ESO207A: Data Structures and Algorithms

Practice Set *4a: Basic Graph Algorithms* –

Problem 1. You are given a tree $T = (V, E)$ in adjacency list format along with a designated root vertex $r \in V$. Recall that u is said to be an *ancestor* of v in T if u lies on the unique simple path from the root to v.

We would like to answer queries of the form "is u an ancestor of v" in *constant* time. To do this, you are allowed to pre-process the tree in linear time. Can you think of an algorithm for pre-processing and for answering queries.

Problem 2. You are given a tree $T = (V, E)$ along with a designated root vertex $r \in V$. This then uniquely defines a rooted, oriented tree. The parent of any node $v \neq r$ is denoted as $p(v)$ and is the node adjacent to v in the unique path from r to v. By convention, set $p(r) = r$. Define $p^1(v) = p(v)$. The grandparent of v is defined as $p(p(v))$ and denoted as $p^2(v)$. Similarly, $p^k(v)$ is defined as $p(p^{k-1}(v))$ and is the kth ancestor of v.

Associated with each vertex in the tree is a non-negative integer label $l(v)$. You have to update the labels of all the vertices in T according to the following rule: $l_{new}(v) = l(p^{l(v)}(v))$.

Problem 3. This problem concerns strongly connected directed graphs and is the counterpart to bipartiteness of undirected graphs discussed in class.

1. Prove that a strongly connected directed graph is bipartite only if it has no odd cycles.
2. Suppose that BFS of G starting from any vertex does not yield any edge whose end points have the same parity (odd/even) of the index of level sets. (i.e., there is no edge (u, v) such that u is in L_i, v is in L_j and i and j are either both odd or both even. Show that in this case, G is bipartite.
3. Conversely, suppose that BFS on G yields an edge (u, v) with both u and v having the same level set index parity. Now construct an odd cycle in G involving at least one of u or v (or both). Hence we have shown that a directed graph is bipartite iff it has no odd cycles (argue).
4. Give a linear time algorithm to test if a directed graph has an odd cycle. Find one such cycle if there is one.