15-750 Graduate Algorithms

Homework # 2 **due: February 13**

Groundrules:

- Homeworks will generally consist of easier *exercises* and harder *problems*. You should do the exercises by yourself, but you may work with a friend or two on the problems if you want — just write down who you worked with. One exception: no fair working with someone who has already figured out (or already knows) the answer. Of course, your writeup should be in your own words.
- If you've seen a problem before (sometimes we will give problems that are "famous"), then say that in your solution (it won't affect your score, we just want to know). Also, if you use any sources other than the book or those on the web page, write that down too. It's fine to look up a complicated sum or inequality, but don't look up an entire solution.

Exercises:

1. Given a rooted tree T, it defines an ancestor relation among the nodes in the tree: Node i is an ancestor of node j if i is on the path from j to the root of the tree. Give a labeling of the nodes in the tree such that each label uses $O(1)$ space, and given a pair of labels, it can be determined in $O(1)$ time if one of them is an ancestor of the other. (Note that we're not counting bits, so that a pointer and an integer of magnitude up to n takes O(1) space.)
2. Give a simple algorithm that computes the minimum spanning tree of a graph G (with n vertices) in $O(n^2)$ time. Don't make use of heaps or other sophisticated data structures.
3. The static optimality theorem tells us that on any sequence of accesses, splay trees are at most a constant factor worse than the best static tree in hindsight for that sequence. Your job is to give an algorithm to compute this optimal static tree. Specifically, given a sequence of counts $c_1, c_2, \ldots, c_n$ (c_i is the number of times element i is accessed), your algorithm should compute the optimal binary search tree in time $O(n^3)$. Hint: think dynamic programming.

Problems:

4. **Continuing with Ex (3)**
   (a) Prove or disprove: if the optimal static tree for elements $i, \ldots, j$ has root r_1, and the optimal static tree for elements $i+1, \ldots, j+1$ has root r_2, then an optimal static tree for elements $i, \ldots, j+1$ has root $r \in [r_1, r_2]$.
   (b) Use your insights from (a) to give an $O(n^2)$-time algorithm for the problem in exercise 3.

5. **Cost of Splaying**

   Let the vertices of an n node graph G be labeled $1, 2, \ldots, n$. The graph G has degree bounded by some constant Δ. This graph induces a distance metric among the nodes in the obvious way: the distance between two nodes i and j, $d(i,j)$, is simply the number of edges on the shortest path from i to j in G. Let s be some distinguished node of G.

   Consider a splay tree that stores the items $1, 2, \ldots, n$. Consider a sequence of accesses σ. Let σ_i be the ith item accessed. Assume that σ contains at least one access to each item. Let $C(\sigma, T)$ be the cost of accessing the sequence σ from some initial tree T.

   (a) If $\Delta = 2$ prove:

   $$C(\sigma, T) = O\left(\sum_i \log(2 + d(s, \sigma_i))\right)$$

   (b) For arbitrary fixed Δ prove:

   $$C(\sigma, T) = O\left(\sum_i (1 + d(s, \sigma_i))\right)$$

   Note: Please make your proofs rigorous, even though they involve only proving a big-oh bound.

6. **Efficienty Representating a Tree Metric**

   Given an arbitrary unrooted tree T with n nodes, the tree induces a distance metric among the nodes of the tree. (Just as the graph G did in problem 5.) Given two nodes i and j the distance $d(i,j)$ between them is simply the number of edges on the path in the tree from i to j. Show how this metric can be represented in such a way that (1) the total space used is $O(n)$ and (2) the time to compute $d(i,j)$ is $O(\log n)$. (Note that we're not counting bits, so that a pointer and an integer of magnitude up to n takes O(1) space.)