CS170 Final Project

Walter Wu

April 29, 2016

Algorithms

1. **Finding All Cycles in a Graph of Length 5 or Less**
   Main Idea. We will execute a DFS on the input graph, assigning pre-visit values to each node visited, such that earlier nodes have smaller pre-visit values. The root node is chosen arbitrarily, and is assigned a pre-visit value of 0. All nodes $v \in V$ except for the root node are initally labeled as $visited(v) = False$, and have a pre-visit value of 9999. This magic number is simply to ensure that the pre-visit number is higher than all visited nodes. During DFS, when we reach a node v, we will first mark $visited(v) = True$, and set $pre - visit(v)$ equal to the number. We then scan its all edges $(v, u) \in E, u \in V$. If $visited(u) = True$, then there must exist a cycle that includes both u and v. We find this cycle by backtracking through our DFS tree from v for at most four nodes. We return a cycle if in those four nodes we reach u. We repeat the DFS search for each node in the graph.

   Proof of correctness. We realize that a cycle can only occur if there is a back edge in the DFS tree. A back edge is defined in this case to be any edge $e = (u, v)$, where $pre - visit(v) > pre - visit(u)$. This is because a DFS will explore as far as it can along a branch, such that$pre - visit$ numbers increase the farther it searches. And edge to a lower $pre - visit$ number indicates that there is a cycle. However, we realize that it is possible that there may be no cycle, as v may not be reachable from u, but was simply visited after u in the DFS. We can find the cycle simply by backtracking along our DFS, stopping after visiting four nodes due to the limitation of our cycle size. This will always find a cycle unless there is both a foward and back edge between two nodes. This is because the DFS will not search through an already visited node. This is countered by running the DFS search on every single node as a root node, so that we can find every cycle.

   Runtime and Justification. The runtime of this algorithm is $O(|V|^3 + |E||V|)$ time. For each DFS search, at every node there can be a maximum of $|V| - 1$ edges we need to check. However, finding a cycle takes constant time for each possible cycle found. We run the DFS $|V|$ times.