Heaven's Light is Our Guide

Computer Science & Engineering

Rajshahi University of Engineering & Technology

Lab Manual

Module- 8

Course Title: Sessional based on CSE 2101

Course No. : CSE 2102

Experiment No. 8

Name of the Experiment: Graph

Duration: 1 Cycle

Background Study: Kenneth H. Rosen, "Discrete Mathematics and its Application", 6^{th} Edition: Chapter 8 (Graph)

Algorithm 1. Constructing Euler Circuits
Algorithm 2. Dijkastra's Algorithm
Algorithm 3. Floyd's Algorithm
Algorithm 4. Graph Coloring

Experiments/Problems: Write programs with these input and output.

[1] Given the vertex pairs associated to the edges of an undirected graph, find the degree of each vertex.
[2] Given the ordered pairs of vertices associated to the edges of a directed graph, determine the in-degree and out-degree of each vertex.
[3] Given the list of edges of a simple graph, determine whether the graph is bipartite.
[4] Given the vertex pairs associated to the edges of a graph, construct an adjacency matrix for the graph. (Produce a version that works when loops, multiple edges, or directed edges are present.)
[5] Given an adjacency matrix of a graph, list the edges of this graph and give the number of times each edge appears.
[6] Given the vertex pairs associated to the edges of an undirected graph and the number of times each edge appears, construct an incidence matrix for the graph.
[7] Given an incidence matrix of an undirected graph, list its edges and give the number of times each edge appears.
[8] Given a positive integer n, generate a simple graph with n vertices by producing an adjacency matrix for the graph so that all simple graphs with n vertices are equally likely to be generated.
[9] Given a positive integer n, generate a simple directed graph with n vertices by producing an adjacency matrix for the graph so that all simple directed graphs with n vertices are equally likely to be generated.
[10] Given the lists of edges of two simple graphs with no more than six vertices, determine whether the graphs are isomorphic.
[11] Given an adjacency matrix of a graph and a positive integer n, find the number of paths of length n between two vertices. (Produce a version that works for directed and undirected graphs.)
[12] *Given the list of edges of a simple graph, determine whether it is connected and find the number of connected components if it is not connected.
[13] Given the vertex pairs associated to the edges of a multi-graph, determine whether it has an Euler circuit and, if not, whether it has an Euler path. Construct an Euler path or circuit if it exists.
[14] ∗Given the ordered pairs of vertices associated to the edges of a directed multi-graph, construct an Euler path or Euler circuit, if such a path or circuit exists.
[15] ∗∗Given the list of edges of a simple graph, produce a Hamilton circuit, or determine that the graph does not have such a circuit.
[16] ∗∗Given the list of edges of a simple graph, produce a Hamilton path, or determine that the graph does not have such a path.
[17] Given the list of edges and weights of these edges of a weighted connected simple graph and two vertices in this graph,find the length of a shortest path between them using Dijkstra's algorithm. Also, find a shortest path.

[18] Given the list of edges of an undirected graph, find a coloring of this graph using the algorithm given in **Algorithm 1**.

[19] Given a list of students and the courses that they are enrolled in, construct a schedule of final exams.

[20] Given the distances between pairs of television stations and the minimum allowable distance between stations, assign frequencies to these stations.

Algorithm 1: This algorithm can be used to color a simple graph:

1. List the vertices v_1, v_2, v_3,..., v_n in order of decreasing degree so that $deg(v_1) \geq deg(v_2) \geq \cdots \geq deg(v_n)$.
2. Assign color 1 to v_1 and to the next vertex in the list not adjacent to v_1 (if one exists), and successively to each vertex in the list not adjacent to a vertex already assigned color1.
3. Then assign color2 to the first vertex in the list not already colored. Successively assign color 2 to vertices in the list that have not already been colored and are not adjacent to vertices assigned color2.
4. If uncolored vertices remain, assign color 3 to the first vertex in the list not yet colored, and use color 3 to successively color those vertices not already colored and not adjacent to vertices assigned color 3.
5. Continue this process until all vertices are colored.

Report:

Your completed work must be submitted through a LAB REPORT.

Read:

[1] Kenneth H. Rosen, "Discrete Mathematics and its Application", 7th Edition: Chapter 10 (Graph).