1. Let \( T \) be a random binary search tree that stores the keys \( 1, \ldots, n \) and, for each \( i \in \{1, \ldots, n\} \), let \( v_i \) the node of \( T \) that stores the key \( i \).

(a) What is the probability that \( v_1 \) is a leaf?

(b) Fix some \( i \in \{2, \ldots, n - 1\} \). What is the probability that \( v_i \) is a leaf in \( T \)? (Hint: The answer doesn’t depend on \( i \).)

(c) What is the expected number of nodes in \( T \) that are leaves?

(d) What is the expected number of nodes in \( T \) that have exactly one child?

(e) What is the expected number of nodes in \( T \) that have exactly two children?

2. Let \( T_1 \) and \( T_2 \) be two binary search trees that each contain the keys the elements \( 1, \ldots, n \). Let \( d_T(i) \) denote the depth (distance from the root) of element \( i \) in tree \( T \).

(a) Show that there exists a ternary (3-ary) search tree \( T_3 \) such that, for every \( j \in \{1, \ldots, n\} \),
\[
    d_{T_3}(j) \leq \min\{d_{T_1}(j), d_{T_2}(j)\}
\]
(Hint: The standard algorithm for deleting a value in a binary search tree does not increase the depth of any node.)

(b) Prove that the converse of the above statement is not true. That is, there exists a ternary search tree \( T_3 \) containing the elements \( 1, \ldots, n \) such that no pair of binary search trees \( T_1 \) and \( T_2 \) has the property that
\[
    \min\{d_{T_1}(j), d_{T_2}(j)\} \leq d_{T_3}(j)
\]
for all \( j \in \{1, \ldots, n\} \). (Hint: In a perfectly balanced ternary tree, all nodes have depth at most \( \lceil \log_3 n \rceil \).)

3. This question is about doing iterated search using biased search trees (instead of fractional cascading). Consider any increasing sequence \( x_0 = -\infty, x_1, \ldots, x_k, x_{k+1} = \infty \) of numbers and let \( I_i, 0 \leq i \leq k \), denote the interval \([x_i, x_{i+1})\). Let \( W_i, 0 \leq i \leq k \), be an arbitrary positive weight associated with \( I_i \) and let \( W = \sum_{i=0}^k W_i \). A biased search tree is a binary search tree built on \( x_1, \ldots, x_k \) in such a way that, given any number \( x \), we can determine the interval \( I_i \) containing \( x \) in \( O(1) + \log(W/W_i) \) time.

(a) Suppose you have two lists \( A \) and \( B \) containing a total of \( n \) numbers. Show how to use a biased search tree on the elements of \( A \) so that, using this search tree, we can locate any element \( x \) in both \( A \) and \( B \) using \( O(1) + \log n \) comparisons. (Hint: \( \log(W/W_i) = \log W - \log W_i \))

(b) Generalize the above construction so that, given lists \( A_1, \ldots, A_r \) containing a total of \( n \) numbers, we can locate any element \( x \) in \( A_1, \ldots, A_r \) using a total of \( O(r) + \log n \) comparisons.

---

1 In a ternary search tree each node contains up to 2 keys \( a \) and \( b \) with \( a < b \) and these are used to determine whether a search for \( x \) search proceeds to the left (\( x < a \)), middle (\( a < x < b \)) or right (\( x > b \)) child.
4. This question is about an application of persistence. Recall that persistent binary search trees take \( O(\log n) \) time per insert/delete/search operation and require \( O(1) \) extra space per insert/delete operation.

Let \( S := \{(x_i, y_i, z_i) : i \in [1, \ldots, n]\} \) be a set of points in \( \mathbb{R}^3 \). We want to design a data structure that accepts a query \((m, z)\).

Design a data structure of size \( O(n) \) that preprocesses \( S \) so that you can quickly answer a query of the form \((m, q)\) that returns \( \min\{z > q : (x, y, z) \in S \text{ and } y > mx\} \). In words, we look at all the points in \( S \) whose projection onto \( xy \)-plane is above the line \( y = mx \) and, among those we find the one whose \( z \)-coordinate is closest to (but bigger than) \( q \).

5. This question is about another application of persistence.

Suppose we are given an array \( x_1, \ldots, x_n \) of (not necessarily sorted) numbers. We want to construct a data structure that supports “range location queries.” Given a query \((a, b, x)\), find the smallest value \( x' \in \{x_a, \ldots, x_b, \infty\} \) that is greater than or equal to \( x \). Describe a data structure of size \( O(n \log n) \) that supports range location queries in \( O(\log n) \) time. (Hint: A range location query \((a, b, x)\) can be answered if we have two binary search trees, one that stores \( x_a, \ldots, x_c \) and one that stores \( x_{c+1}, \ldots, x_b \) for some \( c \in [a, \ldots, b].\))