ALGORITMA *LEXISEARCH* MENGGUNAKAN REPRESENTASI *ADJACENCY* DAN *PATH* UNTUK MENYELESAIKAN *BOTTLENECK TRAVELING SALESMAN PROBLEM*

Skripsi

Untuk memenuhi sebagian persyaratan

mencapai derajat Sarjana S-1

Program Studi Matematika

diajukan oleh

SITI BAHJATUN SANIYAH

14610016

Kepada

PROGRAM STUDI MATEMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA

YOGYAKARTA

2018

SURAT PERSETUJUAN SKRIPSI/TUGAS AKHIR

Hal : Persetujuan Skripsi / Tugas Akhir
Lamp :

Kepada
Yth. Dekan Fakultas Sains dan Teknologi
UIN Sunan Kalijaga Yogyakarta
di Yogyakarta

Assalamu'alaikum wr. wb.

Setelah membaca, meneliti, memberikan petunjuk dan mengoreksi serta mengadakan perbaikan seperlunya, maka kami selaku pembimbing berpendapat bahwa skripsi Saudara:

Nama : Siti Bahjatun Saniyah
NIM : 14610016
Judul Skripsi : Algoritma Lexisearch Menggunakan Representasi *Adjacency* dan *Path* untuk Menyelesaikan *Bottleneck Traveling Salesman Problem*

sudah dapat diajukan kembali kepada Program Studi Matematika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu dalam Program Studi Matematika.

Dengan ini kami mengharap agar skripsi/tugas akhir Saudara tersebut di atas dapat segera dimunaqsyahkan. Atas perhatiannya kami ucapkan terima kasih.

Wassalamu'alaikum wr. wb.

Yogyakarta,
Pembimbing

Muchammad Abrori, S.Si, M.Kom
NIP. 19720423 199903 1 003

KEMENTERIAN AGAMA
UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Marsda Adisucipto Telp. (0274) 540971 Fax. (0274) 519739 Yogyakarta 55281

PENGESAHAN TUGAS AKHIR

Nomor : B-1993/Un.02/DST/PP.00.9/05/2018

Tugas Akhir dengan judul : Algoritma Lexisearch Menggunakan Representasi Adjacency dan Path untuk Menyelesaikan Bottleneck Traveling Salesman Problem

yang dipersiapkan dan disusun oleh:

Nama	: SITI BAHJATUN SANIYAH
Nomor Induk Mahasiswa	: 14610016
Telah diujikan pada	: Jumat, 27 April 2018
Nilai ujian Tugas Akhir	: A

dinyatakan telah diterima oleh Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta

TIM UJIAN TUGAS AKHIR

Ketua Sidang

Muchammad Abrori, S.Si., M.Kom
NIP. 19720423 199903 1 003

Penguji I

Malahayati, S.Si., M.Sc
NIP. 19840412 201101 2 010

Penguji II

Pipit Pratiwi Rahayu, S.Si., M.Sc.
NIP. 19861208 201503 2 006

Yogyakarta, 27 April 2018
UIN Sunan Kalijaga
Fakultas Sains dan Teknologi
DEKAN

Dr. Murtono, M.Si.
NIP. 19691212 200003 1 001

SURAT PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini :

Nama : Siti Bahjatun Saniyah

NIM : 14610016

Program Studi : Matematika

Fakultas : Sains dan Teknologi

Dengan ini menyatakan dengan sesungguhnya bahwa skripsi ini merupakan hasil pekerjaan penulis sendiri dan sepanjang pengetahuan penulis tidak berisi materi yang dipublikasikan atau ditulis orang lain, dan atau telah digunakan sebagai persyaratan penyelesaian Tugas Akhir di Pergurun Tinggi lain, kecuali secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogykarta, 13 April 2018

Yang menyatakan

Siti Bahjatun Saniyah

HALAMAN PERSEMBAHAN

Bismillahirohmanirrohim

Dengan izin Allah, karya sederhana ini penulis persembahkan kepada:

Kedua orangtua tercinta, Bapak Endro Asmoro dan Ibu Mulyani

Almamater tercinta, Universitas Islam Negeri Sunan Kalijaga Yogyakarta

Teman-teman seperjuangan, Element Math 2014

Yogi Kurniawan

MOTTO

Yang terpenting bukanlah 'peran apa yang kamu dapat'

Akan tetapi 'bagaimana kamu menjalankan peranmu dengan sebaik-baiknya'

KATA PENGANTAR

Bismillahirohmanirrohim

Alhamdulillahi robbil'alamin, segala puji bagi Allah SWT, Tuhan semesta alam yang telah melimpahkan rahmat, taufik, hidayah, serta inayah-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul **"Algoritma *Lexisearch* Menggunakan Represenasi *Adjacency* dan *Path* untuk Menyelesaikan *Bottleneck Traveling Salesman Problem*"** sebagai salah satu syarat untuk memperoleh gelar Sarjana Program Studi Matematika. Shalawat serta salam semoga senantiasa tercurah kepada junjungan kita Nabi Muhammad SAW, keluarga, sahabat, dan seluruh umatnya sampai akhir zaman.

Penulis menyadari bahwa penyusunan skripsi ini tidak akan selesai dengan baik tanpa adanya bantuan, bimbingan, serta motivasi dari berbagai pihak. Oleh karena itu, dengan kerendahan hati penulis menyampaikan ucapan terimakasih yang sebesar-besarnya kepada semua pihak yang telah membantu terwujudnya skripsi ini secara langsung maupun tidak langsung. Ucapan terimakasih ini penulis ucapkan utamanya kepada:

1. Prof. Drs. K.H. Yudian Wahyudi, M.A., Ph.D., selaku Rektor UIN Sunan Kalijaga Yogyakarta.
2. Dr. Murtono, M.Si., selaku Dekan Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta.
3. Dr. M. Wakhid Musthofa, S.Si., M.Si., selaku Ketua Program Studi Matematika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta.

4. Malahayati, M.Sc., selaku Dosen Penasehat Akademik mahasiswa Program Studi Matematika angkatan 2014 yang selalu memberikan motivasi, bimbingan, dan dukungan dalam melaksanakan perkuliahan dan penyusunan skripsi.

5. Bapak Muchammad Abrori, S.Si, M.Kom., selaku Dosen Pembimbing Skripsi yang telah dengan sabar memberikan ilmu, arahan, dan dukungannya, serta meluangkan waktunya sehingga penulisan skripsi ini dapat terselesaikan.

6. Bapak/Ibu Dosen dan Staf Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta atas ilmu, bimbingan dan pelayanan selama perkuliahan.

7. Kedua orangtua tercinta, Bapak Endro Asmoro dan Ibu Mulyani atas segala doa dan dukungan baik secara moril maupun materiil, serta selalu memberikan yang terbaik bagi penulis. Terimakasih banyak, tanpa kalian penulis tidak akan sampai ke titik ini.

8. Seluruh keluarga besar penulis yang selalu memberikan motivasi dan dorongan bagi penulis sehingga penulis dapat menyelesikan skripsi ini tepat pada waktunya.

9. Yogi Kurniawan, untuk setiap *support*, motivasi dan segala bentuk bantuan yang diberikan kepada penulis.

10. Sahabatku, Silmi Firdausi Mahfudz dan Ani Herniawati, untuk setiap motivasi dan kesabarannya mendengarkan segala keluh kesah penulis.

11. Teman-teman Matematika 2014, untuk kebersamaannya selama menimba ilmu di Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta. Sukses selalu *Element Math*, *see you on top!*.

12. Semua pihak yang tidak dapat penulis sebutkan satu-persatu yang telah membantu dalam penyusunan skripsi ini.

Penulis menyadari penyusunan skripsi ini masih jauh dari kesempurnaan. Oleh karena itu segala kritik maupun saran yang bersifat membangun dalam pengembangan penelitian ini sangat penulis harapkan. Semoga penyusunan skripsi ini dapat bermanfaat demi kemaslahatan umat.

Yogyakarta, 8 Maret 2018

Penulis

SITI BAHJATUN SANIYAH

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMBANG

$a \in A$	: a anggota himpunan A
c_{ij}	: anggota matriks C dengan pasangan verteks (i, j)
$F(\cdot)$	: suatu fungsi tujuan
min	: nilai minimum
max	: nilai maksimum
■	: akhir suatu bukti
$\rightarrow$	: menuju
$a \equiv b$	: nilai a equivalen dengan nilai b
α_i	: verteks ke$-i$ dalam suatu rute

ABSTRAK

ALGORITMA *LEXISEARCH* MENGGUNAKAN REPRESENTASI *ADJACENCY* DAN *PATH* UNTUK MENYELESAIKAN *BOTTLENECK TRAVELING SALESMAN PROBLEM*

Oleh

SITI BAHJATUN SANIYAH

14610016

Traveling Salesman Problem (TSP) merupakan suatu permasalahan dimana seorang sales harus melalui semua kota dengan jarak yang terpendek dan setiap kota hanya boleh dilalui tepat satu kali. Dalam perkembangannya, muncul variasi baru dari TSP yaitu *Bottleneck Traveling Salesman Problem* (BTSP). BTSP sama seperti TSP tetapi diubah fungsi tujuannya, yaitu meminimumkan putaran terpanjang pada rute yang dilalui sales.

Penelitian ini bertujuan untuk menyelesaikan BTSP dengan menggunakan dua cara untuk merepresentasikan rute yang dilalui sales, yaitu representasi *adjacency* dan representasi *path*. Masalah dalam penelitian ini diselesaikan dengan Algoritma *Lexisearch*. Dasar dari algoritma tersebut adalah menentukan struktur solusi dengan ketentuan: setiap elemen pada ruang solusi (matriks jarak) disusun dalam urutan yang hierarkis sebagai blok dan sub-blok dalam blok, seperti halnya susunan kata dalam kamus. Dalam kasus BTSP, setiap verteks melambangkan 'huruf' pada alfabet, dan setiap rute melambangkan 'kata' yang tersusun dari alfabet tersebut. Penyelesaian dengan algoritma *Lexisearch* dilakukan dengan cara perhitungan manual maupun dengan program komputer berbasis MATLAB R2013a.

Hasil dari penelitian ini yaitu suatu rute yang dilalui sales dengan bobot (jarak) maksimum terkecil. Nilai optimal dari solusi tersebut merupakan bobot terbesar dari rute yang telah diperoleh.

Kata kunci: *Bottleneck Traveling Salesman*, *Lexisearch*, representasi *adjacency*, representasi *path*, tabel alfabet, MATLAB.

BAB I

PENDAHULUAN

Pada bab pendahuluan dijelaskan mengenai latar belakang masalah yang mendasari penelitian ini yang kemudian dirumuskan dalam suatu rumusan masalah. Berdasarkan latar belakang dan rumusan masalah yang telah disusun, ditentukan tujuan penelitian agar penelitian ini memiliki arah yang jelas mengenai apa saja yang ingin dicapai. Selanjutnya pada bab ini juga dijelaskan mengenai manfaat penelitian, tinjauan pustaka, sistematika penulisan, serta metode penelitian skripsi ini.

1.1. Latar Belakang Masalah

Masalah rute terpendek merupakan suatu permasalahan dimana seseorang ingin menentukan rute terpendek antara dua kota dengan dua atau lebih rute alternatif yang tersedia. Masalah rute terpendek sering dijumpai dalam kehidupan sehari-hari di berbagai sektor, antara lain pada bidang transportasi, komunikasi, dan komputasi. Hal ini menjadi sangat penting karena berkaitan dengan meminimumkan biaya dan efisiensi jarak maupun waktu.

Untuk mempermudah dalam menemukan penyelesaiannya, umumnya masalah ini direpresentasikan dengan graf. Tujuannya adalah mengunjungi setiap verteks pada graf dari verteks awal sampai verteks akhir dengan bobot minimum. Dalam hal ini bobot yang digunakan adalah jarak, dan kota-kota yang dikunjungi diasumsikan sebagai verteks yang saling terhubung. Salah satu masalah rute terpendek adalah *Traveling Salesman Problem* (TSP), yaitu suatu permasalahan dimana seorang sales harus melalui semua verteks dengan jarak yang terpendek dan setiap verteks hanya

boleh dilalui tepat satu kali, kemudian kembali ke verteks awal. Fungsi tujuan dari TSP yaitu untuk meminimumkan bobot total pada rute yang dilewati sales.

TSP secara matematis dirumuskan pada tahun 1800 oleh matematikawan Inggris Thomas Kirkman dan matematikawan Irlandia W.R. Hamilton. Teori dari W.R. Hamilton pada graf yaitu melewati seluruh verteks tepat satu kali dan kembali ke verteks awal kemudian dikenal dengan Hamiltonian *cycle* yang merupakan teori dasar pada kasus TSP. Bentuk umum TSP pertama kali dipelajari oleh para ahli matematika selama tahun 1930an di Wina dan di Harvard, terutama oleh Karl Menger. Pada tahun 1950an dan 1960an, TSP menjadi semakin populer di kalangan ilmuwan Eropa dan Amerika Serikat.

Dalam perkembangannya, muncul variasi-variasi baru dari TSP, salah satunya adalah *Bottleneck Traveling Salesman Problem* (BTSP). BTSP merupakan kasus TSP dengan kendala tambahan, misalnya sales mempunyai masalah dengan mobilnya sehingga tidak dapat mengunjungi setiap verteks secara maraton, sehingga BTSP menghasilkan solusi baru yang memenuhi kebutuhan sales dengan kondisi tertentu. BTSP adalah masalah dalam optimasi kombinatorial yang bertujuan untuk menemukan Hamiltonian *cycle* dalam graf berbobot yang meminimumkan bobot maksimum sisi pada rute sales. BTSP pertama kali diformulasikan oleh Gilmore dan Gomory (1964), dan secara keseluruhan oleh Garfinkel dan Gilbert (1978). Jika pada kasus BTSP bobot dari verteks A ke B berbeda dengan bobot dari verteks B ke A, maka kasus tersebut disebut *Asymmetric* BTSP.

Satu dari sekian algoritma yang dirumuskan dan dikembangkan oleh para matematikawan untuk memecahkan TSP maupun BTSP adalah *Lexisearch*. Pendekatan *Lexisearch* atau *Lexicographic Search* adalah pendekatan yang dikembangkan oleh Pandit dalam rangka memecahkan masalah pemuatan pada tahun 1962.

Nama *Lexi-Search* sendiri berasal dari kata *Lexicography*, yaitu ilmu penyimpanan dan pengambilan informasi yang efektif, yang menunjukkan bahwa pencarian solusi optimal dilakukan secara sistematis, sama seperti seseorang mencari arti sebuah kata dalam kamus.

Pada tahun 1997, penelitian Pandit dikembangkan oleh anak didiknya yaitu M. Ramesh dalam tesisnya yang berjudul *"A Lexisearch Approach to Some Combinatorial Programming Problems"*. Ramesh menggunakan algoritma *Lexisearch* untuk menyelesaikan TSP. Pada dasarnya, setiap algoritma untuk TSP dapat pula digunakan untuk menyelesaikan BTSP. Oleh karena itu Zakir H. Ahmed menggunakan algoritma *Lexisearch* untuk menyelesaikan BTSP dalam sebuah jurnal yang ia tulis dengan judul *"A Lexisearch Algorithm for the Bottleneck Traveling Salesman Problem"* pada tahun 2010.

Representasi menurut Kamus Besar Bahasa Indonesia berarti perwakilan. Representasi juga dapat diartikan sebagai sesuatu yang ditangkap oleh indra manusia, diproses oleh akal yang hasilnya adalah sebuah ide yang kemudian diungkapkan kembali dengan bahasa baru. Dalam ilmu matematika, representasi sangat diperlukan untuk memberikan gambaran atas sesuatu secara akurat. Misalnya dalam proses perhitungan dan hasil dari penyelesaian masalah yang tentunya perlu direpresentasikan dengan tepat.

Permasalahan yang dibahas dalam penelitian ini, yaitu BTSP merupakan salah satu kasus dari Hamiltonian *cycle* yang sangat erat hubungannya dengan representasi *adjacency* dan representasi *path*. Representasi *adjacency* pada suatu rute biasanya merupakan representasi yang berhubungan dengan permutasi yang dilakukan dengan mendaftar angka yang merepresentasikan posisi dalam suatu urutan. Sedangkan pada representasi *path* hanya dilakukan dengan mendaftar urutan

rute seperti mencari *walk* dari suatu graf.

Perhitungan untuk mencari nilai optimal dapat dilakukan secara manual, tetapi perhitungan manual tentunya akan memakan waktu. Oleh karena itu, perlu digunakan suatu simulasi untuk membantu proses perhitungan agar lebih efisien. Salah satu *software* yang tepat digunakan dalam suatu algoritma yang melibatkan matriks adalah MATLAB. MATLAB *(Matrix Laboratory)* pertama kali dikenalkan oleh *University of New Mexico* dan *University of Stanford* pada tahun 1970. *Software* ini pertama kali digunakan untuk keperluan analisis numerik, aljabar linier, dan teori tentang matriks. MATLAB memiliki sebuah aplikasi *display* yaitu GUI *(Graphical User Interface)* yang dapat membuat simulasi di MATLAB menjadi lebih sederhana dan praktis digunakan oleh para *user* (pengguna).

Berdasarkan perkembangan penelitian tentang TSP dan algoritma *Lexisearch* sebagai alat untuk pemecahan masalah, menarik untuk mempelajari lebih lanjut penyelesaian salah satu variasi TSP yaitu BTSP menggunakan algoritma *Lexisearch*. Penelitian ini merupakan penjabaran dari jurnal yang ditulis oleh Zakir H. Ahmed, dilengkapi ilustrasi proses perhitungan dengan representasi *adjacency* dan representasi *path* yang belum disajikan. Ditambahkan simulasi dengan GUI MATLAB untuk perhitungan dengan representasi *path*.

1.2. Batasan Masalah

Pembatasan masalah dalam suatu penelitian sangatlah penting untuk menghindari kesalahpahaman objek penelitian serta untuk membantu penulis lebih fokus dan terarah pada tema yang telah ditentukan. Penulisan skripsi ini dibatasi pada :

1. Kasus BTSP yang dibahas adalah *asymmetric* BTSP.

2. Perhitungan batas bawah *(Bound)* dalam algoritma *Lexisearch* menggunakan

formula yang dijelaskan Zakir H. Ahmed (2010) dalam jurnalnya.

3. Pengolahan data dilakukan dengan menggunakan *software* MATLAB R2013a.

1.3. Rumusan Masalah

Berdasarkan latar belakang dan batasan masalah yang telah diuraikan, maka permasalahan yang akan dibahas dalam skripsi ini adalah:

1. Bagaimana konsep dan langkah algoritma *Lexisearch* untuk menyelesaikan BTSP?
2. Bagaimana menyelesaikan BTSP dengan algoritma *Lexisearch* menggunakan representasi *adjacency*?
3. Bagaimana menyelesaikan BTSP dengan algoritma *Lexisearch* menggunakan representasi *path*?
4. Bagaimana membuat simulasi algoritma *Lexisearch* dengan menggunakan MATLAB?

1.4. Tujuan Penelitian

Berdasarkan rumusan masalah di atas, maka tujuan dari penelitian ini di-antaranya:

1. Menjelaskan konsep dan langkah algoritma *Lexisearch* untuk menyelesaikan BTSP.
2. Mengkaji dan menjelaskan langkah-langkah penyelesaian BTSP dengan algoritma *Lexisearch* menggunakan representasi *adjacency*.

3. Mengkaji dan menjelaskan langkah-langkah penyelesaian BTSP dengan algoritma *Lexisearch* menggunakan representasi *path*.

4. Membuat simulasi algoritma *Lexisearch* dengan menggunakan MATLAB.

1.5. Manfaat Penelitian

Hasil dari penelitian ini diharapkan memberi manfaat sebagai berikut:

1. Memberi pengetahuan tentang salah satu variasi dari permasalahan optimasi kombinatorial TSP, yaitu BTSP.

2. Memberi pengetahuan tentang algoritma *Lexisearch* dan langkah-langkahnya dalam menyelesaikan BTSP.

3. Peneliti dan pembaca dapat menjadikan penelitian ini sebagai referensi penelitian lebih lanjut.

1.6. Tinjauan Pustaka

Penulisan skripsi ini merujuk pada jurnal dengan judul *"A Lexisearch Algorithm for the Bottleneck Traveling Salesman Problem"* yang ditulis oleh Zakir H. Ahmed pada tahun 2010. Jurnal tersebut membahas tentang algoritma *Lexisearch* yang digunakan untuk menyelesaikan BTSP, khususnya dengan representasi *adjacency*. Jurnal ini yang kemudian digunakan sebagai literatur utama dalam penulisan skripsi ini.

Referensi lain yang digunakan dalam penelitian ini diantaranya tesis *"A Lexisearch Approach to Some Combinatorial Programming Problems"* yang ditulis oleh M. Ramesh pada Januari 1997. Tesis ini membahas tentang penyelesaian beberapa permasalahan kombinatorial menggunakan algoritma *Lexisearch*. Pada

tahun 2013, G. Vijaya Lakhsmi menulis jurnal *"Lexisearch Approach to Travelling Salesman Problem"* dengan pembahasan serupa tetapi dikhususkan pada masalah TSP dengan representasi *path*, sehingga penulis menggunakannya untuk membantu memahami langkah-langkah penyelesaian BTSP dengan representasi *path* yang tidak dijelaskan dalam jurnal utama.

Selain itu, penulis juga menggunakan buku *"Graphs and Applications: An Introductory Approach"* yang ditulis oleh Joan M. Aldous dan Robin J. Wilson pada tahun 2004, dan *"Combinatorial Optimization: The Traveling Salesman Problem and Its Variations"* yang ditulis oleh Gregory Gutin dan Abraham P. Punen pada tahun 2004. Buku-buku ini membahas tentang dasar-dasar teori graf dan TSP beserta variasinya, dalam hal ini BTSP. Selain buku-buku tersebut, masih ada beberapa referensi lain yang dijadikan rujukan dalam penelitian ini.

Seperti pada jurnal rujukan, penelitian ini membahas tentang penyelesaian BTSP dengan menggunakan algoritma *Lexisearch*. Pada jurnal Zakir H. Ahmed, BTSP diselesaikan hanya dengan representasi *adjacency*, dan pengolahan data dilakukan dengan *software* C++. Pada penelitian ini penyelesaian BTSP dilakukan dengan dua representasi, yaitu representasi *adjacency* dan representasi *path*. Selain itu, pengolahan data dilakukan dengan *software* MATLAB.

1.7. Sistematika Penulisan

Sistematika penulisan ini disusun untuk memberikan gambaran secara menyeluruh dan mempermudah untuk memahami penelitian ini. Secara garis besar, sistematika penulisan skripsi terdiri dari empat bab sebagai berikut:

BAB I PENDAHULUAN

Bab ini membahas tentang latar belakang, batasan masalah, rumusan masalah,

tujuan penelitian, manfaat penelitian, tinjauan pustaka, sistematika penulisan, serta metode penelitian.

BAB II DASAR TEORI

Bab ini memaparkan secara lebih jelas teori-teori yang menjadi dasar dalam memahami bab-bab selanjutnya, yaitu teori tentang graf, digraf, optimasi kombinatorial, representasi *adjacency* dan *path*, algoritma *Lexisearch*, serta pengenalan *software* MATLAB R2013a.

BAB III PEMBAHASAN

Bab ini membahas secara rinci tentang penyelesaian BTSP dengan algoritma *Lexisearch* menggunakan representasi *adjacency* dan *path* lengkap dengan simulasi perhitungannya.

BAB IV PENUTUP

Bab ini berisi kesimpulan umum dari bab-bab sebelumnya, serta saran-saran yang membangun bagi pembaca untuk penelitian lebih lanjut.

1.8. Metode Penelitian

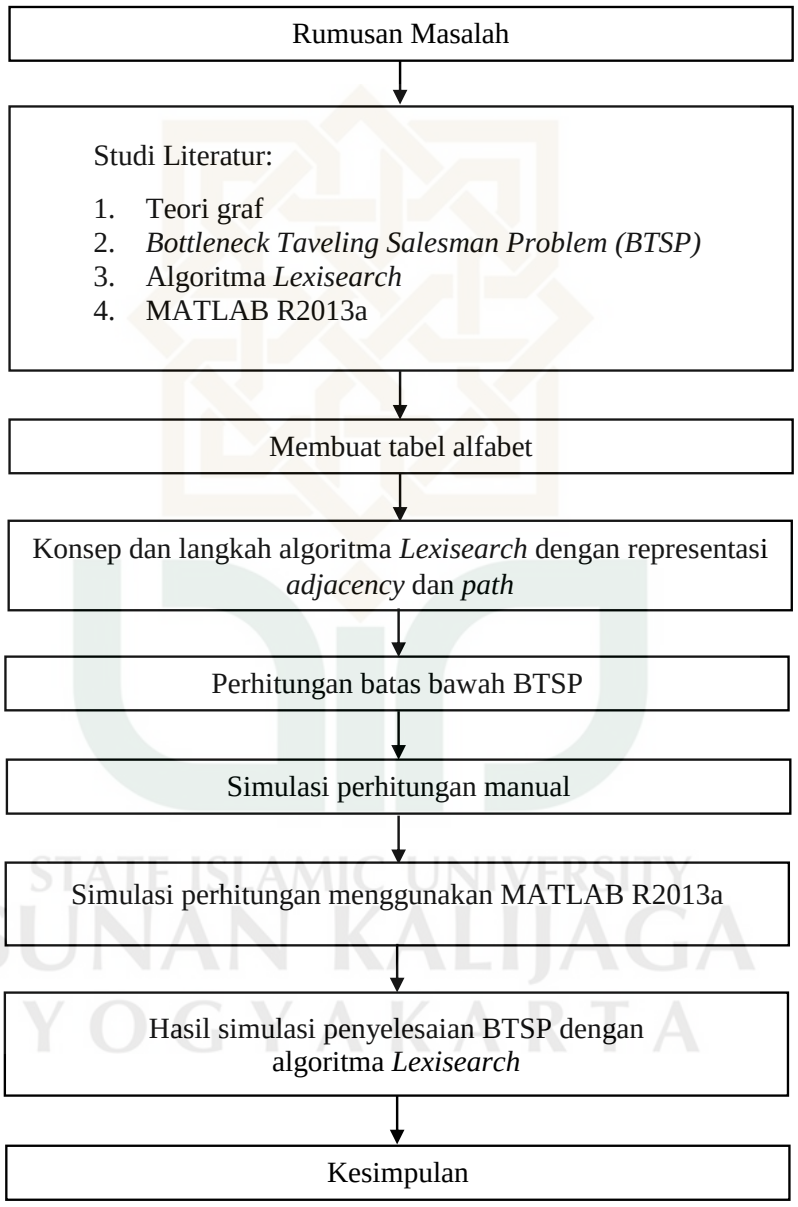
Metode yang digunakan dalam penelitian ini adalah studi literatur, yaitu penulis mempelajari sumber tertulis mengenai algoritma *Lexisearch* untuk menyelesaikan BTSP sesuai dengan tema penelitian.

Dasar teori dalam penelitian ini diawali dengan membahas graf dan bagian-bagiannya sebagai jalan untuk memahami digraf dan bagian-bagiannya. Selanjutnya dasar teori dari graf dan digraf digunakan untuk membahas TSP dan BTSP. Dasar teori tentang algoritma *Lexisearch* dimaksudkan untuk memberi gambaran tentang algoritma yang akan digunakan dalam penyelesaian BTSP. Representasi *adjacency* dan *path* dijelaskan untuk memahami proses penyelesaian algoritma *Lexi-*

search. Selanjutnya dasar teori tentang MATLAB digunakan untuk memberikan gambaran tentang *software* yang akan digunakan dalam pembuatan simulasi.

Pembahasan inti dalam penelitian ini adalah tentang langkah-langkah penyelesaian BTSP menggunakan algoritma *Lexisearch* yang mengacu pada jurnal yang ditulis oleh Zakir H. Ahmed pada tahun 2010 dengan judul *"A Lexisearch Algorithm for the Bottleneck Traveling Salesman Problem"*. Sebelum memulai algoritma *Lexisearch*, terlebih dahulu matriks jarak harus diubah menjadi tabel alfabet. Selanjutnya perhitungan batas bawah menggunakan formula yang telah dijelaskan dalam jurnal. Penyelesaian yang direpresentasikan dengan *path* tidak dibahas dalam jurnal, sehingga digunakan referensi lain. Data yang digunakan untuk simulasi perhitungan dalam penelitian ini juga diambil dari referensi lain, dan perhitungan dikonstruksi sendiri oleh penulis. Pengolahan data dilakukan dengan menggunakan *software* MATLAB R2013a.

Berikut diberikan skema dari penelitian ini.



Gambar 1.1 Skema Penelitian

BAB IV

PENUTUP

Pada bab ini akan diberikan kesimpulan dan saran-saran yang dapat diambil berdasarkan materi-materi yang telah dibahas pada bab-bab sebelumnya.

4.1. Kesimpulan

Kesimpulan yang dapat diambil penulis setelah menyelesaikan pembuatan skripsi ini adalah :

1. Konsep algoritma *Lexisearch* adalah untuk menentukan struktur di ruang solusi yang elemennya tersusun secara hierarkis, seperti susunan kata dalam kamus, dimana setiap kata rumpang merepresentasikan potongan kata (blok) dengan kata rumpang tersebut sebagai *leader*nya. Oleh karena itu, untuk menyelesaikan kasus BTSP secara sistematis diperlukan tabel alfabet sebagai acuan perhitungan, dan 'solusi terbaik' awal merupakan nilai terbesar dalam tabel alfabet tersebut. Pencarian rute optimal dilakukan secara berurutan pada setiap blok dan sub-blok. Perhitungan dengan algoritma *Lexisearch* selalu dimulai dengan memilih elemen pertama tabel alfabet, yaitu verteks $a(1,1)$ sebagai *leader* baru. Selanjutnya hitung nilai solusi rute bagian dengan mencari nilai maksimum dari nilai solusi *leader* dan nilai verteks terpilih. Langkah selanjutnya yaitu menghitung batas bawah dari *leader* terpilih. Setelah diperoleh batas bawah, lakukan pengecekan sub-*tour*. Jika *leader* membentuk sub-*tour*, maka verteks terpilih harus diganti dengan verteks pada blok selanjutnya. Apabila sudah diperoleh rute lengkap, maka diperoleh

nilai solusi dari rute sales. Nilai solusi tersebut menjadi 'solusi terbaik' jika nilainya kurang dari 'solusi terbaik' sebelumnya. Perhitungan dihentikan jika kriteria pemberhentian telah dipenuhi, dan 'solusi terbaik' saat ini adalah nilai optimal dari kasus BTSP.

2. Penyelesaian BTSP menggunakan representasi *adjacency* dilakukan dengan memilih *leader* secara berurutan dari verteks 1, kemudian verteks 2, dan seterusnya hingga verteks terakhir. Kemudian hitung nilai solusi rute bagian. Langkah selanjutnya adalah menghitung batas bawah. Seperti pada pemilihan *leader*, pehitungan batas bawah dilakukan dengan mencari nilai maksimum dari verteks 'sah' (verteks yang belum ada pada rute bagian) pertama pada baris ke $(l + 1)$ sampai baris ke-n. Setelah diperoleh batas bawah, lakukan pengecekan sub-*tour*, yaitu suatu keadaan dimana terdapat setidaknya dua *cycle* dalam satu permutasi. Setelah diperoleh rute lengkap, yaitu keadaan dimana permutasi sudah melibatkan seluruh verteks, maka diperoleh pula nilai solusi dari rute sales. Nilai solusi tersebut menjadi 'solusi terbaik' jika nilainya kurang dari 'solusi terbaik' sebelumnya. Perhitungan dihentikan jika diperoleh panjang *leader* kurang dari 1, dan 'solusi terbaik' saat ini adalah nilai optimal dari kasus BTSP.

3. Penyelesaian BTSP menggunakan representasi *path* dilakukan dengan memilih *leader* secara berkelanjutan dari verteks 1, kemudian verteks terpilih kedua, dan seterusnya hingga verteks terpilih yang terakhir. Kemudian hitung nilai solusi rute bagian. Langkah selanjutnya adalah menghitung batas bawah. Seperti pada pemilihan *leader*, pehitungan batas bawah dilakukan dengan mencari nilai maksimum dari verteks 'sah' pertama pada baris pertama yang belum terpilih sampai baris ke-m, dengan m adalah baris terakhir

yang belum terpilih. Setelah diperoleh batas bawah, lakukan pengecekan sub-*tour*, yaitu suatu keadaan dimana terdapat verteks yang berulang atau terpilih verteks 1 sebelum diperoleh rute lengkap. Setelah diperoleh rute lengkap, yaitu keadaan dimana jalur sudah melibatkan seluruh verteks dan telah kembali ke verteks 1, maka diperoleh pula nilai solusi dari rute sales. Nilai solusi tersebut menjadi 'solusi terbaik' jika nilainya kurang dari 'solusi terbaik' sebelumnya. Perhitungan dihentikan jika sudah sampai pada baris pertama dan kolom ke-n (kolom terakhir) tabel alfabet, yaitu $a(1, n)$, dan 'solusi terbaik' saat ini adalah nilai optimal dari kasus BTSP.

4. Simulasi algoritma *Lexisearch* dibuat dengan GUI MATLAB, sehingga memudahkan para pengguna untuk mengakses program. Komponen-komponen dalam GUI yang digunakan diantaranya *Push Button, Edit Text, Axes, Static Text*, dan *Table*. Perhitungan inti algoritma *Lexisearch* dibuat dengan M-File menggunakan *flow control* dan operasi matriks. Simulasi yang dibuat dalam penelitian ini adalah simulasi algoritma *Lexisearch* dengan representasi *path*. Untuk menyelesaikan kasus BTSP, matriks jarak diinputkan sendiri oleh pengguna, baik secara langsung maupun dengan mengambil data dari file *Microsoft Excel*. Hasil yang ditampilkan pada simulasi ini diantaranya rute optimal, yaitu rute yang memiliki bobot maksimum yang paling minimum, nilai optimal, yaitu bobot maksimum pada rute optimal, dan jumlah iterasi, yaitu banyaknya rute yang ditemukan sebelum diperoleh rute optimal.

4.2. Saran

Setelah menyelesaikan penelitian ini, saran-saran yang dapat penulis sampaikan adalah :

1. Penelitian ini hanya membahas algoritma *Lexisearch* untuk menyelesaikan BTSP. Oleh karena itu penelitian ini dapat dikembangkan lagi, misalnya dengan membandingkan algoritma *Lexisearch* dengan algoritma lain.

2. Dalam penelitian ini tidak dilakukan studi kasus di lapangan, sehingga dapat ditambahkan studi lapangan untuk memecahkan permasalahan yang lebih spesifik.

3. Simulasi yang dibuat dalam penelitian ini hanya algoritma *Lexisearch* dengan representasi *path*. Oleh sebab itu, pengembangan pada penelitian ini dapat dilakukan dengan menambahkan simulasi untuk menyelesaikan BTSP dengan algoritma *Lexisearch* menggunakan representasi *adjacency*.

DAFTAR PUSTAKA

Ahmed, Z. H. (2010). *"A Lexisearch Algorithm for the Bottleneck Travelling Salesman Problem"*, Department of Computer Science. Saudi Arabia : Al-Imam Muhammad Ibn Saud Islamic University.

Ahmed, Z. H. (2011). *"A Data-Guided Lexisearch Algorithm for the Asymmetric Traveling Salesman Problem"*. Department of Computer Science. Saudi Arabia : Al-Imam Muhammad Ibn Saud Islamic University.

Al Mahkya, D. (2014). Membuat GUI MATLAB Sederhana. Diakses Maret 6, 2018 dari http:// www.pojokan-artikel.com/2014/07/membuat-gui-matlab-sederhana.html

Aldous, J. M. dan Wilson, R. J.(2004), *Graph and Applications: An Introductory Approach*. Great Britain : Springer-Verlag.

Aliansyah, H., Fernando, N. dan Prima, A. (2009). *Aplikasi Pencarian Jalur Tercepat di Wilayah Jakarta Pusat Menggunakan Algoritma Genetika dan Logika Fuzzy*. Skripsi S1: Bina Nusantara University.

Firmansyah, A., (2007). *Dasar-dasar Pemrograman MATLAB*. Komunitas *eLearning* IlmuKomputer.Com

Gutin, G. dan Punen, A.P. (Eds.) (2004). *The Traveling Salesman Problem and Its Variations*. New York : Kluwer Academic Publishers.

John LaRusic (2005). The Bottleneck Traveling Salesman Problem and Some Variations. *PhD Thesis*: University of New Brunswick.

Lakhshmi, G. V. (2013) *Lexisearch Approach to Travelling Salesman Problem*. IOSR Journal of Mathematics, 6, 01-08.

M. Ramesh (1997). A Lexisearch Approach to Some Combinatorial Programming Problems. *PhD Thesis*: University of Hyderabad.

LAMPIRAN A

TABEL PENCARIAN

Simbol yang akan dipakai dalam tabel pencarian ini adalah sebagai berikut:

GS : *Go*

JB : *Jump block*

JO : *Jump out*

ST : *Sub-tour*

BS : Nilai solusi terbaik

1.1. Tabel Pencarian dengan Representasi *Adjacency*

Leader					Batas	BS	Keterangan
1	2	3	4	5			
$2 \backsim 5$					56	999	GS
	$1 \backsim 7$				56	999	ST
	$4 \backsim 11$				56	999	GS
		$5 \backsim 56$			66	999	GS
			$3 \backsim 66$		66	999	GS
				$1 \backsim 26$	66	999	GS
					BS =	66	JB, JO
		$1 \backsim 77$			77	66	JO
	$5 \backsim 22$				77	66	JB
	$3 \backsim 44$				56	66	GS

		5 ∽ 56			84	66	JB
		1 ∽ 77			77	66	JO
	2 ∽ 999				999	66	JO
4 ∽ 8					26	66	GS
	1 ∽ 7				74	66	JB
	5 ∽ 22				34	66	GS
		2 ∽ 3			66	66	JB
		1 ∽ 77			77	66	JO
	3 ∽ 44				44	66	GS
		2 ∽ 3			44	66	ST
		5 ∽ 56			56	66	GS
			2 ∽ 34		56	66	GS
				1 ∽ 26	56	66	GS
					BS =	56	JB, JO
		1 ∽ 77			77	56	JO
	2 ∽ 999				999	56	JO
3 ∽ 13					26	56	GS
	1 ∽ 7				55	56	GS
		2 ∽ 3			26	56	ST
		5 ∽ 56			56	56	JO
	4 ∽ 11				26	56	GS
		2 ∽ 3			26	56	GS
			5 ∽ 22		26	56	GS
				1 ∽ 26	26	56	GS
					BS =	26	JB, JO

		$5 \backsim 56$			56	26	JO
	$5 \backsim 22$				34	26	JB
	$2 \backsim 999$				999	26	JO
$5 \backsim 33$					33	26	STOP

1.2. Tabel Pencarian dengan Representasi *Path*

Leader					Batas	BS	Keterangan
$1 \rightarrow \alpha_1$	$\alpha_1 \rightarrow \alpha_2$	$\alpha_2 \rightarrow \alpha_3$	$\alpha_3 \rightarrow \alpha_4$	$\alpha_4 \rightarrow 1$			
$1 \rightarrow 2$					56	999	GS
	$2 \rightarrow 1$				56	999	ST
	$2 \rightarrow 4$				56	999	GS
		$4 \rightarrow 5$			77	999	GS
			$5 \rightarrow 1$		999	999	JB
			$5 \rightarrow 3$		83	999	GS
				$3 \rightarrow 1$	83	999	GS
					BS =	83	JB, JO
		$4 \rightarrow 3$			56	83	GS
			$3 \rightarrow 5$		66	83	GS
				$5 \rightarrow 1$	66	83	GS
					BS =	66	JB, JO
		$4 \rightarrow 1$			84	66	JO
	$2 \rightarrow 5$				77	66	JB
	$2 \rightarrow 3$				56	66	GS
		$3 \rightarrow 5$			84	66	JB
		$3 \rightarrow 1$			77	66	JO

	$2 \rightarrow 2$				999	66	JO
$1 \rightarrow 4$					26	66	GS
	$4 \rightarrow 5$				26	66	GS
		$5 \rightarrow 1$			44	66	ST
		$5 \rightarrow 2$			77	66	JO
	$4 \rightarrow 2$				56	66	GS
		$2 \rightarrow 1$			83	66	JB
		$2 \rightarrow 5$			77	66	JB
		$2 \rightarrow 3$			56	66	GS
			$3 \rightarrow 5$		56	66	GS
				$5 \rightarrow 1$	56	66	GS
					BS =	56	JB, JO
		$2 \rightarrow 2$			999	56	JO
	$4 \rightarrow 3$				66	56	JO
$1 \rightarrow 3$					26	56	GS
	$3 \rightarrow 2$				26	56	GS
		$2 \rightarrow 4$			26	56	GS
			$4 \rightarrow 5$		26	56	GS
				$5 \rightarrow 1$	26	56	GS
					BS =	26	JB, JO
		$2 \rightarrow 5$			84	26	JB
		$2 \rightarrow 2$			999	26	JO
	$3 \rightarrow 5$				56	26	JO
$1 \rightarrow 5$					33	26	STOP

LAMPIRAN B

SOURCE CODE SIMULASI ALGORITMA *LEXISEARCH*

```
function varargout = Algoritma_Lexisearch(varargin)
% ALGORITMA_LEXISEARCH MATLAB code for Algoritma_Lexisearch.fig
%      ALGORITMA_LEXISEARCH, by itself, creates a new
ALGORITMA_LEXISEARCH or raises the existing
%      singleton*.
%
%      H = ALGORITMA_LEXISEARCH returns the handle to a new
ALGORITMA_LEXISEARCH or the handle to
%      the existing singleton*.
%
%
ALGORITMA_LEXISEARCH('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in ALGORITMA_LEXISEARCH.M with the
given input arguments.
%
%      ALGORITMA_LEXISEARCH('Property','Value',...) creates a new
ALGORITMA_LEXISEARCH or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before Algoritma_Lexisearch_OpeningFcn
gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to
Algoritma_Lexisearch_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
Algoritma_Lexisearch

% Last Modified by GUIDE v2.5 17-Apr-2018 11:43:31

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
```

```
                   'gui_OpeningFcn',
@Algoritma_Lexisearch_OpeningFcn, ...
                   'gui_OutputFcn',
@Algoritma_Lexisearch_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Algoritma_Lexisearch is made visible.
function Algoritma_Lexisearch_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Algoritma_Lexisearch (see
VARARGIN)

% Choose default command line output for Algoritma_Lexisearch
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Algoritma_Lexisearch wait for user response (see
UIRESUME)
% uiwait(handles.figure1);
hback = axes('units','normalized','position',[0 0 1 1]);
uistack(hback,'bottom');
[back map] = imread('putih.jpg');
image(back)
colormap(map)
set(hback,'handlevisibility','off','visible','off')

axes(handles.axes1);
image(imread('logo','jpeg'));
grid off;
axis off;

set(handles.text6,'visible','off');
set(handles.text7,'visible','off');
set(handles.text8,'visible','off');
set(handles.text9,'visible','off');
set(handles.text10,'visible','off');
set(handles.text11,'visible','off');
set(handles.helpbtn,'visible','off');
```

```
set(handles.databtn,'visible','off');
set(handles.proses,'visible','off');
set(handles.clearbtn,'visible','off');
set(handles.matriks,'visible','off');
set(handles.leader,'visible','off');
set(handles.BS,'visible','off');
set(handles.iterasi,'visible','off');
set(handles.text12,'visible','off');
set(handles.text13,'visible','off');
set(handles.backbtn,'visible','off');
set(handles.backbtn1,'visible','off');
set(handles.uitable1,'visible','off');
set(handles.text14,'visible','off');
set(handles.backbtn2,'visible','off');


% --- Outputs from this function are returned to the command line.
function varargout = Algoritma_Lexisearch_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in startbtn.
function startbtn_Callback(hObject, eventdata, handles)
% hObject    handle to startbtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.startbtn,'visible','off');
set(handles.text1,'visible','off');
set(handles.text2,'visible','off');
set(handles.text3,'visible','off');
cla(handles.axes1)
set(handles.axes1,'visible','off');
set(handles.text6,'visible','off');
set(handles.text7,'visible','on');
set(handles.text8,'visible','on');
set(handles.text9,'visible','on');
set(handles.text10,'visible','on');
set(handles.text11,'visible','on');
set(handles.helpbtn,'visible','on');
set(handles.databtn,'visible','on');
set(handles.proses,'visible','on');
set(handles.clearbtn,'visible','on');
set(handles.matriks,'visible','on');
set(handles.leader,'visible','on');
set(handles.BS,'visible','on');
set(handles.iterasi,'visible','on');
set(handles.text12,'visible','off');
set(handles.text13,'visible','off');
```

```
set(handles.backbtn,'visible','off');
set(handles.backbtn2,'visible','on');



function matriks_Callback(hObject, eventdata, handles)
% hObject    handle to matriks (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of matriks as text
%        str2double(get(hObject,'String')) returns contents of
matriks as a double
A = str2num(get(handles.matriks, 'String'));
handles.A = A;
guidata(hObject, handles);


% --- Executes during object creation, after setting all
properties.
function matriks_CreateFcn(hObject, eventdata, handles)
% hObject    handle to matriks (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function leader_Callback(hObject, eventdata, handles)
% hObject    handle to leader (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of leader as text
%        str2double(get(hObject,'String')) returns contents of
leader as a double


% --- Executes during object creation, after setting all
properties.
function leader_CreateFcn(hObject, eventdata, handles)
% hObject    handle to leader (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function BS_Callback(hObject, eventdata, handles)
% hObject    handle to BS (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of BS as text
%        str2double(get(hObject,'String')) returns contents of BS
as a double


% --- Executes during object creation, after setting all
properties.
function BS_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BS (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function iterasi_Callback(hObject, eventdata, handles)
% hObject    handle to iterasi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of iterasi as text
%        str2double(get(hObject,'String')) returns contents of
iterasi as a double


% --- Executes during object creation, after setting all
properties.
function iterasi_CreateFcn(hObject, eventdata, handles)
% hObject    handle to iterasi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in databtn.
function databtn_Callback(hObject, eventdata, handles)
% hObject    handle to databtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.text6,'visible','off');
set(handles.text7,'visible','off');
set(handles.text8,'visible','off');
set(handles.text9,'visible','off');
set(handles.text10,'visible','off');
set(handles.text11,'visible','off');
set(handles.helpbtn,'visible','off');
set(handles.databtn,'visible','off');
set(handles.proses,'visible','off');
set(handles.clearbtn,'visible','off');
set(handles.matriks,'visible','off');
set(handles.leader,'visible','off');
set(handles.BS,'visible','off');
set(handles.iterasi,'visible','off');
set(handles.text12,'visible','off');
set(handles.text13,'visible','off');
set(handles.backbtn,'visible','off');
set(handles.backbtn1,'visible','on');
set(handles.uitable1,'visible','on');
set(handles.text14,'visible','on');
set(handles.backbtn2,'visible','off');
set(handles.backbtn2,'visible','off');

[filename pathname]=uigetfile({'*.xlsx'},'File Selector');
fullpathname=strcat(pathname, filename);
data=xlsread(fullpathname);
set(handles.uitable1,'Data',data);


% --- Executes on button press in helpbtn.
function helpbtn_Callback(hObject, eventdata, handles)
% hObject    handle to helpbtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.text7,'visible','off');
set(handles.text8,'visible','off');
set(handles.text9,'visible','off');
set(handles.text10,'visible','off');
set(handles.text11,'visible','off');
set(handles.helpbtn,'visible','off');
set(handles.databtn,'visible','off');
set(handles.proses,'visible','off');
set(handles.clearbtn,'visible','off');
set(handles.matriks,'visible','off');
set(handles.leader,'visible','off');
```

```
set(handles.BS,'visible','off');
set(handles.iterasi,'visible','off');
set(handles.text12,'visible','off');
set(handles.text13,'visible','on');
set(handles.backbtn,'visible','on');
set(handles.backbtn2,'visible','off');


% --- Executes on button press in proses.
function proses_Callback(hObject, eventdata, handles)
% hObject    handle to proses (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
A = handles.A;
[n m] = size (A);

if m~=n
    A=A(:, 1:n);
    [n m] = size (A);
end

big = 9999999;
s = max(A);
t = max(s);
if t > big
    big = t;
end

for i = 1:n
    A(i,i) = big;
end

% MEMBUAT TABEL ALFABET
[B, ind] = sort (A,2);

% PROGRAM INTI
x = max (B);
BS = max(x);

lead = 1;
[a b] = size(lead);
sol = 0;
nilai = 0;
index = 1;
i = 1; j = 1;
iter = 0;

% STOPPING CRITERIA
stop = 0;

for iterasi = 1:10000
if stop == 0
    if b < n
        p = ind(i,j);
        lead = [lead p];
        [a b] = size(lead);
```

```
        index = [index j];
        val = B(i,j);
        nilai = [nilai val];
        check = [sol val];
        sol2 = max(check);
    end

%     JUMP OUT
    for s = 1:b
        if sol2 >= BS || b > n
            if b == 2
                stop = 1;
                break;
            else
                u = lead(b-2);
                j = index(b-1)+1;
                p = ind(u,j);
                lead = lead(1:b-2);
                lead = [lead p];
                [a b] = size(lead);
                index = index(1:b-1);
                index = [index j];
                val = B(u,j);
                nilai = nilai(1:b-1);
                sol = max(nilai);
                nilai = [nilai val];
                check = [sol val];
                sol2 = max(check);
                i = u;
                break;
            end
        end
    end

    %MENCARI SUB TOUR
    if b <= n || p == 1 && stop==0
        for v = 1:b
        g = p;
        for w = 1:b-1
            if g ~= lead(w)
                continue;
            else
%                     GO
                if j < n
                    j = j+1;
                    p = ind(i,j);
                    lead = lead(1:b-1);
                    lead = [lead p];
                    [a b] = size(lead);
                    index = index(1:b-1);
                    index = [index j];
                    val = B(i,j);
                    nilai = nilai(1:b-1);
                    nilai = [nilai val];
                    check = [sol val];
                    sol2 = max(check);
                    break;
```

```
                    end;

                    if j==n
                        if b == 2
                            stop = 1;
                            break;
                        else
                            u = lead(b-2);
                            j = index(b-1)+1;
                            p = ind(u,j);
                            lead = lead(1:b-2);
                            lead = [lead p];
                            [a b] = size(lead);
                            index = index(1:b-1);
                            index = [index j];
                            val = B(u,j);
                            nilai = nilai(1:b-1);
                            sol = max(nilai);
                            nilai = [nilai val];
                            check = [sol val];
                            sol2 = max(check);
                            i = u;
                            break;
                        end
                    end
                end
            end
        end
    end

%     TOUR LENGKAP
    if b == n && stop==0
        i = p;
        for k = 1:n
            if ind(i,k) == 1
                p = ind(i,k);
                j = j+1;
                check = [sol2, B(i,k)];
                sol2 = max(check);
            end
        end

%         JUMP OUT
        if sol2 >= BS || b > n
            u = lead(b-3);
            j = index(b-2)+1;
            p = ind(u,j);
            lead = lead(1:b-3);
            lead = [lead p];
            [a b] = size(lead);
            index = index(1:b -1);
            index = [index j];
            val = B(u,j);
            nilai = nilai(1:b-1);
            sol = max(nilai);
            nilai = [nilai val];
            check = [sol val];
```

```
            sol2 = max(check);
            i = u;
        end

%         HIMPUNAN SOLUSI
        if p == 1 && b == n
            iter = iter+1;
            BS = sol2;
            sn = BS;
            yg = lead;

%             JUMP OUT
            u = lead(b-2);
            j = index(b-1)+1;
            p = ind(u,j);
            lead = lead(1:b-2);
            lead = [lead p];
            [a b] = size(lead);
            index = index(1:b-1);
            index = [index j];
            val = B(u,j);
            nilai = nilai(1:b-1);
            sol = max(nilai);
            nilai = [nilai val];
            check = [sol val];
            sol2 = max(check);
            i = u;
        end
    end

%     BATAS BAWAH
    if j < n && b < n && stop == 0
        x = ind(:,1);
        C = B(:,1);
        for e = 1:n
            for d = 1:n
                for y = 1:b-1
                    x(lead(y),1) = 0;
                    C(lead(y),1) = 0;

                    if x(e,1) == lead(y+1)
                        x(e,1) = ind(e,d+1);
                        C(e,1) = B(e,d+1);
                        break;
                    end
                end
            end
        end
        o = max(C);
        z = [sol2 o];
        bound = max(z);
    end

%     JUMP BLOCK
    if bound >= BS && b<n && j<n && stop ==0
        for w=1:b
            p = ind(i,j);
```

```
            for v = 1:b
                q(v)=p;
            end

            for t=1:b
                if q(t) == lead(t)
                    j=j+1;
                    break
                end
            end

%             GO
            if j < n
                p = ind(i,j);
                lead = lead(1:b-1);
                lead = [lead p];
                [a b] = size(lead);
                index = index(1:b-1);
                index = [index j];
                val = B(i,j);
                nilai = nilai(1:b-1);
                nilai =[nilai val];
                check = [sol val];
                sol2 = max(check);
                break;
            end

%             JUMP OUT
            if j == n && stop == 0
                if b == 2
                    stop = 1;
                else
                    u = lead(b-2);
                    j = index(b-1)+1;
                    p = ind(u,j);
                    lead = lead(1:b-2);
                    lead = [lead p];
                    [a b] = size(lead);
                    index = index(1:b-1);
                    index = [index j];
                    val = B(u,j);
                    nilai = nilai(1:b-1);
                    sol = max(nilai);
                    nilai = [nilai val];
                    check = [sol val];
                    sol2 = max(check);
                    i=u;
                    break
                end
            end
        end
    end

        %MENCARI SUB TOUR
    if b <= n || p == 1 && stop==0
```

```
        for v = 1:b
            g = p;
            for w = 1:b-1
                if g ~= lead(w)
                    continue;
                else
%                       GO
                    if j < n
                        j = j+1;
                        p = ind(i,j);
                        lead = lead(1:b-1);
                        lead = [lead p];
                        [a b] = size(lead);
                        index = index(1:b-1);
                        index = [index j];
                        val = B(i,j);
                        nilai = nilai(1:b-1);
                        nilai = [nilai val];
                        check = [sol val];
                        sol2 = max(check);
                        break;
                    end;

%                     JUMP OUT
                    if j==n
                        if b == 2
                            stop = 1;
                            break;
                        else
                            u = lead(b-2);
                            j = index(b-1)+1;
                            p = ind(u,j);
                            lead = lead(1:b-2);
                            lead = [lead p];
                            [a b] = size(lead);
                            index = index(1:b-1);
                            index = [index j];
                            val = B(u,j);
                            nilai = nilai(1:b-1);
                            sol = max(nilai);
                            nilai = [nilai val];
                            check = [sol val];
                            sol2 = max(check);
                            i = u;
                            break;
                        end
                    end
                end
            end
        end
    end
    i = p;
    j = 1;
    sol = sol2;
    [a b] = size(lead);
```

```
end
end

% MENAMPILKAN OUTPUT
set(handles.leader, 'string', yg);
set(handles.BS, 'string', sn);
set(handles.iterasi, 'string', iter);



% --- Executes on button press in clearbtn.
function clearbtn_Callback(hObject, eventdata, handles)
% hObject    handle to clearbtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.matriks, 'String', '');
set(handles.leader, 'String', '');
set(handles.BS, 'String', '');
set(handles.iterasi, 'String', '');
lead = 1;
BS = 0;



% --- Executes on button press in backbtn.
function backbtn_Callback(hObject, eventdata, handles)
% hObject    handle to backbtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.text7,'visible','on');
set(handles.text8,'visible','on');
set(handles.text9,'visible','on');
set(handles.text10,'visible','on');
set(handles.text11,'visible','on');
set(handles.helpbtn,'visible','on');
set(handles.databtn,'visible','on');
set(handles.proses,'visible','on');
set(handles.clearbtn,'visible','on');
set(handles.matriks,'visible','on');
set(handles.leader,'visible','on');
set(handles.BS,'visible','on');
set(handles.iterasi,'visible','on');
set(handles.text13,'visible','off');
set(handles.backbtn,'visible','off');
set(handles.backbtn2,'visible','on');



% --- Executes on button press in backbtn1.
function backbtn1_Callback(hObject, eventdata, handles)
% hObject    handle to backbtn1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.text7,'visible','on');
set(handles.text8,'visible','on');
set(handles.text9,'visible','on');
set(handles.text10,'visible','on');
```

```
set(handles.text11,'visible','on');
set(handles.helpbtn,'visible','on');
set(handles.databtn,'visible','on');
set(handles.proses,'visible','on');
set(handles.clearbtn,'visible','on');
set(handles.matriks,'visible','on');
set(handles.leader,'visible','on');
set(handles.BS,'visible','on');
set(handles.iterasi,'visible','on');
set(handles.backbtn1,'visible','off');
set(handles.uitable1,'visible','off');
set(handles.text14,'visible','off');
set(handles.backbtn2,'visible','on');


% --- Executes on button press in backbtn2.
function backbtn2_Callback(hObject, eventdata, handles)
% hObject    handle to backbtn2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.startbtn,'visible','on');
set(handles.text1,'visible','on');
set(handles.text2,'visible','on');
set(handles.text3,'visible','on');
cla(handles.axes1)
set(handles.axes1,'visible','on');
set(handles.text6,'visible','off');
set(handles.text7,'visible','off');
set(handles.text8,'visible','off');
set(handles.text9,'visible','off');
set(handles.text10,'visible','off');
set(handles.text11,'visible','off');
set(handles.helpbtn,'visible','off');
set(handles.databtn,'visible','off');
set(handles.proses,'visible','off');
set(handles.clearbtn,'visible','off');
set(handles.matriks,'visible','off');
set(handles.leader,'visible','off');
set(handles.BS,'visible','off');
set(handles.iterasi,'visible','off');
set(handles.backbtn2,'visible','off');

axes(handles.axes1);
image(imread('logo','jpeg'));
grid off;
axis off;
```

DAFTAR RIWAYAT HIDUP

A. Data Pribadi

Nama Lengkap : Siti Bahjatun Saniyah

Tempat, Tanggal Lahir : Sleman, 16 Juli 1996

Jenis Kelamin : Perempuan

Alamat : Ngino XII RT 04 RW 34, Desa Margoagung, Kecamatan Seyegan, Kabupaten Sleman

Agama : Islam

Status : Belum Menikah

No. HP : 082323080223

E-mail : bahjatun.saniyah@gmail.com

B. Riwayat Pendidikan Formal

1. (2001 – 2002) TK ABA Margokaton 1
2. (2002 – 2008) SD Negeri Ngino 2
3. (2008 – 2011) SMP Negeri 1 Godean
4. (2011 – 2014) MAN Yogyakarta 3
5. (2014 – 2018) Universitas Islam Negeri Sunan Kalijaga Yogyakarta