BAB II

DASAR TEORI

2.1 Teori Graf

Graf adalah kumpulan simpul (*nodes*) yang dihubungkan satu sama lain melalui sisi/busur (*edges*). Suatu Graf G terdiri dari dua himpunan yaitu himpunan V dan himpunan E.

a. Verteks (simpul) : V = himpunan simpul yang terbatas dan tidak kosong
b. Edge (sisi/ruas) : E = himpunan busur yang menghubungkan sepasang simpul.

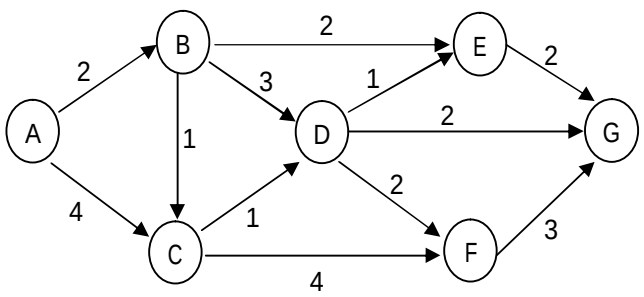
Simpul-simpul pada graf dapat merupakan obyek sembarang seperti kota, atom-atom suatu zat, nama anak, jenis buah, komponen alat elektronik dan sebagainya. Busur dapat menunjukkan hubungan (relasi) sembarang seperti rute penerbangan, jalan raya, sambungan telepon, ikatan kimia, dan lain-lain. Notasi graf: G(V,E) artinya graf G memiliki V simpul dan E busur.

2.1.1 Macam-macam Graf

Menurut arah dan bobotnya, graf dibagi menjadi empat bagian, yaitu :

1. Graf berarah dan memiliki bobot

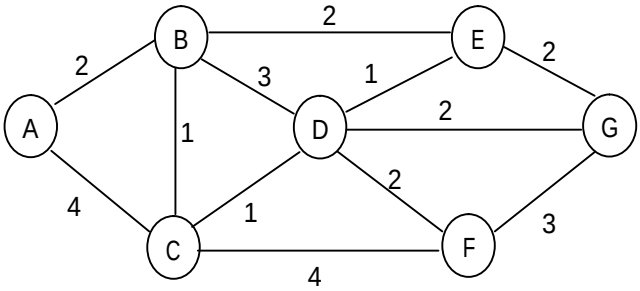
   Tiap busur mempunyai anak panah dan bobot. Gambar 2.1 menunjukkan graf berarah dan berbobot yang terdiri dari tujuh titik yaitu titik A,B,C,D,E,F,G. Titik menujukkan arah ke titik B dan titik C, titik B menunjukkan arah ke titik D dan titik C, dan seterusnya. Bobot antar titik A dan titik B pun telah di ketahui.



Gambar 2.1 *Graf berarah dan memiliki bobot*

2. Graf tidak berarah namun memiliki berbobot

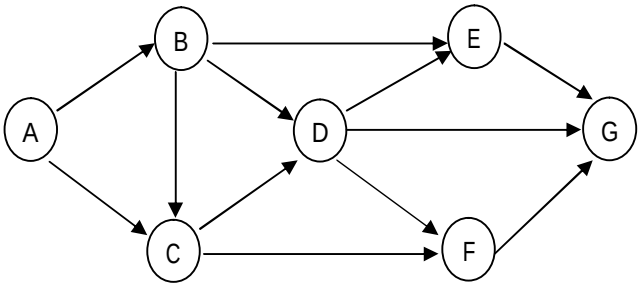
   Tiap busur tidak mempunyai anak panah tetapi mempunyai bobot. Gambar 2.2 menunjukkan graf tidak berarah dan berbobot. Graf terdiri dari tujuh titik yaitu titik A,B,C,D,E,F,G. Titik A tidak menunjukkan arah ke titik B atau C, namun bobot antara titik A dan titik B telah diketahui. Begitu juga dengan titik yang lain.



Gambar 2.2 *Graf tidak berarah namun tidak memiliki bobot*

3. Graf berarah namun tidak memiliki bobot

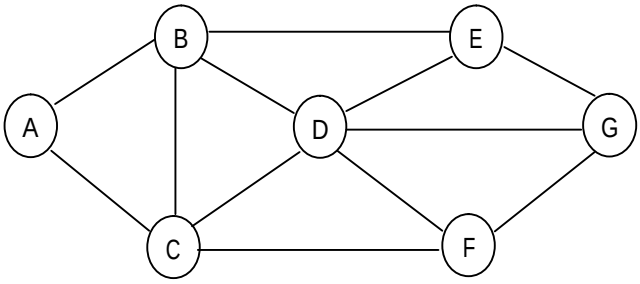
   Tiap busur mampunyai anak panah yang tidak berbobot. Gambar 2.3 menunjukan graf berarah dan tidak berbobot.



Gambar 2.3 Graf berarah namun tidak memiliki bobot

4. Graf tidak memiliki arah dan tidak memiliki bobot

   Tiap busur tidak mempunyai anak panah dan tidak berbobot. Gambar 2.4 menunjukan graf tidak berarah dan tidak berbobot.



Gambar 2.4 *Graf tidak memiliki arah dan tidak memiliki bobot*

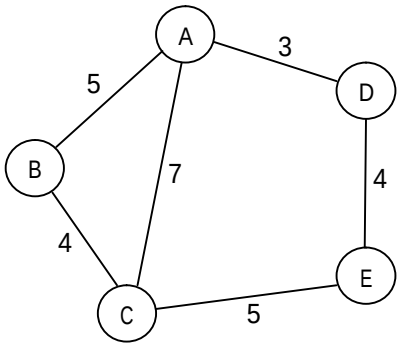
2.2 Definisi Jalur Terpendek

Lintasan terpendek adalah lintasan minimum yang diperlukan untuk mencapai suatu tempat dari tempat tertentu. Lintasan minimum yang dimaksud dapat dicari dengan menggunakan graf. Graf yang digunakan adalah graf yang tidak berbobot, yaitu graf yang tidak memiliki suatu nilai atau bobot.

Ada beberapa macam persoalan lintasan terpendek, antara lain:

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

2.3 Pencarian Jalur Terpendek



Gambar 2.5 *Ilustrasi Graf Jalur Terpendek tidak berarah namun memiliki bobot*

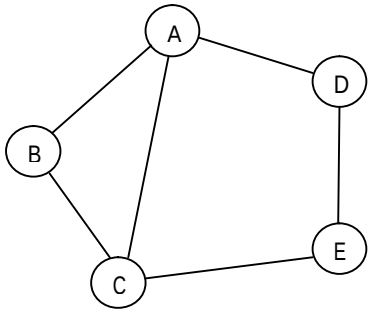
Pada dasarnya permasalahan pencarian jalur terpendek antar kota merupakan pencarian jalur terpendek antar titik yang telah diketahui koordinatnya. Dengan mengetahui konsep pencarian jalur terpendek antar titik, untuk selanjutnya dapat diterapkan pada pencarian jalur terpendek pada berbagai kota yang ingin diketahui. Contoh kasus yang akan diambil adalah pencarian jalur terpendek antara titik A dan titik E.

Terdapat dua jenis kasus yang bias diturunkan dari gambar 2.5. Kasus pertama adalah mengetahui jarak antar node yang ditunjukkan dengan garis penghubung antar titik.. Kasus yang kedua adalah dengan dengan mengetahui koordinat titik saja. Gambar 2.5 merupakan jenis kasus yang pertama yaitu dengan mengetahui jarak antar titik. Sedangkan gambar 2.6 merupakan jenis kasus yang kedua, yaitu dengan mengetahui titik koordinatnya saja. Untuk kasus pertama, penyelesaian cenderung lebih mudah karena jarak antar titik telah diketahui sebagai berikut:

Tabel 2.1 *Tabel Jarak antar Titik*

	A	**B**	**C**	**D**	**E**
A	0	5	7	3	-
B	5	0	4	-	-
C	7	4	0	-	5
D	3	-	-	0	4
E	-	-	5	4	0

Sedangkan untuk kasus kedua yang telah ditunjukkan pada Gambar 2.6, misalnya titik–titik yang telah ditentukan mempunyai koordinat sebagaimana tabel 2.2.



Gambar 2.6 *Ilustrasi Graf Jalur Terpendek tidak memiliki arah dan tidak memiliki bobot*

Tabel 2.2 *Koordinat titik antar kota*

	X	**Y**
A	20	50
B	10	25
C	18	10
D	50	40
E	55	15

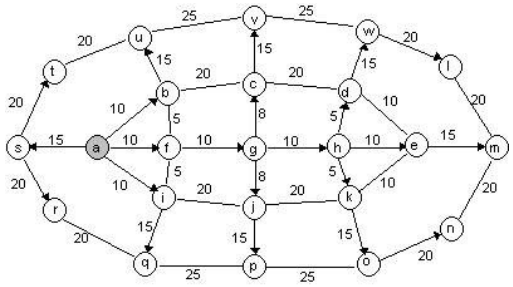
Karena belum diketahui jarak antar titiknya dan hanya diketahui titik koordinat saja, Maka perhitungan dimulai dari penentuan jarak antar simpul titik dengan dengan menggunakan titik yang diketahui. Titik yang diketahui menggunakan koordinat sumbu X dan Y. Langkah di atas merupakan langkah yang harus dilakukan untuk semua metode.

2.4 Algoritma Label-Setting

Untuk lebih mengetahui algoritma Label-Setting, misalkan terdapat suatu jaringan yang sederhana, berarah, dan memiliki grafik terhubung, di mana edge masing-masing memiliki nilai negatif, yang disebut panjang edge. Simpul jaringan dilambangkan oleh bilangan bulat positif. Verteks dengan indeks terbesar *N*, (*i, j*) menunjukkan edge dari simpul *i* ke simpul *j*, dan *h* (*i, j*) panjang edge ini. Dengan jalur dari simpul *i* ke simpul *j* yang kita maksud *P (i, j)* = (*i1, j1*), . . . , (*ik, jk*) dari edge, dengan $i_1 = i$, $j_k = j$ dan $j_L = i_{L+1}$, $1 \leq L < k$. Simpul *i, j* disebut titik awal dan titik akhir. Dengan panjang jalur yang kita maksudkan adalah jumlah dari panjang edge yang sepanjang jalur ini. Jika jalur memiliki awal yang sama dan titik akhir, kemudian kita sebut itu siklus. Sebuah jaringan disebut simetris jika keberadaan edge (*i, j*) sama dengan edge (*j, i*) dan $h(i, j) = h(j, i)$. Contoh pada gambar 1 dan 2 merupakan jalur yang simetris, karena itu kita tidak menulis petunjuk untuk edge pada angka, pada kenyataannya, masing-masing ujung pada angka mewakili dua sisi. Dalam pekerjaan ini, simetri tidak penting.

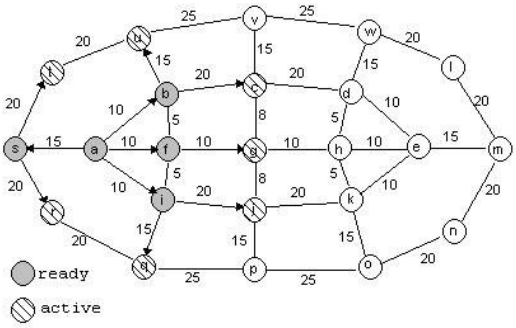
Sebuah subnetwork disebut pohon terarah jika tidak mengandung siklus, dan ada simpul di dalamnya, yang disebut root atau titik awal dengan properti berikut: untuk setiap titik dalam subnetwork yang berbeda dari akar, ada tepat satu

jalur dari akar untuk itu. Dengan pohon yang minimal kita maksud seperti pohon diarahkan di mana setiap jalur memiliki panjang minimal di antara jalur menuju dari titik awal ke titik akhir dalam asli jaringan (lihat Gambar 2.7, simpul 1.).



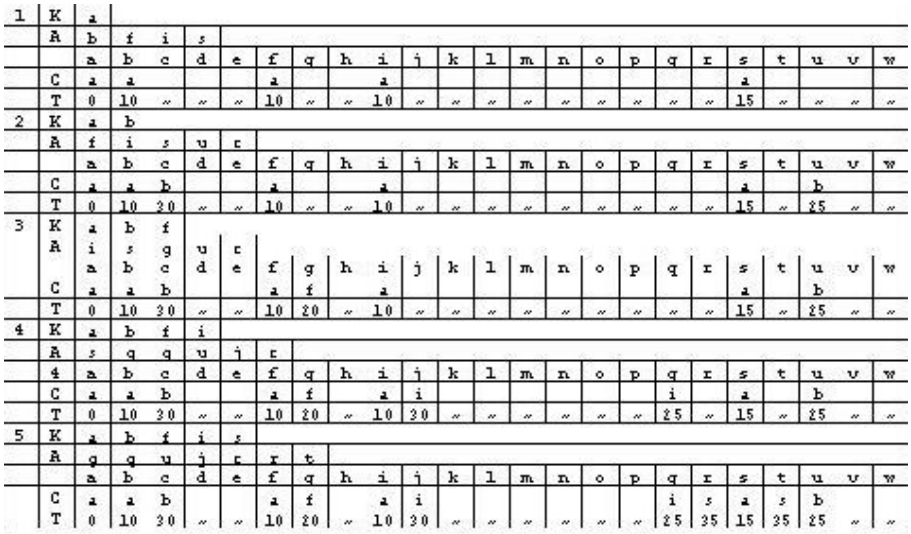
Gambar 2.7 *Jalur Sederhana*

Kita dapat dengan mudah menggambarkan pohon diarahkan dengan label mana untuk setiap simpul (kecuali akar), label menunjukkan titik sebelumnya di jalur terkemuka dari akar ke simpul dipertimbangkan. Dalam diarahkan pohon, masing-masing titik memiliki jarak yang panjang jalur dari akar (lihat gambar 2.8). Untuk simpul *i*, misalkan *c (i)* dan *t (i)* menunjukkan label dan jarak, masing-masing. Jarak dari akar sesuai definisi adalah 0, dan jika perlu, dari sudut pandang teknis, kita juga akan memberikan label kepada akar. Label ini dapat dibedakan label apapun (misalnya akar itu sendiri atau 0).



Gambar 2.8 *Algoritma Label-Setting sederhana*

Tabel 2.3 *Algoritma Label-Setting sederhana*



Algoritma untuk menentukan pohon-pohon minimal dengan beberapa teknik membangun pohon yang penting dari baik teoritis dan praktis sudut pandang. Salah satu alasan adalah bahwa ada tidak diketahui lebih efisien algoritma untuk mencari jalur terpendek lokal untuk dua simpul. Selanjutnya, algoritma ini memiliki implementasi komputer yang mempertahankan efisiensi teoritis. Properti yang terakhir ini tidak berlaku untuk algoritma lain yang dikenal untuk menemukan jalur terpendek antara setiap pasangan simpul, dengan demikian, dalam masalah-masalah praktis dengan jaringan yang sangat besar itu bisa sangat sulit untuk menggunakan prosedur.

2.4.1 Algoritma Label-Setting sederhana

Algoritma Label-Setting secara garis besar dapat ditulis sebagai berikut :

Inisialisasi

- K sebagai simpul akhir.
- Misalkan A adalah himpunan simpul aktif (set aktif). Sebuah titik aktif jika memiliki label temporal dan jarak yang terbatas namun, penyelidikan dari titik belum selesai.

Algoritma

Pada umumnya algoritma Label-Setting terdiri dari 4 tahapan, berikut adalah penjelasannya :

1. Tetapkan jarak ke akar adalah 0 dan jarak tak terbatas untuk simpul selanjutnya. Biarkan label dari akar menjadi dirinya sendiri. Selanjutnya, marilah K adalah himpunan kosong, dan A adalah himpunan tunggal yang berisi akar. Lanjutkan ke Langkah 2.
2. Pilih simpul dari A dengan jarak minimal dan melambangkannya dengan *i*. (Jika ada lebih simpul dengan jarak minimal, kemudian pilih yang terkecil) Hapus *i* dari A., dan masukan ke-K. Lanjutkan ke Langkah 3.
3. Untuk setiap sisi (i, j) dengan t (i) + h (i, j) < t (j):

   - C1. Jika j $\notin$ A, kemudian dimasukkan j menjadi A, ganti label dan jarak sebagai berikut:

   $$c\ (j): = i,\ t\ (j): = t\ (i) + h\ (i, j).$$

   - C2. Jika j $\in$ A, maka memodifikasi label dan jarak j oleh

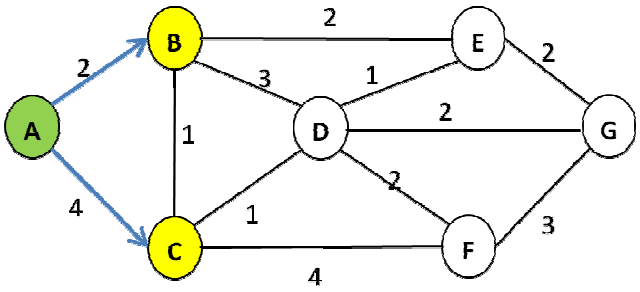
   $$c\ (j): = i,\ t\ (j): = t\ (i) + h\ (i, j).$$

4. Hentikan jika A kosong. Kemudian, label menentukan pohon yang minimal. Jika tidak, lanjutkan Langkah 2[2].

2.4.2 Proses iterasi pada Algoritma Label-Setting

Pada dasarnya Algoritma Label-Setting mencari jalur terpendek ke semua node yang ada pada jaringan dan memberikan label pada jalur yang dianggap sebagai jalur terpendek. Apabila terdapat jalur lain dengan tujuan node yang sama, maka Algoritma Label-Setting akan membandingkan *bobot* dari jalur baru dengan jalur lama, apabila jalur baru memiliki nilai *bobot* yang lebih kecil maka akan memberikan label pada jalur baru dan menghilangkan label pada jalur yang lama.

Berikut contoh proses iterasi pada Algoritma Label-Setting. Dengan menggunakan graf tidak berarah namun memiliki bobot. Sebut saja **K** sebagai simpul akhir, **A** sebagai simpul aktif dan **T** sebagai total *Bobot* dihitung dari node asal. Node asal pada contoh ini adalah pada node A.

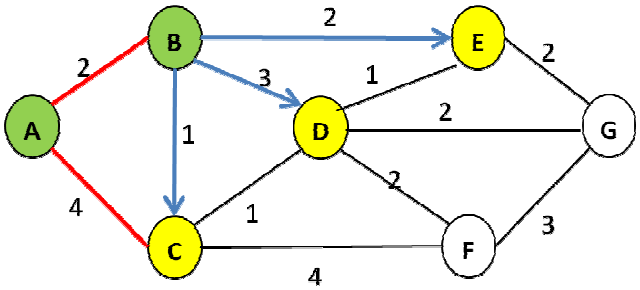


Gambar 2.9 Proses Iterasi ke-1

Tabel 2.4 Hasil proses iterasi ke-1

1	**A**		**A**						
			A	B	C	D	E	F	G
	K		A	A	A				
	T		0	2	4				

Simpul aktif pada iterasi ke-1 adalah node A, dimana node A sedang mengecek *bobot* pada node - node yang ada di sekitarnya. Disini node B dan node C berada dekat dan memiliki jalur ke node A. Hasil yang didapatkan pada iterasi ke-1 simpul akhir untuk menuju node B dan node C adalah node A. dengan total *bobot* masing - masing sebesar 2 untuk node B dan 4 untuk node C. Dan memberikan label pada jalur tersebut.

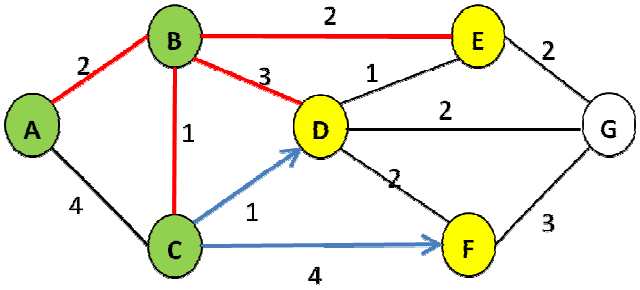


Gambar 2.10 Proses iterasi ke-2

Tabel 2.5 hasil proses iterasi ke-2

2	**A**		**A**	**B**					
			A	B	C	D	E	F	G
	K		A	A	B	B	B		
	T		0	2	3	5	4		

Simpul aktif pada iterasi ke-2 adalah node B, dimana node B sedang mengecek total bobot yang didapat dari node A sebagai node asal, menuju ke node C, node D dan node E yang terhubung ke node B. Hasil yang didapat pada iterasi ke-2 adalah menemukan jalur yang lebih pendek menuju node C, maka label dari A-C akan digantikan dengan B-C dengan total bobot adalah 3 dan menggantikan simpul akhir menuju node C dari A menjadi B.

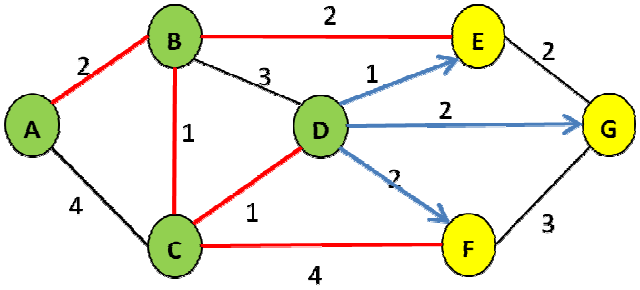


Gambar 2.11 Proses iterasi ke-3

Tabel 2.6 hasil proses iterasi ke-3

3	**A**		**A**	**B**	**C**				
			A	B	C	D	E	F	G
	K		A	A	B	C	B	C	
	T		0	2	3	4	4	7	

Simpul aktif pada iterasi ke-3 adalah node C, dimana node C sedang mengecek total bobot yang didapat dari node A sebagai node asal, menuju ke node D dan node F yang terhubung ke node C. Hasil yang didapat pada iterasi ke-3 adalah menemukan jalur yang lebih pendek menuju node D, maka label dari B-D akan digantikan dengan C-D dengan total bobot adalah 4 dan menggantikan simpul akhir menuju node D dari B menjadi C.

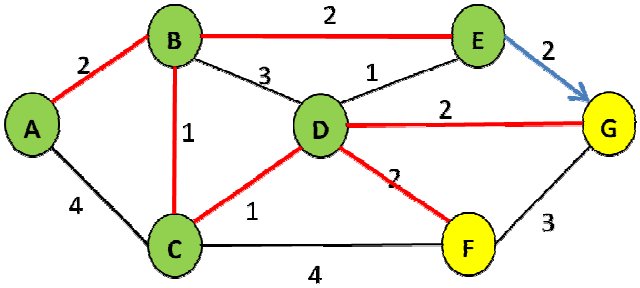


Gambar 2.12 Proses iterasi ke-4

Tabel 2.7 hasil proses iterasi ke-4

4	**A**		**A**	**B**	**C**	**D**			
			A	B	C	D	E	F	G
	K		A	A	B	C	B	D	D
	T		0	2	3	4	4	6	6

Simpul aktif pada iterasi ke-4 adalah node D, dimana node D sedang mengecek total bobot yang didapat dari node A sebagai node asal, menuju ke node E, node F dan node G yang terhubung ke node D. Hasil yang didapat pada iterasi ke-4 adalah menemukan jalur yang lebih pendek menuju node F, maka label dari C-F akan digantikan dengan D-F dengan total bobot adalah 6 dan menggantikan simpul akhir menuju node F dari C menjadi D.

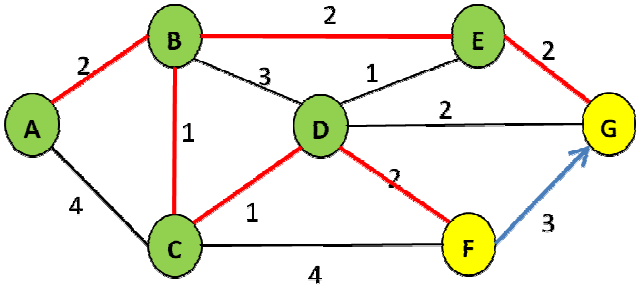


Gambar 2.13 Proses iterasi ke-5

Tabel 2.8 hasil proses iterasi ke-5

5	**A**		**A**	**B**	**C**	**D**	**E**		
			A	B	C	D	E	F	G
	K		A	A	B	C	B	D	E
	T		0	2	3	5	4	7	6

Simpul aktif pada iterasi ke-5 adalah node E, dimana node E sedang mengecek total bobot yang didapat dari node A sebagai node asal, menuju ke node node G yang terhubung ke node E. Hasil yang didapat pada iterasi ke-5 adalah menemukan jalur yang bernilai sama menuju node G. Maka proses akan menggantikan simpul akhir node D menjadi node G. Hal ini disebabkan karena proses akan memperbaharui jalur yang didapat, apabila menemukan jalur baru dengan jumlah *bobot* yang sama, maka akan menggantikannya dengan penemuan jalur yang paling baru.

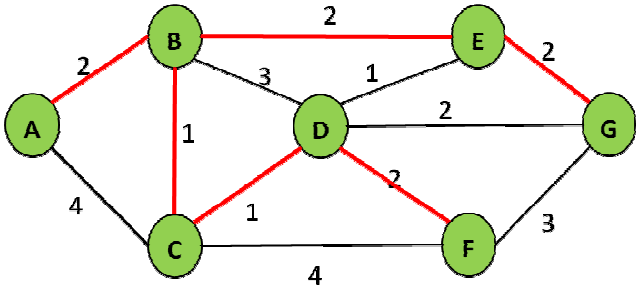


Gambar 2.14 Proses iterasi ke-6

Tabel 2.9 hasil proses iterasi ke-6

6	**A**		**A**	**B**	**C**	**D**	**E**	**F**	
			A	B	C	D	E	F	G
	K		A	A	B	C	B	D	E
	T		0	2	3	5	4	7	6

Simpul aktif pada iterasi ke-6 adalah node F, dimana node F sedang mengecek total bobot yang didapat dari node A sebagai node asal, menuju ke node node G yang terhubung ke node F. Disini proses tidak menemukan jalur yang memiliki nilai labih rendah dari sebelumnya.



Gambar 2.15 Proses iterasi ke-7

Tabel 2.10 hasil proses iterasi ke-7

7	**A**		**A**	**B**	**C**	**D**	**E**	**F**	**G**
			A	B	C	D	E	F	G
	K		A	A	B	C	B	D	E
	T		0	2	3	5	4	7	6

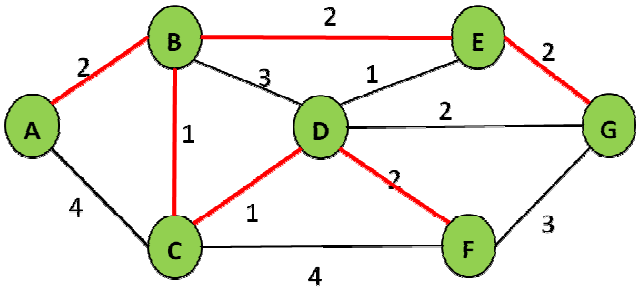
Simpul aktif pada iterasi ke-7 adalah node G, dimana node G merupakan node akhir. Disini proses tidak menemukan jalur yang memiliki nilai labih rendah dari sebelumnya. Maka proses iterasi selesai.

Dari tabel 2.10 kita dapatkan daftar label yang telah diberikan pada semua node. Jalur - jalur yang telah diberikan label di sebut dengan *path tree*.

2.5 *Path Tree*

Path tree pada teori grafh adalah subgrafh dari grafik yang dibangun sehingga jarak antara node asal (*root*) ke semua node memiliki nilai terkecil (minimal). Disebut *Tree* karena jika ada jalur yang memiliki tujuan yang sama (melingkar), maka kita dapat menghapus jalur lama yang memiliki jalur terpanjang tanpa harus menambahkan jarak dari asal ke semua node yang ada pada subgrafh.

Jika setiap pasangan node dalam grafik memiliki jalur terpendek yang unik di antara mereka, maka *path tree* bersifat unik. Hal ini karena jika jalan tertentu dari akar ke beberapa titik minimal, maka setiap bagian dari jalan itu (dari node u ke node v) adalah jalan minimal antara dua node[5].



Gambar 2.16 Graf dengan adanya *Path Tree*

2.6 DFD (*Data Flow Diagram*)

Data Flow Diagram (DFD) adalah alat pembuatan model yang memungkinkan profesional sistem untuk menggambarkan sistem sebagai suatu jaringan proses fungsional yang dihubungkan satu sama lain dengan alur data, baik secara manual maupun komputerisasi. DFD ini sering disebut juga dengan

nama *Bubble chart, Bubble* diagram, model proses, diagram alur kerja, atau model fungsi.

DFD ini adalah salah satu alat pembuatan model yang sering digunakan, khususnya bila fungsi-fungsi sistem merupakan bagian yang lebih penting dan kompleks dari pada data yang dimanipulasi oleh sistem. Dengan kata lain, DFD adalah alat pembuatan model yang memberikan penekanan hanya pada fungsi sistem.

DFD ini merupakan alat perancangan sistem yang berorientasi pada alur data dengan konsep dekomposisi dapat digunakan untuk penggambaran analisa maupun rancangan sistem yang mudah dikomunikasikan oleh profesional sistem kepada pemakai maupun pembuat program. Adapun simbol-simbol dari DFD yang dapat dilihat pada lampiran.

2.6.1 Penggambaran DFD

Tidak ada aturan baku untuk menggambarkan DFD. Tapi dari berbagai referensi yang ada, secara garis besar langkah untuk membuat DFD adalah :

1. Identifikasi terlebih dahulu semua entitas luar yang terlibat di sistem.
2. Identifikasi semua input dan output yang terlibat dengan entitas luar.
3. Buat Diagram Konteks (*diagram context*)

   Diagram ini adalah diagram level tertinggi dari DFD yang menggambarkan hubungan sistem dengan lingkungan luarnya. Caranya :

   a. Tentukan nama sistemnya.
   b. Tentukan batasan sistemnya.
   c. Tentukan terminator apa saja yang ada dalam sistem.
   d. Tentukan apa yang diterima/diberikan terminator dari/ke sistem.
   e. Gambarkan diagram konteks.
4. Buat Diagram *Level Zero*

   Diagram ini adalah dekomposisi dari diagram konteks. Caranya :

   a. Tentukan proses utama yang ada pada sistem.
   b. Tentukan apa yang diberikan/diterima masing-masing proses ke/dari sistem sambil memperhatikan konsep keseimbangan (alur data yang

keluar/masuk dari suatu level harus sama dengan alur data yang masuk/keluar pada level berikutnya).

c. Apabila diperlukan, munculkan data store (master) sebagai sumber maupun tujuan alur data.
d. Gambarkan diagram *level zero*.
   d.1 Hindari perpotongan arus data.
   d.2 Beri nomor pada proses utama (nomor tidak menunjukkan urutan proses).

5. Buat Diagram Level Satu

Diagram ini merupakan dekomposisi dari diagram level zero. Caranya :

a. Tentukan proses yang lebih kecil (sub-proses) dari proses utama yang ada di level zero.
b. Tentukan apa yang diberikan/diterima masing-masing sub-proses ke/dari sistem dan perhatikan konsep keseimbangan.
c. Apabila diperlukan, munculkan data store (transaksi) sebagai sumber maupun tujuan alur data.
d. Gambarkan DFD level Satu
   d.1 Hindari perpotongan arus data.
   d.2 Beri nomor pada masing-masing sub-proses yang menunjukkan dekomposisi dari proses sebelumnya.

6. DFD Level Dua, Tiga, …

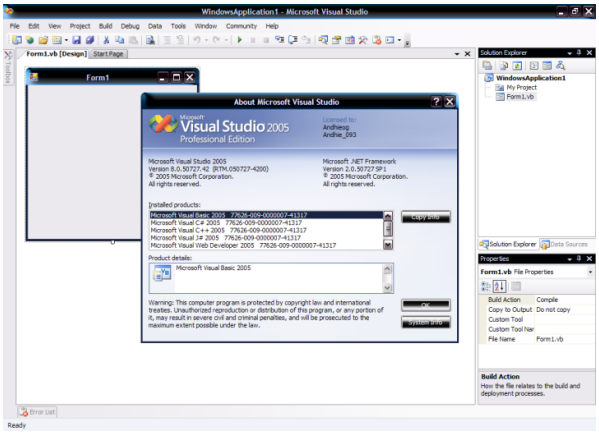
Diagram ini merupakan dekomposisi dari level sebelumnya. Proses dekomposisi dilakukan sampai dengan proses siap dituangkan ke dalam program. Aturan yang digunakan sama dengan level satu.

2.7 Visual Basic 2005

Perancangan program ini yaitu menggunakan perangkat lunak *Visual basic 2005* yang merupakan suatu bahasa pemrograman yang memberikan berbagai macam fasilitas pembuatan aplikasi visual. Keunggulan pada bahasa pemrograman ini terletak pada penggunaannya yang mudah dipahami karena didukung oleh banyak *toolbar* yang terfokus pada objek pembuatan visual serta

bahasa pemrograman yang terstruktur secara umum seperti bahasa pemrograman lainnya.

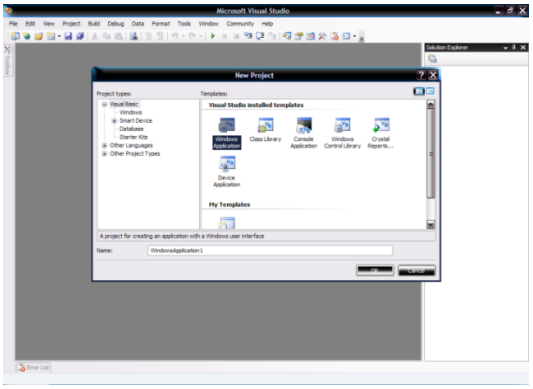
Keunggulan lain yang didapat dari bahasa pemrograman ini adalah banyaknya referensi dan contoh materi ataupun program yang telah banyak dipaparkan serta hasil rancangan aplikasi dapat langsung di eksekusi dan menjadi bagian dari program lain yang berbasis pada sistem operasi windows.



Gambar 2.17 Visual Basic 2005

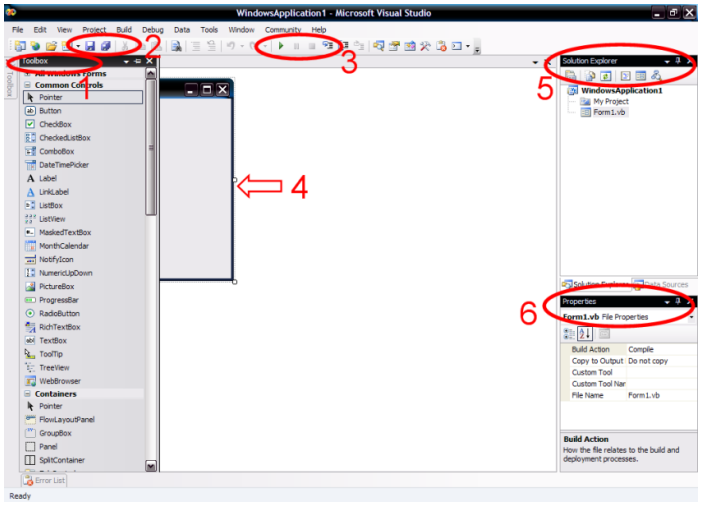
2.7.1 Memulai Visual basic 2005

Ketika kita hendak membuat file baru, maka akan muncul beberapa pilihan diantaranya adalah : windows application, class library, console application, crystal report, device application dan masih banyak lagi seperti pada gambar 2.18



Gambar 2.18 Menu *creating new file* pada Visual Basic 2005

Apabila pengguna hendak membuat program yang dapat dijalankan di system operasi windows, maka pilihlah *windows application*. Dan akan muncul form sebagai berikut



Gambar 2.19 Jendela form baru

Penggunaan Visual basic 2005 berbasis pada GUI, yang memudahkan pengguna dalam hal tampilan program. Berikut adalah penjelasan dari interface pada visual basic 2005.

1. Toolbox : berisikan tool - tool yang dapat digunakan pada program, pengguna hanya harus *click and drag* item yang akan digukanan ke dalan form pada nomor 4.
2. Tombol save file yang telah diperbaharui.
3. Running button : untuk menjalankan program yang telah dibuat.
4. Form : pada jendela ini pengguna akan men-desain tampilan program yang akan dibuat, dan untuk mengisikan perintah program hanya harus menclick kanan pada item atau form dan pilih *View Code*.
5. Solution explorer : berisikan *explorer* dari *form* atau *class* yang telah dibuat
6. Properties : berisikan informasi pada form yang aktif.