EE360C: Algorithms
The University of Texas at Austin
Dr. David Soloveichik

Name: **EID:**

Exam #1 Review: DFS/BFS (Recitation)

Problem 1: Finding bridges

Give an $O(m * (n + m))$ time algorithm to find all bridge edges in an undirected connected graph with n nodes and m edges. Recall that a bridge of a connected graph is an edge whose removal disconnects the graph. Briefly justify the running time of your algorithm.

Solution

First, note that since the graph is undirected and connected, all nodes are reachable from any node; thus the size of the set returned by DFS/BFS is n. For each edge (u, v), let G' be the graph with (u, v) removed. Run DFS/BFS from an arbitrary node on G'. Check if the size of the set returned by the DFS/BFS run is n; if not, (u, v) was a bridge (it disconnects the graph upon removal), so mark (u, v) as a bridge.

BFS/DFS (which takes $\mathcal{O}(n + m)$ time) is run once for each edge (of which there are m) for a total time complexity of $\mathcal{O}(m \cdot (n + m))$.

Problem 2: Bridges again

Suppose G is a connected undirected graph. An edge whose removal disconnects the graph is called a *bridge*. Either prove the following statement or provide a counter-example: every bridge e must be an edge in any depth-first search tree of G and any breath first search tree of G.

Solution

Since $e = (u, v)$ is a bridge, we can consider two partitions of the nodes, P_1 and P_2, in which P_1 is connected, P_2 is connected, and (u, v) is the only edge between a node in P_1 and a node in P_2. Let u be in P_1 and v be in P_2. Consider any run of DFS/BFS starting at (without loss of generality) an arbitrary node w in P_1. Since P_1 is connected, u is reachable from w. Thus, DFS/BFS(u) will execute. When u's edges are checked, (u, v) will be checked. At this point v cannot be explored since we started in P_1, v is in P_2, and (u, v) is the only edge from any node in P_1 to any node in P_2. Since v is not explored when we execute DFS/BFS(u), then (u, v) will be added to the tree; thus the bridge $e = (u, v)$ must be in any DFS/BFS tree.

Problem 3: Good vs. evil

Suppose there are only two types of professional wrestlers: good guys and bad guys. Between any pair of professional wrestlers, there may or may not be a rivalry. Suppose we have n professional wrestlers and we have a list of r pairs of wrestlers for which there are rivalries. Give an $O(n + r)$ algorithm that determines whether it is possible to designate some of the wrestlers as good guys and the remainder as bad guys such that each rivalry is between a good guy and a bad guy. If it is possible to perform such a designation, your algorithm should produce it.

Solution

First, note that the wrestlers can be viewed as nodes, and the rivalries as edges. The question is then the following: give an algorithm which determines if the graph is two-colorable (i.e., find if the graph is bipartite) and if so, give a valid two-coloring (i.e., give the two partitions which must exist if the graph is bipartite).

The algorithm follows: Run BFS on the graph in $\mathcal{O}(n + r)$ time. Call nodes (wrestlers) in level i of the BFS tree "good" if i is even and "bad" if i is odd. Then, check each edge in the graph and ensure that the two nodes of the edge are have different good/bad assignments; if not, return that the graph is not two-colorable (there is no such good/bad designation possible). If all edges are valid (one node is good, one node is bad) then return the good/bad designations.

To see why this algorithm works: if the algorithm does not produce a valid good/bad designation, then there is at least one pair of nodes u, v with an edge (u, v) in the graph, with u and v given the same good/bad designation. Since u and v have the same good/bad designation, their levels in the BFS tree have the same parity (even/odd). Thus, the edge (u, v) is not in the BFS tree (else their levels would have different parity). The path p from u to v in the BFS tree is of even length (if it were odd length, their levels would have different parity). Then the path p plus the edge (u, v) makes a cycle of odd length. From the homework, we know that a graph is two-colorable if and only if it has no odd length cycles. Thus, if the algorithm failed to give a valid good/bad designation, then it is not possible to two-color the graph and thus not possible to give a valid good/bad designation, and we can return "no designation possible". Otherwise, if the algorithm did not fail, then we return the valid good/bad designation.