CSC 323 Spring 2015: Dijkstra SSS (C++)

Ravi Patel

April 21, 2015

**Algorithm steps:**

Step 0: prepare and initialize everything

- load the cost matrix

- get the source node

- set marked array to zero

- set parent array to source

- set the bestCost array to cost matrix of for the source row

Step 1: minNode <-- source node

Step 2: Mark the minNode

Step 3: // expand minNode and computer the new cost

newNode <-- get the next un-marked node,

tempCost <-- bestCost[minNode] + costMatrix[minNode, newNode]

if tempCost < bestCost[newNode]

bestCost[newNode] <-- tempCost

parent[newNode] <-- minNode

Step 3: repeat step 2 until all un-marked nodes are processed.

Step 4: minNode <-- find the node with the smallest bestCost

Step 5: repeat step 2 to step 4 until all nodes are marked.

Step 6: print the cost matrix

print the source node

print all paths from source to each of the nodes in graph with cost

in the format given above // you may trace the path by the parent array.

Source Code:

//

// Dijkstra.cpp

//

//

// Created by Ravi Patel on 4/19/15.

//

//

#include <iostream>

#include <fstream>

using namespace std;

int main(int argc, char\* argv[]){

ifstream inputFile;

inputFile.open(argv[1]);

ofstream outputFile;

outputFile.open(argv[2]);

int nodes, from, to, weight;

inputFile>>nodes;

int parent[nodes];

int marked[nodes];

int bestCost[nodes];

int costMatrix[nodes][nodes];

while(inputFile>>from && inputFile>>to && inputFile>>weight){

costMatrix[from][to]=weight;

}

for(int row = 0; row < nodes; row++){

for(int col=0; col < nodes; col++){

outputFile<<costMatrix[row][col]<<" ";

}

outputFile<<endl;

}

int i=1;

while(i <= nodes+1){

int minNode=i;

marked[minNode]=1;

bool finish=false;

while(finish == false){

int newNode;

for(int row=0; row <=nodes; row++ ){

if(marked[row]!=1){

break;

}

newNode=marked[row];

}

int tempCost= bestCost[minNode] + costMatrix[minNode][newNode];

if(tempCost < bestCost[newNode]){

bestCost[newNode]=tempCost;

parent[newNode]= minNode;

}

finish=true;

}

i++;

return 0;

}

}

Input:

6

1 2 5

2 1 10

1 4 10

4 1 20

3 2 5

3 4 10

3 5 30

5 4 5

4 5 20

1 6 2

2 6 2

3 6 3

4 6 6

5 6 2

6 1 8

6 2 4

6 3 3

6 4 1

6 5 5

Output:

0 0 0 0 1582757072 32767

23671223 1 5 32767 10 32767

2 10 -2046472313 32767 858927408 926299444

2 1717920867 5 1484277317 10 30

3 20 1582758296 32767 0 20

6 0 1582758296 32767 5 32767