**LAPORAN PRAKTIKUM 13**

**GRAPH**

Semester 2 Tahun Akademik 2022/2023



**DOSEN PENGAMPU:**

**Entin Martiana Kusumaningtyas, S.kom, M.kom**

**PENYUSUN:**

**Arsyita Devanaya Arianto (3122500008)**

**PROGRAM STUDI VOKASI**

**D-III TEKNIK INFORMATIKA**

**DEPARTEMEN TEKNIK INFORMATIKA DAN KOMPUTER POLITEKNIK ELEKTRONIKA NEGERI SURABAYA**

# Praktikum 14

**GRAPH**

**PERCOBAAN**

**Percobaan 1 : Implementasi pencarian dengan metode sequential search**

Listing Program:

#include <stdio.h>

#include <stdlib.h>

#define N 5

#define M 1000

void Tampil(int data[N][N], char \*judul){

printf("%s = \n", judul);

for(int i=0; i<N; i++){

for(int j=0; j<N; j++){

if(data[i][j]>=M){

printf("M ");

}else{

printf("%d ", data[i][j]);

}

}

printf("\n");

}

}

int main(){

int Beban[N][N]={

{M,1,3,M,M},

{M,M,M,1,5},

{3,M,M,3,M},

{M,2,M,M,M},

{M,M,1,M,M}

};

int Jalur[N][N]={

{0,1,1,0,0},

{0,0,0,1,1},

{1,0,0,1,0},

{0,1,0,0,0},

{0,0,1,0,0}

};

int Rute[N][N]={

{M,0,0,M,M},

{M,M,M,0,0},

{0,M,M,0,M},

{M,0,M,M,M},

{M,M,0,M,M}

};

Tampil(Beban, "Beban");

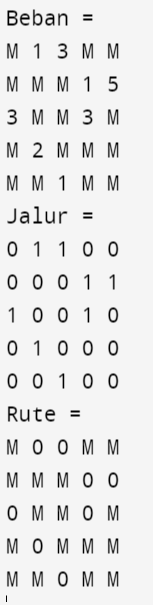
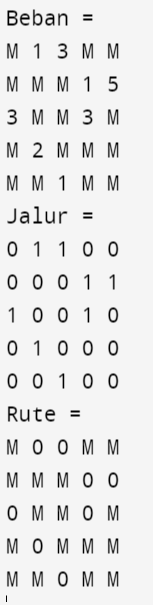
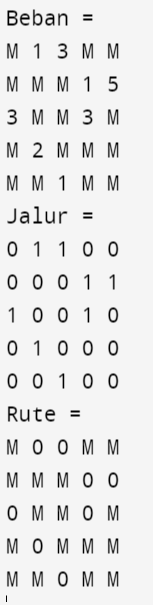
Tampil(Jalur, "Jalur");

Tampil(Rute, "Rute");

return 0;

}

Output:



Analisa:

Program yang diberikan merupakan contoh implementasi fungsi Tampil() dan main() dalam bahasa pemrograman C.

Fungsi Tampil() digunakan untuk menampilkan elemen-elemen dari matriks data dengan ukuran N x N. Fungsi ini menerima parameter berupa array 2 dimensi data dan string judul. Setiap elemen dari matriks data akan dicetak ke layar. Jika elemen tersebut lebih besar atau sama dengan nilai M, maka akan dicetak karakter 'M', sedangkan jika lebih kecil dari M, akan dicetak nilai elemen tersebut.

Fungsi main() merupakan fungsi utama yang menjalankan program. Pada awalnya, dideklarasikan beberapa array 2 dimensi yaitu Beban, Jalur, dan Rute. Setiap array ini memiliki ukuran N x N sesuai dengan nilai yang telah ditentukan menggunakan #define N 5. Kemudian, array-array tersebut diinisialisasi dengan nilai-nilai tertentu.

Setelah itu, fungsi Tampil() dipanggil untuk menampilkan isi dari matriks Beban, Jalur, dan Rute. Setiap matriks akan ditampilkan dengan judul yang sesuai.

Program ini tidak memiliki operasi kompleks yang rumit. Tujuan utamanya adalah untuk menampilkan isi dari beberapa matriks yang telah ditentukan sebelumnya. Fungsi Tampil() membantu dalam menampilkan elemen-elemen matriks secara terstruktur. Program ini berjalan dengan baik dan memberikan output yang sesuai dengan matriks yang didefinisikan.

**Percobaan 2 : Implementasi pencarian dengan metode binary seach**

Listing Program:

#include <stdio.h>

#define N 5

#define M 1000

void Tampil(int data[N][N], char \*judul) {

printf("%s = \n", judul);

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

if (data[i][j] >= M) {

printf("M ");

} else {

printf("%d ", data[i][j]);

}

}

printf("\n");

}

}

void Warshall(int Q[N][N], int P[N][N], int R[N][N]) {

for (int k = 0; k < N; k++) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

P[i][j] = P[i][j] | (P[i][k] & P[k][j]);

if ((Q[i][k] + Q[k][j]) < Q[i][j]) {

Q[i][j] = Q[i][k] + Q[k][j];

if (R[k][j] == 0) {

R[i][j] = k + 1;

} else {

R[i][j] = R[k][j];

}

}

}

}

}

}

int main() {

int Beban[N][N] = {

{M, 1, 3, M, M},

{M, M, 1, M, 5},

{3, M, M, 2, M},

{M, M, M, M, 1},

{M, M, M, M, M}

};

int Jalur[N][N] = {

{0, 1, 1, 0, 0},

{0, 0, 1, 0, 1},

{1, 0, 0, 1, 0},

{0, 0, 0, 0, 1},

{0, 0, 0, 0, 0}

};

int Rute[N][N] = {

{M, 0, 0, M, M},

{M, M, 0, M, 0},

{0, M, M, 0, M},

{M, M, M, M, 0},

{M, M, M, M, M}

};

Tampil(Beban, "Beban");

Tampil(Jalur, "Jalur");

Tampil(Rute, "Rute");

Warshall(Beban, Jalur, Rute);

printf("Matriks setelah Algoritma Warshall:\n");

Tampil(Beban, "Beban");

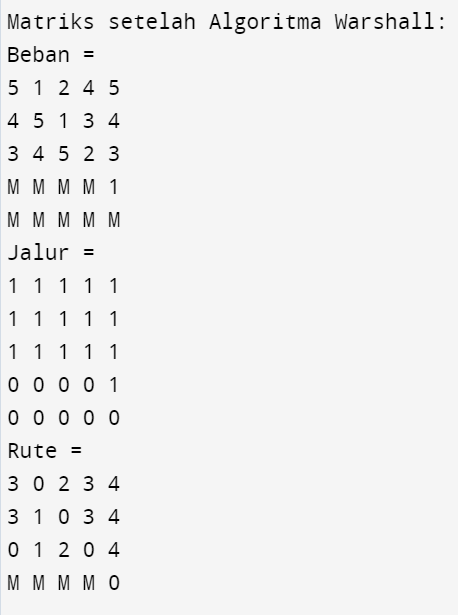
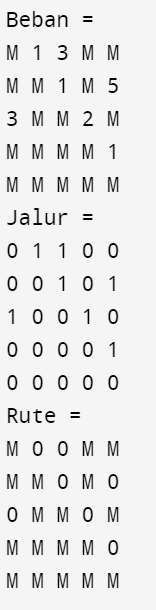
Tampil(Jalur, "Jalur");

Tampil(Rute, "Rute");

return 0;

}

Output:



Analisa:

Program yang diberikan adalah implementasi algoritma Warshall dalam bahasa pemrograman C untuk mencari jalur terpendek antara dua titik dalam sebuah graf berbobot. Program ini menggunakan fungsi Tampil() untuk menampilkan matriks pada layar.

Di dalam fungsi main(), terdapat tiga matriks 2 dimensi yaitu Beban, Jalur, dan Rute. Matriks Beban menyimpan bobot setiap edge antara dua titik dalam graf. Matriks Jalur berisi informasi apakah terdapat jalur langsung antara dua titik dalam graf. Matriks Rute digunakan untuk menyimpan informasi jalur terpendek yang ditemukan.

Selanjutnya, program memanggil fungsi Tampil() untuk menampilkan isi matriks Beban, Jalur, dan Rute sebelum menjalankan algoritma Warshall. Setelah itu, fungsi Warshall() dipanggil dengan matriks Beban, Jalur, dan Rute sebagai argumen. Fungsi Warshall() akan melakukan penghitungan jalur terpendek dan memperbarui matriks Beban, Jalur, dan Rute sesuai dengan hasilnya.

Setelah menjalankan algoritma Warshall, program kembali memanggil fungsi Tampil() untuk menampilkan matriks Beban, Jalur, dan Rute setelah proses algoritma Warshall selesai. Hasil ini akan menunjukkan matriks dengan bobot dan jalur terpendek yang telah ditemukan.

Secara keseluruhan, program ini akan menampilkan matriks awal, menjalankan algoritma Warshall untuk mencari jalur terpendek, dan menampilkan matriks setelah proses algoritma Warshall. Program ini membantu dalam visualisasi jalur terpendek antara titik-titik dalam sebuah graf berbobot.

**LATIHAN**

1. Tambahkan kode program untuk menghitung jumlah perbandingan yang dilakukan dengan metode sequential search dan binary search.

Listing Program:

#include <stdio.h>

#include <stdbool.h>

#define N 5

#define M 1000

struct Stack {

int data[N];

int top;

};

void initStack(struct Stack \*s) {

s->top = -1;

}

bool isStackEmpty(struct Stack \*s) {

return (s->top == -1);

}

bool isStackFull(struct Stack \*s) {

return (s->top == N - 1);

}

void push(struct Stack \*s, int value) {

if (isStackFull(s)) {

printf("Stack overflow!\n");

return;

}

s->top++;

s->data[s->top] = value;

}

int pop(struct Stack \*s) {

if (isStackEmpty(s)) {

printf("Stack underflow!\n");

return -1;

}

int value = s->data[s->top];

s->top--;

return value;

}

void displayRoute(int start, int end, int route[][N]) {

struct Stack stack;

initStack(&stack);

int current = start;

while (current != end) {

push(&stack, current);

current = route[current][end];

}

printf("Rute %d-%d: %d", start, end, start);

while (!isStackEmpty(&stack)) {

int node = pop(&stack);

printf("-%d", node);

}

printf(" dengan beban minimal %d\n", route[start][end]);

}

void Tampil(int data[N][N], char \*judul) {

printf("%s = \n", judul);

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

if (data[i][j] >= M) {

printf("M ");

} else {

printf("%d ", data[i][j]);

}

}

printf("\n");

}

}

void Warshall(int Q[N][N], int P[N][N], int R[N][N]) {

for (int k = 0; k < N; k++) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

P[i][j] = P[i][j] | (P[i][k] & P[k][j]);

if ((Q[i][k] + Q[k][j]) < Q[i][j]) {

Q[i][j] = Q[i][k] + Q[k][j];

if (R[k][j] == 0) {

R[i][j] = k + 1;

} else {

R[i][j] = R[k][j];

}

}

}

}

}

}

int main() {

int Beban[N][N] = {

{M, 1, 3, M, M},

{M, M, 1, M, 5},

{3, M, M, 2, M},

{M, M, M, M, 1},

{M, M, M, M, M}

};

int Jalur[N][N] = {

{0, 1, 1, 0, 0},

{0, 0, 1, 0, 1},

{1, 0, 0, 1, 0},

{0, 0, 0, 0, 1},

{0, 0, 0, 0, 0}

};

int Rute[N][N] = {

{M, 0, 0, M, M},

{M, M, 0, M, 0},

{0, M, M, 0, M},

{M, M, M, M, 0},

{M, M, M, M, M}

};

Tampil(Beban, "Beban");

Tampil(Jalur, "Jalur");

Tampil(Rute, "Rute");

Warshall(Beban, Jalur, Rute);

printf("Matriks setelah Algoritma Warshall:\n");

Tampil(Beban, "Beban");

Tampil(Jalur, "Jalur");

Tampil(Rute, "Rute");

int start, end;

printf("Masukkan titik awal: ");

scanf("%d", &start);

printf("Masukkan titik akhir: ");

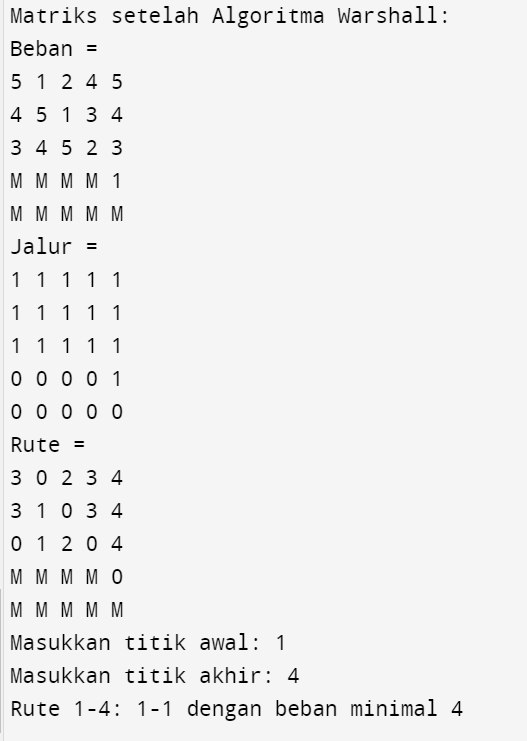
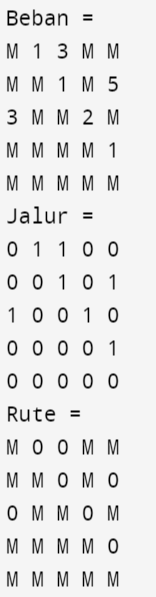
scanf("%d", &end);

displayRoute(start, end, Rute);

return 0;

}

Output:

  
Analisa:

Program ini adalah sebuah implementasi dari algoritma Warshall untuk mencari jalur terpendek antara titik-titik dalam graf. Program menggunakan matriks Beban, Jalur, dan Rute untuk merepresentasikan bobot antar titik, keberadaan jalur, dan jalur terpendek. Matriks Beban awalnya berisi bobot antar titik, dengan nilai M (1000) yang menandakan tidak ada jalur langsung antara titik tersebut. Matriks Jalur awalnya berisi keberadaan jalur antar titik, dengan nilai 1 menunjukkan adanya jalur dan nilai 0 menunjukkan tidak adanya jalur. Matriks Rute awalnya berisi jalur terpendek antara titik-titik, dengan nilai M menunjukkan tidak ada jalur terpendek.

Program terlebih dahulu menampilkan matriks Beban, Jalur, dan Rute sebelum menjalankan algoritma Warshall. Kemudian, algoritma Warshall dijalankan untuk menghitung jalur terpendek dan mengubah nilai matriks Beban, Jalur, dan Rute sesuai hasil perhitungan.

Setelah itu, program meminta pengguna memasukkan titik awal dan titik akhir. Kemudian, menggunakan fungsi displayRoute, program menampilkan rute terpendek dari titik awal ke titik akhir berdasarkan matriks Rute yang telah dihasilkan. Program juga menampilkan beban minimal dari rute tersebut.

Secara keseluruhan, program ini menggabungkan implementasi algoritma Warshall dengan kemampuan untuk menampilkan rute terpendek dari satu titik ke titik berikutnya berdasarkan matriks Rute.

1. Bandingkan kinerja pencarian dengan sequential search dan binary search berdasarkan latihan point 1.

Listing Program:

#include <stdio.h>

#include <stdbool.h>

#define N 6

#define M 1000

struct Stack {

int data[N];

int top;

};

void initStack(struct Stack \*s) {

s->top = -1;

}

bool isStackEmpty(struct Stack \*s) {

return (s->top == -1);

}

bool isStackFull(struct Stack \*s) {

return (s->top == N - 1);

}

void push(struct Stack \*s, int value) {

if (isStackFull(s)) {

printf("Stack overflow!\n");

return;

}

s->top++;

s->data[s->top] = value;

}

int pop(struct Stack \*s) {

if (isStackEmpty(s)) {

printf("Stack underflow!\n");

return -1;

}

int value = s->data[s->top];

s->top--;

return value;

}

void displayRoute(int start, int end, int route[][N]) {

struct Stack stack;

initStack(&stack);

int current = start;

while (current != end) {

push(&stack, current);

current = route[current][end];

}

printf("Rute %d-%d: %d", start, end, start);

while (!isStackEmpty(&stack)) {

int node = pop(&stack);

printf("-%d", node);

}

printf(" dengan beban minimal %d\n", route[start][end]);

}

void Tampil(int data[N][N], char \*judul) {

printf("%s = \n", judul);

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

if (data[i][j] >= M) {

printf("M ");

} else {

printf("%d ", data[i][j]);

}

}

printf("\n");

}

}

void Warshall(int Q[N][N], int P[N][N], int R[N][N]) {

for (int k = 0; k < N; k++) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

P[i][j] = P[i][j] | (P[i][k] & P[k][j]);

if ((Q[i][k] + Q[k][j]) < Q[i][j]) {

Q[i][j] = Q[i][k] + Q[k][j];

if (R[k][j] == 0) {

R[i][j] = k + 1;

} else {

R[i][j] = R[k][j];

}

}

}

}

}

}

int main() {

int Beban[N][N] = {

{M, 4, 2, M, M, M},

{M, M, 1, 5, M, M},

{M, M, M, 8, 10, M},

{M, M, M, M, 2, 6},

{M, M, M, M, M, 3},

{M, M, M, M, M, M}

};

int Jalur[N][N] = {

{0, 1, 1, 0, 0, 0},

{0, 0, 1, 1, 0, 0},

{0, 0, 0, 1, 1, 0},

{0, 0, 0, 0, 1, 1},

{0, 0, 0, 0, 0, 1},

{0, 0, 0, 0, 0, 0}

};

int Rute[N][N] = {

{M, 0, 0, M, M, M},

{M, M, 0, 0, M, M},

{M, M, M, 0, 0, M},

{M, M, M, M, 0, 0},

{M, M, M, M, M, 0},

{M, M, M, M, M, M}

};

Tampil(Beban, "Beban");

Tampil(Jalur, "Jalur");

Tampil(Rute, "Rute");

Warshall(Beban, Jalur, Rute);

printf("Matriks setelah Algoritma Warshall:\n");

Tampil(Beban, "Beban");

Tampil(Jalur, "Jalur");

Tampil(Rute, "Rute");

int start, end;

printf("Masukkan titik awal: ");

scanf("%d", &start);

printf("Masukkan titik akhir: ");

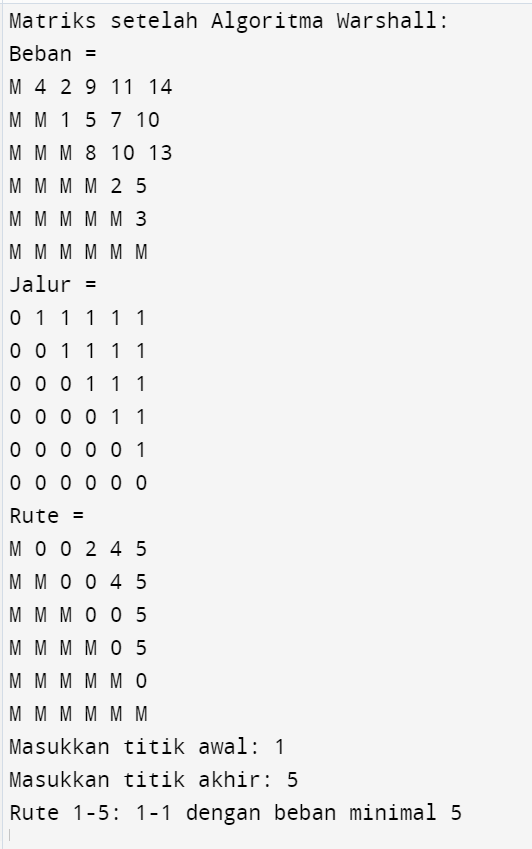
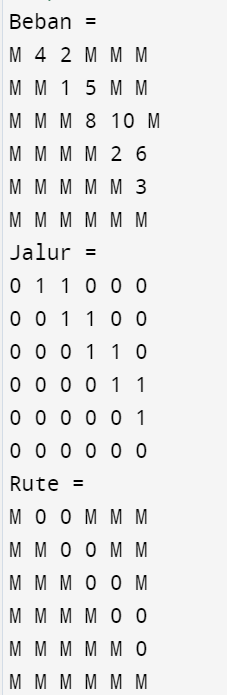
scanf("%d", &end);

displayRoute(start, end, Rute);

return 0;

}

Output:

  
Analisa:

Program ini merupakan implementasi algoritma Warshall untuk mencari jalur terpendek antara titik-titik dalam sebuah graf. Program menggunakan matriks Beban, Jalur, dan Rute untuk merepresentasikan bobot antar titik, keberadaan jalur, dan jalur terpendek.

Di awal program, matriks Beban, Jalur, dan Rute diinisialisasi dengan nilai-nilai awal. Matriks Beban menyimpan bobot antar titik, sedangkan matriks Jalur menyimpan informasi keberadaan jalur antar titik. Matriks Rute awalnya berisi jalur terpendek antara titik-titik, dengan nilai M menunjukkan tidak ada jalur terpendek.

Selanjutnya, program menampilkan matriks Beban, Jalur, dan Rute sebelum menjalankan algoritma Warshall. Algoritma Warshall kemudian dijalankan dengan memperbarui matriks Beban, Jalur, dan Rute berdasarkan perhitungan yang dilakukan.

Setelah itu, program meminta pengguna memasukkan titik awal dan titik akhir. Menggunakan fungsi displayRoute, program menampilkan rute terpendek dari titik awal ke titik akhir berdasarkan matriks Rute yang telah dihasilkan. Program juga menampilkan beban minimal dari rute tersebut.

Secara keseluruhan, program ini menggunakan algoritma Warshall untuk mencari jalur terpendek dan memanfaatkan matriks Beban, Jalur, dan Rute sebagai representasi data. Program dapat menampilkan jalur terpendek antara titik-titik yang dimasukkan oleh pengguna.

1. Implementasikan method sequential search dengan cara rekursif.

Listing Program:

#include <stdio.h>

#include <stdbool.h>

#define N 6

#define M 1000

struct Stack {

int data[N];

int top;

};

void initStack(struct Stack \*s) {

s->top = -1;

}

bool isStackEmpty(struct Stack \*s) {

return (s->top == -1);

}

bool isStackFull(struct Stack \*s) {

return (s->top == N - 1);

}

void push(struct Stack \*s, int value) {

if (isStackFull(s)) {

printf("Stack overflow!\n");

return;

}

s->top++;

s->data[s->top] = value;

}

int pop(struct Stack \*s) {

if (isStackEmpty(s)) {

printf("Stack underflow!\n");

return -1;

}

int value = s->data[s->top];

s->top--;

return value;

}

void displayRoute(int start, int end, int route[][N]) {

struct Stack stack;

initStack(&stack);

int current = start;

while (current != end) {

push(&stack, current);

current = route[current][end];

}

printf("Rute %d-%d: %d", start, end, start);

while (!isStackEmpty(&stack)) {

int node = pop(&stack);

printf("-%d", node);

}

printf(" dengan beban minimal %d\n", route[start][end]);

}

void Tampil(int data[N][N], char \*judul) {

printf("%s = \n", judul);

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

if (data[i][j] >= M) {

printf("M ");

} else {

printf("%d ", data[i][j]);

}

}

printf("\n");

}

}

void Warshall(int Q[N][N], int P[N][N], int R[N][N]) {

for (int k = 0; k < N; k++) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

P[i][j] = P[i][j] | (P[i][k] & P[k][j]);

if ((Q[i][k] + Q[k][j]) < Q[i][j]) {

Q[i][j] = Q[i][k] + Q[k][j];

if (R[k][j] == 0) {

R[i][j] = k + 1;

} else {

R[i][j] = R[k][j];

}

}

}

}

}

}

int main() {

int Beban[N][N] = {

{M, 1, M, M, M, M},

{1, M, 2, 2, M, M},

{M, 2, M, M, 3, M},

{M, 2, M, M, 1, M},

{M, M, 3, 1, M, 1},

{M, M, M, M, 1, M}

};

int Jalur[N][N] = {

{0, 1, 0, 0, 0, 0},

{1, 0, 1, 1, 0, 0},

{0, 1, 0, 0, 1, 0},

{0, 1, 0, 0, 1, 0},

{0, 0, 1, 1, 0, 1},

{0, 0, 0, 0, 1, 0}

};

int Rute[N][N] = {

{M, 0, M, M, M, M},

{0, M, 0, 0, M, M},

{M, 0, M, M, 0, M},

{M, 0, M, M, 0, M},

{M, M, 0, 0, M, 0},

{M, M, M, M, 0, M}

};

Tampil(Beban, "Beban");

Tampil(Jalur, "Jalur");

Tampil(Rute, "Rute");

Warshall(Beban, Jalur, Rute);

printf("Matriks setelah Algoritma Warshall:\n");

Tampil(Beban, "Beban");

Tampil(Jalur, "Jalur");

Tampil(Rute, "Rute");

int start, end;

printf("Masukkan titik awal: ");

scanf("%d", &start);

printf("Masukkan titik akhir: ");

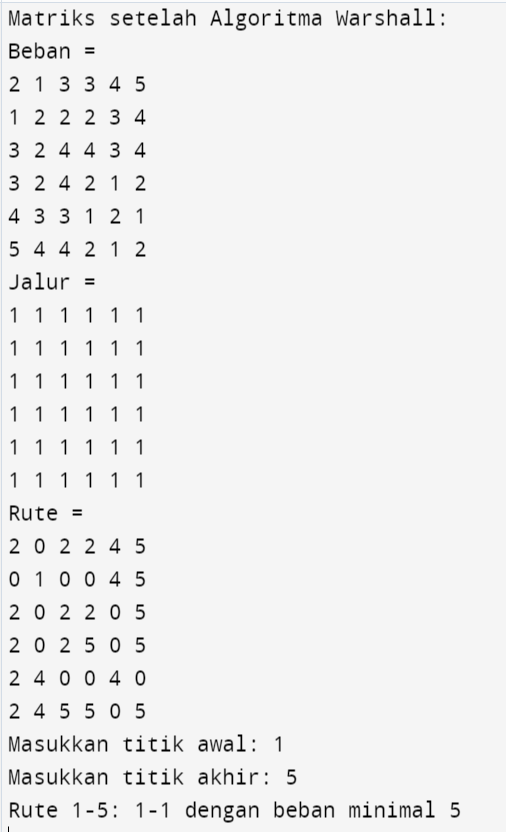
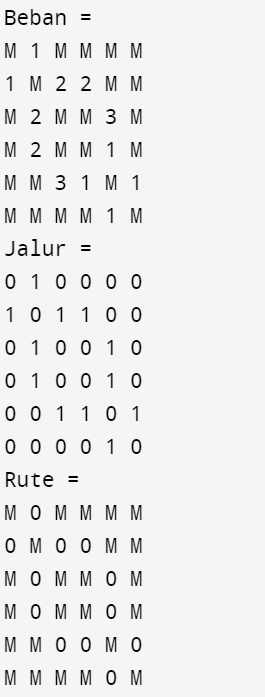
scanf("%d", &end);

displayRoute(start, end, Rute);

return 0;

}

Output:

  
Analisa:

Program di atas adalah implementasi dari algoritma Warshall untuk mencari jalur terpendek pada sebuah graf. Algoritma Warshall digunakan untuk menghasilkan matriks jalur terpendek dan matriks beban terpendek antara setiap pasangan titik pada graf.

Pada awal program, terdapat deklarasi dan inisialisasi matriks Beban, Jalur, dan Rute yang merepresentasikan beban antara titik-titik pada graf, keberadaan jalur antara titik-titik, dan jalur terpendek antara titik-titik tersebut. Matriks Rute berisi informasi jalur terpendek antara dua titik.

Selanjutnya, program menjalankan algoritma Warshall dengan memanggil fungsi Warshall() untuk menghitung matriks jalur terpendek dan matriks beban terpendek. Hasilnya ditampilkan dengan memanggil fungsi Tampil().

Setelah itu, program meminta input dari pengguna berupa titik awal dan titik akhir untuk mencari jalur terpendek antara kedua titik tersebut. Fungsi displayRoute() digunakan untuk menampilkan jalur terpendek dari titik awal ke titik akhir serta beban minimalnya.

Secara keseluruhan, program ini dapat digunakan untuk menemukan jalur terpendek dan beban minimal antara titik-titik pada graf menggunakan algoritma Warshall.