22C:44 Homework 9
Due November 30, 2000

The first two problems are worth 10 points each, the last problem is worth 20 points.

1. A *bipartite graph* is a graph $G = (V, E)$ whose vertex set can be partitioned into sets V_1 and V_2 such that every edge in E is incident on a vertex in V_1 and a vertex in V_2. Use the BFS algorithm determine if a given graph is a bipartite graph or not. The resulting algorithm should run in $O(m + n)$ time for an n-vertex, m-edge graph.

2. Let $G = (V, E)$ be an arbitrary undirected graph. The *distance* between a pair of vertices $u, v \in V$, denoted $d(u, v)$, is defined as the length of a shortest path from u to v. The *eccentricity* of a vertex v, denoted $e(v)$ is the maximum distance between v and any vertex in V. A vertex v is called a *center* of G if it has minimum eccentricity (among all vertices in V).

   (a) Using the BFS algorithm devise an $O(mn + n^2)$ time algorithm to compute all the centers of an n-vertex m-edge graph.

   (b) Since an n-vertex tree has $(n - 1)$ edges, the above algorithm runs in $O(n^2)$ for n-vertex trees. However, it is possible to do much better. Devise an $O(n)$ algorithm to compute the centers of an n-vertex tree.

   Here are several "facts" that will help you get started. You do not have to prove these: (i) a tree has one or two centers (ii) the centers of a tree lie on every longest path in the tree (iii) if a longest path in a tree has even length then the tree has one center, otherwise it has two centers.

3. A *random graph* $G(n, p)$ has n vertices and some number of edges, each of which is placed in the graph with probability p.

   This is a programming exercise that requires you to experiment with random graphs. Generate 1000 instances of a graph $G(n, p)$ and for each instance determine if the graph is connected. Report the fraction of the graphs that are connected. Do this for all pairs (n, p) with $n = 1000, 2000, \ldots, 10,000$ and $p = 1/(5n)$, $p = 1/n$, $p = \ln(n)/n$, and $p = 5\ln(n)/n$. Use an adjacency list implementation for the graph and use an implementation of DFS to determine if a graph is connected. Organize your results in a 10×4 table in which each row corresponds to a value of n and each column corresponds to a value of p. Finally, comment about your results. Specifically, attempt to explain your results for fixed n and increasing p and for fixed p and increasing n.

   Be aware that generating the random graphs may take a couple of hours, so give your program enough time to run.