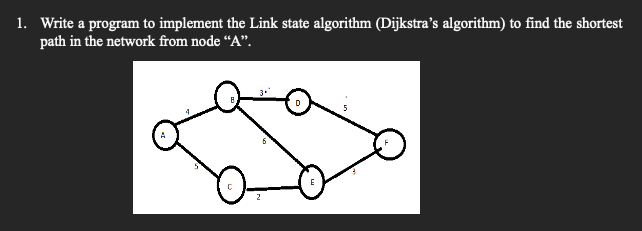
Name: Samyam Budhathoki

Reg no: 22BCE3908

Computer Networks Lab Assessment 4



Code:

#include <stdio.h>

#include <limits.h>

#define V 6

int minDistance(int dist[], int sptSet[]) {

int min = INT\_MAX, min\_index;

for (int v = 0; v < V; v++)

if (sptSet[v] == 0 && dist[v] <= min)

min = dist[v], min\_index = v;

return min\_index;

}

void printSolution(int dist[], int n) {

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

printf("%c \t %d\n", i + 'A', dist[i]);

printf("\n");

}

void dijkstra(int graph[V][V], int src) {

int dist[V];

int sptSet[V];

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

dist[i] = INT\_MAX, sptSet[i] = 0;

dist[src] = 0;

for (int count = 0; count < V - 1; count++) {

int u = minDistance(dist, sptSet);

sptSet[u] = 1;

for (int v = 0; v < V; v++)

if (!sptSet[v] && graph[u][v] && dist[u] != INT\_MAX

&& dist[u] + graph[u][v] < dist[v])

dist[v] = dist[u] + graph[u][v];

}

printSolution(dist, V);

}

int main() {

int graph[V][V] = {

{0, 4, 5, 0, 0, 0},

{4, 0, 0, 3, 0, 5},

{5, 0, 0, 6, 2, 0},

{0, 3, 6, 0, 0, 5},

{0, 0, 2, 0, 0, 3},

{0, 5, 0, 5, 3, 0},

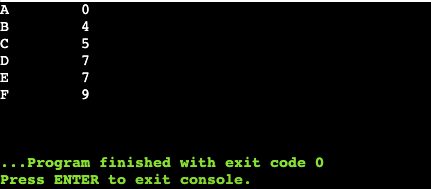
};

dijkstra(graph, 0);

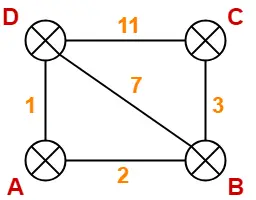
return 0;

}

Output:



Implement the Distance vector algorithm (Bellman-Ford) to calculate the routing tables for each route in the network.



Solution:

Code:

#include <stdio.h>

#include <limits.h>

#define NODES 4

struct Connection {

int start, end, cost;

};

void displayTable(int distances[], int size) {

printf("Node \t Distance from Source\n");

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

printf("%c \t %d\n", i + 'A', distances[i]);

}

}

void calculatePaths(struct Connection connections[], int numConnections, int source) {

int distances[NODES];

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

distances[i] = INT\_MAX;

distances[source] = 0;

for (int i = 0; i < NODES - 1; i++) {

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

int startNode = connections[j].start;

int endNode = connections[j].end;

int connectionCost = connections[j].cost;

if (distances[startNode] != INT\_MAX && distances[startNode] + connectionCost < distances[endNode])

distances[endNode] = distances[startNode] + connectionCost;

}

}

displayTable(distances, NODES);

}

int main() {

struct Connection connections[] = {

{0, 1, 2}, {1, 0, 2}, {0, 3, 1}, {3, 0, 1}, {1, 2, 3},

{2, 1, 3}, {2, 3, 11}, {3, 2, 11}, {3, 1, 7}, {1, 3, 7}

};

int numConnections = sizeof(connections) / sizeof(connections[0]);

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

printf("Routing table for node %c:\n", i + 'A');

calculatePaths(connections, numConnections, i);

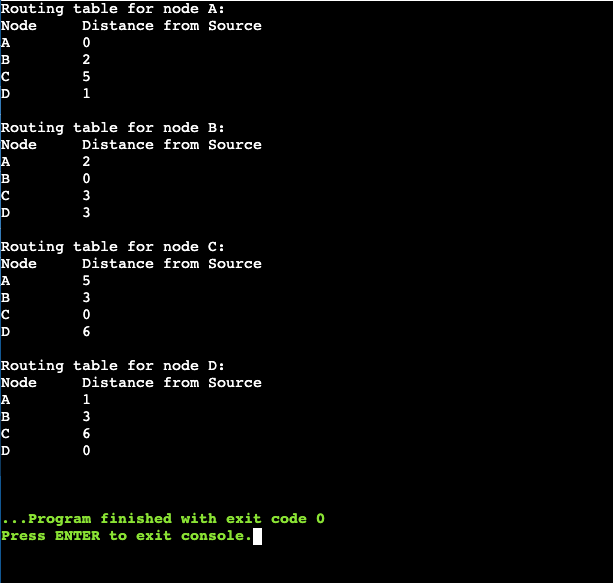
printf("\n");

}

return 0;

}

Output:



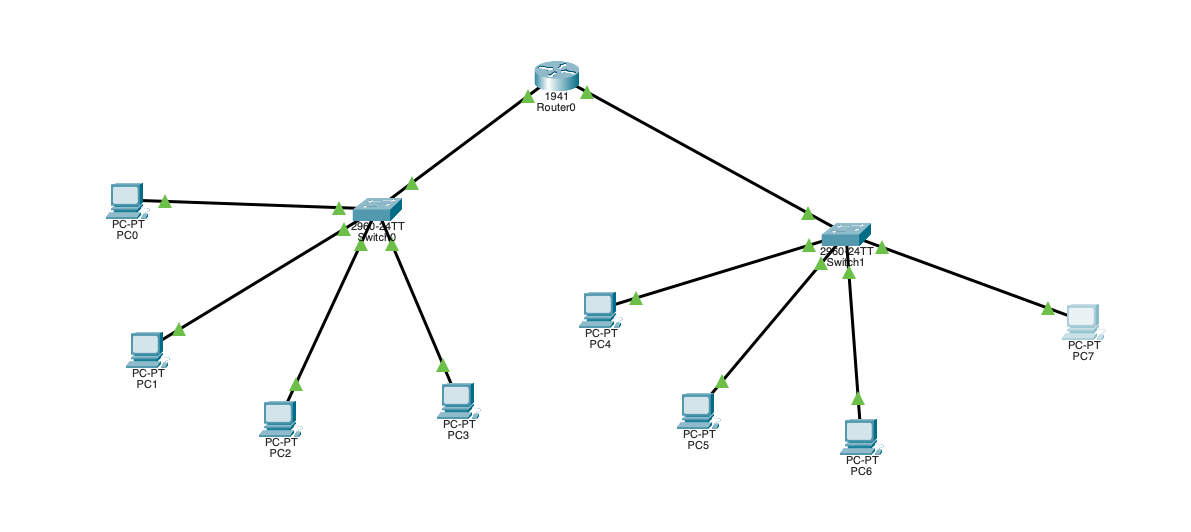
An ISP is granted a block of addresses starting with 150.80.0.0/16. The ISP wants to distribute these blocks to 2600 customers as follows. Implement the subnet using CISCO packet tracer for any two group of subnet. Each subnet having 4 addresses.

a. The first group has 200 medium-size businesses; each needs 128 addresses.

b. The second group has 400 small businesses; each needs 16 addresses.

c. The third group has 2000 households; each needs 4 addresses.

Solution:



For small businesses:

* Each business requires 16 addresses, so we'll use a **/28 subnet** (which provides 16 addresses).
* A total of 400 subnets will be needed.

For households:

* Each household needs 4 addresses, so we'll assign a **/30 subnet** (which provides 4 addresses).
* A total of 2000 subnets are necessary.

**Small Business Subnet:**

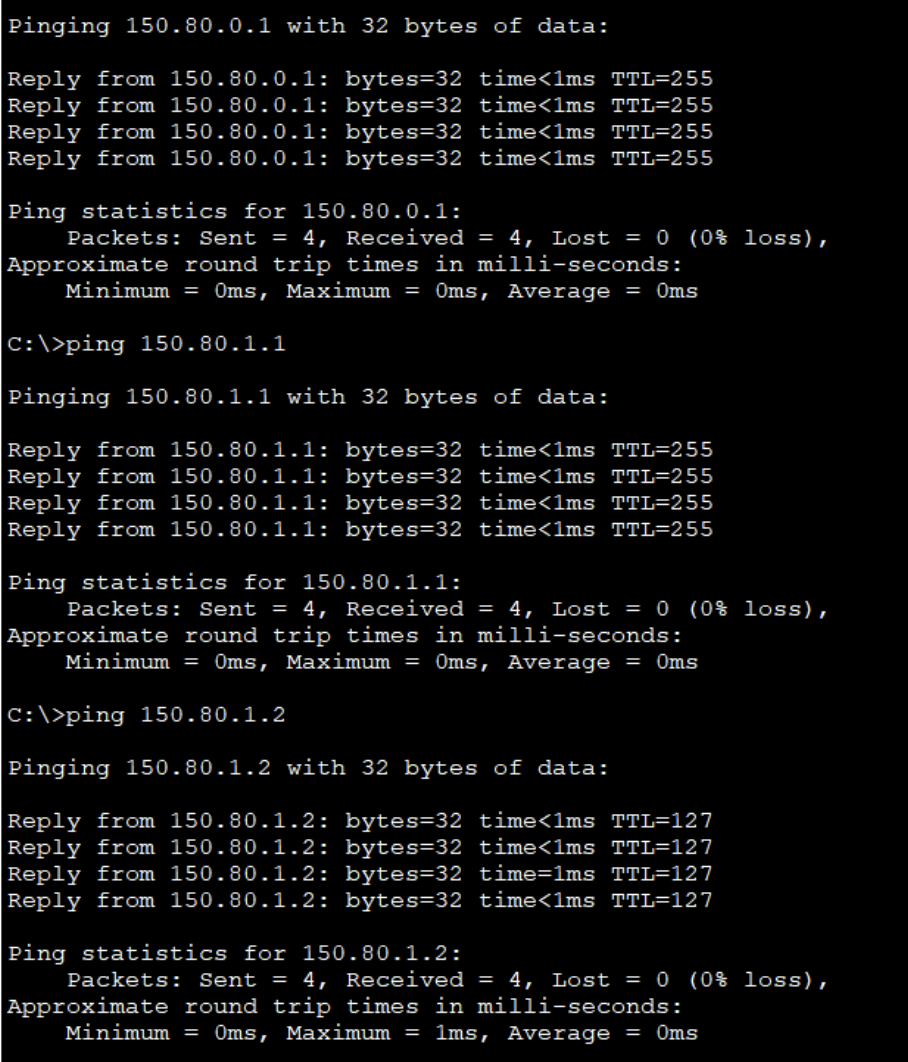
* **Network**: 150.80.0.0/28
* **Usable IP Range**: 150.80.0.1 to 150.80.0.14
* **Gateway**: 150.80.0.1

**Household Subnet:**

* **Network**: 150.80.1.0/30
* **Usable IP Range**: 150.80.1.1 to 150.80.1.2
* **Gateway**: 150.80.1.1

Testing Connectivity

For small business PCs:



For Household PCs

