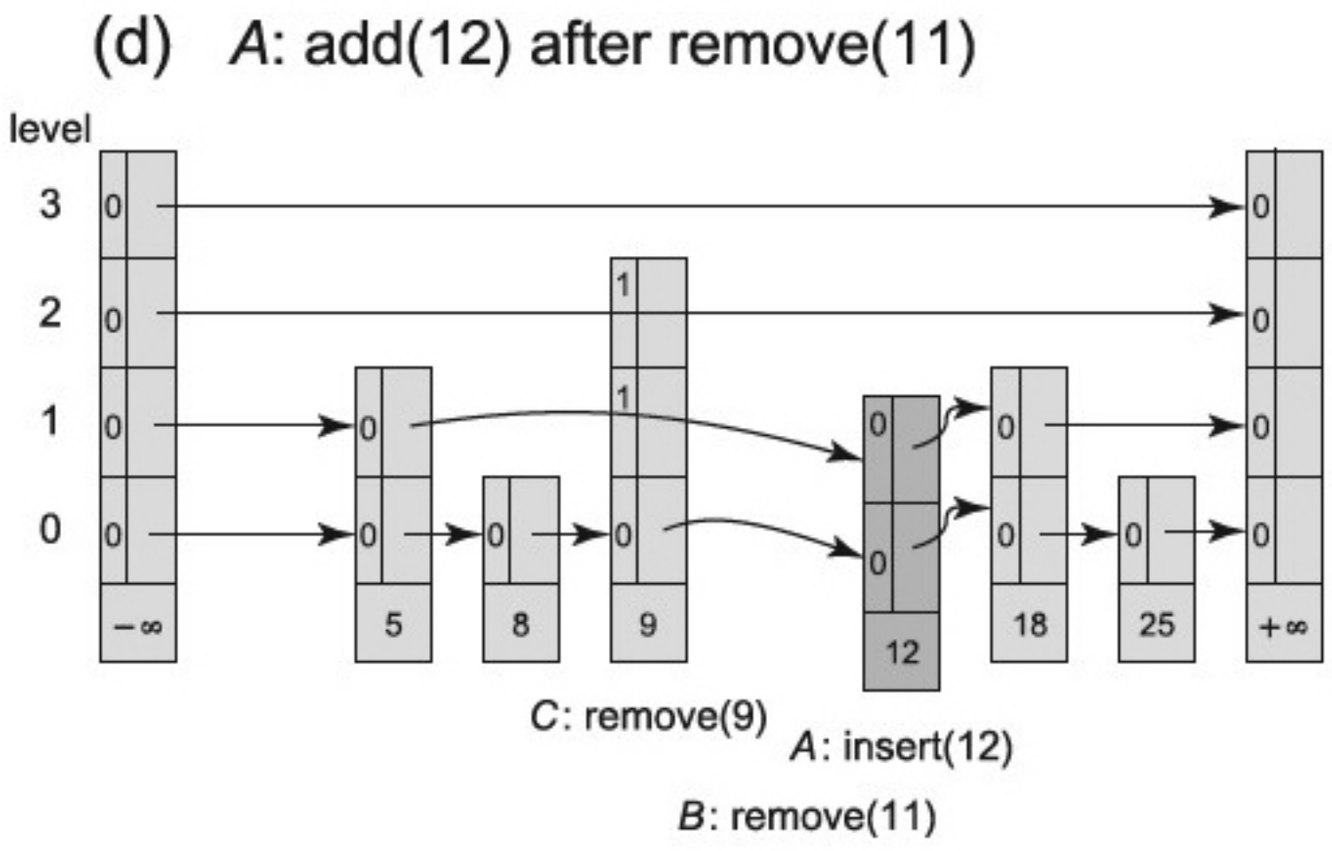

FIGURE 14.9D The LockFreeSkipList class: an add() call. Each node consists of links that are unmarked (a 0) or marked (a 1). In part (a), add*(12)* calls find*(12)* while there are three ongoing remove() calls. The find() method "cleans" the marked links (denoted by 1s) as it traverses the skiplist. The traversal is not the same as a sequential find*(12)*, because marked nodes are unlinked whenever they are encountered. The path in the figure shows the nodes traversed by the pred reference, which always refers to unmarked nodes with keys less than the target key. Part (b) shows the result of redirecting the dashed links. We denote bypassing a node by placing the link in front of it. Node 15, whose bottom-level next reference was marked, is removed from the skiplist. Part (c) shows the subsequent addition of the new node with key 12. Part (d) shows an alternate addition scenario that would occur if the node with key 11 were removed before the addition of the node with key 12. The bottom-level next reference of the node with key 9 is not yet marked, and so the bottom-level predecessor node, whose next reference is marked, is redirected by the add() method to the new node. Once thread *C* completes marking this reference, the node with key 9 is removed and the node with key 5 becomes the immediate predecessor of the newly added node.