Due on Oct 25, 2017 in Lab

Algorithms II - Lab 4: ALGORITHMS FOR GRAPHS

Problem 1:

Write a program that can read a graph that is described by an edge list:

- read the edge list of a graph;
- compute and print the number of edges;
- print the edge list;
- compute and print the number of nodes;
- compute and print the degree of each node.

Demonstrate your code on the “Walther” graph.

Problem 2:

Write a program which can determine whether there is a path from one node to another in a graph:

- read the edge list of a graph;
- read the numeric indices of two nodes;
- determine if there is a path from one node to the other and print “path” or “no path”.

Demonstrate your code on the “Paths” graph, and the following pairs of nodes:

1. from node G (7) to node H (8);
2. from node G (7) to node I (9);

Problem 3:

Write a program which uses the brute force approach to the traveling salesman problem, and prints out the length of the shortest round trip.

- read the edge list of a graph;
- read the edge weights of a graph;
- check all possible itineraries;
- print the sequence of nodes in the shortest itinerary;
- print the total length of the shortest itinerary.

Demonstrate your code on the “TSP” graph.

Problem 4:

Write a program which uses the heuristic approach to the traveling salesman problem. The program should make $\underline{K}$ attempts. On each attempt, it should pick a starting city at random, then repeatedly move to the closest unvisited city, until all cities have been visited and the tour returns home. Keep track of the shortest tour discovered in the **K** attempts, and when done, print out the corresponding list of cities you visited.

- read the distance matrix of a graph;
- read the value K;
- carry out the heuristic K times;
- print the total length of the shortest itinerary you found.

Demonstrate your code on the "KN57" graph (57 cities!) using **K**=20 attempts.

DATA FILES

The following files will be of use to you:

- **kn57_distance_matrix.txt**, the distance matrix for the "KN57" graph;
- **next_perm.m**, computes the next permutation;
- **paths_edges.txt**, the edge list for the "Paths" graph;
- **rand_int2.m**, returns a random integer between I1 and I2;
- **rand_perm.m**, returns a random permutation;
- **tsp_edges.txt**, the edge list for the "TSP" graph;
- **tsp_edge_weights.txt**, the edge lengths for the "TSP" graph;
- **walther_edges.txt**, the edge list for the "Walther" graph;

These files will be available from the class's Blackboard site.