PERBANDINGAN ALGORITMA DIJKSTRA DAN ALGORITMA BELLMAN-FORD PADA JARINGAN GRID

SKRIPSI

MICHI PURNA IRAWAN
07 134 059

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS ANDALAS
PADANG 2011

TANDA PERSETUJUAN SKRIPSI

Dengan ini dinyatakan bahwa:

Nama	:	Michi Purna Irawan
No. Buku Pokok	:	07134059
Jurusan	:	Matematika
Bidang	:	Matematika Terapan
Judul Skripsi	:	Perbandingan Algoritma Dijkstra Dan Algoritma Bellman-Ford Pada Jaringan Grid.

telah diuji dan disetujui skripsinya sebagai salah satu syarat untuk memperoleh gelar Sarjana Sains (S.Si) melalui ujian sarjana yang diadakan pada tanggal 08 Juli 2011 berdasarkan ketentuan yang berlaku.

Pembimbing/Penguji

1.

Budi Rudianto, M.Si

NIP. 132 169 920

Penguji

1.

Narwen, M.Si

NIP. 19671004 199609 1 001

2.

Efendi, M.Si

NIP. 19780717 200212 1 002

Mengetahui, Ketua Jurusan Matematika
FMIPA Universitas Andalas

Dr. Syafrizal Sy

NIP. 19670807 199309 1 001

بسم الله الرحمن الرحيم

"Ya Allah ya Tuhanku, berilah aku ilmu pengetahuan dan masukanlah aku kedalam orang-orang yang saleh. Dan jadikanlah bagiku pujian yang benar dalam (generasi) yang kemudian".

(QS : Asy-Syu'araak 83:84)

Pendidikan bukan lah sesuatu yg diproleh seseorang, tapi pendidikan merupakan suatu proses seumur hidup yang hebat didunia ini. Bukan tempat dimana kita berada melainkan arah yang kita tuju. Arah yang diberikan pendidikan untuk mengawali hidup seseorang dalam menentukan masa depannya.

Alhamdulillah.....
Kupersembahkan karya ini sebagai ungkapan terima kasih dan baktiku terhadap pengorbanan, pengertian, keihklasan, kasih sayang, cinta dan do'a.....

Thanks to:

Ayahanda Ripudin n ibunda Liasma tercinta,,.ku persembahkan semua ini utk klian, untuk tetes keringat, harapan, n keinginan yang kalian bangun, n untuk ratusan kalinya beban pikiran yang tercipta karna demi masa depanku, ini adalah kado pertama untuk kalian, karna ku tahu,ini masih sangat belum apa-apa,,,Ku kn slalu masih mharapkan doa kalian agar ku bisa memberikan kado kedua dan seterusnya tanpa henti yang masih tak terlihat ke depan,.

Buat adik2ku tersyank Royani, Okta, Mintan, ANggi belajarlah yg rajin y...??kakak kn do'akn kalian smoga dpat menyusul seperti kakak.

Ka2k berkrap smoga apa yg tlah ka2k lwati ini akan mnjdi contk, motivasi, n inspirasi klian untk mnuntun k mask dpan.

Buat BIG MY FAMILY wak Ica lanang, wak Ica tine cept smbuh wak, ayuk Ica, trimaksih ats motivasi n dukngan y..!!Arif...gmana tu brooo...da slesai blom kuliah nye....?dengr2 kte nye...Ih ndak slesai ple.....!!y...mdah2an lah cpat slesai jdi

dokter....!!Aminnn. Wawan lukmne titu kan...!Ih ade blom rncane nye..??mamang ni Ih slesai tini..!!klu Ih kh ndulu hehehe.....trus Deplomanye komputernye ktenye Ih ndak slelsai pule thun 2011 ni...smangat sukses slalu buat kakanda wawansyah....!!Aminnn..

thanks to :

Bwt (ayah made) yg slalu membrikan motivasi, smngat, dan kedisiplinan kpada anak2 Basic Science terutama Jurusan Matematika hnya ucapan trimaksih ini pk yg bsa michi ucapkan....!!smoga atas smua yg tlah bpak brikan slama ini akan mnjdi amal ibadah dan akn mndpatkan balasan disisi Tuhan.

Bwt (community my Class Basic Science)Ansori, Azhari, susti, filly, Sepni, Septi, Elva, dwi, Leksi teman2 yg dri kab.Kaur sekses buat kta smua smoga jd guru yg baik....aminnn, slanjtnya mas Agus , Rodi, pk wo (Yogi), Fajar (muka bakwan) hehe..., Markas, Insaf, Angga, Suhardi, Amril, Ampuni(punek waruwek), Sulaiman, Meiman, Agus, Dodi (pk ktua), Nofrisal, Subrata, Santi, Suji, Wina, Fifiti(Fivti emooeet.co.id), Herlin, Uci, Upi, smua y dehhhh.....bnyak tuh lw mo d sbutn pkok y suskses aja lh buat kta smua...smpai ktemu lg UNP y....☺☺ selama dibangku perkuliahan, kalian smua berkesan bnget dihati kyu..... dan akyu minta maaf atas segala kesalahan yang pernah kuperbuat, bukan maksud menyakiti and disengaja tapi kita manusia yang tak luput dari kekhilafan ...☺☺☺

Bwt klurga besar ibuk fatan (buk Mar) smoga shat selalu dn dlam lndunganNYA..aminmm dn tokoh berkahnya jg...smoga usbanya trus maju y buk..!yg slalu membrikan pengrtian dlam keada2n pd akhir blan, (ngutang) hehehe....michi ucpkn trima ksih bnyak y buk...!!smua itu tk kan terlpakan rsa kekluargaan, kbaikan n nsa2 kbersamaan dg kluarga ibuk michi mrasa sneng bsa knal n dkat dngan smua kluarga ibuk slama michi kuliah d UNAND skali lg michi ucapin ma ksih bnyak buk....tnpa smua itu mungkn smua y tk kn sperti skrang ini.....!!

Bwt nda Yetty...maksih y...atas dkungan materil y.....!!smoga itu smua akn mnjdi awl yg baik untk klanjutn rncna dn keinginan diantra kta...☺☺

Dengan Ilmu Kehidupan Mnjdi Mudah
Dengan Seni Kehidupan Mnjdi Indah
Dengan Agama Khidpan Mnjdi Terang

With love

*^^Michi Purna Irawan^^*

KATA PENGANTAR

Syukur Alhamdulillah, segala puji bagi ALLAH SWT yang selalu melimpahkan rahmat dan karunia-NYA dalam menuntun penulis untuk menyelesaikan penulisan skripsi ini yang berjudul **Perbandingan Algoritma Dijkstra Dan Algoritma Bellman-Ford pada Jaringan Grid**. Skripsi ini diajukan sebagai syarat untuk menempuh ujian sarjana pada Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Andalas.

Dengan selesainya skripsi ini perkenankanlah penulis menyampaikan ucapan rasa terima kasih yang sebesar-besarnya kepada bapak Budi Rudianto, M.Si selaku pembimbing yang telah meluangkan waktu dan memberikan bimbingan, arahan, petunjuk serta bantuan, mulai dari penentuan judul hingga sampai skripsi ini selesai.

Selanjutnya izinkanlah pada kesempatan ini penulis mengucapkan rasa ucapan terima kasih kepada :

1. Ibunda Liasma dan Ayahanda Ripudin tercinta serta adik-adik tersayang Royani, Okta Kurniawan, Mintan Halimas dan Delia Nuranggela terima kasih atas segala perhatian dan dukungan baik moril maupun materil kepada penulis selama penyusunan skripsi ini.
2. Bapak Drs. Syafrizal Sy, M. Si selaku ketua Jurusan Matematika dan Bapak Prof. Dr. I Made Arnawa, M. Si selaku Pembimbing Akademik dan koordinator Basic Science Jurusan Matematika Dan Ilmu Pengetahuan Alam Universitas Andalas.
3. Bapak Narwen, M. Si dan Bapak Efendi, M. Si selaku dosen penguji, terima kasih atas kritikan, saran dan solusi yang diberikan.

4. Seluruh Staf Pengajar jurusan Matematika Bapak dan Ibu dosen yang telah memberikan ilmu pengetahuan, semoga bekal ilmu yang bapak dan ibu berikan kepada penulis akan menjadi amal dan selalu mendapat hidayah dari ALLAH SWT.
5. Semua yang telah membantu penulis selama menempuh pendidikan di Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Andalas.

Dengan hati terbuka penulis menerima kritikan, saran yang membangun untuk kesempurnaan skripsi ini, karena penulis menyadari skripsi ini masih jauh dari kesempurnaan. Meskipun demikian penulis berharap semoga skripsi ini dapat menjadi inspirasi untuk peneliti selanjutnya. Akhirnya, kepada pembaca penulis mohon maaf atas segala kekurangan, kesalahan dan kekeliruan dalam penulisan dan kepada Allah penulis mohon ampun dan hidayah-NYa, Amin.

Padang, Juli 2011

Penulis

ABSTRAK

Jaringan Grid adalah suatu kumpulan (*resource*) atau sumber (mesin, CPU, memori) yang saling berkomunikasi satu sama lain, dengan menggunakan cara-cara (*protocol*) tertentu. *Grid Computing* adalah infrastruktur komputasi yang menyediakan akses berskala besar terhadap sumber daya komputasi yang tersebar secara geografis yang saling terhubung menjadi satu kesatuan fasilitas. Dapat direpresentasikan dalam betuk graf. Algoritma *routing* yang dibahas adalah algoritma Dijkstra dan algoritma Bellman-Ford. Analisis algoritma untuk mengetahui kompleksitas pada kedua algoritma tersebut.

Kata kunci: Graf, Jaringan Grid, Algoritma routing, Kompleksitas algoritma, Algoritma Dijkstra, Algoritma Bellman-Ford.

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman


UNIVERSITAS ANDALAS
UNTUK KEDJAJAAN BANGSA

BAB I
PENDAHULUAN

1. 1 Latar Belakang

Seiring berkembangnya ilmu pengetahuan dan teknologi dewasa ini, dalam kehidupan sehari-hari kebutuhan akan mengakses suatu informasi dan mengirim suatu paket data tidaklah asing lagi pada suatu jaringan komputer. Akan tetapi jaringan terus berkembang yang tadinya hanya difokuskan pada jaringan komputer saja, maka diperluas menjadi bermacam sumber daya. Sumber daya yang dimaksud adalah banyaknya informasi yang dapat diakses dalam suatu jaringan. Jaringan tersebut adalah jaringan *Grid.* Suksesnya perancangan dan implementasi *closter computing* pada satu lokasi (*node*), maka para peneliti memperluas ide *closter computing* sebagai sumber daya yang tadinya hanya difokuskan pada komputer saja, maka diperluas menjadi bermacam sumber daya pengetahuan pada jaringan *grid computing*.

Closter computing adalah kumpulan beberapa domain komputer dari suatu lokasi yang terhubung dengan domain komputer lokasi lainnya. Keterhubungan antar domain komputer ini terhubung dengan menggunakan tombol (*switch*) dan pintu masuk (*gate*). Kemudian masing-masing lokasi (*node*) akan dihubungkan menjadi beberapa gabungan lokasi, lokasi tersebut tersebar secara luas dalam suatu wilayah, negara bahkan antar negara. Sehingga secara umum *grid computing* banyak didefinisikan oleh berbagai peneliti namun demikian terdapat konsep yang sama. *Grid computing* adalah imprastruktur yang menyediakan akses berskala besar terhadap sumber daya komputasi yang tersebar secara geografis yang terhubungan menjadi satu kesatuan fasilitas.

Sumber daya ini termasuk antara lain super komputer, system penyimpanan (*storage*) sumber-sumber data, dan *instrument-instrument* atau perangkat lunak lainnya. Suatu graf G yang terdiri dari *closter* (C_k) yang di hubungkan oleh *gate* (Gt_k), dimana $k \in \{0,\ldots,G\text{-}1\}$ untuk setiap *closter* terdiri sejumlah *Network* (Nj_k) yang dihubungkan *Switch* (SWj_k) dan setiap lokasi terdiri dari beberapa *Processing Element* ($PEij_k$) yang terhubung dalam suatu lokal domain seperti *LAN* (*Lokal Area Network*).

Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Graf sering digunakan untuk memodelkan jalur transportasi, penjadwalan, jaringan komputer dan lain sebagainya.

1.2 Perumusan Masalah

Masalah yang akan dibahas adalah kompleksitas yang dibutuhkan oleh algoritma *Dijksta* dan algoritma *Billman-Ford* untuk sampai kesimpul tujuan pada jaringan *Grid*.

1.3 Pembatasan Masalah

Dalam tulisan ini penulis membatasi permasalahannya pada algoritma *Dijkstra* dan algoritma *Bellman-Ford* untuk menentukan kompleksitas kedua algoritma tersebur pada jaringan *Grid*.

1.4 Tujuan Penelitian

Penelitian ini bertujuan untuk mengetahui kompleksitas ruang dan kompleksitas waktu antara algoritma *Dijkstra* dan algoritma *Bellman-Ford* untuk penerapan pada jaringan *Grid*.

1.5 Manfaat Penelitian

Penelitian ini diharapkan kepada pembaca dan penulis tentunya dapat menjadi tambahan wawasan pengetahuan tentang spesifikasi dari kedua algoritma yang dibahas.

1.6 Sistematika Penulisan

Sistematika penulisan ini terdiri dari 4 bab, yaitu :

BAB I : Pendahuluan

Pada BAB ini menjelaskan tentang latar belakang, rumusan masalah, batasan masalah, manfaat penilisan, tujuan penulisan, serta sistematika penulisan.

BAB II : Landasan Teori

Pada BAB ini menjelaskan tentang teori-teori yang melandasi dan berkaitan pada BAB sebelumnya.

BAB III : Pembahasan

Pada BAB ini akan membahas tentang kompleksitas algoritma *Dijkstra* dan algoritma *Bellman-Ford* pada jaringan *Grid*.

BAB IV : Penutup

Pada BAB ini berisi kesimpulan dan saran yang diperoleh dari pembahasan masalah pada BAB sebelumnya.


UNIVERSITAS ANDALAS
UNTUK KEDJAJAAN BANGSA

BAB II

LANDASAN TEORI

Pada bab ini akan dijelaskan teori-teori yang menjadi landasan dalam pembahasan untuk menyelesaikan dan mengetahui kompleksitas kedua algoritma yaitu algoritma *Dijkstra,* algoritma *Bellman-Ford* serta beberapa definisi yang mempunyai hubungan secara teoritis pada algorima *Dijkstra* dan algoritma *Bellman-Ford* yang digunakan pada jaringan *Grid.*

2.1 Graf

2.1.1 Definisi Graf

Graf G didefinisikan sebagai pasangan himpunan (V, E), ditulis dengan notasi $G = (V, E)$, yang dalam hal ini V adalah himpunan tidak kosong dari simpul-simpul (*vertices*) dan E adalah himpunan sisi (*edges*) yang menghubungkan sepasang simpul.

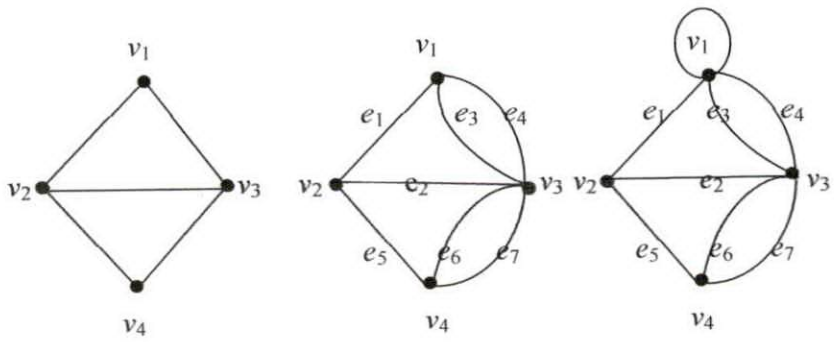
2.2 Jenis-jenis Graf

2.2.1 Graf Sederhana (*Simple Graph)*

Graf sederhana adalah graf yang tidak mengandung *loop* atau gelang maupun sisi ganda. Pada graf sederhana, sisi adalah pasangan yang tak-terurut (*unorderet pairs*). Jadi, untuk menuliskan sisi (u, v) sama saja (v, u).

2.2.2 Graf tak-berarah (*undirected graph*)

Graf yang sisinya tidak mempunyai orientasi arah disebut graf tak-berarah. Dapat dilihat seperti gambar berikut yang merupakan gambar graf tak-berarah:

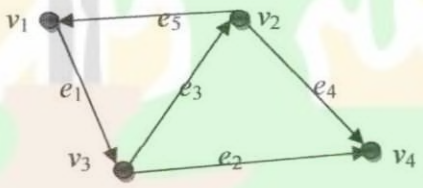


Gambar 2.2.2.1 Graf tak-berarah.

Gambar graf paling kiri merupakan graf sederhana. Gambar graf tengah merupakan gambar graf ganda karena memiliki sisi ganda, sedangkan gambar graf paling kanan merupakan graf semu karena memiliki gelang atau kalang, yakni sisi yang berawal dan berakhir pada simpul yang sama.

2.2.3 Graf berarah (*directed graph* atau *digraph*)

Graf yang setiap sisinya diberikan orientasi arah disebut sebagai graf berarah.



Gambar 2.2.3.1 Graf Berarah.

2.2.4 Graf Berbobot (*Wheighted Graph*)

Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga bobot (*weight*).

2.3 Terhubung (*Connected*)

Dua buah simpul dalam graf G dikatakan terhubung jika terdapat lintasan misalkan simpul u dan simpul v maka pasti simpul yang pertama dapat mencapai

pada simpul yang kedua. Dua buah simpul terminal pada jaringan *Grid* hanya dapat berkomunikasi jika terhubung.

2.4 Bersisian (*Incident*)

Untuk sebarang sisi $e = (u, v)$ sisi e dikatakan bersisian dengan simpul u dan v.

2.5 Bertetangga (*Adjacent*)

Dua buah simpul pada graf G dikatakan bertetangga jika keduanya terhubung langsung dengan sebuah sisi.

2.6 Lintasan (*Path*)

Lintasan yang panjangnya n dari simpul awal v_0 ke simpul tujuan v_n di dalam graf G adalah barisan berselang-seling simpul-simpul dan sisi-sisi 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 sisi dari graf G. Sebuah lintasan dikatakan sederhana (*simple path*) jika semua simpulnya berbeda dan setiap sisi yang dilalui hanya satu kali. Lintasan yang berawal dan berakhir pada simpul yang sama disebut lintasan tertutup (*Closed path*), sedangkan lintasan yang tidak berawal dan berakhir pada simpul yang sama disebut lintasan terbuka (*open path*). Pada tulisan ini akan dibahas untuk lintasan terbuka (*open path*).

2.7 Algoritma

Urutan logis langkah-langkah penyelesaian masalah yang disusun secara sistematis disebut algoritma.

2.7.1 Algoritma Routing

Routing adalah proses untuk meneruskan paket data dari suatu simpul ke simpul yang lainnya dengan berdasarkan asas jalur terpendek. Jadi dari penjelasan tersebut algoritma *Routing* adalah prosedur atau langkah-langkah dalam menjalankan suatu perintah tertentu untuk meneruskan suatu paket data tertentu dari satu simpul ke simpul lainnya dengan asas jalur terpendek.

2.7.2 Tujuan Algoritma Routing

Tujuan dari algoritma *routing* adalah agar pengiriman paket suatu data tersebut dapat dikirimkan dari simpul awal ke simpul tujuan secara efektif dan efisien berdasarka asas jalur terpendek.

2.8 Algoritma Dijkstra

Algoritma *Dijkstra* adalah sebuah algoritma dalam memecahkan permasalahan jarak terpendek (*shortest path problem)* untuk sebuah graf berarah (*directed graph*) dengan bobot-bobot sisinya (*edgeweights)* yang bernilai tak-negatif. Pada tahun 1959 sebuah tulisan sepanjang tiga halaman yang berjudul *A Note on Two Problems in Connexion with Graphs* diterbitkan pada jurnal *Numerische Mathematik*. Seorang ilmuan komputer berkebangsaan Belanda yang bernama *Edsger W. Dijkstra* seorang ilmuwan komputer berumur dua puluh

sembilan tahun mengusulkan algoritma-algoritma untuk solusi dari dua masalah teoritis graf dasar yaitu *the minimum weight Algoritma Dijkstra* untuk masalah jalan terpendek adalah satu dari algoritma-algoritma paling ternama pada ilmu komputer dan sebuah algoritma paling populer pada oparasi pencarian (*Opration Routing*). Algoritma *Dijkstra* merupakan salah satu varian dari algoritma *greedy*, yaitu salah satu bentuk algoritma populer dalam pemecahan persoalan yang terkait dengan masalah optimasi.

2.8.1 Prinsip Kerja Algoritma Dijkstra

Algoritma *Djikstra* melakukan komputasi pencarian *rute* terpendek dari simpul sumber ke simpul tujuan berdasarkan bobot minimum pada setiap sisi yang menghubungkan simpul-simpul pada suatu graf.

Komputasi adalah cara atau proses untuk menemukan pemecahan masalah dari *n* data input dengan menggunakan suatu algoritma. Jika ada sisi yang bernilai negatif, algoritma *Dijkstra* tidak membenarkannya. Disinilah masalahnya, padahal dapat ditemukan jalan yang lebih dekat menuju simpul yang paling dekat untuk mencapai tujuannya. Sehingga biasa dikatakan algoritma *Dijkstra* tidak dapat mengatasi nilai negatif sebagai lintasan terpendek dalam suatu graf. Sehingga membuat lintasan yang diambil semakin banyak untuk mencapai simpul tujuan. Sesuai dengan prinsip *greedy* yang secara harafiah berarti tamak atau rakus namun tidak dalam konteks negatif, algoritma *greedy* ini hanya memikirkan solusi terbaik yang akan diambil pada setiap langkah tanpa memikirkan konsekuensi ke depan.

Intinya algoritma *Dijkstra* ini berupaya membuat pilihan nilai optimum lokal pada setiap langkah dan berharap agar nilai optimum lokal ini mengarah kepada nilai optimum global. Prinsipnya, ambillah apa yang bisa anda dapatkan saat ini (*take what you can get now*!), keputusan yang telah diambil pada setiap langkah tidak akan bisa diubah kembali. Algoritma *Dijkstra* memiliki kompleksitas waktu sebesar $T(n) = O(|E|+(|V|*\log |V|))$. Elemen-elemen penyusun algoritma *Dijkstra* yaitu:

1.Himpunan kandidat

Himpunan ini berisi elemen-elemen yang memiliki peluang untuk membentuk solusi. Pada persoalan lintasan terpendek dalam graf, himpunan kandidat ini adalah himpunan simpul pada graf tersebut.

2. Himpunan solusi

Himpunan ini berisi solusi dari permasalahan yang diselesaikan dan elemennya terdiri dari elemen dalam himpunan kandidat namun tidak semuanya atau dengan kata lain himpunan solusi ini adalah upabagian dari himpunan kandidat.

3. Fungsi seleksi

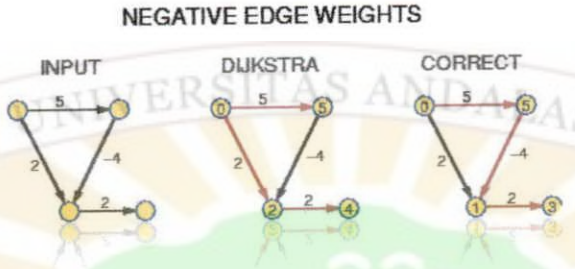
Fungsi seleksi adalah fungsi yang akan memilih setiap kandidat yang yang memungkinkan untuk menghasilkan solusi optimal pada setiap langkahnya.

4. Fungsi kelayaka

Fungsi kelayakan akan memeriksa apakah suatu kandidat yang telah terpilih (*terseleksi*) melanggar *constraint* atau tidak. Apabila kandidat melanggar *constraint* maka kandidat tidak akan dimasukkan ke dalam himpunan solusi.

5. Fungsi objektif

Fungsi objektif akan memaksimalkan atau meminimalkan nilai solusi. Tujuannya adalah memilih satu saja solusi terbaik dari masing-masing anggota himpunan solusi. Gambar berikut ini merupakan ilustrasi kelemahan algoritma *Dijkstra*,



Gambar 2.8.1.1 Ilustrasi kelemahan algoritma *Dijsktra*.

Jalur terpendek (*Shortest path)* dalam graf adalah jumlah total minimum bobot (*cost minimum*) dari sisi-sisi yang menghubungkan simpul sumber dengan simpul tujuan. Terdapat beberapa macam persoalan lintasan terpendek antara lain:

- Lintasan terpendek antara dua buah simpul tertentu (*a pair shortets path*).
- Lintasan terpendek antara semua pasangan simpul (*all pairs shortest path*).
- Lintasan terpendek dari simpul tertentu kesemua simpul yang lain (*single-source shoertest path*).
- Lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu (*intermediate shortest path*).

Pada tulisan ini akan dibahas masalah lintasan terpendek dari suatu simpul sumber ke simpul tujuan pada graf berarah, berbobot dan terhubung dengan menggunakan algoritma *Dijkstra* dan algoritma *Bellman-Ford.* Lintasan dari simpul asal ke simpul yang baru haruslah merupakan lintasan yang terpendek diantara semua lintasannya ke simpul-simpul yang belum terpilih. Pada setiap iterasi, himpunan

solusi ini akan berubah jika terdapat sisi dengan bobot lebih kecil dari sisi pada himpunan solusi. Secara umum langkah-langkah atau algoritma *Dijkstra* adalah sebagai berikut:

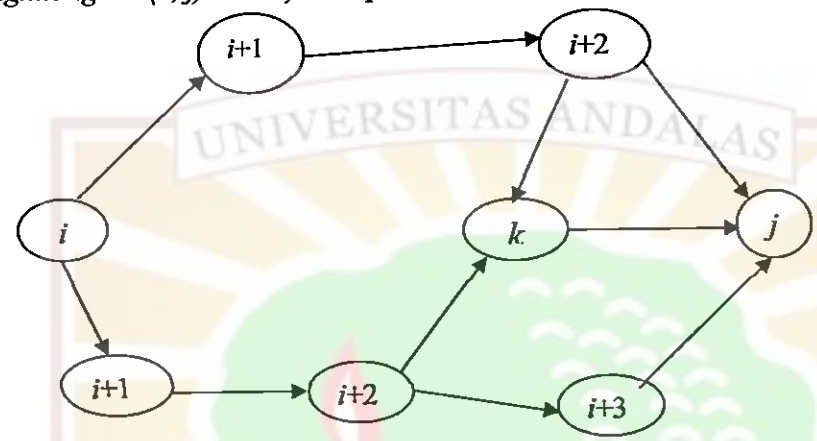
1. Pada langkah ke-k : cari sisi k yang dapat dijangkau dari simpul s dengan bobot sisi k minimum, dilambangkan dengan T_k.
2. Pada langkah $k+1$: cari simpul v dengan jarak minimum dari s dan dapat dijangkau dengan simpul-simpul yang ada pada T_k *(kecuali v sendiri)*.
3. $T_{k+1} = T_k \, U \, \{v\}$.
4. Algoritma berhenti ketika semua simpul sudah dikunjungi.

2.9 Algoritma Bellman-Ford

Algoritma *Bellman-Ford* merupakan algoritma untuk mencari lintasan terpendek (*shortest path*), dimana *Bellman-Ford* menghitung jarak terpendek dari satu sumber pada suatu graf berarah (*digraph*) berbobot. Maksudnya dari satu sumber ialah bahwa ia menghitung semua jarak terpendek yang berawal dari satu simpul (*node*). Algoritma *Bellman-Ford*, seperti halnya algoritma *Dijkstra*, digunakan untuk mencari lintasan terpendek pada sebuah graf berarah yang membedakan keduanya adalah pada algoritma *Bellman-Ford* bisa digunakan untuk graf yang memiliki sisi dengan bobot negatif, walaupun menggunakan waktu yang lebih lama. Algoritma ini memiliki kompleksitas sebesar $O(|V|.|E|)$ dimana V adalah jumlah simpul dan E adalah jumlah dan sisi. Sedangkan kompleksitas ruangnya sebesar $O(|V|)$.

Teorema Bellman [Ruspaniza, 2003]

Misalkan G adalah graf berarah dengan pembobotan. Jika P : i → j adalah suatu lintasan terpendek dari i ke j di dalam graf G dan (k, j) adalah sisi yang terakhir yang menghubungkan lintasan P maka, P_k : i→ k (diperoleh dengan menghilangkan (k, j) dari P) merupakan suatu lintasan terpendek dari i→k.



Gambar 2.9.1 Lintasan terpendek dari *i* ke *j* dengan *k* sebagai simpul antara.

Bukti:

Misalkan $P : i \rightarrow j$ adalah suatu lintasan terpendek dari *i* ke *j* di dalam graf *G* dan *(k, j)* adalah rusuk terakhir pada lintasan *P*. Akan dibuktikan bahwa $P_k : i \rightarrow k$ adalah suatu lintasan terpendek dari $i \rightarrow k$. Andaikan $P_k = i \rightarrow k$ bukan suatu lintasan terpendek dari $i \rightarrow k$, maka terdapat lintasan lain dari $i \rightarrow k$ sebutlah P_k^* : $i \rightarrow k$ yang lebih pendek dari $P_k : i \rightarrow k$, dengan demikian $P_k^* < Pk$. Dengan menambahkan rusuk *(k, j)* pada P_k^* sehingga diperoleh lintasan baru dari *i* ke *j*, sebutlah $P^* : i \rightarrow j = P_k^* : i \rightarrow k + (k, j)$. Karena $P_k^* < P_k$ maka akan didapat:

$$P_k^* : i \rightarrow k + (k, j) < P_k : i \rightarrow k + (k, j)$$

$$P^* : i \rightarrow j < P : i \rightarrow j$$

Ini berarti bahwa $P^* : i \rightarrow j$ lebih pendek dari $P : i \rightarrow j$. Hal ini bertentangan *(kontradiksi)* dengan pernyataan bahwa $P : i \rightarrow j$ adalah lintasan terpendek $i \rightarrow j$. Dengan demikian haruslah $P_k : i \rightarrow k$ adalah lintasan terpendek dari $i \rightarrow k$.

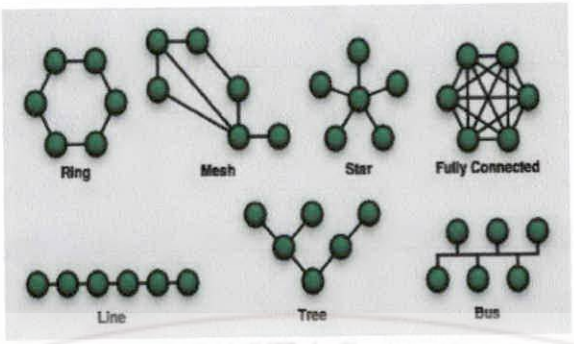
Langka-langkah atau algoritma *Bellman-Ford* adalah sebagai berikut:

1. Temukan jalur terpendek antara simpul *s* dan simpul lainnya, sehingga jalur ini adalah paling banyak memiliki 1 lompatan (*hop*).
2. Cari jalur terpendek antara *s* dan simpul lainnya dengan memiliki paling banyak dua lompatan.
3. Lakukan iterasi sampai jalur terpendek memiliki jumlah lompatan paling banyak berjumlah diameter dari graf. Diameter graf adalah jarak maksimum antara pasangan simpul pada graf, diukur dengan lompatan (*hop*).

2.10 Jaringan (*Network*)

Jaringan adalah suatu sistem yang menghubungkan komunikasi antara suatu fasilitas ke fasilitas yang lain. Secara pisual atau gambar, jaringan ini memiliki bentuk yang tersebar secara geografis yang berupa *Ring, Mesh, Start, Full connected, Line, Tree*, dan *Bus* artinya dalam suatu simpul pada suatu jaringan dapat menyebar dan mempunyai hubungan dalam suatu wilayah, negara bahkan antar negara dan dapat dilihat seperti gambar berikut.

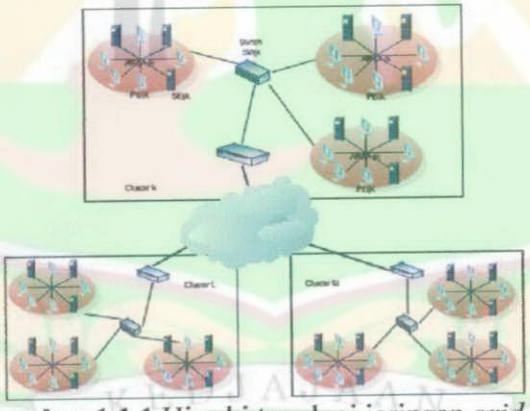
Gambar di bawah ini merupakan bentuk-bentuk topologi jaringan :



Gambar 2.10.1 Graf topologi jaringan.

2.10.1 Jaringan Grid (*Grid Network*)

Jaringan Grid adalah suatu kumpulan sumber (*resource*) yaitu mesin, CPU, memori yang berkomunikasi satu sama lain dengan menggunakan cara-cara (*protocol*) tertentu. Tampak seperti pada gambar berikut yang merupakan topologi jaringan *Grid*.



Gambar 1.1.1 Hirarki topologi jaringan *grid*.

2.11 Kompleksitas Waktu dan Ruang Algoritma

Kompleksitas waktu adalah suatu ukuran waktu yang diperlukan untuk mengeksekusi atau menjalankan suatu algoritma. Secara teori model abstrak

pengukuran waktu atau ruang harus *independen* dari pertimbangan mesin dan *compiler* apapun. Model yang seperti itu dapat digunakan untuk membandingakan algoritma yang berbeda. Besaran yang dipakai untuk menerangkan model abstrak pengukuran waktu atau ruang adalah kompleksitas algoritma. Ada dua macam kompleksitas algoritma, yaitu kompleksitas waktu dan kompleksitas ruang.

2.11.1 Kompleksitas Waktu

Kompleksitas waktu didefinisikan sebagai jumlah tahapan komputasi yang dibutuhkan untuk menjalankan algoritma sebagai fungsi dari ukuran masukan *n*.

2.11.2 Kompleksitas Ruang

Kompleksitas ruang didefinisikan sebagai jumlah memori yang digunakan oleh struktur yang terdapat dalam algoritma sebagai fungsi ukuran masukan *n*. Dengan menggunakan besaran kompleksitas waktu atau ruang algoritma, maka dapat ditentukan laju peningkatan waktu atau ruang yang diperlukan algoritma dengan meningkanya ukuran masukan *n*.

2.12 Terminologi Kompleksitas Waktu dan Ruang

Terminologi yang diperlukan dalam membahas kompleksitas waktu dan kompleksitas ruang suatu algoritma adalah:

1. Ukuran besara masukan data untuk suatu algoritma, *n*. Sebagai contoh, dalam algoritma pengurutan elemen-elemen larik, *n* adalah jumlah elemen larik, sedangkan dalam algoritma perkalian matriks *n* adalah ukuran matriks *n x n*. Pada beberapa kasus, ukuran masukan lebih tepat

menggunakan dua buah besaran, misalnya jika masukan algoritma adalah graf, maka ukuran masukan adalah jumlah simpul atau *vertex* dan jumlah sisi atau *edge*.

2. Kompleksitas waktu, $T(n)$, adalah jumlah operasi yang dilakukan untuk mengeksekusi atau menjalankan algoritma sebagai fungsi dari ukuran masukan dari n.
3. Kompleksitas ruang, $S(n)$, adalah ruang memori yang dibutuhkan algoritma sebagai fungsi dari ukuran masukan n.

BAB III

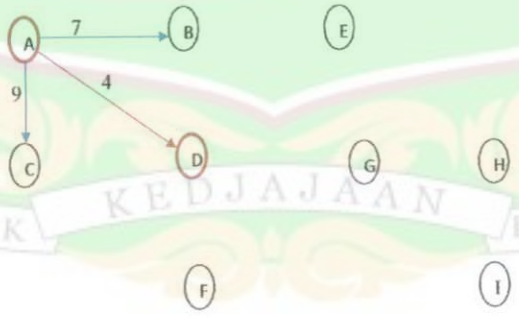
PEMBAHASAN

3.1 Contoh Studi Kasus Algoritma Dijkstra

Misalkan diberikan suatu graf berbobot, berarah dan terhubung yang merepresentasikan kondisi keterhubungan pada jaringan *Grid*, dengan ilustrasi dimana simpul-simpul diinisialisasikan sebagai lokasi pada setiap *close computing* yang terhubung dari perpaduan antara pintu masuk (*gate*) dan tombol (*switch*) yang diinisialisasikan sebagai sisi. Seperti pada gambar berikut:

Gambar 3.1.1 langkah pertama.

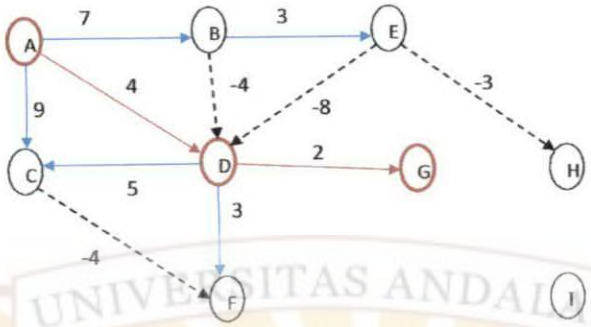
Gambar 3.1.1 Tentukan lokasi awal sebagai lokasi sumber yang akan dijadikan sebagai *routing* ke lokasi tujuan dan itu adalah lokasi *A*.



Gambar 3.1.2 Langkah kedua.

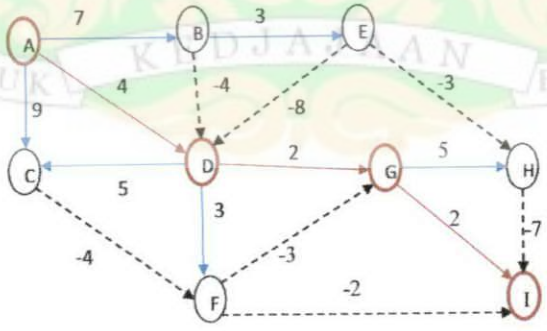
Gambar 3.1.2 gambar ini menjelaskan bahwa dari tiga alternatif sisi yang diprioritaskan pada lokasi *A* untuk menuju lokasi selanjutnya adalah sisi yang

berwarna merah yang menginisialkan bahwa bobot yang paling minimum dari tiga kemungkinan pada tiga sisi tersebut, yang akan di*routing*kan oleh simpul *D*.



Gambar 3.1.3 langkah ketiga.

Gambar 3.1.3 gambar ini menjelaskan bahwa dari tiga lokasi yaitu lokasi *C*, lokasi *D* dan lokasi *E* yang masing-masing lokasi memiliki peluang me*routing*kan suatu paket data, namun demikian untuk lokasi *C* dan lokasi *E* masing-masing sisi yang memiliki kemungkinan untuk meneruskan paket data berbobot minus sesuai dengan algoritma *Dijkstra* hal tersebut tidak terdefinisi, maka kedua lokasi tersebut tidak terpilih untuk menjadi *routing* selanjutnya, sehingga lokasi *D* yang akan menjadi *routing* suatu paket data kelokasi tujuan, dari tiga kemungkinan sisi pada lokasi *D* sesuai dengan yang telah diinisialkan pada gambar 3.1.2 terpilih sisi yang bebobot (2) yang dalam artian lokasi *G* yang terpilih sebagai *routing* selanjutnya.



Gambar 3.1.4 langkah keempat.

Gambar 3.1.4 gambar ini menjelaskan bahwa dengan terpilihnya lokasi *G* pada gambar langkah 3.1.3 dengan tidak menutup peluang kemungkinan untuk *routing* pada lokasi *F* dan lokasi *H* yang menjadi prioritas untuk menyampaikan suatu paket data ke lokasi tujuan , karena lokasi *F* dan lokasi *H* memiliki kemungkinan masing-masing sisi pada kedua lokasi tersebut berbobot minus maka lokasi *G* yang menjadi *routing* suatu paket data ke lokasi tujuan yaitu lokasi I dengan dua prioritas sisi yaitu berbobot (2) dan berbobot (5) dari kedua sisi tersebut yang terpilih untuk langsung me*routing*kan ke lokasi tujuan yaitu lokasi *I* adalah sisi berbobot (2). Sesuai dengan algoritma *Dijkstra* karena semua kemungkinan atau alternaif sisi pada graf tersebut telah di perhitungkan maka eksepansi ini berhenti. Jadi, dengan demikian berdasarkan rumus hitung untuk kompleksitas waktu algoritma *Dijkstra* maka dapat dihitung $T(n) = O(E+(|V|*\log |V|)) = O(15+ (9*\log 9)) = (15+9(0,99)) = (15 + (8.91)) = 23,91$ satuan waktu. Dan untuk kompleksitas ruangnya sebesar $S(n) = O(V) = 9$.

3.1.1 Analisis Algoritma Dijkstra

1. Algoritma *Dijkstra* memandang bahwa untuk mencapai optimum global, dapat dilakukan dengan pendekatan yakni mengambil suatu nilai pada saat iterasi pertama adalah merupakan nilai optimum lokal dengan harapan resultan akhirnya akan menghasilkan nilai yang optimum juga.

Nilai optimum lokal adalah nilai-nilai keputusan sementara dari iterasi pada algoritma *Dijkstra*, sedangakan nilai optimum global adalah nilai total bobot minimum dari keseluruhan iterasi yang dilakukan oleh algoritma *Dijkstra*.

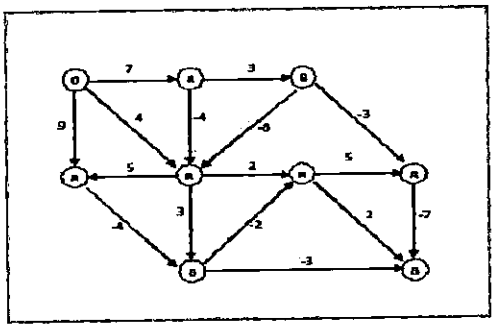
2. Algoritma *Dijkstra* hanya berbasiskan pada pengambilan keputusan setiap saat tanpa mempertimbangkan dampak ke depannya, maka algoritma *Dijkstra* tidak selalu menjamin bahwa nilai yang dihasilkan adalah nilai optimum yang sesungguhnya, tetapi dapat dikatakan bahwa nilai yang dihasilkan merupakan suatu hampiran dari nilai optimum global.

3. Algoritma *Dijkstra* hanya mempertimbangkan suatu solusi secara sesaat sehingga kebutuhan ruang dan waktunya sangat sedikit, untuk algoritma *Dijkstra, kompleksitas* waktu untuk suatu graf dengan sisi E dan simpul V dapat ditulis sebagai suatu fungsi dari $|E|$ dan $|V|$ sehingga $O(|E| + |V|*\log|V|) = O(|E| + \log|V|)$ dimana O adalah lambang dari *kompleksitas.*

4. Algoritma *Dijkstra* tidak akan mendefinisikan bobot dari sisi yang bernilai negatif pada setiap iterasi pencarian lintasan terpendek dari simpul awal kesimpul tujuan (*nilai bobot harus >=0*).

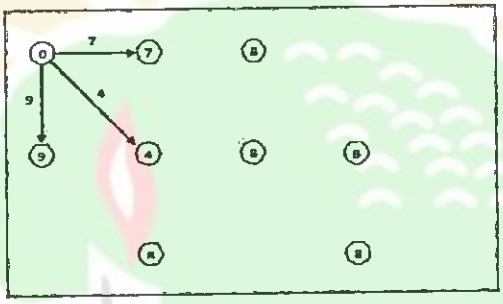
3.2 Contoh Studi Kasus Algoritma Bellman-Ford

Diberikan suatu graf berbobot, berarah dan terhubung yang merepresentasikan keterhubungan pada jaringan *grid*, dengan ilustrasi dimana simpul-simpul diinisialisasikan sebagai lokasi (*node*) pada setiap *close computing* yang terhubung dari perpaduan antara pintu masuk (*gate*) dengan tombol (*switch*) yang dinyatakan sebagai sisi. Seperti pada gambar berikut:



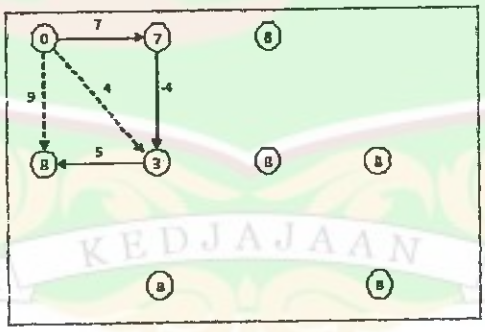
Gambar 3.2.1 langkah pertama algoritma *Bellman-Ford.*

Gambar 3.2.1 diberikan graf berbobot, berarah dan terhubung seperti gambar di atas.



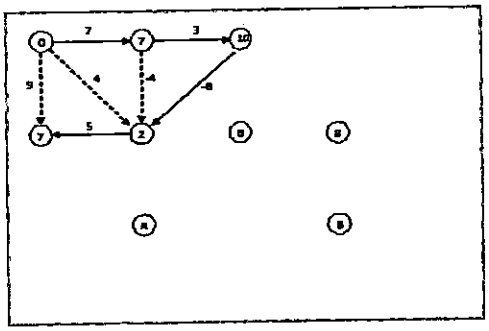
Gambar 3.2.2 langkah kedua algoritma *Bellman-Ford.*

Gambar 3.2.2 Pada gambar ini menyatakan bahwa ada tiga kemungkinan sisi yang akan dilewati oleh simpul yang telah didedekasikan sebagai simpul awal.



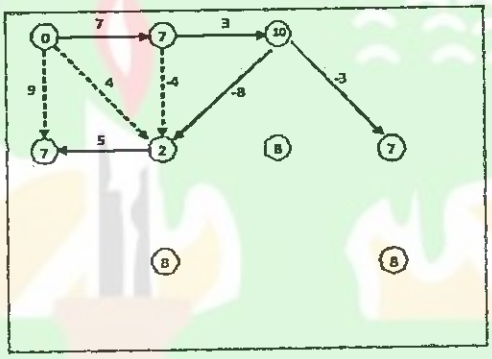
Gambar 3.2.3 langkah ketiga algoritma *Bellman-Ford.*

Gambar 3.2.3 gambar ini menyatakan bahwa sisi yang telah diperhitungkan yang paling minimum dari kemungkinan sisi yang menghubungkan untuk langkah selanjutnya adalah sisi yang berbobot (7).



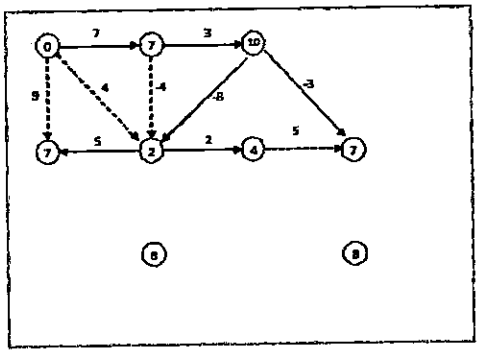
Gambar 3.2.4 langkah keempat algoritma *Bellman-Ford.*

Gambar 3.2.4 gambar ini menjelaskan bahwa dari lokasi yang berbobot (7) memiliki dua sisi prioritas untuk melanjutkan kelangkah simpul selanjutnya yaitu sisi yang bebobot (3) yang menjadi sisi penghubung ke simpul selanjutnya yang akan menjadi simpul *routing*.



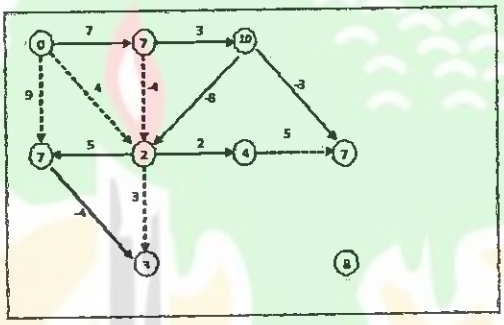
Gambar 3.2.5 langkah kelima algoritma *Bellman-Ford.*

Gambar 3.2.5 gambar ini menjelaskan bahwa dari rute yang telah ditetapkan oleh gambar 3.2.4 maka dari lokasi yang berbobot (10) memiliki dua prioritas lokasi yang akan dilanjutkan untuk lokasi selanjutnya yaitu sisi berbobot (-8) dan sisi berbobot (-3).



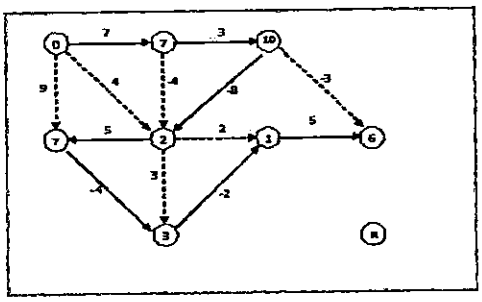
Gambar 3.2.6 langkah keenam algoritma *Bellman-Ford.*

Gambar 3.2.6 gambar ini menjelaskan bahwa sisi yang berbobot (-8) telah terseleksi dan ditetapkan sebagai penghubung lokasi yang akan menjadi *routing* selanjutnya.



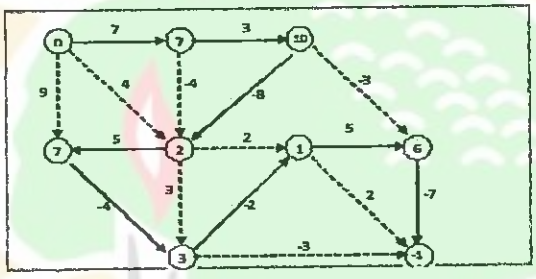
Gambar 3.2.7 langkah ketujuh algoritma *Bellman-Ford.*

Gambar 3.2.7 gambar ini menunjukkan bahwa prioritas lokasi yang berbobot (2) akan melanjutkan dari gambar 3.2.6 adalah tiga sisi yang akan diseleksi yaitu sisi berbobot (5), sisi berbobot (2) dan sisi berbobot (3) dari prioritas seleksi sisi tersebut yang paling minimum yang terseleksi untuk *routing* simpul selanjutnya adalah sisi yang berobot (5).



Gambar 3.2.8 langkah kedelapan algoritma *Bellman-Ford.*

Gambar 3.2.8 gambar ini menjelaskan bahwa dari gambar 3.2.7 yang telah ditetapkan sebagai *routing* yang akan melewati sisi yang berbobot (-4) yang menghasilkan bobot lokasi (3).



Gambar 3.2.9 langkah kesembilan algoritma *Bellman-Ford.*

Gambar 3.2.9 gambar ini menjelaskan dari lokasi yang berbobot (3) prioritas bobot sisi yang akan dilanjutkan dari gambar 3.2.8 adalah sisi berbobot (-2) dan sisi berbobot (-3) dari prioritas kedua sisi tersebut yang memiliki sisi bobot minimum untuk lokasi *routing* selanjutnya adalah sisi yang berbobot (-2) dan dilanjutkan oleh dua prioritas yaitu sisi berbobot (5) dan berbobot (2) yang terseleksi untuk *routing* selanjutnya adalah sisi berbobot (5) yang akan disampaikan ke lokasi tujuan oleh sisi yang berbobot (-7). Sesuai dengan langkah-langkah atau algoritma *Bellman-Ford* tersebut maka eksepansi berhenti. Sesuai dengan rumus kompleksitas waktu yang dimiliki algoritma *Bellman-Ford* yaitu $O(|V|.|E|)$, maka dapat dihitung yaitu:

Jumlah simpul = 9

Jumlah sisi = 15

Maka kompleksitas waktu algoritma *Bellman-Ford* dapat dihitung dimana $T(n) = O(|V|.|E|) = O(|9|.|15|) = 135$ satuan waktu. Sedangkan kompleksitas ruangnya adalah $O(|V|) = O(|9|) = 9$.

3.2.1 Analisis Algoritma Bellman-Ford

1. Algoritma *Bellman-Ford* dapat menjamin ditemukannya solusi optimum pada pohon ruang status.
2. Untuk algoritma *Bellman-Ford* pembuktian solusi selalu merupakan solusi optimum global dapat dilakukan dengan menggunakan induksi.
3. Algoritma *Bellman-Ford* pada umumnya memerlukan waktu yang lebih lama dari pada algoritma *Dijkstra*, karena algoritna *Bellman-Ford* pada setiap iterasi memerlukan informasi bobot minimum (*cost minimum*) sekalipun bobotnya *negative* untuk melanjutkan kelangkah selanjutnya.
4. Algoritma *Bellman-Ford* memiliki *kompleksitas* waktu untuk suatu graf dengan sisi E dan simpul V dapat ditulis dalam bentuk suatu fungsi dari $|E|$ dan $|V|$ sebagai $O(|V|.|E|)$ sementara *kompleksitas* ruangnya adalah $O(|V|)$.

BAB IV

PENUTUP

4.1 Kesimpulan

Dari pembahasan pada bab sebelumnya maka dapat disimpulkan bahwa algoritma *Disjkstra* dan algoritma *Bellman-Ford* ini mempunyai kelebihan dan kekurangan pada saat menjalankan atau mengeksekusi algoritmanya dimana untuk algoritma *Bellman-Ford* dapat menyelesaikan masalah lintasan terpendek dengan kasus graf berbobot negatif yang tidak dapat diselesaikan oleh algoritma *Dijkstra*, karena hal ini merupakan hal yang paling prinsip diantara kedua algoritma, namun demikian algoritma *Dijkstra* lebih cepat pada saat menjalankan atau mengeksekusi algoritmanya dari pada algoritma *Bellman-Ford* dengan kasus bobot sisi pada graf tidak ada sisi negatif (sisi ≥ 0). Karena algoritma *Dijkstra* menggunakan prinsip *Greedy*. Jadi, Algoritma *Djikstra* lebih cocok jika digunakan dalam suatu jaringan karena algoritma tersebut dapat mengetahui konfigurasi keseluruhan jaringan dengan kebutuhan waktu (*running time*) yang lebih kecil. Sehingga berdasarkan masalah yang dapat diselesaikan, maka kedua algoritma ini mempunyai kelebihan dan kekurangan yaitu:

- Algoritma *Dijkstra* untuk masalah *Single-source Shortest Path.*
- Algoritma *Bellman-Ford* untuk masalah *Pairs Shortest Path.*

Berdasarkan urutan penyelesaian persoalan lintasan terpendek untuk masing-masing algoritma tersebut kompleksitasnya adalah $O(|V|.|E|) > O(|E|+|V|)\log V)$ = *Bellman-Ford>Dijkstra.*

4.2 Saran

Kepada peneliti selanjutnya, yang akan membahas tentang perbandingan algoritma *Dijkstra* dan algoritma *Bellman-Ford* disarankan agar memilih persoalan lintasan terpendek yang langsung secara praktek dapat dilihat pada kenyataan dalam kehidupan sehari-hari, seperti seorang salesman, pengiriman surat oleh pegawai kantor pos dalam suatu wilayah dan lain-lain untuk mendapatkan spesifikasi, kontribusi dan perbedaan dari kedua algoritma tersebut.

DAFTAR PUSTAKA

Darman Irfan, dkk, 2010. *Algoritma Routing di Lingkungan Jaringan Grid Menggunakan Teori Graf.* Teknik Elektro Universitas Siliwangi: Bandung.

http://id.wikipedia.org/wiki/Algoritma_Bellman-Ford. tanggal akses 19 Mei jam11.20 wib.

http://amicta.web.id/pic/floyd-warshall25.jpg. tanggal akses 08-Desember-2010 19:13 wib.

http://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm. tanggal akses 08-Desember 2010 19:13).

http://en.wikipedia.org/wiki/Dijkstra's_algorithm. tanggal akses 08-Desember-2010 19:13).

http://en.wikipedia.org/wiki/Routing_Information_Protocol tanggal akses 04-Desember-2010 jam17:31 wib.

http://student.eepis-its.edu/~izankboy/laporan/Jaringan/ccna2-6.pdf tanggal akses 04-Desember-2010 17:31 wib.

http://www8.cs.umu.se/~jopsi/dinf504/bellman_ford.gif. tanggal akses 08-Desember-2010 19:13).

http://www.egr.unlv.edu/~jjtse/CS477/Dijkstra%20SP.jpg. tanggal akses 08-Desember-2010 19:13).

Liu, C.L, 1985. *Element of Discrete Mathematics*, McGraw-Hill, Inc, International.

Munir, Rinaldi, 2005. *Matematika Diskrit*. Edisi Ketiga, Penerbit Infomatika Bandung: Bandung.

Ruspaniza-98134027. 2003, *Penggunaan algoritma Floyd Untuk Menentukan Lintasan Terpendek. Skripsi Sarjana Matematika.* Universitas Andalas: Padang.

DAFTAR RIWAYAT HIDUP

Penulis dilahirkan pada tanggal 21 juli 1988 di Desa Pancur Negara, Kab. Kaur, dari Ayah yang bernama Ripudin dan Ibu Liasma. Penulis tamat di SDN 21 Rigangan pada tahun 2000, tamat di SMPN 2 Tanjung Ganti pada tahun 2003, dan tamat di SMA N 2 Kaur Utara yang kini telah diganti menjadi SMA N 4 Kaur pada tahun 2007, setelah tamat SMA penulis mendapatkan kesempatan untuk mengikuti tes Program S1 Basic Science ikatan Dinas Guru Berasrama di Kab. Kaur yang alhamdulilah lulus. Tahun ajaran 2007/2008 itu juga penulis melanjutkan Pendidikan S1 Basic Science Guru Berasrama di Fakultas Matematika Dan Ilmu Pengetahuan Alam Universitas Andalas dan, kemudian penulis menamatkan pendidikan Program S1 Guru Berasrama (Basic Scence) di Universitas Andalas pada Tahun 2011, sekaligus mendapat gelar Sarjana Sains.