Nama: Hasna Azifatun Najwa

NIM : K1320039

CONTOH SOAL DAN PENYELESAIAN MASALAH GRAF MENGGUNAKAN ALGORITMA DIJKSTRA

Algoritma Dijkstra adalah salah satu algoritma yang digunakan untuk menyelesaikan masalah jarak terpendek *(shortest path problem)* pada sebuah graf yang terarah *(directed graph). Shortest path* merupakan persoalan untuk mencari lintasan terpendek antara dua buah *vertex* pada graph berbobot yang memiliki gabungan nilai jumlah bobot pada *edge graph* yang dilalui dengan jumlah yang paling minimum. Dalam bidang ilmu komputer, graf seringkali digunakan untuk memodelkan beragam permasalahan nyata untuk mendeteksi kondisi kebuntuan dalam sebuah sistem operasi dan bagaimana merencanakan rute yang efisien untuk transportasi. Dengan membuat sebuah simulasi pencarian jalur terpendek dengan menggunakan Algoritma Djikstra yang dapat membantu dalam pencarian jalur terpendek melalui skema yang telah ditetapkan dan digambarkan.

Algoritma Dijkstra atau bisa juga dikatakan sebagai algoritma Greedy merupakan algoritma yang lebih efisien dibandingkan algoritma Warshall untuk mencari lintasan terpendek, meskipun implementasinya juga lebih sukar. Secara sederhana, algoritma ini dapat menentukan jalur terpendek dari graf berbobot nilai positif, dari titik awal ke semua titik yang dikehendaki, sehingga nantinya akan ditemukan jalur terpendek dari titik awal dan titik tujuan.

Langkah langkah dalam penyelesaian jalur terpendek ini dengan menggunakan Algoritma Dijkstra secara umum dapat dituliskan sebagai berikut:

1. Membuat daftar jarak, daftar vertex sebelumnya, daftar vertex yang telah dikunjungi dan daftar vertex saat ini.
2. Semua diberikan nilai pada daftar jarak tidak terhingga kecuali vertex awal yang merupakan titik permulaan yang diberi nilai 0.
3. Vertex yang telah dikunjungi diberi nilai false.
4. Semua nilai pada vertex sebelumnya diberikan nilai khusus yang menyatakan belum terdefinisi, seperti null.
5. Vertex saat ini di-set menjadi vertex awal.
6. Markingvertex menjadi telah dikunjungi.

7. Update daftar jarak dan daftar vertex sebelumnya berdasarkan vertex mana yang dapat segera dikunjungi dari vertex saat ini.

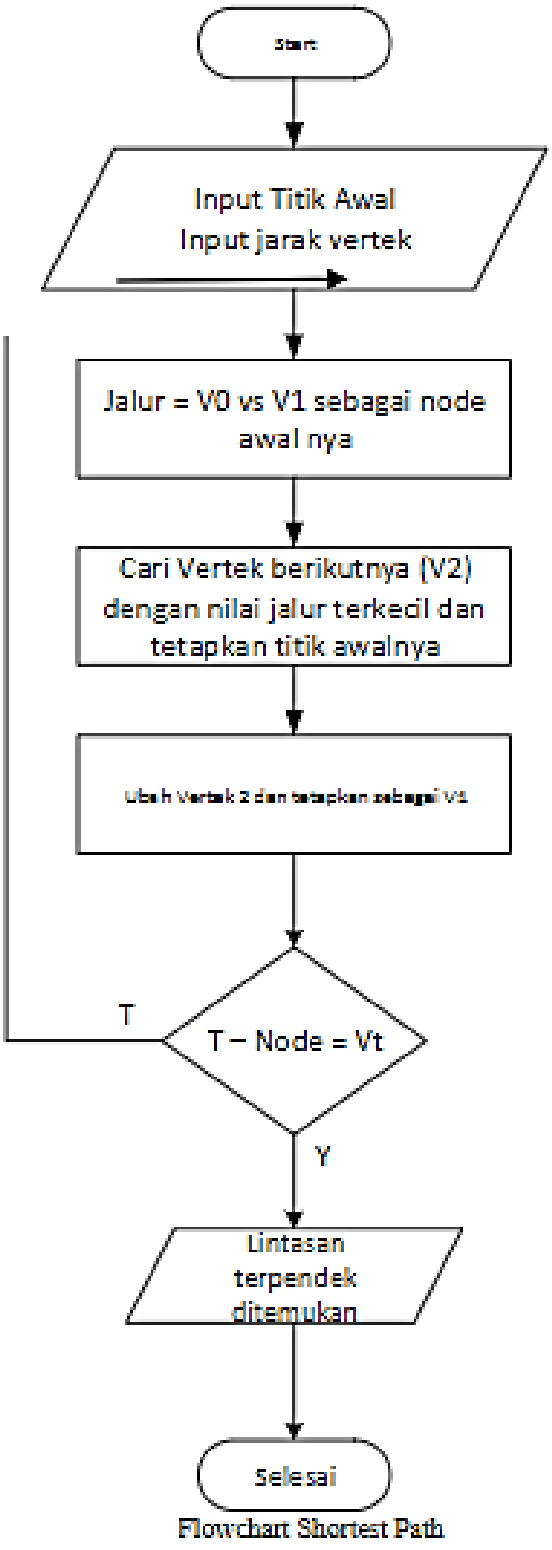
8. Perbarui vertex saat ini ke semua vertex yang belum dikunjungi yang dapat dicapai oleh shortest path dari vertex awal

9. Ulangi dari langkah 6 sampai semua titik dikunjung

Proses penyelasaian *shortest path*

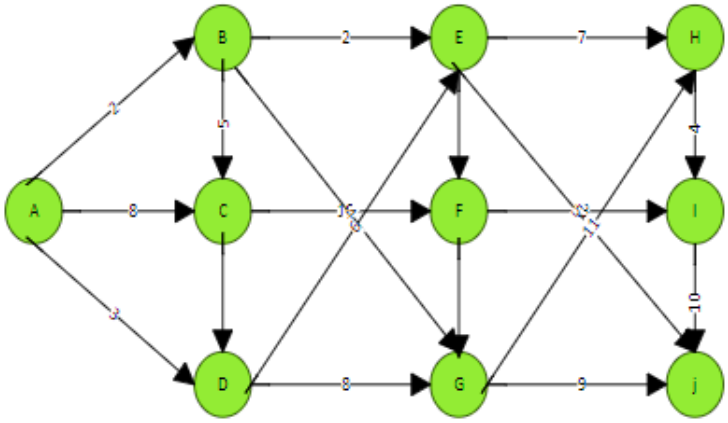
Alogirtma Sistem Geografis dengan Shortest Path ini dapat dituliskan sebagai berikut:

1. Mulai
2. Inisialisasi peta.
3. Membaca data seluruh node dan bobot edge pada database node dan jarak.
4. Input node awal dan node tujuan.
5. Memeriksa pada database jalur shortest path apakah query jalur dengan node awal dan node tujuan dari input langkah 3 telah tersimpan. Bila telah ada lanjut ke langkah
6. Melakukan proses algoritma Dijkstra dengan input node dari langkah 3, simpan hasil proses pada database jalur shortest path
7. Melakukan query data pada peta.
8. Menampilkan jalur terpendek pada peta.
9. Selesai.



Flowchart Shortest Path

Contoh graf yang akan diselesaikan dengan algoritma dijkstra



Langkah - langkah untuk menentukan jarak terpendek dari A ke J dengan menggunakan algoritma Dijkstra adalah sebagai berikut :

1. Pada awalnya status dari node yang belum terpilih diinisialisasikan dengan "0" dan yang sudah terpilih diinisialisasikan dengan "1" dimulai dari node A2.
2. Tentukan bobot dari node yang langsung berhubungan dengan node sumber yaitu node A, seperti : dari node A ke node B=2, dari node A ke node C=8, dari node A ke node D=3, dan untuk node E, F, G, H, I, J diinisialisasikan dengan "-" karena tidak ada lintasan (arc) yang menghubungkan secara langsung dengan node A3.
3. Predecessor (node sumber) dari node A, B, C, D adalah A, karena jarak dihitung dari node A, sehingga node A disebut Predecessor (node sumber), sedangkan untuk node F, G, H, I, J diinisialisasikan dengan "-" dikarenakan tidak ada lintasan (arc) yang langsung menghubungkan dari node A, sehingga jarak tidak ada.

Hasil Tabel Iterasi Dijkstra

Iterasi Ke 1										
Node	A	B	C	D	E	F	G	H	I	J
Status	1	0	0	0	0	0	0	0	0	0
Bobot	-	2	8	3	-	-	-	-	-	-
Prodecesor	A	A	A	A	-	-	-	-	-	-
Iterasi Ke 2										
Node	A	B	C	D	E	F	G	H	I	J
Status	1	1	0	0	0	0	0	0	0	0
Bobot	-	2	7	3	4	-	6	-	-	-
Prodecessor	A	A	B	A	B	-	B	-	-	-
Iterasi Ke 3										
Node	A	B	C	D	E	F	G	H	I	J
Status	1	1	0	1	0	0	0	0	0	0
Bobot	-	2	7	3	4	-	6	-	-	-
Prodecessor	A	A	B	A	B	-	B	-	-	-
Iterasi Ke 4										
Node	A	B	C	D	E	F	G	H	I	J
Status	1	1	0	1	1	0	0	0	0	0
Bobot	-	2	7	3	4	13	6	11	-	9

Prodec essor	A	A	B	A	B	E	B	E	-	E
Iterasi Ke 5										
Node	A	B	C	D	E	F	G	H	I	J
Status	1	1	0	1	1	0	1	0	0	0
Bobot	-	2	7	3	4	13	6	11	-	9
Prodec essor	A	A	B	A	B	G	B	E	-	E
Iterasi Ke 6										
Node	A	B	C	D	E	F	G	H	I	J
Status	1	1	1	1	1	0	1	0	0	0
Bobot	-	2	7	3	4	13	6	11	-	9
Iterasi Ke 7										
Node	A	B	C	D	E	F	G	H	I	J
Status	1	1	1	1	1	0	1	0	0	1
Bobot	-	2	7	3	4	13	6	11	19	9
Prodec essor	A	A	B	A	B	G	B	E	J	E
Iterasi Ke 8										
Node	A	B	C	D	E	F	G	H	I	J
Status	1	1	1	1	1	0	1	1	0	1
Bobot	-	2	7	3	4	13	6	11	15	9
Prodec essor	A	A	B	A	B	G	B	E	H	E
Iterasi Ke 9										
Node	A	B	C	D	E	F	G	H	I	J
Status	1	1	1	1	1	1	1	1	0	1
Bobot	-	2	7	3	4	13	6	11	15	9
Prodec essor	A	A	B	A	B	G	B	E	H	E
Iterasi Ke 10										
Node	A	B	C	D	E	F	G	H	I	J
Status	1	1	1	1	1	1	1	1	1	1

Bobot	-	2	7	3	4	13	6	11	15	9
Predecessor	**A**	**A**	**B**	**A**	**B**	**G**	**B**	**E**	**H**	**E**

Program akan berhenti karena semua node sudah terpilih. Sehingga akan menghasilkan jalur terpendek dari node A ke setiap node yang ada untuk melihat jalur mana yang terpilih dapat ditelusur dari *predecessor*-nya, sehingga akan terdapat (disamping):

A	➡	B	: A – B	: 2
A	➡	C	: A – B – C	: 7
A	➡	D	: A – D	: 3
A	➡	E	: A – B – E	: 4
A	➡	F	: A – B – E – F	: 13
A	➡	G	: A – B – G	: 6
A	➡	H	: A – B – E – H	: 11
A	➡	I	: A – B – E – H – I	: 15
A	➡	J	: A – B – E – J	: 9