RANCANG BANGUN PERANGKAT LUNAK VISUALISASI GRAFIS ALGORITMA DIJKSTRA

Didik Hariyanto [1], Yuwono Indro Hatmojo [2]

[1,2] Jurusan Pendidikan Teknik Elektro, Fakultas Teknik, Universitas Negeri Yogyakarta
[1] didik_hr@uny.ac.id , [2] hatmojo_yu@yahoo.com

ABSTRAK

Tulisan ini membahas tentang bagaimana membuat perangkat lunak yang dapat digunakan untuk menampilkan secara visual grafis dari algoritma pencari jalur terpendek (dijkstra). Salah satu keuntungan dari dibuatnya perangkat lunak ini adalah membantu user dalam memahami algoritma pemrograman yang digunakan sebagai bagian dari bentuk media pembelajaran yang interaktif.

Pembuatan perangkat lunak dalam tulisan ini dengan berdasarkan pada metode rancang bangun software dengan model sequential. Dimana tahap awal yang dilakukan adalah analisis, yang terdiri dari analisis kebutuhan pemakai, analisis kerja dan analisis teknologi. Tahap selanjutnya adalah perancangan atau desain yang meliputi desain Data Flow Diagram, desain menu dalam bentuk HIPO (Hierarchy plus Input-Proses-Output) dan desain tampilan. Setelah itu dilakukan tahap menterjemahkan modul-modul hasil desain dengan menggunakan bahasa pemrograman ke dalam bentuk aplikasi atau biasa disebut coding/implementation. Tahap terakhir adalah pengujian perangkat lunak dengan menggunakan sistem pengujian Black Box Testing.

Dari hasil pengujian dengan menggunakan sistem pengujian Black Box Testing yang terdiri dari 22 item pengujian, didapatkan hasil bahwa semua item pengujian dapat dilakukan dengan baik dan benar atau didapatkan performance system sebesar 100 %.

Kata Kunci : rancang bangun, visualisasi, grafis, algoritma, djikstra

1. Pendahuluan

Salah satu upaya untuk membuat suatu teori atau model matematis dapat lebih mudah dipahami adalah dengan mencoba menghadirkan sesuatu bentuk yang semula abstrak menjadi bentuk visual yang mudah dimengerti.

Visualisasi dalam bentuk grafis yang dikembangkan dengan menggunakan suatu *software* merupakan suatu cara yang efektif dalam menghadirkan suatu bentuk kasat mata akan suatu teori atau model matematis. Dengan adanya bentuk visual dan ditambahkan proses yang interaktif, maka akan sangat membantu *user* untuk lebih mudah memahami dan juga dapat melakukan proses belajar secara mandiri.

Algoritma pencari jalur terpendek dengan model Dijkstra merupakan suatu algoritma yang digunakan untuk mencari jalur yang paling cepat antara satu *node* (titik) dengan *node* yang lain diantara banyak *node* yang saling terhubung oleh *edge* (garis). Algoritma ini banyak digunakan sebagai salah satu bentuk pemecahan masalah dalam hal penentuan jarak tercepat antar kota, proses *routing* dalam jaringan komputer, dll. Untuk memahami cara kerja dari algoritma ini, dibutuhkan suatu proses yang relatif lama dan dibutuhkan konsentrasi yang baik.

Dengan menggunakan metode secara visual dalam bentuk grafis yang interaktif, *user* dapat lebih cepat dan mudah untuk memahami algoritma ini, dan *user* dapat melakukan berbagai percobaan contoh-contoh kasus secara cepat dan tepat. Percobaan contoh-contoh kasus dapat dilakukan secara mandiri dengan menciptakan skenario-skenario permasalahan yang ingin dipecahkan.

Pertanyaan yang muncul kemudian adalah : 1) Bagaimana rancang bangun perangkat lunak visualisasi grafis algoritma pencari jalur terpendek (*Dijkstra*) ?, 2) Bagaimana unjuk kerja perangkat lunak tersebut bila dilakukan pengujian dengan menggunakan sistem pengujian *Black Box Testing* ?

2. Visualisasi Grafis

Sesuai yang disampaikan oleh Joko Purwadi dalam Seminar Nasional Ilmu Komputer & Teknologi Informasi pada Agustus 2005, tampilan program sistem secara grafis, secara ideal akan membantu pemahaman pemakai dalam mempergunakan/mengoperasikan sistem. Beberapa karakteristik yang sangat mempengaruhi dalam mendesain tampilan program sistem dengan desain grafis yang terstruktur dan baik, yaitu :

a. Bentuk penampilan, struktur secara fisik dan hubungan antar bagian secara keseluruhan
b. Penempatan posisi, gerakan fisik menu, alur sistem, dan proses interaksi yang terjadi terhadap suatu perubahan terjadi hubungan yang interaktif
c. Ukuran bentuk tampilan, jumlah tampilan per modul menu, kecenderungan penggunaan, pengelompokkan fungsi menu yang serasi atau seragam

Penempatan posisi atau pemetaan menu disesuaikan dengan obyek yang berelasi, dengan

mempertimbangkan obyek lain dalam program sistem

3. Algoritma Pencari Jalur Terpendek (Dijkstra)

Berdasarkan terminologi teori graph, maka suatu jaringan akan terdiri dari suatu himpunan titik-titik yang disebut *node*. Node-node tersebut saling dihubungkan oleh suatu garis dan disebut *edge* (V.K. Balakrishnan : 1997).

Untuk setiap dua node dapat terjadi beberapa lintasan, dimana lintasan dengan bobot yang minimum disebut sebagai lintasan atau rute terpendek. Bobot di sini dapat berupa jarak, waktu tempuh, atau ongkos transportasi dari satu node ke node yang lainnya yang membentuk rute tertentu (Gary Chartrand & Ortrud R. Oellermann : 1993).

Algoritma mencari rute terpendek ini dikembangkan oleh Dijkstra (Jong Jek Siang: 2002).

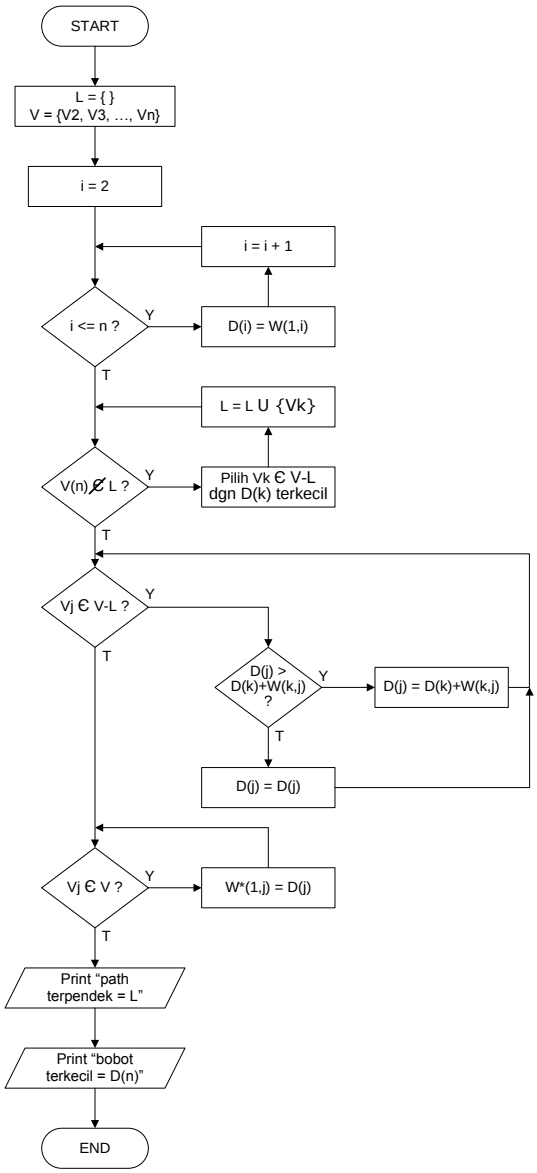
Misalkan :
V(G) = $\{V_1,V_2,...,V_n\}$
L = Himpunan titik-titik Є V(G) yang sudah terpilih dalam alur path terpendek
D(j) = Jumlah bobot path terkecil dari V_1 ke V_j
W(i,j) = Bobot garis dari titik V_i ke titik V_j
W*(1,j) = Jumlah bobot terkecil dari V_1 ke V_j

Maka, algoritma Dijkstra untuk mencari path terpendek adalah sebagai berikut :

1. L = { };
   V = $\{V_2, V_3, ..., V_n\}$
2. Untuk i = 2, ..., n, lakukan D(i) = W(1,i)
3. Selama V_n Є L lakukan :
   a. Pilih titik V_k Є V-L dengan D(k) terkecil
      L = L U $\{V_k\}$
   b. Untuk setiap V_j Є V-L lakukan :
      Jika D(j) > D(k) + W(k,j) maka ganti D(j) dengan D(k) + W(k,j)
4. Untuk setiap V_j Є V, W*(1,j) = D(j)

   Menurut algoritma diatas, path terpendek dari titik V_1 ke V_n adalah melalui titik-titik dalam L secara berurutan dan jumlah bobot path terkecilnya adalah D(n).

Bila digambarkan dalam bentuk *flowchart* akan terlihat seperti gambar 1 di bawah ini.



Gambar 1. Flowchart Algoritma Dijkstra

4. Pembahasan

Rancang bangun perangkat lunak visualisasi grafis algoritma pencari jalur terpendek (*Dijkstra*) menggunakan model *sequential* (Pressman : 2002).

Adapun tahapan-tahapan dalam pengembangan perangkat lunak ini, yaitu :

4.1. Analisis

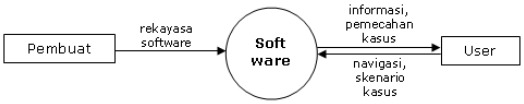
Pada tahapan analisis ini yang dilakukan adalah analisis kebutuhan pemakai. Hasil identifikasi dari analisis kebutuhan pemakai ini adalah:

a. Perangkat lunak diharapkan dapat menarik minat pemakai untuk menggunakan program ini sebagai media visualisasi algoritma pencari jalur terpendek (*Dijkstra*).
b. Perangkat lunak harus mudah digunakan.
c. Perangkat lunak dapat digunakan untuk memecahkan kasus-kasus pencarian jarak terpendek.

4.2. Desain

Setelah data pada tahap analisis terkumpul, maka tahapan selanjutnya adalah membuat desain. Desain dapat didefinisikan sebagai proses penerapan berbagai macam-macam teknik dan prinsip dengan tujuan untuk mendefinisikan peralatan, proses atau sistem secara rinci sehingga mudah dalam penerapannya. Adapun rincian desain yang digunakan adalah sebagai berikut :

a. Context Diagram



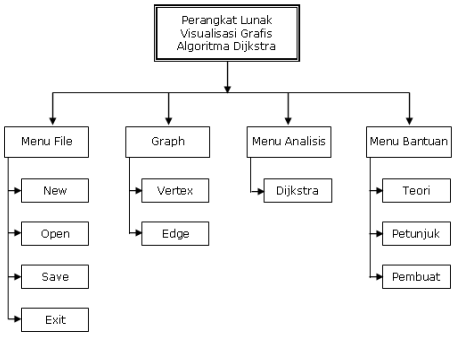
Gambar 2. *Data Flow Diagram* (DFD) Level 0

Context diagram atau yang disebut dengan *Data Flow Diagram* (DFD) Level 0 merupakan alat yang digunakan untuk mendokumentasikan proses dalam sistem. Tujuannya adalah memberikan pandangan proses dalam sistem secara umum. Ada pihak yang memberi masukan pada sistem dan ada pihak yang menerima keluaran dari sistem.

Pada Gambar 2 di atas, pembuat adalah orang yang melakukan rekayasa terhadap *software*. Sedangkan user adalah pengguna yang dapat melakukan navigasi dan meminta pemecahan masalah dari skenario kasus yang diinginkan.

b. Desain menu

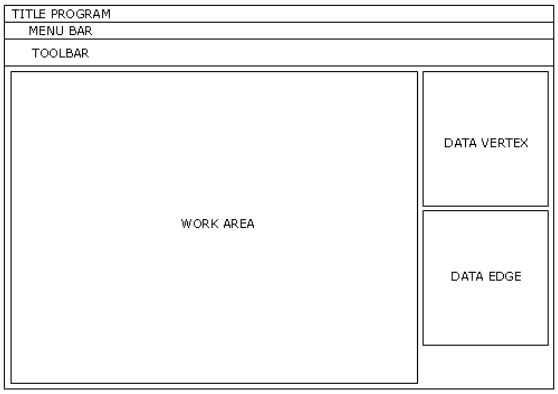
Desain menu dari *software* ini, dibuat dengan model HIPO *(Hierarchy Input-Proses-Output)*. Desain menu ini dibuat dengan maksud mempermudah pemrogram dalam membuat aplikasinya. Untuk lebih jelasnya mengenai desain menu ini dapat dilihat pada Gambar 3.



Gambar 3. Desain Menu

c. Desain tampilan

Desain tampilan/antarmuka dibuat untuk memudahkan *programmer* dalam menterjemahkan ke dalam bentuk bahasa pemrograman. Desain tampilan dalam aplikasi ini menggunakan desain tampilan yang sesuai dengan desain aplikasi pemrograman visual, seperti terlihat pada gambar di bawah ini.

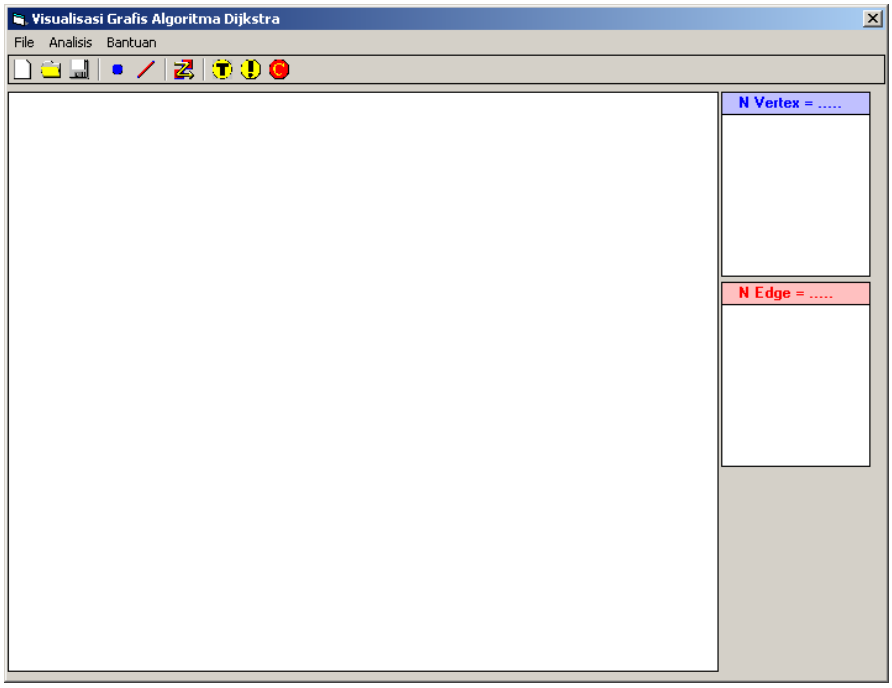


Gambar 4. Desain Tampilan

4.3. Implementasi

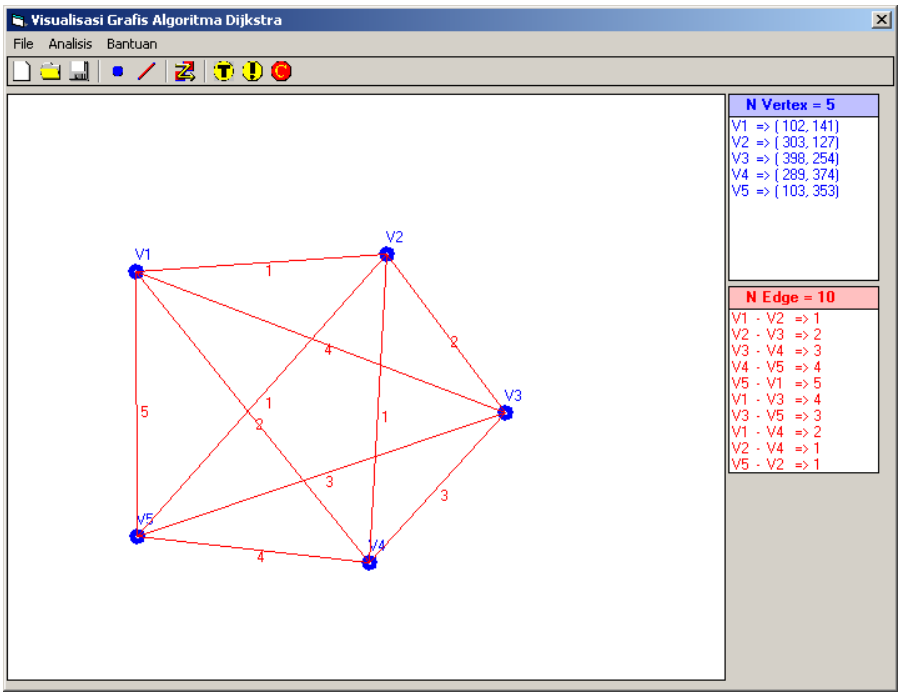
Pada tahap implementasi ini merupakan hasil dari terjemahan tahap desain ke dalam bentuk bahasa pemrograman. Bahasa pemrograman yang dipakai adalah Microsoft Visual Basic versi 6.0. Dengan bantuan *contect diagram*, desain menu dan desain tampilan yang telah dibuat maka pembuatan program dalam bentuk prosedur-prosedur dan modul-modul lebih mudah.

Adapun hasil *implementasi* pembuatan perangkat lunak dapat dilihat pada gambar di bawah ini.



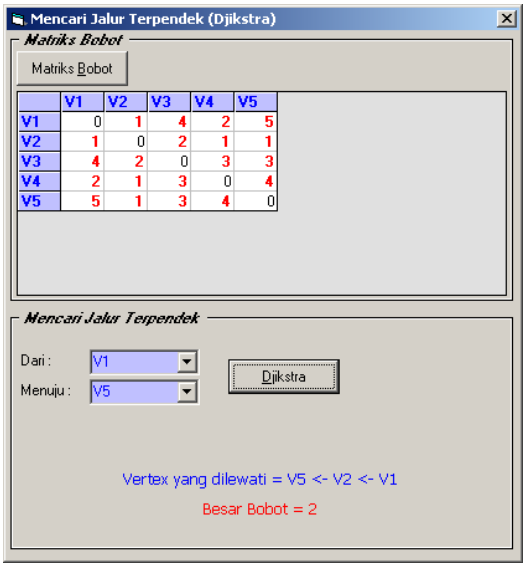
Gambar 5. Tampilan Utama Hasil Implementasi

Di bawah ini, gambar yang menunjukkan hasil aplikasi program pada saat diberikan masukan berupa Vertex dan Edge beserta Bobot untuk masing-masing Edge.



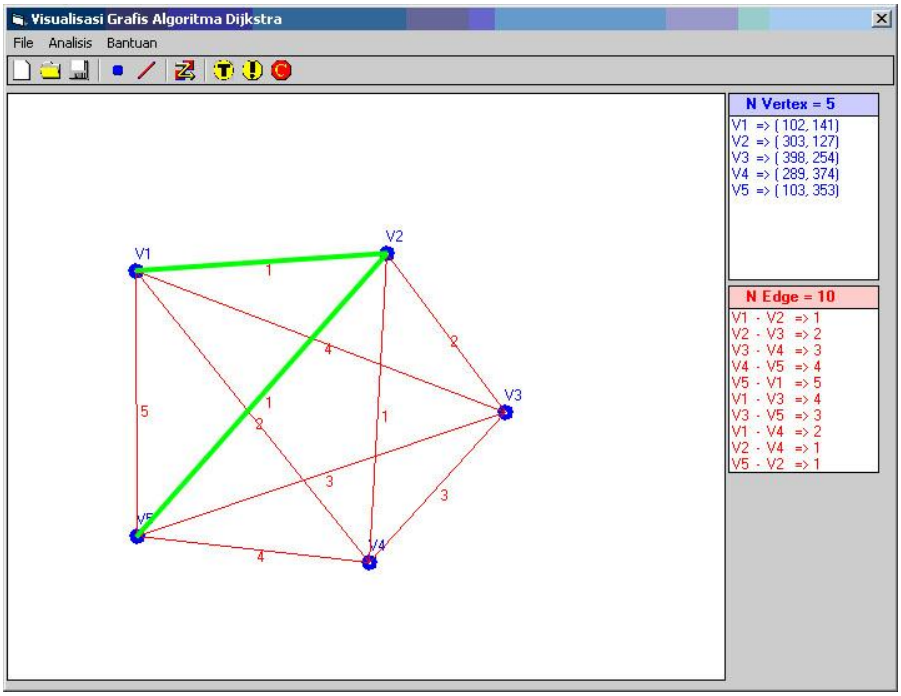
Gambar 6. Tampilan pada saat ada masukan Data Vertex dan Edge

Proses perhitungan Matrik Bobot dan pencarian jarak terpendek dari titik V1 ke titik V5 beserta nilai Bobot dari jarak terpendek dan titik-titik yang dilewati.



Gambar 7. Proses perhitungan algoritma Dijkstra

Hasil pencarian jarak terpendek dari titik V1 ke titik V5 dengan menggunakan algoritma Dijkstra diperlihatkan dengan garis berwarna hijau dan tebal (V1-V2-V5) seperti terlihat pada gambar di bawah ini.



Gambar 8. Hasil jarak terpendek algoritma Dijkstra

4.4. Pengujian

Tahap pengujian dilakukan untuk mengetahui unjuk kerja atau *performance system*. Pengujian perangkat lunak (*software*) visualisasi grafis algoritma Dijkstra menggunakan metode pengujian *Black Box Testing*.

Pengujian *Black Box Testing* dilakukan oleh tim pembuat perangkat lunak untuk mengetahui fungsi-fungsi dalam program dapat berjalan dengan benar. Dalam pengujian ini terdapat 22 item yang diujikan, seperti terlihat pada tabel di bawah ini.

Tabel 1. Tabel Pengujian *Black Box Testing*

No	Item Pengujian	Hasil		Keterangan
		Bisa	Tidak	
1.	Fungsi Menu Bar	√		sesuai scr visual
2.	Fungsi ToolBar	√		sesuai scr visual
3.	Fungsi Shortcut	√		sesuai scr visual
4.	Fungsi untuk membuat File Graph baru	√		sesuai scr visual

5.	Fungsi untuk menyimpan File Graph	√		sesuai scr visual
6.	Fungsi untuk membuka File Graph	√		sesuai scr visual
7.	Fungsi untuk membuat Vertex	√		sesuai scr visual
8.	Fungsi untuk membuat Edge	√		sesuai scr visual
9.	Fungsi untuk memberikan Bobot untuk Edge	√		sesuai scr visual
10.	Fungsi untuk menampilkan Data Vertex	√		sesuai scr visual
11.	Fungsi untuk menampilkan Data Edge	√		sesuai scr visual
12.	Fungsi untuk menampilkan Data Bobot masing-masing Edge	√		sesuai scr visual
13.	Fungsi untuk menghitung Matrik Bobot	√		sesuai scr visual
14.	Fungsi untuk menentukan Vertex awal	√		sesuai scr visual
15.	Fungsi untuk menentukan Vertex akhir	√		sesuai scr visual
16.	Hasil pencarian jarak terpendek berupa Bobot	√		sesuai scr visual
17.	Hasil pencarian jarak terpendek berupa Vertex-Vertex yang dilalui	√		sesuai scr visual
18.	Hasil pencarian jarak terpendek berupa jalur garis dengan warna hijau	√		sesuai scr visual
19.	Fungsi untuk menampilkan Teori Dijkstra	√		sesuai scr visual
20.	Fungsi untuk menampilkan Petunjuk Penggunaan Program	√		sesuai scr visual
21.	Fungsi untuk menampilkan Pembuat Program	√		sesuai scr visual
22.	Fungsi untuk mengakhiri program (Exit)	√		sesuai scr visual

Unjuk kerja atau *performance system* perangkat lunak dalam tulisan ini diukur dengan melakukan pengujian terhadap perangkat lunak dengan menggunakan pengujian *Black Box Testing*. Pengujian yang dilakukan terdiri dari 22 item pengujian sesuai dengan tabel 1. Pengujian dilakukan oleh tim pengembang dan pembuat perangkat lunak untuk mengetahui bahwa semua fungsi yang terdapat dalam perangkat lunak sudah dapat berjalan sesuai dengan spesifikasi program. Dari ke-22 item pengujian, semuanya dapat dilakukan dengan baik, benar dan sesuai dengan pengamatan hasil secara visual atau bisa disimpulkan 100% item pengujian dapat berjalan sesuai dengan spesifikasi program.

Namun ada beberapa keterbatasan dari sistem yang dibangun, dimana pengujian dilakukan dengan jumlah maksimal node sebanyak 89 dan edge sebanyak 248. Bila dikaji secara teoritis, sistem mampu menampung sebanyak 676 node, dikarenakan daya tampung *workarea* pada sistem terbatas pada ukuran 576 x 470 pixel, sedangkan radius pembuatan node ditambah dengan jarak bebas mempunyai ukuran ±20 pixel. Sehingga dari ukuran *workarea* sebesar 576 x 470 = 270.720 dibagi dengan 20 x 20 = 400 didapatkan hasil sejumlah 676 node yang mampu ditampung.

5. Kesimpulan dan Saran

a. Rancang bangun perangkat lunak visualisasi grafis algoritma Dijkstra ini menggunakan metode *sequential* yang terdiri dari tahap analisis, desain, implementasi, dan pengujian. Sistem pengujian yang digunakan adalah *Black Box Testing*.
b. Unjuk kerja perangkat lunak visualisasi grafis algoritma Dijkstra dalam tulisan ini diukur dengan melakukan pengujian terhadap perangkat lunak dengan menggunakan pengujian *Black Box Testing*. Pengujian yang dilakukan terdiri dari 22 item pengujian. Pengujian dilakukan oleh tim pengembang dan pembuat perangkat lunak untuk mengetahui bahwa semua fungsi yang terdapat dalam perangkat lunak sudah dapat berjalan sesuai dengan spesifikasi program. Dari ke-22 item pengujian, semuanya (100% item pengujian) dapat dilakukan dengan baik dan benar.

Daftar Pustaka:

Gary Chartrand & Ortrud R. Oellermann, 1993, *"Applied and Algorithmic Graph Theory"*, New York : McGraw-Hill Inc.

Joko Purwadi & Jazi Eko Istiyanto, *"Pemrograman Interaktif SIPP : Program Informasi Pengaturan dan Penjadwalan Parkir Berbasis Cerdas"*, http://jazi.staff.ugm.ac.id/E01%20-%20Joko%20Purwadi.pdf, diakses tanggal 13 April 2007.

Jong Jek Siang, 2002, *"Matematika Diskrit dan Aplikasinya pada Ilmu Komputer"*, Yogyakarta : Andi Offset.

Pressman SR, 2002, "*Software Engineering*", Singapore : McGraw-Hill.

V.K. Balakrishnan, Ph.D., 1997, *"Schaum's Outline of Theory and Problems of Graph Theory"*, New York : McGraw-Hill Inc.