Our probabilistic analysis of skip lists assumes that the sequence of operations used to query and update the data structure are independent of the random choices used to build the data structure. However, if a malicious adversary has additional information about the data structure, they can force worse performance.

To show how an adversary may do so, consider an arbitrary skip list. We’ll define the rank-pair of an element \( x \) to be the ordered pair \( r(x) := (L(x), p(x)) \) where \( L(x) \) is the number of levels of the skip list containing \( x \) (not counting the bottom level) and \( p(x) \) is the position of \( x \) in the skip list. For example, if the least element appears in 3 levels other than the bottom, it would have a rank-pair of \((3, 0)\). No two elements share a position, so the rank-pairs of the elements must be distinct.

We can compare the rank-pairs of elements lexicographically. In other words, given two distinct elements \( x \) and \( y \), we’ll say \( r(x) < r(y) \) if and only if either 1) \( L(x) < L(y) \) or 2) \( L(x) = L(y) \) but \( p(x) > p(y) \). (We emphasize that if \( L(x) = L(y) \), then \( r(x) \) is smaller when \( x \)’s position is greater than that of \( y \).) Again, rank-pairs are distinct, so one of \( r(x) < r(y) \) or \( r(y) < r(x) \) must be true for any pair of distinct elements \( x \) and \( y \). We say a subset of elements is increasing if for every pair of elements \( x < y \), we have \( r(y) < r(x) \). Similarly, a subset of elements is decreasing if for every pair of elements \( x < y \), we have \( r(x) < r(y) \).

(a) Suppose we have a skip list over \( n \) elements where the entire collection of elements is increasing. Argue that the worst-case time to search for an element is \( \Omega(n) \).

Advice: It may help to draw a small example where the elements have greater rank-pairs as you go from left to right.

(b) Suppose we have a skip list over \( n \) elements where the entire collection of elements is decreasing. Argue that the worst-case time to search for an element is \( \Omega(n) \).

(c) Given a sequence \( \langle a_1, a_2, \ldots, a_z \rangle \), a subsequence is any subset of its values that still appear in the same order; in particular, members of a subsequence do not need to be consecutive. The Erdős-Szekeres Theorem can be stated as follows: Any permutation of \( k^2 + 1 \) distinct numbers has either an increasing subsequence of length \( k + 1 \) or a decreasing subsequence of length \( k + 1 \).

Suppose after constructing a skip list with \( n \) elements, an adversary discovers the number of levels that contain each item. Using the Erdős-Szekeres Theorem and the previous two parts, argue that by removing a subset of elements, the adversary can force the worst-case search time to be \( \Omega(\sqrt{n}) \).
2. A binary tree is called **full** if every non-leaf node has exactly two children.

   (a) Kyle has a 27-node full binary tree in which every node is labeled with a unique letter of the Roman alphabet or the character &. Preorder and postorder traversals of the tree visit the nodes in the following order:

   - **Preorder:** I Q J H L E M V O T S B R G Y Z K C A & F P N U D W X
   - **Postorder:** H E M L J V Q S G Y R Z B T C P U D N F W & X A K O I

   Draw Kyle’s tree.  
   *Advice: The preorder traversal implies the root must be I, and Q must be its left child. What does the postorder list tell you about Q’s descendants?*

   (b) In what order does an **inorder** traversal of Kyle’s tree visit the nodes?

   (c) It turns out any full binary tree can be uniquely identified by its preorder and postorder node sequences, but that may not be the case for arbitrary binary trees. Draw a pair of distinct binary trees that have identical preorder node sequences and identical postorder node sequences.  
   *Advice: Your trees should contain very few nodes.*