SinkrOn
JURNAL & PENELITIAN TEKNIK INFORMATIKA

Jurnal & Penelitian Teknik Informatika
Volume 2 Nomor 2, Oktober 2017

e-ISSN : 2541-2019
p-ISSN : 2541-044X

Pencarian Jalur Terpendek dengan Algoritma Dijkstra

Muhammad Khoiruddin Harahap
Politeknik Ganesha Medan
Jl.Veteran No. 194 Manunggal
choir.harahap@yahoo.com

Nurul Khairina
Politeknik Ganesha Medan
Jl.Veteran No. 194 Manunggal
nurulkhairina27@gmail.com

Abstrak *—* Persoalan dalam menemukan jalur terpendek seiring dengan penghematan waktu yang tersingkat. Hal ini menjadi penting dalam kedinamisan masyarakat perkotaan. Jumlah rute yang ditempuh juga menjadi persoalan tersendiri untuk mencapai tempat tujuannya. Kita akan menentukan titik-titik manakah yang harus dilalui sehingga mendapatkan tempat tujuan dengan jarak terpendek dan penggunaan waktu yang tersingkat dengan menggunakan algoritma Dijkstra. Pencarian lintasan terpendek merupakan persoalan optimasi. Nilai pada sisi *graph* bisa dinyatakan sebagai jarak antar kota. Lintasan terpendek bisa diartikan sebagai proses minimalisasi bobot pada lintasan. Untuk mengatasi permasalahan itu maka diperlukan adanya suatu simulasi yang dapat membantu menentukan jalur terpendek. Algoritma Dijkstra bisa juga dikatakan sebagai algoritma Greedy yang pada pembahasan ini mampu memudahkan kita mencari jalur rute terpendek dan menjadi lebih efekfif.

Kata Kunci *— Dijkstra, Greedy, Jalur terpendek, Optimasi waktu*.

I. PENDAHULUAN

Penentuan sebuah jalur terpendek merupakan hal yang penting dan dibutuhkan sehubungan dengan optimasi waktu yang digunakan serta beberapa penghematan dibidang lainnya. Dengan jalur terpendek yang dilalui, membuat pekerjaan lebih efektif, cepat dan dapat tentunya terjadi penghematan biaya. Jalur terpendek dapat diartikan sebagai nilai minimal dari suatu lintasan, yaitu jumlah nilai dari keseluruhan bentuk lintasan. Untuk membantu menentukan lintasan perpendek dapat memilih jalur yang terpendek dari tempat asal ke tujuan. Hal ini terkadang tidak dapat membantu secara maksimal dikarenakan banyaknya jumlah jalan yang harus dipilih dan tidak dapat diperkirakan jarak tempuh pada jalur itu.

Untuk itu diperlukan sebuah sistem untuk membantu melakukan pencarian lintasan terpendek yang dapat merepresentasikan data [5]. Data tersebut dapat disimpan, diolah dan disajikan dalam bentuk yang lebih sederhana sehingga memudahkan dalam menentukan lintasan terpendek. Algoritma Djikstra yang juga merupakan algoritma Greedy bisa menentukan lintasan terpendek.

Penelitian ini adalah membuat sebuah simulasi pencarian jalur terpendek dengan menggunakan Algoritma Djikstra yang dapat membantu dalam pencarian jalur terpendek. Dengan skema yang telah ditetapkan dan digambarkan.

II. LANDASAN TEORI

A. Pengertian *Graph*

Graph adalah struktur diskrit yang terdiri dari adanya simpul (*vertex)* dan adanya sebuah sisi (*edge)*, *graph* adalah pasangan himpunan (V,E) dimana V merupakan sebuah himpunan yang tidak kosong dari sebuah *vertex* dan E adalah himpunan sisi yang menghubungkan sepasang simpul dalam graph tersebut [2].

Jika dilihat arah pada sisi, secara umum *graph* dapat dibedakan menjadi dua jenis, yaitu:

1. Graph Tak Berarah
   Graph tak berarah adalah graph yang sisinya tidak memiliki arah. Graph tak berarah, urutan pasangan simpul yang dihubungkan oleh sisi diabaikan. Maka, bisa disebut
   (vj, vk) = **(vk, vj)**
   adalah sisi yang sama.

2. Graph Berarah
   Graph berarah adalah graph yang sisinya mempunyai orientasi arah yang jelas. sisi yang berarah disebut dengan busur (*arc)*. Pada graph berarah, **(vj, vk)** dan **(vk, vj)** dinyatakan 2 buah busur arah yang berbeda,atau dengan kata lain :
   (vj,vk) ≠ (vk,vj).

Jurnal & Penelitian Teknik Informatika
Volume 2 Nomor 2, Oktober 2017
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Busur **(vj,vk)**, simpul **vj** dinamakan simpul asal (*initial vertex)* dan simpul vk disebut dengan simpul terminal (*terminal vertex*) [6].

B. Jalur Lintasan Terpendek *(Shortest Path)*

Kata Terpendek pada persoalan sebuah lintasan memiliki pengertian yaitu proses minimalisasi bobot pada sebuah lintasan graph.

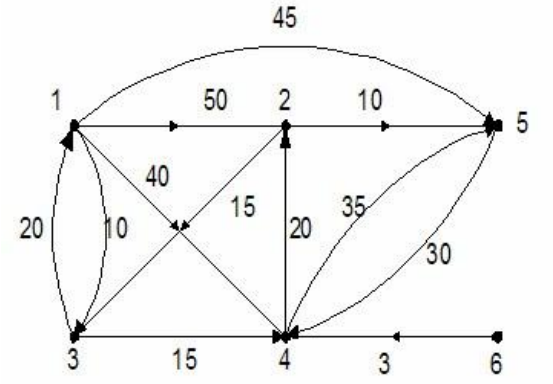
Beberapa jenis persoalan lintasan terpendek antara lain:

a. Lintasan terpendek antara dua buah simpul.
b. Lintasan terpendek antara semua pasangan simpul.
c. Lintasan terpendek dari simpul tertentu ke semua simpul yang lain.
d. Lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu [1].

Dalam pemecahan persoalan tentang graf berbobot

$$G = (V,E)$$

dan sebuah simpul a. Penentuan lintas terpendek dari a ke setiap simpul lainnya di G. Asumsi yang kita buat adalah bahwa semua sisi berbobot positif perhatikan Gambar 2.1 dibawah ini :



Gambar 1. Graph contoh persoalan lintasan terpendek
Source : https://rahadikusuma.blogspot.co.id

C. Definisi Algoritma

Kata algoritma (*algorithm*) diambil dari kata algorism yang berasal dari nama seorang ilmuwan dari Arab yang terkenal yaitu abu ja'far Muhammad Ibnu musa Al Khuwarizmi dan oleh orang barat dilafalkan menjadi Algorism. Dan pengertian algorism itu sendiri adalah sekumpulan instruksi atau perintah yang dibuat secara jelas dan sistematis berdasarkan urutan yang logis untuk penyelesaian suatu masalah [3].

1. Komponen Algoritma

Algoritma memiliki 5 komponen urutan yaitu *finiteness* (terbatas), *definiteness (*Kepastian), *input* (masukan), *output (*keluaran*)*, dan *effectiveness* (efektivitas). Dalam merancang algoritma ada tiga komponen yang harus dimiliki yaitu :

a. Komponen Masukkan(*Input*)
b. Komponen Keluaran (*Output*)
c. Komponen Proses (*Processing*)

2. Notasi (Tata cara penulisan) Algoritma

Penulisan algoritma terkadang sulit untuk dipahami dan maksud dari algoritma tersebut. Selain itu juga sulit untuk menuliskan algoritmanya. Agar mempermudah bisa dilakukan notasi – notasi algoritma. Notasi algoritma merupakan rancangan penyelesaian masalah (algoritma) yang dituliskan dalam notasi (tata cara penulisan).

Notasi algoritma yang sering dijumpai ada 3 macam yaitu [4] :

1. diagram alir (*flowchart*),
2. deskriptif
3. *pseudo-code*

3. Aturan Penulisan Teks Algoritma

Algoritma merupakan deskripsi langkah - langkah penyelesaian dari sebuah permasalahan. Deskripsi tersebut bisa dituliskan ke dalam bentuk notasi apapun, yang penting mudah dibaca dan dipahami. Tidak ada notasi yang baku dalam penulisan teks algoritma. Tiap orang dapat membuat aturan penulisan dan algoritma sendiri. Namun, agar notasi algoritma dapat dengan mudah ditranslasi ke alam notasi bahasa pemrograman, maka sebaiknya notasi algoritma itu berkoresponden dengan notasi bahasa pemrograman secara umum. Teks algoritma dinotasikan ke dalam tiga bagian (blok) : judul (header) agoritma, deklarasi, dan deskripsi.

D. Algoritma *Dijkstra*

Algoritma Dijkstra merupakan algoritma yang digunakan untuk menentukan jarak terpendek dari satu *vertex* ke *vertex* yang lainnya pada suatu graph berbobot, jarak antar vertex adalah nilai bobot dari setiap *edge* pada graph. Suatu bobot harus bernilai posisif (**bobot >= 0**). Algoritma Dijkstra ditemukan oleh Edger Wybe Dijkstra. Algoritma Dijkstra dikenal juga sebagai algoritma greedy yaitu algoritma yang penyelesaian masalah dengan mencari nilai maksimum. Cara kerja algoritma Dijkstra dalam pencarian jarak terpendek adalah perhitungan dari vertex

Jurnal & Penelitian Teknik Informatika
Volume 2 Nomor 2, Oktober 2017
e-ISSN : 2541-2019
p-ISSN : 2541-044X

asal ke vertex terdekatnya, kemudian ke vertex yang kedua, dan seterusnya [7].

III. Analisa Masalah Umum

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 atau dapat dinyatakan juga sebagai berikut :
Diberikan sebuah graf berbobot (dengan himpunan *vertex* V, himpunan *edge* E, dan fungsi bobot bernilai bilangan *real* yang dapat ditulis dengan :

$$f : E \rightarrow \mathbf{R}$$

dan diberikan elemen v' dari V , sehingga dapat dicari sebuah lintasan P dari v ke setiap v dari V.

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. *Marking vertex* 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 dikunjungi.

A. Proses penyelesaian *Shortest Path*

Pada proses Tentukan *Shortest Path* membutuhkan data masukan berupa data asal dan data akhir, dan kemudian proses akan mengambil data dari *database node* dan jarak. Lalu sistem akan melakukan pemeriksaan pada *database* jalur *shortest path*, bila data awal dan data akhir yang dipilih sudah pernah diproses sebelumnya, maka akan diambil hasil proses dari *database* jalur *shortest path*, apabila belum pernah, maka sistem akan memproses data tersebut, selanjutnya sistem akan menyimpan hasil perhitungan dan *query* jalur *shortest path* pada *database* jalur *shortest path*. Masing-masing entitas data yang tercantum pada DFD shortest part ditampilkan pada tabel di bawah ini.

Tabel 1. Entitas Data Pada Shortest Path

Nama	Keterangan
Data Asal	Node yang dipilih sebagai vertex asal shortest path
Data tujuan	Node yang dipilih sebagai vertex akhir shortest path
Jalur Shortest Path	Hasil perhitungan shortest path

Secara garis besar, algoritma Sistem Informasi 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.
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.

Jurnal & Penelitian Teknik Informatika
Volume 2 Nomor 2, Oktober 2017
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Gambar 2. Flowchart Shortest Path

IV. HASIL PENELITIAN

A. Analisa Sistem Algoritma *Dijkstra*

Aplikasi penerapan algoritma dalam penelitian ini dirancang dengan menggunakan bahasa pemograman Vb.Net. Dimana cara mencari jalur terpendek akan memperhatikan beberapa factor, diantaranya adalah :

1. Input Graph

Sebuah graph dapat didefenisikan sebagai pasangan himpunan (V,E) dimana V=himpunan verteks {v1,v2,v3,……vn} dan yang menghubungkan antara satu verteks dengan verteks lain diberi nilai dengan himpunan edge (arc) dan dapat ditulis dengan V{e1,e2….., en}. Berdasarkan orientasi arah pada sisi, graph dapat dibedakan atas dua jenis, yaitu :

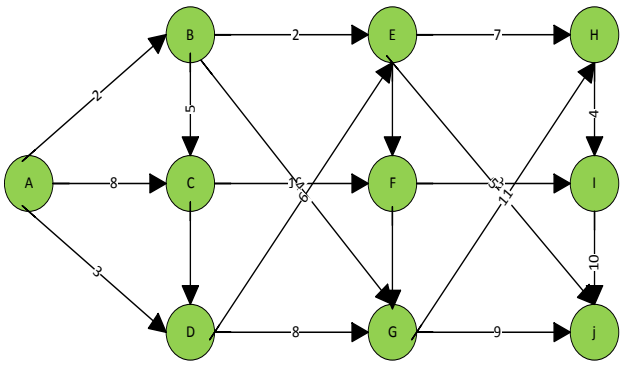
1. Graph Berarah (directed graph atau digraph)
2. Graph tak berarah (Undirected graph).

Selain itu kita mengenal juga dengan Graph berbobot (Weight Graph) dan Graph tak berbobot. Jika graph yang diberikan adalah graph berbobot, maka elemen matriks yang terhubung antara vertex adalah bobot graph. Dalam aplikasi yang dirancang, maka graph harus diinput dan ditentukan sesuai peta lokasi yang telah di analisa dan diamati.

2. *Proses Graph*

Pada algoritma Dijkstra node digunakan karena algoritma Dijkstra menggunakan diagram pohon *(tree)* untuk penentuan jalur lintasan terpendek dan menggunakan *graph* yang berarah. Algoritma Dijkstra mencari jarak terpendek dari node asal ke nodel terdekatnya, kemudian ke node berikutnya, dan seharusnya. Secara umum sebelum dilakukan I iterasi, algoritma sudah mengidenfikasi jarak terdekat dari i-1 node terdekatnya. Jika seluruh node berbobot tertentu yang (positif), maka node terdekat berikutnya dari node asal dapat ditemukan selama node berdekatan dengan node Ti. Kumpulan node yang berdekatan dengan node di inilah yang merupakan kandidat dari algoritma Dijikstra untuk memilih node berikutnya dari node asal.

Contoh dari graph yang akan diselesaikan dengan algoritma Dijkstra adalah sebagai berikut :



Gambar 3. Graph Untuk 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 A
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 A
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.

Tabel 2. Hasil 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	-	-	-	-	-	-
Predecessor	A	A	A	A	-	-	-	-	-	--

Dari penelusuran jarak terpendek dari algoritma Dijkstra ini maka di dapat hasil seperti tabel berikut :

Tabel 3. Hasil Iterasi Algoritma 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
Prodecessor	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
Prodecessor	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
Prodecessor	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
Prodecessor	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
Prodecessor	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 :

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

Jurnal & Penelitian Teknik Informatika
Volume 2 Nomor 2, Oktober 2017
e-ISSN : 2541-2019
p-ISSN : 2541-044X

V. KESIMPULAN

Dari penelitian dan hasil implementasi algoritma Dijkstra berdasarkan lintasannya maka dapat diambil kesimpulan bahwa :

1. Dalam penentuan jalur terpendek Algoritma Dijkstra beroperasi dari node awal atau sumber ke node tujuan. Dimana masing-masing node memiliki nilai jarak yang telah ditetapkan.
2. Dengan menggunakan metode Dijkstra ini, dapat ditentukan jalur terpendek dari sebuah jalur perjalanan dengan menentukan vertex awal dan vertex tujuan serta membandingkan nilai vertex tersebut.

REFERENSI

[1] B.A. Pradhana." Studi dan Implementasi Persoalan Lintasan Terpendek Suatu Graf Dengan Algoritma Dijkstra dan Algoritma Bellman-Ford". Bandung. 2007.
[2] J. Daud. "Studi Efektifitas Penggunaan jalan Kota Medan". Jurnal Sistem Teknik Industri, No. 3 Vol. 6. 2005.
[3] L. Fanani, E.M. Adams, S. Wicaksono, A. Satrio. 2012. "Rancang Bangun Aplikasi Web Pencarian Rute Terpendek Antar Kota Medan Algoritma Dijsktra". Jurnal Basic Science And Techonology, 1(3),7-11, Malang. 2012.
[4] Iryanto. "Pengantar Teori dan Aplikasi Dan Diagram DFD". Medan: USU Press. 2003
[5] R. Munir. "Deskripsi Sistem. Edisi ke-2". Bandung: Informatika. 2003.
[6] M.R. Rizaldi, "Pencarian Jalur Terpendek dalam GPS dengan Menggunakan Teori Graf". Universitas Yogyakarta. 2007
[7] R. Saputra. "Sistem Informasi Pencarian Obyek Kota Medan Dengan Algoritma Dijkstra." Jurnal Matematika, No. 1, Vol.14, Hal 19-24, April 2011.