Consider the B-tree \( T \) of order 4 illustrated below. (Recall that “order 4” means, in part, that each node must contain at least one key but no more than three.) For each operation in the list (a) through (h), show the B-tree that results from performing that operation on \( T \). (Each operation is to be applied to the tree \( T \) illustrated below, not to the tree resulting from applying all the previous operations to \( T \).) You need not draw the entire tree each time—just show that portion of it that was changed in carrying out the operation, as well as a little surrounding context.

Assume, in carrying out the operations, that redistribution is used whenever possible. That is, split an overflowing node only if both of its adjacent siblings are full. Similarly, concatenate/merge two nodes only if both of the siblings adjacent to the underflowing node are on the verge of underflowing.\(^1\) Follow the algorithms presented in class, which correspond to those described on the relevant web page. Keep in mind that redistribution involves two adjacent siblings and their parent, not three or more siblings and not first (or second, etc.) cousins (i.e., nodes with a common grandparent, great-grandparent, etc.).

\[
\begin{array}{c}
\text{(a) insert 31} \\
\text{(b) delete 82} \\
\text{(c) insert 68} \\
\text{(d) delete 76} \\
\text{(e) insert 55} \\
\text{(f) delete 25} \\
\text{(g) insert 94} \\
\text{(h) delete 6}
\end{array}
\]

\[
\begin{array}{c}
\text{++---++---++---} \\
\mid \text{23 36 76} \mid \\
\text{++---++---++---} \\
\text{+/ /} \\
\text{++---++---++---} \\
\text{+/ /} \\
\text{++---++---++---} \\
\text{+/ /} \\
\text{++++++} \\
\text{+/ /} \\
\text{++++++} \\
\text{+/ /} \\
\text{++++++} \\
\text{+/ /} \\
\text{++++++} \\
\text{+/ /} \\
\text{++++++} \\
\text{+/ /} \\
\text{++++++} \\
\text{+/ /} \\
\text{++++++} \\
\text{+/ /} \\
\text{++++++} \\
\text{+/ /} \\
\text{++++++} \\
\text{+/ /} \\
\text{++++++} \\
\text{+/ /} \\
\text{++++++} \\
\text{+/ /} \\
\end{array}
\]

\(^1\)Not every node has two adjacent siblings, of course. A node that is the leftmost or rightmost child of its parent has only one, and the root has none.