**AMRITA SCHOOL OF ENGINEERING, AMRITAPURI**

# I SEMESTER M.Tech. CSN

**18SN601 - Practical Algorithms for Programmers**

**Programming Assignment 2**

**Due Date: November 13, 2018**

Write a program to implement the following graph algorithms:

Topological sort of a DAG

Dijkstra’s Algorithm for weighted directed graph

Kruskal’s Algorithm for weighted, connected, undirected graph

Articulation points for a connected, undirected graph

Strongly connected components of a directed graph

REQUIRED ASSUMPTIONS

1. Each graph will have 7 vertices
2. Graph vertices 1 to 7 correspond to the characters A to G, respectively
3. Vertices on an adjacency list are in the increasing order of numeric value.
4. The shortest paths found by Dijkstra’s Algorithm all begin at vertex 1 (A).
5. The graph used for testing Kruskal’s Algorithm will have 12 edges and will be connected.
6. Weights for edges are ints between 1 and 100, inclusive.

REQUIRED DATA STRUCTUREs

Adjacency list representation of a graph; nodes on an adjacency list should have fields to accommodate all algorithms implemented ( vertices u and v thst an edge connects, weight, accepted next); vertices should be in increasing order of numeric value ( v component for an edge (u,v))

Linked implementation of a queue (for Topological sort)

Table similar to the one used in Dijkstra’s algorithm

Disjoint set ADT ( for Kruskal’s algorithm)

Priority queue implemented as a heap ( array of adjacency list nodes; for Kruskal’s Algorithm)

All arrays required for finding articulation points. ( low[ ], num[ ], visited[ ], parent[ ] )

1-D array of int ( dfsnum[]; for finding strongly connected components)

1-D array of char (map[] for mapping vertices 1..7 to A..G)

REQUIRED I/O FILES

Test your program with the input file graph.dat (page 2). Your program should contain the appropriate I/O statements to read from this file. Each group of 7 lines ( from the beginning of the file) represents one graph ( for a total of 6 graphs). Each one of these seven lines provides information to construct the adjacency list for vertex v, v= 1, 2, 3, 4, 5, 6, 7. The first int on a line indicates the number of vertices adjacent to v. The remaining pairs of ints indicate an adjacent vertex ( w) and the weight of the edge (v, w).

The exceptions are graph 4 and 5 ( used to test for articulation points): each of these two graphs is described by 8 lines. The first 7 lines are as above, the 8th line contains an int indicating which vertex should be the root of the dfs tree. Thus line 29 indicates the root of graph 4’s dfs tree should be vertex3 ( C) and line 37 indicates the root of graph 5’s dfs tree should be vertex 1 (A).

Your program should contain the appropriate I/O statements to write to an output file. The name of the output file should be supplied as a command line argument. A minimally acceptable output is attached (last page 3).

**Input**

3 2 1 3 1 4 1

2 4 1 5 1

1 6 1

3 3 1 6 1 7 1

2 4 1 7 1

0

1 6 1

2 2 2 4 1

2 4 3 5 10

2 1 4 6 5

4 3 2 5 2 6 8 7 4

1 7 6

0

1 6 1

3 2 2 3 4 4 1

3 1 2 4 3 5 10

3 1 4 4 2 6 5

6 1 1 2 3 3 2 5 7 6 8 7 4

3 2 10 4 7 7 6

3 3 5 4 8 7 1

3 4 4 5 6 6 1

2 2 1 4 1

2 1 1 3 1

3 2 1 4 1 7 1

4 1 1 3 1 5 1 6 1

2 4 1 6 1

2 4 1 5 1

1 3 1

3

2 2 1 4 1

2 1 1 3 1

3 2 1 4 1 7 1

4 1 1 3 1 5 1 6 1

2 4 1 6 1

2 4 1 5 1

1. 3 1

1

1 2 1

2 3 1 6 1

1 1 1

2 6 1 7 1

1 4 1

1 5 1

**Output**

The topological sort of the first graph is:

|  |  |
| --- | --- |
| VERTEX | NUMBER |
| A | 1 |
| B | 2 |
| E | 3 |
| D | 4 |
| C | 5 |
| G | 6 |
| F | 7 |

For the second graph:

Shortest path for the first graph:

Shortest path from A to A: A (distance = 0)

Shortest path from A to B: A (distance = 2)

Shortest path from A to C: A (distance = 3)

Shortest path from A to D: A (distance = 1)

Shortest path from A to E: A (distance = 3)

Shortest path from A to F: A (distance = 6)

Shortest path from A to G: A (distance = 5)

The edges in the minimum spanning tree for the third graph are:

(A, B) (C, D) (A, D) (D, G) (F, G) (E, G)

Its cost is 16.

For the fourth graph, the articulation points are:

D

C ( root of the dfs tree)

For the fifth graph, the articulation points are:

D

C

The strongly connected components of the sixth graph are:

{ G D E } { B A C F }