PROGRAM APLIKASI *TRAVELLING SALESMAN* PADA PENDISTRIBUSIAN AGEN MINUMAN RINGAN MENGGUNAKAN TURBO PASCAL 7.1

Yushi Nurdini
Program Studi Teknik Informatika – Fakultas Teknik, Matematika dan IPA
Universitas Indraprasta PGRI

Abstract. *The Economic Crisis that still continues, causing many companies should lay off their employees. In order to survive they become drinks agents with the capital that they get from their severance pay. For the baverage producers it will become an apportunities to gain a huge advantage. But if there is a long distance between the subscribers it will become a big issue that should be solved it can makes the operational cost getting bigger. This writing explains the process to get the shortest path between the subscribers. Besides it will minimize delivery time, it also minimizes the operational cost. To design an application, I read pascal books as reference and I use pascal programming language Turbo Pascal 7.1 to make a simple application.*

Key Words: employees, distribution, cost, turbo pascal.

PENDAHULUAN

Latar Belakang

Krisis ekonomi yang belum berakhir sampai saat ini menyebabkan banyak perusahaan yang dengan terpaksa harus mem-PHK-kan karyawannya. Nasib para karyawan yang terkena PHK ini tergantung dari kreativitas mereka sendiri, hal yang paling mudah untuk bertahan hidup adalah membuka warung dengan modal dari uang pesangon yang mereka terima. Apabila mereka ingin warung mereka berkembang maka mereka harus melengkapi warung mereka dengan berbagai macam barang yang dibutuhkan oleh para konsumen, salah satunya adalah minuman (soft drink). Mereka membutuhkan kerjasama atau berhubungan langsung dengan produsen minuman.

Bagi para produsen minuman hal ini memberikan suatu peluang untuk meraih keuntungan yang sangat besar. Tetapi apabila jarak warung-warung yang menjadi langganan mereka berjauhan maka ini akan menjadi salah satu masalah yang harus mereka pikirkan karena hal ini membuat biaya operasional untuk mengantarkan barang pesanan mereka bertambah besar.

Untuk itu dibutuhkan suatu solusi mengatasi hal tersebut, dalam hal ini penulis memberikan suatu jalan untuk menentukan rute yang akan ditempuh untuk menekan biaya operasional dengan menggunakan "**Travelling Salesman**".

Rumusan Masalah

Dalam penulisan ilmiah ini, penulis membatasi masalah pada pembuatan aplikasi pada masalah penentuan rute dengan cara "**Travelling Salesman**" yang meliputi:

- Jarak yang ditempuh dari suatu simpul awal hingga sampai simpul terakhir yang diasumsikan sebagai waktu yang ditempuh dengan menggunakan satuan menit sebagai jaraknya.
- Jumlah tempat minimalnya 2 maksimal 10.
- Variabel-variabel yang lainnya seperti bagaimana kendaraan jalan, tingkat kemacetan, dan lain-lain diabaikan atau dianggap tidak ada.
- Diasumsikan antara simpul-simpul yang ada saling berhubungan 2 arah

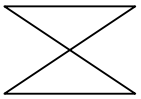
Tujuan Penulisan

- Menentukan waktu terpendek untuk mengantarkan pesanan
- Membantu perusahaan untuk memperkecil biaya operasional
- Mendapatkan jalur terpendek dengan menggunakan dua cara, yaitu dengan perhitungan cara manual dan bahasa pemrograman.

LANDASAN TEORI

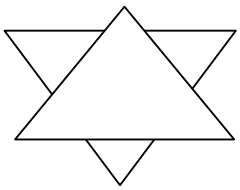
Kelahiran Teori Graph

Mungkin anda di masa kanak-kanak dahulu pernah bermain tebakan untuk menggambarkan suatu bentuk geometri tertentu tanpa mangangkat alat tulis dari kertas, dan tidak diperkenankan untuk mengulang garis yang telah tergambar. Beberapa bagian dapat dibangun sesuai selera dengan syarat diatas seperti pada gambar di bawah ini:



Gambar 1. Graph yang memenuhi syarat

Tetapi bangun geometri pada gambar di bawah tidak dapat dibuat sesuai dengan persyaratan yang sudah ditetapkan



Gambar 2. Graph yang tidak memenuhi syarat

Masalah diatas merupakan jenis masalah yang menandai lahirnya Teori Graph. Suatu cabang sains yang pada waktu ini berkembang dengan pesat.

Graph secara formal

Kata Graph didalam matematika mempunyai bermacam-macam arti, kita mengenal graph (graphic) suatu fungsi dan relasi, pada pembahasan ini kita menggunakan graph dalam pengertian lain.

Suatu graph G lengkapnya kita tulis G(V,E), adalah suatu koleksi atau pasangan dua himpunan:

Himpunan E yang merupakan pasangan tak terurut dari simpul disebut *rusuk* atau *sisi* atau *edge*. Suatu ruas r_1 dan r_2 yang mempunyai kedua simpul ujungnya sama, yakni $r_1 = (u,u)$ disebut *self loop* (gelung). Graph secara umum disebut *multigraph* yaitu graph yang mempunyai gelung dan ruas sejajar, sedangkan graph sederhana adalah graph yang tidak mempunyai gelung maupun ruas sejajar.

Keterhubungan

Perjalanan atau walk pada suatu graph G adalah barisan simpul dan ruas secara berganti-ganti: $v1,e1,v2,e2,\ldots\ldots,e_{n-1},v_n$. Disini ruas e1 menghubungkan simpul v_i dan v_{i+1}.

- Lintasan atau *trail* adalah *walk* dengan semua ruas dalam barisan adalah berbeda
- Jalur atau *path* adalah perjalanan yang semua simpul dalam barisan adalah berbeda
- Sirkuit atau *cycle* adalah suatu lintasan tertutup yang derajat setiap simpul dua

Suatu graph disebut terhubung apabila untuk setiap dua simpul dari graph G selalu terdapat jalur yang menghubungkan kedua simpul tersebut.

Matrik dalam Diagraf

Diagraf G = (S, E) mempunyai n simpul yang telah dinomori semua, demikian juga dengan m adalah jumlah sisi yang telah diberi nomor.

1. Matrik Adjasensi

Yang dimaksud dengan matrik adjasensi dari suatu diagraf G = (S, E) adalah matrik A (A_{ij}) dengan dimensi n x n, yang mempunyai ketentuan $A_{ij} = 1$ bila ada ruas (V_i, V_j) atau 0 dalam hal lain.

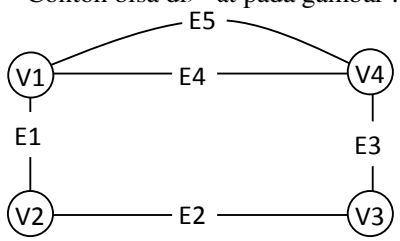
Dari definisi diagraf, yang tidak mempunyai silklus, tidak ada sisi paralel, maka akan diperoleh matrik diagraf dengan elemen-elemen diagonal dengan 0. Bila nomor-nomor simpul dari grafik

diubah, maka baris dan kolom dari matrik adjasensi akan saling bertukaran.

2. Matrik Insidensi

Matrik insidensi berdimensi (n x m), dimana m adalah jumlah sisi. Sisi ini diberi nomor secara leksikografis, yaitu nomor paling rendah dari sisi yang mempunyai tingkat leksikigrafis terendah.

Contoh bisa dilihat pada gambar :



Gambar 3.

Dari gambar diatas dapat dibuat matrik ajasensi dan insidensi, sebagai berikut:

Tabel 1. Matrik Ajasensi

	V1	V2	V3	V4
V1	0	1	0	1
V2	1	0	1	0
V3	0	1	0	1
V4	1	0	1	0

Tabel 2. Matrik Insidensi

	E1	E2	E3	E4	E5
V1	1	0	0	1	1
V2	1	1	0	0	0
V3	0	1	1	0	0
V4	0	0	1	1	1

Definisi *Travelling Salesman*

Travelling Salesman adalah teori yang menjelaskan tentang rute perjalanan yang ditempuh dengan cara mencari *path* atau jalur terpendek (waktu terpendek) sehingga sedemikian rupa, semua tempat yang ada dapat dikunjungi dan kembali ke tempat semula dengan solusi yang optimal. Solusi yang optimal disini adalah dengan waktu seminimal mungkin dapat mencapai hasil yang ditetapkan.

Penarikan kembali bahwa problem salesman adalah mendapatkan paling sedikit harga perjalanan dari daerah-daerah N dalam wilayahnya.

Untuk mempermudah dalam mendapatkan solusi dengan metode *travelling salesman*, kita harus membawa permasalahan ke dalam permodelan graph. Berikut ini adalah tahap-tahap metode *travelling salesman*:

- Tentukan banyaknya simpul
- Pemberian nama kepada semua simpul
- Menetapkan sebuah simpul sebagai simpul awal
- Cari perjalanan ke simpul terdekat lainnya
- Apabila terdapat jarak yang sama pilih salah satu
- Lanjutkan perjalanan ke simpul terdekat lainnya
- Lakukan kembali langkah tersebut diatas sampai semua simpul selesai dikunjungi
- Setelah sampai pada simpul yang terakhir kemudian kembali lagi ke simpul awal

Masalah Jalur Terpendek

Suatu masalah jalur terpendek menyangkut suatu jaringan tersambung yang memiliki suatu nilai tidak negatif yang dikaitkan dengan tiap cabang. Salah satu simpul ditunjuk sebagai sumber (source) dan simpul lainnya ditunjuk sebagai muara (sink). Istilah ini sama sekali tidak berarti bahwa cabang-cabang ini merupakan jaringan terorientasi dan hanya menyatakan arah dalam algoritma pemecahan yang akan ditetapkan. Tujuannya adalah menentukan lintasan yang menghubungkan sumber (source) dan muara (sink) sedemikian rupa sehingga bobot nilai yang berkaitan dengan tiap-tiap cabang dalam lintasan ini bobot minimum.

Kemungkinan membentuk jalur terpendek dilakukan satu demi satu. Hal tersebut dilakukan dengan memperhatikan pengukuran optimasi dari jumlah panjang jalur yang mungkin dapat dibentuk. Solusi optimalnya adalah jalur yang panjangnya minimal.

Menentukan jalur terpendek (shortest path)

- Berangkat dari titik awal secara beraturan, menentukan jarak terpendek ke simpul berikutnya, meningkat terus sampai titik terakhir. Misal untuk suatu harga n terdapat (n-1) simpul (tidak termasuk titik awal) yang merupakan simpul terdekat ke titik awal sepanjang rantai atau busur terpendek. Selanjutnya (n-1) simpul ini ditambah titik awal disebut sebagai simpul asli dan simpul lain disebut sebagai simpul baru.
- Untuk menilai suatu calon simpul asli, maka simpul tersebut harus tersambung dengan cabang ke salah satu simpul-simpul asli, simpul tersebut harus merupakan simpul baru yang paling dekat ke simpul asli.

Pemilihan simpul baru

Untuk simpul yang ada dihubungkan oleh satu cabang kesimpul baru, hitung:

- Jarak terpendek yang telah diketahui dan titik awal ke simpul itu kemudian ditambah.
- Jarak dari simpul asli ke simpul baru terdekat. Jumlah ini menyatakan jarak dari titik awal ke simpul baru, pilih yang terpendek. Ulangi sampai tujuan (titik akhir tercapai).

Algoritma Rute Terdekat

Dalam algoritma rute terpendek terdapat dua algoritma untuk menentukan rute terdekat yaitu jaringan **asiklis** dan **siklis** . Sebuah jaringan dikatakan asiklis jika tidak memiliki loop, jika memiliki loop jaringan tersebut bersifat siklis. Algoritma siklis adalah lebih umum dalam arti bahwa algoritma ini mencakup kasus asiklis. Tetapi, algoritma asiklis lebih efisien, karena melibatkan lebih sedikit perhitungan.

1. Algoritma Asiklis

Algoritma asiklis didasari oleh penggunaan perhitungan rekursif, yang merupakan dasar dari perhitungan pemrograman dinamis. Rumus rekursif mengharuskan bahwa jarak terdekat u_j ke node j dihitung hanya setelah kita menghitung jarak terdekat u_i ke setiap node sebelumnya i yang dihubungakan ke j dengan sebuah busur langsung.

2. Algoritma Siklis

Algoritma siklis tidak akan berjalan baik jika jaringan yang bersangkutan kebetulan mancakup loop yang terarah.

Algoritma siklis berbeda dengan algoritma asiklis dalam hal bahwa algoritma ini memungkinkan banyak kesempatan sebagaimana yang diperlukan untuk mengevaluasi ulang sebuah node . Ketika terlihat bahwa jarak terdekat ke sebuah node telah dicapai, node tersebut dikeluarkan dari pertimbangan lebih lanjut. Proses ini berakhir ketika node tujuan dievaluasi.

Masalah Rute Terdekat Dipandang sebagai Model Transshipment

Model transportasi standar mengasumsikan bahwa rute langsung antara sebuah sumber dan sebuah tujuan adalah rute dengan biaya minimum. Suatu prosedur alternatif dari penggunaan model transportasi biasa (dengan algoritma rute terdekat yang dimasukkan kedalamnya) adalah model transshipment.

Model yang baru ini memiliki ciri tambahan yang mengijinkan unit-unit dikirimkan dari semua sumber untuk melewati node-node antara atau sementara sebelum pada akhirnya mencapai tujuan. Akibatnya, algoritma baru ini menggabungkan baik algoritma transportasi biasa dengan algoritma rute terdekat menjadi satu prosedur. Masalah rute terdekat dapat dirumuskan sebagai sebuah model transshipment.

Kita dapat memandang jaringan rute terdekat sebagai sebuah model transportasi dengan satu sumber dan satu tujuan. Penawaran di sumber adalah satu unit dan permintaan di tujuan juga satu unit. Satu unit akan mengalir dari sumber ke tujuan melalui rute yang dapat diterima dalam jaringan tersebut.

Tujuannya adalah meminimumkan jarak yang ditempuh oleh unit tersebut sementara mengalir dari sumber ke tujuan.

Bahasa Pemrograman Turbo Pascal 7.1

Pascal adalah bahasa tingkat tinggi (*high level language*) yang orientasinya pada segala tujuan, dirancang oleh Proffesor Niklaus Wirth dari Technical University di Zurich, Switzerland. Nama pascal diambil sebagai penghargaan terhadap Blaise Pascal ahli matematika dan philosopi terkenal abad 17 dari Perancis.

Proffesor Niklaus Wirth memperkenalkan kompiler bahasa pascal pertama kali untuk komputer CDC 6000 (Control Data Corporation) yang dipublikasikan pada tahun 1971 dengan tujuan untuk membantu mengajar program komputer secara sistematis, khususnya untuk memperkenalkan program yang terstruktur (*structured programming*). Jadi pascal adalah bahasa yang ditujukan untuk membuat program terstruktur.

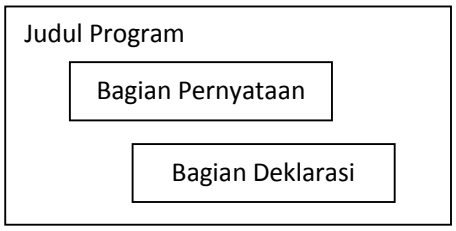
Struktur Program Pascal

Struktur dari suatu program pascal terdiri dari sebuah judul program (*program heading*) dan suatu blok program (*program blok*) atau badan program (*body program*). Blok program ini dibagi lagi menjadi dua bagian, yaitu bagian deklarasi (*declaration part*) dan bagian penyataan (*statement part*). Bagian deklarasi dapat terdiri dari deklarasi label (*label declaration*), deklarasi konstanta (*constanta declaration*), deklarasi tipe (*type declaration*), deklarasi variabel (*variables declaration*), deklarasi prosedur (*procedure declaration*) dan deklarasi fungsi (*function declaration*). Secara ringkas, struktur suatu program pascal dapat terdiri dari:

Judul Program

1. Blok program
    a. Bagian deklarasi
        - Deklarasi label
        - Deklarsi konstanta
        - Deklarasi tipe
        - Deklarasi variabel
        - Deklarasi prosedur
        - Deklarasi fungsi

    b. Bagian pernyataan



Gambar 4. Struktur Program Pascal

METODE

Penulis melakukan penelitian dengan menggunakan metode studi kepustakaan untuk mengumpulkan data dan mengambil informasi dari buku-buku dan sumber referensi lainnya yang berhubungan dengan pembahasan masalah. Setelah itu untuk merancang tampilan output yang akan menggambarkan tentang perjalanan yang dianjurkan dengan waktu seminimal mungkin dengan menggunakan bahasa pemprograman turbo pascal versi 7.1

HASIL DAN PEMBAHASAN

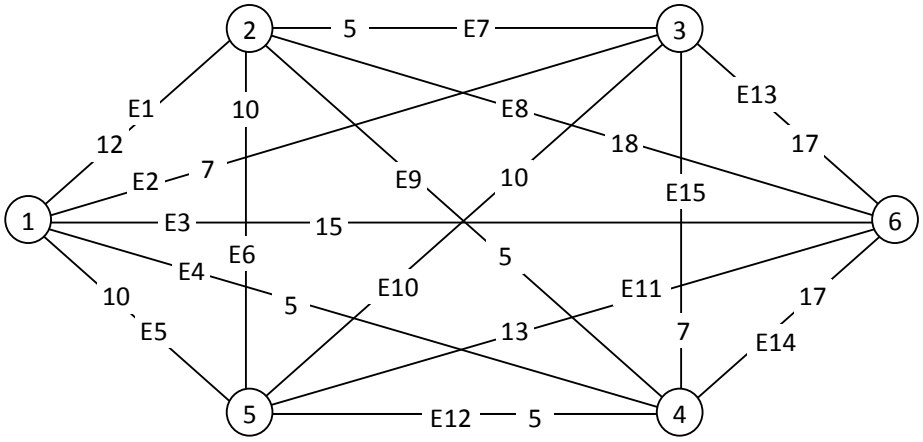
Dasar dan Kasus Permasalahan

Pada bagian ini pembahasan akan lebih mengarah dan lebih khusus lagi. Pada ilustrasi di bawah ini akan disajikan permasalahan yang sering ditemukan dalam kehidupan sehari-hari. Agen minuman akan mengirim pesanan yang diminta pelanggan setelah mereka menghubungi melalui telepon. Mobil pengantar minuman akan mendatangi para pelanggan sesuai dengan banyaknya pelanggan yang memesan dan kapasitas atau daya angkut mobil itu sendiri. Dikarenakan letak warung para pelanggan

berjauhan maka diperlukan cara untuk mengatasi hal ni apabila dapat ditemukan cara yang tepat dapat memberikan keuntungan pada agen minuman, keuntungan ini dapat berupa nama baik dimata pelanggan karena pesanan mereka diantarkan dengan tepat, efisien waktu pengantaran dan penghematan bahan bakar yang dampaknya sedikit banyak mempengaruhi keuntungan.

Penyelesaian Masalah

Dengan menggunakan model matematis, yaitu dengan teori *travelling salesman* akan dicoba untuk mendapatkan waktu seminimal mungkin. Warung-warung tempat langganan agen minuman yang berlokasi disekitar Bogor Selatan dapat dilihat pada ilustrasi graph berikut:



Gambar 5. Lokasi warung-warung dalam bentuk graph tak berarah

Berikut ini adalah contoh lama waktu pengantaran barang ke warung-warung dalam satuan menit dengan jalan yang akan dilalui dalam keadaan normal:

Pada simpul 1(base):	pada simpul ke-2:	pada simpul ke-3:
Base 2 = 12 menit	2 ke 3 = 5 menit	3 ke 4 = 7 menit
Base 3 = 7	2 ke 4 = 5	3 ke 5 = 10
Base 4 = 5	2 ke 5 = 10	3 ke 6 = 17
Base 5 = 10	2 ke 6 = 18	
Base 6 = 15		
Pada simpul ke-4:	pada simpul ke-5:	pada simpul ke-6
4 ke 5 = 5 menit	5 ke 6 = 13 menit	6 ke 1 = 15 menit
4 ke 6 = 17		

Apabila kita akan mencari dengan cara manual maka akan mendapatkan hasil perjalanan dengan lama tempuh yang minimal melalui daerah-daerah berikut:

Base	simpul 4	simpul 2	simpul 3	simpul 5	simpul 6	base	
5	5	5	10	13	15		= 53

Tiap-tiap pengambilan simpul yang berbeda akan menghasilkan waktu tempuh yang berbeda pula. Pada gambar diatas berlaku graph dua arah sehingga dapat dibuat matrik adjasensi dan insidensinya.

Table 3. Matrik Adjasensi

	V1	V2	V3	V4	V5	V6
V1	0	1	1	1	1	1
V2	1	0	1	1	1	1
V3	1	1	0	1	1	1
V4	1	1	1	0	1	1
V5	1	1	1	1	0	1
V6	1	1	1	1	1	0

Tabel 4. Matrik Insidensi

	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12	E13	E14	E15
V1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
V2	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0
V3	0	1	0	0	0	0	1	0	0	1	0	0	1	0	1
V4	0	0	0	1	0	0	0	0	1	0	0	1	0	1	1
V5	0	0	0	0	1	1	0	0	0	1	1	1	0	0	0
V6	0	0	1	0	0	0	0	1	0	0	1	0	1	1	0

Penjelasan perhitungan secara manual:

Pada simpul 1 (base)

Base ke 2 = 17
Base ke 3 = 7
Base ke 4 = 5 • (simpul 4)
Base ke 5 = 10
Base ke 6 = 15

keterangan symbol:
• = alternatif 1
= alternatif 2
∇ = alternatif 3

•

Simpul 4 ke 2 = 5 • (simpul 2)
Simpul 4 ke 3 = 7
Simpul 4 ke 5 = 5 (simpul 5)
Simpul 4 ke 6 = 17

•

Simpul 2 ke 3 = 5 • (simpul 3)
Simpul 2 ke 5 = 10
Simpul 2 ke 6 = 18

Simpul 5 ke 2 = 10 (simpul 2)
Simpul 5 ke 3 = 10 ∇ (simpul 3)
Simpul 5 ke 6 = 12

•

Simpul 3 ke 5 = 10 •
Simpul 3 ke 6 = 17

Simpul 2 ke 3 = 5
Simpul 2 ke 6 = 18

∇

Simpul 3 ke 2 = 5 ∇
Simpul 4 ke 6 = 17

•

Simpul 5 ke 6 = 13 •

Simpul 3 ke 6 = 17

∇

Simpul 2 ke 6 = 18 ∇

•		∇
Simpul 6 ke 1 = 15 •	Simpul 6 ke 1 = 15	Simpul 6 ke 1 = 15 ∇
•		∇
Jumlah • = 53 menit	jumlah = 57 menit	jumlah ∇ = 58 menit

Jadi kesimpulan waktu terpendek yang dianjurkan adalah:

Base	**simpul 4**	**simpul 2**	**simpul 3**	**simpul 5**	**simpul 6**	**base**
5	5	5	10	13	15	= 53

Penjelasan perhitungan secara program:

- procedure input;
  var
  i,j,x,y,bar: byte;
  begin
  {input jumlah simpul}
  window (11,9,56,18);
  repeat
  clrscr;

Menginputkan jumlah simpul (jumsim) dan harus antara 2-10 dengan kondisi apabila memasukkan nilai 0 (nol) pada jumsim, maka akan keluar repeat.

- write ('masukkan jumlah simpul (2-10) = '); readln (jumsim);
  if jumsim = 0 then exit;
  until (jumsim > 2) and (jumsim < 10);
  clrscr;
  writeln ('jumlah simpul = ',jumsim);

Menginputkan nama-nama simpul yang terdapat pada graph.

- writeln ('input nama simpul');

Perulangan (loop) untuk input nama-nama simpul sebanyak jumlah simpul tersebut.

- for i:= 1 to jumsim do
  begin
  write ('simpul ke ' , i , ' = '); readln (namsim[i]);

end;

Perulangan (loop) sebanyak jumlah simpul untuk menampilkan urutan nama-nama simpul secara deret (diakhiri dengan koma) hingga simpul terakhir tanpa diakhiri dengan koma.

- write ('nama simpul: ');
  for i:= 1 to jumsim do
  begin
  if j = jumsim then
  writeln (namsim [j])
  else
  write (namsim [i] , ' , ');
  end; (*selesai untuk urutan nama simpul*)

Menginputkan jarak antar simpul.

- writeln ('input jarak antar simpul');
  gotoxy (12,23); writeln (' ');

```
goyoxy (12,23); writeln ('isikan 999 untuk keluar   ');
bar:=2;
```

Perulangan (loop) untuk menginput jarak dari simpul ke I menuju simpul ke j dimulai pada baris ke 2 (bar = 2) dengan kondisi bila diisi dengan nilai 999 akan keluar kemudian memindahkan nilai path (I,j) ke path (j,I)

```
for i:= 1 to jumsim do
for j:= i+1 to jumsim do
begin
        gotoxy (1,bar); write ('jarak dari ',namsim [i],' ke ',namsim[j],' = ');
         readln (path [i,j]);
if path [i,j]=999 then exit;
path [j,i]:= path [i,j];
```

Kondisi jika jarak simpul (i) ke smpul (j) ≠0, maka kondisi jarak simpul dicetak pada kolom 30.

```
if path [i,j] <> 0 then
begin
    gotoxy (30,bar); writeln (namsim [i],' ke ' ,namsim [j], ' = ',
  path[i,j],' menit ');
end;
```

Kondisi selama baris < 7, baris akan bertambah 1, bila baris > 7 tampilan baris-baris akan bergeser naik.

```
if bar < 7 then
    inc (bar)
  else
      begin
          gotoxy (1,2); delline;
      end;
   end;
   path [jumsim,jumsim]:=0;
end;
```

Pemberian nilai awal total jarak dan simpul ke-1,t(1)

```
procedure proses;
label LAB2,LAB3'
cl: real;
begin
```

Perulangan (loop) untuk menentukan path selanjutnya yang harus dilalui, dimana nilai cl sebagai pembanding untuk mencari nilai jarak terkecil yang harus dilalui.

```
jarak:=0;
  t[1]:=1;
```

Perulangan (loop) untuk menjaga jangan sampai suatu simpul terlewati dua kali apabila ditemukan jarak = 0 dan jarak = nilai terakhir maka selesai

```
for 1:=2 to jumsim do
begin
   cl:= le 30;
```

```
for j:= 1 to jumsim do
begin
```

Membandingkan jarak antar simpul (path(t(1),j) untuk mencari jarak terkecil, dimana cl untuk menampung jarak terkecil dan i1 untuk menampung nomor simpulnya.

- for k:= 1 to (1-1) do
  if t[k]=j then goto LAB2;
  if path [t[l],j]=0 then goto LAB2;
  if path [t[l-1],j]=t1 [jumsim] then goto LAB3;

Setelah nomor simpul atau jaraknya didapat, pindahkan ke t (l) untuk pencarian selanjutnya.

- if path [t[l-1],j] >= cl then goto LAB2;

```
cl:= (path [i,j]);
l1:=j;
LAB2:{label1 LAB2}
end;
```

Perulangan (loop) untuk menampung simpul-simpul yang merupakan jalur terpendek secara berurutan dengan memindahkannya ke tl(k).

- t[1]:=11;
  t[1]:=[11];
  end;
  t[1]:=1;

Looping untuk menghitung total jarak terpendek.

- for k:=1 to jumsim do
  t1[k]:= t[k]

Perulangan untuk mencetak urutan jalur yang dianjurkan

- for i:=1 to jumsim-1 do
  jarak:= jarak+path [t1[1],t1[1+1]];
  lab3:{label LAB3}
  end;

Jalur simpul terakhir tanpa diakhiri dengan tanda panah (->).

- write ('jalur yang dianjurkan:');
  for l:= 1 to jumsim do
  if l = jumsim then

Jalur simpul diakhiri dengan tanda panah (->).

- writeln (namsim [t1[l+l]]);
  else

Pencetakan total jarak terpendek yang ditempuhl

- write (namsim [t1[l+1]]), ' , ' ,chr (26));
  writeln ('dari ke jarak')'
  x:=1;
  y:=1;
  coun:= 0;

Perulangan (loop) untuk mencetak jarak minimal antara simpul l dan i + l

```
for 1:= l to jumsim do
begin
        gotoxy (x,y);
       writeln ('  ',namsim[l],'     ',namsim[t1[l+1],'    ',
        path [t1[l+1],t1[l]]);
        inc (y);
```

kondisi bila baris > 7 akan tercetak pada kolom disebelah kanan ((coun-1)*24)

```
if y > 7 then
 begin
  inc (coun);
  x:= (coun-1)*24;
```

Perulangan (loop) untuk mencetak nama-nama simpul dalam satu kolom sebanyak jumlah simpul.

```
For l:= 1 to jumsim do
 gotoxy (1,8); write ('dengan jumlah jarak =',jarak);
  readln;
  end;
```

Perulangan (loop) untuk mencetak jarak tiap simpul dalam bentuk matrik adjasensi.

```
for l:= 1 to jumsim do
      begin
        gotoxy (i*2+i,1);
         writeln (namsim [l]: 0);
```

Kondisi berapapun nilai jarak antar simpul dicetak dengan lebar 3 karakter tiap nilai jarak.

```
for j:=1 to jumsim do
      begin
            gotoxy (j*2+j,l+l);
```

Perancangan Input / Output

Untuk mencari jarak terpendek dalam permasalahan yang dibahas pada penulisan ilmiah ini selain menggunakan perhitungan dengan cara manual, dapat pula dicari hasilnya dengan bantuan komputer menggunakan program pascal yang terdapat dalam lampiran. Perancangan input/output adalah sebagai berikut: 1) Isi Data, untuk menginput jumlah simpul, nama masing-masing simpul dan jarak antar simpul. 2) Tampilan Data, untuk menampilkan jalur yang dianjurkan dan jumlah jarak terpendek. 3) Matrik, untuk menampilkan matrik adjasensi

Untuk memilih salah satu tombol pada menu, gunakan tanda panah keatas atau tanda panah kebawah pada keyboard.

Tekan tombol “Isi Data” maka akan muncul kalimat perintah untuk memasukkan banyaknya simpul seperti pada gambar . Jumlah simpul yang diinput memiliki ketentuan harus > 2 dan < 10. Bila kondisi jumlah simpul tidak memenuhi ketentuan tersebut, akan terus ditanyakan hingga ketentuan tersebut terpenuhi. Bila diinput jumlah simpul = 0, maka akan keluar dari isi data. Input simpul misalkan = 6, lalu tekan tombol enter pada keyboard.

```
Masukkan jumlah simpul (2-10): 6
```

Gambar 5. Isi Data jumlah simpul

Input berikutnya seperti gambar yang akan menanyakan nama masing-masing simpul. Nama tiap simpul dari ke-10 simpul-simpul tersebut adalah:

```
jumlah simpul = 6

input nama simpul
simpul ke-1 = 1
simpul ke-2 = 2
simpul ke-3 = 3
simpul ke-4 = 4
simpul ke-5 = 5
simpul ke-6 = 6
```

Gambar 6. isi data nama-nama simpul

Tahap penginputan yang terakhir adalah memasukkan nilai jarak antar simpul dengan menginput nilai 0 bila antar simpul tidak terhubung, bisa dilihat pada gambar .

```
jumlah simpul = 6

nama simpul = 1,2,3,4,5,6

input jarak antar simpul
jarak dari 1 ke 2 = 12      1 ke 2 = 12      menit
jarak dari 1 ke 3 = 7       1 ke 3 = 7       menit
jarak dari 1 ke 4 = 5       1 ke 4 = 5       menit
jarak dari 1 ke 5 = 10      1 ke 5 = 10      menit
jarak dari 1 ke 6 = 15      1 ke 6 = 15      menit
jarak dari 2 ke 3 = 5       2 ke 3 = 5       menit
jarak dari 2 ke 4 = 5       2 ke 4 = 5       menit
jarak dari 2 ke 5 = 10      2 ke 5 = 10      menit
jarak dari 2 ke 6 = 18      2 ke 6 = 18      menit
jarak dari 3 ke 4 = 7       3 ke 4 = 7       menit
jarak dari 3 ke 5 = 10      3 ke 5 = 10      menit
jarak dari 3 ke 6 = 17      3 ke 6 = 17      menit
jarak dari 4 ke 5 = 5       4 ke 5 = 5       menit
jarak dari 4 ke 6 = 17      4 ke 6 = 17      menit
jarak dari 5 ke 6 = 13
```

Gambar 7. isi data jarak antar simpul

Dalam menginput jarak antara simpul bila terjadi kesalahan dalam menginput ketik "999" lalu akan keluar dari isi data, kemudian penginputan diulang kembali dengan menekan kotak tombol isi data.

Setelah penginputan data selesai, data akan diproses sehingga menampilkan output berupa total jarak terpendek serta jalur perjalanannya dapat dilihat pada gambar 3.6.

```
jalan yang dianjurkan: 1→4→2→3→5→6→1→

dari        ke          jarak
1           4           5
4           2           5
2           3           5
3           5           10
5           6           13
6           1           15


Dengan jumlah jarak = 53
```

Gambar 8. Jalur terpendek

Setelah jalur terpendek didapat, pilih menu matrik maka akan ditampilkan matrik adjasensi. Dapat dilihat pada gambar di bawah ini

Tabel 5. matrik adjasensi

V	1	2	3	4	5	6
1	0	1	1	1	1	1
2	1	0	1	1	1	1
3	1	1	0	1	1	1
4	1	1	1	0	1	1
5	1	1	1	1	0	1
6	1	1	1	1	1	0

PENUTUP

Kesimpulan

Setelah menyelesaikan pembuatan program aplikasi *traveling salesman*, penulis melihat bahwa metode *traveling salesman* merupakan suatu cara yang dapat membantu para agen minuman untuk mendapatkan waktu yang terpendek dalam mengantarkan pesanan serta dapat membantu perusahaan dalam memperkecil biaya operasional.

Hal yang penting dalam pembuatan aplikasi *traveling salesman* ini adalah mendapatkan jalur terpendek sehingga waktu yang dibutuhkan untuk mengantarkan pesanan menjadi minimum selain itu dapat membantu memperkecil biaya operasional perusahaan.

Saran

Dalam pembuatan program aplikasi *traveling salesman* diperlukan ketelitian misalnya, dalam menentukan variable. Selain menentukan variable hal lain yang harus diperhatikan adalah rumus dalam menentukan matrik adjasensi serta rumus untuk mendapatkan jalur terpendek, apabila salah memasukkan rumus maka jalur terpendek yang diinginkan tidak akan di dapat.

Metode ini dapat dikembangkan dengan mempercantik tampilan dengan

merubah tampilan background serta menggunakan metode transportasi lainnya seperti metode greedy. Dapat juga ditambahkan matrik insidensi.

Untuk pembuatan program aplikasi *traveling salesman* yang sederhana dapat digunakan bahasa pemrograman pascal versi 7.1, tetapi penulis menyarankan untuk mendapatkan hasil yang lebih menarik dapat menggunakan bahasa C atau dengan bahasa pemrograman Delphi.

DAFTAR PUSTAKA

D. Suryadi H. S. 1995. **Pengantar Teori dan Algoritma Graph**. Jakarta. Penerbit Gunadarma.

Hamdy A Taha. 1996. **Riset Operasi Jilid 1,Jakarta. penerbit** Binarupa Aksara.

Hartono Partoharsodjo,1989 **Tuntutan Praktis Pemrograman Bahasa Pascal dengan Contoh Program pada Turbo Pascal 5.0**,Jakarta. Penerbit PT Elex Media Komputindo.

Jogiyanto H. M. 1989. T**eori dan Aplikasi Program Komputer Bahasa Pascal**, Yogyakarta, Penerbit Andi Offset Yogyakarta.

Seymor Lipschutz, ph.D, 2002. **Seri Penyelesaian Soal Schaum: Matematika Diskrit 2, Jakarta. Penerbit** Salemba Teknika.