PERBANDINGAN ALGORITMA KRUSKAL DENGAN ALGORITMA GENETIKA DALAM PENYELESAIAN MASALAH *MINIMUM SPANNING TREE* (MST)

SKRIPSI

Oleh :

RISKA

G1A006049

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS BENGKULU

2014

PERBANDINGAN ALGORITMA KRUSKAL DENGAN ALGORITMA GENETIKA DALAM PENYELESAIAN MASALAH *MINIMUM SPANNING TREE* (MST)

SKRIPSI

Diajukan untuk Memenuhi Persyaratan dalam Menyelesaikan
Pendidikan Strata Satu (S1) pada Program Studi Teknik Informatika
Fakultas Teknik Universitas Bengkulu

Oleh :

RISKA
G1A006049

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS BENGKULU
2014

HALAMAN MOTTO DAN PERSEMBAHAN

Motto :

Allah tidak membebani seseorang melainkan sesuai dengan kesanggupannya. (QS. Al Baqarah:286.)

Berusaha dan berdoa adalah langkah terbaik dalam mencapai suatu keberhasilan. Berusaha tanpa berdoa adalah kesombongan. Berdoa tanpa berusaha adalah mustahil.

Skripsi adalah harga mati, namun jangan mati karena skripsi

Persembahan :

AlhamdulillahiRabbil'alamiin…,
Akhirnya usai sudah perjuangan panjangku untuk menyelesaikan karya ini yang kupersembahkan khusus kepada:

- ♥ Ibu dan Apa (Hj. Mayarnis, S.Pd & Irisman(Alm))
- ♥ Apak (Markondi Can Idris)
- ♥ Adik-adik Ku : Risko, Riske & Riski
- ♥ Keluarga Besar di Solok dan Pasilihan
- ♥ Almameterku

KATA PENGANTAR

Assalamu'alaikum Wr.Wb

Alhamdulillah, segala puji bagi Allah SWT atas segala rahmat, hidayah dan inayah-Nya, sehingga penulis dapat menyelesaikan skripsi dengan judul "**Perbandingan Algoritma Kruskal dengan Algoritma Genetika dalam Penyelesaian Masalah *Minimum Spanning Tree* (MST)"** ini dengan baik**.** Sholawat serta salam juga dipanjatkan kepada Nabi Muhammad SAW beserta para kerabat dan sahabat-sahabatnya.

Laporan skripsi ini disusun sebagai salah satu syarat guna memperoleh gelar Sarjana Teknik Universitas Bengkulu dan sekaligus sebagai sarana untuk mengimplementasikan secara langsung ilmu dan teori yang telah diperoleh selama menjalani masa studi di Program Studi Teknik Informatika Fakultas Teknik Universitas Bengkulu.

Keberhasilan penyusunan skripsi ini tidak lepas dari bimbingan, dukungan dan bantuan dari berbagai pihak. Oleh karena itu dalam kesempatan ini penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada :

1. Bapak Khairul Amri, S.T., M.T selaku Dekan Fakultas Teknik Universitas Bengkulu.
2. Ibu Desi Andreswari, S.T., M.Cs selaku Ketua Program Studi Teknik Informatika Universitas Bengkulu
3. Bapak Rusdi Efendi, S.T., M.Kom dan Ibu Dr. Diyah Puspitaningrum,S.T., M.Kom selaku Dosen Pembimbing yang bersedia meluangkan waktu untuk memberikan bimbingan, arahan serta masukan kepada penulis dalam penulisan skripsi ini.
4. Bapak Funny Farady C. S.Kom.,M.T dan Bapak Aan Erlansari,S.T.,M.Eng selaku Dosen Penguji yang telah memberikan masukan dan arahan kepada penulis dalam penyelesaian skripsi ini.
5. Bapak dan Ibu Dosen pengajar di Program Studi Teknik Informatika yang telah memberikan ilmu yang berharga bagi penulis selama masa perkuliahan.

6. Staf-staf Administrasi di Fakultas Teknik khususnya staf Program Studi Teknik Informatika, Mbak Fenty yang telah membantu penulis dalam mengurus kelengkapan administrasi selama masa perkuliahan.
7. Sahabat-sahabatku, Rona Yuridia, Fadlilah S,T, Rahmadi S.T, Putri Asyura S.T, Siti Maulinda S.T, yang selalu memberikan dorongan semangat dan dukungan disaat suka dan duka sehingga penulis tetap bertahan menyelesaikan skripsi ini.
8. Semua Teman-teman seperjuangan Teknik Informatika, terutama Yosi, Riza, Rika, Budiman, Sakti, Hadi, *never give up guys!*
9. Pihak-pihak lain yang telah memberikan bantuan dan dorongan atas terciptanya karya ini.

Penulis menyadari bahwa penyusunan skripsi ini masih jauh dari kesempurnaan, maka dengan segala keterbukaan penulis mengharapkan segala kritik dan saran yang membantu proses penyempurnaan di masa mendatang. Akhir kata penulis mengharapkan semoga skripsi ini dapat bermanfaat bagi penulis maupun pembaca.

Wassalamu'alaikum Wr.Wb.

Bengkulu, Juni 2014

Penulis

PERBANDINGAN ALGORITMA KRUSKAL DENGAN ALGORITMA GENETIKA DALAM PENYELESAIAN MASALAH *MINIMUM SPANNING TREE* (MST)

Oleh :
RISKA
G1A006049

ABSTRAK

Penelitian ini bertujuan untuk membangun suatu sistem aplikasi penyelesaian masalah *Minimum Spanning Tree* dengan menggunakan Algoritma Kruskal dan Algoritma Genetika. Permasalahan pada *Minimum Spanning Tree* adalah bagaimana menghitung jarak minimum pada sebuah graf lengkap dimana semua titik simpul terhubung dan *edge* yang terpilih tidak membentuk sirkuit. Sistem aplikasi ini dibangun dengan menggunakan pemrograman Visual Basic 6.0 dan database MySQL. Hasil keseluruhan proses pada sistem aplikasi *Minimum Spanning Tree* adalah jarak minimum yang dihitung dengan menggunakan Algoritma Kruskal dan Algoritma Genetika. Hasil yang ditampilkan berupa teks dan visualisasi gafik yang menunjukkan *tree* minimum dari sebuah graf. Secara umum Algoritma Kruskal menunjukkan hasil yang lebih baik dari Algoritma Genetika dengan memperhatikan parameter jarak minimum yang dihasilkan dan waktu proses algoritma. Untuk data 5-25 simpul Algoritma Kruskal menghasilkan nilai jarak minimum lebih baik daripada Algoritma Genetika sampai dengan 50% dan waktu proses algoritma 50 kali lebih cepat.

Kata kunci:
***Minimum Spanning Tree,* MST, Kruskal, Genetika**

THE COMPARATION OF KRUSKAL ALGORITHM VERSUS GENETIC ALGORITHM IN SOLVING MINIMUM SPANNING TREE PROBLEM

By :
RISKA
G1A006049

ABSTRACT

This research aims to develop a problem solving system of Minimum Spanning Tree using Kruskal Algorithm and Genetic Algorithm. The problem of Minimum Spanning Tree is how to calculate minimum distance in a complete graph where each nodes are connected and the selected edge should not make sircuit. This application system is built using Visual Basic 6.0 programming and MySQL database. The result of the whole process of this Minimum Spanning Tree aplication system is minimum distance which is calculated using Kruskal Algorithm and Gentic Algorithm. The displayed result are both text and visualization graph which show the minimum tree of a graph. Kruskal Algorithm generally shows better result than Genetic Algorithm with minimum distance resulted and running time as the parameters. For 5-25 nodes, Kruskal Algorithm generate minimum distance up to 50% and running time process 50 times faster than Genetic Algorithm.

Keyword:
Minimum Spanning Tree, MST, Kruskal, Genetic

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1. 1. Latar Belakang

Seperti yang diketahui bahwa perkembangan teknologi informasi dalam beberapa dekade terakhir ini sangatlah pesat. Perkembangan teknologi informasi tidak hanya memberikan kemudahan dalam pengaksesan informasi, tetapi juga penggunaan komputer sebagai alat bantu mutlak dalam pengolahan data untuk memperoleh hasil yang cepat dan akurat. Salah satu penggunaan teknologi komputer yeng berkembang saat ini adalah untuk pengimplementasian teori graf sebagai alat bantu pengambilan keputusan, pencarian rute terpendek dan kecepatan waktu tempuh atau perhitungan biaya minimum untuk menghasilkan keuntungan terbesar.

Teori Graf adalah salah satu cabang ilmu matematika yang sudah ada sejak lama dan memiliki segi terapan di banyak bidang ilmu pengetahuan dan teknologi informasi. Dalam Teori Graf, graf linier didefinisikan sebagai graf G(*V,E)* yang terdiri dari himpunan obyek V = {v_1, v_2, ...} yang disebut *verteks* (simpul) dan himpunan garis E = {e_1, e_2, ...} yang disebut *edge*. Graf memiliki banyak konsep, salah satu diantaranya adalah konsep pohon *(tree)*. Pemilihan konsep pohon sebagai salah satu konsep terapan graf disebabkan karena konsep pohon merupakan konsep yang sangat dekat dengan kehidupan nyata. Secara tidak disadari, manusia banyak yang menggunakan pohon sebagai pemodelan berbagai hal dalam kehidupan sehari-hari. Teori graf mempunyai penerapan yang luas. Beberapa contoh topik yang dikembangkan dan diselesaikan dalam teori graf

bentuk pohon adalah seperti kasus Pohon Keputusan (*Decision Tree*), *Travelling Salesman Problem* (TSP), Penjadwalan dan juga *Minimum Spanning Tree* (MST).

Minimum spanning tree (MST) merupakan sebuah permasalahan dalam suatu graf yang mana banyak aplikasinya baik secara langsung maupun tidak langsung telah dipelajari. Masalah *Minimum spanning tree* hampir serupa dengan masalah rute terpendek (*shortest route*), kecuali bahwa tujuannya adalah untuk menghubungkan seluruh simpul dalam jaringan sehingga total panjang cabang tersebut diminimisasi. Jaringan yang dihasilkan merentangkan (menghubungkan) semua titik dalam jaringan tersebut pada total jarak (panjang) minimum.

Konsep MST diterapkan pada graf berbobot dan tak berarah. Tujuan dari konsep MST adalah menemukan jalur terpendek yang menghubungkan semua titik (*verteks*) yang terdapat pada sebuah graf. Dari (Sholihah, 2013) dijelaskan, definisi MST adalah bobot dari suatu pohon perentang T dalam suatu graf tersambung berbobot G adalah jumlah sisi bobot T. Pohon perentang minimum adalah pohon perentang dari G dengan bobot minimum.

Dalam kehidupan sehari-hari, banyak permasalahan optimasi yang dapat diselesaikan menggunakan *Minimum spanning tree.* Beberapa masalah tersebut antara lain pencarian jarak terpendek, biaya termurah, dan kebutuhan tenaga dalam pembangunan jalan, pemasangan jaringan kabel telepon, dan kabel jaringan listrik. Penyelesaian masalah-masalah ini pada dasarnya adalah menentukan semua *spanning tree* yang mungkin dibuat dan memperhitungkan *weight* (bobot) yang terkecil.

Permasalahan *Minimum spanning tree* sederhana mungkin bisa diselesaikan dengan melakukan perhitungan manual. Namun untuk kasus

spanning tree yang besar dan kompleks, perhitungan manual akan sulit dilakukan karena akan memakan waktu yang lama. Oleh sebab itu dibutuhkan satu program aplikasi komputer yang dapat melakukan perhitungan nilai minumum spanning tree dengan cepat dan akurat.

Algoritma *Greedy* merupakan metode paling populer untuk memecahkan persoalan optimasi. Algoritma *greedy* adalah algoritma yang memecahkan masalah langkah perlangkah dimana pada setiap langkah dibuat pilihan optimum (*local optimum*) dengan harapan bahwa langkah berikutnya mengarah ke solusi optimum global (*global optimum*). Algoritma *Greedy* yang umum digunakan untuk permasalahan *minimum spanning tree* adalah Algoritma Kruskal.

Dari penelitian yang telah dilakukan sebelumnya, Algoritma Kruskal terbukti memberikan hasil minimun yang cukup baik untuk permasalahan *Minumum Spanning Tree*. Namun penelitian tersebut hanya berpusat pada satu algoritma itu saja tanpa membandingkannya dengan algoritma lain. Untuk itu pada penelitian ini peneliti mencoba melakukan suatu komparasi (perbandingan) dengan algoritma lain yaitu Algoritma Genetika.

Algoritma Genetika adalah algoritma pencarian heuristik yang didasarkan atas mekanisme evolusi biologis. Keberagaman pada evolusi biologis adalah variasi dari kromosom antar individu organisme. Variasi kromosom ini akan mempengaruhi laju reproduksi dan tingkat kemampuan organisme untuk tetap hidup (Kusumadewi, 2003) Algoritma Genetika menggunakan prinsip pencarian solusi neighborhood berdasarkan mekanisme seleksi alam (*natural selection*) dan genetika alam (*natural genetics*) yang dapat digunakan untuk memecahkan masalah optimasi seperti permasalahan *Minimum spanning tree*.

Berdasarkan hal diatas, pada penelitian ini peneliti akan membuat suatu perangkat lunak yang dapat melakukan perbandingan terhadap Algoritma Kruskal dan Aloritma Genetika dari sudut pandang hasil jarak minimum dan waktu komputasi dalam permasalahan *Minimum Spanning Tree*.

1. 2. Rumusan Masalah

Berdasarkan latar belakang yang telah dikemukakan diatas, maka rumusan masalah yang dibahas pada penelitian ini adalah :

1. Bagaimana membangun satu aplikasi M*inimum Spanning Tree* menggunakan Algoritma Kruskal dan Algoritma Genetika.
2. Bagaimana hasil perhitungan Algoritma Kruskal dan Algoritma Genetika pada permasalahan *Minimum Spanning Tree*
3. Algoritma manakah yang lebih efektif diantara Algoritma Kruskal dan Algoritma Genetika dalam menentukan M*inimum Spanning Tree*.

1. 3. Batasan Masalah

Adapun yang menjadi batasan masalah dalam penelitian adalah :

1. Konsep Graf yang disajikan adalah graf lengkap dimana semua titik tiap graf terhubung (*Connected Graph*), graf tidak berarah (*Undirected Graph*) dan graph berbobot (*Weighted Graph*)
2. Nilai koordinat yang di inputkan adalah bernilai bulat positif
3. Jumlah simpul yang di inputkan lebih besar atau sama dengan 3 (simpul ≥ 3)

4. Nilai parameter populasi Algoritma Genetika tidak lebih dari n^{n-2}, dimana n adalah jumlah simpul.
5. Visualisasi graf berupa simulasi tanpa skala dengan jarak sebenarnya.
6. Keoptimalan yang dibandingkan adalah keoptimalan hasil algoritma berupa lama waktu *running time* dan hasil akhir nilai minimum yang didapatkan.
7. Hasil akhir adalah berupa nilai *minimum* paling kecil yang dihasilkan (*bestcase)*

1. 4. Tujuan Penilitian

Tujuan dari penelitian ini yaitu :

1. Membangun satu aplikasi M*inimum Spanning Tree* menggunakan Algoritma Kruskal dan Algoritma Genetika.
2. Membandingkan hasil nilai minimum dan waktu proses dari Algoritma Kruskal dengan Algoritma Genetika dalam penyelesaian permasalahan *Minimum Spanning Tree*
3. Menyimpulkan algoritma manakah yang lebih baik diantara Algoritma Kruskal dan Algoritma Genetika dalam menentukan nilai minimum dari *Spanning Tree*.

1. 5. Manfaat Penelitian

Manfaat penelitian ini antara lain:

1. Mengimplementasikan dan menerapkan teori graf untuk menyelesaikan kasus *Minimum Spanning Tree* (MST) sehingga dapat

berguna bagi peneliti-peneliti yang ingin melakukan penelitian *Spanning Tree* dan *Minimum Spanning Tree* selanjutnya.

2. Menemukan hasil minimum terbaik dari Algoritma Kruskal dan Algoritma Genetika untuk permasalahan *Minumum Spanning Tree.*

BAB II

LANDASAN TEORI

2. 1. Teori Umum Graf

Dalam matematika dan ilmu komputer, teori graf adalah cabang kajian yang mempelajari sifat-sifat graf. Secara informal, suatu graf adalah himpunan benda-benda yang disebut simpul (*vertex* atau *node*) yang terhubung oleh sisi (*edge*) atau busur (*arc*). Biasanya graf digambarkan sebagai kumpulan titik-titik (melambangkan simpul) yang dihubungkan oleh garis-garis (melambangkan sisi) atau garis berpanah (melambangkan busur). Suatu sisi dapat menghubungkan suatu simpul dengan simpul yang sama. Sisi yang demikian dinamakan gelang.

Menurut (Munir, 2006) menyatakan graf *G* secara matematis sebagai pasangan himpunan (*V,E*) dimana :

V = himpunan tidak kosong dari simpul – simpul

$$v_i : \{v_1, v_2, ..., v_n\} \quad \text{.....(2.1)}$$

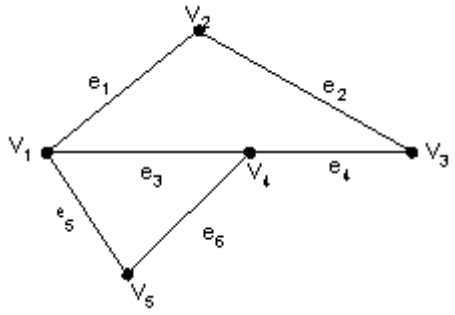
E = himpunan busur yang menghubungkan sepasang simpul

$$v_j : \{e_1, e_2, ..., e_n\} \quad \text{.....(2.2)}$$

Busur $e_i = (v_i, v_j)$

adalah pasangan simpul dengan $v_i, v_j \subset V$ .....(2.3)

dan nilai $i,j = 1,2,3,\ldots n$



Gambar 2.1 Graf G (V,E)

Sebagai contoh, graf G dalam Gambar 2.1 mempunyai titik-titik v1; v2; v3; v4; v5, sedangkan sisi-sisinya dinyatakan oleh e1 = {v1; v2}, e2 = {v2; v3}, e3 = {v1; v4}, e4 = {v4; v3}, e5 = {v1;v5}, e6 = {v5; v4}.

Dari pengertian diatas, dapat disimpulan bahwa sebuah graf *G* didefinisikan sebagai pasangan (*V,E*), dimana *V* adalah sekumpulan titik dan *E* adalah relasi biner pada *V* yang artinya *E* adalah kumpulan busur/sisi yang dapat menghubungkan titik-titik *V*.

Beberapa terminologi yang berhubungan dengan teori graf diantaranya :

1. Bertetangga (*adjacent*)

   Dua buah titik pada graf tak-berarah *G* dikatakan bertetangga jika keduanya terhubung langsung dengan sebuah busur.

2. Bersisian (*incident*)

   Untuk sembarang busur $e = (v_j,v_k)$, busur *e* dikatakan bersisian dengan titik v_j dan v_k.

3. Titik Terpencil (*isolated vertex*)

   Titik yang tidak mempunyai busur yang bersisian dengannya disebut titik terpencil.

4. Graf Kosong (*null graph atau empty graph*)

   Graf yang himpunan busurnya merupakan himpunan kosong disebut graf kosong.

5. Derajat (*degree*)

   Derajat suatu titik pada graf tak-berarah adalah jumlah busur yang bersisian dengan titik tersebut. Pada graf berarah, derajat suatu titik ialah jumlah busur yang masuk ke titik ditambah dengan jumlah busur yang keluar dari titik.

6. Lintasan (*path*)

   Lintasan yang panjangnya n dari titik awal v_0 ke titik tujuan v_n di dalam graf G ialah barisan berselang-seling titik-titik dan busur-busur yang berbentuk $v_0, e_1, v_1, e_2, v_2, \ldots, v_{n-1}, e_n, v_n$, sedemikian sehingga $e_1 = (v_0,v_1)$, $e_2 = (v_1,v_2)$, $\ldots$, $e_n = (v_{n-1},v_n)$ adalah busur-busur dari graf G. Panjang lintasan adalah jumlah busur dalam lintasan tersebut.

7. Siklus (*cycle*) atau Sirkuit (*circuit*)

   Lintasan yang berawal dan berakhir pada titik yang sama disebut sirkuit atau siklus. Panjang sirkuit adalah jumlah busur di dalam sirkuit tersebut.

8. Terhubung (*connected*)

   Graf tak-berarah G disebut graf terhubung jika untuk setiap pasang titik v_i dan v_j dalam himpunan V terdapat lintasan dari v_i ke v_j. Jika tidak, maka graf G tak terhubung.

9. *Cut-Set*

   Cut-set dari graf terhubung *G* adalah himpunan busur yang bila dibuang dari *G* menyebabkan *G* tidak terhubung. Jadi, *cut-set* selalu menghasilkan dua buah komponen terhubung. Nama lain untuk *cut-set* ialah *bridge* (jembatan).

10. Graf Berbobot (*weighted graph*)

    Sebuah graf dapat dikembangkan dengan memberi bobot pada setiap busur. Graf berbobot (*weighted graph*) adalah graf yang setiap busurnya diberi harga (bobot). Jika suatu graf melambangkan jaringan jalan, maka bobotnya dapat berarti panjang jalan maupun batas kecepatan pada batas tertentu. Jenis graf berbobot juga akan digunakan dalam penelitian ini. Dalam hal ini, bobot pada penelitian merepresentasikan jarak antartitik.

Graf dapat dikelompokkan menjadi beberapa jenis tergantung pada sudut pandang pengelompoknya. Pengelompokan graf dapat dipandang berdasarkan ada tidaknya cabang ganda atau cabang gelang, berdasarkan jumlah simpul, atau berdasarkan orientasi arah pada sisi (Munir, 2006).

1. Berdasarkan Ada Tidaknya Busur Ganda.

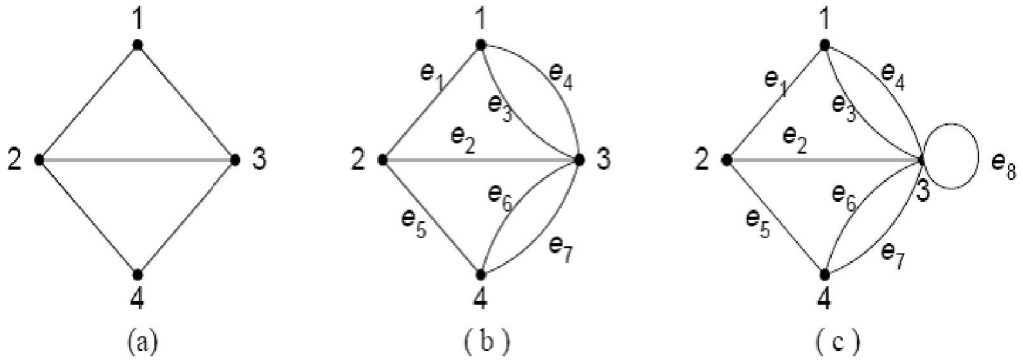
   Secara umum graf dapat digolongkan menjadi dua jenis :

   a. Graf sederhana (*simple graph*).

   Graf sederhana adalah graf yang tidak memiliki gelang maupun busur ganda. Graf sederhana dapat dilihat pada Gambar 2.2(a)

b. Graf tak-sederhana (*unsimple-graph*).

Graf tak-sederhana adalah graf yang memiliki gelang maupun busur-ganda. Ada dua macam graf tak-sederhana, yaitu graf ganda (*multigraph*), dan graf semu (*pseudograph*). Graf ganda adalah graf yang memiliki busur ganda, Contoh dapat dilihat pada Gambar 2.2(b) yaitu busur e_3 dan e_4 serta busur e_6 dan e_7, sedangkan graf semu adalah graf yang memiliki gelang (*loop*) dapat dilihat pada Gambar 2.2(c). Gelang (*loop*) pada Gambar 2.2(c) adalah busur e_8.



Gambar 2.2 Contoh graf sederhana (a) dan tak-sederhana (b)

2. Berdasarkan Jumlah Titik.

Secara umum graf dapat digolongkan menjadi dua jenis:

a. Graf berhingga (*limited graph*).

Graf berhingga adalah graf yang memiliki jumlah titik yang berhingga (dapat dihitung).

b. Graf tak-berhingga (*unlimited graph*).

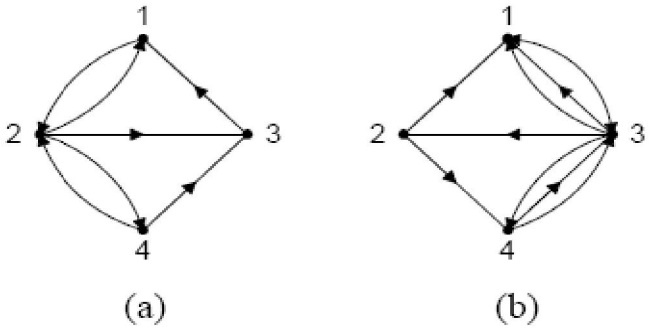
Graf tak-berhingga adalah graf yang memiliki jumlah titik yang tak-berhingga (sangat banyak).

3. Berdasarkan Orientasi Arah Pada Busur.

Secara umum graf dapat digolongkan menjadi dua jenis:

a. Graf berarah (*directed graph*).

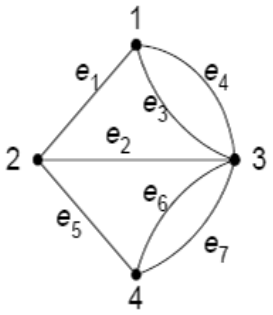
Graf berarah adalah graf yang setiap busurnya diberikan orientasi arah. Misalkan (v_j,v_k) dan (v_k,v_j) menyatakan dua busur yang berbeda, dengan kata lain $(v_j,v_k) \neq (v_k,v_j)$. Untuk busur (v_j,v_k), titik v_j disebut titik asal (*initial vertex*) dan untuk titik v_k disebut titik terminal (*terminal vertex*). Contoh graf berarah dan berarah-ganda dapat dilihat pada Gambar 2.3.



Gambar 2.3 Contoh Graf berarah (a) dan graf berarah-ganda (b)

b. Graf tak-berarah (*undirected graph*).

Graf tak-berarah adalah graf yang busurnya tidak memiliki orientasi arah. Pada graf tak-berarah, urutan pasangan titik yang dihubungkan oleh busur tidak diperhatikan, dengan kata lain: $(v_j,v_k) = (v_k,v_j)$.



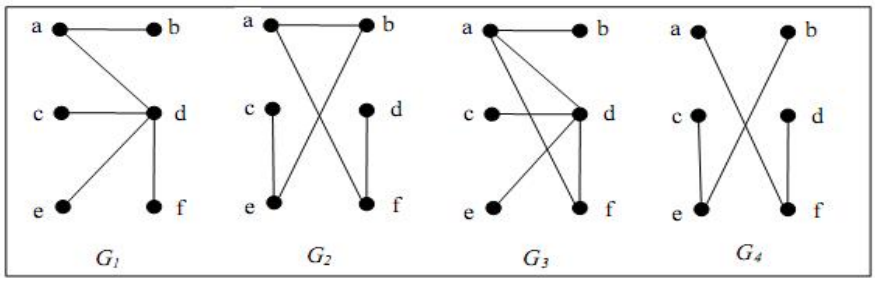
Gambar 2.4 Contoh Graf tak-berarah

2. 2. Konsep Dasar *Spanning Tree* (MST)

2.2.1 Konsep Pohon (*Tree*)

Pohon (*tree*) didefinisikan sebagai graf terhubung yang tidak mengandung sirkuit. Karena merupakan graf terhubung, maka pohon selalu terdapat jalur *(path)* yang menghubungkan setiap dua simpul dalam pohon (Ayu, 2012). Misalkan A merupakan sebuah himpunan berhingga simpul (*vertex*) pada suatu graf G yang terhubung. Untuk setiap pasangan simpul di A dapat ditentukan suatu lintasan yang menghubungkan pasangan simpul tersebut. Suatu graf terhubung yang setiap pasangan simpulnya hanya dapat dihubungkan oleh suatu lintasan tertentu, maka graf tersebut dinamakan pohon (*tree*). Dengan kata lain, pohon (*tree*) merupakan graf tak-berarah yang terhubung dan tidak memiliki sirkuit.

Menurut sejarah, teori pohon (*tree*) telah digunakan sejak tahun 1857 yaitu matematikawan Inggris Arthur cayley menerapkan teori pohon untuk menghitung jumlah senyawa kimia (Lubis, 2011). Konsep pohon merupakan salah satu konsep dari graf yang terapannya banyak digunakan baik di bidang ilmu komputer maupun bidang lain yang mengkaji pohon sebagai obyek matematika. Pada kehidupan sehari-hari tanpa disadari kita telah menerapkan konsep pohon untuk menggambarkan hirarki, misalnya hirarki silsilah keluarga, pertandingan olah raga, ataupun struktur organisasi.



Gambar 2.5 G1 dan G2 adalah contoh tree, sedangkan G3 dan G4 bukan tree

Pada Gambar 2.5, Hanya G1 dan G2 yang pohon sedangkan G3 dan G4 bukan pohon. G3 bukan pohon karena ia mengandung sirkuit a, d, f, a sedangkan G4 bukan pohon karena ia tidak terhubung (jangan tertipu dengan persilangan dua busur dalam hal ini busur (a, f) dan busur (b, e) karena titik silangnya bukan menyatakan simpul). Jadi disimpulkan, sebuah graf G dengan *n* verteks dikatakan sebuah *tree* jika :

1. G terhubung dan tak memuat sirkuit, atau
2. G terhubung dan memiliki *n – 1* edge, atau
3. G tak memuat sirkuit dan memiliki *n – 1* edge, atau
4. Terdapat tepat satu *path* diantara setiap pasangan verteks-verteks di G, atau
5. G setidaknya merupakan sebuah graf terhubung.

Pohon dinotasikan sama dengan :

T = (V,E) ……(2.4)

dimana :

T = *Tree* (pohon)
V=*Vertices* atau node atau *vertex* atau simpul, V merupakan himpunan tidak kosong; V = {v1,v2,…,vn}
E = *Edges* atau *Arcs* atau sisi yang menghubungkan simpul;
E = {e1,e2,…,en}

Istilah-istilah yang berkaitan dengan pohon (*tree*) :

1. *Tree* berakar: *tree* yang mempunyai *vertex* yang diletakkan pada bagian paling atas, dan mempunyai kesan *vertex-vertex* berikutnya terhubung ke bawah dari *vertex* ini. *Vertex* ini disebut *root* (akar).

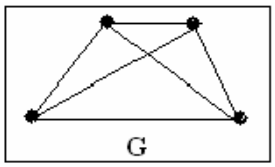
2. *Parent* (orang tua): yaitu *vertex* yang terletak lebih di atas dibanding *vertex* yang bertetangga di bawahnya, *vertex* ini harus mempunyai *child* yang merupakan *vertex* bertetangga di bawahnya.
3. *Child* (anak): *vertex* yang bertetangga dengan *parent*, terletak lebih di bawah dibanding *parent*.
8. *Right child* (anak kanan): *child* yang terletak di kanan.
9. *Left child* (anak kiri): *child* yang terletak di sebelah kiri.
10. *Right Subtree* (*sub tree* kanan): *sub tree* yang terletak di kanan.
11. *Left Subtree* (*sub tree* kiri): *sub tree* yang terletak di kiri.
12. *Sibling* (Saudara Kandung): *vertex* yang sejajar dengan *vertex* bersangkutan dengan *parent* yang sama.
13. *Ancestor* (nenek moyang): dari suatu *vertex* adalah *vertex* yang dilewati *path* dari *vertex* tersebut ke *root*, tidak termasuk *vertex* itu sendiri, tetapi *root* termasuk.
14. *Descendant* (anak cucu): semua *vertex* yang berasal dari *vertex* tersebut dan berakhir pada daun.
15. *Leaf* (daun): *vertex* yang tidak mempunyai *child*.
16. *Internal vertex* (*vertex* dalam): *vertex* yang bukan *root* dan bukan daun.

2.2.2 *Spanning Tree* (Pohon Perentang)

Spanning Tree (Pohon Perentang) adalah pohon yang merupakan subset dari suatu graf yang tidak mengandung sirkuit dimana semua simpul pada pohon merentang sama dengan simpul pada graf. Misalkan G = (V, E) adalah graf tak-berarah terhubung yang bukan pohon, yang berarti di G terdapat beberapa sirkuit.

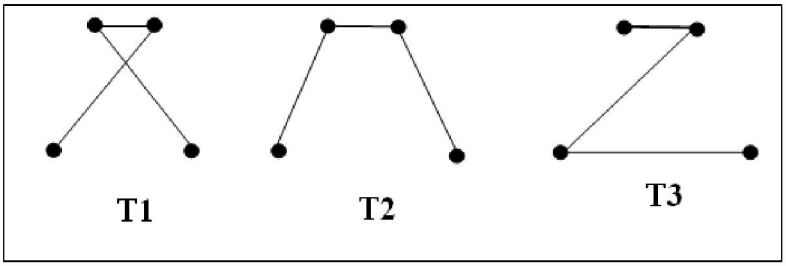
G ini dapat diubah menjadi pohon T = (V1, E1) dengan cara memutuskan sirkuit-sirkuit yang ada. Caranya mula-mula dipilih sebuah sirkuit, lalu hapus satu buah sisi dari sirkuit ini. G akan tetap terhubung dan jumlah sirkuitnya berkurang satu. Bila proses ini dilakukan berulang-ulang sampai semua sirkuit di G hilang, maka G menjadi sebuah pohon T, yang dinamakan pohon merentang (*spanning tree*). Disebut pohon merentang karena semua simpul pada pohon T sama dengan semua simpul pada graf G, dan sisi-sisi pada pohon T $\subseteq$ sisi-sisi pada graf G. Dengan kata lain V1 = V dan E1 $\subseteq$ E (Munir, 2012). Dari setiap graf terhubung mempunyai paling sedikit satu buah pohon merentang (*Spanning Tree*).

Pada satu *Spanning Tree,* sisi pada *spanning tree* T disebut dengan sebuah cabang (*branch*) dari T sedangkan sisi dari graf G yang tidak terdapat di dalam *spanning tree* disebut dengan tali (*chord*). Suatu sisi bisa saja merupakan *branch* untuk suatu *spanning tree* T tetapi merupakan *chord* untuk *spanning tree* yang lainnya.



Gambar 2.6 Suatu graf G yang bisa dijadikan pohon

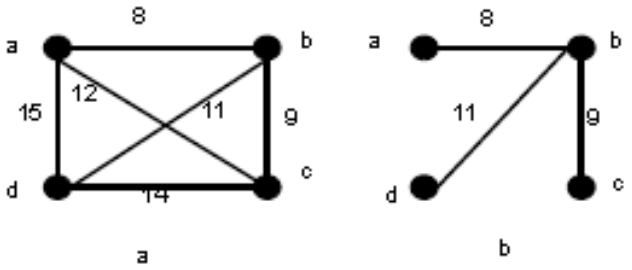
Dari gambar 2.6, sebuah graf G dapat dibangun pohon (*tree*) dengan cara menghapus sirkuit satu demi satu sehingga diperoleh *tree* seperti terlihat pada gambar 2.7 dibawah ini :



Gambar 2.7 T1,T2,dan T3 adalah solusi pohon dari graf G pada gambar 2.6

2.2.3 *Minimum Spanning Tree* (MST)

Pohon Rentangan pada suatu graf adalah subgraf minimal yang menghubungkan semua simpul pada graf, apabila graf tersebut adalah graf berbobot (*Weighted Graf*), kemudian dari pohon rentang yang dimiliki oleh graf didefinisikan sebagai penjumlahan dari bobot – bobot seluruh cabang pada pohon rentang maka akan diperoleh pohon rentang yang memiliki bobot. Pohon rentang yang memiliki bobot terkecil pada suatu graph berbobot tersebut disebut Pohon rentang minimum (*Minimum Spanning Tree*) atau dengan pengertian lain *Minimum Spanning Tree* adalah metode yang digunakan untuk menentukan *spanning tree* yang menjangkau semua titik dengan total bobot sisi minimum pada graph terhubung dan berbobot. Contoh bentuk graf lengkap dan *Minimum Spanning Tree* yang dihasilkan pada gambar 2.8 dibawah ini.



Gambar 2.8 (a) Graf berbobot dengan (b) Pohon rentangan minimunnya

Salah satu contoh aplikasi *Minimum Spanning Tree* secara langsung adalah permasalahan pemasangan jaringan dengan meminimasi jumlah penggunaan kabel/pipa untuk menghubungkan bangunan secara bersamaan (*connected*) atau pemasangan kabel jaringan telepon/listrik dengan menghitung minimalisasi biaya yang bisa digunakan dan penggunaan kabel n sependek mungkin.

Permasalahan *Minimum Spanning Tree* secara sederhana dapat diselesaikan dengan cara mudah. Dari Efendi (2003), persoalan *Spanning Tree* dapat diselesaikan dengan cara :

1. Memilih sembarang salah satu *nodes,* kemudian menghubungkan *nodes* tersebut dengan *nodes* lain yang terdekat
2. Menentukan *nodes* lain yang belum terhubung, jarak yang paling dekat dengan *nodes* yang sudah terhubung pada langkah sebelumnya, kemudian menghubungkan *nodes* ini
3. Mengulangi langkah ini sehingga seluruh *nodes* dapat terpenuhi

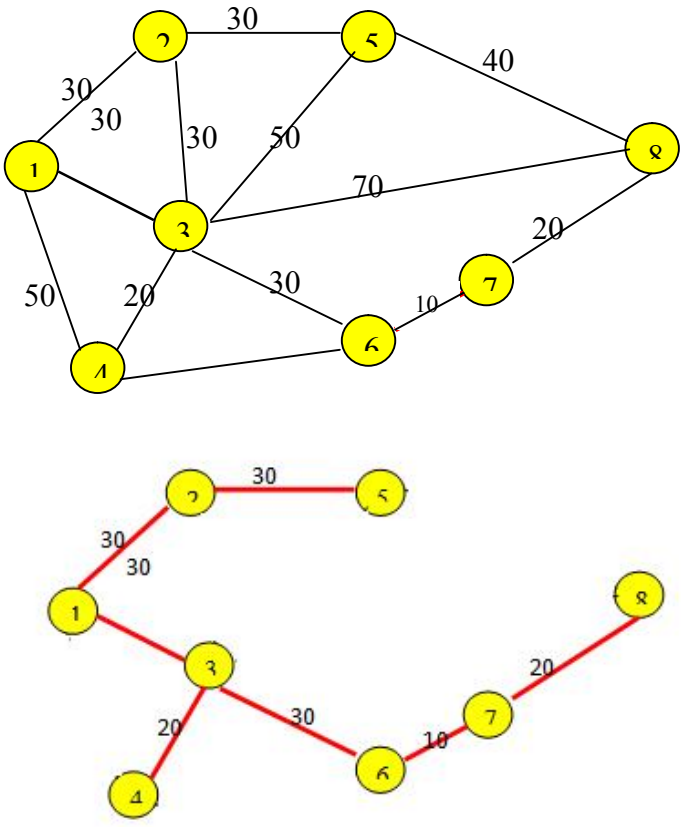
Beberapa penelitian sebelumnya telah pernah dilakukan untuk permasalahan *minimum spanning tree* adalah pengimplementasian Algoritma Prim, Algoritma Kruskal, dan algoritma dengan prinsip Greedy atau beberapa algoritma optimasi untuk penentuan nilai minimum pada *Minimum Spanning Tree* (MST).

Pada Algoritma Prim dilakukan dengan cara mencari sisi dengan bobot *minimum* sebagai langkah awal, kemudian dilanjutkan dengan mencari sisi tetangganya yang bernilai minimum namun tidak membuat *loop* (sirkuit).

Langkah kedua dilakukan terus hingga semua ruas setelah terhubung sehingga ditemukan *Minimum Spanning Tree* (*MST*) nya. Algoritma Prim dapat menyelesaikan permasalahan Minimum Spanning Tree (*MST*) secara cepat, terutama pada graf yang cukup besar dibandingkan dengan cara manual.

Contoh permasalahan *minimum spanning tree* adalah : Dimana dalam suatu propinsi, ada 8 kota (v_1, v_2, …, v_8) yang akan dihubungkan dengan jaringan listrik yang dibutuhkkan. Dari penelitian Suhendro (2008), di lakukan penyelesaian menggunakan algoritma Prim, dan didapatkan nilai minimum dari graf adalah 160.

Visualisasi data kota dan jarak antar kota dapat dilihat pada gambar 2.9 berikut ini :



Gambar 2.9 (a). Graf Awal perencanaan Jaringan Listrik (b). Hasil Minimum Spanning Tree dari graf perencanaan jaringan listrik

3. 3. Algoritma Kruskal

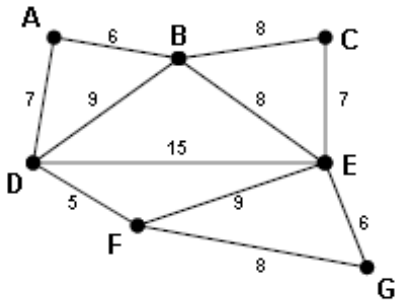
Algoritma Kruskal adalah juga tergolong algoritma *Greedy* dalam teori graf yang digunakan untuk mencari *minimum spanning tree*. Algoritma ini pertama kali muncul pada tahun 1956 dalam sebuah tulisan yang ditulis oleh Joseph Kruskal. Algoritma kruskal adalah sebuah algoritma dalam teori graf yang mencari sebuah *minimum spanning tree* untuk sebuah graf berbobot yang terhubung.

Pada Algoritma Kruskal, sisi (edge) dari graf diurut terlebih dahulu berdasarkan bobotnya dari kecil ke besar. Sisi yang dimasukkan ke dalam himpunan T adalah sisi graph G yang sedemikian sehingga T adalah Tree (pohon). Sisi dari Graph G ditambahkan ke T jika ia tidak membentuk cycle.

Secara Umum Algoritma Kruskal ditulis :

1. T masih kosong
2. pilih sisi (i,j) dengan bobot minimum
3. pilih sisi (i,j) dengan bobot minimum berikutnya yang tidak membentuk cycle di T, tambahkan (i,j) ke T
4. Ulangi langkah 3 sebanyak (n-2) kali.
5. Total langkah (n-1) kali

Contoh penyelesaian kasus *minimun spanning tree* dengan algoritma kruskal, diketahui sebuah graf berbobot seperti gambar 2.10 dibawah ini



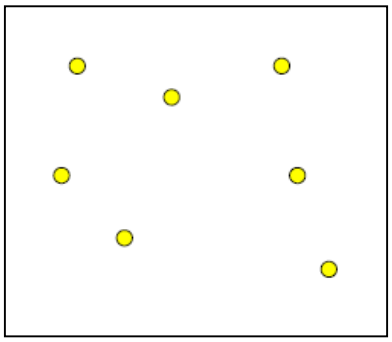
Gambar 2.10 Gambaran Awal graf

Tabel 2. 1 Daftar urut nilai Graf dari sisi terbesar sampai terkecil

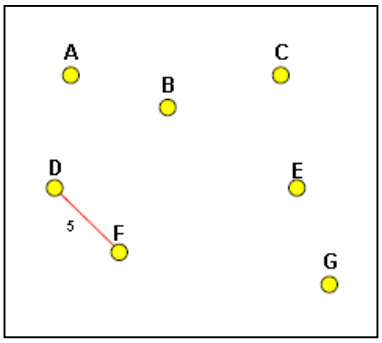
Bobot	Ruas		
15	D,E		
9	B,D	E,F	
8	B,C	B,E	F,G
7	A,D	C,E	
6	A,B	E,G	
5	D,F		

Penyelesaian :

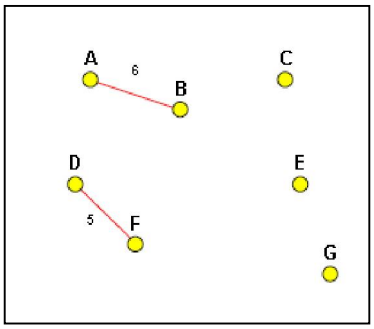
1. Mula-mula kita buat sekumpulan titik G hanya terdiri dari simpul-simpul saja.



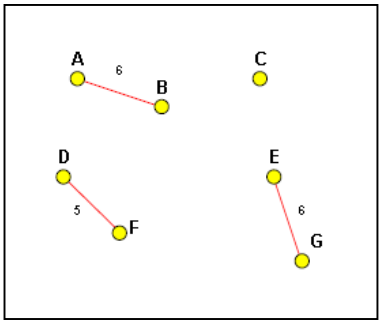
2. Urutkan Ruas dari bobot kecil ke besar **(DF, AB, EG, AD, CE, BC, BE, FG, BD, EF, DE)**, kemudian berdasarkan urutan tersebut, kita menambahkan ruas dengan mencegah terbentuknya sirkuit.



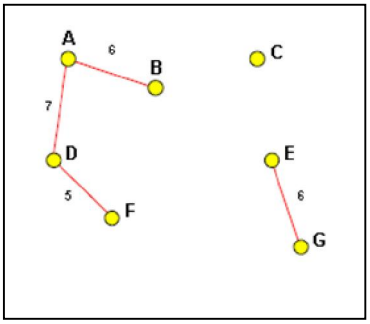
a. Pilih ruas terkecil secara acak, tambahkan ruas DF ke dalam tree



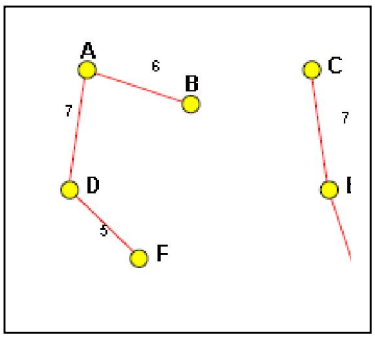
b. Penambahan ruas AB



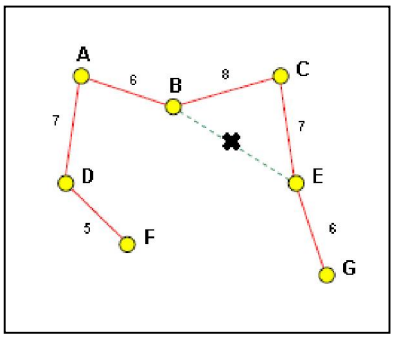
c. Penambahan ruas EG



d. Penambahan ruas AD

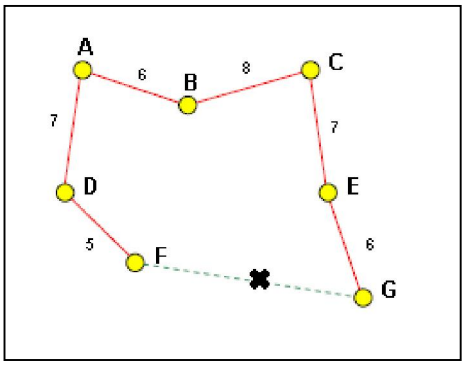


e. Penambahan ruas CE

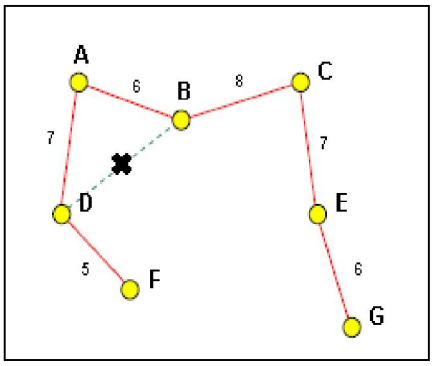


f. Penambahan Ruas BC, dan ruas BE dihapus karena membentuk sirkuit.

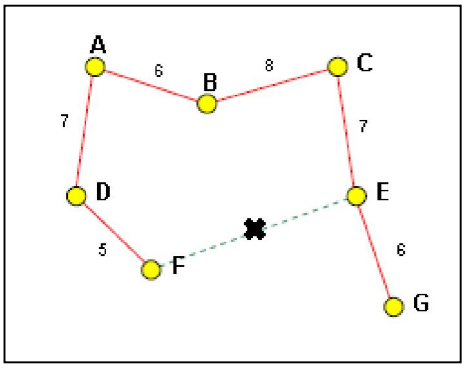
lanjutan….



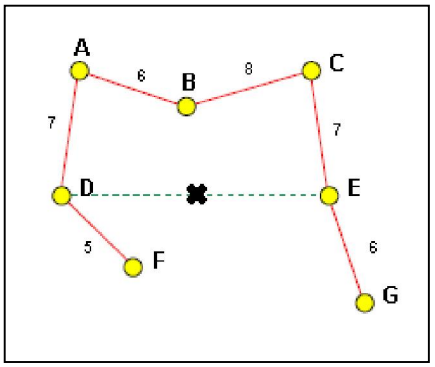
g. Ruas FG dihapus karena membentuk sirkuit.



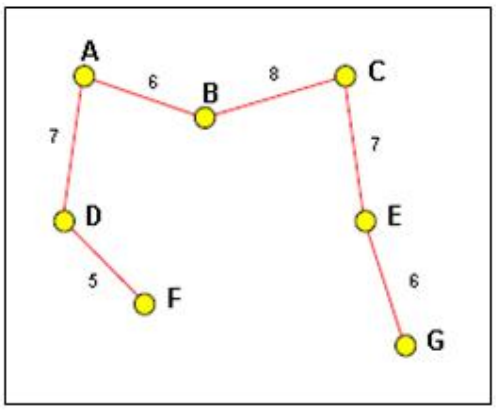
h. Ruas DB dihapus karena membentuk sirkuit.



i. Ruas EF dihapus karena membentuk sirkuit.



j. Ruas DE dihapus karena membentuk sirkuit



k. SELESAI.
MST Graf G dengan Nilai Bobot : 38

Gambar 2.11 Teknik penyelesaian Spanning Tree dengan Metode Kruskal

Pada Gambar 2.12 langkah-langkah teknik penyelesaian *Minimum Spanning Tree* adalah sebagai berikut :

a. Pilih ruas terkecil dari keseluruhan graf secara acak. Ruas DF adalah ruas terkecil dengan bobot 5 (gambar 2.11 (a))
b. Setelah ruas 1 didapatkan, ulangi lagi pemilihan ruas terkecil secara acak dari keseluruhan graf. Ruas AB dan ruas EG adalah ruas terkecil dengan bobot 6. Pilih acak ruas AB (Gambar 2.11(b))
c. Lanjutkan untuk ruas yang selanjutnya, Ruas EG adalah ruas terkecil dengan bobot 6 dan tidak membentuk sirkuit jika dihubungkan. Maka pilih ruas EG (Gambar 2.11(c))
d. Pilih ruas selanjutnya dengan bobot yg terkecil saat ini, yaitu ruas AD dan ruas CE dengan bobot 7. Pilih acak ruas AD (Gambar 2.11(d))
e. Lanjutkan untuk ruas CE adalah ruas terkecil dengan 7 dan tidak membentuk sirkuit jika dihubungkan. Maka pilih ruas CE (Gambar 2.11(e))
f. Ruas selanjutnya dengan bobot yg terkecil saat ini yaitu ruas BC, BE dan FG dengan bobot 8. Pilih acak ruas BC dan hapus ruas BE (Gambar 2.11(f)) dan ruas FG karena akan membentuk sirkuit jika dihubungkan (Gambar 2.11(g))
g. Untuk ruas-ruas selanjutnya, BD, EF,dan DE dihapus karena akan membentuk sirkut jika dihubungkan (Gambar 2.11(h), Gambar 2.11(i), Gambar 2.11(j))
h. Gambar 2.11(k) adalah *Minimum Spanning Tree* yang dihasilkan.

3. 4. Algoritma Genetika

Algoritma genetika merupakan suatu metoda pencarian yang didasarkan pada mekanisme dari seleksi dan genetika natural. Proses pencarian yang heuristik dan acak sehingga penekanan pemilihan operator yang digunakan sangat menentukan keberhasilan algoritma genetika dalam menemukan solusi *optimum* suatu masalah yang diberikan.

Dalam Kusumadewi & Purnomo (2005) dijelaskan, Pada algoritma ini, teknik pencarian dilakukan sekaligus atas sejumlah solusi yang mungkin dikenal dengan istilah populasi. Individu yang terdapat dalam satu populasi disebut dengan kromosom. Kromosom ini merupakan suatu solusi yang masih berbentuk simbol. Populasi awal dibangun secara acak, sedangkan pupulasi berikutnya merupakan evolusi kromosom-kromosom melalui iterasi yang disebut dengan generasi. Pada setiap generasi, kromosom akan melalui proses evaluasi dengan menggunakan nilai *fitnes.* Nilai *fitness* dari suatu kromosom akan menunjukkan kualitas kromosom dalam populasi tersebut. Generasi baru yang dikenal dengan istilah anak *(offsprings)* terbentuk dari gabungan 2 kromosom generasi sekarang yang bertindak sebagai induk (*parent)* dengan menggunakan operator penyilangan (*crossover).* Selain operator penyilangan, suatu kromosom dapat juga dimodifikasi dengan menggunakan operator mutasi. Populasi generasi yang baru dibentuk dengan cara menyeleksi nilai *fiitness* dari kromosom induk dan nilai *fiitness* dari kromosom anak, serta menolak kromosom-kromosom yang lainnya sehingg ukuran populasi (jumlah kromosom dalam suatu populasi) konstan.

Pada Algoritma Genetika, penyelesaian permasalahan *Minimum Spanning Tree* hampir sama dengan penyelesaian pada masalah optimasi lainnya. Perbedaannya hanyalah pada proses pengkodean kromosom (*encoding),* perhitungan bobot (*decoding*) dan rekombinasi atau persilangan (*crossover*). Selain itu pada kasus *Minimum Spanning Tree,* ditambahkan satu langkah lagi yaitu proses modifikasi derajat untuk pengecekan adanya sirkuit atau tidak.

Langkah-langkah Algoritma Genetika pada *Minimum Spanning Tree* adalah Sebagai berikut :

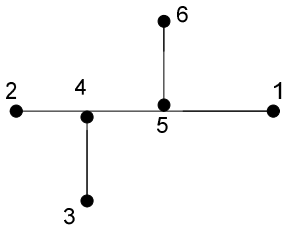
1. Pengkodean Kromosom (*Encoding*)

Dalam Algoritma genetika pengkodeaan kromoson (*Enoding*) sangat penting untuk merepresentasikan masalah awal. Pada *Minimum Spanning Tree* (MST), kromosom merupakan representasi dari phon rentangan (*Spanning Tree*) dan Gen representasi dari simpul yang membentuk *tree* tersebut. Pengkodean kromosom pada MST menggunakan teknik *decoder* kromosom bernama *Prufer Number*. Pengkodean *Prufer Number* ini bertujuan untuk mengkodekan pohon-pohon yang dihasilkan dari sebuah graf, agar masalah MST dapat diselesaikan dengan Algoritma Genetika. Langkah-langkah *Prufer Number* adalah sebagai berikut :

a. Langkah 1 : Perhatikan semua titik berderajat 1 dan pilih titik berderajat terkecil
b. Langkah 2 : Perhatikan titik yang berdekatan dengan titik yang telah kita pilih di langkah 1, dan letakkan label ini pada posisi pertama dari barisan yang akan dibentuk

c. Langkah 3 : Hapuslah titik yang dipilih pada langkah 1 beserta sisi yang berinsidensi, hingga tinggal pohon yang lebih kecil. Ulangi langkah 1-3 untuk pohon yang tertinggal. Lanjutkan hingga dua titik. Jika telah selesai, maka barisan *Prufer Number* yang dikehendaki telah terbentuk, dimana kumpulan kode simpul sebanyak *n-2* (n jumlah simpul dari graf)

Contoh :



Gambar 2. 12 Pohon T

Diberikan contoh sebuah pohon T dari Gambar 2.12 dimana akan dibentuk barisan *Prufer Number* nya. Langkah-langkah *Prufer Number* dari pohon T adalah :

- Perhatikan pohon T, simpul bernomor terkecil dan berderajat 1 yaitu simpul 1, simpul 2, Simpul 3, simpul 6.
- Sedangkan simpul-simpul yang menghubungkannya yaitu simpul 4, simpul 4, simpul 5, simpul 5.
- Sehingga *prufer number* dari pohon T adalah (5,4,4,5)

Nilai/ barisan *Prufer Number* inilah yang nantinya akan kita jadikan sebagai kromosom awal Algoritma Genetika.

2. Perhitungan Bobot (*Decoding*)

Proses *decoding* berguna untuk mengkodekan nilai-nilai gen-gen pembentuk individu atau mengubah setiap kromosom menjadi sebuah pohon, sehingga jumlah bobot dari setiap pohon dapat dihitung. Langkah-langkah *decoding Prufer Number* pada MST adalah :

a. Misalkan P adalah *Prufer Number* dari *tree* dan Q adalah himpunan simpul yang tidak terdapat pada P. Q disebut sebagai *eligeble verteks*
b. Misalkan *i* adalah digit paling kiri dari P dan *j* adalah suatu simpul dari Q dimana *j* merupakan simpul dengan nomor terkecil. Tambahkan ruas dari *i* ke *j*. Buang *j* dari Q. Jika i tidak ada lagi di P, letakkan j pada Q. Ulangi proses ini sehingga tidak ada lagi digit P yang tertinggal
c. Tambahkan 2 Ruas terakhir yang tersisa sehingga pada akhirnya membentuk pohon dengan n-1 ruas.

Jika proses decoding *decoding Prufer Number* telah selesai, selanjutnya dapat dihitung nilai fitness dan membangkitkan popolasi awal dan seleksi dari Algoritma genetika.

a. Hitung Bobot tiap kromosom terpilih.
b. Hitung nilai *fitness* (F_k) :

$$Fk = \frac{1}{Bobot} \qquad \text{.....(2.5)}$$

c. Hitung Probalilitas tiap individu (P_k)

$$Pk = \frac{Fk}{\sum Fk} \qquad \text{.....(2.6)}$$

d. Hitung *fitness* komulatif (Q_k)

$$Qk = \sum_1^k P(k) \qquad \text{.....(2.7)}$$

e. Pilih (seleksi) induk yang akan menjadi kandidat untuk di-crossover. Teknik seleksi yang digunakan adalah teknik roda *Roulette,* yaitu dengan cara :

- Bangkitkan bilangan random R_k antara 1-0 sebanyak jumlah kromosom yang dibentuk.
- Jika Q_k lebih besar dan mendekati R_k, maka pilih kromosom ke- k sebagai kandidat induk.
- Dan untuk selanjutnya akan dsilangkan (*CrossOver)*

3. Rekombinasi atau Persilangan (*CrossOver*)

Crossover (penyilangan) dilakukan atas 2 kromosom untuk menghasilkan kromosom anak (*offspring*). Metode *crossover* yang paling sering digunakan pada algoritma genetika adalah metode *crossover* satu titik (*one-point crossover*). Langkah-langkah *crossover* adalah :

a. Bangkitkan kembali nilai acak R_k sebanyak populasi yang ada.
b. Jumlah kromosom yang akan di *crossover* adalah sebanyak peluang *crossover* (Pc) yang ditentukan.

   Peluang *crossover* ditentukan oleh pengguna
c. Tentukan posisi titik potong dengan membangkitkan bilangan acak 1 sampai panjang kromosom-1 (jika panjang kromosom 5, posisi titik potong 1-4).
d. Tukarkan bagian kanan titik potong dari kedua bagian kromosom induk tersebut
e. Untuk selanjutnya kromosom dapat di mutasi

3. Mutasi

Mutasi yang digunakan pada algoritma genetika pada dasarnya akan mengubah secara acak nilai suatu bit pada posisi tertentu. Pada mutasi ini sangat dimungkinkan munculkan kromosom baru yang semula belum muncul dalam populasi awal. Langkah mutasi adalah :

a. Bangkitkan nilai acak antara 0-1
b. Tentukan Peluang Mutasi (P_m)
c. Dengan menggunakan peluang mutasi P_m, maka diharapkan sebanyak peluang mutasi dari total gen akan mengalami mutasi.

$$\text{PanjangTotal Gen} = \text{Ukuran populasi x Panjang Kromosom} \quad \ldots\ldots\ldots\ldots (2.8)$$

d. Kromosom dengan nilai acak < P_m akan dimutasi, dengan menukar gen sesuai posisi mutasi. Posisi mutasi ditentukan dengan membangkitkan bilangan acak 1 sampai panjang kromosom
e. Populasi akhir proses mutasi ini akan dijadikan populasi awal untuk generasi selanjutnya.

4. Pelestarian kromosom

Karena teknik seleksi pada algoritma genetika dilakukan secara random, sehingga ada kemungkinan kromosom yang baik tidak bisa masuk pada generasi selanjutnya karena tidak terpilih pada seleksi, maka diperlukan adanya pelestarian kromosom. Langkah-langkah pelestarian kromosom adalah :

a. Tentukan peluang kelestarian kromosom

b. Bangkitkan bilangan acak sebanyak jumlah populasi. Kromosom dengan nilai acak < peluang kelestarian akan terkena pergantian kromosom.

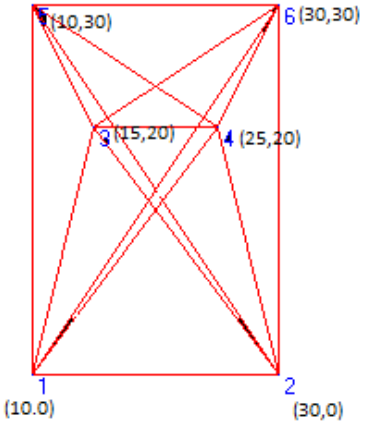
Hasil akhir kromosom yang terpilih digunakan untuk individu awal generasi selanjutnya.

Contoh Permasalahan Minimum spanning Tree dengan Algoritma Genetika

Berikut adalah contoh *Minimum Spanning Tree* menggunakan Algoritma Genetika (Astuti, 2013). Yaitu diketahui titik-titik *node* dalam sebuah *cartesius* seperti yang terlihat pada tabel 2.2 dan graf lengkap dari semua titik *node* pada Gambar 2.13 :

Tabel 2.2 Data Simpul dan Koordinat nya

***Node* ke -**	**X**	**Y**
1	10	0
2	30	0
3	15	20
4	25	20
5	10	30
6	30	30



Gambar 2. 13 Graf Lengkap Tabel 2.2

Dengan rumus jarak antar *node,* data jarak dari tabel 2.2 dihitung dengan rumus *Euclidean* :

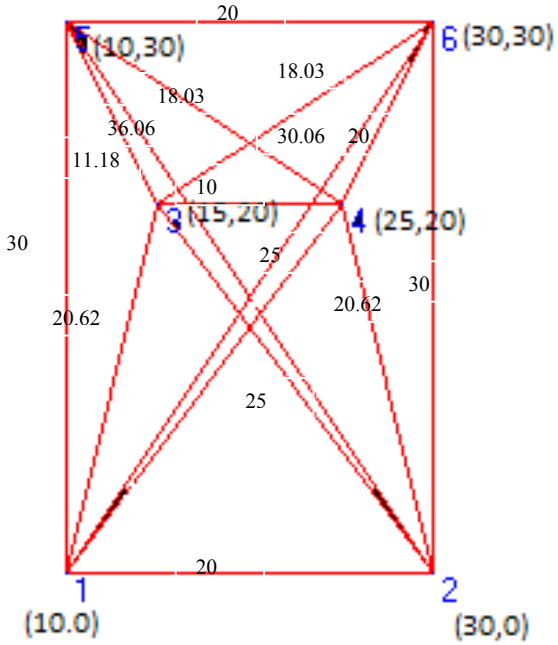
$$Ci,j = \sqrt{(Xi - Xj)^2 + (Yi - Yj)^2} \quad \text{.....(2.9)}$$

Maka jarak antar titik didapat adalah pada tabel 2.3:

Tabel 2.3 Jarak Antar Titik

	1	2	3	4	5	6
1	0.00	20.00	20.62	25.00	30.00	36.06
2	20.00	0.00	25.00	20.62	36.06	30.00
3	20.62	25.00	0.00	10.00	11.18	18.03
4	25.00	20.62	10.00	0.00	18.03	11.18
5	30.00	36.06	11.18	18.03	0.00	20.00
6	36.06	30.00	18.03	11.18	20.00	0.00

Sehingga diperoleh Graf lengkap nya adalah pada Gambar 2.14 sebagai berikut :



Gambar 2.14 Graf Lengkap dan bobot edge nya

Dengan parameter Algoritma genetika sebagai berikut :

Ukuran Populasi : 10

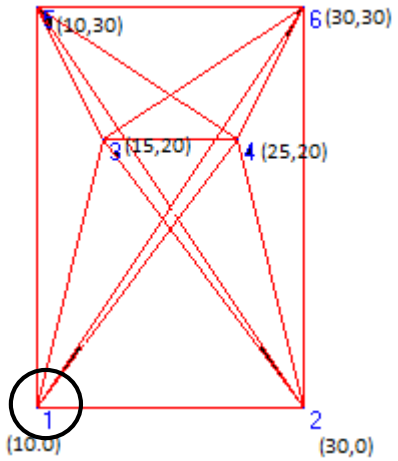
Peluang Crossover : 0.5

Peluang Mutasi : 0.1

Peluang Kelestarian : 0.1

Pengkodean Kromosom dengan teknik Prufer Number:

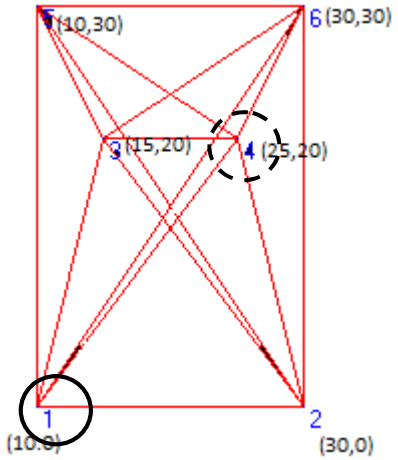
a. Perhatikan semua titik berderajat terkecil, namun karena meggunakan graf lengkap, dimana semua titik memiliki derajat yang sama, maka pilih secara acak satu titik. Proses nya dapat dilihat pada Gambar 2.15



Gambar 2. 15 Graf awal dan pilih titik 1

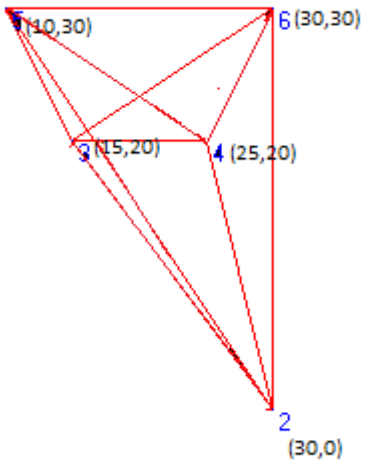
Pilih Titik 1.

b. Perhatikan titik yang berdekatan dengan titik yang telah kita pilih di langkah 1 (Titik 1), dan letakkan label ini pada posisi pertama dari barisan yang akan dibentuk, misal label 4. Gambar 2.16



Gambar 2. 16 Pilih titik 1 dari graf, letakkan pada titik 4

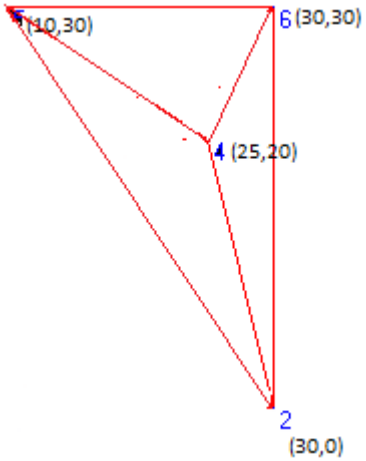
c. Hapuslah titik yang dipilih pada langkah 1 beserta sisi yang berinsidensi, hingga tinggal pohon, sehingga graf dan titik yang tersisa dapat dilihat pada gambar 2.17



Gambar 2. 17 Graf setelah di hapus titik 1 dari graf

Barisan *Prufer Number* = {4}

d. Ulangi langkah (a) sampai (c) kembali, sampai didapatkan barisan *prufer number* . Hapus titik 3, letakkan pada label 2. Graf dan titik yang tersisa pada gambar 2.18

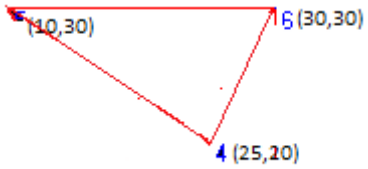


Gambar 2. 18 Graf setelah di hapus titik 3 dari graf

Barisan *Prufer Number* = {4, 2}

e. Titik tersisa, titik 2, titik 4, titik 6, dan titik 5.

Selanjutnya hapus titik 2, letakkan pada titik 6. Gambar 2.19



Gambar 2. 19 Graf setelah di hapus titik 2 dari graf

Barisan *Prufer Number* = {4, 2, 6}

f. Titik tersisa, titik 4, titik 6, dan titik 5.

Selanjutnya hapus titik 4, letakkan pada titik 5.

Gambar 2. 20 Graf setelah di hapus titik 4 dari graf

Tidak ada lagi titik yang harus dihapus pada gambar 2.20, sehingga barisan *Prufer Number* yang terbentuk *:* {4, 2, 6, 5}

Lakukan pembentukan barisan *Prufer Number* sebanyak jumlah populasi yang dibangkitkan

Bagkitkan populasi awal :

Populasi awal yang dibangkitkan pada tabel 2.4 :

Tabel 2.4 Populasi Awal dan Nilai-nilainya

	Populasi Awal	Bobot	Nilai Fitness Fk= 1/bobot	Probabilitas Pk =Fk/∑Fk	Kumulatif Qk
1	4 2 6 5	118.03	0.0085	0.0941	0.094
2	3 5 1 1	127.24	0.0079	0.0873	0.182
3	5 6 3 5	101.18	0.0099	0.1095	0.291
4	1 5 3 2	121.18	0.0083	0.0914	0.383
5	6 6 4 1	127.12	0.0079	0.0871	0.470
6	6 1 6 1	127.86	0.0078	0.0866	0.558

7	4 5 1 1	122.86	0.0081	0.0901	0.648
8	2 3 4 5	93.03	0.0107	0.1190	0.767
9	2 1 3 4	97.86	0.0102	0.1132	0.880
10	4 4 5 3	92.86	0.0108	0.1193	1.000
			∑ Fk= 0.0900		

Seleksi Kromosom :

Bangkitkan bilangan acak sebanyak populasi :
1 0.942
2 0.800
3 0.282
4 0.098
5 0.480
6 0.442
7 0.646
8 0.622
9 0.093
10 0.011

Populasi baru hasil seleksi tabel 2.5 :

Tabel 2.5 Populasi Baru Hasil seleksi

	Rk	Qk	Kromosom	Fitness	Kromosom Asal
1	0.942	1.000	4 4 5 3	0.0108	10
2	0.800	0.880	2 1 4 3	0.0102	9
3	0.282	0.291	5 6 5 3	0.0099	3
4	0.098	0.189	3 5 1 1	0.0079	2
5	0.480	0.470	6 6 4 1	0.0079	5
6	0.442	0.470	6 6 4 1	0.0079	5
7	0.646	0.648	4 5 1 1	0.0081	7
8	0.622	0.648	4 5 1 1	0.0081	7
9	0.093	0.094	4 2 6 5	0.0085	1
10	0.011	0.094	4 2 6 5	0.0085	1

Rekombinasi / persilangan (Crossover) :

Bangkitkan bilangan acak sebanyak populasi :
1 0.158
2 0.765
3 0.373
4 0.078
5 0.628
6 0.073

7 0.877
8 0.623
9 0.212
10 0.388

Kromosom ke- k yang dipilih sebagai induk jika nilai acak Rk < Pc (Pc= 0.5),

Maka Kromosom terpilih yang menjadi induk dapat dilihat pada tabel 2.6 :

Tabel 2.6 Kromosom yang terpilih untuk di Crossover

	Rk	Kromosom	Fitness	Kromosom Asal
1	0.158	4 4 5 3	0.0108	1
2	0.373	5 6 3 5	0.0099	3
3	0.078	3 5 1 1	0.0079	4
4	0.073	6 6 4 1	0.0079	6
5	0.212	4 2 6 5	0.0085	9
6	0.388	4 2 6 5	0.0085	10

Kromosom terpilih kemudian dilakukan crossover pada tabel 2.7 :

Tabel 2.7 Proses Crossover

Posisi *Crossover*	Kromosom Induk	Kromosom Anak	No Kromosom
2	4 4 5 3	4 4 3 5	1
	5 6 3 5	5 6 5 3	3
2	3 5 1 1	3 5 4 1	4
	6 6 4 1	6 6 1 1	6
3	4 2 6 5	4 2 6 5	9
	4 2 6 5	4 2 6 5	10

Mutasi :

PanjangTotal Gen = Ukuran populasi x Panjangkromosom
= 10 x 6 = 60

Bangkitkan bilangan acak [0 1]

1 0.259
2 0.481
3 0.917
4 0.780
5 0.244

6 0.825
7 0.298
8 0.024
9 0.324
10 0.565

Dengan menggunakan peluang mutasi (Pm) = 0.1, maka diharapkan 10% dari total gen yang ada akan mengalami mutasi. Kromosom dengan nilai acak < Pm akan dimutasi, dengan menukar gen sesuai posisi mutasi. Posisi mutasi dibangkitkan bilangan acak [1 6]. Proses Mutasi pada tabel 2.8 :

Tabel 2.8 Proses Mutasi

	Kromosom	*Fitness*	Posisi Mutasi	Hasil Mutasi
1	4 5 1 1	0.0081	2 dan 3	4 1 5 1

Pelestarian Kromosom :

Kromosom dengan nilai acak < peluang kelestarian (Kb=0.1) akan terkena pergantian kromosom.

Bangkitkan bilangan acak :
1 0.623
2 0.936
3 0.904
4 0.796
5 0.342
6 0.253
7 0.510
8 0.426
9 0.313
10 0.871

Karena tidak ada nilai acak yang lebih kecil dari nilai peluang kelestarian (0.1), maka tidak ada pergantian kromosom.

Hasil Populasi Akhir Generasi ke- 1

Tabel 2.9 Kromosom Akhir Generasi ke- 1

	Kromosom	Bobot	Fitness
1	4 4 3 5	86.80	0.0108
2	2 1 3 4	97.86	0.0102
3	5 6 5 3	101.18	0.0099
4	3 5 4 1	115.27	0.0079
5	6 6 4 1	127.12	0.0079
6	6 6 1 1	139.09	0.0079
7	4 5 1 1	122.86	0.0081
8	4 1 5 1	125.33	0.0081
9	4 2 6 5	118.03	0.0085
10	4 2 6 5	118.03	0.0085

Dari tabel 2.9 adalah hasil akhir dari generasi ke-1. Pada Generasi ke- 1, nilai *fitness* tertinggi dimiliki oleh kromosom 1 yaitu kromosom [1] = 4-4-3-5 dengan jumlah bobot *tree* 86.80 dan *fitness* 0.0115. Hasil ini belum merupakan hasil optimal karena belum ada generasi pembanding, sehingga proses dilanjutkan ke generasi selanjutnya hingga kritesia optimal terpenuhi atau telah mencapai batas iterasi yang di tentukan.

3. 5. Rancangan Percobaan

Pada tahap pengembangan sistem, salah satu tahapan yang dilakukan adalah *testing* atau pengujian sistem. *Software* yang dibangun harus diuji coba, dimana difokuskan terhadap tiga aktivitas yakni logika internal perangkat lunak, pemastian bahwa semua perintah yang ada telah dicoba, dan fungsi eksternal untuk memastikan bahwa dengan masukan tertentu suatu fungsi akan menghasilkan keluaran sesuai dengan yang di kehendaki (Febriansyah, 2013). Salah satu teknik uji dan analisis dalam pengembangan sistem adalah dengan menggunakan rancangan percobaan.

Perancangan percobaan adalah suatu uji atau sederetan uji baik menggunakan statistika deskripsi maupun statistik inferensi yang bertujuan untuk mengubah peubah input menjadi suatu output yang merupakan respon dari percobaan tersebut atau Perancangan percobaan adalah prosedur untuk menempatkan perlakuan ke dalam satuan-satuan percobaan dengan tujuan utama mendapatkan data yang memenuhi persyaratan ilmiah (Raupong, 2011). Tujuan dasar suatu rancangan percobaan adalah untuk membandingkan efek-efek dari berbagai tingkatan (kondisi) suatu percobaan.

Berdasarkan model dan variasi variabel nya, rancangan percobaan terdiri atas beberapa model (Pratisto, 2005):

1. Rancangan Acak Lengkap/ *Complete Randomized Design*

   Rancangan acak lengkap merupakan rancangan percobaan yang paling sederhana diantara rancangan percobaan lainnya. Rancangan Acak Lengkap biasa diterapkan pada percobaan satu faktor, dan biasanya digunakan jika kondisi unit percobaan yang digunakan relatif homogen, yaitu unit eksperimen dianggap sama dan seragam atau hanaya satu faktor yang diselidiki.

2. Rancangan Acak Kelompok / *Randomized Complete Block Design*

   Rancangan ini merupakan salah satu rancangan yag paling banyak digunakan dalam percobaan ilmiah, rancangan ini dicirikan dengan kelompok dalam jumlah yang sama dimana tiap kelompok diberikan perlakuan-perlakuan.

3. Rancangan Bujur sangkar Latin/ *Latin square Design*

   Rancangan ini dikenal dengan nama bujursangkar latin karena tataletak percobaan berbentuk bujursangkar. Setiap perlakuan hanya diberikan sekali untuk setiap baris dan sekali untuk setiap kolom, dan pengelompokan satuan percobaan berdasarkan kriteria melalui pengelompokan baris dan kolom

4. Rancangan Bujur Sangkar Graeco-Latin

   Rancangan Bujur Sangkar Graeco-Latin dapat digunakan untuk mengendalikan tiga sumber keragaman atau biasa disebut pengelompokan dalam 3 arah.

5. Rancangan Faktorial

   Rancangan Faktorial digunakan untuk penelitian jika faktor-faktor yang mempengaruhi ada dua atau lebih. Atau diartikan, Rancangan Faktorial adalah suatu percobaan mengenai sekumpulan perlakuan yang terdiri atas semua kombinasi yang mungkin dari taraf beberapa faktor. Sekumpulan kombinasi tersebut adalah yang dinyatakan sebagai faktorial.

3. 6. Data Flow Diagram (DFD)

Data Flow Diagram (DFD) yaitu representasi grafik dari sebuah sistem. DFD menggambarkan komponen-komponen sebuah sistem, aliran-aliran data di mana komponen-komponen tersebut, dan asal, tujuan, dan penyimpanan dari data tersebut. DFD bisa juga dikatakan sebagai suatu model logika data atau proses yang dibuat untuk menggambarkan dari mana asal data dan kemana tujuan data

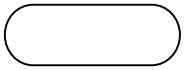
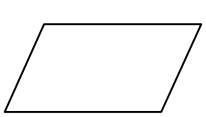
yang keluar dari sistem, dimana data disimpan, proses apa yang menghasilkan data tersebut dan interaksi antara data yang tersimpan dan proses yang dikenakan pada data tersebut.

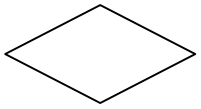
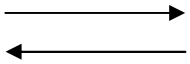
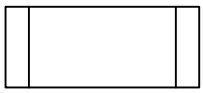
DFD merupakan alat yang cukup popular sekarang ini, karena dapat menggambarkan arus data didalam sistem dengan terstruktur dan jelas. Lebih lanjut DFD juga merupakan dokumentasi yang baik. DFD merupakan alat yang digunakan pada metodologi pengembangan sistem yang terstruktur. Kelebihan utama pendekatan aliran data, yaitu :

1. Kebebasan dari menjalankan implementasi teknis sistem.
2. Pemahaman lebih jauh mengenai keterkaitan satu sama lain dalam sistem dan subsistem.
3. Mengkomunikasikan pengetahuan sistem yang ada dengan pengguna melalui diagram aliran data.
4. Menganalisis sistem yang diajukan untuk menentukan apakah datadata dan proses yang diperlukan sudah ditetapkan

Data Flow Diagram biasa nya digambarkan dalam bentuk simbol. Daftar simbol yang ada pada *Data Flow Diagram* adalah pada tabel 2.10 :

Tabel 2.10 Simbol Data Flow Diagram (DFD)

Simbol	Keterangan
	Terminal Yaitu menyatakan mulai dan selesainya suatu proses
	*Input/Output* Menyatakan proses *input* dan *Output* data

	Proses Sesuatu yang melakukan transformasi terhadap data.
	*Decision* Menyatakan pengambilan keputusan sesuai dengan suatu kondisi
	Aliran Data Aliran data menghubungkan keluaran dari suatu objek atau proses yang terjadi pada suatu masukan.
	*Data Store* Objek pasif dalam DFD yang menyimpan data untuk penggunaan lebih lanjut.
	Titik konektor Konektor jika masih pada halaman yang sama
	Titik konektor Konektor jika pada halaman yang lain

Dari penjelasan diatas dapat diambil kesimpulan bahwa *Data Flow Diagram* (DFD) merupakan alat perancangan system yang berorientasi pada alur data yang dapat digunakan untuk penggambaran analisa maupun rancangan system yang mudah dikomunikasikan oeh *professional* sistem kepada pemakai maupun pembuat program. Secara garis besar langkah untuk membuat DFD adalah sebagai berikut :

1. Identifikasi terlebih dahulu semua entitas luar yang terlibat di sistem.
2. Identifikasi semua *input* dan *output* yang terlibat dengan entitas luar.
3. Buat Diagram Konteks. Diagram ini adalah diagram level tertinggi dari DFD yang menggambarkan hubungan sistem dengan lingkungan luarnya.

Adapun langkah-langkah dalam pembuatan diagram ini yaitu: a) Tentukan nama sistemnya, b) Tentukan batasan sistemnya, c) Tentukan terminator apa saja yang ada dalam sistem, d) Tentukan apa yang diterima/diberikan terminator dari/ke sistem, e) Gambarkan diagram konteks.

4. Buat Diagram Level *Zero*. Diagram ini adalah dekomposisi dari diagram konteks. Adapun Caranya yaitu : a) Tentukan proses utama yang ada pada sistem, b) Tentukan apa yang diberikan/diterima masing-masing proses ke/dari sistem sambil memperhatikan konsep keseimbangan (alur data yang keluar/masuk dari suatu level harus sama dengan alur data yang masuk/keluar pada level berikutnya), c) Apabila diperlukan, munculkan *Data store* (master) sebagai sumber maupun tujuan alur data, d) Gambarkan diagram level *zero*.
5. Buat Diagram Level Satu. Diagram ini merupakan dekomposisi dari diagram level *zero*. Adapun langkah-langkahnya yaitu: a) Tentukan proses yang lebih kecil dari proses utama yang ada di level *zero*, b) Tentukan apa yang diberikan/diterima masing-masing sub-proses ke/dari sistem dan perhatikan konsep keseimbangan, c) Apabila diperlukan, munculkan *Data store* (transaksi) sebagai sumber maupun tujuan alur data, d) Gambarkan DFD level Satu.
6. DFD Level Dua, Tiga, dan seterusnya. Diagram ini merupakan dekomposisi dari level sebelumnya. Proses dekomposisi dilakukan sampai dengan proses siap dituangkan ke dalam program. Aturan yang digunakan sama dengan level satu.

3. 7. Penelitian Terkait Sebelumnya

Pada penelitian kali ini, penulis juga berpedoman pada penelitian-penelitian yang telah dilakukan sebelumnya. Adapaun penelitian terkait yang pernah dilakukan sebelumnya adalah :

1. Judul "Studi Strategi Penggunaan Algoritma *Greedy* Untuk Membangun *Minimum Spanning Tree* Pada Graf Berbobot (*Weighted Graph*)", diteliti oleh Sahat Hamonangan Simorangkir, FMIPA Universitas Sumatera Utara. Dari penelitian tersebut, dijelaskan untuk menyelesaikan masalah *minimum spanning tree* strategi yang digunakan adalah *algoritma greedy* yaitu algoritma Prim dan algoritma Kruskal dengan membandingkan dengan algoritma program dinamik. Dengan kedua algoritma tersebut,dapat diselesaikan persoalan *minimum spanning tree* dari graf berbobot dengan 10 simpul dan dari hasil perhitungan diperoleh solusi optimal yatu 96 (Simorangkir, 2010).
2. Judul "Penerapan Algoritma Semut Untuk Pemecahan Masalah *Spanning Tree* Pada Kasus Pemasangan Jaringan Kabel Telepon". Diteliti oleh Rusdi Efendi, Fakultas Teknik Informatika, Universitas Islam Indonesia. Pada penelitian ini disimpulkan Algoritma semut dapat dijadikan salah satu solusi permasalahan masalah *spanning tree* dan hasil yang dihasilkan belum tentu menghasilkan hasil optimum, walaupun dari keseluruhan generasi (untuk 1 kali proses) merupakan solusi yang optimum, ini disebabkan proses penentuan titik awal selalu secara random (Efendi, 2003).

3. Judul " Optimasi Irigasi Sawah dengan Menggunakan algoritma Genetika" Diteliti oleh Bilqis Amaliah, dkk, Jurusan Informatika, Institut Sepuluh November. Penelitian ini mengusulkan optimasi pengaturan distribusi air pada saluran irigasi sawah menggunakan *Generalized Minimum Spanning Tree* Menggunakan Algoritma Genetika. Hasil dari penelitian ini adalah jumlah minimum saluran irigasi sawah dan minimum total panjang saluran irigasi sawah dan dihasilkan minimum panjang saluran irigasi adalah 658 dengan ketentuan total generasi 50 dan peluang *crossover* 0.1 (Bilqis Amaliah, 2009)

BAB III

METODOLOGI PENELITIAN

3. 1. Jenis Penelitian

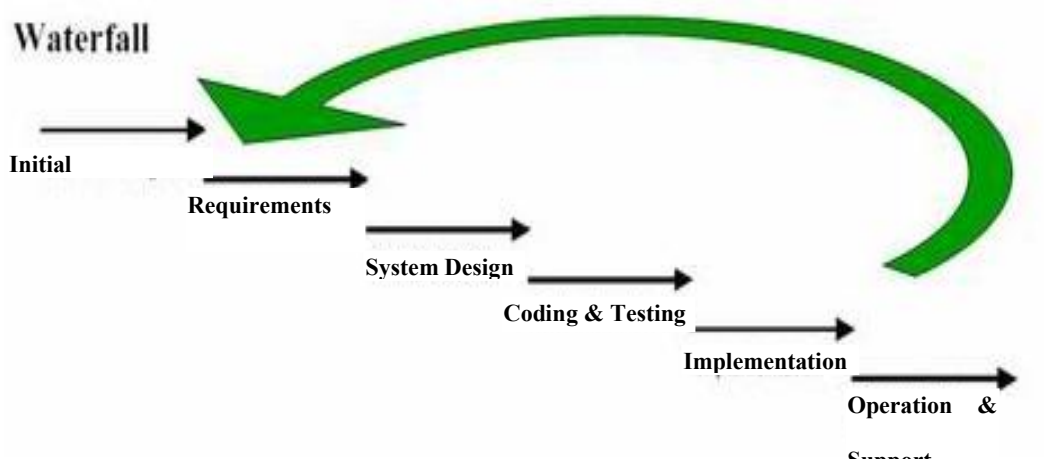
Metode penelitian yang akan digunakan peneliti adalah metode penelitian terapan. Penelitian terapan dilakukan dengan tujuan menerapkan, menguji, dan mengevaluasi kemampuan suatu teori yang diterapkan dalam memecahkan masalah-masalah praktis. Pada penelitian ini, penelitian terapan digunakan karena tujuan dari penelitian ini dalah menerapkan algoritma kruskal dan algoritma genetika dalam menghasilkan perangkat lunak yang dapat mengukur dan membandingkan kinerja algoritma genetika tersebut pada penyelesaian masalah MST *(Minimum Spanning Tree)*

3. 2. Metode Pengumpulan Data

Teknik pengumpulan data yang digunakan pada penelitian ini adalah Metode studi Pustaka, yaitu, mengkaji berbagai sumber pustaka yang dapat berupa text book, buku panduann belajar pemrograman, ataupun sumber lainnya dari internet yang berhubungan dengan *Minimum Spanning Tree* (MST), Algoritma Kruskal dan Algoritma Genetika dan juga berpedoman pada penelitian sejenis sebelumnya yang berkaitan.

3. 3. Metode Pengembangan Sistem

Metode pengembangan sistem yang digunakan dalam penelitian ini adalah metode Sekuensial Linier atau biasa disebut model Waterfall versi Zhao pada Gambar 3.1



Gambar 3.1 Metode Waterfall ((Zhao, 2010)

Model ini melakukan pendekatan secara sistematis dan berurutan. Disebut dengan waterfall karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan.

Metode ini terdiri dari enam tahapan sebagai berikut :

1. Penelitian Awal (*Initial Investigation*)

   Tahap awal ini dibutuhkan untuk menentukan rumusan masalah yang akan menjadi acuan tahapan pengembangan sistem selanjutnya. Kemudian akan dikaji batasan masalah, sasaran, serta pemodelan masalah yang benar agar tidak melenceng dari tujuan awal pengembangan sistem.

2. Definisi Kebutuhan (*Requirements Definition*)

   Tahap ini sama halnya dengan melakukan analisis terhadap sistem. Pada tahap ini dilakukan pengumpulan kebutuhan sistem yang berupa data input, proses yang terjadi dan output yang diharapkan. Proses

pendefinisian kebutuhan difokuskan untuk mengetahui sifat dari program yang akan dibuat sehingga diperlukan pemahaman tentang domain informasi dari software, misalnya fungsi yang dibutuhkan, user interface, kebutuhan algoritma, dsb. Hasil analisis akan digambarkan secara terstruktur berupa diagram aliran data (DFD).

3. Perancangan Sistem (*System Design*)

   Tahap ini menterjemahkan analisa kebutuhan ke dalam bentuk rancangan sebelum dilakukan penulisan program (pengkodean). Perancangan yang dilakukan berupa perancangan antarmuka (input dan output), perancangan file-file atau basis data dan merancang prosedur atau algoritma.

4. Pengkodean dan Pengujian (*Coding and Testing*)

   Hasil perancangan sistem kemudian diubah menjadi bentuk yang dimengerti oleh mesin yaitu ke dalam bahasa pemrograman melalui proses coding (pengkodean). Jika rancangannya rinci maka penulisan program dapat dilakukan dengan cepat. Selama proses pengkodean juga dilakukan proses pengujian agar sistem yang dibuat sesuai dengan algoritma yang ditentukan. Dengan demikian, apabila terdapat kekurangan yang terlihat pada proses pengujian dapat langsung diubah sesuai dengan kebutuhan. Metode pengujian yang dilakukan dibedakan pada dua teknik pengujian, yaitu pengujian perangkat lunak dan pengujian hasil. Pengujian perangkat lunak dilakukan pada aplikasi yang dibagun menggunakan teknik pendekatan *black box testing* (ujicoba blackbox), dan hasil akhir dengan metode rancangan percobaan.

5. Implementasi (*Implementation*)

   Tahap implementasi sistem merupakan tahap mempersiapkan sistem agar siap untuk dioperasikan.

6. Pengoperasian dan Dukungan (*Operation and Support*)

   Tahap ini adalah tahap akhir pengembangan sistem yaitu pengoperasian sistem secara nyata telah dilakukan. Namun dalam pengoperasiannya tetap dibutuhkan dukungan agar sistem dapat digunakan dalam jangka panjang yaitu dengan melakukan pemeliharaan. Pemeliharaan suatu software diperlukan, termasuk di dalamnya adalah pengembangan, karena software yang dibuat tidak selamanya hanya seperti itu. Ketika dijalankan mungkin saja masih ada kesalahan kecil yang tidak ditemukan sebelumnya, atau ada penambahan fitur-fitur yang belum ada pada software tersebut.

3. 4. Metode Pengujian

Metode pengujian yang dilakukan dibedakan pada dua teknik pengujian, yaitu pengujian perangkat lunak dan pengujian hasil. Pengujian perangkat lunak dilakukan pada aplikasi yang dibagun menggunakan teknik pendekatan *black box testing* (ujicoba *blackbox*). Ujicoba blackbox menyinggung ujicoba yang dilakukan pada interface software. Walaupun didesain untuk menemukan kesalahan, ujicoba *blackbox* digunakan untuk mendemonstrasikan fungsi software yang dioperasikan; apakah input diterima dengan benar, dan ouput yang dihasilkan benar; apakah integritas informasi eksternal terpelihara. Sedangkan untuk hasil perhitungan yang dihasilkan, digunakan teknik Rancangan Percobaan.

Dimana setiap input di hitung dan dinilai kemudian hasil akhirnya dibandingkan dengan pengujian sebelumnya.

Pada pengujian hasil perhitungan algoritma yang di hasilkan pada penelitian, diuji dengan metode Uji Perancangan percobaan. Perancangan percobaan adalah suatu uji atau sederetan uji baik menggunakan statistika deskripsi maupun statistik inferensi yang bertujuan untuk mengubah peubah input menjadi suatu output yang merupakan respon dari percobaan tersebut atau Perancangan percobaan adalah prosedur untuk menempatkan perlakuan ke dalam satuan-satuan percobaan dengan tujuan utama mendapatkan data yang memenuhi persyaratan ilmiah (Raupong, 2011). Tujuan dasar suatu rancangan percobaan adalah untuk membandingkan efek-efek dari berbagai tingkatan (kondisi) suatu percobaan.

Metode pengujian dilakukan dengan melibatkan sejumlah nilai yang dipilih dari data masukan. Sejumlah nilai di-*input*-kan kepada kedua algoritma, kruskal dan genetika, yang kemudian dihitung nilai minimumnya, tiap percobaan diberikan nilai input yang berbeda. Nilai-nilai yang dimaksudkan adalah jumlah simpul, dan parameter Genetika yaitu jumlah populasi, peluang *crossover*, peluang Mutasi dan Peluang Kelestarian. Jika saat ini adalah percobaan ke-n maka nilai input pada percobaan ke- n+1 lebih besar dari nilai input sebelumnya. Setelah beberapa percobaan, nantinya dihitung perbandingan dari hasil sebelumnya.

3. 5. Jadwal Penelitian

Penelitian ini direncanakan dimulai pada bulan Desember 2013 – Mei 2014.

No	Kegiatan	Des				Jan				Feb				Mar				Apr				Mei			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Investigasi Awal																								
	a. Latar Belakang Penelitian	■																							
	b. Tujuan dan Ruang Lingkup Penelitian	■	■																						
2	Defenisi kebutuhan																								
	a. Mengumpulkan data			■	■																				
	b. Analisis kebutuhan				■	■																			
3	Analisis dan desain sistem						■	■	■	■															
4	Pembuatan *Coding*									■	■	■	■	■	■	■	■	■	■	■	■	■	■		
5	Pengujian																					■	■		
6	Laporan																							■	■