University of Waterloo
CS 466 — Advanced Algorithm
Spring 2013
Problem Set 5
Siwei Yang - 20258568

1. [4 marks: Vertex Cover for Trees] A tree is a connected graph with no cycles. As a tree has no cycles, it is bipartite (as it has no odd length cycles) and hence the polynomial time algorithm we saw in the class, to find the minimum vertex cover applies to trees.

   The purpose of this problem is to find a much simpler polynomial time algorithm to find a minimum vertex cover in a tree. Recall that a vertex cover in a graph is a subset S of vertices such that for every edge in the graph at least one of its end points is in S.

   Give a polynomial time algorithm (different from the one we saw for bipartite graphs) to find a minimum vertex cover in a tree.

   Note: The algorithm for this problem is iterative in nature.

   First Preprocess the graph to build a tree structure by creating nodes and go over the edges to connect them with each other to form a tree.

   Second Scan all nodes to find a list of leaf nodes. Create an empty list for covering vertices.

   Third Delete an arbitrary leaf node, a, and its parent, b. Save b into the covering vertices list. Check b's parent, c, for siblings. If no other sibling exists, add c into the leaf node list as it is a leaf node in the updated tree.

   Forth Repeat the third step until exhausted the leaf node list or all nodes in the list have no parents(depending on the node processing order, the tree might be broken into multiple components). Now we have minimum vertex cover saved in the other list.

2. [6 marks: Tournament] A tournament is a directed graph with exactly one directed edge between every pair of vertices. i.e. take a complete undirected graph and give arbitrary orientation to each edge, what you get is a tournament. A feedback vertex set in a tournament T is a subset S of vertices such that $T \setminus S$ (T with vertices of S removed) has no directed cycles.

   - [3 marks] Prove that a tournament has a directed cycle if and only if it has a directed cycle of length 3.

     Note: Using mathematical induction on the length of cycles.

     Let the base case be: If a tournament has a directed cycle of length 3, then it has a directed cycle of length 3. This is trivially proven by logic.

     Let the induction be: Assume a tournament has a directed cycle, C, of length k + 1, then it has at least three connectted vertices, a, b, c. Without loss of generality, assume $a \to b$ and $b \to c$. Then, given there is an edge between a and c,

     - [if $a \leftarrow c$] a, b, c form a directed cycle of length 3
     - [if $a \to c$] a, c and rest of vertices of C form a directed cycle of length k

     Special case: If a tournament has a directed cycle of length 2, then it's not a tournament any more. Because the cycle adds two edges between two vertices which contradicts with the tournament definition.

     Thus, **a tournament has a directed cycle only if it has a directed cycle of length 3**. And, **a tournament has a directed cycle if it has a directed cycle of length 3**, which is trivially proven by logic.

   - [3 marks] Use Part (a) to give a $O(3^k n^c)$ algorithm to determine whether a given tournament on n vertices has a feedback vertex set of size at most k. Here c is some constant independent of k. Give an estimate for your c.

     Note: The algorithm for this problem is recursive in nature.

     First Preprocess the graph to find a directed cycle of length 3. One naive approach is to choose three arbitrary vertices from the graph and check is they form a directed cycle. If there isn't

any directed cycle of length 3, then report success since this graph is cycle free. Otherwise proceed to the second step. This process takes $O(n^3)$ time.

Second If the number of vertices in the graph is already k less than the original graph, report failure. Otherwise, iterate on the three vertices from the identified cycle, send the graph without one of the vertices to the first step again. If any of the three attempts succeeded, then report success since the original graph has a feedback vertex set of size at most k. If none of them succeeded, report failure.

The algorithm has a recurrance on number of vertices left in the graph:

$$T(n) = 3 * T(n-1) \tag{1}$$

And, we know there will be at most k vertices removed from the graph, thus the inferred runtime is $O(3^k n^3)$.