Proof of the lemma. We show that we can get from any tree to a left path (any node has just a left child) with at most \( n - 1 \) rotations. Since a rotation can be undone by another rotation, we can then get from the path to any tree with \( n - 1 \) more rotations, so we use \( \leq 2n - 2 \) rotations in total. (If you care to know, the optimum is \( 2n - 6 \)).

The tree always has \( n - 1 \) edges; we show that with one rotation, we can increase the number of edges on the left path starting from the root by one. So we need at most \( n - 1 \) rotations to get all edges on that path. We proceed as follows: find a node \( x \) which branches off the left path, and rotate it up to the path. Then the edge above \( x \) becomes a left-path edge. Any other edge does not change status. See the figure:

\[
\begin{array}{c}
\text{\ldots z \ldots} \\
\text{\ldots x \ldots} \\
\text{\ldots y} \\
\text{A B} \\
\end{array} \quad \Rightarrow \quad \\
\begin{array}{c}
\text{\ldots z \ldots} \\
\text{\ldots y \ldots} \\
\text{\ldots x A} \\
\end{array}
\]

Competitiveness with \( O(\lg n) \) guarantee. We conceptually break the operations into chunks of \( n \). We keep the count of the current operation, the cost of the current chunk, and a history of all operations ever performed. At the beginning of each chunk (the current operation is divisible by \( n \)), we configure the tree to what the \( \alpha \)-competitive BST would look like. Finding how that BST would look like is free: we simulate the BST from the beginning of time, but this is just a conceptual step, and we’re not doing any actual rotation on the real tree. Then, we configure the real BST to what it should be using \( \leq 2n \) rotations. A normal operation is executed by calling the competitive BST algorithm. However, when the cost of the current chunk reaches \( n \lg n \), we switch to an \( O(\lg n) \) tree (say splay trees). Until the end of the chunk, we just use the splay tree algorithm, which gives \( O(n \lg n) \) cost. The cost for chunk \( i \) is \( O(n) + \min\{T_i, O(n \lg n)\} \), where \( T_i \) is the cost of the competitive BST for chunk \( i \). Summing up, we get a cost of \( O(n) \cdot \left\lceil \frac{m}{n} \right\rceil + \sum_i \min\{T_i, O(n \lg n)\} = O(m) + \min\{\sum_i T_i, O(m \lg n)\} = O(m) + \min\{\alpha \text{OPT}, O(m \lg n)\} = O(\min\{\alpha \text{OPT}, m \lg n\}) \) – because \( \text{OPT} \geq m \).