1. Let $S$ be a set of $n$ points drawn uniformly and independently at random from a unit square (i.e., the $x$ and $y$ coordinates of each point are drawn at uniformly at random from $[0,1]$. We say that a point $(x_1,y_1)$ dominates a point $(x_2,y_2)$ if $x_1 > x_2$ and $y_1 > y_2$. A point of $S$ is a maximal element if there is no other point of $S$ that dominates it. What is the expected number of maximal elements in $S$? Why? (Hint: Your answer should be two sentences long.)

2. Suppose you are given a sorted array of $n$ keys $k_1,\ldots,k_n$. Show how make a random treap on $k_1,\ldots,k_n$ in $O(n)$ worst-case time. The resulting treap should be a random treap: each key should be assigned a random priority independently and uniformly of other nodes.

3. Let $T_1$ and $T_2$ be two binary search trees whose nodes contain 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 $\lceil \log_3 n \rceil$.)

4. 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 $[x_i,x_{i+1}+1)$. 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 $O(1) + \log(W/W_i)$ comparisons. (Hint: $\log(W/W_i) = \log W - \log W_i$.)

---

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.
(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.

5. 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.

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 \(c \in \{a, \ldots, b\}\).)

6. We say that a point \((x_1, y_1)\) dominates a point \((x_2, y_2)\) if \(x_1 \geq x_2\) and \(y_1 \geq y_2\). Let \(S\) be a set of \(n\) points in the plane. A dominance query on \(S\) consists of a query point \(q\) and returns the subset of \(S\) that dominates \(q\).

Describe and analyze a preprocessing algorithm, data structure and query algorithm that can answer dominance queries on the set \(S\). Your algorithms should be as efficient as possible both in terms of running time and memory requirements. You may use any data structuring technique we have described in class.

7. The persistent graph structure described in class, and in the notes, uses tables with \(d + 1\) rows. The amortization argument shows that \(n_1\) calls to MakeNode() and \(n_2\) calls to ChangeLabel/ChangeEdge results in the creation of at most \(2n_1 + n_2\) tables. This is still too much. Describe and analyze a modification of this scheme where, for any integer \(k\), there are at most \(n_1 + (n_1 + n_2)/k\) tables created (though the tables may have a few more rows than before).