Representasi Graf Dalam Pencarian Rute Terpendek Dengan Menggunakan Algoritma A*

Yudha Setia Racana Ganesha Putra
Jurusan Teknik Informatika ITB, Bandung 40132, email: if14118@students.if.itb.ac.id

Teori graf merupakan topik yang banyak mendapat perhatian, karena model-modelnya sangat berguna untuk aplikasi yang luas, seperti masalah dalam jaringan komunikasi, transportasi, ilmu komputer, dan lain sebagainya. Banyak sekali struktur yang bisa direpresentasikan dengan graf, dan banyak masalah yang bisa diselesaikan dengan bantuan graf. Salah satunya adalah merepresentasikan pencarian rute terpendek, dimana tempat direpresentasikan oleh node, dan penghubung antar tempat itu direpresentasukan oleh edge. Dalam masalah pencarian rute terpendek, terdapat algoritma yang bernama heuristic depth first, dimana algoritma ini menggunakan dua nilai untuk mencari rute terpendek, yaitu nilai yang terdapat pada edge, dan nilai heuristic yang terdapat pada node.

***Kata Kunci**: Graf, Rute Terpendek, A*, node, edge, Graph Search.*

1. PENDAHULUAN

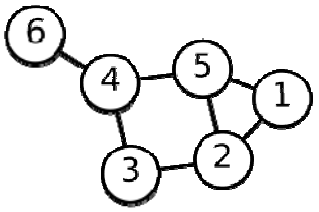
Teori graf merupakan topik yang banyak mendapat perhatian, karena model-modelnya sangat berguna untuk aplikasi yang luas, seperti masalah dalam jaringan komunikasi, transportasi, ilmu komputer, dan lain sebagainya. Banyak sekali struktur yang bisa direpresentasikan dengan graf, dan banyak masalah yang bisa diselesaikan dengan bantuan graf. Salah satunya adalah merepresentasikan pencarian rute terpendek, dimana *node* merepresentasikan tempat dan *edge* merepresentasikan jarak.

Permasalahan rute pencarian ini dapat diselesaikan dengan banyak algoritma seperti *Breadth-First, Depth-First*, A*, *Lowest Cost*, dan *Heuristic Depth-First*. Algoritma seperti *Breadth-First* dan *Depth-First* tidak membutuhkan nilai apapun untuk mencari nilai terpendek. *Sedangkan Lowest Cost* dan *Heuristic Depth-First* hanya memerlukan satu buah nilai ntuk melakukan pencarian, yaitu nilai jarak yang terdapat pada *edge*-nya atau nilai *heuristic* yang terdapat pada *node*-nya. Sedangkan algoritma A* memiliki kedua nilai tersebut.

Algoritma A* adalah pengembangan dari algoritma *Heuristic Depth-First* dan algoritma *Lowest Cost*, dimana algoritma *Heuristic Depth-First* mencari rute terpendek dengan menggunakan nilai *heuristic* dan algoritma *Lowest Cost* adalah algoritma yang mencari rute dengan menggunakan nilai yang terdapat pada nilai pada *edge*-nya. Dengan menggnakan kedua nilai tersebut, algoritma A* mencari rute terpendek dengan menjumlahkan jarak yang telah ditempuh dan sisa jarak yang akan ditempuh.

2. TEORI GRAF

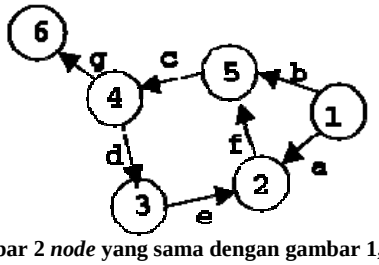
Di matematika dan ilmu komputer, teori graf adalah cabang ilmu yang mempelajari sifat-sifat graf. Secara informal, suatu graf adalah himpunan benda-benda yang disebut *node* yang terhubung oleh *edge-edge*. Biasanya, graf digambarkan sebagai kumpulan titik-titik (melambangkan *node*) yang dihubungkan oleh garis-garis (melambangkan *edge*).



Gambar 1 Graf dengan 6 *node* dan 7 *edge*

Sebuah struktur graf bisa dikembangkan dengan memberi bobot pada tiap *edge*. Graf berbobot dapat digunakan untuk melambangkan banyak konsep berbeda. Sebagai contoh jika suatu graf melambangkan jaringan jalan maka bobotnya bisa berarti panjang jalan maupun batas kecepatan tertinggi pada jalan tertentu. Ekstensi lain pada graf adalah dengan membuat *edge*nya berarah, yang secara teknis disebut graf berarah atau digraf (*directed graf*).

Suatu graf G dapat dinyatakan sebagai $G = <V, E>$. Graf G terdiri atas himpunan V yang berisikan *node*/*node* pada graf tersebut dan himpunan dari E yang berisi *edge* pada graf tersebut. Himpunan E dinyatakan sebagai pasangan dari *node* yang ada dalam V. Sebagai contoh definisi dari graf pada gambar diatas adalah : $V = \{1,2,3,4,5,6\}$ dan $E = \{(1,2),(1,5),(2,3),(3,4),(4,5),(5,2),(4,6)\}$.



Gambar 2 *node* yang sama dengan gambar 1, tapi merupakan digraf

Pada digraf maka pasangan-pasangan ini merupakan pasangan terurut. Untuk menyatakan digraf (gambar kedua yang menggunakan tanda panah) kita dapat menggunakan himpunan *edge* sebagai berikut :

E = { < 1,2 > , < 1,5 > , < 2,5 > , < 3,2 > , < 4,3 > , < 5,4 > , < 4,6 > }

Dalam himpunan *edge* untuk digraf, urutan pasangan *node* menentukan arah dari *edge* tersebut.

Dalam teori graf, formalisasi ini untuk memudahkan ketika nanti harus membahas terminologi selanjutnya yang berhubungan dengan graf. Beberapa terminologi berhubungan dengan teori graf :

- *Degree* atau derajat dari suatu *node*, jumlah *edge* yang dimulai atau berakhir pada *node* tersebut. *Node* 5 berderajat 3. *Node* 1 berderajat 2.
- *Path* suatu jalur yang ada pada graf, misalnya antara 1 dan 6 ada path $b \longrightarrow c \longrightarrow g$
- *Cycle* siklus path yang kembali melalui titik asal 2 $f \longrightarrow c \longrightarrow d \longrightarrow e$ kembali ke 2.
- *Tree* merupakan salah satu jenis graf yang tidak mengandung cycle. Jika *edge* f dan a dalam digraf diatas dihilangkan, digraf tersebut menjadi sebuah tree. Jumlah *edge* dalam suatu tree adalah nV - 1. Dimana nV adalah jumlah *node.*

3. RUTE TERPENDEK

Dalam Teori Graf, masalah pencarian rute terpendek adalah menemukan jalan dengan biaya terkecil antara dua *node* yang dihubungkan oleh beberapa *edge.* Misalnya bagaimana cara menemukan cara tercepat untuk mencari satu lokasi ke lokasi yang lainnya ke dalam sebuah peta. *Node*-nya melambangkan sebuah kota, dan *edge-nya* melambangkan jalur-jalur penerbangan yang ada sebagai penghubung diantara kedua kota tersebut.

Pencarian rute terpendek dapat menggunakan berbagai macam algoritma seperti *Breadth-First, Depth-First,* A*, *Lowest Cost,* dan *Heuristic Depth-First.* Dimana algoritma *Breadth-First* dan *algoritma Depth-First* hanya mencari *goal node* dengan cara acak dan tidak membutuhkan nilai apapun, algoritma *Heuristic Depth-First* dan *Lowest Cost* membutuhkan satu nilai untuk menemukan *goal node,* sedangkan algoritma A* membutuhkan dua nilai untuk mencari *goal node* dengan rute terpendek.

4. ALGORITMA A*

Algoritma A* adalah suatu algoritma pencarian yang memanfaatkan nilai *heuristic* yang ada di setiap *node* dan biaya dari tiap *edge* yang terdapat pada bagian graf yang menjadi solusi persoalan. Nilai diperoleh dengan melakukan penjumlahan terhadap nilai *heuristic* dan *cost* dari *edge* yang menjadi solusi.

Algoritma ini merupakan pengembangan dari algoritma *Heuristic Depth-First* dan algoritma *Lowest Cost* dimana kedua algoritma tersebut hanya menggunakan satu nilai, yaitu nilai yang tertera pada *edge* pada algoritma *Lowest Cost*, dan nilai pada *node* pada algoritma *Heuristic Depth-First*.

Anggap fungsi h(N) mewakili nilai fungsi *heuristic* dari tiap *node* yang dilewati dari *node* awal sampai *node* tujuan, maka fungsi nilai f(N) sebagai nilai total pada algoritma *Heuristic Depth-First* adalah

$$f(N) = h(N) \quad (1)$$

Jika c(N) adalah *cost* seluruh jalur yang dilewati, maka fungsi nilai f(N) pada pada algoritma *Lowest Cost* adalah

$$f(N) = c(N) \quad (2)$$

Maka, dari persamaan (1) dan (2), nilai total f(N) pada algoritma A* yang memakai kedua nilai tersebut adalah

$$f(N)=h(N)+c(N) \quad (3)$$

Dimana jika f(N) semakin kecil, maka semakin besar prioritas untuk menempuh rute tersebut.

Hasil upagraf yang menjadi solusi persoalan adalah upa graf yang menghubungkan *node* awal sampai kepada *node* tujuan yang memiliki nilai f(N) terkecil.

Tiap-tiap *node* merepresentasikan kota yang dihubungkan oleh rute penerbangan. Nilai *cost* dari tiap edge diumpamakan sebagai ukuran hargatiket pesawat yang dibutuhkan untuk mencapai kota yang dihubungkan oleh *edge* tersebut. Nilai *heuristic* diperoleh dari perkiran lama waktu yang dibutuhkan untuk mencapai *goal node*.

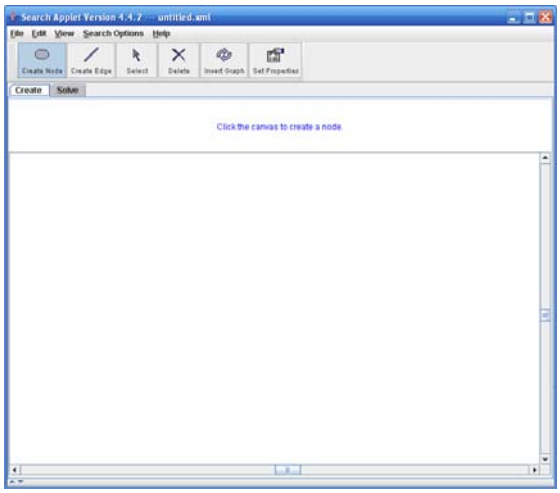
Pada persoalan *graf search*, setiap simpul yang berada diantara *goal node* dan *start node* digambarkan sebagai sebuah *node* pada graf yang berfungsi sebagai *agent.* Setiap simpul yang menghubungkan 2 *node* pada graf menggambarkan *cost* yang harus dilakukan oleh agent untuk menempuh simpul tersebut. Sedangkan pada *node*-nya terdapat nilai *heuristic*, yaitu perkiraan *cost* dari *current node* ke *goal node.*

Hal -hal yang harus diperhatikan dalam pencarian rute terpendek dengan menggunakan algoritma A* adalah :

1. Berapa *cost* yang sudah ditempuh dari *start node* ke *current node*, dapat dikatakan berapakah penjumlahan angka-angka yang sudah dilalui pada *start node* hingga ke *current node* tersebut.
2. Berapa perkiraan cost yang akan ditempuh oleh *node* tersebut untuk menuju ke *goal node*. Pada kasus ini karena dari setiap *node* memiliki banyak jalur untuk menuju *goal node*, maka kami mengambil perkiraan jalur terpendek yang akan ditempuh oleh titik/*node* tersebut untuk ke *goal node*. pada kasus ini *node* tersebut juga harus mengetahui ke *node* mana saja yang akan dilalui untuk memenuhi syarat jalur terpendek. Jalur terpendek disini berarti jalur yang memiliki nilai solusi atau *cost* terkecil.
3. Berapa jumlah dari cost yang sudah ditempuh dari *start node* dengan *cost* perkiraan yang akan ditempuh pada *node* menuju *goal node*.

5. EKSPLORASI APLIKASI

Terdapat banyak aplikasi dalam merepresentasikan graf dalam menyelesaikan masalah pencarian rute terpendek. Salah satu diantaranya adalah *Graph Search.* Di dalam aplikasi *Graph Search*, kita dapat membuat graf dengan membuat *node*-nya terlebih dahulu dan menyambungkan *node-node* tersebut dengan *edge-nya.*



Gambar 3 Aplikasi *Graph Search*

Fitur-fitur yang ada di dalam aplikasi *Graf Search* antara lain :

1. menu File

   Di dalam menu file terdapat beberapa fungsi yang dapat digunakan
   - Create New Graf

     Menghapus graf yang sedang ditampilkan dan menciptakan graf baru yang kosong
   - Load Sampel Graf

     Memunculkan sebuah graf yang berasal dari graf contoh yang merupakan bawaan aplikasi
   - Load From File

     Me-load problem dari local disk
   - Load From URL

     Me-load problem dari suatu lokasi URL
   - Save Graf

     Men-save graf kelocal hardisk
   - Print

     Mencetak tampilan yang terdapat pada solve canvas
   - Quit

     Keluar dari aplikasi

2. menu Edit

   Di dalam menu Edit terdapat beberapa fungsi yang dapat digunakan.
   - Undo

     Membatalkan operasi terakhir dan kembali ke keadaan sebelum operasi terakhir di lakukan.
   - Redo

     Merupakan kebalikan dari operasi Undo yaitu mengembalikan keadaan ke dalam keadaan terakhir sebelum operasi undo terakhir dilakukan.
   - View Prolog Code

     Menampilkan suatu frame yang berisi kode prolog yang merepresentasikan keaadaan graf. Kode yang ditampilkan tidak dapat diedit tetapi dapat dikopi ke dalam text editor.
   - View/Edit Text Reresentation

     Menampilkan frame yang berisi text yang merepresentasikan graf. Text yang ditampilkan dapat di edit. Graf dapat di edit dengan melakukan perubahan terhadap text dan menekan tombol “Update”. Dengen menggunakan fasilitas ini graf secara tidak langsung dapat diangkitkandari text file dengan mem-paste-kan text yang merepresentasikan graf dari file text tersebut ke dalam text yang terdapat di dalam frame.
   - View/Edit XML Representation

     Menampilkan frame yang berisi kode XML yang merepresentasikan graf. Code XML yang ditampilkan dapat di edit. Graf dapat di edit dengan melakukan update terhadap kode XML yang trdapat di dalam frame dan kemudian menekan “Update”. Dengan menggunkan fasilitas ini, graf dapat diupdate dengan mem-paste-kan kode XML yang merepresentasikan graf dari

file XML ke dalam text field yang terdapat di dalam frame.

3. menu View

   Di dalam menu View ada beberapa fungsi yang dapat dgunakan.

   - Font Size

     Melakukan perubahan terhadapukuran huruf yang digunakan di dalam graf.
   - Line Width

     Melakukan perubahan terhadap ukuran lebar dari garis yang digunakan dalam graf.
   - Autoscale

     Secara otomatis menyesuaikan ukuran graf ke dalam ukuran canvas yang digunakan untuk menampilkan graf.
   - Pan/Zoom

     Merubah mode untuk mengubah scala tampilan graf di canvas. Ketika memilih mode “Pan”, click tombol mouse sebelah kanan dan tahan lalu gerakkan di canvas untuk menggerakkan graf. Ketika menggunakan mode “Zoom”, click tombol mouse sebelah kanan dan tahan kemudian gerakkan di kanvas untuk memperbesar dan memperkecil ukuran graf di canvas.
   - Reset Labels

     Mereset label *edge* pada *graf* yang sebelumnya digeser dari kedudukan semula.
   - Enable Anti-Aliasing

     Menghidupkan atau mematikan efek anti aliasing .
   - Show Message Panel

     Menyembunyikan atau menampilkan message panel pada canvas.
   - Show Button Text

     Menampilkan atau menyembunyikan text yang terdapat pada tombol.
   - Show Button

     Menampilkan atau menyembunyikan toolbar.
   - Show Current Path

     Menampilkan atau menyembunyikan text area yang terdapat di bagian bawah aplikasi pada solve window yang menapilkan jalur dari *node* ketika melakukan pencarian.
   - Show *Node* Heuristic

     Menampilkan atau menyembunyikan nilai heuristic dari tiap *node*.
   - Show Edge Cost

     Menampilkan atau menyembunyikan harga tiap edge.
   - Show Quiz Result

     Menampilkan atau menyembunyikan frontier information dalam text selama quiz.

4. menu Search Option

   - Search Algorithm

     Memilih salah satu algoritma pencarian. Ada beberap algoritma pencarian yang dapat digunakan yaitu deep first, breadth first, lowest cost first, best first, heuristic best first, A*, dan user defined.
   - Pruning

     Memilih salah satu pruning option untuk melakukan pencarian. Pruning option yang tersedia yaitu multi-path pruning, loop detection, dan none(tidak menggunkan pruning option).
   - Set Cost and Heuristic

     Menapilkan suatu dialog box yang memeberikan pilihan untuk menganti semua nilai heuristic *node* atau semua harga edge dengan suatu nilai tertentu. Nilai baru di assign dengan mengalikan atau menjumlahkan nilai heuristic atau harga edge dengan nilai yang dimasukkan oleh user.
   - Set Node Heuristic Automatically

     Memberikan nilai terhadap *node* heuristic secara otomatis dengan mengassign nilai heuristic tiap *node* dengan jaraktiap *node* terhadap goal *node*. Jika tidak terdapat goal *node* maka nilai heuristic yang di assign adalah 0.
   - Set Edge Cost Automatically

     Mengmasukkan nilai cost untuk tiap edge dengan jarak 2 *node* yang dihubungkan oleh tiap masing-masing edge.
   - Animation Speed

     Mengatur speed untuk auto search pada solve mode. Speed ini menentukan waktu untuk menampilakan waktu untuk tiap langkah. Speed yang dapat digunakan yaitu very fast (0 detik), fast (0,1 detik), medium(0,5 detik), dan slow(1 detik).
   - Auto Search Option

     Menampilkan dialog box yang menanyakan jumlah maksimum step untuk tiap algoritma dan apakah searching dihentikan ketika goal *node* ditemukan.

5. menu Help

   Pada menu help terdapat beberapa fungsi yang dapat digunakan

   - Legend of *Node*s/Edges

Menampilkan suatu dialog box yang menjelaskan makna dari tiap bentuk dan warna dari *node* dan edge.

- Help
  Menampilkan suatu halaman web yang berisi menu bantuan untuk aplikasi ini.
- Tutorial
  Menampilakan suatu halaman yang berisi menu tutorial bagaimanan caranya menggunakan aplikasi ini.
- About CI space.
  Menanpilkan informasi mengenai proyek milik CI Space.
- About This Applet
  Menampilkan identifikasi dari aplikasi ini.

Disamping beberapa fungsi yang disebutkan di atas, terdapat beberapa fungsi ynag terdapat pada bagian toolbar

Bagian Pembuatan :

1. Tombol Create *Node*
   Tombol yang berguna untuk membentuk suatu *node*. Penggunaannya adalah dengan mengklik tombol ini lalu mengklick kanvas kemudian lakukan pengisian terhadap parameter yang muncul di dialog box. Lakukan pengisian terhadap nama *node*, nilai heuristic dan tipe *node* yang menandakan apakah *node* tersebut merupakan goal *node*, regular *node*, atau star *node*. Klik OK untuk menyetujui atau klik cancel untuk menggagalkan operasi.

2. Tombol Create Edge
   Tombol ini berguna untuk membentuk suatu edge. Penggunaannya adalah dengan menklik tombol ini lalu mengklik *node* asal dan kemudian klik *node* yang akan dituju lalu akan muncul dialog box. Lakukan pengisian nilai dari *node* kemudian tekan OK untuk menyetujui atau tekan Cancel untuk menggagalkan.

3. Tombol Select
   Tombol ini berguna untuk meilih *node* atau edge yang terdapat di graph. Penggunannya adalah dengan mengklik tombol ini kmudian lakukan klik *node* atau edge yang akan dipilih

4. Tombol Delete
   Tombol ini berguna untuk menghapus *node* atau edge yang terdapat di graph. Penggunannya adalah dengan mengklik tombol ini kemudian lakukan klik *node* atau edge yang akan dipilih.

5. Tombol Invert graph
   Tombol ini berguna untuk membalikkan semua arah edge yang ada pada graph. Penggunaannya adalah dengan mengklik tombol ini.

6. Tombol Set Properties
   Tombol yang berguna untuk melakukan pengesetan ulang terhadap properties milik *node* atau edge. Penggunaannya adalah dengan mengklik tombol ini kemudian klik *node* atau edge yang akan diupdate propertiesnya kemudian lakukan pengupdateaan pada dialog box yang muncul.

Bagian Pemecahan :

1. Tombol Fine Step
   Bila tombol ini diklik satu kali maka *node* pertama pada *current frontier* menjadi *current node*. Ketika diklik untuk kedua kalinya, semua tetangga dari *current node* diperlihatkan. Ketika diklik untuk ketiga kalinya, tetangga akan dimasukkan ke *frontier* dan *frontier* yang telah diupdate ditandai.

2. Tombol Step
   Setiap tombol ini diklik, pencarian berlanjut mengunjungin satu *node* yang merupakan *current node*. *Node-node* pada *frontier* dan *node* tetangga dari *current node* ditandai. Perhatikan bahwa satu klik dari *Step button* sama dengan tiga kali klik *Fine Step Button.*

3. Tombol Auto Search
   Tombol ini akan membuat computer melakukan pencarian secara otomatis, tanpa peril menekan tombol apapun lagi. Fungsi ini akan berhenti jika *goal node* telah tercapai atau tidak ada solusi ketika pencarian telah berjalan sebanyak 50 langkah.

4. Tombol Stop Search
   Tombol ini berfungsi untuk menghentikan pencarian yang sedang berlangsung ketika memakai fungsi auto search.

5. Tombol Reset Search
   Tombol ini berfungsi untuk mengembalikan pencarian ke awal, yaitu

dari *start node*.

6. Tombol Select
   Tombol ini berfungsi untuk memilih *node* yang ada.

7. Tombol Inspect *Node* Paths
   Tombol ini akan menampilkan jalur-jalur yang bisa ditempuh oleh *agent* sehingga bisa mencapai *node* yang dipilih.

8. Tombol Quiz
   Tombol ini berfungsi untuk mengaktifkan mode *quiz*. Pada mode ini pengguna harus menebak *node* yang benar sampai *goal* tercapai. Untuk mengulangi *quiz* tekan tombol *Reset Search.*

Untuk contoh pengimplementasian perepresentasian graf dalam pencarian rute terpendek, dapat dibuat sebuah graf pada gambar 4

Pada graf tersebut, *node* A merepresentasikan *start node* dan *node* Q merepresentasikan *goal node*, sedangkan *node-node* yang lain adalah rute-rute yang harus ditempuh yang dihubungkan dengan *edge.*

Langkah-langkah detail dari pencarian rute terpendek tersebut adalah dengan membandingkan nilai *edge* yang akan ditempuh dan nilai *heuristic* yang ada pada *node* yang akan ditempuh dengan menggunakan persamaan (3)

A → B f(N) = 66,3 + 12,6 = 78,9
A → C f(N) = 61,2 + 10,3 = 71,5
A → D f(N) = 61,7 + 21,7 = 83,4

Maka dipilihlah rute A → C dengan nilai f(N) terkecil.

Kemudian, dilakukan pengecekan, apakah A merupakan *goal node.* Karena A bukan merupakan *goal node*, maka pencarian diteruskan menjadi

A → C → G f(N) = 50,6 + 22,8 = 71,5

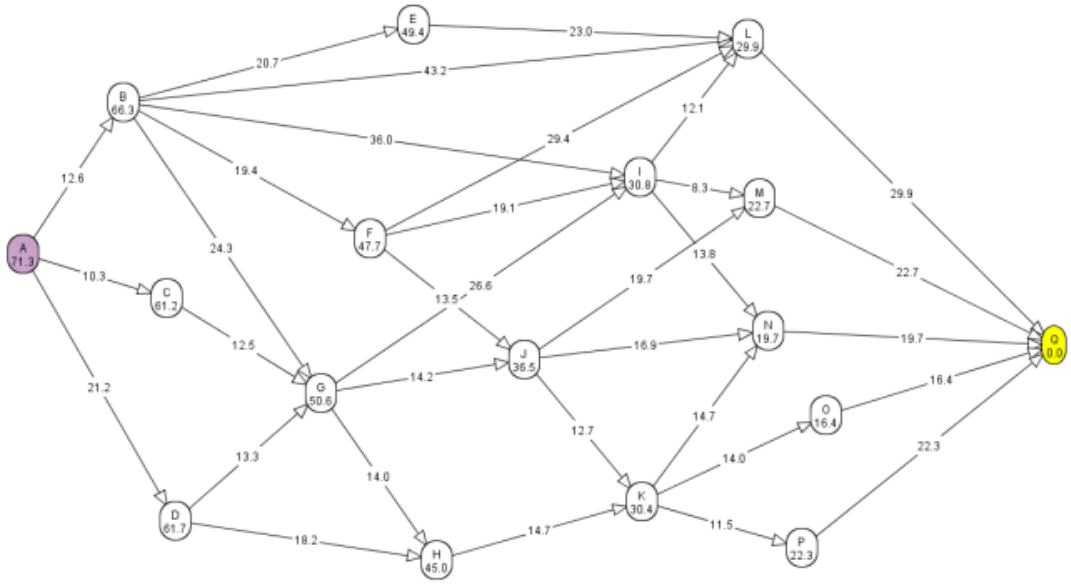
Sehingga terdapat beberapa pilihan

A → C → G → I f(N) = 30,8 + 49,4 = 80,2
A → C → G → J f(N) = 36,5 + 37,0 = 73,5
A → C → G → H f(N) = 45,0 + 36,8 = 81,8

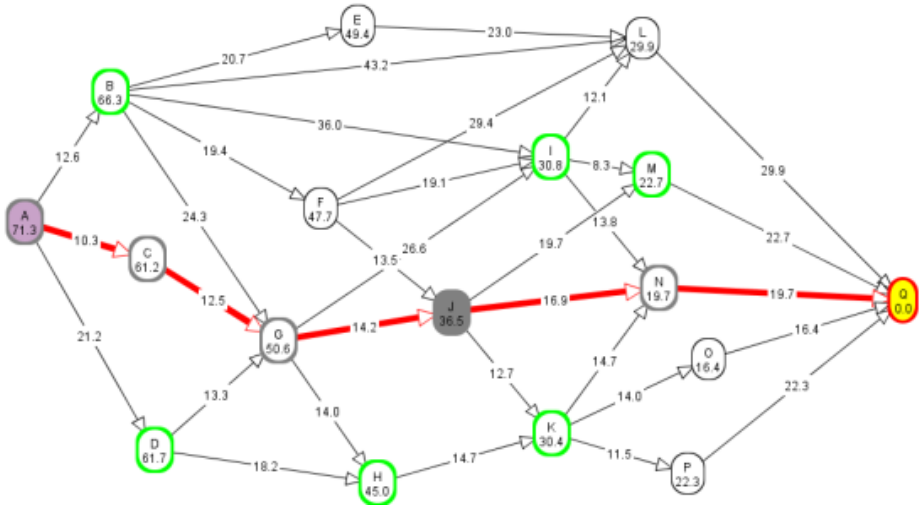
Maka dipilihlah rute A → C → G→J dengan nilai f(N) terkecil.

Pencarian ini dilakukan sampai *node* Q sebagai *goal node* ditemukan.

Setelah dilakukam pencarian dengan algoritma A*, maka graf tersebut akan menjadi seperti graf pada gambar 5.



Gambar 4 Contoh Representasi Graf Dalam Pencarian Rute Terpendek



Gambar 5 Contoh Representasi Graf Dalam Pencarian Rute Terpendek setelah dilakukan pencarian dengan A*

6. KESIMPULAN

Graf dapat dipakai dalam menyelesaikan permasalahan pencarian rute terpendek yang menggunakan algoritma A*.

Untuk merepresentasikan graf dalam menyelesaikan permasalahan dengan pencarian rute terpendek, dapat digunakan aplikasi *Graph Search* yang bisa membuat graf sesuai dengan keinginan kita.

DAFTAR REFERENSI

[1] Munir, Rinaldi (2004). Bahan Kuliah IF2153 Matematika Diskrit, Departemen Teknik Informatika, Bandung

[2] Graph Theory http://en.wikipedia.org/wiki/Graph_theory. Tanggal akses : 26 Desember 2007 Pukul 21.00

[3] A* Search Algorithm http://en.wikipedia.org/wiki/A%2A_search_algorithm. Tanggal akses : 26 Desember 2007 Pukul 22.00

[4] Shortest Path Problem http://en.wikipedia.org/wiki/Shortest_path_problem. Tanggal akses : 26 Desember 2007 Pukul 21.30

[5] Graph Search http://www.cs.ubc.ca/labs/lci/CIspace/version4/search. Tanggal akses : 27 Desember 2007 Pukul 23.30