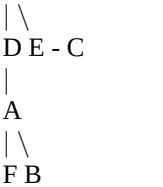
CS 344 Homework IV

Antonina Mitrofanova

April 10, 2007

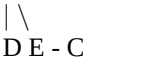
Problem 7.4
a. 10 points

b. 10 points



Problem 7.5
a. 10 points

b. 10 points



|
A
| \
F B

Problem 7.7 , 30 points
Proof by contradiction. We need to prove that in the graph after BFS, there are neither forward edges nor back edges. Let us assume that a forward/back edge exists. Let $a - b$ be a back (or equivalently, forward) edge, where a is an ancestor of b. Now, let us consider the path from a to b in the tree produced by BFS. This path will have length ≥ 2. Let the first node on the tree path descending from a to b be c. When the breadth-first search processes a, it necessarily explores all the undiscovered (adjacent) neighbors of a, that include c, as well as b. So either b is already discovered by the breadth-first search, or $a - b$ should be included as a tree edge. Thus, we reach a contradiction. As a result, this is not possible to have a forward or back edge in the graph after BFS.

Problem 7.48, 30 points One of the solutions can be as follows.

1. Perform a BFS on a graph G (starting at the arbitrary source node).
2. If the graph is Bipartite, it is possible to partition the graph into two groups of nodes A and B, so that nodes in group A will only share edges with nodes in group B, but not among themselves (same applies for nodes in group B).

   It is important to observe that if we run BFS on such a graph, then nodes from one group will all have odd distances from the source; at the same time nodes in the other groups will all have even distances from the source.
3. As a result, if the graph is bipartite, every edge of the graph should connect one even-distanced node and one odd-distanced node. No edges between even-distanced nodes (or between odd-distanced nodes) are allowed.
4. After running BSF, we check for every edge - whether it connects an even and odd nodes. If we find at least one edge connecting two even-distanced nodes or two odd-distanced nodes, the graph is NOT bipartite.

   You can perform the same idea with assigning different colors to nodes - say blue and red (similar to even and odd distance calculations).

   The time complexity in both cases is $O(V + E)$.