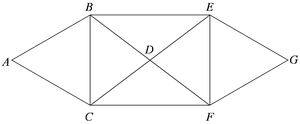
ALGORITMA DIJKSTRA  digunakan untuk mencari rute terpendek……

berikut ini contohnya ::

Graph G dengan matrik berhubungan langsung sebagai berikut ::

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G |
| A | 0 | 60 | 70 | ∞ | ∞ | ∞ | ∞ |
| B | 60 | 0 | 50 | 50 | 20 | ∞ | ∞ |
| C | 70 | 50 | ∞ | 40 | ∞ | 70 | ∞ |
| D | ∞ | 50 | 40 | 0 | 60 | 50 | ∞ |
| E | ∞ | 20 | ∞ | 60 | 0 | 30 | 40 |
| F | ∞ | ∞ | 70 | 50 | 30 | 0 | 50 |
| G | ∞ | ∞ | ∞ | ∞ | 40 | 50 | 0 |

GAMBAR GRAPHNYA SEPERTI INI ::

[](https://elnicovengeance.files.wordpress.com/2011/06/picture2.png)untuk bobotnya dapat dilihat pada tabel yang sebenarnya sebuah “matrik berhubungan langsung” di atas..

PENYELESAIANNYA ::

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Titik Vi | A | B | C | D | E | F | G |
| x (Vi) | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 0 |
| T | A | B | C | D | E | F | G |
|  |  |  |  |  |  |  |  |
| Titik Vi | A | B | C | D | E | F | G |
| x (Vi) | ∞ | ∞ | ∞ | ∞ | 40 | 50 | 0 |
| T | A | B | C | D | E | F | – |
|  |  |  |  |  |  |  |  |
| Titik Vi | A | B | C | D | E | F | G |
| x (Vi) | ∞ | 60 | ∞ | 100 | 40 | 50 | 0 |
| T | A | B | C | D | – | F | – |
|  |  |  |  |  |  |  |  |
| Titik Vi | A | B | C | D | E | F | G |
| x (Vi) | ∞ | 60 | 120 | 100 | 40 | 50 | 0 |
| T | A | B | C | D | – | – | – |
|  |  |  |  |  |  |  |  |
| Titik Vi | A | B | C | D | E | F | G |
| x (Vi) | 120 | 60 | 110 | 100 | 40 | 50 | 0 |
| T | A | – | C | D | – | – | – |
|  |  |  |  |  |  |  |  |
| Titik Vi | A | B | C | D | E | F | G |
| x (Vi) | 120 | 60 | 110 | 100 | 40 | 50 | 0 |
| T | A | – | C | – | – | – | – |
|  |  |  |  |  |  |  |  |
| Titik Vi | A | B | C | D | E | F | G |
| x (Vi) | 120 | 60 | 110 | 100 | 40 | 50 | 0 |
| T | A | – | – | – | – | – | – |
|  |  |  |  |  |  |  |  |
| Titik Vi | A | B | C | D | E | F | G |
| x (Vi) | 120 | 60 | 110 | 100 | 40 | 50 | 0 |
| T | – | – | – | – | – | – | – |

Jadi secara umum cara untuk menyelesaikan [algoritma dijkstra](https://elnicovengeance.wordpress.com/2011/05/13/seo-diperoleh-dari-algoritma-matematika/" \o "SEO DIPEROLEH DARI ALGORITMA MATEMATIKA" \t "_blank) adalah ::

Input :: Graph bobot G dengan s,t elemen V(G)

**Step 1** :: Label titik dengan λ(s)=0 dan untuk setiap titik v di G selain s, label titik v dengan λ(v)=∞ (dalam praktek ∞ diganti dengan bilangan yang “sangat besar” atau diibaratkan sebagai bilangan yang sangat besar). Tulis T = V(G).

**Step 2** :: Misalkan u ε T denga λ(u) minimum

**Step 3** :: Jika u = t, STOP, dan beri pesan: “Panjang lintasan terpendek dari s ke t adalah λ(t)”.

**Step 4** :: untuk setiap sisi e = uv, v ε T, diganti label v dengan λ(v)= minimum {λ(v),λ(u)+W(e)}.

**Step 5** :: Tulis T = T – {u}, dan kembali ke step 2………

keterangan simbol ::

λ = Lamda

ε = elemen

ara Kerja Algoritma Dijkstra

Cara kerja Algoritma dijkstra memakai strategi greedy, dimana pada setiap langkah di pilih sisi dengan bobot terkecil yang menghubungkan sebuah simpul yang sudah terpilih dengan simpul yang sudah terpilih dengan simpul lain yang belum terpilih.

Algoritma Dijkstra membutuhkan parameter tempat asal dan tempat tujuan. Hasil akhir dari algoritma ini adalah jarak terpendek dari tempat asal ke tempat ujuan beserta rutenya.

Algoritma ini sering digunakan pada *routing*

Algoritma dijkstra mencari lintasan terpendek dalam sejumlah langkah. Algoritma ini menggunakan strategi *greedy* sebagai berikut :

Untuk setiap simpul sumber(*source*) dalam graf, algortima ini akan mencari jalur dengna cost minimum antara simpul tersebut dengan simpul lainnya. Algoritma juga dapat digunakan untuk mencari total biaya(*cost*) dari lintasan terpendek yang dibentuk dari sebuah simpul ke sebuah simpul tujuan. Sebagai contoh, bila simpul pada graf merepresentasikan kota dan bobot sisi merepresentasikan jarak antara 2 kota yang mengapitnya, maka algoritma dijkstra dapat digunakan untuk mencari rute terpendek antara sebuah kota dengan kota lainnya.

 Penerapan Algoritma Dijkstra pada Jaringan Komputer

 jaringan komputer dapat dimodelkan sebagai sebuah graf, dengan setiap simpul menyatakan sebuah komputer/router dan sisi di dalam graf menyatakan saluran komunikasi (sering disebut link). setiap sisi mempunyai label nilai (yang disebut bobot).bobot tersebut dapat menyatakan jarak geografis (dalam km), kecepatan transfer data, waktu pengiriman ).

mencari lintasan terpendek dari router asal ke router tujuan dapat diartikan sebagai menentukan lintasan terpendek dari simpul asal ke simpul tujuan di dalam graf yang merepresentasikan jaringan komputer tersebut. Algoritma Dijkstra adalah algoritma yang banyak digunakan untuk mencari lintasan terpendek.

Redistributed Routing Protocol

adalah cara untuk meredistribusikan kembali routing tabel yang dibentuk oleh suatu routing protocol untuk diteruskan ke routing protocol lain. Dengan redistribute kita bisa membentuk routing tabel yang lengkap dari suatu topologi walaupun menggunakan routing protocol yang berbeda .

Cara kerjanya?

Cara kerjanya yaitu router yang menjadi penghubung antara network dengan routing protocol yang berbeda akan menggunakan routing protocol sesuai dengan routing protocol yang dipergunakan oleh kedua network tersebut, misal interface F0/0 pada router tersebut berhubungan dengan network yang menggunakan RIP maka router tersebut harus menggunakan RIP dan pada F0/1 menggunakan OSPF maka router tersebut juga harus menggunakan OSPF sesuai dengan network tempat interface tersebut terhubung. Untuk membuat agar routing tabel yang dibentuk oleh RIP bisa diteruskan menuju ke OSPF maka dipergunakan redistribute RIP, dan sebaliknya agar routing tabel yang terbentuk pada OSPF bisa diteruskan menuju RIP maka dipergunakanlah redistribute OSPF.

[**Algoritma spf / Link-State**](http://dhewitkj1.blogspot.com/2008/12/algoritma-link-state.html)

Link-State  
  
Algoritma link-state juga dikenal dengan algoritma Dijkstra atau algoritma  
shortest path first (SPF). Algoritma ini memperbaiki informasi database dari  
informasi topologi. Algoritma distance vector memiliki informasi yang tidak  
spesifik tentang distance network dan tidak mengetahui jarak router.  
Sedangkan algortima link-state memperbaiki pengetahuan dari jarak router  
dan bagaimana mereka inter-koneksi.  
  
Fitur-fitur yang dimiliki oleh routing link-state adalah:  
- Link-state advertisement (LSA) – adalah paket kecil dari informasi  
routing yang dikirim antar router  
- Topological database – adalah kumpulan informasi yang dari LSA-LSA  
- SPF algorithm – adalah hasil perhitungan pada database sebagai hasil  
dari pohon SPF  
- Routing table – adalah daftar rute dan interface  
  
  
  
  
  
  
  
  
  
  
  
  
Konsep Link-State  
  
**Proses discovery dari routing link-state**  
Ketika router melakukan pertukaran LSA, dimulai dengan jaringan yang  
terhubung langsung tentang informasi yang mereka miliki. Masing-masing  
router membangun database topologi yang berisi pertukaran informasi LSA.  
Algoritma SPF menghitung jaringan yang dapat dicapai. Router membangun  
logical topologi sebagai pohon (tree), dengan router sebagai root. Topologi ini  
berisi semua rute-rute yang mungkin untuk mencapai jaringan dalam protokol  
link-state internetwork. Router kemudian menggunakan SPF untuk  
memperpendek rute. Daftar rute-rute terbaik dan interface ke jaringan yang  
dituju dalam table routing. Link-state juga memperbaiki database topologi  
yang lain dari elemen-elemen topologi dan status secara detail.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Gambar 10.2 jaringan link-state discovery  
  
Router pertama yang mempelajari perubahan topologi link-state melewatkan  
informasi sehingga semua router dapat menggunakannya untuk proses  
update. Gambar 10.3 adalah informasi routing dikirim ke semua router dalam  
internetwork. Untuk mencapai keadaan konvergen, setiap router mempelajari  
router-router tetangganya. Termasuk nama dari router-router tetangganya,  
status interface dan cost dari link ke tetangganya. Router membentuk paket  
LSA yang mendaftar informasi ini dari tetangga-tetangga baru, perubahan  
cost link dan link-link yang tidak lagi valid. Paket LSA ini kemudian dikirim  
keluar sehinggan semua router-router lain menerima itu.  
  
  
  
  
  
  
  
  
  
  
Gambar 10.3 perubahan topologi link-state  
  
Pada saat router menerima LSA, ia kemudian meng-update table routing  
dengan sebagian besar informasi yang terbaru. Data hasil perhitungan  
digunakan untuk membuat peta internetwork dan lagoritma SPF digunakan  
untuk menghitung jalur terpendek ke jaringan lain. Setiap waktu paket LSA  
menyebabkan perubahan ke database link-state, kemudian SPF melakukan  
perhitungan ulang untuk jalur terbaik dan meng-update table routing.  
Ada beberapa titik berat yang berhubungan dengan protokol link-state:  
  
- Processor overhead  
- Kebutuhan memori  
- Konsumsi bandwidth  
  
Router-router yang menggunakan protokol link-state membutuhkan memori  
lebih dan proses data yang lebih daripada router-router yang menggunakan  
protokol distance vector. Router link-state membutuhkan memori yang cukup  
untuk menangani semua informasi dari database, pohon topologi dan table  
routing. Gambar 10.4 menunjukkan inisialisasi paket flooding link-state yang  
mengkonsumsi bandwidth. Pada proses inisial discovery, semua router yang  
menggunakan protokol routing link-state mengirimkan paket LSA ke semua  
router tetangganya. Peristiwa ini menyebabkan pengurangan bandwidth yang  
tersedia untuk me-routing trafik yang membawa data user. Setelah inisial  
flooding ini, protokol routing link-state secara umum membutuhkan bandwidth  
minimal untuk mengirim paket-paket LSA yang menyebabkan perubahan  
topologi.

Metode Link State melakukan tracking atau penyelidikan terhadap semua koneksi yang ada dalam jaringan. Status dari koneksi-koneksi tersebut, jenis dan tipe koneksi, bahkan kecepatan dari koneksi tersebut semuanya dikumpulkan menjadi sebuah informasi. Kumpulan informasi tersebut kemudian di olah untuk menghasilkan daftar-daftar rute yang ada ke suatu tujuan. Kemudian melalui perhitungan itu, semua rute yang ada dipilih salah satu yang terbaik berdasarkan semua parameter yang dikumpulkannya. Hasilnya adalah sebuah rute terbaik yang langsung dapat digunakan oleh data untuk mencapai ke suatu tujuan.  
  
Routing protokol metode Link State akan mengetahui ketika sebuah koneksi di dalam jaringan Anda putus atau mati. Ketika ada yang putus, proses perhitungan metode ini akan segera melepaskan rute-rute yang berhubungan dengan koneksi tersebut, sehingga tidak ada data yang nyasar dan terjebak di jalur putus tersebut. Metode Link State mungkin akan memilih sebuah rute yang didalamnya terdapat banyak hop namun lebih cepat koneksinya

**Lintasan Terpendek dengan Algoritma Dijkstra**

***Lintasan Terpendek (Shortest Path)***

Lintasan terpendek merupakan lintasan minimum yang diperlukan untuk mencapai suatu tempat dari tempat tertentu. Lintasan yang dimaksud tersebut dapat dicari dengan menggunakan graf.

Persoalan dalam mencari lintasan terpendek ini sering terjadi dalam kehidupan sehari hari. Graft yang digunakan dalam pencarian lintasan terpendek adalah graft berbobot (weight graph), yaitu graft yang setiap sisinya diberikan suatu nilai atau bobot. Bobot pada sisi graft dapat menyatakan jarak antar kota, waktu pengiriman pesan, ongkos pembangunan, dan sebagainya.asumsi yang digunakan adalah bahwa semua bobot bernilai positif. Kata “terpendek” berarti meminimisasi bobot pada suatu lintasan di dalam graft.

Hingga saat ini, sudah banyak algoritma mencari lintasan terpendek yang pernah ditulis. Akan tetapi algoritma lintasan terpendek yang paling terkenal adalah algoritma dijkstra. Algoritma dijkstra pertama kali dikembangkan oleh E.W.Dijkstra yaitu seorang ilmuan computer berkebangsaan belanda yang pada perkembangannya menggunakan struktur data yang berbeda-beda, serta memakai strategi greedy, dimana pada setiap langkah dipilih sisi-sisi dengan bobot terkecil yang menghubungkan setiap simpul yang sudah terpilih dengan simpul lainnya.

Terdapat beberapa jenis persoalan lintasan terpendek, anatara lain:

1.      Lintasan terpendek antara dua simpul tertentu.

2.      Lintasan terpendek antara semua pasangan simpul.

3.      Lintasan terpendek dari simpul tertentu ke semua simpul yang lain.

4.      Lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu.

Dibawah ini merupakan contoh program membuat lintasan terpendek menggunakan bahasa pemprograman c++.

#include<stdio.h>

#include<conio.h>

#include<process.h>

#include<string.h>

#include<math.h>

#define IN 99

#define N 6

int dijkstra(int cost[][N], int source, int target);

int dijsktra(int cost[][N],int source,int target)

{

int dist[N],prev[N],selected[N]={0},i,m,min,start,d,j;

char path[N];

for(i=1;i< N;i++)

{

dist[i] = IN;

prev[i] = -1;

}

start = source;

selected[start]=1;

dist[start] = 0;

while(selected[target] ==0)

{

min = IN;

m = 0;

for(i=1;i< N;i++)

{

d = dist[start] +cost[start][i];

if(d< dist[i]&&selected[i]==0)

{

dist[i] = d;

prev[i] = start;

}

if(min>dist[i] && selected[i]==0)

{

min = dist[i];

m = i;

}

}

start = m;

selected[start] = 1;

}

start = target;

j = 0;

while(start != -1)

{

path[j++] = start+65;

start = prev[start];

}

path[j]='\0';

strrev(path);

printf("%s", path);

return dist[target];

}

int main()

{

int cost[N][N],i,j,w,ch,co;

int source, target,x,y;

printf("\t\*\*\*\*Lintaan Algoritma Terpendek (DIJKSRTRA's ALGORITHM)\*\*\*\*\n\n");

for(i=1;i< N;i++)

for(j=1;j< N;j++)

cost[i][j] = IN;

for(x=1;x< N;x++)

{

for(y=x+1;y< N;y++)

{

printf(" Masukkan nilai dari jalur antara simpul %d dan %d: ",x,y);

scanf("%d",&w);

cost [x][y] = cost[y][x] = w;

}

printf("\n");

}

printf("\n Masukkan asal simpul: ");

scanf("%d", &source);

printf("\n Masukkan target simpul: ");

scanf("%d", &target);

co = dijsktra(cost,source,target);

printf("\n Jalur Terpendek: %d",co);

getch();

return(0);

}

Algoritma program diatas adalah sebagai berikut:

Pada program diatas memiliki dua blok program yaitu terdari blok int dijkstra(int cost[][N], int source, int target);. Blok ini digunakan untuk membuat jalur graf yang nantinya akan dihubungkan oleh jalan yang mempunyai jarak yang bernilai tertentu. Yang kedua adalah blok int main() , blok ini digunakan untuk memanggil fungsi-fungsi yang terdapat pada blok program selanjutnya kemudian untuk menginput serta mencetak output program tersebut.

Dalam setiap program harus ada pendeklarasian agar setiap variable yang digunakan tidak terdapat error serta menghindari penggunaan variable yang tidak dikenal oleh program.

int cost[N][N],i,j,w,ch,co;

int source, target,x,y;

Header program dibuat agar setiap penggunanya dapet mengerti mengenai program yang mereka kerjakan. Header tersebut ditulis dengan sintaks sebagai berikut:

printf("\t\*\*\*\*Lintaan Algoritma Terpendek (DIJKSRTRA's ALGORITHM)\*\*\*\*\n\n");.

Kemudian gunakan dua fungsi for awal untuk me-looping dari suatu graf yang telah dibuat pada blok program int dijstra yang mana blok tersebut menggunakan I,,j serta N sebagai varable…..

for(i=1;i< N;i++)

for(j=1;j< N;j++)

cost[i][j] = IN; .

Setelah itu dua for selanjutnya digunakan untuk memastikan simpul yang berhubungan yaitu ditandai dengan variable x dan y berturut dan akan saling terhubung hngga bertemu antara simpul 4 dan 5. x dan y merupakan pendeklarasian jalur simpul mana saja yang terhubung dan akan di input oleh bobot atau nilai antara simpul yang saling terintegrasi yang disebut dengan jarak yang didefinisikan tadi. Variabel yang digunakan untuk input nilai menggunakan w yang mewakili weight.

for(x=1;x< N;x++)

{

for(y=x+1;y< N;y++)

{

printf(" Masukkan nilai dari jalur antara simpul %d dan %d: ",x,y);

scanf("%d",&w);

cost [x][y] = cost[y][x] = w;

}

printf("\n");

}

Kemudian inputan pada simpul awal yang akan dicari nilai terdekatnya dengan simpul berikutnya, dimana &source merupakan variabel dari simpul yang akan dicari.

printf("\n Masukkan asal simpul: ");

scanf("%d", &source);

Setelah menginput awal simpul kemudian menginput target simpul yang akan dicari, dimana variabel &target merupakan tujuan terakhir untuk mencari nilai terdekatnya.

printf("\n Masukkan target simpul: ");

scanf("%d", &target);

setelah itu mencetak nilai terdekat yang didapatkan dari hasil mencari pada program dari setiap jalurnya dengan mencari nilai terkecil dengan memanggil co sebagai variabelnya yang ada dalam blok program diatas.

co = dijsktra(cost,source,target);

printf("\n Jalur Terpendek: %d",co);

setelah semuanya selesai maka langkah selanjutnya adalah mencoba programnya. Seperti yang tertera pada gambar dibawah ini.