#### Name: Teresa Condon

#### CS 305 HW 6 Report

**1. Questions (include these in your write-up):**

1a. (.5 pt) If your program does not meet the specifications, please note the differences and anything that is not working correctly. If it works, write “Meets all specifications.”

Meets all specifications.

1b. (1.5 pts, 0.25 pts each) What is the output from your program for the following pairs of cities? (The top row has been done for you.)

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Destination** | **Path** | **Miles** |
| PDX | MCO | PDX -> ATL -> MCO | 2572 |
| LAX | PVD | LAX -> SAN -> JFK -> PVD | 2693 |
| ATL | JFK | ATL -> LAX -> SAN -> JFK | 4491 |
| SEA | LAX | SEA -> PDX -> LAX | 965 |
| PHX | DEN | PHX -> SEA -> PDX -> DEN | 2227 |
| PVD | JFK | PVD -> JFK | 114 |
| DFW | ATL | DFW -> SAN -> LAX -> ATL | 3219 |

2. (1 pt) Use **Prim**’s algorithm to find the minimum spanning tree for the following subset of the airport graph. Start the algorithm using vertex LAX.

SEA

DFW

LAX

PDX

SAN

JFK

ATL

1658

1168

1051

130

2440

109

2447

835

1942

2168

In your answer, show the final minimum spanning tree edges and all the vertices.

LAX -> SAN (109)

LAX -> PDX (835)

PDX -> SEA (130)

SAN -> DFW (1168)

LAX -> ATL (1942)

SAN -> JFK (2440)

SEA

DFW

LAX

PDX

SAN

JFK

ATL

1168

130

2440

109

835

1942

3. (1.5 pts) Show each step of **Kruskal**’s algorithm for finding the minimum spanning tree for the graph in #2. Draw the minimum spanning tree at each step of the algorithm. For example, the first step would be to put edge (PDX, SEA) into the spanning tree. Be sure to enumerate all the steps and show the final minimum spanning tree.

Add edge (LAX, SAN)

Add edge (PDX, SEA)

Add edge (LAX, PDX)

Add edge (SAN, DFW)

Add edge (LAX, ATL)

Add edge (SAN, JFK)

SEA

DFW

LAX

PDX

SAN

JFK

ATL

1168

130

2440

109

835

1942

4 a. (.25 pt) How much time did you spend in total on this homework assignment (including the report)?

7 hours.

b. (.25 pt) What was the most challenging part for you when completing this assignment?

Debugging.

**Appendix A: (copy this statement if it applies to you)** I verify that the code and this write-up were authored by me. I have documented the help I have received in comments in the code files. I have not distributed my code to anyone else except via this homework submission.

**I verify that the code and this write-up were authorized by me. I have documented the help I have received in comments in the code files. I have not distributed my code to anyone else except via this homework submission.**

**Appendix B**: Copy and paste your dijkstra.c here (use Courier New 8pt font so the characters line up correctly)

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <assert.h>

#include "graph.h"

#include "main.h"

/\*

\* @AUTHOR Teresa Condon

\* @Date April 16, 2016

\* @Class CS 305 HW 6.

Implements Dijkstra's Algorithm (which finds the shortest path in a graph).

\*/

/\*

Implements Dijkstra's algorithm as follows:

Find source node by name lookup

Set the source node’s dValue to 0.

While the graph is not empty :

Let U = getMin(g) //use getMin function

For each neighbor N of U:

If N’s dValue > [U’s dValue + cost(U, N)], update N’s dValue to [U’s dvalue + cost(U, N)]. Also, update the predecessor for N to be U.

Set U’s color to black

\*/

void dijkstra(Graph \* g, char \* source)

{

int i, srcIndx = -1;

for (i=0; i < g->V; i++) // find node by name

{

if (strcmp(source, g->array[i].label) == 0)

{

srcIndx = i;

}

}

assert(srcIndx >= 0 );

g->array[srcIndx].dValue = 0;

while (!isEmpty(g))

{

int u = getMin(g);

AdjListNode \*np = g->array[u].head; // a pointer to the first neighbor

while (np != NULL) // Foreach neighbor N of U

{

int n = np->dest;

int newVal = g->array[u].dValue + np->cost;

if (g->array[n].dValue > newVal)

{

g->array[n].dValue = newVal;

g->array[n].pred = u;

}

np=np->next;

g->array[u].color = BLACK;

}

}

}

/\*

\* Return 1 if graph is empty or has no white nodes

\*/

int isEmpty(Graph \* g)

{

if (g->array->head == NULL )

{

return 1;

}

int i;

for (i=0; i<g->V; i++)

{

if (g->array[i].color == WHITE)

{

return 0;

}

}

return 1;

}

// returns the lowest index of a white (unvisited) node

int getMin(Graph \* g)

{

if (isEmpty(g))

{

return -1;

}

int i, min\_i = -1, min = INF;

for (i=0; i<g->V; i++ )

{

if (g->array[i].color == WHITE && g->array[i].dValue < min) // if "white" and minimal

{

min = g->array[i].dValue;

min\_i = i;

}

}

assert(min\_i >= 0);

return min\_i;

}