Problem 1 (CLRS 12.1-3). (3 points) Give a non-recursive algorithm that performs an in-order traversal of a binary tree.

(a) An easy solution uses a stack as an auxiliary data structure.

(b) A more complicated, but elegant, solution uses no stack but assumes that we can test two pointers for equality. Getting familiar with threaded binary trees is a good start.

Problem 2 (CLRS 12.3-1). (1 point) Give a recursive version of the Tree-Insert procedure.

Problem 3 (CLRS 12.3). (4 points) In this problem, we prove that the average depth of a node in a randomly built binary search tree with \( n \) nodes is \( O(\lg n) \). Although this result is weaker than that of Theorem 12.4 in CLRS, the technique we shall use reveals a surprising similarity between the building of a binary search tree and the execution of Randomized-Quicksort from Section 7.3.

We define the total path length \( P(T) \) of a binary tree \( T \) as the sum, over all nodes \( x \) in \( T \), of the depth of node \( x \), which we denote by \( d(x, T) \).

(a) Argue that the average depth of a node in \( T \) is

\[
\frac{1}{n} \sum_{x \in T} d(x, T) = \frac{1}{n} P(T).
\]

Thus, we wish to show that the expected value of \( P(T) \) is \( O(n \lg n) \).

(b) Let \( T_L \) and \( T_R \) denote the left and right subtrees of tree \( T \), respectively. Argue that if \( T \) has \( n \) nodes, then

\[
P(T) = P(T_L) + P(T_R) + n - 1.
\]

(c) Let \( P(n) \) denote the average total path length of a randomly built binary search tree with \( n \) nodes. Show that

\[
P(n) = \frac{1}{n} \sum_{i=0}^{n-1} (P(i) + P(n - i - 1) + n - 1).
\]

(d) Show how to rewrite \( P(n) \) as

\[
P(n) = \frac{2}{n} \sum_{k=1}^{n-1} P(k) + \Theta(n).
\]
Recalling the alternative analysis of the randomized version of quicksort given in Problem 7-3 in CLRS, conclude that \( P(n) = O(n \lg n) \).

At each recursive invocation of quicksort, we choose a random pivot element to partition the set of elements being sorted. Each node of a binary search tree partitions the set of elements that fall into the subtree rooted at that node.

Describe an implementation of quicksort in which the comparisons to sort a set of elements are exactly the same as the comparisons to insert the elements into a binary search tree. (The order in which comparisons are made may differ, but the same comparisons must occur.)

**Problem 4 (CLRS 13.1-5).** (1 point) Show that the longest simple path from a node \( x \) in a red-black tree to a descendant leaf has length at most twice that of the shortest simple path from node \( x \) to a descendant leaf.

**Problem 5 (CLRS 13.1-6).** (1 point) What is the largest possible number of internal nodes in a red-black tree with black-height \( k \)? What is the smallest possible number?

**Problem 6 (CLRS 13.1-7).** (1 point) Describe a red-black tree on \( n \) keys that realizes the largest possible ratio of red internal nodes to black internal nodes. What is this ratio? What tree has the smallest possible ratio, and what is the ratio?