoro (2010-2013), SMA Negeri 1 Bojonegoro (2013-2016). Penulis melanjutkan studi dengan berkuliah pada program sarjana (S1) di Departemen Teknik Informatika ITS. Selama kuliah di Teknik Informatika ITS, penulis mengambil bidang minat Algoritma dan Pemrograman (AP). Selama menempuh perkuliahan, penulis mendapatkan beasiswa dari Beasiswa Unggulan yang diadakan oleh Kementrian Pendidikan dan Kebudayaan. Selain itu, penulis pernah menjadi asisten dosen Komputasi Numerik. Penulis juga menjadi administrator di Laboratorium MIS. Penulis juga aktif mengikuti organisasi kemahasiswaan, yaitu sebagai Sekretaris Departemen Kesejahteraan Mahasiswa di Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) ITS. Selain aktif mengikuti organisasi, penulis juga aktif dalam kegiatan kepanitiaan Schematics, yaitu sebagai staff WebKes pada tahun 2017 dan menjadi Bendahara pada tahun 2018. Saat ini penulis sedang mengikuti program *Fast Track* yang diadakan oleh Departemen Teknik Informatika ITS. Penulis dapat dihubungi melalui surel di `zevi2308@gmail.com`.