ANALISIS PERBANDINGAN ALGORITMA PRIM DENGAN ALGORITMA DIJKSTRA DALAM PEMBENTUKAN MINIMUM SPANNING TREE (MST)

Edy Rahman Syahputra

Sistem Informasi, Sekolah Tinggi Teknik Harapan
Jl.H.M. Joni No.7 C, Medan, Sumatera Utara

ydeaja@yahoo.com

Abstrak

Membandingkan suatu besaran yang diukur dengan alat ukur yang digunakan sebagai satuan adalah bentuk pengukuran. Perlunya suatu perbandingan merupakan bentuk pengembangan ilmu pengetahuan. Matematika merupakan ilmu yang merupakan dasar dari ilmu-ilmu pengetahuan yang ada. *Minimum Spanning Tree* (MST) adalah suatu bentuk pohon rentang minimum yang digunakan untuk membentuk jalur terpendek atau terkecil. Algoritma untuk mencari jalur terpendek dan jarak terkecil dilakukan dan diaplikasikan menggunakan cara masing-masing algoritma dan metode itu sendiri, sehingga masih dimungkinkan belum diketahui algoritma mana yang paling optimal. Untuk mengetahui algoritma mana yang optimal maka dilakukanlah sebuah langkah untuk membandingkan antara algoritma prim dengan algoritma dijkstra. Dari pengujian yang dilakukan didapatkan hasil algoritma prim efektif dalam pembentukan *Minimum Spanning Tree* (MST) dan total jarak yang dihasilkan lebih kecil dibanding algoritma dijkstra. Sedangkan algoritma dijkstra efektif dalam menghasilkan jarak antar vertex awal menuju vertex tujuan dibandingkan algoritma prim.

Kata Kunci : Algoritma Prim, Algoritma Djikstra, MST

1. Pendahuluan

Banyak permasalahan yang dapat dimodelkan dengan menggunakan graf, khususnya di bidang teknologi informasi. Salah satunya adalah masalah dalam pencarian bobot terpendek, atau termurah guna meminimal biaya. Banyak aplikasi yang membutuhkan adanya pencarian jarak / jalur terpendek dari suatu graf. Salah satu cabang dari teori graf yang banyak dikembangkan saat ini adalah penggunaan teori pohon dimana setiap graf terhubung memiliki pohon merentang [1].

Konsep pohon dianggap sebagai salah satu konsep yang paling populer saat ini dan banyak digunakan untuk menyelesaikan beberapa masalah yang ditemui manusia dalam kehidupan sehari-harinya seperti ketika hendak direntangkan jaringan kabel listrik yang menghubungkan sejumlah lokasi dengan panjang kabel yang digunakan sependek-pendeknya mungkin, melihat pengelompokan data yang tersebar pada suatu ruang, ataupun pada perencanaan jaringan transportasi/distribusi barang seperti membangun jalur kereta api yang menghubungkan sejumlah kota ataukah seorang salesman yang mengadakan kunjungan sejumlah kota sebanyak n kota, dimana salesman tersebut harus mengunjungi seluruh kota tersebut dengan meminimal jarak perjalanan [2].

Untuk menentukan rute perjalanan terkadang kita harus menggunakan beberapa metode dalam matematika, banyak didalam matematika algoritma yang membahasa mengenai teroi graf diantaranya algoritma prim dan algoritma dijkstra. Algoritma prim adalah sebuah algoritma dalam teori graph untuk membuat sebuah lintasan dalam bentuk spanning tree untuk sebuah graf berbobot yang saling terhubung [3], sementara algoritma dijkstra banyak digunakan untuk mencari jalur terdekat dan terpendek untuk mengoptimalisasikan jarak [4].

Selama ini seluruh metode dan algoritma untuk mencari jalur terdekat dan jarak terkecil dilakukan dan diaplikasikan menggunakan cara masing-masing algoritma dan metode itu sendiri, sehingga masih dimungkinkan belum diketahui lagoritma mana yang paling optimal. Untuk mengetahui algoritma mana yang optimal maka dilakukanlah sebuah langkah untuk membandingkan algoritma tersebut, dengan membandingkan algoritma prim dengan algoritma dijkstra.

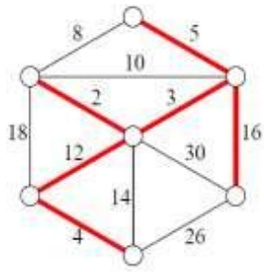
2. Minimum Spanning Tree (MST)

Pohon merupakan sebuah bentuk graf yang saling terhubung dengan tidak membentuk suatu sirkuit. Untuk pengertian pohon yang lainnya yaitu graf yang tidak mempunyai arah yang bersifat membentuk satu lintasan antar node yang terhubung. Beberapa pohon dapat membentuk hutan (*forest*) yaitu kumpulan pohon yang saling lepas. Konsep pohon ini memiliki penerapan yang luas, baik di dalam ilmu komputer maupun diluar bidang ilmu komputer [5].

Pohon membeantu untuk mempelajari struktur data. Para ahli bahasa menggunakan konsep pohon parsing (*parse tree*) untuk menguraikan kalimat. Dalam kehidupan sehari-hari kita telah menggunakan konsep pohon untuk menggambarkan diagram system gugur pertandingan olaharaga, silsilah keluarga, struktur organisasi, dan lain-lain [6].

Misalkan terdapat graf yang sangat kompleks, dimana ada beberapa alternatif untuk melakukan kunjungan-kunjungan (*visiting*) dari satu simpul ke simpul- simpul yang lainnya, tentu dapat segera dicari tahu alternatif yang terbaik untuk melakukannya. Dalam hal ini, alternatif yang cukup baik adalah dengan mencari jarak yang terdekat antar simpul itu [7].

Contoh aplikasi pohon rentang yaitu pada pemeliharaan jalan raya dengan dana terbatas. Alokasi dana yang terbatas akan menyebabkan pertimbangan hanya merawat ruas jalan seminimal mungkin, namun semua daerah masih tetap terhubung satu dengan yang lain [8].



Gambar 1. Pohon Rentang Minimum

Di sini akan digunakan dua buah algoritma untuk menyusun pohon rentang minimum yaitu algoritma Prim dan Dijkstra. Proses atau langkah-langkah penyusunan pohon menggunakan kedua algoritma ini berbeda.

3. Algoritma Prim

Algoritma Prim dimulai dari simpul yang berubah-ubah di setiap tingkatnya, diperbolehkan menambah cabang baru untuk membuat susunan pohon baru. Algoritma ini akan tertahan ketika simpul yang sedang dieksplorasi pada graf sudah sampai pada simpul yang dituju. Strategi yang digunakan adalah strategi Greedy dengan menganggap bahwa pada setiap langkah dari pohon merentangnya adalah *augmented* dan dipilih simpul yang nilainya paling kecil dari semua simpul yang ada [7].

Algoritma Prim menitikberatkan pada pemilihan bobot minimum berdasarkan simpul yang diambil. Dan karena tidak perlu mengurutkan terlebih dahulu, algoritma Prim cocok untuk pohon dengan jumlah simpul banyak. Algoritma Prim akan selalu berhasil menemukan pohon merentang minimum tetapi pohon merentang yang dihasilkan tidak selalu unik.

Langkah-langkah kerja Algoritma Prim :

1. Ambil sisi dari graf *G* yang berbobot minimum, masukkan ke dalam *T*
2. Pilih sisi (*u, v)* yang memiliki bobot minimum dan bersisian dengan simpul di *T*. Tetapi *(u, v)* tidak membentuk sirkuit di *T*. Tambahkan *(u, v)* ke dalam *T*.
3. Ulangi langkah 2 sebanyak (*n*-2) kali.

Jumlah seluruh langkah pada algoritma Prim ini adalah 1+(*n*-2) = *n* – 1, yaitu sama dengan jumlah sisi pada pohon merentang dengan *n* buah simpul [9].

Algoritma Prim dalam *pseudocode* yaitu:

Procedure Prim (input G:graf, output T:pohon)
{ Membentuk pohon merentang minimum T dari graf terhubung G}.
Masukan: graf-berbobot terhubung G = (V, E),
yang mana |V| = n
Keluaran: pohon merentang minimum T = (V, E')}
Deklarasi
i, p, q, u, v : integer
Algoritma Cari sisi (p,q) dari E yang berbobot terkecil
T ← { (p,q)}
for i ← 1 to n-1 do
Pilih sisi (u,v) dari E yang bobotnya terkecil namun bersisian dengan suatu simpul didalam T
T ← T U { (u,v)}
Endfor [7].

4. Algoritma Dijkstra

Algoritma dijkstra adalah algoritma pencarian grafik yang memecahkan masalah jalur terpendek dari satu sumber dengan nilai jalur yang dihasilkan tidak negatif, dan menghasilkan pohon jalur terpendek. Algoritma ini sering digunakan dalam pencarian rute. Untuk sumber simpul (node) tertentu dalam grafik, algoritma menghasilkan jalur dengan biaya terendah (yaitu lintasan terpendek) antara *vertex* dan *vertex* lainnya [10].

Langkah-langkah penentuan lintasan terpendek dari graf G dengan n-buah simpul dengan simpul awal a menggunakan algoritma dijkstra adalah sebagai berikut:

1. Langkah 0 (inisialisasi) : $s_i = 0$ dan $d_i = m_{ai}$ untuk $i = 1,2,...,n$
2. Langkah 1 : isi s_a dengan 1 dan isi d_a dengan ∞
3. Langkah 2: untuk setiap $s_i = 0$ dan $i=1,2,..,n$, pilih $d_j = min\ \{d1,d2,....,dn\}$ lalu isi s_j dengan 1 dan perbaharui d_i, dengan : d_i *(baru)* $= mid\ \{d_i$ *(lama)*, $d_j+m_{ji}\}$. Pada lintasan tambahkan simpul *j* sebagai simpul terpilih untuk lintasan selanjutnya.
4. Langkah 3: mengulangi langkah 2 *sampai* $s_j=1$, untuk $j=1,2,...,n$
5. Membuat himpunan simpul berdasarkan urutan yang diperoleh yang merupakan lintasan terpendek dengan bobot d_i." [4].

Pseudokode Dijkstra
*procedur**e** dijkstra (w, a, z, L)*
L(a) : = 0
S : = { }
*fo**r** semua vertex x ≠ a do*
L(x) : = ∞
T : = himpunan semua vertex
while z (T do
begin
pilih v (T dengan minimum L (v)
T : = T – {v}
S : = S union {v}
for setiap x(T di samping v
do
L(x) : = min {L(x), L (v) + w (v, w)}
end
end dijkstra

5. Metode Penelitian

Adapun tahapan mengenai metodilogi dalam penelitian ini akan diuraikan dalam bentuk kerangka kerja. Kerangka kerja merupakan langkah-langkah yang digunakan untuk menyelesaikan permasalahan yang dibahas. Untuk lebih jelasnya mengenai kerangka kerja sebagai berikut:

1. Identifikasi masalah : Tahapan ini merupakan tahapan yang akan digunakan untuk merumuskan masalah yang akan menjadi objek penelitian. Perumusan masalah dilakukan untuk menentukan masalah apa saja yang terdapat pada objek penelitian serta memberikan batasan dari permasalahan yang akan diteliti.
2. Analisa masalah : Langkah analisis masalah adalah langkah untuk dapat memahami masalah yang telah ditentukan ruang lingkup atau batasannya. Dengan menganalisa masalah yang telah ditentukan tersebut, maka diharapkan masalah dapat dipahami dengan baik.
3. Mempelajari literature : Berdasarkan pemahaman dari masalah, maka ditentukan tujuan yang akan dicapai dari penulisan penelitian ini. Pada tujuan ini ditentukan target yang dicapai, terutama yang dapat mengatasi masalah-masalah yang ada. Setelah masalah dianalisa, maka dipelajari literatur yang berhubungan dengan permasalahan. Kemudian literatur-literatur yang dipelajari tersebut

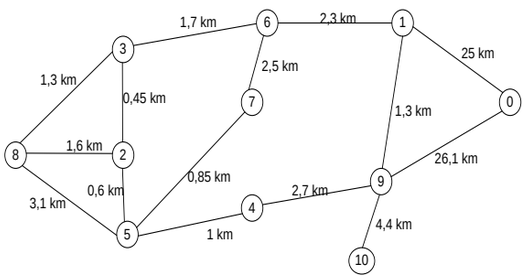
diseleksi untuk dapat ditentukan literatur mana yang akan digunakan dalam penelitian ini. Sumber literatur didapatkan dari jurnal, artikel, yang membahas tentang algoritma prim dan dijkstra dan bahan bacaan lain yang mendukung penelitian.

4. Pengujian : Tahap ini dilakukan mekanisme pungujian dengan cara memasukan berupa node-node yang terhubung dan membandingkan hasil dari pengujian algoritma prim dan dijkstra.
5. Analisa hasil : Berdasarkan analisis yang telah dilakukan maka dapat diketahui algortima mana yang lebih optimal dalam pembentukan *Minimum Spanning Tree* (MST).

6. Analisa Masalah

6.1 Kasus 1

Sebuah perusahaan yang bergerak dalam bidang air bersih membangun jaringan baru yang menghubungkan dengan semua wilayah dari jaringan yang lama dengan asumsi semakin panjang pipa yang dipasang semakin mahal biaya yang harus dikeluarkan. Jaringan dibangun memakan biaya sesedikit mungkin. Dengan bentuk graf sebagai berikut:

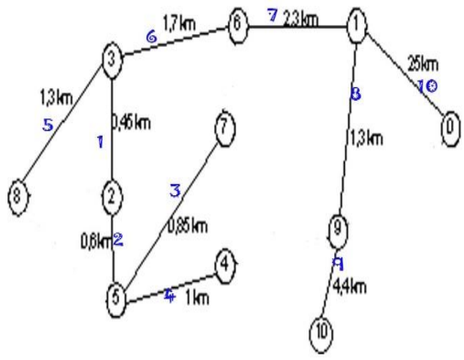


Gambar 2. Model Graph Pada Kasus 1

Algoritma Prim:
Adapun langkah-langkah algoritma prim dalam penyelesaian kasus1 adalah sebagai berikut:

1. Pilih lintasan (sisi) dengan bobot terkecil, dalam kasus ini adalah sisi (2,3).
2. Cek *vertex* berikutnya, *vertex* manakah yang lebih dekat dengan *vertex* lintasan terkecil tadi. Dalam kasus ini vertex 8, 6, lebih dekat kevertex 3 dan vertex. Tambahkan kepohon rentang. Lintasan terkecil yang dekat dengan lintasan (2,3) adalah (2,5).
3. Cek vertex berikutnya yang terdekat dengan vertex 2, 3, dan 5 yang tidak membentuk sirkuit. Lintasan yang dipilih adalah (5,7).
4. Selanjutnya pilih kembali vertex yang terdekat dengan vertex 2, 3, 5, dan 7 yang tidak membentuk sirkuit. Lintasan yang dipilih adalah (5,4).
5. Selanjutnya cek vertex yang terdekat dengan vertex 2, 3, 4, 5, dan 7 yang tidak membentuk sirkuit. Lintasan yang dipilih adalah (3,8).
6. Selanjutnya cek vertex yang terdekat dengan vertex 3, 4, dan 7 yang tidak membentuk sirkuit. Lintasan yang dipilih adalah (3,6).
7. Selanjutnya cek vertex yang terdekat dengan vertex 4 dan 6 yang tidak membentuk sirkuit. Lintasan yang dipilih adalah (6,1).
8. Selanjutnya cek vertex yang terdekat dengan vertex 1 dan 4 yang tidak membentuk sirkuit. Lintasan yang dipilih adalah (1,9).
9. Selanjutnya cek vertex yang terdekat dengan vertex 1, dan 9 yang tidak membentuk sirkuit. Lintasan yang dipilih adalah (9,10).
10. Dan lintasan yang terakhir dipilih adalah (1,0).

Adapun hasil lintasan yang dipilih adalah (2,3), (2,5), (5,7), (5,4), (3,8), (3,6), (6,1), (1,9), (9,10) dan (1,0) dengan total bobot minimum 38,9 km dengan membentuk *minimum spanning tree* (MST) sebagai berikut:

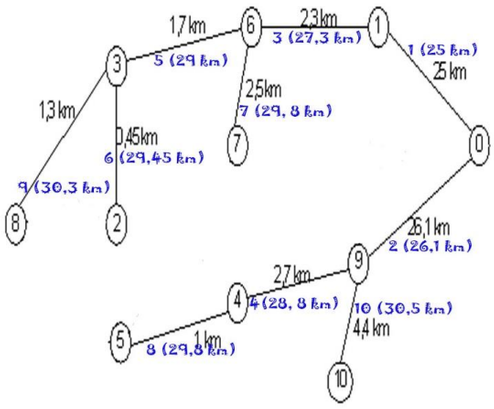


Gambar 3. Pembentukan MST dengan Algoritma Prim pada kasus 1

Algoritma Dijkstra:
Adapun langkah-langkah algoritma dijkstra dalam penyelesaian kasus1 adalah sebagai berikut:

1. Tentukan awal vertex, dalam kasus 1 vertex awal adala 0.
2. Vertex yang terhubung dengan vertex 0 adalah vertex 1 dan 9. Dipilih lintasan yang terpendek adalah (0,1).
3. Selanjutnya cek vertex yang terdekat dengan 0 dan 1 yang mempunyai jumlah bobot terkecil dihitung dari awal node dan tidak membentuk sirkuit. Dipilih lintasan (0,9).
4. Cek vertex yang terdekat dengan vertex 1 dan 9 yang mempunyai jumlah bobot terkecil dihitung dari awal node dan tidak membentuk sirkuit. Dipilih lintasan (1,6).
5. Cek vertex yang terdekat dengan vertex 6, dan 9 yang mempunyai jumlah bobot terkecil dihitung dari awal node dan tidak membentuk sirkuit. Dipilih lintasan (9,4).
6. Cek vertex yang terdekat dengan vertex 4, 6 dan 9 yang mempunyai jumlah bobot terkecil dihitung dari awal node dan tidak membentuk sirkuit. Dipilih lintasan (6,3).
7. Cek vertex yang terdekat dengan vertex 3, 4, 6 dan 9 yang mempunyai jumlah bobot terkecil dihitung dari awal node dan tidak membentuk sirkuit. Dipilih lintasan (3,2).
8. Cek vertex yang terdekat dengan vertex 2, 3, 4, 6 dan 9 yang mempunyai jumlah bobot terkecil dihitung dari awal node dan tidak membentuk sirkuit. Dipilih lintasan (6,7).
9. Cek vertex yang terdekat dengan vertex 2, 3, 4 dan 9 yang mempunyai jumlah bobot terkecil dihitung dari awal node dan tidak membentuk sirkuit. Dipilih lintasan (4,5).
10. Cek vertex yang terdekat dengan vertex 2, 3, 5 dan 9 yang mempunyai jumlah bobot terkecil dihitung dari awal node dan tidak membentuk sirkuit. Dipilih lintasan (3,8).
11. Karena hanya tinggal 1 lintasan yang belum dipilih dan tidak membentuk sirkuit maka lintasan yang dipilih dengan total bobot terkecil dari node awal adalah (9,10).

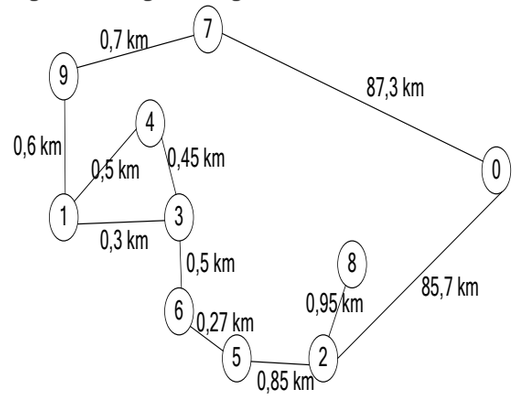
Adapun hasil lintasan yang dipilih adalah (0,1), (0,9), (1,6), (9,4), (6,3), (3,2), (6,7), (4,5), (3,8), dan (9,10) dengan total bobot minimum 67,45 km dengan membentuk *minimum spanning tree* (MST) sebagai berikut:



Gambar 4. Pembentukan MST dengan Algoritma Dijkstra pada kasus 1

6.2 Kasus 2

Pada sebuah perusahaan telekomunikasi ingin membangun jaringan baru dengan biaya minimal tetapi tetap terhubung dengan seluruh wilayah. Dengan bentuk graf sebagai berikut:

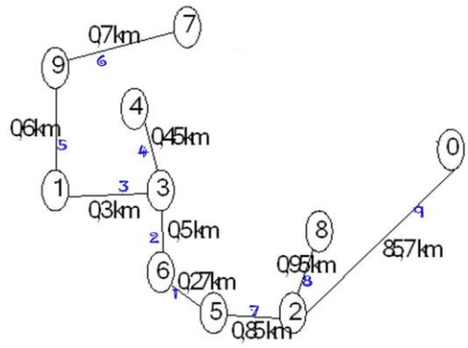


Gambar 5. Model Graph Pada Kasus 2

Algoritma Prim:
Adapun langkah-langkah algoritma prim dalam penyelesaian kasus2 adalah sebagai berikut:

1. Pilih lintasan (sisi) dengan bobot terkecil, dalam kasus ini adalah sisi (5,6).
2. Cek *vertex* berikutnya, *vertex* manakah yang lebih dekat dengan *vertex* lintasan terkecil tadi. Dalam kasus ini vertex 2, 3, lebih dekat kevertex 5 dan vertex 6. Tambahkan kepohon rentang. lintasan terkecil yang dekat dengan lintasan (5,6) adalah (6,3).
3. Cek vertex berikutnya yang terdekat dengan vertex 3, dan 5 yang tidak membentuk sirkuit. Lintasan yang dipilih adalah (3,1).
4. Selanjutnya pilih kembali vertex yang terdekat dengan vertex 1, 3, dan 5 yang tidak membentuk sirkuit. Lintasan yang dipilih adalah (3,4).
5. Selanjutnya cek vertex yang terdekat dengan vertex 1 dan 5 yang tidak membentuk sirkuit. Lintasan yang dipilih adalah (1,9).
6. Selanjutnya cek vertex yang terdekat dengan vertex 5 dan 9 yang tidak membentuk sirkuit. Lintasan yang dipilih adalah (9,7).
7. Selanjutnya cek vertex yang terdekat dengan vertex 7 dan 7 yang tidak membentuk sirkuit. Lintasan yang dipilih adalah (5,2).
8. Selanjutnya cek vertex yang terdekat dengan vertex 2 dan 7 yang tidak membentuk sirkuit. Lintasan yang dipilih adalah (2,8).
9. Lintasan terakhir yang belum dipilih dan terdekat dari vertex 2 dan 7. Dan lintasan yang terakhir dipilih adalah (2,0).

Adapun hasil lintasan yang dipilih adalah (5,6), (6,3), (3,1), (3,4), (1,9), (9,7), (5,2), (2,8), dan (2,0) dengan total bobot minimum 90,32 km dengan membentuk *minimum spanning tree* (MST) sebagai berikut:

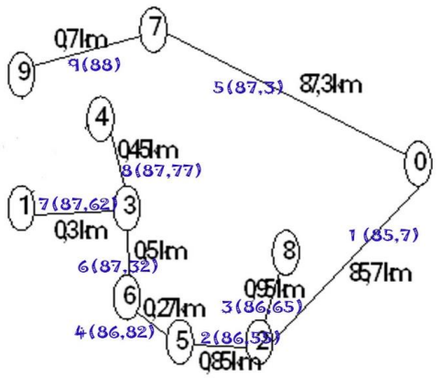


Gambar 6. Pembentukan MST dengan Algoritma Prim pada kasus 2

Algoritma Dijkstra:
Adapun langkah-langkah algoritma dijkstra dalam penyelesaian kasus 2 adalah sebagai berikut:

1. Tentukan awal vertex, dalam kasus 2 vertex awal adala 0.
2. Vertex yang terhubung dengan vertex 0 adalah vertex 2 dan 7. Dipilih lintasan yang terpendek adalah (0,2).
3. Selanjutnya cek vertex yang terdekat dengan 0 dan 2 yang mempunyai jumlah bobot terkecil dihitung dari awal node dan tidak membentuk sirkuit. Dipilih lintasan (2,5).
4. Cek vertex yang terdekat dengan vertex 0, 2 dan 5 yang mempunyai jumlah bobot terkecil dihitung dari awal node dan tidak membentuk sirkuit. Dipilih lintasan (2,8).
5. Cek vertex yang terdekat dengan vertex 0, dan 5 yang mempunyai jumlah bobot terkecil dihitung dari awal node dan tidak membentuk sirkuit. Dipilih lintasan (5,6).
6. Cek vertex yang terdekat dengan vertex 0, dan 6 yang mempunyai jumlah bobot terkecil dihitung dari awal node dan tidak membentuk sirkuit. Dipilih lintasan (0,7).
7. Cek vertex yang terdekat dengan vertex 6, dan 7 yang mempunyai jumlah bobot terkecil dihitung dari awal node dan tidak membentuk sirkuit. Dipilih lintasan (6,3).
8. Cek vertex yang terdekat dengan vertex 3 dan 7 yang mempunyai jumlah bobot terkecil dihitung dari awal node dan tidak membentuk sirkuit. Dipilih lintasan (3,1).
9. Cek vertex yang terdekat dengan vertex 1, 3 dan 7 yang mempunyai jumlah bobot terkecil dihitung dari awal node dan tidak membentuk sirkuit. Dipilih lintasan (3,4).
10. Karena hanya tinggal 1 lintasan yang belum dipilih dan tidak membentuk sirkuit maka lintasan yang dipilih dengan total bobot terkecil dari node awal adalah (7,9).

Adapun hasil lintasan yang dipilih adalah (0,2), (2,5), (2,8), (5,6), (0,7), (6,3), (3,1), (3,4), dan (7,9) dengan total bobot minimum 177,02 km dengan membentuk *minimum spanning tree* (MST) sebagai berikut:



Gambar 7. Pembentukan MST dengan Algoritma Dijkstra pada kasus 2

7. Pembahasan

Berdasarkan hasil dari pengujian terdapat perbedaan hasil yang sangat signifikan. Adapun hasilnya sebagai berikut :

Tabel 1. Perbandingan Hasil Lintasan Pada Kasus 1

Vertex Tujuan	Lintasan Yang Dilalui Antar Vertex	
	Algoritma Prim	Algoritma Dijkstra
0-1	0-1	0-1
0-2	0-1-6-3-2	0-1-6-3-2
0-3	0-1-6-3	0-1-6-3
0-4	0-1-6-3-2-5-4	0-9-4
0-5	0-1-6-3-2-5	0-9-4-5
0-6	0-1-6	0-1-6
0-7	0-1-6-3-2-5-7	0-1-6-7
0-8	0-1-6-3-8	0-1-6-3-8
0-9	0-1-9	0-9
0-10	0-1-9-10	0-9-10

Berdasarkan table 1 pada kasus 1 didapatkan beberapa lintasan yang berbeda antara algoritma prim dengan algoritma dijkstra daiantaranya adalah dari vertex tujuan 0 menuju vertex 4 (algoritma prim menghasilkan lintasan 0-1-6-3-2-5-4, sedangkan algoritma dijkstra menghasilkan lintasan 0-9-4), vertex 0 menuju vertex 5 (algoritma prim menghasilkan lintasan 0-1-6-3-2-5, sedangkan algoritma dijkstra menghasilkan lintasan 0-9-4-5), vertex 0 menuju vertex 7 (algoritma prim menghasilkan lintasan 0-1-6-3-2-5-7, sedangkan algoritma dijkstra menghasilkan lintasan 0-1-6-7), vertex 0 menuju vertex 9 (algoritma prim menghasilkan lintasan 0-1-9, sedangkan algoritma dijkstra menghasilkan lintasan 0-9), dan vertex 0 menuju vertex 10 (algoritma prim menghasilkan lintasan 0-1-9-10, sedangkan algoritma dijkstra menghasilkan lintasan 0-9-10).

Tabel 2. Perbandingan Hasil Lintasan Pada Kasus 2

Vertex Tujuan	Lintasan Yang Dilalui Antar Vertex	
	Algoritma Prim	Algoritma Dijkstra
0-1	0-2-5-6-3-1	0-2-5-6-3-1
0-2	0-2	0-2
0-3	0-2-5-6-3	0-2-5-6-3
0-4	0-2-5-6-3-4	0-2-5-6-3-4
0-5	0-2-5	0-2-5
0-6	0-2-5-6	0-2-5-6
0-7	0-2-5-6-3-1-9-7	0-7
0-8	0-2-8	0-2-8
0-9	0-2-5-6-3-1-9	0-7-9

Berdasarkan table 2 pada kasus 2 didapatkan lintasan yang berbeda antara algoritma prim dengan algoritma dijkstra yaitu dari vertex tujuan 0 menuju vertex 7 (algoritma prim menghasilkan lintasan 0-2-5-6-3-1-9-7, sedangkan algoritma dijkstra menghasilkan lintasan 0-7) dan dari vertex tujuan 0 menuju vertex 9 (algoritma prim menghasilkan lintasan 0-2-5-6-3-1-9, sedangkan algoritma dijkstra menghasilkan lintasan 0-7-9).
Sedangkan untuk total jarak lintasan antar vertex adalah sebagai berikut :

Tabel 3. Perbandingan Hasil Jarak Lintasan Pada Kasus 1

Vertex Tujuan	Jarak Lintasan Yang Dilalui Antar Vertex	
	Algoritma Prim	Algoritma Dijkstra
0-1	25	25
0-2	29,45	29,45
0-3	29	29
0-4	31,05	28,8
0-5	30,05	29,8
0-6	27,3	27,3
0-7	30,9	29,8
0-8	30,3	30,3
0-9	26,3	26,1
0-10	30,7	30,5

Berdasrkan table 3 pada kasus 1 diatas terdapat perbedaan jarak lintasan antara algoritma prim dengan algoritma dijkstra yaitu dari vertex tujuan 0 menuju vertex 4 (algoritma prim menghasilkan jarak lintasan 31,05; algoritma dijkstra menghasilkan jarak lintasan 28,8), dari vertex 0 menuju vertex 5 (algoritma prim menghasilkan jarak lintasan 30,05; algoritma dijkstra menghasilkan jarak lintasan 29,8), dari vertex 0 menuju vertex 7 (algoritma prim menghasilkan jarak lintasan 30,09; algoritma dijkstra menghasilkan jarak lintasan 29,8), dari vertex 0 menuju vertex 9 (algoritma prim menghasilkan jarak lintasan 26,3; algoritma dijkstra menghasilkan jarak lintasan 26,1), dan dari vertex 0 menuju vertex 10 (algoritma prim menghasilkan jarak lintasan 30,7; algoritma dijkstra menghasilkan jarak lintasan 30,5).

Tabel 4. Perbandingan Hasil Jarak Lintasan Pada Kasus 2

Vertex Tujuan	Jarak Lintasan Yang Dilalui Antar Vertex	
	Algoritma Prim	Algoritma Dijkstra
0-1	87,62	87,62
0-2	85,7	85,7
0-3	87,32	87,32
0-4	87,77	87,77
0-5	86,55	86,55
0-6	86,82	86,82
0-7	88,92	87,3
0-8	86,65	86,65
0-9	88,22	88

Berdasrkan table 4 pada kasus 2 diatas terdapat perbedaan jarak lintasan antara algoritma prim dengan algoritma dijkstra yaitu dari vertex tujuan 0 menuju vertex 7 (algoritma prim menghasilkan jarak lintasan 88,92; algoritma dijkstra menghasilkan jarak lintasan 87,3), dan dari vertex 0 menuju vertex 9 (algoritma prim menghasilkan jarak lintasan 88,22; algoritma dijkstra menghasilkan jarak lintasan 88).

8. Kesimpulan

Setelah dilakukan serangkaian pengujian dan analisa dalam penelitian ini, maka dapat diambil kesimpulan sebagai berikut:

1. Dari hasil pengujian pada 2 kasus yang berbeda terdapat bentuk hasil *Minimum Spanning Tree* (MST) antara algoritma prim dengan algoritma dijkstra.
2. Berdasarkan hasil lintasan antar vertex pada kasus 1 dan kasus 2 terdapat perbedaan antara algoritma prim dengan algoritma dijkstra.
3. Berdasarkan hasil jarak lintasan antar vertex pada kasus 1 dan kasus 2 terdapat perbedaan antara algoritma prim dengan algoritma dijkstra.
4. Dari hasil total jarak lintasan didapatkan hasil berbeda antara algoritma prim dan dijkstra, pada kasus 1 algoritma prim menghasilkan total jarak lintasan 38,9; sedangkan algoritma dijkstra menghasilkan total jarak lintasan 67,45. Pada kasus 2 algoritma prim menghasilkan total jarak lintasan 90,32; sedangkan algoritma dijkstra menghasilkan total jarak lintasan 177,02.
5. Dari hasil seluruh pengujian yang dilakukan, algoritma prim dalam membentuk *Minimum Spanning Tree* (MST) dan hasil total jarak lintasan lebih efektif dibandingkan algoritma dijkstra, sedangkan algoritma dijkstra lebih efektif dalam menghasilkan jarak dari vertex awal menuju vertex tujuan.

9. Daftar Pustaka

[1] Syahfitri Y. (2011). "*Membandingkan Algoritma Prim Dengan Algoritma Kruskal Dalam Penyelesaian Travelling Salesman Problem (TSP) Pada Penerbit Zikrul Hakim Medan"*. Tesis. Universitas Putra Indonesia "YPTK" Padang.

[2] Djafar I dan Ibrahim A. (2011). *"Implementasi Pohon Merentang Minimum Dalam Menentukan Prioritas Pemeliharaan Jalur Jalan Kota Dengan Biaya Minimal"*. Jurnal DIGIT. Vol 1. No 2, pp. 132-142.

[3] Nurhayani O. D. (2010). "*Algorima dan Pemrograman Pendekatan Pemrograman Modular"*. Universitas Diponegoro.

[4] Salaki T.D. (2010). "*Penentuan Lintasan Terpendek Dari FMIPA Ke Rektorat Dan Fakultas Lain Di UNSRAT Manado Menggunakan Algoritma Dijkstra"*. Jurnal Ilmiah Sains Vol.11 No.1. 74-76.

[5] Sedgewick, R., dan Wayne, K., (2011). "*Algorithm 4th Edition*". Addison-Wesley USA.

[6] Munir. (2010). "*Matematika Diskrit Edisi Ketiga Revisi Keempat"*. Informatika. Bandung.

[7] Nugraha W D. (2011). *"Aplikasi Algoritma Prim Untuk Menentukan Minimum Spanning Tree Suatu Graf Berbobot Dengan Menggunakan Pemrograman Berorientasi Objek"*. Jurnal Ilmiah Foristek. Vol 1. No 2. Hal 70-79.

[8] Amin, I Husni. (2014). "*Visualisasi Pohon Rentang Minimum Menggunakan Algoritma Kruskal dan Prim"*. Jurnal DINAMIKA TEKNIK. Vol 8. No 1. Hal 44-53.

[9] Raj P and Sood M. (2013). "*Ant Colony Optimazation is The Limited Case of Prim's Algorithm"*. International Journal of Computer Science and Information Technologies, ISSN:0975-9646. Vol.3(3), 4202-4204.

[10] Subadra N,. (Eds.). (2011). "*Directed Graph Algorithms for Tours- A Case Study"*. Journal of Emerging Trends in Engineering and Applied Sciences (JETEAS) 2(4); 615-618. Hall-615.