SPRING 2005: **COT 5993** Intro. to Algorithms

[Homework 4; Due Apr 19 at start of class]

How to write algorithmic solutions: An ideal algorithmic solutions must show **Basic Idea**, **Algorithm**, and **Time and Space Complexity Analysis**.

Reminder: Add a signed statement that you have adhered to the collaboration policy for this class and that what you are presenting is your own work.

Problems

35. (**Regular**) Given below is an algorithm for checking whether a given connected, $G(V, E)$ undirected graph has an odd length cycle. The algorithm is a simple modification of DFS and it is called as ODDCycle-Visit$(G, 1, +1)$.

ODDCycle-Visit$(G,\ u,\ b)$
Comment: DFS in graph G from vertex u
    $color[u] \leftarrow gray$
    $label[u] \leftarrow b$
    for each vertex $v \in Adj[u]$ **do**
        if $color[v] = white$ **then**
            $\pi[v] \leftarrow u$
            ODDCycle-Visit$(G,\ v,\ -b)$
        else if $label[u] = label[v]$ **then**
            Print "Odd Cycle Exists"; **Stop**
    $color[u] \leftarrow$ BLACK

(a) Analyze the time complexity of the above algorithm.

(b) Prove the following claims, which together prove the correctness of the above algorithm:

Claim 1: Prove that every node in the graph is labeled by the algorithm with labels +1 or -1.

Claim 2: If the graph has an odd cycle, then regardless of what algorithm is used for labeling the nodes (with labels +1 and -1), there must exist two adjacent vertices with the same label.

Claim 3: If $e = (u, v)$ is a **tree edge** of the DFS tree, then the algorithm makes sure that $label[u] \neq label[v]$.

Claim 4: If the above algorithm encounters an edge $e = (u, v)$ with $label[u] = label[v]$, then e is a **back edge** of the DFS tree, and this edge along with the unique path in the tree from u to v forms an odd cycle.

Claim 5: If there exists an edge $e = (u, v)$ with $label[u] = label[v]$, then the algorithm will find it.

Claim 6: If there exists an odd cycle in the graph, then the algorithm will find it.

Convince yourself that proving the above claims is enough to prove correctness of the algorithm.

36. (**Exercise**) Write down the *incidence matrix*, B, for the graph in Figure 22.1 (p528). The definition of incidence matrix is given in problem 22.1-7 (p531).

37. (**Regular**) Solve problem 22.3-1 only for the undirected case.

38. (**Regular**) 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.

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

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

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

42. (**Exercise**) Verify that the above algorithm is correct by computing the number of distinct paths between every pair of vertices for the undirected graph G described as follows. The graph G has 7 vertices numbered 0 through 6. Vertex 0 is connected to 1 and 2. Vertex 3 is connected to 1, 2, 4, and 5. Vertex 6 is connected to 4 and 5. No other edges exist in G.