CS 558: Data Structures and Algorithms II
Winter 2002-03, Homework 2

1. (20 Pts.) Consider the list-based (fast-find) method for disjoint set union, as compared to the tree-based (fast-union) method, with path compression. Consider any intermixed sequences of Union and Finds. Suppose both methods are used to perform the sequence concurrently, and that they both do the links in the same way, in the sense that throughout the process the canonical elements are the same for both methods. Prove that the list-based method makes at least as many changes of canonical-element pointers as the tree-based method makes of parent pointers.

2. (25 Pts.) Prove that the tree-based disjoint set union with compression but without linking by rank takes \(O(m \log n)\) time to perform \(m\) Finds intermixed with Unions, if \(n\) is the number of initial singleton sets and \(m \geq n\).

3. (25 Pts.) As severe budget cuts hit the DEPTA, the authority is planning to phase out several stops along the Red line. To compensate for the inconvenience, the DEPTA is planning a user friendly 1-800-service that you can call at any time to find if your favorite subway stop is still in service, and if not, the names of the two stops on either side of your favorite stop. However, they would like your help in designing the data structures underlying this state-of-the-art service. Below is a formal description of the task. Assume that the original locations of the stops of the Red line were numbered 1 to \(n\) and all stops are active initially. Your data structure should support the following two operations:

- **Delete**\((i)\): deactivates the subway stop at location \(i\).
- **NearBy**\((i)\): returns a pair \((j, k)\) with \(j \leq i \leq k\) such that \(j\) is the largest numbered active subway stop satisfying \(j \leq i\) and \(k\) is the smallest numbered active subway stop with \(i \leq k\).

Your data structure may use \(O(n)\) space. Your goal is to minimize the total time taken to execute a sequence of \(D\) Delete operations and \(N\) Nearby queries (interleaved arbitrarily), with \(n, D \ll N\) (say \(n, D \leq N/\sqrt{n}\)). Express your running time as a function of \(n, N,\) and \(D\).
