TA Lab Week 7 – Graphs

The goal of this lab is to work through some of the most practical and widely tackled problems in graph theory by designing and implementing efficient solutions mainly based on graph traversal algorithms.

Exercise 1 – Diameter, Radius and Center

The eccentricity *ecc(v)* of a vertex *v* in a graph G is the greatest distance from *v* to any other vertex. The radius *rad(G)* of G is the value of the smallest eccentricity. The diameter *diam(G)* of G is the value of the greatest eccentricity. The center of G is the vertex (or set of vertices) *v* such that *ecc(v)=rad(G)*. Design and implement efficient algorithms to support the following API on a directed connected graph G:

- *diameter()*: compute the maximum eccentricity of any vertex in G;
- *radius()*: compute the minimum eccentricity of any vertex in G;
- *center()*: finds the vertex (or vertices) whose eccentricity is the radius.

Hint

(diameter) Compute the transitive closure of G, then traverse the connectivity matrix to find the diameter. If G has no cycles (i.e., it is a directed tree), you can substantially improve its speed. How? *[pick a vertex s; run BFS from s; then run BFS again from the vertex that is furthest from s.]*

(center): consider vertices on the longest path.

Exercise 2 – Euler cycle

A Euler cycle in a graph is a cycle (not necessarily simple) that uses every edge in the graph exactly once. Design and implement a linear-time algorithm to determine whether a graph has an Euler cycle, and if so, find one.

Hint: Use depth-first search and piece together the cycles you discover.

Exercise 3 – Hamiltonian Circuit

Given a directed acyclic graph, design a linear-time algorithm to determine whether it has a Hamiltonian path and if so, find one. A Hamiltonian path is one that passes through every vertex exactly once.

Hint: Topological Sort

Exercise 4 – Shortest directed cycle

Given a digraph G, design an efficient algorithm to find a directed cycle with the minimum number of edges (or report that the graph is acyclic). The running time of your algorithm should be at most proportional to V(E + V) and use space proportional to E + V, where V is the number of vertices and E is the number of edges.

Hint: run BFS from each verte