ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - IF184802

STUDI PERBANDINGAN KINERJA ALGORITMA DIJKSTRA DAN SHORTEST PATH FASTER ALGORITHM SEBAGAI METODE PENYELESAIAN MINIMUM STEINER TREE PADA STUDI KASUS E-OLYMP 1445 ROAD NETWORK

NURLITA DHUHA FATMAWATI
NRP 05111640000092

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

Dosen Pembimbing 2
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*]

TUGAS AKHIR - IF184802

STUDI PERBANDINGAN KINERJA ALGORITMA DIJKSTRA DAN SHORTEST PATH FASTER ALGORITHM SEBAGAI METODE PENYELESAIAN MINIMUM STEINER TREE PADA STUDI KASUS E-OLYMP 1445 ROAD NETWORK

NURLITA DHUHA FATMAWATI
NRP 05111640000092

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

Dosen Pembimbing 2
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*]

UNDERGRADUATE THESES - IF184802

PERFORMANCE COMPARATIVE STUDY BETWEEN DIJKSTRA AND SHORTEST PATH FASTER ALGORITHM TO SOLVE MINIMUM STEINER TREE ON E-OLYMP 1445 ROAD NETWORK CASE STUDY

NURLITA DHUHA FATMAWATI
NRP 05111640000092

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

Supervisor 2
M. M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil.

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

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

STUDI PERBANDINGAN KINERJA ALGORITMA DIJKSTRA DAN SHORTEST PATH FASTER ALGORITHM SEBAGAI METODE PENYELESAIAN MINIMUM STEINER TREE PADA STUDI KASUS E-OLYMP 1445 ROAD NETWORK

TUGAS AKHIR

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

Oleh:
Nurlita Dhuha Fatmawati
NRP. 05111640000092

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Rully Soelaiman, S.Kom., M.Kom. .....
NIP. 19700213199402100 (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*]

ABSTRAK

STUDI PERBANDINGAN KINERJA ALGORITMA DIJKSTRA DAN SHORTEST PATH FASTER ALGORITHM SEBAGAI METODE PENYELESAIAN MINIMUM STEINER TREE PADA STUDI KASUS E-OLYMP 1445 ROAD NETWORK

Nama	:	Nurlita Dhuha Fatmawati
NRP	:	05111640000092
Departemen	:	Departemen Teknik Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas, ITS
Pembimbing I	:	Rully Soelaiman, S.Kom., M.Kom.
Pembimbing II	:	M. M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil.

Abstrak

Steiner Tree berperan besar dalam permodelan VLSI chip, penyelesaian permasalahan routing, dan networking. Permasalahan yang timbul selanjutnya adalah bagaimana cara mengkomputasi Minimum Steiner Tree dengan efisien. Topik Tugas Akhir ini mengulas dua algoritma yang digunakan untuk menyelesaikan permasalahan Minimum Steiner Tree dengan efisien, yaitu menggunakan Dijkstra dan Shortest Path Faster Algorithm. Melalui pengujian dan studi kasus, didapat hasil bahwa Shortest Path Faster Algorithm memiliki kinerja yang lebih baik dari Dijkstra untuk menyelesaikan permasalahan Minimum Steiner Tree.

Kata Kunci: shortest path faster algorithm; dijkstra; minimum steiner tree; teori graf;

[*Halaman ini sengaja dikosongkan*]

ABSTRACT

PERFORMANCE COMPARATIVE STUDY BETWEEN DIJKSTRA AND SHORTEST PATH FASTER ALGORITHM TO SOLVE MINIMUM STEINER TREE ON E-OLYMP 1445 ROAD NETWORK CASE STUDY

Name	:	Nurlita Dhuha Fatmawati
Student ID	:	05111640000092
Department	:	Informatics Engineering Department, Faculty of Intelligent Electrical and Informatics Technology, ITS
Supervisor I	:	Rully Soelaiman, S.Kom., M.Kom.
Supervisor II	:	M. M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil.

Abstract

Steiner Tree plays a major role in modeling VLSI chips, solving routing, and networking problems. The next problem that arises is how to efficiently compute the Minimum Steiner Tree. This Final Project Topic reviews two algorithms that are used to efficiently solve the Minimum Steiner Tree problem, using Dijkstra and Shortest Path Faster Algorithm. Through testing and case studies, the results show that the Shortest Path Faster Algorithm has better performance than Dijkstra to solve the Minimum Steiner Tree problem.

Keywords: shortest path faster algorithm; dijkstra; minimum steiner tree; draph theory;

[*Halaman ini sengaja dikosongkan*]

KATA PENGANTAR

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

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

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

1. Allah SWT atas segala nikmat dan rahmat yang diberikan kepada penulis selama ini.
2. Ibu, Bapak, Kakak, dan keluarga penulis yang selalu memberikan dukungan dan kasih sayang untuk penulis yang menjadi semangat selama perkuliahan maupun pengerjaan Tugas Akhir.
3. Bapak Rully Soelaiman S.Kom.,M.Kom., selaku pembimbing penulis. Ucapan terima kasih penulis sampaikan atas segala perhatian, didikan, pengajaran, dan nasihat yang telah diberikan oleh beliau selama masa studi penulis. Berkat beliau pengerjaan TA ini dapat diselesaikan dengan lancar.
4. Bapak M. M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil., selaku pembimbing penulis. Ucapan terima kasih penulis sampaikan atas segala perhatian, didikan, pengajaran, dan nasihatnya. Berkat beliau buku TA ini dapat diselesaikan dengan baik.
5. Seluruh tenaga pelajar dan karyawan Departemen Informati-

ka, Institut Teknologi Sepuluh Nopember yang telah memberi ilmu dan waktunya untuk mempersiapkan penulis agar siap menghadapi masa depan.

6. Fadhil Musaad yang telah memberikan banyak dukungan berupa semangat dan bantuan untuk penulis selama pengerjaan TA ini.
7. Jonathan, Chendrasena, Rizaldi, William, Chyntia, Rogo selaku teman-teman *administrator* Laboratorium Pemrograman 2 yang telah menjadi keluarga dan membantu penulis selama di kampus.
8. Ferdinand, Yolanda, Dewi Ayu, Alvin, Modista selaku teman-teman angkatan 2016 yang telah menemani penulis selama masa studi di Departemen Informatika, Institut Teknologi Sepuluh Nopember.
9. William, Michael, Ferdinand, Fadhil, Affan selaku teman-teman seperjuangan dalam *Competitive Programming*.
10. Teman-teman, kakak-kakak, dan adik-adik mahasiswa Departemen Informatika, Institut Teknologi Sepuluh Nopember, yang senantiasa membantu, menemani, memberi semangat dan meninggalkan kenangan selama kurang lebih 3.5 tahun masa studi penulis.
11. Seluruh pihak yang tidak bisa penulis sebutkan satu-persatu yang telah memberikan dukungan selama penulis menyelesaikan TA ini.

Penulis menyadari bahwa buku ini jauh dari kata sempurna. Maka dari itu, penulis memohon maaf apabila terdapat salah kata maupun makna pada buku ini. Akhir kata, penulis mempersembahkan buku ini sebagai wujud nyata kontribusi penulis dalam ilmu pengetahuan.

Surabaya, 12 Desember 2019

Nurlita Dhuha Fatmawati

[*Halaman ini sengaja dikosongkan*]

DAFTAR ISI

[*Halaman ini sengaja dikosongkan*]

DAFTAR GAMBAR

DAFTAR TABEL

[*Halaman ini sengaja dikosongkan*]

DAFTAR PSEUDOCODE

[*Halaman ini sengaja dikosongkan*]

DAFTAR KODE SUMBER

[*Halaman ini sengaja dikosongkan*]

DAFTAR NOTASI

V	Himpunan dari vertex suatu graf.
E	Himpunan dari edge suatu graf.
S	Himpunan dari *Steiner Point*.
P	Himpunan dari koordinat hasil pemampatan koordinat.
$G(V, E)$	Graf yang merupakan pasangan dari himpunan *vertex* dan *edge*.
$w(a, b)$	Bobot dari *edge* yang menghubungkan *vertex* a dan *vertex* b.
$d(a, b)$	Jarak Manhattan dari titik a ke titik b.
$ST_{i,j,K}$	*Steiner Tree* dengan (i, j) adalah titik koordinat *root Steiner Tree*, dan K adalah sebuah himpunan berisi *Steiner Points* pada *Steiner Tree* tersebut.
$MST_{i,j,K}$	*Minimum Steiner Tree* dengan (i, j) adalah titik koordinat *root Minimum Steiner Tree*, dan K adalah sebuah himpunan berisi *Steiner Points* pada *Minimum Steiner Tree* tersebut.
$T_{i,j}$	Himpunan dari *vertex* tetangga dari titik (i, j).
K_n	*Steiner Points* ke - n pada himpunan K.
$est(i, j)$	Perkiraan bobot dari *vertex* i ke *vertex* j.

[*Halaman ini sengaja dikosongkan*]

BAB I
PENDAHULUAN

Pada bab ini, akan dijelaskan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi pengerjaan, dan sistematika penulisan tugas akhir.

1.1 Latar Belakang

Permasalahan Steiner Tree adalah salah satu permasalahan NP-hard yang terkenal. Diberikan suatu graf $G = (V, E)$, bobot edge $c : E \rightarrow R+$, dan sebuah set $Y \subseteq V$, temukan *tree* $T \subseteq E$ yang menghubungkan seluruh terminal dengan jumlah bobot minimal. *Steiner Tree* berperan besar dalam permodelan VLSI *chip*, penyelesaian permasalahan *routing* dan *networking*. Permasalahan yang timbul selanjutnya adalah bagaimana cara menghitung *Minimum Steiner Tree* dengan efisien.

Terdapat beberapa solusi aproksimasi untuk permasalahan ini, yang pertama adalah solusi jarak terpendek, cari *tree* dengan jalur terpendek dari suatu *vertex* ke seluruh graf. Pangkas bagian *tree* yang tidak menuju suatu *vertex*. Kompleksitas $O(N^2)$. Selanjutnya algoritma *greedy* atau *nearest participant first* (Takahashi, Matsuyama 1980). Mulai dari suatu *vertex*, cari *vertex* terdekat dari *tree* yang sudah terbentuk, gabungkan *vertex* tersebut dengan *tree*, ulang sampai *tree* terhubung dengan semua *vertex*. Kompleksitas $O(M * N^2)$.

Topik tugas akhir ini mengacu pada permasalahan pencarian *Minimum Steiner Tree* pada E-Olymp dengan kode 1445 berjudul Road Network [3]. Inti permasalahan pada Road Network adalah menghitung jarak yang dibutuhkan untuk membangun jalan yang menghubungkan N pangkalan militer sehingga semua titik terhubung secara langsung maupun tidak langsung dengan $1 \leq N \leq 10$. Pertanyaan utama yang mendasari permasalahan Road Network adalah

bagaimana cara menghitung jarak minimal yang dibutuhkan untuk menghubungkan N pangkalan militer secara efisien.

Selain menggunakan algoritma naif dan pendekatan di atas, terdapat dua algoritma yang bisa diadaptasikan untuk mencari *Minimum Steiner Tree* yaitu algoritma Dijkstra dan *Shortest Path Faster Algorithm*. Algoritma Dijkstra menyelesaikan permasalahan *single-source shortest path* pada *weighted graph* $G(V, E)$ untuk kasus di mana semua bobot pada *edge* adalah sebuah bilangan nonnegatif. Algoritma ini berulang kali memilih sebuah *vertex* $u \in V$ yang memiliki perkiraan bobot minimal. Pada permasalahan *Minimum Steiner Tree*, algoritma Dijkstra digunakan untuk mencari panjang minimal tiap konfigurasi *Steiner Tree* .

Kemudian *Shortest Path Faster Algoritm* (SPFA), SPFA juga menyelesaikan permasalahan *single-source shortest path* pada *weighted graph* $G(V, E)$. Berbeda dengan algoritma Dijkstra, algoritma ini berulang kali memilih sebuah *vertex* $u \in V$ manapun yang memiliki kemungkinkan untuk terjadi *relaxation*, tanpa mencari vertex yang memiliki perkiraan bobot minimal.

Karena kedua algoritma di atas memiliki kompleksitas yang hampir sama, diduga kedua algoritma tersebut dapat mencari *Minimum Steiner Tree* dengan efektifitas dan efisiensi yang setara.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut :

1. Bagaimana kinerja algoritma Dijkstra yang diimplementasikan dalam menyelesaikan permasalahan *Minimum Steiner Tree* pada E-Olymp 1445 - Road Network?
2. Bagaimana kinerja *Shortest Path Faster Algorithm* yang diimplementasikan dalam menyelesaikan permasalahan *Minimum Steiner Tree* pada E-Olymp 1445 - Road Network?
3. Bagaimana perbandingan kinerja algoritma Dijkstra dan

Shortest Path Faster Algorithm pada saat menyelesaikan permasalahan *Minimum Steiner Tree* pada E-Olymp 1445 - Road Network?

1.3 Batasan Masalah

Permasalahan yang dibahas pada tugas akhir ini memiliki beberapa batasan, yaitu sebagai berikut :

1. Perbandingan antara algoritma Dijkstra dan *Shortest Path Faster Algorithm* sebagai penyelesaian permasalahan *Minimum Steiner Tree* pada E-Olymp 1445 - Road Network.
2. Pemilihan algoritma Dijkstra sebagai salah satu metode terbatas pada observasi penulis.
3. Pemilihan *Shortest Path Faster Algorithm* sebagai salah satu metode terbatas pada observasi penulis.

Berikut merupakan batasan pada situs *E-Olymp* :

1. Implementasi dilakukan menggunakan bahasa pemrograman C++.
2. Batas nilai N di antara 1 sampai 10.
3. Batas nilai x di antara -1000 sampai 1000 yang merupakan titik absis dari pangkalan militer.
4. Batas nilai y di antara -1000 sampai 1000 yang merupakan titik ordinat dari pangkalan militer.
5. N, x, dan y adalah bilangan bulat.
6. Batas waktu yang diberikan adalah 2 detik.
7. Batas memori yang diberikan adalah 64 MB.
8. Dataset yang digunakan untuk pengujian kinerja algoritma adalah dataset pada permasalahan E-Olymp 1445 - Road Network.

1.4 Tujuan

Tujuan tugas akhir ini adalah mengetahui algoritma dengan kinerja yang lebih baik antara algoritma Dijkstra dan *Shortest Path Faster Algorithm* untuk menyelesaikan permasalahaan *Minimum Steiner Tree* pada E-Olymp 1445 - Road Network.

1.5 Manfaat

Tugas akhir ini mampu memberikan pemahaman algoritma yang tepat di antara algoritma Dijkstra dan *Shortest Path Faster Algorithm* dalam menyelesaikan permasalahan *Minimum Steiner Tree* dengan efisien.

1.6 Metodologi

Metodologi pengerjaan yang digunakan pada tugas akhir ini memiliki beberapa tahapan. Tahapan-tahapan tersebut yaitu :

1. Penyusunan proposal
   Pada tahapan ini penulis memberikan penjelasan mengenai apa yang penulis akan lakukan dan mengapa tugas akhir ini dilakukan. Penjelasan tersebut dituliskan dalam bentuk proposal tugas akhir.
2. Studi literatur
   Pada tahapan ini penulis mengumpulkan referensi yang diperlukan guna mendukung pengerjaan tugas akhir. Referensi yang digunakan dapat berupa hasil penelitian yang sudah pernah dilakukan, buku, artikel internet, atau sumber lain yang bisa dipertanggungjawabkan.
3. Implementasi algoritma
   Pada tahapan ini penulis mulai mengembangkan algoritma yang digunakan untuk menyelesaikan permasalahan *Minimum Steiner Tree* .
4. Pengujian dan evaluasi

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

5. Penyusunan buku
   Pada tahapan ini penulis menyusun hasil pengerjaan tugas akhir mengikuti format penulisan tugas akhir.

1.7 Sistematika Penulisan

Sistematika laporan tugas akhir yang akan digunakan adalah sebagai berikut :

1. BABI : PENDAHULUAN
   Bab ini berisi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi dan sistematika penulisan tugas akhir.
2. BAB II : DASAR TEORI
   Bab ini berisi dasar teori mengenai permasalahan dan algoritma penyelesaian yang digunakan dalam tugas akhir
3. BAB III : DESAIN
   Bab ini berisi desain algoritma dan struktur data yang digunakan dalam penyelesaian permasalahan.
4. BAB IV : IMPLEMENTASI
   Bab ini berisi implementasi berdasarkan desain algortima yang telah dilakukan pada tahap desain.
5. BAB V : UJI COBA DAN EVALUASI
   Bab ini berisi uji coba dan evaluasi dari hasil implementasi yang telah dilakukan pada tahap implementasi.
6. BAB VI : PENUTUP
   Bab ini berisi kesimpulan dan saran yang didapat dari hasil uji coba yang telah dilakukan.

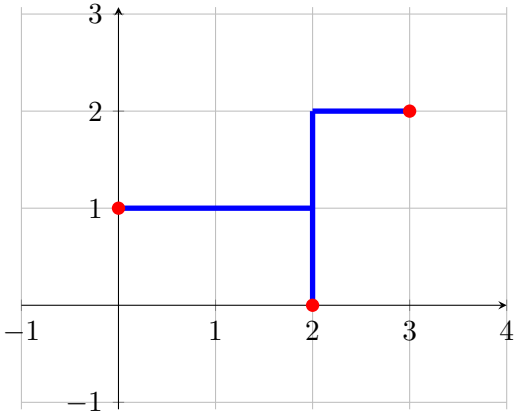
[*Halaman ini sengaja dikosongkan*]

BAB II
DASAR TEORI

Pada bab ini, dasar teori yang digunakan sebagai landasan pengerjaan Tugas Akhir ini akan dijabarkan. Pertama, bab ini akan menjelaskan mengenai deskripsi soal E-Olymp 1445 - Road Network. Kemudian, pembahasan mengenai analisis soal beserta dengan beberapa metode penyelesaian yang dapat digunakan akan dijabarkan.

2.1 Deskripsi Permasalahan Umum

Diberikan N titik pada Bidang Kartesius. Tujuan dari permasalahan ini adalah menentukan jarak terpendek untuk menghubungkan semua titik tersebut. Besarnya batasan pada permasalahan ini adalah $1 \leq N \leq 10$ dengan $-1000 \leq x, y \leq 1000$ dengan batas waktu 2 detik untuk setiap kasus uji yang diberikan. Contoh solusi optimal untuk kasus uji 3 titik dapat dilihat pada Gambar 2.1.



Gambar 2.1 Contoh Solusi Optimal

2.2 Masalah Akibat Penyelesaian Secara Naif

Pendekatan umum sekaligus naif untuk menyelesaikan permasalahan ini adalah dengan menggunakan pendekatan *Backtrack Recursion*. *Backtrack* dilakukan untuk mencoba semua konfigurasi garis untuk menghubungkan semua titik yang diberikan. Dimulai dengan sebuah konfigurasi garis T yang berisi salah satu titik. Selama T belum menghubungkan semua titik yang diberikan, dicari titik terdekat dari T yang belum terhubung dengan T lalu titik tersebut dihubungkan dengan T. *Basecase* pada persoalan ini adalah ketika sudah terbentuk konfigurasi garis dengan semua titik yang diberikan sudah terhubung kemudian dibandingkan mana konfigurasi garis yang memiliki total panjang garis minimal. Kompleksitas untuk *Backtrack Recursion* adalah $O(2^M)$. Batasan koordinat yang diberikan soal adalah $-1000 \leq x, y \leq 1000$, sehingga terdapat 4×10^6 titik. Untuk mendapatkan solusi optimal pada *worst case* dibutuhkan komputasi sebanyak $2^{4 \times 10^6}$ *Time limit* untuk soal ini adalah 2 detik. Dalam satu detik, komputasi yang bisa dilakukan adalah sebanyak sekitar 10^7. Sehingga pendekatan secara naif terlalu lama dan dibutuhkan pendekatan lain untuk menyelesaikan permasalahan ini.

2.3 Pemodelan Steiner Tree

Diberikan *undirected* graf $G = (V, E)$ dimana $V = v_1, v_2, v_3, ..., v_n$ adalah himpunan dari *vertex* pada G, $E \subseteq \{v_i, v_j | v_i \in V, v_j \in V, v_i \neq v_j\}$ adalah himpunan dari *edge* pada G. *Tree* dari G adalah *connected subgraph* dari G dimana jika suatu *edge* dihilangkan maka *subgraph* tersebut menjadi tidak terhubung [1]. $S \subseteq V$ adalah himpunan *vertex* yang disebut *Steiner Points*. *Steiner Tree* dari G dan S adalah sebuah *tree* yang membentang melalui S [5]. *Steiner Tree* dapat melalui *vertex* di luar S. *Minimum Steiner Tree* dari G dan S adalah sebuah *Steiner Tree* dimana total bobot minimal di antara semua *Steiner Tree* dari G dan

S. Contoh *Minimum Steiner Tree* dapat dilihat pada Gambar 2.2. Pemodelan graf dari permasalahan E-Olymp 1445 - Road Network memiliki properti yang sama dengan *Steiner Tree* . G direpresentasikan sebagai Bidang Kartesius. V direpresentasikan sebagai semua titik pada Bidang Kartesius. Garis yang menghubungkan setiap vertex yang bertetangga sebagai E, suatu vertex A bertetangga dengan vertex B jika bobotnya bernilai 1, dapat dilihat pada Persamaan 2.1.

$$E \subseteq \{v_i, v_j | v_i \in V, v_j \in V, v_i \neq v_j, w(v_i, v_j) = 1\} \tag{2.1}$$

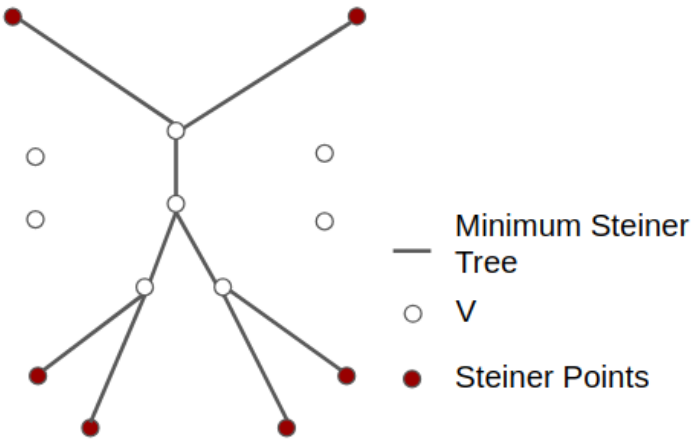
S atau *Steiner Points* adalah N titik yang diberikan. Tujuan dari permasalahan tersebut adalah mencari sebuah *tree* yang membentang melalui *Steiner Points* dengan bobot minimal. Sehingga permasalahan E-Olymp 1445 - Road Network dapat dimodelkan menjadi permasalahan *Minimum Steiner Tree*. Contoh pemodelan *Minimum Steiner Tree* pada permasalahan E-Olymp 1445 - Road Network dapat dilihat pada Gambar 2.3.

2.4 Pemampatan Koordinat (Coordinate Compression)

Batasan koordinat yang diberikan soal adalah $-1000 \leq x, y \leq 1000$. Sedangkan banyaknya *Steiner Points* maksimal adalah 10. Cara mendapatkan bobot minimal untuk menghubungkan dua titik pada Bidang Kartesius adalah dengan melewati titik perpotongannya. Maka, koordinat bisa dimampatkan dengan menyimpan suatu himpunan P berisi titik perpotongan dari setiap pasang titik pada S.

$$P = \{(\pi_1(i), \pi_2(j)) | i, j \in S, i \neq j\}$$

Untuk *Steiner Points* yaitu $(0, 4), (1, 5), (3, 0), (4, 1)$, hasil dari pemampatan koordinatnya yaitu sebagai berikut.



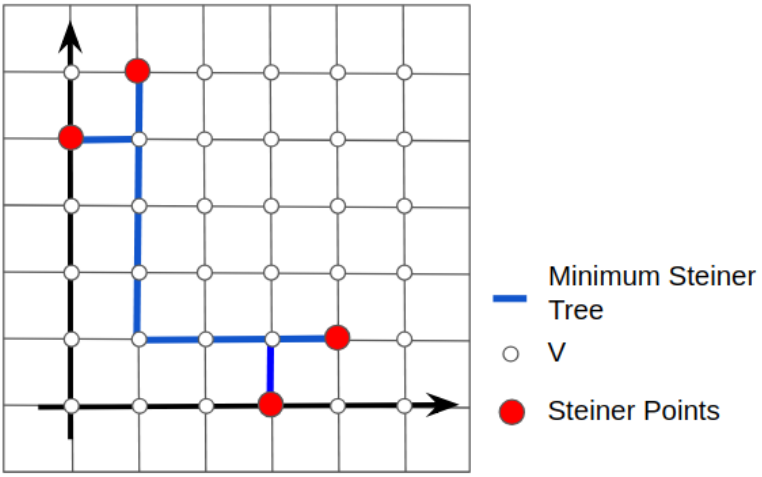
Gambar 2.2 Contoh *Minimum Steiner Tree*

- (0,0)
- (1,0)
- (3,0)
- (4,0)
- (0,1)
- (1,1)
- (3,1)
- (4,1)
- (0,4)
- (1,4)
- (3,4)
- (4,4)
- (0,5)
- (1,5)
- (3,5)
- (4,5)

Untuk visualisasi dapat dilihat pada Gambar 2.4.

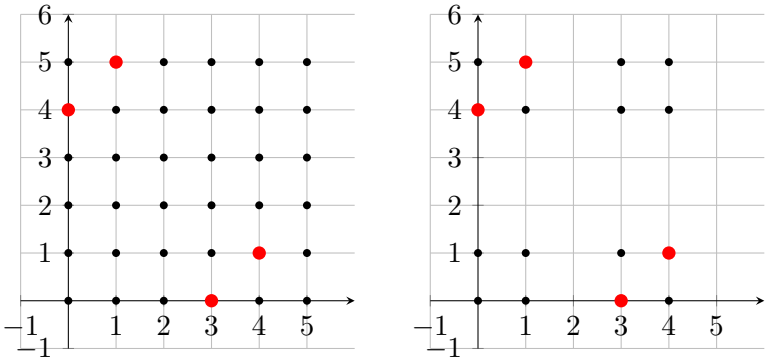
2.5 Analisis Submasalah Optimal

Pada subbab ini dijelaskan mengenai submasalah-submasalah yang jawabannya dapat membangun jawaban akhir. Hasil akhir yang dicari adalah *Steiner Tree* yang membentang melalui S. Dari semua kemungkinan jawaban, dicari *Steiner Tree* yang memiliki bobot minimal. Sebuah *Steiner Tree* dapat



Gambar 2.3 Contoh Pemodelan *Minimum Steiner Tree* pada Permasalahan E-Olymp 1445 - Road Network

dibangun dengan digabungkannya dua *Steiner Tree* berbeda. Pada metode ini, dibutuhkan penambahan properti *root*. *Root* adalah suatu *vertex* pada *tree* yang tidak mempunyai *parent vertex*. Suatu *Steiner Tree* dapat dibentuk dari dua *Steiner Tree* dengan *root* yang sama. Diberikan dua *Steiner Tree* yang dapat dilihat pada Gambar 2.5. Kedua *Steiner Tree* tersebut digabungkan sehingga dihasilkan *Steiner Tree* yang dapat dilihat pada Gambar 2.6. Selanjutnya, suatu *Steiner Tree* dapat dibentuk dari *Steiner Tree* lain yang *root*nya disambungkan ke *vertex* tetangga, kemudian *vertex* tetangga tersebut menjadi *root* dari *Steiner Tree* baru. Pembentukan *Steiner Tree* tersebut dapat dilihat pada Gambar 2.7.



Gambar 2.4 Contoh *Steiner Points* dan Hasil Pemampatan Koordinatnya

2.6 Pemodelan Relasi Rekuren

Sebuah *Steiner Tree* dapat dibentuk dari dua *Steiner Tree* yang memiliki *vertex root* yang sama.

$$ST_{i,j,K} = ST_{i,j,L} + ST_{i,j,K \cap \overline{L}}$$

$$K \subset S$$

$$L \subset K$$

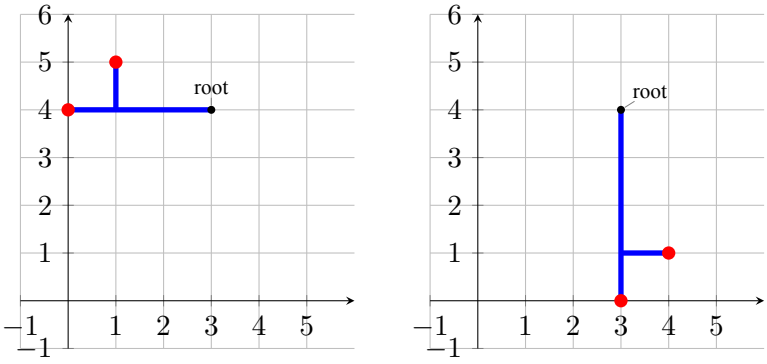
$$(i,j) \in P$$

Sebuah *Steiner Tree* juga dapat dibentuk dari *Steiner Tree* lain yang rootnya disambungkan ke *vertex* tetangga.

$$ST_{i,j,K} = ST_{m,n,K} + d((i,j),(m,n))$$

$$(m,n) \in T_{i,j}$$

$$(i,j),(m,n) \in P$$



Gambar 2.5 Contoh Kedua Steiner Tree yang Akan Digabungkan

Sehingga *Minimum Steiner Tree* dapat dibentuk dengan cara membandingkan seluruh konfigurasi pembentukan *Steiner Tree* dengan bobot minimal.

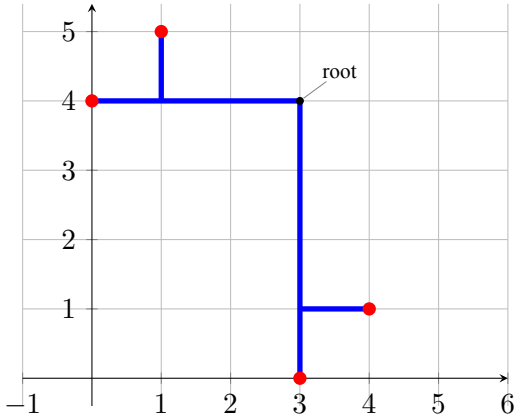
$$A = \underset{L \subset K}{min}(MST_{i,j,L} + MST_{i,j,K \cap \overline{L}})$$

$$B = \underset{(m,n) \in T_{i,j}}{min}(MST_{m,n,K} + d((i,j),(m,n))$$

$$MST_{i,j,K} = \begin{cases} d((i,j),K_0), & \text{if } |K| = 1. \\ min(A,B), & \text{otherwise.} \end{cases}$$

2.7 Strategi Pencarian Jarak Terpendek pada Steiner Tree

Pada subbab ini dijelaskan mengenai cara mencari panjang total terpendek untuk membangun suatu *Steiner Tree* menggunakan dua algoritma berbeda yaitu *Dijkstra* dan *Shortest Path Faster Algorithm* (SPFA).

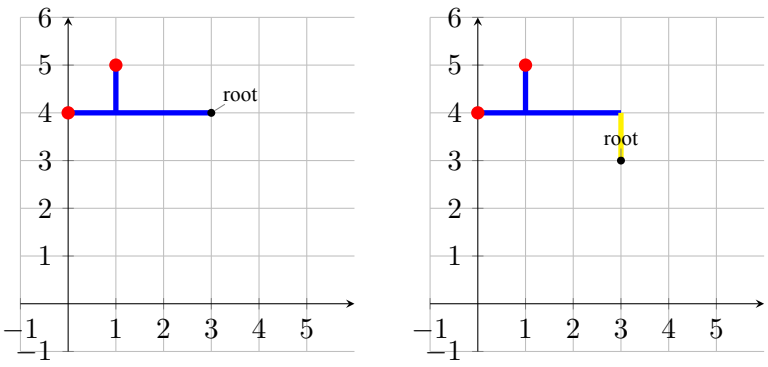


Gambar 2.6 Contoh Hasil Akhir Penggabungan Dua *Steiner Tree*

2.7.1 Algoritma Dijkstra

Algoritma Dijkstra menyelesaikan permasalahan *single-source shortest path* pada *weighted directed graph* $G(V, E)$ untuk kasus di mana semua bobot pada *edge* adalah sebuah bilangan non-negatif. Algoritma Dijkstra menyimpan sebuah *set* N berisi *vertex* dan perkiraan minimal bobotnya saat ini. Algoritma ini berulang kali memilih sebuah *vertex* $u \in V - N$ yang memiliki perkiraan bobot minimal. Kemudian dilakukan *relaxation* pada seluruh *edge* yang meninggalkan u. *Relaxation* adalah sebuah proses dimana untuk sebuah *vertex* u dan v, jika u dan v terkoneksi oleh suatu *edge* secara langsung dan jika perkiraan bobot dari *vertex* awal ke u saat ini ditambah bobot u ke v lebih kecil dari perkiraan bobot *vertex* awal ke v saat ini, maka perkiraan bobot *vertex* awal ke v diubah nilainya menjadi perkiraan bobot dari *vertex* awal ke u saat ini ditambah *weight* u ke v [2].

$$est(s,v) = \begin{cases} est(s,u) + w(u,v), & \text{if } est(s,u) + w(u,v) < est(s,v). \\ est(s,v), & \text{otherwise.} \end{cases}$$



Gambar 2.7 Contoh Steiner Tree Awal dan Steiner Tree Baru

Algoritma Dijkstra memiliki kompleksitas $O(V + ElogV)$. Cara kerja Dijkstra dapat dilihat pada *Pseudocode* 2.1. Pada permasalahan *Minimum Steiner Tree*, diperlukan pembentukan semua konfigurasi *initial steiner tree*, kemudian algoritma Dijkstra digunakan untuk mencari panjang minimal tiap konfigurasi *Steiner Tree*. Terdapat dua jenis *relaxation* yang dibutuhkan pada permasalahan ini, yaitu *Root Relaxation* dan *Neighbour Vertex Relaxation*. *Root Relaxation* adalah ketika *Steiner Tree* akan digabungkan dengan *Steiner Tree* lain yang memiliki root yang sama. *Relaxation* ini dituliskan dalam Persamaan 2.2.

$$nextWeight = MST_{i,j,L} + MST_{i,j,K \cap \overline{L}}$$

$$MST_{i,j,K} = \begin{cases} nextWeight, & \text{if } nextWeight < MST_{i,j,K}. \\ MST_{i,j,K}, & \text{otherwise.} \end{cases} \tag{2.2}$$

$$MST_{i,j,K} = \begin{cases} d((i,j),K_0), & \text{if } |K| = 1. \\ min(A,B), & \text{otherwise.} \end{cases}$$

***Pseudocode* 2.1** Pseudocode Algoritma Dijkstra

Input: $Graph, source$

```
1:  create vertex state Q
2:  for each vertex v in Graph do
3:      dist[v] ← INF
4:  end for
5:  dist[source] ← 0
6:  add source to Q
7:  while Q is not empty do
8:      u ← vertex in Q with min dist[u]
9:      remove u from Q
10:     for each neighbor v of u do
11:         alt ← dist[u] + length(u, v)
12:         if alt < dist[v] then
13:             dist[v] ← alt
14:             add v to Q
15:         end if
16:     end for
17: end while
18: return dist[]
```

Neighbour Vertex Relaxation adalah ketika sebuah *Steiner Tree* dibentuk dengan cara menyambungkan *root* suatu *Steiner Tree* dengan *vertex* tetangga v yang nantinya akan menjadi *root* baru. *Relaxation* ini dituliskan dalam Persamaan 2.3.

$$nextWeight = MST_{x,y,K} + d_{i,j,x,y}$$

$$MST_{i,j,K} = \begin{cases} nextWeight, & \text{if } nextWeight < MST_{i,j,K}. \\ MST_{i,j,K}, & \text{otherwise.} \end{cases} \tag{2.3}$$

Diberikan tiga terminal dengan koordinat (0,1), (2,0), dan (4,3). Maka jalannya algoritma Dijkstra dapat dilihat pada Tabel 3.6. Pada Tabel tersebut kolom berwarna gelap menandakan terjadinya *relaxation*. *Initial steiner tree* pada kasus uji ini dapat dilihat pada Lampiran D.

2.7.2 Shortest Path Faster Algorithm (SPFA)

Shortest Path Faster Algorithm (SPFA) memiliki properti yang mirip dengan Dijkstra. Algoritma ini juga melakukan *relaxation*. Tidak seperti Dijkstra, algoritma ini dapat menangani permasalahan dengan bobot negatif. Pada algoritma ini setiap *vertex* digunakan sebagai kandidat untuk melakukan *relaxation* pada semua *vertex* tetangganya. SPFA menyimpan kandidat *vertex* pada sebuah queue dan menambahkan *vertex* baru ke dalam queue jika *relaxation* berhasil dilakukan dan *vertex* tersebut belum ada di queue. SPFA berhenti ketika tidak ada lagi kandidat *vertex* [4]. Algoritma ini memiliki rata-rata kompleksitas $O(|E|)$ dan kompleksitas waktu *worst-case* $O(V \times E)$. *Pseudocode* dari SPFA dapat dilihat pada *Pseudocode* 2.2. Sama seperti Dijkstra, pada permasalahan *Minimum Steiner Tree* digunakan untuk mencari minimal panjang tiap konfigurasi *Steiner Tree*. *Relaxation* yang diperlukan dapat dilihat

Tabel 2.1 Langkah- langkah Dijkstra

(a) Bagian 1

No	State	Relaxation		
		State	Next Value	Now Value
1	MST[0,0,3] = 3	MST[0,0,7]	10	10
		MST[2,0,3]	5	3
		MST[0,1,3]	4	3
2	MST[0,1,3] = 3	MST[0,1,7]	9	9
		MST[0,0,3]	4	3
		MST[2,1,3]	5	3
		MST[0,3,3]	5	7
3	MST[2,0,3] = 3	MST[2,0,7]	8	8
		MST[0,0,3]	5	3
		MST[4,0,3]	5	7
		MST[2,1,3]	4	3
4	MST[2,1,3] = 3	MST[2,1,7]	7	7
		MST[0,1,3]	5	3
		MST[2,0,3]	4	3
		MST[4,1,3]	5	7
		MST[2,3,3]	5	7
5	MST[0,3,3] = 5	MST[0,3,7]	9	11
		MST[0,1,3]	7	3
		MST[2,3,3]	7	5
6	MST[2,0,6] = 5	MST[2,0,7]	8	8
		MST[0,0,6]	7	9
		MST[4,0,6]	7	5
		MST[2,1,6]	6	5
7	MST[2,1,6] = 5	MST[2,1,7]	7	7
		MST[0,1,6]	7	9
		MST[2,0,6]	6	5
		MST[4,1,6]	7	5
		MST[2,3,6]	7	5
8	MST[2,3,3] = 5	MST[2,3,7]	7	9
		MST[0,3,3]	7	5
		MST[2,1,3]	7	3
		MST[4,3,3]	7	11

(b) Bagian 2

No	State	Relaxation		
		State	Next Value	Now Value
9	MST[2,3,6] = 5	MST[2,3,7]	9	7
		MST[0,3,6]	7	9
		MST[2,1,6]	7	5
		MST[4,3,6]	7	5
10	MST[4,0,3] = 5	MST[4,0,7]	8	10
		MST[2,0,3]	7	3
		MST[4,1,3]	6	5
11	MST[4,0,6] = 5	MST[4,0,7]	10	8
		MST[2,0,6]	7	5
		MST[4,1,6]	6	5
12	MST[4,1,3] = 5	MST[4,1,7]	7	9
		MST[2,1,3]	7	3
		MST[4,0,3]	6	5
		MST[4,3,3]	7	7
13	MST[4,1,6] = 5	MST[4,1,7]	9	7
		MST[2,1,6]	7	5
		MST[4,0,6]	6	5
		MST[4,3,6]	7	5
14	MST[4,3,6] = 5	MST[4,3,7]	11	11
		MST[2,3,6]	7	5
		MST[4,1,6]	7	5
15	MST[0,1,5] = 6	MST[0,1,7]	9	9
		MST[0,0,5]	7	8
		MST[2,1,5]	8	6
		MST[0,3,5]	8	6
16	MST[0,3,5] = 6	MST[0,3,7]	11	9
		MST[0,1,5]	8	6
		MST[2,3,5]	8	6
17	MST[2,1,5] = 6	MST[2,1,7]	7	7
		MST[0,1,5]	8	6
		MST[2,0,5]	7	8
		MST[4,1,5]	8	6
		MST[2,3,5]	8	6

(c) Bagian 3

No	State	Relaxation		
		State	Next Value	Now Value
18	MST[2,3,5] = 6	MST[2,3,7]	9	7
		MST[0,3,5]	8	6
		MST[2,1,5]	8	6
		MST[4,3,5]	8	6
19	MST[4,1,5] = 6	MST[4,1,7]	9	7
		MST[2,1,5]	8	6
		MST[4,0,5]	7	8
		MST[4,3,5]	8	6
20	MST[4,3,5] = 6	MST[4,3,7]	11	11
		MST[2,3,5]	8	6
		MST[4,1,5]	8	6
21	MST[0,0,5] = 7	MST[0,0,7]	9	10
		MST[2,0,5]	9	7
		MST[0,1,5]	8	6
22	MST[0,0,6] = 7	MST[0,0,7]	8	9
		MST[2,0,6]	9	5
		MST[0,1,6]	8	7
23	MST[0,1,6] = 7	MST[0,1,7]	7	9
		MST[0,0,6]	8	7
		MST[2,1,6]	9	5
		MST[0,3,6]	9	7
24	MST[0,1,7] = 7	-	-	-

pada Rumus 2.2 dan 2.3. Diberikan tiga terminal dengan koordinat (0,1), (2,0), dan (4,3). Maka jalannya SPFA dapat dilihat pada Tabel 3.9. Pada Tabel tersebut kolom bercetak tebal menandakan terjadinya *relaxation*, sedangkan kolom berwarna gelap menandakan terjadinya *relaxation* dan penambahan *vertex* pada queue. *Initial steiner tree* pada kasus uji ini dapat dilihat pada Lampiran D.

***Pseudocode* 2.2** Pseudocode SPFA

Input: G, s

```
1:  for each vertex v ≠ in V(G) do
2:      d[v] ← INF
3:  end for
4:  d[s] = 0
5:  add s to Q
6:  while Q is not empty do
7:      u ← poll Q
8:      for each edge (u, v) in E(G) do
9:          if d[u] + w(u, v) < d[v] then
10:             d[v] ← d[u] + w(u, v)
11:             if v is not in Q then
12:                 add v to Q
13:             end if
14:         end if
15:     end for
16: end while
```

Tabel 2.2 Langkah-langkah SPFA

(a) Bagian 1

No	State	Relaxation		
		State	Next Value	Now Value
1	MST[0,0,3]	MST[0,0,7]	10	10
		MST[2,0,3]	5	3
		MST[0,1,3]	4	3
2	MST[0,0,5]	MST[0,0,7]	10	10
		MST[2,0,5]	10	8
		MST[0,1,5]	9	6
3	MST[0,0,6]	MST[0,0,7]	10	10
		MST[2,0,6]	11	5
		MST[0,1,6]	10	9
4	MST[0,0,7]	MST[2,0,7]	12	8
		MST[0,1,7]	11	9
5	MST[0,1,3]	MST[0,1,7]	9	9
		MST[0,0,3]	4	3
		MST[2,1,3]	5	3
		MST[0,3,3]	**5**	**7**
6	MST[0,1,5]	MST[0,1,7]	9	9
		MST[0,0,5]	7	8
		MST[2,1,5]	8	6
		MST[0,3,5]	8	6
7	MST[0,1,6]	MST[0,1,7]	9	9
		MST[0,0,6]	10	9
		MST[2,1,6]	11	5
		MST[0,3,6]	11	9
8	MST[0,1,7]	MST[0,0,7]	10	10
		MST[2,1,7]	11	7
		MST[0,3,7]	11	11
9	MST[0,3,3]	**MST[0,3,7]**	**9**	**11**
		MST[0,1,3]	7	3
		MST[2,3,3]	7	7
10	MST[0,3,5]	MST[0,3,7]	11	9
		MST[0,1,5]	8	6
		MST[2,3,5]	8	6

(b) Bagian 2

No	State	Relaxation		
		State	Next Value	Now Value
11	MST[0,3,6]	MST[0,3,7]	11	9
		MST[0,1,6]	11	9
		MST[2,3,6]	11	5
12	MST[0,3,7]	MST[0,1,7]	11	9
		MST[2,3,7]	11	9
13	MST[2,0,3]	MST[2,0,7]	8	8
		MST[0,0,3]	5	3
		MST[4,0,3]	**5**	**7**
		MST[2,1,3]	4	3
14	MST[2,0,5]	MST[2,0,7]	8	8
		MST[0,0,5]	10	7
		MST[4,0,5]	10	8
		MST[2,1,5]	9	6
15	MST[2,0,6]	MST[2,0,7]	8	8
		MST[0,0,6]	7	9
		MST[4,0,6]	7	5
		MST[2,1,6]	6	5
16	MST[2,0,7]	MST[0,0,7]	10	10
		MST[4,0,7]	10	10
		MST[2,1,7]	9	7
17	MST[2,1,3]	MST[2,1,7]	7	7
		MST[0,1,3]	5	3
		MST[2,0,3]	4	3
		MST[4,1,3]	**5**	**7**
		MST[2,3,3]	**5**	**7**
18	MST[2,1,5]	MST[2,1,7]	7	7
		MST[0,1,5]	8	6
		MST[2,0,5]	7	8
		MST[4,1,5]	8	6
		MST[2,3,5]	8	6

(c) Bagian 3

No	State	Relaxation		
		State	Next Value	Now Value
19	MST[2,1,6]	MST[2,1,7]	7	7
		MST[0,1,6]	7	9
		MST[2,0,6]	6	5
		MST[4,1,6]	7	5
		MST[2,3,6]	7	5
20	MST[2,1,7]	MST[0,1,7]	9	9
		MST[2,0,7]	8	8
		MST[4,1,7]	9	9
		MST[2,3,7]	9	9
21	MST[2,3,3]	**MST[2,3,7]**	**7**	**9**
		MST[0,3,3]	7	5
		MST[2,1,3]	7	3
		MST[4,3,3]	**7**	**11**
22	MST[2,3,5]	MST[2,3,7]	9	7
		MST[0,3,5]	8	6
		MST[2,1,5]	8	6
		MST[4,3,5]	8	6
23	MST[2,3,6]	MST[2,3,7]	9	7
		MST[0,3,6]	7	9
		MST[2,1,6]	7	5
		MST[4,3,6]	7	5
24	MST[2,3,7]	MST[0,3,7]	9	9
		MST[2,1,7]	9	7
		MST[4,3,7]	**9**	**11**
25	MST[4,0,3]	**MST[4,0,7]**	**8**	**10**
		MST[2,0,3]	7	3
		MST[4,1,3]	6	5
26	MST[4,0,5]	MST[4,0,7]	10	8
		MST[2,0,5]	10	7
		MST[4,1,5]	9	6

(d) Bagian 4

No	State	Relaxation		
		State	Next Value	Now Value
27	MST[4,0,6]	MST[4,0,7]	10	8
		MST[2,0,6]	7	5
		MST[4,1,6]	6	5
28	MST[4,0,7]	MST[2,0,7]	10	8
		MST[4,1,7]	9	9
29	MST[4,1,3]	**MST[4,1,7]**	**7**	**9**
		MST[2,1,3]	7	3
		MST[4,0,3]	6	5
		MST[4,3,3]	7	7
30	MST[4,1,5]	MST[4,1,7]	9	7
		MST[2,1,5]	8	6
		MST[4,0,5]	7	8
		MST[4,3,5]	8	6
31	MST[4,1,6]	MST[4,1,7]	9	7
		MST[2,1,6]	7	5
		MST[4,0,6]	6	5
		MST[4,3,6]	7	5
32	MST[4,1,7]	MST[2,1,7]	9	7
		MST[4,0,7]	8	8
		MST[4,3,7]	9	9
33	MST[4,3,3]	**MST[4,3,7]**	**7**	**9**
		MST[2,3,3]	9	5
		MST[4,1,3]	9	5
34	MST[4,3,5]	MST[4,3,7]	11	7
		MST[2,3,5]	8	6
		MST[4,1,5]	8	6
35	MST[4,3,6]	MST[4,3,7]	11	7
		MST[2,3,6]	7	5
		MST[4,1,6]	7	5
36	MST[4,3,7]	MST[2,3,7]	9	7
		MST[4,1,7]	9	7

(e) Bagian 5

No	State	Relaxation		
		State	Next Value	Now Value
37	MST[0,0,5]	MST[0,0,7]	9	10
		MST[2,0,5]	9	7
		MST[0,1,5]	8	6
38	MST[0,0,6]	**MST[0,0,7]**	**8**	**9**
		MST[2,0,6]	9	5
		MST[0,1,6]	8	7
39	MST[2,0,5]	MST[2,0,7]	7	8
		MST[0,0,5]	9	7
		MST[4,0,5]	9	7
		MST[2,1,5]	8	6
40	MST[0,1,6]	MST[0,1,7]	7	9
		MST[0,0,6]	8	7
		MST[2,1,6]	9	5
		MST[0,3,6]	9	7
41	MST[0,3,6]	MST[0,3,7]	9	9
		MST[0,1,6]	9	7
		MST[2,3,6]	9	5
42	MST[4,0,5]	MST[4,0,7]	9	8
		MST[2,0,5]	9	7
		MST[4,1,5]	8	6
43	MST[0,0,7]	MST[2,0,7]	10	7
		MST[0,1,7]	9	7
44	MST[2,0,7]	MST[0,0,7]	9	8
		MST[4,0,7]	9	8
		MST[2,1,7]	8	7
45	MST[0,1,7]	MST[0,0,7]	8	8
		MST[2,1,7]	9	7
		MST[0,3,7]	9	9

BAB III
DESAIN

Pada bab ini akan dijelaskan desain algoritma yang akan digunakan untuk menyelesaikan Tugas Akhir.

3.1 Deskripsi Umum Sistem

Pada subbab ini akan dijelaskan mengenai gambaran secara umum dari algoritma yang dirancang. Sistem akan diawali dengan menerima masukan berupa nilai N yang menyatakan banyaknya *Steiner Points*. N baris selanjutnya berisi nilai x dan y yang berupa titik koordinat *Steiner Points* pada bidang kartesius. Nilai masukan ini mengikuti batasan pada Subbab 2.1. Setelah menerima masukan, maka masukan tersebut akan diolah untuk menghitung *Minimum Steiner Tree* dari N *Steiner Points* yang telah dimasukkan sebelumnya. Sistem menggunakan tiga tahapan metode pada pencarian *Minimum Steiner Tree* yaitu pemampatan koordinat, pembangunan *Initial Steiner Tree*, kemudian pencarian *Shortest Path*. Ketiga metode ini akan dijalankan secara berurutan untuk mendapatkan nilai dari *Minimum Steiner Tree*. Pada tahap pencarian *Shortest Path* sistem menggunakan dua algoritma yang berbeda yaitu *Dijkstra* dan *Shortest Path Faster Algorithm*. Masing-masing metode akan digunakan untuk dibandingkan kinerjanya. Variabel yang digunakan sistem dapat dilihat pada Tabel 3.1. *Pseudocode* fungsi MAIN ditunjukkan pada *Pseudocode* 3.1. Fungsi MAIN merupakan bagian fungsi utama yang menerima masukan dan memproses masukan tersebut dengan memanggil fungsi SOLVE. Fungsi SOLVE inilah yang nantinya akan menjalankan metode-metode yang dibutuhkan untuk mennyelesaikan permasalahan *E-Olymp 1445 - Road Network*. Fungsi INPUT merupakan fungsi untuk menerima masukan, dan fungsi PRINT merupakan fungsi untuk menampilkan hasil.

Tabel 3.1 Daftar variabel yang digunakan sistem

Nama Variabel	Tipe Data	Keterangan
N	*integer*	Menyimpan banyaknya *Steiner Points*.
X	*integer*	Menyimpan nilai absis dari *Steiner Points*.
Y	*integer*	Menyimpan nilai ordinat dari *Steiner Points*.
$xIdx$	*integer*	Menyimpan banyaknya nilai absis dari *Steiner Points* yang *unique*.
$yIdx$	*integer*	Menyimpan banyaknya nilai ordinat dari *Steiner Points* yang *unique*.
$compX$	*integer*	Menyimpan nilai absis dari koordinat *Steiner Points* yang *unique* dan terurut.
$compY$	*integer*	Menyimpan nilai ordinat dari kkoordinat *Steiner Points* yang *unique* dan terurut.
MST	*integer*	Menyimpan nilai dari *Minimum Steiner Tree*
$visited$	*boolean*	Penanda apakah suatu *state Steiner Tree* sudah berada di dalam *queue*

***Pseudocode* 3.1** Fungsi MAIN

```
1: N ← INPUT()
2: X, Y ← array[1..N]integer
3: for i ← 1, N do
4:     X[i], Y[i] ← INPUT()
5: end for
6: ans ← SOLVE(N, X, Y)
7: PRINT(ANS)
```

3.2 Desain Penyelesaian Pemampatan Koordinat

Fungsi Pemampatan Koordinat adalah fungsi yang melakukan pemampatan pada koordinat di bidang kartesius sehingga hanya tersisa koordinat yang memungkinkan untuk dilewati *Minimum Steiner Tree*. Cara kerja fungsi ini mengacu pada penjelasan subbab 2.4. Masukan, proses dan keluaran dari fungsi ini tercantum pada Tabel 3.2. *Pseudocode* fungsi ini dapat dilihat pada *Pseudocode* 3.2.

***Pseudocode* 3.2** Fungsi PEMAMPATANKOORDINAT

```
1:  xIdx, yIdx ← 2
2:  compX ← array [1..size] integer
3:  compY ← array [1..size] integer
4:  SORT(X)
5:  SORT(Y)
6:  compX[1] ← X[1]
7:  compY[1] ← Y[1]
8:  for i ← 2 to N do
9:      if X[i]! = compX[xIdx − 1] then
10:         compX[xIdx + 1] ← X[i]
11:         xIdx ← xIdx + 1
12:     end if
13:     if Y[i]! = compY[yIdx − 1] then
14:         compY[yIdx + 1] ← Y[i]
15:         yIdx ← yIdx + 1
16:     end if
17: end for
```

3.3 Desain Penyelesaian Pembangunan Initial Steiner Tree

Subbab ini akan menjelaskan desain fungsi yang akan digunakan untuk menyelesaikan pembangunan *Initial Steiner Tree*.

Tabel 3.2 Masukan, proses, dan keluaran dari fungsi PEMAMPATAN KOORDINAT

Masukan	Proses	Keluaran
-	Melakukan pemampatan koordinat pada bidang kartesius berdasarkan *Steiner Points*.	-

Tabel 3.3 Masukan, proses, dan keluaran dari fungsi INITIAL STEINER TREE

Masukan	Proses	Keluaran
-	Melakukan pembangunan *initial steiner tree*	-

3.3.1 Desain Fungsi Pembangunan Initial Steiner Tree

Fungsi Pembangunan *Initial Steiner Tree* adalah fungsi yang membentuk semua *Steiner Tree* tanpa mempertimbangkan jarak minimalnya. *Steiner Points* yang telah dilewati direpresentasikan dalam bentuk bit. Jika bit ke x menyala, maka *Steiner Tree* ke x sudah terlewati. Masukan, proses dan keluaran dari fungsi ini tercantum pada Tabel 3.3. *Pseudocode* fungsi ini dapat dilihat pada *Pseudocode* 3.3

3.3.2 Desain Fungsi Jarak Manhattan

Fungsi Jarak Manhattan adalah fungsi yang menghitung jarak *manhattan* dari dua titik pada bidang kartesius. Fungsi ini menjumlahkan selisih nilai absis dan selisih nilai ordinat dari dua titik. Masukan, proses dan keluaran dari fungsi ini tercantum pada tabel

***Pseudocode* 3.3** Fungsi INITIALSTEINERTREE

```
1: MST ← array[1..xIdx][1..yIdx][1..(1 ≪ N)] integer
2: visited ← array[1..xIdx][1..yIdx][1..(1 ≪ N)] boolean
3: SET(MST, 0)
4: q ← queue of STEINERTREE
5: for (i, j) in xIdx, yIdx do
6:     for k ← 1 to 1 ≪ N − 1 do
7:         for l ← 1 to N do
8:             if k&(1 ≪ l) then
9:                 MST[i, j, k] ← MST[i, j, k]+
   MANHATTAN(X[l], Y[l], compX[i], compY[j])
10:            end if
11:        end for
12:        population ← CountPopulation(k)
13:        if population > 1 then
14:            st ← STEINERTREE(i, j, k, MST[i, j, k])
15:            PUSH(q, st)
16:            visited[i, j, k] ← true
17:        end if
18:    end for
19: end for
```

3.5. *Pseudocode* fungsi ini dapat dilihat pada *Pseudocode* 3.4.

***Pseudocode* 3.4** Fungsi MANHATTAN

Input: x_1, y_1, x_2, y_2
1: $xDiff \leftarrow |x_1 - x_2|$
2: $yDiff \leftarrow |y_1 - y_2|$
3: $ans \leftarrow xDiff + yDiff$
4: **return** ans

3.3.3 Desain Fungsi Count Population

Fungsi COUNTPOPULATION adalah fungsi yang menghitung berapa banyak bit yang menyala di suatu bilangan. Masukan, proses dan keluaran dari fungsi ini tercantum pada Tabel 3.4. *Pseudocode* fungsi ini dapat dilihat pada *Pseudocode* 3.5.

***Pseudocode* 3.5** Fungsi COUNTPOPULATION

Input: n
1: $ans \leftarrow 0$
2: **while** $n > 0$ **do**
3: **if** n mod 2 **then** $ans \leftarrow ans + 1$
4: $n \leftarrow n/2$
5: **end if**
6: **end while**
7: **return** ans

3.4 Desain Fungsi Dijkstra

Fungsi DIJKSTRA adalah fungsi yang menghitung jarak terpendek dari sebuah *Steiner Tree*. Cara kerja fungsi ini mengacu pada penjelasan Subbab 2.7.1. Masukan, proses dan keluaran dari fungsi ini tercantum pada Tabel 3.6. *Pseudocode* fungsi ini dapat dilihat pada *Pseudocode* 3.6.

Tabel 3.4 Masukan, proses, dan keluaran dari fungsi COUNTPOPULATION

Masukan	Proses	Keluaran
Sebuah bilangan bulat n	Menghitung bit yang menyala pada n	Banyaknya bit yang menyala pada n

Tabel 3.5 Masukan, proses, dan keluaran dari fungsi MANHATTAN DISTANCE

Masukan	Proses	Keluaran
Empat bilangan bulat yaitu x_1, y_1, x_2, y_2 yang menyatakan nilai absis dan ordinat dari titik pertama dan nilai abisi dan ordinat dari titik kedua	Melakukan perhitungan jarak *manhattan* dari dua titik	Jarak *manhattan* dari dua titik

***Pseudocode* 3.6** Fungsi DIJKSTRA

```
1:  while q is not empty do
2:      top ← min STEINERTREE in q
3:      DELETE(q, top)
4:      x, y, bit, val ← top
5:      if bit = 1 ≪ N − 1 then return val
6:      end if
7:      if val ≠ MST[x, y, bit] then continue
8:      end if
9:      RELAXATION(MST, top)
10: end while
```

Tabel 3.6 Masukan, proses, dan keluaran dari fungsi DIJKSTRA

Masukan	Proses	Keluaran
-	Melakukan perhitungan *Minimum Steiner Tree*	Nilai *Minimum Steiner Tree* dari N *Steiner Points*

Masukan	Proses	Keluaran
Empat buah bilangan bulat x, y, bit, val yang menyatakan nilai absis, ordinat, bit penanda *Steiner Tree* yang telah terlewati, dan nilai yang akan dibandingkan pada *Relaxation*.	Melakukan *relaxation* pada *Minimum Steiner Tree*	-

Tabel 3.7 Masukan, Proses, dan Keluaran dari Fungsi RELAXATION

3.4.1 Desain Fungsi Relaxation

Fungsi RELAXATION adalah fungsi yang melakukan semua *relaxation* yang dibutuhkan untuk mendapatkan *Minimum Steiner Tree*. Masukan, proses dan keluaran dari fungsi ini tercantum pada tabel 3.7. *Pseudocode* fungsi ini dapat dilihat pada *Pseudocode* 3.7.

3.4.2 Desain Fungsi Relax

Fungsi RELAX adalah fungsi yang melakukan pembaruan data ketika metode *Dijkstra* menemukan jarak yang lebih pendek dibandingkan dengan jarak yang diketahui saat ini. Masukan, proses

***Pseudocode* 3.7** Fungsi RELAXATION

Input: x, y, bit, val

1: $i \leftarrow bit$
2: **while** $i < 1 \ll N$ **do**
3: $\quad nextVal \leftarrow val + MST[x, y, i \oplus bit])$
4: $\quad$ RELAX$(x, y, i, MST[x, y, i]), nextVal)$
5: $\quad i \leftarrow (i + 1)|bit$
6: **end while**
7: **if** $x > 1$ **then**
8: $\quad nextVal \leftarrow val + X[x] - X[x - 1]$
9: $\quad$ RELAX$(x - 1, y, bit, MST[x - 1, y, bit], nextVal)$
10: **end if**
11: **if** $y > 1$ **then**
12: $\quad nextVal \leftarrow val + Y[y] - Y[y - 1]$
13: $\quad$ RELAX$(x, y - 1, bit, MST[x, y - 1, bit], nextVal)$
14: **end if**
15: **if** $x < xIdx$ **then**
16: $\quad nextVal \leftarrow val + X[x + 1] - X[x]$
17: $\quad$ RELAX$(x + 1, y, bit, MST[x + 1, y, bit], nextVal)$
18: **end if**
19: **if** $y < yIdx$ **then**
20: $\quad nextVal \leftarrow val + Y[y + 1] - Y[y]$
21: $\quad$ RELAX$(x, y + 1, bit, MST[x, y + 1, bit], nextVal)$
22: **end if**

dan keluaran dari fungsi ini tercantum pada Tabel 3.8. *Pseudocode* fungsi ini dapat dilihat pada *Pseudocode* 3.8.

***Pseudocode* 3.8** Fungsi RELAX

Input: $x, y, bit, val, nextVal$

```
1: if val > nextVal then
2:     MST[x, y, bit] ← nextVal
3:     st ← STEINERTREE(x, y, bit, nextVal)
4:     PUSH(q, st)
5: end if
```

3.4.3 Desain Fungsi Solve

Pada subbab ini akan dijelaskan mengenai fungsi SOLVE untuk menyelesaikan permasalahan *Minimum Steiner Tree*. *Pseudocode* dari fungsi ini dapat dilihat pada *Pseudocode* 3.9.

***Pseudocode* 3.9** Fungsi SOLVE

Input: N, X, Y

```
1: xIdx ← 0
2: yIdx ← 0
3: compX, compY ← KOMPRESIKOORDINAT(N, X, Y)
4: MST ← INITIALSTEINERTREE(N, X, Y, compX, compY)
5: ans ← DIJKSTRA(N, MST)
6: return ans
```

3.5 Desain Fungsi Shortest Path Faster Algorithm (SPFA)

Fungsi SPFA adalah fungsi yang menghitung jarak terpendek dari sebuah *Steiner Tree*. Fungsi ini dapat dilihat pada *Pseudocode* 3.10. Fungsi RELAXATION yang dipanggil oleh fungsi ini dapat dilihat pada *Pseudocode* 3.7. Fungsi COUNTPOPULATION yang dipanggil oleh fungsi ini dapat dilihat pada *Pseudocode* 3.5. Cara

Tabel 3.8 Masukan, proses, dan keluaran dari fungsi RELAX pada DIJKSTRA

Masukan	Proses	Keluaran
Lima bilangan bulat $x, y, bit, val,$ dan $nextVal$ yang menyatakan nilai absis, ordinat, representasi *Steiner Tree* yang telah terlewati dalam bentuk bilangan bulat, nilai yang akan dibandingkan dan nilai yang menjadi pembanding.	Melakukan relaksasi	-

Tabel 3.9 Masukan, proses, dan keluaran dari fungsi SPFA

Masukan	Proses	Keluaran
-	Melakukan perhitungan *Minimum Steiner Tree*	Nilai *Minimum Steiner Tree* dari N *Steiner Tree*

kerja fungsi ini mengacu pada penjelasan subbab 2.7.2. Masukan, proses dan keluaran dari fungsi ini tercantum pada tabel 3.9.

***Pseudocode* 3.10** Fungsi SPFA

```
1:  while q is not empty do
2:      top ← q.top
3:      DELETE(q, top)
4:      x, y, bit, val ← top
5:      visited[x, y, bit] = false
6:      if bit = 1 ≪ N − 1 then return val
7:      end if
8:      if val ≠ MST[x, y, bit] then continue
9:      end if
10:     RELAXATION(MST, top)
11: end while
```

3.5.1 Desain Fungsi Relax

Fungsi RELAX adalah fungsi yang melakukan pembaruan data ketika metode *Shortest Path Faster Algorithm* menemukan jarak yang lebih pendek dibandingkan dengan jarak yang diketahui saat ini. Masukan, proses dan keluaran dari fungsi ini tercantum pada tabel 3.10. *Pseudocode* fungsi ini dapat dilihat pada *Pseudocode* 3.11.

***Pseudocode* 3.11** Fungsi RELAX pada *Shortest Path Faster Algorithm*

Input: $x, y, bit, val, nextVal$

```
1: if val > nextVal then
2:     MST[x, y, bit] ← nextVal
3:     if !visited[x, y, bit] then
4:         st ← STEINERTREE(x, y, bit, nextVal)
5:         PUSH(q, st)
6:         visited[x, y, bit] ← true
7:     end if
8: end if
```

Tabel 3.10 Masukan, proses, dan keluaran dari fungsi RELAX pada SPFA

Masukan	Proses	Keluaran
Lima bilangan bulat $x, y, bit, val,$ dan $nextVal$ yang menyatakan nilai absis, ordinat, bit penanda *Steiner Tree* yang telah terlewati, nilai yang akan dibandingkan dan nilai yang menjadi pembanding.	Melakukan pembaruan data	-

3.5.2 Desain Fungsi Solve

Pada subbab ini akan dijelaskan mengenai fungsi SOLVE untuk menyelesaikan permasalahan. *Pseudocode* dari fungsi ini dapat dilihat pada *Pseudocode* 3.12.

***Pseudocode* 3.12** Fungsi SOLVE

Input: N, X, Y

1: $xIdx \leftarrow 0$
2: $yIdx \leftarrow 0$
3: $compX, compY \leftarrow$ KOMPRESIKOORDINAT(N, X, Y)
4: $MST \leftarrow$ INITIALSTEINERTREE$(N, X, Y, compX, compY)$
5: $ans \leftarrow$ SPFA(N, MST)
6: **return** ans

BAB IV
IMPLEMENTASI

4.1 Lingkungan implementasi

Lingkungan implementasi dan pengembangan yang dilakukan adalah sebagai berikut.

1. Perangkat Keras
   (a) Processor Intel® Core™ i7-7500 CPU @ 2.7GHz (4 CPUs), 3.0GHz
   (b) Random Access Memory 8192MB
2. Perangkat Lunak
   (a) Sistem Operasi Linux Mint 18.3 Cinnamon 64-bit
   (b) Visual Studio Code
   (c) Bahasa Pemrograman C++
   (d) Kompiler GCC 7.4.0 (Ubuntu 7.4.0-1ubuntu1 18.04.1)

4.2 Implementasi Program Utama

Subbab ini menjelaskan implementasi proses algoritma secara keseluruhan berdasarkan desain yang telah dijelaskan pada Bab 3. Program ini merupakan program yang digunakan untuk menyelesaikan permasalahan E-Olymp 1445 - Road Network. Program juga akan diuji dengan data uji untuk membuktikan kebenaran algoritma.

4.2.1 Header yang diperlukan

Header digunakan untuk memanggil *library* C++ berisi fungsi-fungsi yang dibutuhkan. Implementasi program membutuhkan delapan buah *header* yaitu `iostream`, `queue`, `algorithm`, `string.h`, `stdlib.h`, `vector`, `iomanip`, dan

`limits.h`. Dapat dilihat pada Kode Sumber 4.1.

```
1 #include<iostream>
2 #include<queue>
3 #include<algorithm>
4 #include<string.h>
5 #include<stdlib.h>
6 #include<vector>
7 #include<iomanip>
8 #include<limits.h>
```

Kode Sumber 4.1 *Header* yang diperlukan

Header `iostream` berisi fungsi standar *input output* yang digunakan oleh bahasa C++. *Header* `queue` berisi struktur data yang digunakan untuk menyimpan urutan pemrosesan Dijkstra dan SPFA. *Header* `algorithm` berisi modul yang memiliki fungsi-fungsi yang sangat berguna dalam membantu mengimplementasi algoritma yang telah dibangun, contohnya adalah fungsi `sort`. *Header* `string.h` berisi fungsi-fungsi yang berhubungan dengan *string processing*. *Header* `stdlib.h` berisi fungsi-fungsi untuk manajemen memori dinamis, aritmatika, dll, contohnya adalah fungsi `abs`. *Header* `vector` berisi struktur data yang digunakan untuk menyimpan hasil pemampatan koordinat. *Header* `iomanip` berisi fungsi-fungsi untuk memanipulasi *input* dan *output* contohnya adalah fungsi `setprecision`. *Header* `limits.h` berisi variabel-variabel konstan yang mendefinisikan suatu batasan dari tipe data, contohnya adalah `INT_MAX`.

4.2.2 Preprocessor

Pre-processor seperti `using namespace std` merupakan perintah yang dapat digunakan untuk menyajikan perintah atau deklarasi kepada sebuah aplikasi *compiler* yang menyatakan bahwa program akan menggunakan seluruh berkas, *class*, atau fungsi yang menjadi bagian dari sebuah `namespace std` yang bersangkutan. Pre-processor dapat dilihat pada Kode Sumber 4.2.

```
1   using namespace std;
```

Kode Sumber 4.2 *Preprocessor* yang diperlukan

4.2.3 Variabel Global

Variabel global digunakan untuk memudahkan dalam mengakses data yang digunakan lintas fungsi/struct. Kode sumber implementasi variabel global dapat dilihat pada Kode Sumber 4.3. Variabel `n` sebagai banyaknya titik masukan atau *Steiner Points*, variabel `MST` sebagai *array* untuk menyimpan bobot tiap tiap konfigurasi *Steiner Tree*, variabel `X` dan `Y` sebagai *array* untuk menyimpan nilai absis dan ordinat dari tiap *Steiner Points*, variabel `visited` digunakan pada SPFA sebagai *array* yang menyimpan apakah suatu state *Steiner Tree* berada di dalam queue atau tidak, variabel `compX` dan `compY` sebagai *array* yang menyimpan nilai absis dan ordinat dari hasil pemampatan koordinat, variabel `q` adalah struktur data *queue* yang digunakan pada SPFA, variabel `pq` adalah struktur data *priority queue* yang digunakan pada algoritma Dijkstra.

```
1 int n;
2 int MST[11][11][1<<11];
3 int X[15], Y[15];
4 bool visited[11][11][1<<11];
5 vector<int> compX, compY;
6 queue<state> q;
7 priority_queue<state> pq;
```

Kode Sumber 4.3 Variabel global yang didefinisikan

4.2.4 Implementasi Fungsi Main

Fungsi main adalah implementasi algoritma yang dirancang pada Pseudocode 3.1. Implementasi fungsi main dapat dilihat pada Kode Sumber 4.4.

```
1 int main(){
2     int n, X[15], Y[15];
3     cin>>n;
4     for(int i = 0; i<n; i++){
5         cin >> X[i] >> Y[i];
6     }
7     double ans=solve(n, X, Y);
8     cout << setprecision(3) << fixed << ans
          <<endl;
9 }
```

Kode Sumber 4.4 Fungsi main

4.2.5 Implementasi Struct State Minimum Steiner Tree

Pada subbab ini akan dijelaskan mengenai implementasi dari struct `state`. Terdapat empat atribut yaitu x, y, bit, dan val. x dan y menyatakan nilai absis dan ordinat *root Steiner Tree*, bit menyatakan *Steiner Points* dalam representasi bit, val menyatakan bobot dari *Steiner Tree*. Pada struct terdapat juga *overloading operation* dari operasi kurang dari $(<)$ yang dibutuhkan oleh *priority queue*. Implementasi dari struct `state` dapat dilihat pada Kode Sumber 4.5.

```
1  struct state {
2      int x, y, bit, val;
3      bool operator < (const state s) const {
4            if (val>s.val) return true;
5          if (val<s.val) return false;
6          if (x>s.x) return true;
7          if (x<s.x) return false;
8          if (y>s.y) return true;
9          if (y<s.y) return false;
10         if (bit>s.bit) return true;
11         return false;
12     }
13 };
```

Kode Sumber 4.5 Struct State

4.2.6 Implementasi Fungsi Pemampatan Koordinat

Pada subbab ini akan dijelaskan mengenai implementasi pada subbab 2.4 dan Pseudocode 3.2. Implementasi dari fungsi pemampatan koordinat dapat dilihat pada Kode Sumber 4.6.

```
1  void pemampatanKoordinat(){
2      int listX[15], listY[15];
3
4      for(int i=0; i<n; i++){
5          listX[i]=X[i];
6          listY[i]=Y[i];
7      }
8
9      sort(listX,listX+n);
10     sort(listY,listY+n);
11
12     int xIdx, yIdx;
13
14     xIdx = 1;
15     yIdx = 1;
16
17     compX.push_back(listX[0]);
18     compY.push_back(listY[0]);
19
20     for(int i=1; i<n; i++){
21         if(listX[i]!=compX[xIdx-1]){
22             compX.push_back(listX[i]);
23             xIdx++;
24         }
25
26         if(listY[i]!=compY[yIdx-1]){
27             compY.push_back(listY[i]);
28             yIdx++;
29         }
30     }
31 }
```

Kode Sumber 4.6 Fungsi Pemampatan Koordinat

4.2.7 Implementasi Fungsi Pembangunan Initial Steiner Tree

Pada subbab ini akan dijelaskan mengenai implementasi pada Pseudocode 3.3. Implementasi fungsi pembangunan *Initial Steiner Tree* dapat dilihat pada Kode Sumber 4.7.

```
 1 void initialSteinerTree(){
 2     int xIdx, yIdx;
 3     xIdx = compX.size();
 4     yIdx = compY.size();
 5
 6     for(int i=0; i<xIdx; i++){
 7         for(int j=0; j<yIdx; j++){
 8             for(int k=1; k<(1<<n); k++){
 9                 for(int l=0; l<n; l++){
10                     if(k&(1<<l))
11                         MST[i][j][k] +=
12                             manhattanDistance(
13                                 compX[i], X[l],
14                                 compY[j], Y[l]
15                             );
16                 }
17                 if(countPopulation(k)>1){
18                     state temp;
19                     temp.x=i;
20                     temp.y=j;
21                     temp.bit=k;
22                     temp.val=MST[i][j][k];
23                     q.push(temp);
24                     pq.push(temp);
25                     visited[i][j][k]=1;
26                 }
27             }
28         }
29     }
30 }
```

Kode Sumber 4.7 Fungsi Pembangunan Initial Steiner Tree

4.2.8 Implementasi Fungsi Manhattan Distance

Pada subbab ini akan dijelaskan mengenai implementasi fungsi *Manhattan Distance* dari Pseudocode 3.4. Implementasi dari fungsi `Manhattan Distance` dapat dilihat pada Kode Sumber 4.8

```
1 int manhattanDistance(int x1, int y1, int x2, int
      y2){
2     return abs(x1-x2) + abs(y1-y2);
3 }
```

Kode Sumber 4.8 Fungsi Manhattan Distance

4.2.8.1 Implementasi Fungsi Count Population

Pada subbab ini akan dijelaskan mengenai implementasi fungsi Count Population dari Pseudocode 3.5. Fungsi ini digunakan untuk menghitung banyaknya bit yang menyala dari suatu bilangan bulat. Implementasi dari fungsi `CountPopulation` dapat dilihat pada Kode Sumber 4.9.

```
 1 int countPopulation(int k){
 2     int ans = 0;
 3
 4     while(k>0){
 5         if(k % 2) ans++;
 6         k/=2;
 7     }
 8
 9     return ans;
10 }
```

Kode Sumber 4.9 Fungsi Count Population

4.2.9 Implementasi Fungsi yang Digunakan Algoritma Dijkstra

Pada subbab ini akan dijelaskan mengenai implementasi dari fungsi-fungsi yang digunakan pada metode Dijkstra. Desain untuk

fungsi-fungsi ini dapat dilihat pada Subbab 3.4.

4.2.9.1 Implementasi Fungsi Dijkstra

Pada subbab ini akan dijelaskan mengenai implementasi fungsi Dijkstra yang dijelaskan pada Subbab 2.7.1 dan Pseudocode 3.6. Implementasi dari fungsi `Dijkstra` dapat dilihat pada Kode Sumber 4.10

```
1  int dijkstra(){
2      int xIdx, yIdx;
3
4      xIdx = compX.size();
5      yIdx = compY.size();
6
7      while(!pq.empty()){
8          state top = pq.top();
9          pq.pop();
10
11         int x = top.x;
12         int y = top.y;
13         int bit = top.bit;
14         int val = top.val;
15
16         if(val!=MST[x][y][bit]) continue;
17
18         if(bit==(1<<n)-1) return val;
19
20         relaxation(top);
21     }
22 }
```

Kode Sumber 4.10 Fungsi Dijkstra

4.2.9.2 Implementasi Fungsi Relax pada Dijkstra

Pada subbab ini akan dijelaskan mengenai implementasi fungsi `Relax` dari Pseudocode 3.8. Implementasi dapat dilihat pada Kode Sumber 4.11.

```
1  void relax(int x, int y, int bit, int val, int
       nextVal){
2      if(val>nextVal){
3          MST[x][y][bit]=nextVal;
4          state temp;
5          temp.x=x;
6          temp.y=y;
7          temp.bit=bit;
8          temp.val=MST[x][y][bit];
9          pq.push(temp);
10     }
11 }
```

Kode Sumber 4.11 Fungsi Relax Dijkstra

4.2.9.3 Implementasi Fungsi Solve pada Dijkstra

Pada subbab ini akan dijelaskan mengenai implementasi fungsi `Solve` dari Pseudocode 3.9. Implementasi dapat dilihat pada Kode Sumber 4.12.

```
1  double solve(){
2      if (n==1) return 0;
3      memset(MST,0,sizeof MST);
4      pemampatanKoordinat(n, compX, compY, X, Y);
5      initialSteinerTree(n, compX, compY, X, Y);
6      double ans = (double)dijkstra(n, compX,
           compY, X, Y);
7      return ans;
8  }
```

Kode Sumber 4.12 Fungsi Solve Dijkstra

4.2.9.4 Implementasi Fungsi Relaxation

Pada subbab ini akan dijelaskan mengenai implementasi fungsi Relaxation dari Pseudocode 3.7. Implementasi dari fungsi `Relaxation` dapat dilihat pada Kode Sumber 4.13.

```
 1 void relaxation(state top){
 2     int xIdx, yIdx;
 3     xIdx = compX.size();
 4     yIdx = compY.size();
 5
 6     int x = top.x;
 7     int y = top.y;
 8     int bit = top.bit;
 9     int val = top.val;
10     for(int i=(bit+1)|bit; i<(1<<n); i=(i+1)|bit){
11         int nextVal = val + MST[x][y][i^bit];
12         relax(x, y, i, MST[x][y][i], nextVal);
13     }
14     if(x){
15         int nextVal = val+compX[x]-compX[x-1];
16         relax(x-1, y, bit, MST[x-1][y][bit],
               nextVal);
17     }
18     if(y){
19         int nextVal = val+compY[y]-compY[y-1];
20         relax(x, y-1, bit, MST[x][y-1][bit],
               nextVal);
21     }
22     if(x<xIdx-1){
23         int nextVal = val+compX[x+1]-compX[x];
24         relax(x+1, y, bit, MST[x+1][y][bit],
               nextVal);
25     }
26     if(y<yIdx-1){
27         int nextVal = val+compY[y+1]-compY[y];
28         relax(x, y+1, bit, MST[x][y+1][bit],
               nextVal);
29     }
30 }
```

Kode Sumber 4.13 Fungsi Relaxation

4.2.10 Implementasi fungsi yang digunakan pada Shortest Path Faster Algorithm

Pada subbab ini akan dijelaskan mengenai implementasi dari fungsi-fungsi yang digunakan pada metode *Shortest Path Faster Algorithm*. Desain untuk fungsi-fungsi ini dapat dilihat pada Subbab 3.4.

4.2.10.1 Implementasi Fungsi Shortest Path Faster Algorithm

Pada subbab ini akan dijelaskan mengenai implementasi fungsi dari *Shortest Path Faster Algorithm* (SPFA) yang mengacu pada Subbab 2.7.2 dan Pseudocode 3.10. Implementasi dari fungsi `SPFA` dapat dilihat pada Kode Sumber 4.14.

```
 1 int spfa(){
 2     int xIdx, yIdx;
 3     xIdx = compX.size();
 4     yIdx = compY.size();
 5     int ans=INT_MAX;
 6     while(!q.empty()){
 7         state top = q.front();
 8         q.pop();
 9         int x = top.x;
10         int y = top.y;
11         int bit = top.bit;
12         top.val = MST[x][y][bit];
13         int val = top.val;
14         if(bit==(1<<n)-1){
15             ans=min(ans,val);
16         }
17         visited[x][y][bit]=false;
18         relaxation(top);
19     }
20     return ans;
21 }
```

Kode Sumber 4.14 Fungsi Shortest Path Faster Algorithm

4.2.10.2 Implementasi Fungsi Relax pada SPFA

Pada subbab ini akan dijelaskan mengenai implementasi fungsi *Relax* dari Pseudocode 3.11. Implementasi dapat dilihat pada Kode Sumber 4.15.

```
 1 void relax(int x, int y, int bit, int val, int
       nextVal){
 2     if(val>nextVal){
 3         MST[x][y][bit]=nextVal;
 4         if(!visited[x][y][bit]){
 5             state temp;
 6             temp.x=x;
 7             temp.y=y;
 8             temp.bit=bit;
 9             temp.val=MST[x][y][bit];
10             visited[x][y][bit]=true;
11             q.push(temp);
12         }
13     }
14 }
```

Kode Sumber 4.15 Fungsi Relax SPFA

4.2.10.3 Implementasi Fungsi Solve pada SPFA

Pada subbab ini akan dijelaskan mengenai implementasi fungsi `Solve` dari Pseudocode 3.12. Implementasi dapat dilihat pada Kode Sumber 4.16.

```
1 double solve(){
2     memset(visited, false, sizeof visited);
3     if (n==1) return 0;
4     vector<int> compX, compY;
5     pemampatanKoordinat(n, compX, compY, X, Y);
6     memset(MST,0,sizeof MST);
7     initialSteinerTree(n, compX, compY, X, Y);
8     return (double)spfa(n, compX, compY, X, Y);
9 }
```

Kode Sumber 4.16 Fungsi Solve pada SPFA

BAB V
UJI COBA DAN EVALUASI

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

5.1 Lingkungan Uji Coba

Lingkungan uji coba digunakan untuk uji coba kebenaran adalah salah satu sistem yang digunakan situs penilaian daring E-Olymp.

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

1. Perangkat Keras
   (a) Processor Intel® Core™ i5-4460S CPU @ 2.90GHz (4 CPUs)
   (b) Random Access Memory 5905MB
2. Perangkat Lunak
   (a) Sistem Operasi Ubuntu 18.04.3 LTS
   (b) Visual Studio Code
   (c) Bahasa Pemrograman C++
   (d) Kompiler GCC 7.4.0 (Ubuntu 7.4.0-1ubuntu1 18.04.1)

5.2 Skenario Uji Coba

Subbab ini akan menjelaskan hasil pengujian program untuk menyelesaiakan permasalahan E-Olymp 1445 - Road Network. Pengujian dilakukan untuk dua algoritma Dijkstra dan *Shortest Path Faster Algorithm*. Metode pengujian yang dilakukan adalah sebagai berikut.

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

Dalam pengujian lokal, dibuat beberapa kasus uji berdasarkan batasan nilai N pada subbab 1.3 dengan menggunakan nilai X dan Y acak. Sampel yang digunakan sebagai kasus uji dalam pengujian lokal dibuat menggunakan program pembuat kasus uji dengan mengambil N nilai X dan Y acak dari setiap rentang nilai $-1000 \geq X, Y \geq 1000$.

Untuk pengujian luar, uji coba dilakukan dengan mengirimkan kode program dengan algoritma Dijkstra dan kode program dengan algoritma *Shortest Path Faster Algorithm* ke situs penilaian daring E-Olymp masing masing sebanyak 10 kali.

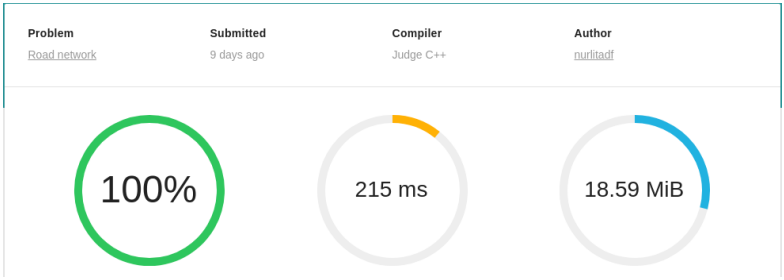
5.3 Uji Coba Kebenaran

Pada subbab ini akan dibahas mengenai uji coba kebenaran yang dilakukan dengan mengirim kode sumber terkait ke dalam situs penilaian daring E-Olymp. Berikut bukti hasil pengujian.

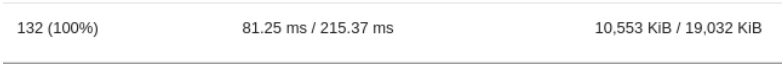
1. Kode sumber dengan metode Dijkstra (Gambar 5.1 dan Gambar 5.2)
2. Kode sumber dengan metode *Shortest Path Faster Algorithm* (Gambar 5.3 dan Gambar 5.4)

5.4 Uji Coba Kinerja Lokal

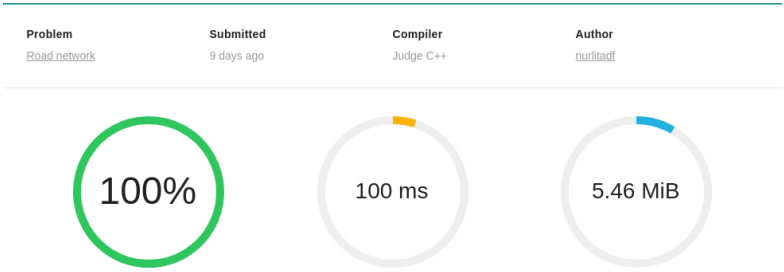
Pada subbab ini akan ditampilkan hasil uji coba kinerja dari algoritma Dijkstra dan *Shortest Path Faster Algorithm* untuk menyelesaikan permasalahan E-Olymp 1445 - Road Network. Pengujian dilakukan terhadap kelompok masukkan yang telah dijelaskan pada subbab 5.2. Detail masukkan dapat dilihat pada Lampiran A.



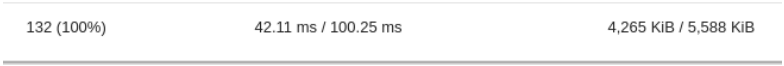
Gambar 5.1 Umpan Balik Metode Dijkstra



Gambar 5.2 Jumlah Data Uji dan Rata-Rata Waktu dan Memori dengan Metode Dijkstra



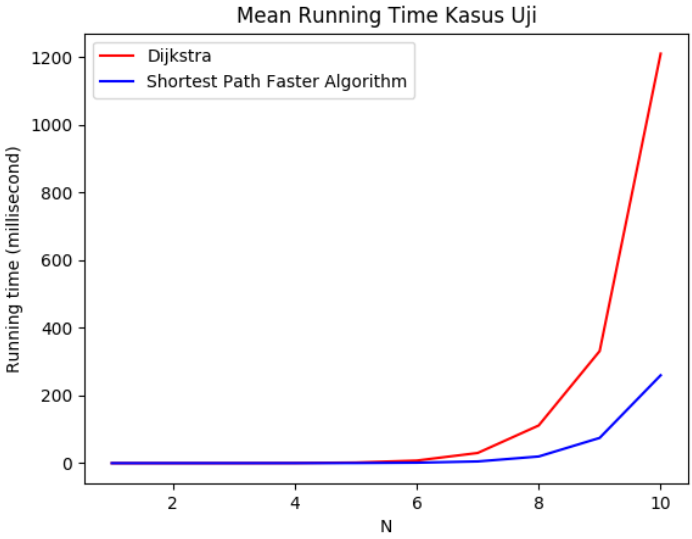
Gambar 5.3 Umpan Balik Metode Shortest Path Faster Algorithm



Gambar 5.4 Jumlah Data Uji dan Rata-Rata Waktu dan Memori dengan Metode Shortest Path Faster Algorithm

Langkah pengujian kinerja dilakukan dengan langkah sebagai berikut.

1. Rekam waktu tepat sebelum komputasi penyelesaian masalah dilakukan.
2. Melakukan komputasi penyelesaian masalah untuk masukan kasus uji.
3. Rekam waktu tepat setelah komputasi penyelesaian masalah dilakukan.
4. Menghitung durasi waktu komputasi dengan mengurangi waktu selesai komputasi dengan waktu sebelum komputasi.
5. Ulangi untuk seluruh kasus uji.



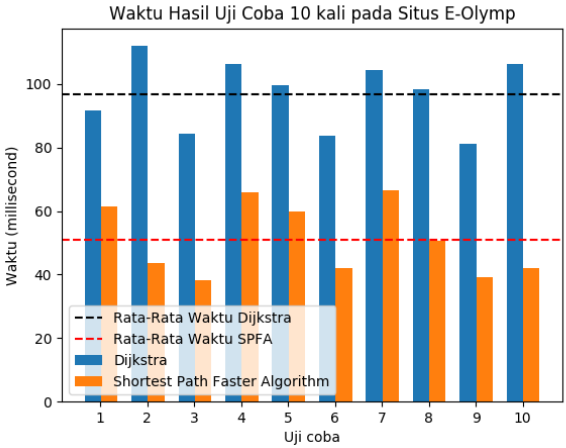
Gambar 5.5 Rata-rata *Running Time* untuk setiap nilai N Kasus Uji

Grafik 5.5 menampilkan rata-rata kinerja masing-masing metode. Pada grafik tersebut dapat dilihat bahwa rata-rata *running time* algoritma Dijkstra dan *Shortest Path Faster Algorithm* relatif sama untuk nilai $N < 6$. Namun untuk nilai $N \geq 6$, rata-rata *running*

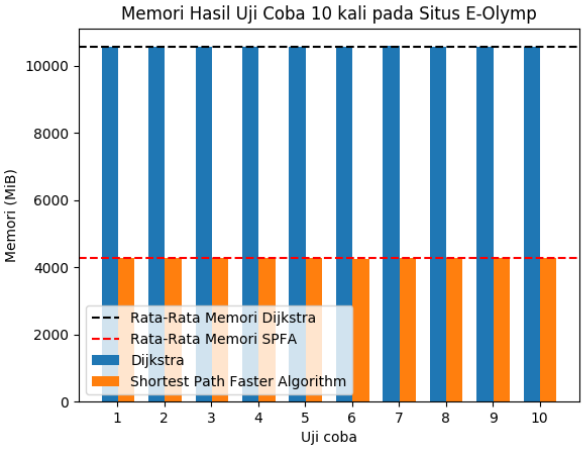
time dari algoritma *Shortest Path Faster Algorithm* cenderung lebih rendah dari pada algoritma Dijkstra.

5.5 Uji Coba Kinerja Luar

Pada subbab ini akan ditampilkan hasil uji coba kinerja dari algoritma Dijkstra dan *Shortest Path Faster Algorithm* untuk menyelesaikan permasalahan E-Olymp 1445 - Road Network. Pengujian dilakukan dengan cara submit kode program ke situs penilaian daring E-Olymp. Detail mengenai hasil uji kinerja dapat dilihat pada Lampiran B dan Lampiran C. Rata rata hasil pengumpulan kode berkas dengan *Dijkstra* adalah $96,825$ *millisecond* dengan memori $10,8$ MB, sementara dengan *Shortest Path Faster Algorithm* adalah $50,991$ *millisecond* dengan memori $4,3$ MB. Grafik waktu dan memori hasil pengumpulan kode berkas dapat dilihat pada Gambar 5.6 dan Gambar 5.7.



Gambar 5.6 Grafik Waktu Hasil Pengumpulan Kode Berkas sebanyak 10 kali



Gambar 5.7 Grafik Memori Hasil Pengumpulan Kode Berkas sebanyak 10 kali

5.6 Analisis dan Kesimpulan Umum

Dari pengujian kinerja didapatkan informasi mengenai kinerja Dijkstra dan *Shortest Path Faster Algorithm*

1. Rata-rata *running time* algoritma *Shortest Path Faster Algorithm* dan Dijkstra relatif sama untuk nilai $N < 6$.
2. Rata-rata *running time* dari algoritma *Shortest Path Faster Algorithm* cenderung lebih rendah dari pada algoritma Dijkstra ketika $N \geq 6$.
3. Algoritma *Shortest Path Faster Algorithm* lebih cepat dari Dijkstra dikarenakan struktur data yang digunakan pada *Shortest Path Faster Algorithm* yaitu *queue* memiliki kompleksitas operasi sebesar $O(1)$, sedangkan struktur data yang digunakan Dijkstra yaitu *priority queue* memiliki kompleksitas operasi sebesar $O(logN)$.
4. Rata-rata penggunaan memori algoritma *Shortest Path Faster*

Algorithm cenderung lebih rendah dari pada algoritma Dijkstra.

[*Halaman ini sengaja dikosongkan*]

BAB VI
KESIMPULAN

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

6.1 Kesimpulan

Berdasarkan penjabaran di bab-bab sebelumnya, dapat disimpulkan beberapa poin terkait penyelesaian permasalahan E-Olymp 1445 - Road Network.

1. Permasalahan E-Olymp 1445 - Road Network dapat diselesaikan dengan memodelkan graf ke dalam bentuk *Steiner Tree*.
2. Permasalahan E-Olymp 1445 - Road Network dengan batasan pada soal dapat diselesaikan menggunakan algoritma Dijkstra dengan waktu minimum $81,25$ *millisecond*, waktu maksimum $111,99$ *millisecond*, memori $10,8$ MB dan *Shortest Path Faster Algorithm* dengan waktu minimum $39,07$ *millisecond*, waktu maksimum $66,48$ *millisecond*, memori $4,3$ MB.
3. Algoritma Dijkstra dan *Shortest Path Faster Algorithm* dapat digunakan untuk menyelesaikan persoalan *Minimum Steiner Tree*.
4. Kinerja dari algoritma *Shortest Path Faster Algorithm* lebih baik dari pada algoritma Dijkstra pada penyelesaian E-Olymp 1445 - Road Network.

6.2 Saran

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

yang dapat diambil melalui tugas akhir ini:

1. Algoritma *shortest path* selain Dijkstra dan *Shortest Path Faster Algorithm* dapat diuji kinerjanya untuk menyelesaikan permasalahan *Minimum Steiner Tree*.

DAFTAR PUSTAKA

[1] E. A. Bender and S. G. Williamson, *Lists, Decision and Graphs*. 2010.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithm*, 3rd ed. 2009, ch. 24.

[3] E-Olymp. (2019). Road Network, [Online]. Available: https://www.e-olymp.com/en/problems/1445.

[4] S. Halim and F. Halim, *Competitive Programming*, 3rd ed. 2013, ch. 9.

[5] G. Robins and A. Zelikovsky, "Minimum Steiner Tree Construction", 2008. [Online]. Available: https://www.researchgate.net/profile/G_Robins/publication/228526520_Minimum_Steiner_Tree_Construction/links/00b49520e42cfe6ff5000000/Minimum-Steiner-Tree-Construction.pdf.

[*Halaman ini sengaja dikosongkan*]

LAMPIRAN A: DATA UJI

Tabel A.1 Data Uji dengan N = 1

No	x	y	No	x	y
1	-135	-966	26	-708	-958
2	515	564	27	-346	597
3	767	796	28	-240	-561
4	-511	-58	29	890	-347
5	-534	-719	30	-280	756
6	418	-632	31	-438	230
7	242	-721	32	-957	-939
8	-785	785	33	255	453
9	308	-673	34	328	-369
10	-541	-592	35	794	93
11	733	433	36	-969	973
12	305	-282	37	834	800
13	-772	-264	38	393	-148
14	-664	-276	39	783	-846
15	867	909	40	-264	650
16	-490	-712	41	376	29
17	500	-976	42	249	-970
18	852	267	43	-818	-991
19	820	-659	44	-974	-928
20	-792	843	45	-781	-253
21	-378	183	46	385	-219
22	769	421	47	977	429
23	19	984	48	-600	-212
24	205	-673	49	-119	285
25	-690	222	50	-23	-769

Tabel A.2 Data Uji dengan N = 2 (1)

No	x	y	No	x	y
1	-135 515	-966 564	26	-708 -346	-958 597
2	767 -511	796 -58	27	-240 890	-561 -347
3	-534 418	-719 -632	28	-280 -438	756 230
4	242 -785	-721 785	29	-957 255	-939 453
5	308 -541	-673 -592	30	328 794	-369 93
6	733 305	433 -282	31	-969 834	973 800
7	-772 -664	-264 -276	32	393 783	-148 -846
8	867 -490	909 -712	33	-264 376	650 29
9	500 852	-976 267	34	249 -818	-970 -991
10	820 -792	-659 843	35	-974 -781	-928 -253
11	-378 769	183 421	36	385 977	-219 429
12	19 205	984 -673	37	-600 -119	-212 285
13	-690 -708	222 -958	38	-23 936	-769 8
14	-346 -240	597 -561	39	761 -636	769 153
15	890 -280	-347 756	40	-380 -136	704 357
16	-438 -957	230 -939	41	353 943	-760 -399

Tabel A.3 Data Uji dengan N = 2 (2)

17	255 328	453 -369	42	828 -833	-876 411
18	794 -969	93 973	43	754 -843	-613 139
19	834 393	800 -148	44	-275 -876	692 -318
20	783 -264	-846 650	45	-964 968	-438 13
21	376 249	29 -970	46	-207 578	903 -447
22	-818 -974	-991 -928	47	671 263	942 848
23	-781 385	-253 -219	48	645 -239	-874 -446
24	977 -600	429 -212	49	924 -287	-297 308
25	-119 -23	285 -769	50	-615 -281	-563 -862

Tabel A.4 Data Uji dengan N = 3

No	x	y	No	x	y
1	-135	-966	26	-708	-958
	515	564		-346	597
	767	796		-240	-561
2	-511	-58	27	890	-347
	-534	-719		-280	756
	418	-632		-438	230
3	242	-721	28	-957	-939
	-785	785		255	453
	308	-673		328	-369
4	-541	-592	29	794	93
	733	433		-969	973
	305	-282		834	800
5	-772	-264	30	393	-148
	-664	-276		783	-846
	867	909		-264	650
6	-490	-712	31	376	29
	500	-976		249	-970
	852	267		-818	-991
7	820	-659	32	-974	-928
	-792	843		-781	-253
	-378	183		385	-219
8	769	421	33	977	429
	19	984		-600	-212
	205	-673		-119	285
9	-690	222	34	-23	-769
	-708	-958		936	8
	-346	597		761	769
10	-240	-561	35	-636	153
	890	-347		-380	704
	-280	756		-136	357
11	-438	230	36	353	-760
	-957	-939		943	-399
	255	453		828	-876

Tabel A.5 Data Uji dengan N = 3 (2)

12	328 794 -969	-369 93 973	37	-833 754 -843	411 -613 139
13	834 393 783	800 -148 -846	38	-275 -876 -964	692 -318 -438
14	-264 376 249	650 29 -970	39	968 -207 578	13 903 -447
15	-818 -974 -781	-991 -928 -253	40	671 263 645	942 848 -874
16	385 977 -600	-219 429 -212	41	-239 924 -287	-446 -297 308
17	-119 -23 936	285 -769 8	42	-615 -281 -176	-563 -862 -124
18	761 -636 -380	769 153 704	43	277 -876 789	107 -42 -839
19	-136 353 943	357 -760 -399	44	520 -269 215	756 870 -691
20	828 -833 754	-876 411 -613	45	-578 808 -710	-558 242 10
21	-843 -275 -876	139 692 -318	46	-74 121 312	51 -151 834
22	-964 968 -207	-438 13 903	47	715 -729 835	697 -567 -348

Tabel A.6 Data Uji dengan N = 3 (3)

23	578 671 263	-447 942 848	48	-134 316 627	-331 -9 105
24	645 -239 924	-874 -446 -297	49	-291 -583 16	147 440 190
25	-287 -615 -281	308 -563 -862	50	306 632 795	-4 -329 922

Tabel A.7 Data Uji dengan N = 4 (1)

No	x	y	No	x	y
1	-135	-966	26	-708	-958
	515	564		-346	597
	767	796		-240	-561
	-511	-58		890	-347
2	-534	-719	27	-280	756
	418	-632		-438	230
	242	-721		-957	-939
	-785	785		255	453
3	308	-673	28	328	-369
	-541	-592		794	93
	733	433		-969	973
	305	-282		834	800
4	-772	-264	29	393	-148
	-664	-276		783	-846
	867	909		-264	650
	-490	-712		376	29
5	500	-976	30	249	-970
	852	267		-818	-991
	820	-659		-974	-928
	-792	843		-781	-253
6	-378	183	31	385	-219
	769	421		977	429
	19	984		-600	-212
	205	-673		-119	285
7	-690	222	32	-23	-769
	-708	-958		936	8
	-346	597		761	769
	-240	-561		-636	153

Tabel A.8 Data Uji dengan N = 4 (2)

8	890	-347	33	-380	704
	-280	756		-136	357
	-438	230		353	-760
	-957	-939		943	-399
9	255	453	34	828	-876
	328	-369		-833	411
	794	93		754	-613
	-969	973		-843	139
10	834	800	35	-275	692
	393	-148		-876	-318
	783	-846		-964	-438
	-264	650		968	13
11	376	29	36	-207	903
	249	-970		578	-447
	-818	-991		671	942
	-974	-928		263	848
12	-781	-253	37	645	-874
	385	-219		-239	-446
	977	429		924	-297
	-600	-212		-287	308
13	-119	285	38	-615	-563
	-23	-769		-281	-862
	936	8		-176	-124
	761	769		277	107
14	-636	153	39	-876	-42
	-380	704		789	-839
	-136	357		520	756
	353	-760		-269	870

Tabel A.9 Data Uji dengan N = 4 (3)

15	943 828 -833 754	-399 -876 411 -613	40	215 -578 808 -710	-691 -558 242 10
16	-843 -275 -876 -964	139 692 -318 -438	41	-74 121 312 715	51 -151 834 697
17	968 -207 578 671	13 903 -447 942	42	-729 835 -134 316	-567 -348 -331 -9
18	263 645 -239 924	848 -874 -446 -297	43	627 -291 -583 16	105 147 440 190
19	-287 -615 -281 -176	308 -563 -862 -124	44	306 632 795 238	-4 -329 922 -280
20	277 -876 789 520	107 -42 -839 756	45	-470 127 -808 538	-642 842 -159 -980
21	-269 215 -578 808	870 -691 -558 242	46	-169 -771 155 802	-71 254 546 -218

Tabel A.10 Data Uji dengan N = 4 (4)

22	-710	10	47	-793	-490
	-74	51		486	-376
	121	-151		508	-942
	312	834		814	-630
23	715	697	48	-389	3
	-729	-567		41	964
	835	-348		-76	836
	-134	-331		-317	11
24	316	-9	49	-806	367
	627	105		-148	944
	-291	147		765	947
	-583	440		521	-848
25	16	190	50	-567	750
	306	-4		407	588
	632	-329		-148	208
	795	922		927	59

Tabel A.11 Data Uji dengan N = 5 (1)

No	x	y	No	x	y
1	-135	-966	26	-708	-958
	515	564		-346	597
	767	796		-240	-561
	-511	-58		890	-347
	-534	-719		-280	756
2	418	-632	27	-438	230
	242	-721		-957	-939
	-785	785		255	453
	308	-673		328	-369
	-541	-592		794	93
3	733	433	28	-969	973
	305	-282		834	800
	-772	-264		393	-148
	-664	-276		783	-846
	867	909		-264	650
4	-490	-712	29	376	29
	500	-976		249	-970
	852	267		-818	-991
	820	-659		-974	-928
	-792	843		-781	-253
5	-378	183	30	385	-219
	769	421		977	429
	19	984		-600	-212
	205	-673		-119	285
	-690	222		-23	-769

Tabel A.12 Data Uji dengan N = 5 (2)

6	-708	-958	31	936	8
	-346	597		761	769
	-240	-561		-636	153
	890	-347		-380	704
	-280	756		-136	357
7	-438	230	32	353	-760
	-957	-939		943	-399
	255	453		828	-876
	328	-369		-833	411
	794	93		754	-613
8	-969	973	33	-843	139
	834	800		-275	692
	393	-148		-876	-318
	783	-846		-964	-438
	-264	650		968	13
9	376	29	34	-207	903
	249	-970		578	-447
	-818	-991		671	942
	-974	-928		263	848
	-781	-253		645	-874
10	385	-219	35	-239	-446
	977	429		924	-297
	-600	-212		-287	308
	-119	285		-615	-563
	-23	-769		-281	-862

Tabel A.13 Data Uji dengan N = 5 (3)

11	936	8	36	-176	-124
	761	769		277	107
	-636	153		-876	-42
	-380	704		789	-839
	-136	357		520	756
12	353	-760	37	-269	870
	943	-399		215	-691
	828	-876		-578	-558
	-833	411		808	242
	754	-613		-710	10
13	-843	139	38	-74	51
	-275	692		121	-151
	-876	-318		312	834
	-964	-438		715	697
	968	13		-729	-567
14	-207	903	39	835	-348
	578	-447		-134	-331
	671	942		316	-9
	263	848		627	105
	645	-874		-291	147
15	-239	-446	40	-583	440
	924	-297		16	190
	-287	308		306	-4
	-615	-563		632	-329
	-281	-862		795	922

Tabel A.14 Data Uji dengan N = 5 (4)

16	-176	-124	41	238	-280
	277	107		-470	-642
	-876	-42		127	842
	789	-839		-808	-159
	520	756		538	-980
17	-269	870	42	-169	-71
	215	-691		-771	254
	-578	-558		155	546
	808	242		802	-218
	-710	10		-793	-490
18	-74	51	43	486	-376
	121	-151		508	-942
	312	834		814	-630
	715	697		-389	3
	-729	-567		41	964
19	835	-348	44	-76	836
	-134	-331		-317	11
	316	-9		-806	367
	627	105		-148	944
	-291	147		765	947
20	-583	440	45	521	-848
	16	190		-567	750
	306	-4		407	588
	632	-329		-148	208
	795	922		927	59

Tabel A.15 Data Uji dengan N = 5 (5)

	238	-280		276	-31
	-470	-642		241	-217
21	127	842	46	28	-389
	-808	-159		153	196
	538	-980		614	752
	-169	-71		159	-905
	-771	254		587	400
22	155	546	47	106	338
	802	-218		-234	959
	-793	-490		281	-912
	486	-376		462	-199
	508	-942		-760	895
23	814	-630	48	-892	647
	-389	3		40	-40
	41	964		-588	-34
	-76	836		-981	688
	-317	11		936	260
24	-806	367	49	-530	-480
	-148	944		871	180
	765	947		716	42
	521	-848		-512	-568
	-567	750		137	633
25	407	588	50	832	-758
	-148	208		-30	-846
	927	59		758	809

Tabel A.16 Data Uji dengan N = 6 (1)

No	x	y	No	x	y
1	-135	-966	26	-708	-958
	515	564		-346	597
	767	796		-240	-561
	-511	-58		890	-347
	-534	-719		-280	756
	418	-632		-438	230
2	242	-721	27	-957	-939
	-785	785		255	453
	308	-673		328	-369
	-541	-592		794	93
	733	433		-969	973
	305	-282		834	800
3	-772	-264	28	393	-148
	-664	-276		783	-846
	867	909		-264	650
	-490	-712		376	29
	500	-976		249	-970
	852	267		-818	-991
4	820	-659	29	-974	-928
	-792	843		-781	-253
	-378	183		385	-219
	769	421		977	429
	19	984		-600	-212
	205	-673		-119	285
5	-690	222	30	-23	-769
	-708	-958		936	8
	-346	597		761	769
	-240	-561		-636	153
	890	-347		-380	704
	-280	756		-136	357

Tabel A.17 Data Uji dengan N = 6 (2)

6	-438	230	31	353	-760
	-957	-939		943	-399
	255	453		828	-876
	328	-369		-833	411
	794	93		754	-613
	-969	973		-843	139
7	834	800	32	-275	692
	393	-148		-876	-318
	783	-846		-964	-438
	-264	650		968	13
	376	29		-207	903
	249	-970		578	-447
8	-818	-991	33	671	942
	-974	-928		263	848
	-781	-253		645	-874
	385	-219		-239	-446
	977	429		924	-297
	-600	-212		-287	308
9	-119	285	34	-615	-563
	-23	-769		-281	-862
	936	8		-176	-124
	761	769		277	107
	-636	153		-876	-42
	-380	704		789	-839
10	-136	357	35	520	756
	353	-760		-269	870
	943	-399		215	-691
	828	-876		-578	-558
	-833	411		808	242
	754	-613		-710	10

Tabel A.18 Data Uji dengan N = 6 (3)

11	-843	139	36	-74	51
	-275	692		121	-151
	-876	-318		312	834
	-964	-438		715	697
	968	13		-729	-567
	-207	903		835	-348
12	578	-447	37	-134	-331
	671	942		316	-9
	263	848		627	105
	645	-874		-291	147
	-239	-446		-583	440
	924	-297		16	190
13	-287	308	38	306	-4
	-615	-563		632	-329
	-281	-862		795	922
	-176	-124		238	-280
	277	107		-470	-642
	-876	-42		127	842
14	789	-839	39	-808	-159
	520	756		538	-980
	-269	870		-169	-71
	215	-691		-771	254
	-578	-558		155	546
	808	242		802	-218
15	-710	10	40	-793	-490
	-74	51		486	-376
	121	-151		508	-942
	312	834		814	-630
	715	697		-389	3
	-729	-567		41	964

Tabel A.19 Data Uji dengan N = 6 (4)

16	835	-348	41	-76	836
	-134	-331		-317	11
	316	-9		-806	367
	627	105		-148	944
	-291	147		765	947
	-583	440		521	-848
17	16	190	42	-567	750
	306	-4		407	588
	632	-329		-148	208
	795	922		927	59
	238	-280		276	-31
	-470	-642		241	-217
18	127	842	43	28	-389
	-808	-159		153	196
	538	-980		614	752
	-169	-71		159	-905
	-771	254		587	400
	155	546		106	338
19	802	-218	44	-234	959
	-793	-490		281	-912
	486	-376		462	-199
	508	-942		-760	895
	814	-630		-892	647
	-389	3		40	-40
20	41	964	45	-588	-34
	-76	836		-981	688
	-317	11		936	260
	-806	367		-530	-480
	-148	944		871	180
	765	947		716	42

Tabel A.20 Data Uji dengan N = 6 (5)

21	521	-848	46	-512	-568
	-567	750		137	633
	407	588		832	-758
	-148	208		-30	-846
	927	59		758	809
	276	-31		-758	220
22	241	-217	47	-834	-960
	28	-389		-329	-726
	153	196		244	711
	614	752		235	656
	159	-905		-766	-189
	587	400		-100	-831
23	106	338	48	628	370
	-234	959		-311	55
	281	-912		-893	962
	462	-199		-904	-848
	-760	895		-607	233
	-892	647		785	782
24	40	-40	49	33	-245
	-588	-34		937	-210
	-981	688		-880	736
	936	260		567	-714
	-530	-480		333	796
	871	180		-439	-423
25	716	42	50	506	353
	-512	-568		790	740
	137	633		-837	-310
	832	-758		466	348
	-30	-846		617	712
	758	809		-598	281

Tabel A.21 Data Uji dengan N = 7 (1)

No	x	y	No	x	y
1	-135	-966	26	-708	-958
	515	564		-346	597
	767	796		-240	-561
	-511	-58		890	-347
	-534	-719		-280	756
	418	-632		-438	230
	242	-721		-957	-939
2	-785	785	27	255	453
	308	-673		328	-369
	-541	-592		794	93
	733	433		-969	973
	305	-282		834	800
	-772	-264		393	-148
	-664	-276		783	-846
3	867	909	28	-264	650
	-490	-712		376	29
	500	-976		249	-970
	852	267		-818	-991
	820	-659		-974	-928
	-792	843		-781	-253
	-378	183		385	-219
4	769	421	29	977	429
	19	984		-600	-212
	205	-673		-119	285
	-690	222		-23	-769
	-708	-958		936	8
	-346	597		761	769
	-240	-561		-636	153

Tabel A.22 Data Uji dengan N = 7 (2)

	890	-347		-380	704
	-280	756		-136	357
	-438	230		353	-760
5	-957	-939	30	943	-399
	255	453		828	-876
	328	-369		-833	411
	794	93		754	-613
	-969	973		-843	139
	834	800		-275	692
	393	-148		-876	-318
6	783	-846	31	-964	-438
	-264	650		968	13
	376	29		-207	903
	249	-970		578	-447
	-818	-991		671	942
	-974	-928		263	848
	-781	-253		645	-874
7	385	-219	32	-239	-446
	977	429		924	-297
	-600	-212		-287	308
	-119	285		-615	-563
	-23	-769		-281	-862
	936	8		-176	-124
	761	769		277	107
8	-636	153	33	-876	-42
	-380	704		789	-839
	-136	357		520	756
	353	-760		-269	870

Tabel A.23 Data Uji dengan N = 7 (3)

	943	-399		215	-691
	828	-876		-578	-558
	-833	411		808	242
9	754	-613	34	-710	10
	-843	139		-74	51
	-275	692		121	-151
	-876	-318		312	834
	-964	-438		715	697
	968	13		-729	-567
	-207	903		835	-348
10	578	-447	35	-134	-331
	671	942		316	-9
	263	848		627	105
	645	-874		-291	147
	-239	-446		-583	440
	924	-297		16	190
	-287	308		306	-4
11	-615	-563	36	632	-329
	-281	-862		795	922
	-176	-124		238	-280
	277	107		-470	-642
	-876	-42		127	842
	789	-839		-808	-159
	520	756		538	-980
12	-269	870	37	-169	-71
	215	-691		-771	254
	-578	-558		155	546
	808	242		802	-218

Tabel A.24 Data Uji dengan N = 7 (4)

	-710	10		-793	-490
	-74	51		486	-376
	121	-151		508	-942
13	312	834	38	814	-630
	715	697		-389	3
	-729	-567		41	964
	835	-348		-76	836
	-134	-331		-317	11
	316	-9		-806	367
	627	105		-148	944
14	-291	147	39	765	947
	-583	440		521	-848
	16	190		-567	750
	306	-4		407	588
	632	-329		-148	208
	795	922		927	59
	238	-280		276	-31
15	-470	-642	40	241	-217
	127	842		28	-389
	-808	-159		153	196
	538	-980		614	752
	-169	-71		159	-905
	-771	254		587	400
	155	546		106	338
16	802	-218	41	-234	959
	-793	-490		281	-912
	486	-376		462	-199
	508	-942		-760	895

Tabel A.25 Data Uji dengan N = 7 (5)

	814	-630		-892	647
	-389	3		40	-40
	41	964		-588	-34
17	-76	836	42	-981	688
	-317	11		936	260
	-806	367		-530	-480
	-148	944		871	180
	765	947		716	42
	521	-848		-512	-568
	-567	750		137	633
18	407	588	43	832	-758
	-148	208		-30	-846
	927	59		758	809
	276	-31		-758	220
	241	-217		-834	-960
	28	-389		-329	-726
	153	196		244	711
19	614	752	44	235	656
	159	-905		-766	-189
	587	400		-100	-831
	106	338		628	370
	-234	959		-311	55
	281	-912		-893	962
	462	-199		-904	-848
20	-760	895	45	-607	233
	-892	647		785	782
	40	-40		33	-245
	-588	-34		937	-210

Tabel A.26 Data Uji dengan N = 7 (6)

	-981	688		-880	736
	936	260		567	-714
	-530	-480		333	796
21	871	180	46	-439	-423
	716	42		506	353
	-512	-568		790	740
	137	633		-837	-310
	832	-758		466	348
	-30	-846		617	712
	758	809		-598	281
22	-758	220	47	231	-944
	-834	-960		434	181
	-329	-726		289	218
	244	711		-37	879
	235	656		530	-544
	-766	-189		-331	650
	-100	-831		-809	793
23	628	370	48	937	82
	-311	55		588	-946
	-893	962		659	-349
	-904	-848		407	5
	-607	233		948	127
	785	782		252	414
	33	-245		-969	-131
24	937	-210	49	125	-567
	-880	736		707	913
	567	-714		-954	140
	333	796		94	-107
	-439	-423		916	-944
	506	353		-229	445
	790	740		-488	-3
25	-837	-310	50	-348	-740
	466	348		-210	-855
	617	712		342	-623
	-598	281		757	-443

Tabel A.27 Data Uji dengan N = 8 (1)

No	x	y	No	x	y
1	-135	-966	26	-708	-958
	515	564		-346	597
	767	796		-240	-561
	-511	-58		890	-347
	-534	-719		-280	756
	418	-632		-438	230
	242	-721		-957	-939
	-785	785		255	453
2	308	-673	27	328	-369
	-541	-592		794	93
	733	433		-969	973
	305	-282		834	800
	-772	-264		393	-148
	-664	-276		783	-846
	867	909		-264	650
	-490	-712		376	29
3	500	-976	28	249	-970
	852	267		-818	-991
	820	-659		-974	-928
	-792	843		-781	-253
	-378	183		385	-219
	769	421		977	429
	19	984		-600	-212
	205	-673		-119	285
4	-690	222	29	-23	-769
	-708	-958		936	8
	-346	597		761	769
	-240	-561		-636	153
	890	-347		-380	704
	-280	756		-136	357
	-438	230		353	-760
	-957	-939		943	-399

Tabel A.28 Data Uji dengan N = 8 (2)

5	255	453	30	828	-876
	328	-369		-833	411
	794	93		754	-613
	-969	973		-843	139
	834	800		-275	692
	393	-148		-876	-318
	783	-846		-964	-438
	-264	650		968	13
6	376	29	31	-207	903
	249	-970		578	-447
	-818	-991		671	942
	-974	-928		263	848
	-781	-253		645	-874
	385	-219		-239	-446
	977	429		924	-297
	-600	-212		-287	308
7	-119	285	32	-615	-563
	-23	-769		-281	-862
	936	8		-176	-124
	761	769		277	107
	-636	153		-876	-42
	-380	704		789	-839
	-136	357		520	756
	353	-760		-269	870
8	943	-399	33	215	-691
	828	-876		-578	-558
	-833	411		808	242
	754	-613		-710	10
	-843	139		-74	51
	-275	692		121	-151
	-876	-318		312	834
	-964	-438		715	697

Tabel A.29 Data Uji dengan N = 8 (3)

9	968	13	34	-729	-567
	-207	903		835	-348
	578	-447		-134	-331
	671	942		316	-9
	263	848		627	105
	645	-874		-291	147
	-239	-446		-583	440
	924	-297		16	190
10	-287	308	35	306	-4
	-615	-563		632	-329
	-281	-862		795	922
	-176	-124		238	-280
	277	107		-470	-642
	-876	-42		127	842
	789	-839		-808	-159
	520	756		538	-980
11	-269	870	36	-169	-71
	215	-691		-771	254
	-578	-558		155	546
	808	242		802	-218
	-710	10		-793	-490
	-74	51		486	-376
	121	-151		508	-942
	312	834		814	-630
12	715	697	37	-389	3
	-729	-567		41	964
	835	-348		-76	836
	-134	-331		-317	11
	316	-9		-806	367
	627	105		-148	944
	-291	147		765	947
	-583	440		521	-848

Tabel A.30 Data Uji dengan N = 8 (4)

13	16	190	38	-567	750
	306	-4		407	588
	632	-329		-148	208
	795	922		927	59
	238	-280		276	-31
	-470	-642		241	-217
	127	842		28	-389
	-808	-159		153	196
14	538	-980	39	614	752
	-169	-71		159	-905
	-771	254		587	400
	155	546		106	338
	802	-218		-234	959
	-793	-490		281	-912
	486	-376		462	-199
	508	-942		-760	895
15	814	-630	40	-892	647
	-389	3		40	-40
	41	964		-588	-34
	-76	836		-981	688
	-317	11		936	260
	-806	367		-530	-480
	-148	944		871	180
	765	947		716	42
16	521	-848	41	-512	-568
	-567	750		137	633
	407	588		832	-758
	-148	208		-30	-846
	927	59		758	809
	276	-31		-758	220
	241	-217		-834	-960
	28	-389		-329	-726

Tabel A.31 Data Uji dengan N = 8 (5)

17	153	196	42	244	711
	614	752		235	656
	159	-905		-766	-189
	587	400		-100	-831
	106	338		628	370
	-234	959		-311	55
	281	-912		-893	962
	462	-199		-904	-848
18	-760	895	43	-607	233
	-892	647		785	782
	40	-40		33	-245
	-588	-34		937	-210
	-981	688		-880	736
	936	260		567	-714
	-530	-480		333	796
	871	180		-439	-423
19	716	42	44	506	353
	-512	-568		790	740
	137	633		-837	-310
	832	-758		466	348
	-30	-846		617	712
	758	809		-598	281
	-758	220		231	-944
	-834	-960		434	181
20	-329	-726	45	289	218
	244	711		-37	879
	235	656		530	-544
	-766	-189		-331	650
	-100	-831		-809	793
	628	370		937	82
	-311	55		588	-946
	-893	962		659	-349

Tabel A.32 Data Uji dengan N = 8 (6)

21	-904	-848	46	407	5
	-607	233		948	127
	785	782		252	414
	33	-245		-969	-131
	937	-210		125	-567
	-880	736		707	913
	567	-714		-954	140
	333	796		94	-107
22	-439	-423	47	916	-944
	506	353		-229	445
	790	740		-488	-3
	-837	-310		-348	-740
	466	348		-210	-855
	617	712		342	-623
	-598	281		757	-443
	231	-944		28	-280
23	434	181	48	120	-24
	289	218		404	-629
	-37	879		947	435
	530	-544		240	-372
	-331	650		425	-496
	-809	793		-459	472
	937	82		644	635
	588	-946		-636	559
24	659	-349	49	248	135
	407	5		-439	760
	948	127		-868	-230
	252	414		577	-78
	-969	-131		-528	-524
	125	-567		-144	-772
	707	913		-410	442
	-954	140		-52	710

Tabel A.33 Data Uji dengan N = 8 (7)

25	94	-107	50	975	-649
	916	-944		-919	921
	-229	445		343	-122
	-488	-3		-452	-233
	-348	-740		382	-354
	-210	-855		-762	-417
	342	-623		-720	-841
	757	-443		-859	528

Tabel A.34 Data Uji dengan N = 9 (1)

No	x	y	No	x	y
1	-135	-966	26	-708	-958
	515	564		-346	597
	767	796		-240	-561
	-511	-58		890	-347
	-534	-719		-280	756
	418	-632		-438	230
	242	-721		-957	-939
	-785	785		255	453
	308	-673		328	-369
2	-541	-592	27	794	93
	733	433		-969	973
	305	-282		834	800
	-772	-264		393	-148
	-664	-276		783	-846
	867	909		-264	650
	-490	-712		376	29
	500	-976		249	-970
	852	267		-818	-991
3	820	-659	28	-974	-928
	-792	843		-781	-253
	-378	183		385	-219
	769	421		977	429
	19	984		-600	-212
	205	-673		-119	285
	-690	222		-23	-769
	-708	-958		936	8
	-346	597		761	769

Tabel A.35 Data Uji dengan N = 9 (2)

4	-240	-561	29	-636	153
	890	-347		-380	704
	-280	756		-136	357
	-438	230		353	-760
	-957	-939		943	-399
	255	453		828	-876
	328	-369		-833	411
	794	93		754	-613
	-969	973		-843	139
5	834	800	30	-275	692
	393	-148		-876	-318
	783	-846		-964	-438
	-264	650		968	13
	376	29		-207	903
	249	-970		578	-447
	-818	-991		671	942
	-974	-928		263	848
	-781	-253		645	-874
6	385	-219	31	-239	-446
	977	429		924	-297
	-600	-212		-287	308
	-119	285		-615	-563
	-23	-769		-281	-862
	936	8		-176	-124
	761	769		277	107
	-636	153		-876	-42
	-380	704		789	-839

Tabel A.36 Data Uji dengan N = 9 (3)

	-136	357		520	756
	353	-760		-269	870
	943	-399		215	-691
	828	-876		-578	-558
7	-833	411	32	808	242
	754	-613		-710	10
	-843	139		-74	51
	-275	692		121	-151
	-876	-318		312	834
	-964	-438		715	697
	968	13		-729	-567
	-207	903		835	-348
	578	-447		-134	-331
8	671	942	33	316	-9
	263	848		627	105
	645	-874		-291	147
	-239	-446		-583	440
	924	-297		16	190
	-287	308		306	-4
	-615	-563		632	-329
	-281	-862		795	922
	-176	-124		238	-280
9	277	107	34	-470	-642
	-876	-42		127	842
	789	-839		-808	-159
	520	756		538	-980
	-269	870		-169	-71
	802	-218		-234	959
	-793	-490		281	-912
	486	-376		462	-199
	508	-942		-760	895
13	814	-630	38	-892	647
	-389	3		40	-40
	41	964		-588	-34
	-76	836		-981	688
	-317	11		936	260
	-806	367		-530	-480

Tabel A.37 Data Uji dengan N = 9 (4)

	215	-691		-771	254
	-578	-558		155	546
	808	242		802	-218
	-710	10		-793	-490
10	-74	51	35	486	-376
	121	-151		508	-942
	312	834		814	-630
	715	697		-389	3
	-729	-567		41	964
	835	-348		-76	836
	-134	-331		-317	11
	316	-9		-806	367
	627	105		-148	944
11	-291	147	36	765	947
	-583	440		521	-848
	16	190		-567	750
	306	-4		407	588
	632	-329		-148	208
	795	922		927	59
	238	-280		276	-31
	-470	-642		241	-217
	127	842		28	-389
12	-808	-159	37	153	196
	538	-980		614	752
	-169	-71		159	-905
	-771	254		587	400
	155	546		106	338

Tabel A.38 Data Uji dengan N = 9 (5)

	241	-217		-834	-960
	28	-389		-329	-726
	153	196		244	711
	614	752		235	656
15	159	-905	40	-766	-189
	587	400		-100	-831
	106	338		628	370
	-234	959		-311	55
	281	-912		-893	962

Tabel A.39 Data Uji dengan N = 9 (6)

	462	-199		-904	-848
	-760	895		-607	233
	-892	647		785	782
	40	-40		33	-245
16	-588	-34	41	937	-210
	-981	688		-880	736
	936	260		567	-714
	-530	-480		333	796
	871	180		-439	-423
	716	42		506	353
	-512	-568		790	740
	137	633		-837	-310
	832	-758		466	348
17	-30	-846	42	617	712
	758	809		-598	281
	-758	220		231	-944
	-834	-960		434	181
	-329	-726		289	218
	244	711		-37	879
	235	656		530	-544
	-766	-189		-331	650
	-100	-831		-809	793
18	628	370	43	937	82
	-311	55		588	-946
	-893	962		659	-349
	-904	-848		407	5
	-607	233		948	127

Tabel A.40 Data Uji dengan N = 9 (7)

	785	782		252	414
	33	-245		-969	-131
	937	-210		125	-567
	-880	736		707	913
19	567	-714	44	-954	140
	333	796		94	-107
	-439	-423		916	-944
	506	353		-229	445
	790	740		-488	-3
	-837	-310		-348	-740
	466	348		-210	-855
	617	712		342	-623
	-598	281		757	-443
20	231	-944	45	28	-280
	434	181		120	-24
	289	218		404	-629
	-37	879		947	435
	530	-544		240	-372
	-331	650		425	-496
	-809	793		-459	472
	937	82		644	635
	588	-946		-636	559
21	659	-349	46	248	135
	407	5		-439	760
	948	127		-868	-230
	252	414		577	-78
	-969	-131		-528	-524

Tabel A.41 Data Uji dengan N = 9 (8)

	125	-567		-144	-772
	707	913		-410	442
	-954	140		-52	710
	94	-107		975	-649
22	916	-944	47	-919	921
	-229	445		343	-122
	-488	-3		-452	-233
	-348	-740		382	-354
	-210	-855		-762	-417
	342	-623		-720	-841
	757	-443		-859	528
	28	-280		295	-741
	120	-24		-156	-16
23	404	-629	48	29	-579
	947	435		463	501
	240	-372		-546	-682
	425	-496		729	44
	-459	472		317	-767
	644	635		-246	291
	-636	559		-859	-165
	248	135		211	484
	-439	760		270	317
24	-868	-230	49	-749	-348
	577	-78		963	-954
	-528	-524		-208	801
	-144	-772		-794	-67
	-410	442		328	58

Tabel A.42 Data Uji dengan N = 9 (9)

	-52	710		-250	-828
	975	-649		-959	779
	-919	921		-850	504
	343	-122		-163	-396
25	-452	-233	50	379	-878
	382	-354		648	-747
	-762	-417		-644	959
	-720	-841		544	-946
	-859	528		-650	-688

Tabel A.43 Data Uji dengan N = 10 (1)

No	x	y	No	x	y
1	-135	-966	26	-708	-958
	515	564		-346	597
	767	796		-240	-561
	-511	-58		890	-347
	-534	-719		-280	756
	418	-632		-438	230
	242	-721		-957	-939
	-785	785		255	453
	308	-673		328	-369
	-541	-592		794	93
2	733	433	27	-969	973
	305	-282		834	800
	-772	-264		393	-148
	-664	-276		783	-846
	867	909		-264	650
	-490	-712		376	29
	500	-976		249	-970
	852	267		-818	-991
	820	-659		-974	-928
	-792	843		-781	-253
3	-378	183	28	385	-219
	769	421		977	429
	19	984		-600	-212
	205	-673		-119	285
	-690	222		-23	-769
	-708	-958		936	8
	-346	597		761	769
	-240	-561		-636	153
	890	-347		-380	704
	-280	756		-136	357

Tabel A.44 Data Uji dengan N = 10 (2)

4	-438	230	29	353	-760
	-957	-939		943	-399
	255	453		828	-876
	328	-369		-833	411
	794	93		754	-613
	-969	973		-843	139
	834	800		-275	692
	393	-148		-876	-318
	783	-846		-964	-438
	-264	650		968	13
5	376	29	30	-207	903
	249	-970		578	-447
	-818	-991		671	942
	-974	-928		263	848
	-781	-253		645	-874
	385	-219		-239	-446
	977	429		924	-297
	-600	-212		-287	308
	-119	285		-615	-563
	-23	-769		-281	-862
6	936	8	31	-176	-124
	761	769		277	107
	-636	153		-876	-42
	-380	704		789	-839
	-136	357		520	756
	353	-760		-269	870
	943	-399		215	-691
	828	-876		-578	-558
	-833	411		808	242
	754	-613		-710	10

Tabel A.45 Data Uji dengan N = 10 (3)

7	-843	139	32	-74	51
	-275	692		121	-151
	-876	-318		312	834
	-964	-438		715	697
	968	13		-729	-567
	-207	903		835	-348
	578	-447		-134	-331
	671	942		316	-9
	263	848		627	105
	645	-874		-291	147
8	-239	-446	33	-583	440
	924	-297		16	190
	-287	308		306	-4
	-615	-563		632	-329
	-281	-862		795	922
	-176	-124		238	-280
	277	107		-470	-642
	-876	-42		127	842
	789	-839		-808	-159
	520	756		538	-980
9	-269	870	34	-169	-71
	215	-691		-771	254
	-578	-558		155	546
	808	242		802	-218
	-710	10		-793	-490
	-74	51		486	-376
	121	-151		508	-942
	312	834		814	-630
	715	697		-389	3
	-729	-567		41	964

Tabel A.46 Data Uji dengan N = 10 (4)

10	835	-348	35	-76	836
	-134	-331		-317	11
	316	-9		-806	367
	627	105		-148	944
	-291	147		765	947
	-583	440		521	-848
	16	190		-567	750
	306	-4		407	588
	632	-329		-148	208
	795	922		927	59
11	238	-280	36	276	-31
	-470	-642		241	-217
	127	842		28	-389
	-808	-159		153	196
	538	-980		614	752
	-169	-71		159	-905
	-771	254		587	400
	155	546		106	338
	802	-218		-234	959
	-793	-490		281	-912
12	486	-376	37	462	-199
	508	-942		-760	895
	814	-630		-892	647
	-389	3		40	-40
	41	964		-588	-34
	-76	836		-981	688
	-317	11		936	260
	-806	367		-530	-480
	-148	944		871	180
	765	947		716	42

Tabel A.47 Data Uji dengan N = 10 (5)

13	521	-848	38	-512	-568
	-567	750		137	633
	407	588		832	-758
	-148	208		-30	-846
	927	59		758	809
	276	-31		-758	220
	241	-217		-834	-960
	28	-389		-329	-726
	153	196		244	711
	614	752		235	656
14	159	-905	39	-766	-189
	587	400		-100	-831
	106	338		628	370
	-234	959		-311	55
	281	-912		-893	962
	462	-199		-904	-848
	-760	895		-607	233
	-892	647		785	782
	40	-40		33	-245
	-588	-34		937	-210
15	-981	688	40	-880	736
	936	260		567	-714
	-530	-480		333	796
	871	180		-439	-423
	716	42		506	353
	-512	-568		790	740
	137	633		-837	-310
	832	-758		466	348
	-30	-846		617	712
	758	809		-598	281

Tabel A.48 Data Uji dengan N = 10 (6)

16	-758	220	41	231	-944
	-834	-960		434	181
	-329	-726		289	218
	244	711		-37	879
	235	656		530	-544
	-766	-189		-331	650
	-100	-831		-809	793
	628	370		937	82
	-311	55		588	-946
	-893	962		659	-349
17	-904	-848	42	407	5
	-607	233		948	127
	785	782		252	414
	33	-245		-969	-131
	937	-210		125	-567
	-880	736		707	913
	567	-714		-954	140
	333	796		94	-107
	-439	-423		916	-944
	506	353		-229	445
18	790	740	43	-488	-3
	-837	-310		-348	-740
	466	348		-210	-855
	617	712		342	-623
	-598	281		757	-443
	231	-944		28	-280
	434	181		120	-24
	289	218		404	-629
	-37	879		947	435
	530	-544		240	-372

Tabel A.49 Data Uji dengan N = 10 (7)

19	-331	650	44	425	-496
	-809	793		-459	472
	937	82		644	635
	588	-946		-636	559
	659	-349		248	135
	407	5		-439	760
	948	127		-868	-230
	252	414		577	-78
	-969	-131		-528	-524
	125	-567		-144	-772
20	707	913	45	-410	442
	-954	140		-52	710
	94	-107		975	-649
	916	-944		-919	921
	-229	445		343	-122
	-488	-3		-452	-233
	-348	-740		382	-354
	-210	-855		-762	-417
	342	-623		-720	-841
	757	-443		-859	528
21	28	-280	46	295	-741
	120	-24		-156	-16
	404	-629		29	-579
	947	435		463	501
	240	-372		-546	-682
	425	-496		729	44
	-459	472		317	-767
	644	635		-246	291
	-636	559		-859	-165
	248	135		211	484

Tabel A.50 Data Uji dengan N = 10 (8)

22	-439	760	47	270	317
	-868	-230		-749	-348
	577	-78		963	-954
	-528	-524		-208	801
	-144	-772		-794	-67
	-410	442		328	58
	-52	710		-250	-828
	975	-649		-959	779
	-919	921		-850	504
	343	-122		-163	-396
23	-452	-233	48	379	-878
	382	-354		648	-747
	-762	-417		-644	959
	-720	-841		544	-946
	-859	528		-650	-688
	295	-741		539	621
	-156	-16		629	790
	29	-579		830	148
	463	501		393	-379
	-546	-682		-52	599
24	729	44	49	111	834
	317	-767		-344	861
	-246	291		-995	-746
	-859	-165		197	713
	211	484		315	-967
	270	317		874	-749
	-749	-348		-845	521
	963	-954		-496	-932
	-208	801		37	605
	-794	-67		-877	387

Tabel A.51 Data Uji dengan N = 10 (9)

25	328	58	50	917	219
	-250	-828		7	102
	-959	779		-435	-607
	-850	504		-750	958
	-163	-396		14	-245
	379	-878		557	-875
	648	-747		-412	770
	-644	959		543	-850
	544	-946		-976	-261
	-650	-688		420	-104

[*Halaman ini sengaja dikosongkan*]

LAMPIRAN B: Hasil Uji Coba Dijkstra Pada Situs E-Olymp sebanyak 10 Kali

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

6340329	Dec 13, 2019, 7:12:11 AM	Judge C++	106.28 ms	10567	✓ Accepted
6340328	Dec 13, 2019, 7:12:06 AM	Judge C++	81.25 ms	10553	✓ Accepted
6340327	Dec 13, 2019, 7:12:01 AM	Judge C++	98.35 ms	10555	✓ Accepted
6340326	Dec 13, 2019, 7:11:52 AM	Judge C++	104.47 ms	10585	✓ Accepted
6340324	Dec 13, 2019, 7:11:42 AM	Judge C++	83.61 ms	10556	✓ Accepted
6340323	Dec 13, 2019, 7:10:54 AM	Judge C++	99.67 ms	10559	✓ Accepted
6340321	Dec 13, 2019, 7:10:33 AM	Judge C++	106.44 ms	10568	✓ Accepted
6340320	Dec 13, 2019, 7:10:23 AM	Judge C++	84.51 ms	10556	✓ Accepted
6340319	Dec 13, 2019, 7:09:59 AM	Judge C++	111.99 ms	10558	✓ Accepted
6340248	Dec 13, 2019, 6:10:48 AM	Judge C++	91.68 ms	10572	✓ Accepted

Gambar B.1 Hasil Pengumpulan Kode Program Utama

[*Halaman ini sengaja dikosongkan*]

LAMPIRAN C: Hasil Uji Coba Shifting Evaluation Values Pada Situs SPOJ sebanyak 10 Kali

Berikut merupakan lampiran hasil uji coba kode program dengan algoritma *Shortest Path Faster Algorithm* sebanyak 10 kali.

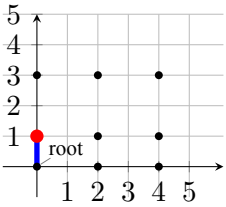
6340359	Dec 13, 2019, 7:21:50 AM	Judge C++	42.11 ms	4265	✓ Accepted
6340355	Dec 13, 2019, 7:21:09 AM	Judge C++	39.07 ms	4266	✓ Accepted
6340354	Dec 13, 2019, 7:21:03 AM	Judge C++	50.69 ms	4288	✓ Accepted
6340353	Dec 13, 2019, 7:20:57 AM	Judge C++	66.48 ms	4270	✓ Accepted
6340351	Dec 13, 2019, 7:20:51 AM	Judge C++	42.03 ms	4259	✓ Accepted
6340350	Dec 13, 2019, 7:20:44 AM	Judge C++	59.97 ms	4291	✓ Accepted
6340349	Dec 13, 2019, 7:20:33 AM	Judge C++	66.04 ms	4269	✓ Accepted
6340348	Dec 13, 2019, 7:20:27 AM	Judge C++	38.27 ms	4281	✓ Accepted
6340347	Dec 13, 2019, 7:20:22 AM	Judge C++	43.80 ms	4262	✓ Accepted
6340345	Dec 13, 2019, 7:20:15 AM	Judge C++	61.45 ms	4266	✓ Accepted

Gambar C.1 Hasil Pengumpulan Kode Program Utama

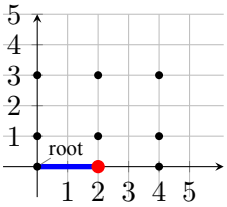
[*Halaman ini sengaja dikosongkan*]

LAMPIRAN D: Contoh Konfigurasi Initial Steiner Tree

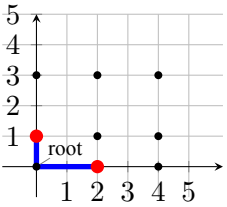
$MST_{0,0,[(0,1)]}$



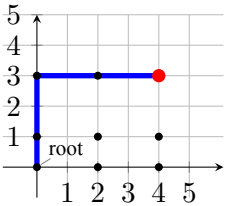
$MST_{0,0,[(2,0)]}$



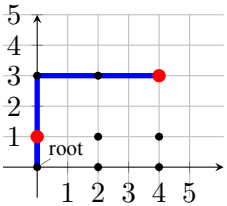
$MST_{0,0,[(0,1),(2,0)]}$



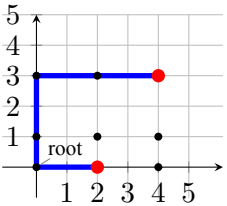
$MST_{0,0,[(4,3)]}$



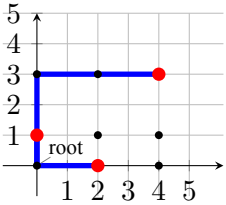
$MST_{0,0,[(0,1),(4,3)]}$



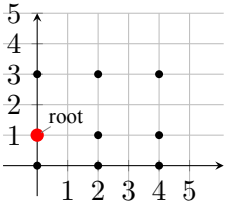
$MST_{0,0,[(2,0),(4,3)]}$



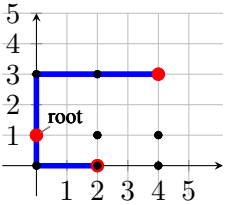
$MST_{0,0,[(0,1),(2,0),(4,3)]}$



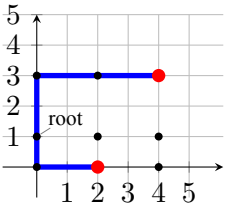
$MST_{0,1,[(0,1)]}$



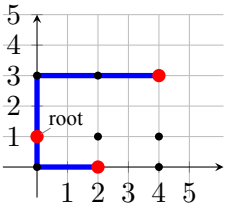
$MST_{0,1,[(2,0)]}$



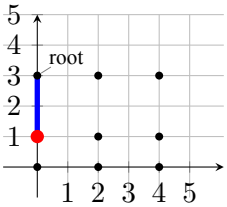
$MST_{0,1,[(2,0),(4,3)]}$



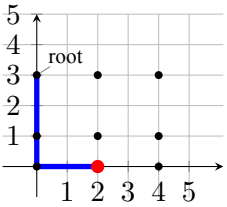
$MST_{0,1,[(0,1),(2,0),(4,3)]}$



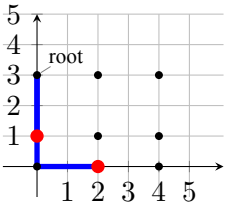
$MST_{0,3,[(0,1)]}$



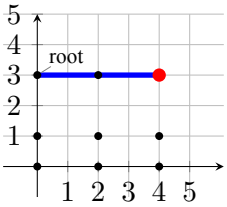
$MST_{0,3,[(2,0)]}$



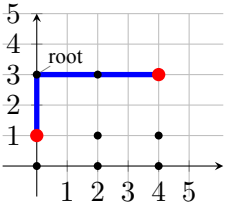
$MST_{0,3,[(0,1),(2,0)]}$



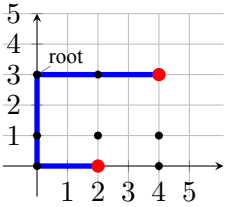
$MST_{0,3,[(4,3)]}$



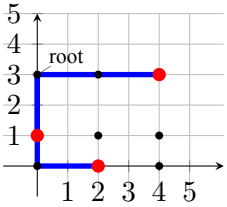
$MST_{0,3,[(0,1),(4,3)]}$



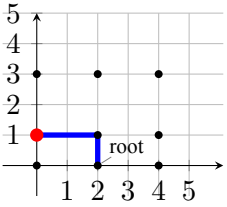
$MST_{0,3,[(2,0),(4,3)]}$



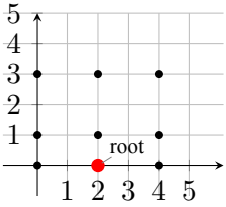
$MST_{0,3,[(0,1),(2,0),(4,3)]}$



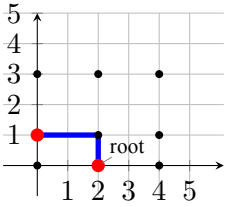
$MST_{2,0,[(0,1)]}$



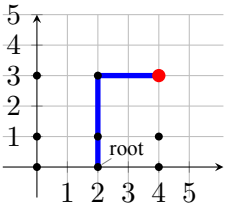
$MST_{2,0,[(2,0)]}$



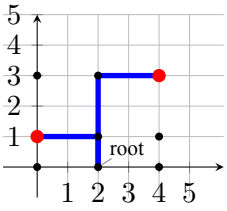
$MST_{2,0,[(0,1),(2,0)]}$



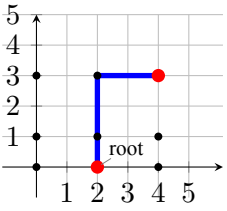
$MST_{2,0,[(4,3)]}$



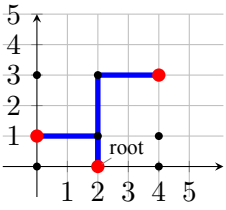
$MST_{2,0,[(0,1),(4,3)]}$



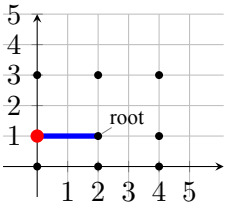
$MST_{2,0,[(2,0),(4,3)]}$



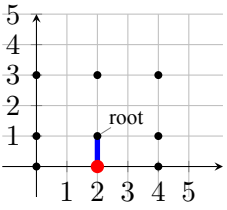
$MST_{2,0,[(0,1),(2,0),(4,3)]}$



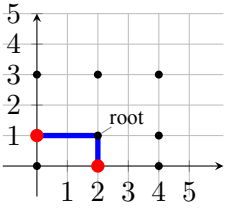
$MST_{2,1,[(0,1)]}$



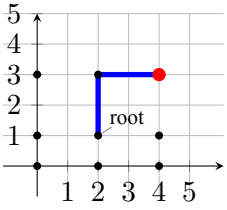
$MST_{2,1,[(2,0)]}$



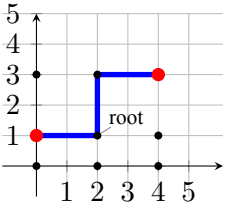
$MST_{2,1,[(0,1),(2,0)]}$



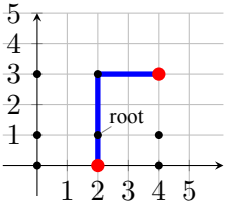
$MST_{2,1,[(4,3)]}$



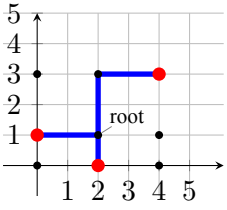
$MST_{2,1,[(0,1),(4,3)]}$



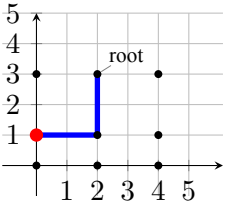
$MST_{2,1,[(2,0),(4,3)]}$



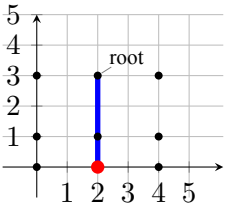
$MST_{2,1,[(0,1),(2,0),(4,3)]}$



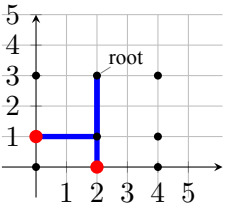
$MST_{2,3,[(0,1)]}$



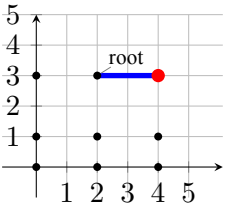
$MST_{2,3,[(2,0)]}$



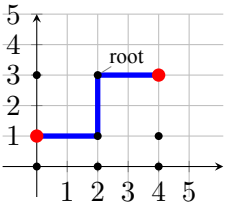
$MST_{2,3,[(0,1),(2,0)]}$



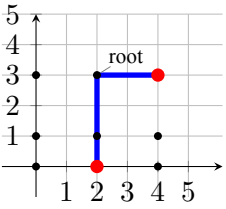
$MST_{2,3,[(4,3)]}$



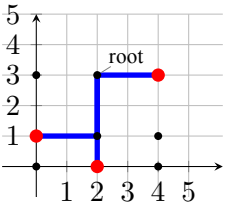
$MST_{2,3,[(0,1),(4,3)]}$



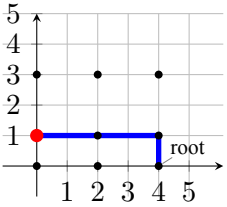
$MST_{2,3,[(2,0),(4,3)]}$



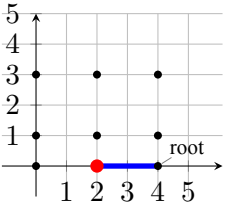
$MST_{2,3,[(0,1),(2,0),(4,3)]}$



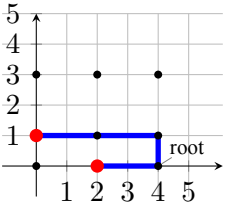
$MST_{4,0,[(0,1)]}$



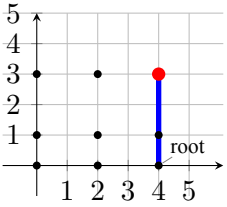
$MST_{4,0,[(2,0)]}$



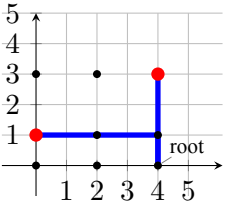
$MST_{4,0,[(0,1),(2,0)]}$



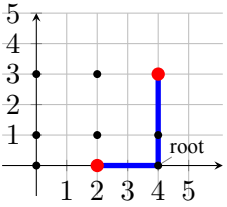
$MST_{4,0,[(4,3)]}$



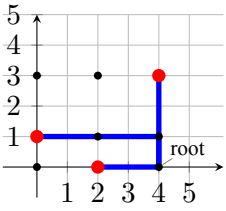
$MST_{4,0,[(0,1),(4,3)]}$



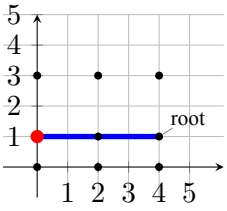
$MST_{4,0,[(2,0),(4,3)]}$



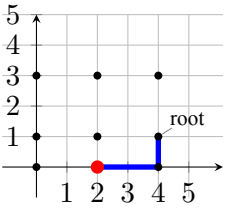
$MST_{4,0,[(0,1),(2,0),(4,3)]}$



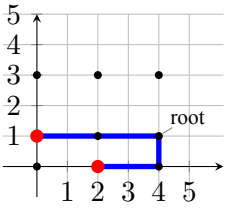
$MST_{4,1,[(0,1)]}$



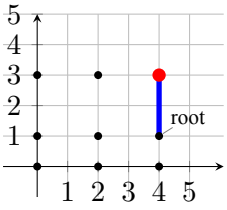
$MST_{4,1,[(2,0)]}$



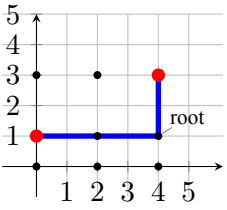
$MST_{4,1,[(0,1),(2,0)]}$



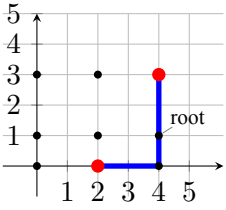
$MST_{4,1,[(4,3)]}$



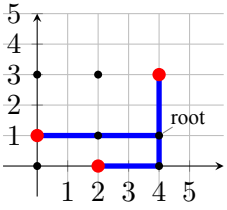
$MST_{4,1,[(0,1),(4,3)]}$



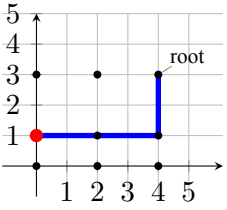
$MST_{4,1,[(2,0),(4,3)]}$



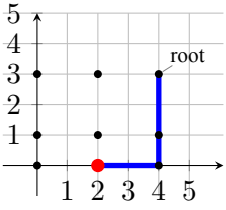
$MST_{4,1,[(0,1),(2,0),(4,3)]}$



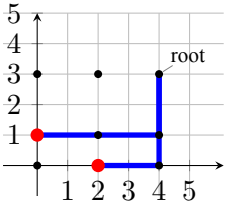
$MST_{4,3,[(0,1)]}$



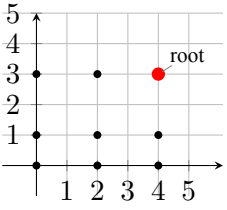
$MST_{4,3,[(2,0)]}$



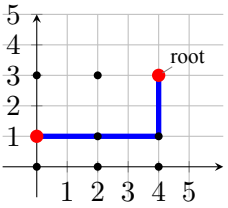
$MST_{4,3,[(0,1),(2,0)]}$



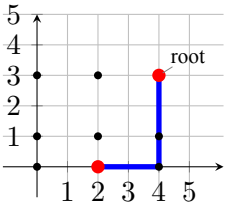
$MST_{4,3,[(4,3)]}$



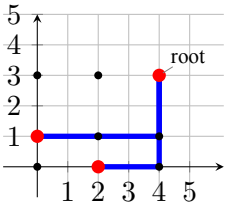
$MST_{4,3,[(0,1),(4,3)]}$



$MST_{4,3,[(2,0),(4,3)]}$



$MST_{4,3,[(0,1),(2,0),(4,3)]}$



BIODATA PENULIS

Penulis bernama Nurlita Dhuha Fatmawati, putri kedua dari dua bersaudara yang lahir pada tanggal 17 April 1998 di Salatiga. Penulis telah mengenyam pendidikan di Sekolah Dasar Islam Al-Azhar 22 Salatiga pada tahun 2004 hingga 2010, Sekolah Menengah Pertama Al-Azhar 18 Salatiga pada tahun 2010 hingga 2013, dan Sekolah Menengah Atas Negeri 1 Salatiga pada tahun 2013 hingga 2016. Pada masa penulisan, penulis sedang menempuh masa studi S1 di Institut Teknologi Sepuluh Nopember, Surabaya di Departemen Teknik Informatika.

Selama masa studi, penulis memiliki ketertarikan mengenai *Competitive Programming* yang mendorong penulis untuk aktif mengikuti berbagai kompetisi dan menjadi finalis pada ACM ICPC Asia-Jakarta, Compfest, Arkavidia, dan lain-lain. Penulis juga meraih juara 1 pada kompetisi programming Hology 2018, juara 1 pada Techphoria 2018, dan juara 2 pada Techphoria 2019. Keinginan penulis dalam menambah pengalaman juga mendorong penulis menjadi asisten dosen pada mata kuliah Dasar Pemrograman, Struktur Data, dan Perancangan dan Analisis Algoritma.

Penulis juga cukup aktif di organisasi kemahasiswaan. Penulis menjadi staf Departemen Teknologi HMTC dan menjadi Koordinator NPC Schematics. Penulis dapat dihubungi melalui surel nurlitadf17@gmail.com