In the Depth Determination problem, we maintain a forest \( F = \{T_i\} \) of rooted trees under three operations:

- **MAKE-TREE(v)** creates a tree whose only node is v.
- **FIND-DEPTH(v)** returns the depth of node v within its tree.
- **GRAFT(r, v)** makes node r, which is assumed to be the root of a tree, become the child of node v, which is assumed to be in a different tree than r but may not itself be a root.

**a.** Suppose that we use a tree representation similar to a disjoint-set forest: \( p[v] \) is the parent of node v, except that \( p[v] = v \) if v is a root. If we implement GRAFT(r, v) by setting \( p[r] \) to v and FIND-DEPTH(v) by following the find path up to the root, returning a count of all nodes other than v encountered, show that the worst-case running time of a sequence of \( m \) MAKE-TREE, DEPTH, and GRAFT operations is \( \Theta(m^2) \).

By using the union-by-rank and path-compression heuristics, we can reduce worst-case running time. We use the disjoint-set forest \( S = S_i \) where each set \( S_i \) (which is itself a tree) corresponds to a tree \( T_i \) in the forest \( F \). The tree structure within a set \( S_i \), however, does not necessarily correspond to that of \( T_i \). In fact, the implementation of \( S_i \) does not record the exact parent-child relationships but nevertheless allows us to determine any node’s depth in \( T_i \).

The key idea is to maintain in each node v a "pseudo-distance" \( d[v] \), which is defined so that the sum of the pseudo-distances along the path from v to the root of its set \( S_i \) equals the depth of v in \( T_i \). That is, if the path from v to its root in \( S_i \) is \( v_0, v_1, ..., v_k \), where \( v_0 = v \) and \( v_k \) is \( S_i \)’s root, then the depth of v in \( T_i \) is \( \sum_{0 \leq j \leq k} d[v_j] \).

**b.** Give an implementation of MAKE-TREE.

**c.** Show how to modify FIND-SET to implement FIND-DEPTH. Your implementation should perform path compression, and its running time should be linear in the length of the find path. Make sure that your implementation updates pseudo-distances correctly.

**d.** Show how to implement GRAFT(r, v), which combines the sets containing r and v, by modifying the UNION and LINK procedures. Make sure that your implementation updates pseudo-distances correctly. Note that the root of a set \( S_i \) is not necessarily the root of the corresponding tree \( T_i \).

**e.** Give a tight bound on the worst-case running time of a sequence of \( m \) MAKE-TREE, FIND-DEPTH, and GRAFT operations, \( n \) of which are MAKE-TREE operations.