CS146-Lab9 Name: Michael Huang Due date: April 27th, 2016

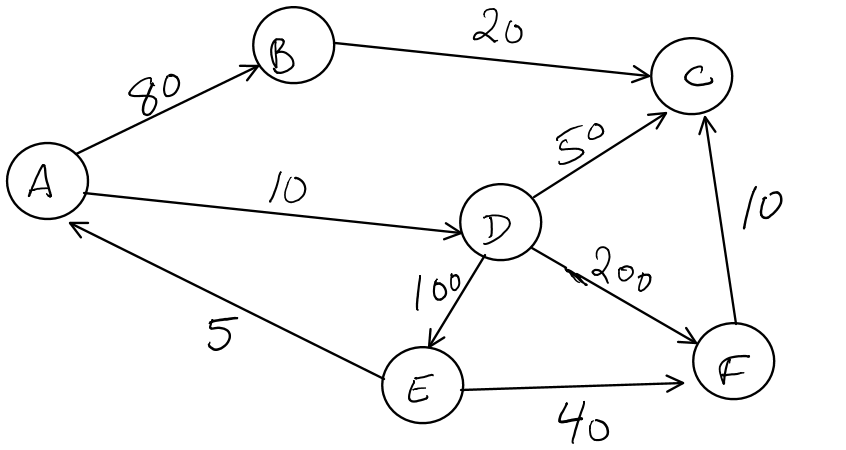
Ex-1: Write a program to solve the Dijkstra’s algorithm explained in class using adjacency matrix and the tables used.

Code:

|  |
| --- |
| public class Path {  private int numOfVertices;  private int source;  public void dijkstra(int graph[][], int src) {  numOfVertices = graph.length;  source = src;  int[] distance = new int[numOfVertices];  int[] parent = new int[numOfVertices];  boolean[] known = new boolean[numOfVertices];  for (int i = 0; i < numOfVertices; i++) {  parent[i] = -1;  distance[i] = 99999;  known[i] = false;  }  distance[src] = 0;  for (int count = 0; count < numOfVertices - 1; count++) {  int u = minDistance(distance, known);  known[u] = true;  for (int v = 0; v < numOfVertices; v++) {  if (!known[v] && graph[u][v] != 0 && distance[u] + graph[u][v] < distance[v]) {  parent[v] = u;  distance[v] = distance[u] + graph[u][v];  }  }  }  printTable(distance, numOfVertices, parent);  }    private int minDistance(int distance[], boolean known[]) {  int min = 99999;  int min\_index = -1;  for (int i = 0; i < numOfVertices; i++) {  if (known[i] == false && distance[i] <= min) {  min = distance[i];  min\_index = i;  }  }  return min\_index;  }    private void printPath(int parent[], int j) {  if (parent[j] == -1) {  return;  }  printPath(parent, parent[j]);  System.out.printf("%d ", j);  }  private void printTable(int distance[], int n, int parent[]) {  int src = source;  int temp = src;  String path = "";  for (int i = 0; i < numOfVertices; i++) {  if (distance[i] == 99999) {  path = "No path.";  } else {  path = src + "";  }  System.out.printf("\n%d -> %d \t\t %d\t\t%s ", temp, i, distance[i], path);  printPath(parent, i);  }  System.out.println();  }  public static void main(String[] args) {  int graph[][] = { { 0, 80, 0, 10, 0, 0 }, { 0, 0, 20, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0 },  { 0, 0, 50, 0, 100, 200 }, { 5, 0, 0, 0, 0, 40 }, { 0, 0, 10, 0, 0, 0 } };  Path dj = new Path();  final int a = 0;  final int b = 1;  final int c = 2;  final int d = 3;  final int e = 4;  final int f = 5;  System.out.println("A = 0, B = 1, C = 2, D = 3, E = 4, F = 5.");  System.out.printf("Vertex\t\tDistance\tPath");  dj.dijkstra(graph, a);  }  } |

Worked with Raymond Gevorkian and Khoi Dinh on this code.

Ex-2: Given the following graph, show the shortest path using Dijkstra’s algorithm. Show the steps that you should follow to get to the final table as we did in class. Also use the program in Ex-1 and show it gives you the same answer.



Initialize from vertex A:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vertex number | Vertex name | Known | dv | pv |
| 0 | A | F | Infinity | 0 |
| 1 | B | F | Infinity | 0 |
| 2 | C | F | Infinity | 0 |
| 3 | D | F | Infinity | 0 |
| 4 | E | F | Infinity | 0 |
| 5 | F | F | Infinity | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vertex number | Vertex name | Known | dv | pv |
| 0 | A | T <Declare> | 0 | 0 |
| 1 | B | F <dv updated> | 80 | A (0) |
| 2 | C | F | Infinity | 0 |
| 3 | D | F <dv updated> | 10 | A (0) |
| 4 | E | F | Infinity | 0 |
| 5 | F | F | Infinity | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vertex number | Vertex name | Known | dv | pv |
| 0 | A | T | 0 | 0 |
| 1 | B | T <Declare> | 80 | A (0) |
| 2 | C | F <dv updated> | 80 + 20 | B (1) |
| 3 | D | F | 10 | A (0) |
| 4 | E | F | Infinity | 0 |
| 5 | F | F | Infinity | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vertex number | Vertex name | Known | dv | pv |
| 0 | A | T | 0 | 0 |
| 1 | B | T | 80 | A (0) |
| 2 | C | F <dv updated> | 10 + 50 = 60 | D (3) |
| 3 | D | T <Declare> | 10 | A (0) |
| 4 | E | F <dv updated> | 10 + 100 = 110 | D (3) |
| 5 | F | F <dv updated> | 10 + 200 = 210 | D (3) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vertex number | Vertex name | Known | dv | pv |
| 0 | A | T | 0 | 0 |
| 1 | B | T | 80 | A (0) |
| 2 | C | T <Declare> | 10 + 50 = 60 | D (3) |
| 3 | D | T | 10 | A (0) |
| 4 | E | F | 10 + 100 = 110 | D (3) |
| 5 | F | F | 10 + 200 = 210 | D (3) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vertex number | Vertex name | Known | dv | pv |
| 0 | A | T | 0 | 0 |
| 1 | B | T | 80 | A (0) |
| 2 | C | T | 10 + 50 = 60 | D (3) |
| 3 | D | T | 10 | A (0) |
| 4 | E | T <Declare> | 10 + 100 = 110 | D (3) |
| 5 | F | F <dv updated> | 10+100+40=150 | E (4) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vertex number | Vertex name | Known | dv | pv |
| 0 | A | T | 0 | 0 |
| 1 | B | T | 80 | A (0) |
| 2 | C | T | 10 + 50 = 60 | D (3) |
| 3 | D | T | 10 | A (0) |
| 4 | E | T | 10 + 100 = 110 | D (3) |
| 5 | F | T <Declare> | 10+100+40=150 | E (4) |

A 🡪 A: dv = 0, parents: 0.

A 🡪 B: dv = 80, parents: A 🡪 B.

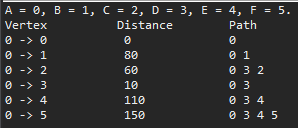
A 🡪 C: dv = 60, parents: A 🡪 D 🡪 C.

A 🡪 D: dv = 10, parents: A 🡪 D.

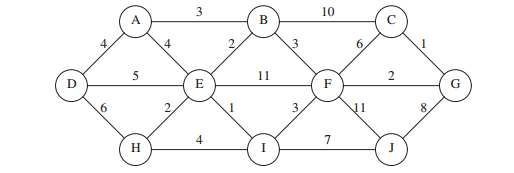
A 🡪 E: dv = 110, parents: A 🡪 D 🡪 E.

A 🡪 F: dv = 150, parents: A 🡪 D 🡪 E 🡪 F.

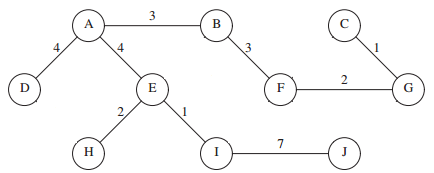
Code output from source of vertex A:



Ex-3: Find a minimum spanning tree for the graph in figure 9-84 (page 420) using prim’s algorithm. What is the total cost of all paths of this minimum spanning tree?



Minimum spanning tree:



Total cost:

A 🡪 B: 3

A 🡪 C: 3 + 3 + 2 + 1 = 9

A 🡪 D: 4

A 🡪 E: 4

A 🡪 F: 3 + 3 = 6

A 🡪 G: 3 + 3 + 2 = 8

A 🡪 H: 4 + 2 = 6

A 🡪 I: 4 + 1 = 5

A 🡪 J: 4 + 1 + 7 = 12

Total: 3 + 9 + 4 + 4 + 6 + 8 + 6 + 5 + 12 = 57.