1. For the persistent search tree depicted below, with nodes showing key discriminators and links labeled with time stamps, draw each of the implied normal search trees which are current at each of the respective times, 0, 1, 2, and 3.

2. A priority verifier supports the operations, insert, delete, and not-all-bigger(z). The latter operation outputs “yes” provided there is an element currently in the set having key \( \leq z \), an “no” otherwise. \( z \) is provided by the user. Can a priority verifier have a comparison based implementation so that its operations have amortized cost \( o(\log n) \) when there are \( n \) elements in the set? Explain.

3. Consider the disjoint set problem in the setting of a universe of size \( n \), and suppose that a sequence of unions and finds is guaranteed to have the structure

   \[ \text{union} \ast \text{find} \ast \text{union} \ast \text{find} \ast \]

(i.e. the sequence has two phases, each consisting of uninterrupted unions preceding uninterrupted finds). Show that the tree-based structure discussed in class, with performance improving heuristics in place, implements such a sequence in \( O(m + n) \) time, where \( m \) is the number of find operations.

4. Given a skiplist, show, with explanation, how the search operation can be modified so that the expected cost of a search is \( O(\log j) \), where \( j \) is the position in the list at which the search terminates.

5. Do exercise 21.3-4 (p. 572) in the text.