# ICS 261—Sample solutions on LCAs and augmented tree structures

David Eppstein, ICS, UC Irvine

March 4, 1999

1. Suppose that instead of least common ancestor queries in a tree, we only need to answer ancestor-descendant queries: is node x an ancestor of node y? Describe a simple data structure for answering these queries in constant time.

   Solution: Store the preorder and postorder number of each node. Node $x$ is an ancestor of node $y$ if and only if $\text{preorder}(x) < \text{preorder}(y)$ and $\text{postorder}(x) > \text{postorder}(y)$.

2. Suppose we wish to maintain a tree, subject to cutting and linking operations, and answer LCA queries: given two vertices as input, output their lowest common ancestor in the current tree. Describe how to modify Tarjan's cutting and linking trees data structure to solve this problem in $O(\log n)$ amortized time per update or query.

   Solution: to find the LCA of $x$ and $y$, first expose $x$, then expose $y$. The LCA is the point at which the second expose makes its last join operation. If the second expose doesn't perform any splits and joins, $y$ is an ancestor of $x$.

3. Recall that the *standard deviation* of a set of $n$ numbers $x_1, x_2, \ldots x_n$ is defined as

$$\sqrt{\frac{\sum x_i^2 - (\sum x_i)^2/n}{n-1}}.$$

   (a) Show how to preprocess an array of $n$ items $A[0], \ldots A[n-1]$ in linear time, so that one can compute the standard deviation of any contiguous subarray in constant time.

   Solution: Store arrays $B[i]$ (the sum of the first $i$ values of $A$) and $C[i]$ (the sum of the first $i$ squared values). The variance of the subarray from $A[i]$ to $A[j-1]$ is

$$\sqrt{\frac{(C[j]-C[i])-(B[j]-B[i])^2/(j-i)}{j-i-1}}.$$

   (b) Describe how to maintain an array of $n$ items, subject to updates that change the value of a single item, and answer queries that request the standard deviation of a contigous subarray, in time $O(\log n)$ per update or query.

   Solution: Use a complete binary tree, having the array values at its leaves, where each internal node stores both the sum of the values of its descendants and the sum of the squared values. Any changed value causes a constant amount of work updating each of $O(\log n)$ internal nodes, and the sums needed to calculate any subarray's standard deviation can be computed by combining the values from $O(\log n)$ internal nodes.