Dynamizing static search structures.

(a) To perform a successor search we start from the root node, perform a search for a successor and follow the link to it’s left until we reach a leaf node.

The runtime recurrence is then: \( T(n) = S(\Theta(n^{1/c})) + T(\Theta(n^{1-1/c})) \)

For fusion trees we have:

\[
T(n) = O(\log_{\omega} n^{1/c}) + T(\Theta(n^{1-1/c}))
\]

Hence

\[
T(n) = O\left( \sum_{i=0}^{\infty} c^{-1} \log_{\omega} n^{(1-1/c)^i} \right) = O\left( \sum_{i=0}^{\infty} c^{-1} (1 - 1/c)^i \log_{\omega} n \right)
\]

(b) The space recurrence is: \( C(n) = \Theta(n^{1/c})(C(n^{1-1/c}) + 1) \). Since we have \( \Theta(n^{1/c}) \) subtrees of size \( O(n^{(1-1/c)}) \) plus \( \Theta(n^{1/c}) \) for the space at the current level. We see that the recurrence solves to \( C(n) = O(n) \).

(c) We will constrain the number of nodes in a subtree rooted at a node at depth \( d \) to be

\[
k = \Theta(n^{(1-1/c)^d})
\]

When inserting or deleting a node, we make sure that all the nodes on our path satisfy the given property. when merging or splitting a node with \( k \) children we have to reconstruct its parent. The node’s parent will have \( \Theta(k^{c/(c-1)}) \) descendands, and \( \Theta(k^{1/(c-1)}) \) children. Thus, rebuilding the parent would take \( O(k^{b/c-1}) \).

At any given level, we have to rebuild the node only after \( \Theta(k) \) descendands have been inserted/removed. Hence the amortized cost is \( O(k^{b/c-1}) \). Choosing \( b \geq 1 \) gives us \( O(1) \) amortized cost per level, and the total of \( O(\log \log n) \)