CS 473: Fundamental Algorithms, Spring 2011

HW 1

Homework is due by **Monday, 23:59:59, January 31**

Problem 1 is due by **Sunday, 23:59:59, January 30**

This homework contains four problems. **Read the instructions for submitting homework on the course webpage**.

Collaboration Policy: For this homework, Problems 2–4 can be worked in groups of up to three students.

Problem 1 should be answered in Compass as part of the assessment HW1-Online and should be done individually.

---

1. (30 pts) Short questions to be answered on compass individually.

2. (10 pts) Present an algorithm that takes an undirected graph, and directs each one of the edges of the graph such that the resulting directed graph is a DAG. What is the running time of your algorithm?

3. (40 pts) For a DAG G let $\mathrm{lp}(v, G)$ denote the longest directed path in G that starts in v (if there are several such paths, we arbitrarily choose one of them). Here, the length of a path is the number of edges in it. The *influence* of a vertex v in G, denoted by $i(v)$, is the number of edges of $\mathrm{lp}(v, G)$. Two vertices x and y in a DAG G are *unrelated* if there is no directed path from x to y, and no directed path from y to x in G.
   (a) (5 pts) Prove, that if the edge (x, y) is in G then $i(x) > i(y)$.
   (b) (5 pts) Prove, that if there is a path between x and y in G then $i(x) > i(y)$.
   (c) (5 pts) Conclude, that if there are k vertices in G that have all the same influence then they are all unrelated.
   (d) (20 pts) Present a linear time algorithm that computes for each vertex in G its influence. Prove the correctness of your algorithm.
   (e) (5 pts) Prove, that in a DAG there is a path of length $\lfloor \sqrt{n} \rfloor$, or there are $\lfloor \sqrt{n} \rfloor$ vertices which are all unrelated to each other.
   Present an algorithm that outputs this path, or the set of unrelated vertices. How fast is your algorithm?

4. Ancestor and LCA queries.
   (20 pts.)

   For a node v in a rooted tree, we denote by $\mathrm{depth}(v)$ the distance of vertex v to the root of T. Recall that u is an *ancestor* of v in the rooted tree, if the path from the root to v in T passes through u. The *least common ancestor* of two nodes x and y in a rooted tree, denoted by $\mathrm{lca}(x, y)$, is the lowest node in the tree that is an ancestor of both x and y.
   (a) (10 pts) You are given a binary tree $T = (V, E)$, along with a designated root node $r \in V$. You wish to preprocess the tree so that queries of the form "is u an ancestor of v?" can be answered in constant time. The preprocessing itself should take linear time. How can this be done?

(b) (5 pts) You are given a binary tree T, and you would like to preprocess it in linear time, such that given a query made out of three nodes x, y and z, the algorithm can in constant time decide if z is the least common ancestor (i.e., LCA) of x and y.

(c) (5 pts) Show how to preprocess a given binary tree T, such that given a query made out of two vertices x and y, it computes in $O(\text{depth}(\text{lca}(x, y)))$ the LCA of x and y.