BAB II
TINJAUAN PUSTAKA

Pada bab ini dibahas mengenai dasar teori yang digunakan untuk menunjang penulisan penelitian mengenai aplikasi sistem pendukung keputusan pemilihan alat transportasi umum Kota Malang berdasar jalur terpendek dengan menggunakan algoritma *Floyd-Warshall*. Beberapa dasar teori yang dimaksud diantaranya adalah Sistem Pendukung Keputusan (*Decision Support System*), Sistem Transportasi Umum Kota Malang, Graf, dan Pencarian Jarak Terpendek.

2.1 Sistem Pendukung Keputusan (*Decision Support System*)

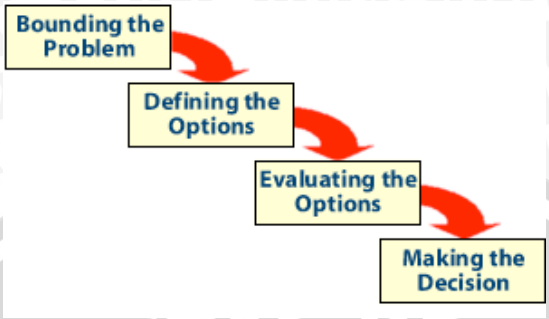
2.1.1 Pengertian Sistem Pendukung Keputusan

Pengambilan keputusan adalah proses yang kompleks dan dipengaruhi oleh banyak faktor baik manusia dan non-manusia. Proses pengambilan keputusan menggunakan empat elemen *roadmap* sebagai panduan untuk proses pengambilan keputusan. Setiap kelompok dapat mengidentifikasi apa yang harus dilakukan nantinya dan juga dapat memperkirakan dimana terjadinya. *Roadmap* tidak sama seperti satu set petunjuk yang harus diikuti, seperti *checklist*. *Roadmap* adalah satu set peluang dalam aliran logis, yang mengarah ke pengembangan dari sebuah keputusan yang kuat [POU-06].

Decision Support System atau Sistem Pendukung Keputusan (SPK), secara umum didefinisikan sebagai sebuah sistem yang mampu memberikan kemampuan baik kemampuan pemecahan masalah maupun kemampuan pemgkomunikasian untuk masalah semi-terstruktur. Secara khusus, SPK didefinisikan sebagai sebuah sistem yang mendukung kerja seorang manajer maupun sekelompok manajer dalam memecahkan masalah semi-terstruktur dengan cara memberikan informasi ataupun usulan menuju pada keputusan tertentu [AZW-10].

SPK merupakan area pembuatan aplikasi sistem informasi, yang membantu para pembuat keputusan untuk menarik suatu keputusan yang efisien di suatu waktu. SPK menyediakan bantuan yang mudah dimengerti bagi para pembuat keputusan non teknis untuk dapat menemukan metode terbaik dengan cepat. SPK adalah perangkat lunak yang menetapkan hubungan yang diperlukan antara kondisi saat ini dan kebutuhan manajemen yang diperlukan [POU-06].

Tahapan dalam pengambilan keputusan yaitu pembatasan masalah, definisi alternatif keputusan, evaluasi alternatif keputusan, membuat keputusan [POU-06].



Gambar 2.1 Empat Elemen *Roadmap* untuk Mencapai Pengambilan Keputusan yang Baik
Sumber: [POU-06]

2.1.2 Konsep Dasar SPK

Pada awalnya Turban & Aronson (1998), mendefinisikan SPK sebagai sistem yang digunakan untuk mendukung dan membantu pihak manajemen melakukan pengambilan keputusan pada kondisi semi terstruktur dan tidak terstruktur. Pada dasarnya konsep SPK hanyalah sebatas pada kegiatan membantu para manajer melakukan penilaian serta menggantikan posisi dan peran manajer [AZW-10].

Konsep SPK pertama kali diperkenalkan pada awal tahun 1970-an oleh Michael Scott Morton, yang selanjutnya dikenal dengan istilah "*Management Decision System"*. Konsep SPK merupakan sebuah sistem interaktif berbasis komputer yang membantu pembuatan keputusan memanfaatkan data dan model untuk menyelesaikan masalah-masalah yang bersifat tidak terstruktur dan semi terstruktur. SPK dirancang untuk menunjang seluruh tahapan pembuatan keputusan, yang dimulai dari tahapan mengidentifikasi masalah, memilih data yang relevan, menentukan pendekatan yang digunakan dalam proses pembuatan keputusan sampai pada kegiatan mengevaluasi pemilihan alternatif [AZW-10].

2.1.3 Komponen SPK

Ada 3 komponen SPK yaitu : [SAU-10]

1. *Data Component* (Manajemen Data)

   Database Management System (DBMS) menyediakan akses ke data serta semua program kontrol yang diperlukan untuk mendapatkan data tersebut

dalam bentuk yang sesuai untuk analisis dalam pertimbangan. Data mencakup fakta tentang operasi internal, tren, kecerdasan dan atau riset pasar, dan informasi yang tersedia secara umum. DBMS harus cukup canggih untuk memberikan akses pengguna ke data bahkan ketika mereka tidak tahu dimana data berada secara fisik. DBMS memfasilitasi penggabungan data dari sumber yang berbeda. DBMS juga harus cukup canggih untuk menggabungkan data tanpa instruksi jelas dari pengguna mengenai bagaimana seseorang menyelesaikan tugas itu.

2. *Model Management* (Manajemen Model)

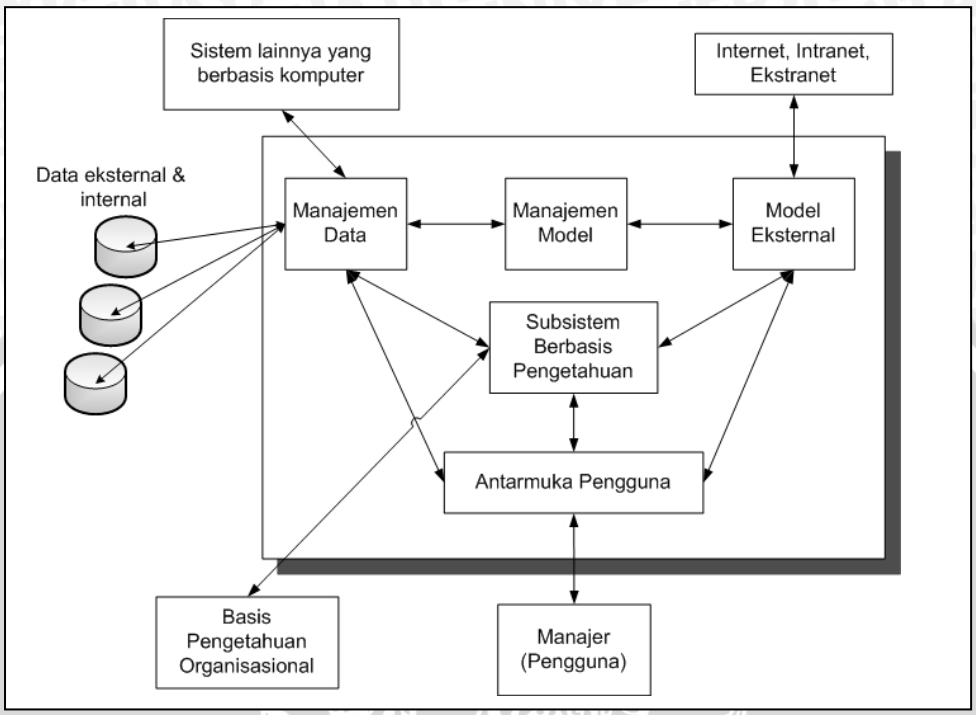
   Model Base Management System (MBMS) melacak semua model dalam SPK yang mungkin dijalankan selama analisis serta kontrol untuk menjalankan model. MBMS mungkin terdiri dari sintak yang diperlukan untuk menjalankan pekerjaan, format di mana data harus dimasukkan sebelum menjalankan model (dan untuk menempatkan data seperti dalam format), dan format data setelah menjalankan pekerjaan . MBMS juga menghubungkan antara model sehingga *output* dari satu model dapat menjadi *input* ke dalam model lain. MBMS menyediakan mekanisme untuk analisis sensitivitas dari model setelah dijalankan. MBMS memberikan bantuan konteks-sensitif dan model yang sensitif untuk membantu asumsi model pertanyaan pengguna untuk menentukan apakah mereka sudah sesuai untuk keputusan dalam pertimbangan.

3. *User Interface* (Antarmuka Pengguna)

   User interface atau antarmuka pengguna merupakan semua mekanisme dimana informasi adalah masukan ke sistem dan keluaran dari sistem. *User interface* terdiri dari semua layar *input* oleh pengguna yang meminta data dan model. *User interface* juga terdiri dari semua layar *output* dimana pengguna mendapatkan hasil. Banyak pengguna berpikir antarmuka pengguna sebagai SPK yang nyata karena itu adalah bagian dari sistem yang mereka lihat.

Ada satu lagi komponen SPK yaitu manajemen berbasis pengetahuan yang sifatnya opsional, tetapi bisa memberikan banyak manfaat karena memberikan intelegensi bagi ketiga komponen utama tersebut. Komponen manajemen berbasis

pengetahuan juga bisa diinterkoneksikan dengan repositori pengetahuan perusahaan (bagian dari sistem manajemen pengetahuan), yang kadang-kadang disebut basis pengetahuan organisasional. [KUS-07]



Gambar 2.2 Arsitektur Pemodelan Sistem Pendukung Keputusan
Sumber: [KUS-07]

2.1.4 Model Analisa SPK

Model analisa SPK terdiri dari [BEH-09]:

1. *What-if analysis*

   Analisa yang digunakan untuk menentukan apa yang terjadi pada *output*, jika nilai variabel diubah

2. *Sensitivity analysis*

   Merupakan jenis analisa yang lebih baik daripada *what-if analysis*. Analisa ini dapat membantu dalam menentukan kepekaan perubahan suatu variabel.

3. *Goal seeking analysis*

   Analisa ini membantu dalam melakukan perhitungan mundur, yang berarti membantu dalam menentukan nilai *input* ketika *output* sudah diketahui.

4. *Optimization analysis*

   Analisa untuk membantu dalam optimasi sumber daya menggunakan model statistik.

2.2 Sistem Transportasi Umum Kota Malang

Transportasì telah dipahami secara luas sebagai salah satu faktor penentu dalam pertumbuhan dan perkembangan suatu wìlayah. Ia merupakan tulang punggung bagi aktivítas ekonomi dan aktivítas sosial. Dengan transportasi terjadi aktivitas pemindahan sumberdaya-sumberdaya ekonomi, sehingga muncul harga pasar dari basil pertukaran di antara sumberdaya-sumberdaya ekonomi. Dengan transportasì pula terjadi ínteraksi sosial-budaya di antara individu/masyarakat satu dengan individu atau masyarakat laìnnya.

Di Kota Malang terdapat banyak transportasi umum, antara lain taksi, ojek, dan angkutan kota dengan berbagai tujuan. Pilihan yang paling banyak digunakan adalah moda transportasi angkutan kota atau angkot. Angkot memiliki tarif yang sama, jarak dekat maupun terjauh. Sebenarnya tarif angkutan kota Malang adalah Rp 2.300,00 untuk umum dan Rp 1.500,00 untuk pelajar berseragam atau mahasiswa [PER-09]. Dalam kenyataannya para sopir angkutan kota memasang tarif yang sama, yaitu Rp 2.500,00. Hal tersebut ditujukan agar para penumpang mudah membayar dengan nominal yang bulat dan sopir juga tidak kesulitan untuk mencari uang kembalian. Ada banyak kode angkot di Malang, berikut datanya:

Tabel 2.1 Jumlah Armada dan Panjang Trayek Angkutan Kota Tahun 2009

No	Nama Trayek	Rute Trayek	Jarak Tempuh (Km)	Jumlah (unit)
1	AL	Arjosari – Landungsari	17,2	106
2	AH	Arjosari – Hamid Rusdi	16	300
3	AT	Arjosari – Tidar	12.7	53
4	ASD	Arjosari – Sarangan – Dieng	18	45
5	CKL	Cemorokandang – Landungsari	22	89
6	MK	Madyopuri – Karang Besuki	11,3	62
7	ABB	Arjosari – Borobudur – Bunulrejo	16	65
8	AJH	Arjosari – Janti – Hamid Rusdi	18,6	81
9	ABH	Arjosari – Borobudur – Hamid Rusdi	17.7	84

10	TSG	Tawangmangu – Soekarno Hatta – Gasek	10	27
11	JPK	Joyogrand – Piranha – Karangploso	10	60
12	AMH	Arjosari – Mergosono – Hamid Rusdi	15,5	217
13	HML	Hamid Rusdi – Mergosono – Landungsari	19,8	45
14	HT	Hamid Rusdi – Tirtosari	8	6
15	HA	Hamid Rusdi – Arjosari	16,6	160
16	HL	Hamid Rusdi – Landungsari	17,6	108
17	HST	Hamid Rusdi – Sarangan – Tasikmadu	28	106
18	MH	Mulyorejo – Hamid Rusdi	10,7	17
19	ADL	Arjosari – Dinoyo – Landungsari	14,5	125
20	LDH	Landungsari – Dinoyo – Hamid Rusdi	17,2	170
21	LH	Landungsari – Hamid Rusdi	18,5	118
22	HM	Hamid Rusdi – Madyopuro	10	62
23	MKS	Madyopuro – Klayatan – Sukun	6	11
24	MM	Madyopuro – Mulyorejo	15,2	68
25	JDM	Joyogrand – Dinoyo – Mergan	10	51
Jumlah				**2.236**

Sumber: [DIN-09]

Moda transportasi umum kedua yang sering digunakan adalah ojek. Ojek kebanyakan digunakan ketika tidak ada trayek angkot yang sesuai dengan tujuan pengguna. Ada dua jenis ojek di Kota Malang, yaitu ojek yang ada di pangkalan tempat-tempat umum dan ojek taksi yang menerapkan sistem argo maupun tawar menawar. Harga ojek lebih murah daripada harga taksi. Hal ini juga tergantung pada keahlian pengguna dalam menawar harga.

Moda transportasi yang menjadi pilihan terakhir adalah taksi karena memiliki tarifyang relatif mahal. Perusahaan taksi yang beroperasi di Malang antara lain Taksi Citra Kendedes, Taksi Argo, Taksi Mandala, dan Taksi Bima.

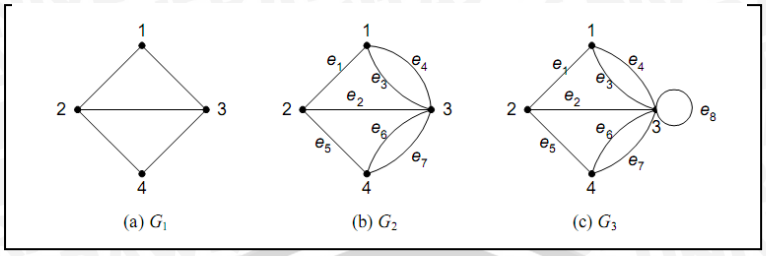
Tabel 2.2 Daftar Tarif Angkutan Umum Kota Malang

No	Jenis Angkutan	Keterangan	Tarif (Rp)
1	Angkutan Kota	Sekali jalan	2.500
2	Ojek	Tarif 500 meter pertama (minimal)	4.000
		Tarif per 500 meter	1.000
3	Taksi [SKW-05]	Tarif pertama/buka pintu (flag fall)	5.000
		Tarif per kilometer	3.000
		Tarif tunggu (per jam)	30.000
		Tarif total minimal	20.000

2.3 Graf

Teori graf merupakan pokok bahasan yang sudah tua usianya namun memiliki banyak terapan sampai saat ini [MUN-01]. Graf dapat didefinisikan sebagai kumpulan simpul-simpul yang dihubungkan dengan garis. Simpul biasa dinyatakan dengan istilah verteks dan garis biasa dinyatakan dengan istilah edges atau busur. Graf G (V, E), adalah koleksi atau pasangan dua himpunan, Himpunan V yang elemennya disebut simpul atau titik, atau vertex, atau point, atau node. Himpunan E yang merupakan pasangan tak terurut dari simpul, disebut ruas atau rusuk, atau sisi, atau edge, atau line. Banyaknya simpul (anggota V) disebut order Graf G, sedangkan banyaknya ruas (anggota E) disebut ukuran (size) Graf G.

Graf dapat digolongkan berdasarkan dari beberapa hal. Secara umum graf dapat digolongkan menjadi dua jenis, yaitu graf sederhana dan graf tak-sederhana. Berdasarkan jumlah simpul pada suatu graf, maka secara umum graf dapat digolongkan menjadi dua jenis, yaitu graf berhingga, dan graf tak-berhingga. Berdasarkan orientasi arah pada sisi, maka secara umum graf dibedakan atas dua jenis, yaitu graf berarah dan graf tak-berarah.



Gambar 2.3 (G1) Graf Sederhana, (G2) Graf Ganda, (G3) Graf Semu
Sumber: [MUN-01]

2.4 Pencarian Jalur Terpendek

Pencarian jalur terpendek merupakan suatu permasalahan yang sering timbul pada pengguna transportasi, karena pengguna transportasi dalam melakukan perjalanan membutuhkan solusi bagaimana jalur yang akan dilalui adalah jalur atau jarak yang paling minimum sehingga efisiensi waktu dapat terpenuhi.

Dalam melakukan pemilihan terhadap jalur terpendek, dapat dilakukan dengan metode algoritma. Algoritma merupakan kumpulan instruksi atau perintah yang dibuat secara jelas dan sistematis berdasarkan urutan yang logis (logika) untuk penyelesaian suatu masalah. Sedangkan algoritma pencarian jalur terpendek adalah algoritma yang menentukan bagaimana memilih jalur optimal antara asal dan tujuan dengan memperhitungkan waktu kalkulasi terpendek.

Ada beberapa contoh algoritma pencarian jarak terpendek, antara lain algoritma Dijkstra dan algoritma *Floyd-Warshall*. Algoritma Dijkstra merupakan algoritma yang paling sering digunakan dalam menentukan jalur terpendek, sederhana (sifat *greedy* atau rakus dalam pemilihan graf) dalam penggunaannya dengan hanya menggunakan verteks-verteks sederhana pada jaringan jalan yang tidak rumit [CHA-06]. Pada beberapa kasus algoritma Dijkstra dengan sifat *greedy*, yaitu tidak memikirkan konsekuensi yang akan terjadi pada saat memilih keputusan maka algoritma ini tidak memberikan solusi yang terbaik. Algoritma *Floyd-Warshall* adalah salah satu varian dari pemrograman dinamis, yaitu suatu metode yang melakukan pemecahan masalah dengan memandang solusi yang akan diperoleh sebagai suatu keputusan yang saling terkait [NOV-07].

2.4.1 Karakteristik Pemrograman Dinamis

Beberapa karakteristik yang dimiliki oleh program dinamis antara lain:

1. Persoalan dibagi atas beberapa tahap, yang setiap tahapnya hanya akan diambil satu keputusan.
2. Masing-masing tahap terdiri atas sejumlah status yang saling berhubungan dengan status tersebut. Status yang dimaksud di sini adalah berbagai kemungkinan masukan yang ada pada tahap tersebut.
3. Ketika masuk ke suatu tahap, hasil keputusan akan transformasi.
4. Jarak ataupun waktu tempuh pada suatu tahap akan meningkat secara teratur seiring bertambahnya jumlah tahapan.
5. Jarak ataupun waktu tempuh yang ada pada suatu tahap tergantung dari jarak ataupun waktu tempuh tahapan yang telah berjalan pada tahap itu sendiri.
6. Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan pada tahap sebelumnya.
7. Terdapat hubungan rekursif yang menyatakan bahwa akan memberikan keputusan terbaik untuk setiap status.
8. Prinsip optimalitas berlaku pada persoalan yang dimaksud.

Dalam proses penyelesaian menggunakan program dinamis, pendekatan yang dilakukan bisa jadi ada dua macam, yaitu pendekatan maju (forward) dan mundur (backward), dan perlu untuk diketahui pula bahwa solusi yang dihasilkan dari kedua pendekatan itu adalah tidak sama. Solusi dari program dinamis bisa jadi lebih dari satu macam.

2.4.2 Algoritma *Floyd-Warshall*

Algoritma *Floyd-Warshall* adalah salah satu varian dari pemrograman dinamis, yaitu suatu metode yang melakukan pemecahan masalah dengan memandang solusi yang akan diperoleh sebagai suatu keputusan yang saling terkait [NOV-07]. Artinya solusi-solusi tersebut dibentuk dari solusi yang berasal dari tahap sebelumnya dan ada kemungkinan solusi lebih dari satu.

Algoritma ini menghitung bobot terkecil dari semua jalur yang menghubungkan sebuah pasangan titik, dan juga sekaligus untuk semua pasangan titik. Implementasi algoritma ini berupa graf yang direpresentasikan sebagai

matriks keterhubungan, yang isinya ialah bobot atau jarak sisi yang menghubungkan tiap pasangan titik, dilambangkan dengan indeks baris dan kolom. Ketiadaan sisi yang menghubungkan sebuah pasangan dilambangkan dengan tak-hingga.

Prinsip dari algoritma ini adalah "jika solusi total optimal, maka bagian solusi sampai suatu tahap (misalnya tahap ke-i) juga optimal", yang mempunyai pengertian bahwa selain diperolehnya suatu jalur terpendek dari simpul awal ke simpul akhir, juga akan diperoleh nilai-nilai jalur antar simpul.

Algoritma *Floyd-Warshall* membandingkan semua kemungkinan lintasan pada graf untuk setiap sisi dari semua simpul. Menariknya, algoritma ini mampu mengerjakan proses perbandingkan ini sebanyak V3 kali (bandingkan dengan kemungkinan jumlah sisi sebanyak V2 (kuadrat jumlah simpul) pada graf, dan setiap kombinasi sisi diujikan). Hal tersebut bisa terjadi karena adanya perkiraan pengambilkan keputusan (pemilihan jalur terpendek) pada setiap tahap antara dua simpul, hingga perkiraan tersebut diketahui sebagai nilai optimal,

Misalkan terdapat suatu graf G dengan simpul-simpul V yang masing-masing bernomor 1 s.d. N (sebanyak N buah). Misalkan pula terdapat suatu fungsi shortestPath(i, j, k) yang mengembalikan kemungkinan jalur terpendek dari i ke j dengan hanya memanfaatkan simpul 1 s.d. k sebagai titik perantara.

Tujuan akhir penggunaan fungsi ini adalah untuk mencari jalur terpendek dari setiap simpul i ke simpul j dengan perantara simpul 1 s.d. k+1. Ada dua kemungkinan yang terjadi:

1. Jalur terpendek yang sebenarnya hanya berasal dari simpul-simpul yang berada antara 1 hingga k.
2. Ada sebagian jalur yang berasal dari simpul-simpul i s.d. k+1, dan juga dari k+1 hingga j.

Perlu diketahui bahwa jalur terpendek dari i ke j yang hanya melewati simpul 1 s.d. k telah didefinisikan pada fungsi shortestPath(i, j, k) dan telah jelas bahwa jika ada solusi dari i s.d. k+1 hingga j, maka panjang dari solusi tadi adalah jumlah (konkatenasi) dari jalur terpendek dari I s.d. k+1 (yang melewati simpul-

simpul 1 s.d. k), dan jalur terpendek dari k+1 s.d. j (juga menggunakan simpul-simpul dari 1 s.d. k).

Maka dari itu, rumus untuk fungsi shortestPath(i,j, k) bisa ditulis sebagai suatu notasi rekursif sebagai berikut:

```
Basis-0
shortestPath(i, j, 0) =
edgeCost(i,j);
    Rekurens
     shortestPath(i, j, k) =
min(shortestPath(i, j, k-1),
    shortestPath(i, k, k-1) +
    shortestPath(k, j, k-1));
```

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0 \\ \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}) & \text{if } k \geq 1 \end{cases} \quad (2\text{-}1)$$

Rumus tersebut adalah inti dari algoritma *Floyd-Warshall*. Algoritma ini bekerja dengan menghitung shortestPath(i,j,1) untuk semua pasangan (i,j), kemudian hasil tersebut akan digunakan untuk menghitung shortestPath(i,j,2) untuk semua pasangan (i,j), dan seterusnya. Proses ini akan terus berlangsung hingga k = n dan kita telah menemukan jalur terpendek untuk semua pasangan (i,j) menggunakan simpul-simpul perantara [THO-03]. Berikut merupakan *pseudocode* dari algoritma *Floyd-Warshall*:

```
function fw(int[1..n,1..n] graph) {
    // Inisialisasi
    var int[1..n,1..n] jarak := graph
    var int[1..n,1..n] sebelum
    for i from 1 to n
        for j from 1 to n
            if jarak[i,j] < Tak-hingga
                sebelum[i,j] := i
    // Perulangan utama pada algoritma
    for k from 1 to n
```

```
        for i from 1 to n
            for j from 1 to n
                if jarak[i,j] > jarak[i,k] + jarak[k,j]
                    jarak[i,j] = jarak[i,k] + jarak[k,j]
                    sebelum[i,j] = sebelum[k,j]
    return jarak }
```

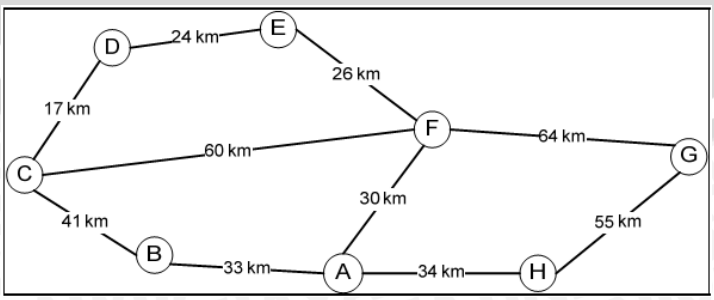
2.4.3 Perbandingan Pemrograman Dinamis dengan *Greedy*

Algoritma Dijkstra merupakan salah satu varian dari algoritma *greedy*, yaitu salah satu bentuk algoritma populer dalam pemecahan persoalan yang terkait dengan masalah optimasi. Sifatnya sederhana dan lempang (*straightforward*). Sesuai dengan artinya 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.

Penggunaan strategi *greedy* pada algoritma Dijkstra adalah pada setiap langkah, ambil sisi berbobot minimum yang menghubungkan sebuah simpul yang sudah terpilih dengan sebuah simpul lain yang belum terpilih. Lintasan dari simpul asal ke simpul yang baru haruslah merupakan lintasan yang terpendek di antara semua lintasannya ke simpul-simpul yang belum terpilih.

Perbandingan antara pemrograman dinamis, yaitu algoritma *Floyd-Warshall* dengan *greedy*, yaitu algoritma *dijkstra* dapat dilihat dari studi kasus berikut:

Misalkan terdapat suatu graf berbobot yang merepresentasikan kondisi keterhubungan antarkota di suatu daerah, dengan ilustrasi sebagai berikut.



Gambar 2.4 Representasi Keterhubungan Antarkota dalam Graf Berbobot

Sumber: [NOV-07]

Misalkan seseorang akan melakukan perjalanan dari kota A ke kota C. Orang tersebut mencoba untuk menerapkan algoritma *Dijkstra* dan algoritma *Floyd-Warshall* untuk mencari jalur terpendek dari kota A ke kota C.

Berikut ini adalah penelusuran jalur apabila orang tersebut menggunakan algoritma Dijkstra (prinsip *greedy*):

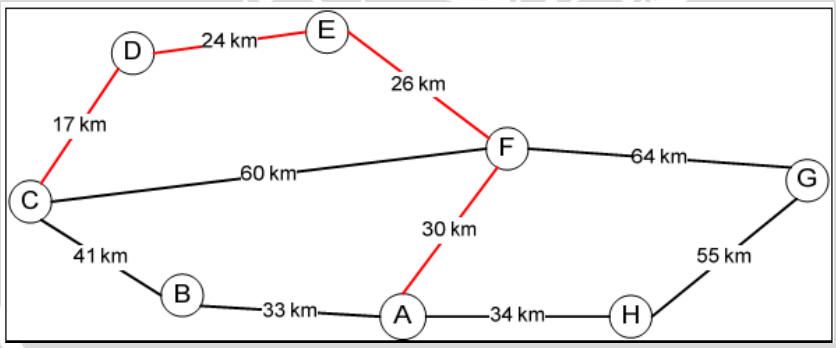
Tahap 1: Dari kota A, orang tersebut akan memilih kota F dengan bobot minimum dari kota A (30 km).

Tahap 2: Dari kota F, orang tersebut kemudian memilih kota E yang memiliki bobot minimum dari kota F (26 km).

Tahap 3: Dari kota E, orang tersebut akan melanjutkan perjalanan ke kota D (satu-satunya simpul yang terhubung).

Tahap 4: Dari kota D, orang tersebut lalu melanjutkan perjalanan dan sampai ke kota C.

Total jarak yang ditempuh oleh orang tersebut adalah = 97 km dengan jalur (A – F – E – D – C). Dalam representasi graf, warna merah pada sisi graf menunjuk ke jalur terpendek menurut algoritma *Dijkstra*.



Gambar 2.5 Representasi Keterhubungan Antarkota Menggunakan Algoritma *Dijkstra*
Sumber: [NOV-07]

Sekarang, orang tersebut mencoba menerapkan algoritma *Floyd-Warshall* dengan pendekatan pemrograman dinamis maju (forward).

Basis:

$$f_1(s) = cx_1 s$$

Rekurens:

$$f_k(s) = \min x_k \{cx_k s + f_{k-1}(x_k), k = 2,3,4, \ldots$$

Tahap 1:

$$f_1(s) = cx_1 s$$

s	Solusi optimum	
	f_1 (s)	x_1
B	33	A
F	30	A
H	34	A

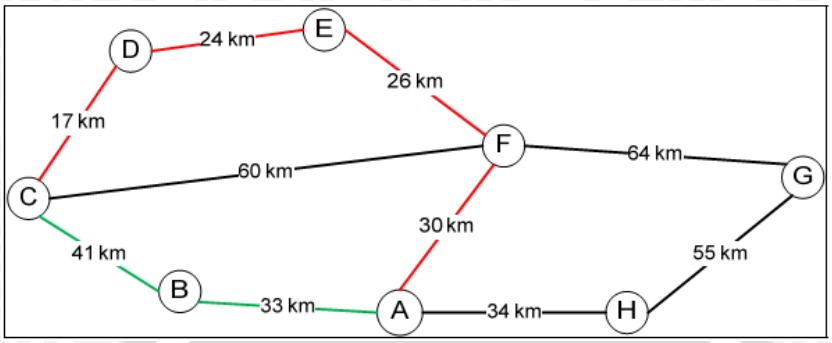
Tahap 2:

$$f_2(s) = \min x_2 \{cx_2 s + f_1(x_2)\}$$

x_2 / s	$f_2(x_2, s) = cx_2 s + f_1(x_2)$			Solusi Optimum	
	B	F	H	$f_2(s)$	x_2
C	**74**	90		**74**	B
E		56		56	F
G		94	89	89	H

Dari hasil pencarian jalur terpendek dari A ke C menggunakan algoritma *Floyd-Warshall* (pemrograman dinamis), ditemukan bahwa jarak terpendek dari A ke C adalah 74 km dengan jalur (A – B – C).

Berikut ini representasi graf setelah menggunakan kedua algoritma tadi. Dalam representasi graf, warna merah pada sisi graf menunjuk ke jalur terpendek menurut algoritma Dijkstra, sementara warna hijau menurut algoritma *Floyd-Warshall*.



Gambar 2.6 Representasi Keterhubungan Antarkota Menggunakan Algoritma *Dijkstra* (sisi berwarna merah) dan Algoritma *Floyd-Warshall* (sisi berwarna hijau)
Sumber: [NOV-07]

Terdapat perbedaan yang cukup signifikan untuk perbedaan penerapan kedua algoritma tadi, dengan selisih jarak 23 km. Ini berarti algoritma Dijkstra gagal memberi solusi optimum untuk kasus di atas.

2.4.4 Keunggulan Algoritma *Floyd-Warshall*

Berikut merupakan keunggulan algoritma *Floyd-Warshall*:

1. Algoritma Floyd-warshall dapat menangani masalah lintasan terpendek dengan kasus graf berbobot negatif yang tidak dapat diselesaikan oleh algoritma greedy.
2. Algoritma Dijkstra yang menerapkan prinsip *greedy* tidak selalu berhasil memberikan solusi optimum untuk kasus penentuan lintasan terpendek (*single pair shortest path*).
3. Algoritma Floyd-warshall digunakan untuk menyelesaikan masalah *All-pairs Shortest Path*, sedangkan algoritma greedy digunakan hanya untuk menyelesaikan masalah *Single-source Shortest Path*.