Reto Semestral – Ronda 3

Layla Tame

A01192934

Sacar costo mínimo de un grafo dirigido para llegar de la ciudad A(src) a la ciudad B (dst) con un número limite de escalas(k). Si no se puede, regresar -1.

Propuesta de solución

Floyd, en la reconstrucción de caminos si se pasa del numero de escalas descartar (calculo del minimo)

Utilizar el algoritmo de Floyd, cuando se cumpla la condición de que el valor calculado sea menor al valor de la matriz actual (if), se reconstruye el camino con la matriz como está y se cuenta el número de escalas, si este es menor a k, se actualiza el valor. De lo contrario, se queda el valor existente.

Void Floyd(int matriz costos, int escalas(esc))

For i=0, < size

For j= 0, < size

For k=0, < size

If(cost[j][i]+cost[i][k] < cost[j][k]

If -> reconstruir camino (matriz caminos) < esc

Actualizar costo //Cost[j][k] = cost[j][i]+cost[i][k]

Agregar escala a caminos

//camino[j][k] = camino[i][k]

Arbol de expansión minima?? Con restricciones?

//

// main.cpp

// Floyd

//

// Created by Layla Tame on 2/26/19.

// Copyright © 2019 Layla Tame. All rights reserved.

//

// Ejercicio para probar y analizar el algoritmo de Floyd.

#include <cstdlib>

#include <iostream>

using namespace std;

#define TAM 7

int camino[TAM][TAM];

int iSize[TAM][TAM] = {0};

/\*int costos[TAM][TAM] = {

{ 0, 9999, 9999, 500, 9999, 9999, 100},

{ 9999, 0, 345, 9999, 9999, 9999, 480},

{ 340, 9999, 0, 9999, 9999, 222, 9999},

{ 77, 9999, 9999, 0, 9999, 1075, 9999},

{ 9999, 900, 900, 9999, 0, 9999, 9999},

{ 9999, 9999, 760, 9999, 9999, 0, 120},

{ 9999, 500, 9999, 9999, 230, 9999, 0}};\*/

int costos1[TAM][TAM] = {

{ 0, 9999, 9999, 500, 9999, 9999, 100},

{ 9999, 0, 345, 9999, 9999, 9999, 480},

{ 340, 9999, 0, 9999, 9999, 222, 9999},

{ 77, 9999, 9999, 0, 9999, 1075, 9999},

{ 9999, 900, 900, 9999, 0, 9999, 9999},

{ 9999, 9999, 760, 9999, 9999, 0, 120},

{ 9999, 500, 9999, 9999, 230, 9999, 0}};

int costos[TAM][TAM] = {

{ 0, 9999, 9999, 3, 9999, 9999, 6},

{ 9999, 0, 3, 9999, 9999, 9999, 7},

{ 2, 9999, 0, 4, 9999, 6, 9999},

{ 3, 9999, 9999, 0, 9999, 8, 9999},

{ 9999, 5, 6, 9999, 0, 9999, 9999},

{ 5, 9999, 7, 9999, 9999, 0, 11},

{ 9999, 7, 9999, 9999, 10, 9999, 0}};

int costos2[7][7] = { {0, 9999, 9999, 500, 9999, 9999, 9999}, { 9999, 0, 9999, 9999, 9999, 9999, 480}, { 340, 9999, 0, 9999, 9999, 9999, 9999}, { 9999, 9999, 9999, 0, 9999, 1075, 9999}, { 9999, 900, 9999, 9999, 0, 9999, 9999}, { 9999, 9999, 760, 9999, 9999, 0, 120}, { 9999, 9999, 9999, 9999, 230, 9999, 0} };

void despliegaMatriz(int m[TAM][TAM])

{

for(int i=0;i<TAM;i++)

{

for(int j=0;j<TAM;j++)

cout << m[i][j] << " ";

cout << endl;

}

}

void despliegaCamino(int camino[TAM][TAM], int i, int j)

{

if(camino[i][j] == i)

{

return;

}

despliegaCamino(camino, i, camino[i][j]);

cout << camino[i][j] << " ";

}

int min(int a,int b)

{

if(a<b)

return(a);

return(b);

}

void floyd(int p[TAM][TAM])

{

for (int i = 0; i < TAM; i++)

{

for (int j = 0; j < TAM; j++)

{

if (i == j)

camino[i][j] = 0;

else if (p[i][j] != 9999)

camino[i][j] = i;

else

camino[i][j] = -1;

}

}

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

for(int i=0;i<TAM;i++){ //I = 2 0,1 1,2 < 0,2 k = 1, i = 0, j = 2 //PARA IR DE 0 A 2, ES DE 0 A 1 Y DE 1 A 2

for(int j=0;j<TAM;j++){ //J = 1 INFINITO

//cout << "AHORITA: " << i << "," << j << endl;

//cout << "Sumando: " << p[i][k] << " con " << p[k][j] << endl;

if (p[i][k]+p[k][j] < p[i][j])

{

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

camino[i][j] = camino[k][j]; //AL CAMINO DE 0 AL 2, se agrega

//camino[j][iSize[j]+1] = k;

//camino[j].push\_back(i, k);

iSize[i][j] = iSize[i][j] + 1;

}

//p[i][j]=min(p[i][j],p[i][k]+p[k][j]);

//camino[2].push\_back(k);

//i = 0, j = 2 > i = 0, k = 1 + k = 1, j= 2

}

}

}

}

void floydLimitada(int p[TAM][TAM], int nodo)

{

for(int k=0;k<nodo+1;k++)

for(int i=0;i<nodo+1;i++)

for(int j=0;j<nodo+1;j++)

p[i][j]=min(p[i][j],p[i][k]+p[k][j]);

}

void printPath(int camino[][TAM], int i, int j)

{

if (camino[i][j] == i)

return;

printPath(camino, i, camino[i][j]);

cout << camino[i][j] << " ";

}

int main()

{

int n1, n2;

cout << "Ingrese dos nodos: ";

cin >> n1 >> n2;

floyd(costos2);

cout << endl << "Matriz floyd" << endl;

despliegaMatriz(costos);

cout << endl << "Costo: " << endl << costos[n1][n2] << endl << endl;

cout << "Camino: " << endl;

//printPath(camino, n1, n1);

despliegaMatriz(camino);

cout << endl << "Camino del " << n1 << " al " << n2 << ": " << n1 << " ";

despliegaCamino(camino, n1, n2);

cout << n2 << endl;

cout << endl << "Matriz tamaño: " << endl;

despliegaMatriz(iSize);

cout << endl << "Escalas: " << iSize[n1][n2] << endl;

cout << "Residuo: " << 9/5;

/\*for (int i = 0; i < TAM; i++)

{

cout << camino[n2][i] << " ";

}\*/

/\*

cout << endl << "Pasando máximo por qué nodo? ";

cin >> nodo;

cout << endl << "De que nodo a que nodo? ";

cin >> n1 >> n2;

floydLimitada(costos1, nodo);

despliegaMatriz(costos1);

cout << endl << costos1[n1][n2];

\*/

//system("pause");

}