Ph.D. Qualifying Exam
Design and Analysis of Algorithms and Data Structures

Computer Science Department
Indiana University

Fall 1998

This exam has two problems.

Problem I

This question requires you to describe an algorithm for measuring the similarity of two unrooted, leaf-labeled, binary trees. Each tree has n leaves. The leaves of a single tree are labeled with distinct integers in the range 1 to n. Your algorithm (see below) should be reasonably efficient for large n.

An unrooted binary tree consists of nodes with one neighbor (leaves) and nodes with three neighbors (internal nodes). The tree is called binary because rooted binary trees also contain nodes with three neighbors (two children and a parent). The total number of nodes in an n-leaf unrooted binary tree is $2n-2$. In the input for this problem, an unrooted binary trees is represented as a rooted tree. For each node, i with $1 \leq i \leq 2n-2$, the pointer to the parent is given (it is represented as an integer in the range 1 to $2n-2$). The root has an empty pointer (represented by zero). The leaves are the nodes with $1 \leq i \leq n$, where the leaf with label j has index j.

Example. The tree

$$
\begin{array}{ccccc}
1 & & & & 2 \\
& \backslash & & / & \\
& & 5 - 6 & & \\
& / & & \backslash & \\
3 & & & & 4
\end{array}
$$

has as one of its representations (the one rooted on node 5) 5, 6, 5, 6, 0, 5.

Removing an edge in the unrooted binary tree divides the leaves into two subsets. The edge is uniquely identified by its associated subsets. Thus, in the above example, the edge between nodes 5 and 6 is associated with the subsets (1,3) and (2,4).

Given the pair of trees A, B with labeled leaves, the errors of type 1 are the edges in tree A that are not in tree B. The errors of type 2 are the edges in tree B that are not in tree A.

Example. Suppose tree A is as above, and tree B is

```
1       3
 \     /
  6 – 5
 /     \
2       4 .
```

Then there is one error of type 1 (because edge (1,3) (2,4) is the only edge in tree A that is not in tree B) and one error of type 2.

Design a reasonably efficient algorithm to count errors of each type. You should have enough detail that it is clear how to program the algorithm and that it is clear what the running time is. It is suggested that you give the key ideas behind your algorithms before you give any details. To a large extent, a good list of key ideas will make it clear which algorithm you have in mind.

You should be willing to use techniques in your algorithms that perform fast in practice (such as hash tables) when it is appropriate to do so.

A worst case time of n^3 or n^4 is satisfactory, n^6 is disappointingly slow, 2^n is unsatisfactory. A time of n^2 or better is either excellent or impossible.

Problem II

Given a set A of n distinct numbers and a number i, with $1 \leq i \leq n$, the following algorithm, called PICK, finds the ith smallest element x in A, i.e., it finds the element $x \in A$ that is larger than exactly $i-1$ other elements of A.

1. Divide the n elements of the input array into $\lfloor n/9 \rfloor$ groups of 9 elements each and at most one group made up of the remaining $n \bmod 9$ elements.

2. Find the median of each of the $\lceil n/9 \rceil$ groups by insertion sorting the elements of each group (of which there are 9 at most) and then picking the median from the sorted list of group elements.

3. Use PICK recursively to find the median x of the $\lceil n/9 \rceil$ medians found in step 2.

4. Partition the input array around the median-of-medians x. Let k be the number of elements on the low side of the partition, so $n-k$ is the number of elements on the high side.

5. Use PICK recursively to find the ith smallest element on the low side if $i \leq k$, or the $(i-k)$th smallest element on the high side if $i > k$.

Your task is to analyze the running time of the algorithm, where running time is expressed as the worst case number of element to element comparisons only.