The ranking officer must notify her subordinates in some sequence, after which they will recursively broadcast the message as quickly as possible to their subtrees. This is just like the homework problem on triathalon scheduling from the chapter on greedy algorithms: the subtrees must be "started" one at a time, after which they complete recursively in parallel. Using the solution to that problem, she should talk to the subordinates in decreasing order of the time it takes for their subtrees (recursively) to be notified.

Hence, we have the following set of sub-problems: for each subtree T' of T, we define $x(T')$ to be the number of rounds it takes for everyone in T' to be notified, once the root has the message. Suppose now that T' has child subtrees $T_1, \ldots, T_k$, and we label them so that $x(T_1) \geq x(T_2) \geq \cdots \geq x(T_k)$. Then by the argument in the above paragraph, we have the recurrence

$$x(T') = \min_j \left[j + x(T_j) \right].$$

If T' is simply a leaf node, then we have $x(T') = 0$.

The full algorithm builds up the values $x(T')$ using the recurrence, beginning at the leaves and moving up to the root. If subtree T' has d' edges down from its root (i.e. d' child subtrees), then the time taken to compute $x(T')$ from the solutions to smaller sub-problems is $O(d' \log d')$ — it is dominated by the sorting of the subtree values. Since a tree with n nodes has $n - 1$ edges, the total time taken is $O(n \log n)$.

By tracing back through the sorted orders at every subtree, we can also reconstruct the sequence of phone calls that should be made.

[1] ex449.390.867