FALL 2007: **COT 5407** Intro. to Algorithms

[Homework 5; Due Dec 4 at start of class]

General submission guidelines and policies: Add the following statement and sign it: **I have adhered to the collaboration policy for this class and what I am presenting is my own work.** Without this statement, your homework will not be graded.

Problems

33. (**Regular**) Modify DFS or BFS to design an algorithm called CheckForODDCycle for checking whether a given connected, undirected, unweighted, simple graph $G(V, E)$ has an odd length cycle. Provide an argument why you think the algorithm is correct and analyze its time complexity.

34. (**Regular**) The adjacency list representation consists of n lists, one for each vertex. This is usually implemented as a linked list of "edge" records. Sorting one of these n lists can be done in $O(n)$ time using bucket sort (with n buckets). So it is trivial to sort all the n lists in $O(n^2)$ time. Assuming the adjacency list representation, design a linear-time ($O(m+n)$) algorithm to sort each of the n adjacency lists of a given simple undirected graph $G(V, E)$. **Hint:** Use radix sort with n buckets.

35. (**Regular**) In an undirected graph, the adjacency list representation consists of n linked lists of "edge" objects. For each undirected edge of the form $e = \{v_i, v_j\}$, the adjacency list representation contains two edge objects. One edge object is in the list $Adj[i]$ containing the source index i and the destination index j, while the other edge object is in the list $Adj[j]$ containing the source index j and the destination index i. Since an undirected edge can be thought of as being composed of two "directed" edges, we could think of one edge object as representing the edge $e = (v_i, v_j)$, with the other referring to the edge $e' = (v_j, v_i)$. We also say that the two "directed" edges $e = (v_i, v_j)$ and $e' = (v_j, v_i)$ are *partners* of each other. In general, for a given edge e, it takes $O(n)$ time to locate its partner edge object, unless we can store in each edge object a pointer to its partner edge object. Note that storing the index of a vertex in the edge object is not sufficient to locate the partner edge object. Design a linear-time ($O(m + n)$) algorithm so that each edge object contains a pointer to its partner edge object. Assume that such a field already exists in each edge record (called `PartnerEdge`), initialized to `NULL`. **Hint:** It helps to use the algorithm from the previous problem and sort each of the adjacency lists first.

36. (**Exercise**) Given a weighted undirected graph G with non-negative edge weights, if the edge weights are all increased by a positive additive constant, can the minimum spanning tree change? Can the output of Dijkstra's algorithm change for some (fixed) start vertex s? What if they are decreased by a positive constant? What if the edge

weights are all multiplied by a positive constant? Give (very) simple examples, if you claim that they can change.

37. (**Exercise**) Does Dijkstra's algorithm work correctly if some edge weights are negative? Does it work correctly if some edge weights are negative, but there are no negative weight cycles?

38. (**Extra Credit**) Problem 23.2-7, page 574.

39. (**Extra Credit**) Problem 23-3, page 577.

40. (**Regular**) Modify Floyd-Warshall's algorithm to output the number of distinct shortest paths between every pair of vertices in an unweighted undirected graph.