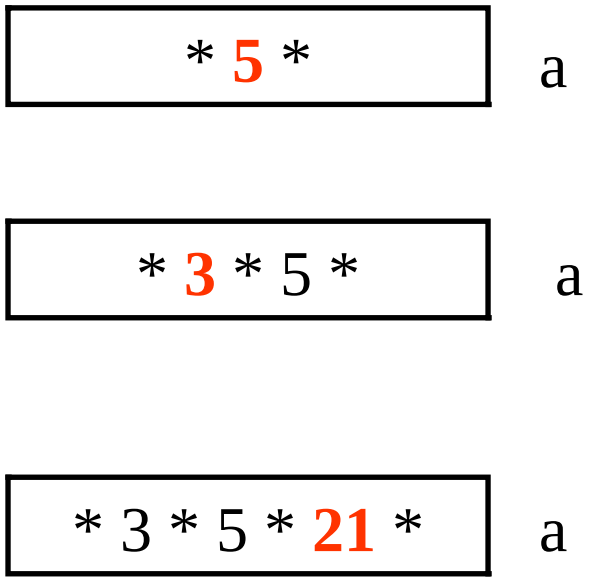
B-Tree Example

IS 320

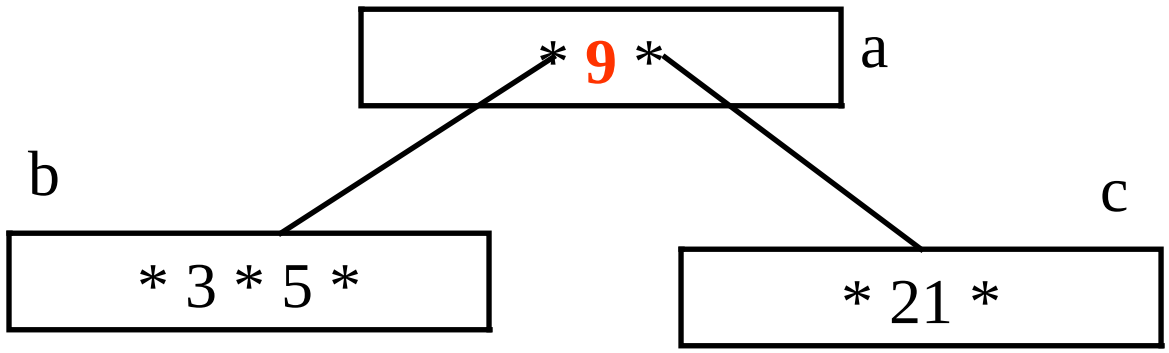
Operations

- B-Tree of order 4
  - Each node has at most 4 pointers and 3 keys, and at least 2 pointers and 1 key.
- Insert: 5, 3, 21, 9, 1, 13, 2, 7, 10, 12, 4, 8
- Delete: 2, 21, 10, 3, 4

Insert 5, 3, 21

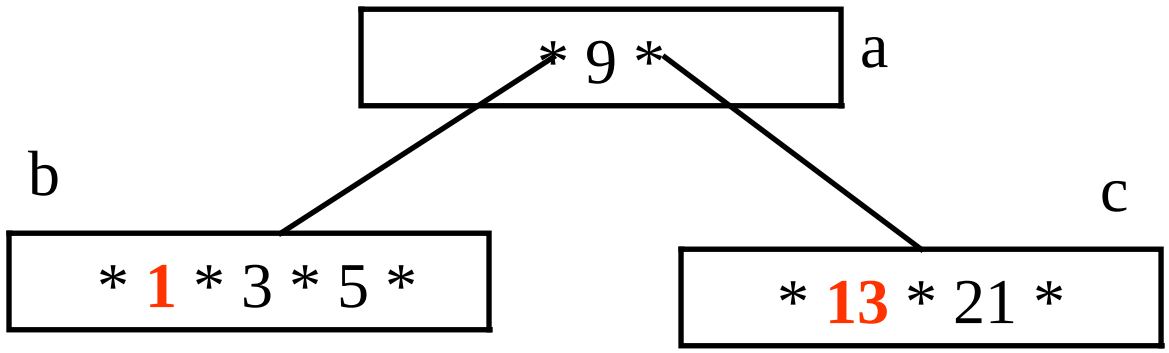


Insert 9



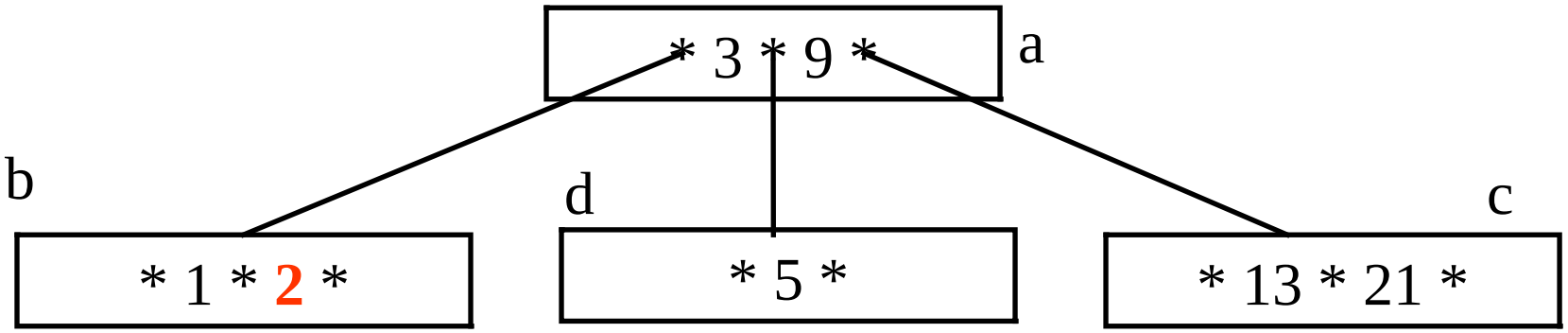
Node a splits creating 2 children: b and c

Insert 1, 13



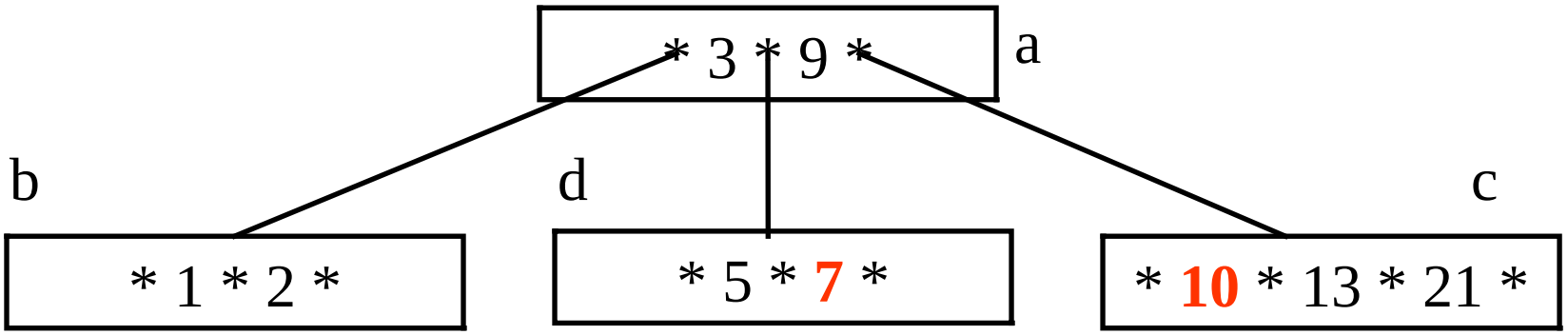
Nodes b and c have room to insert more elements

Insert 2



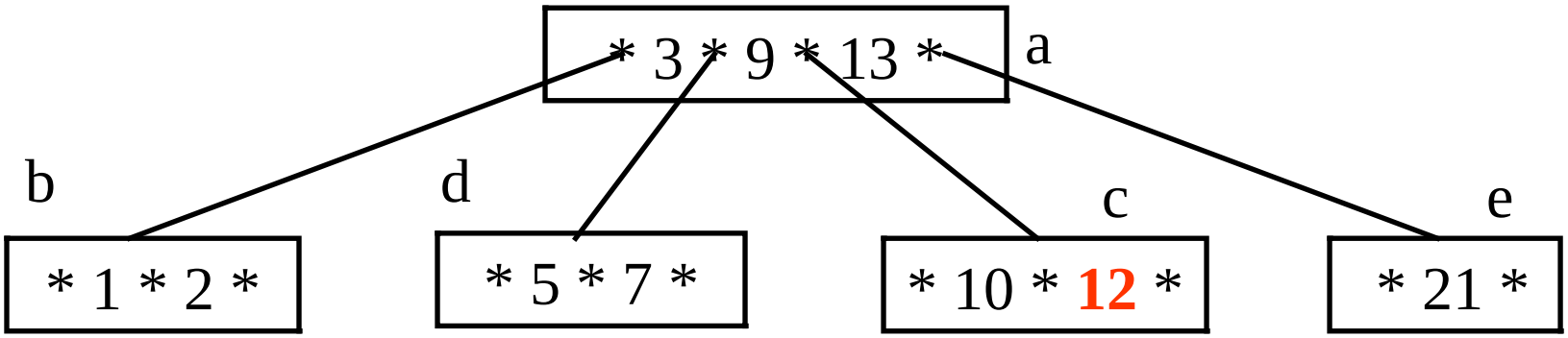
Node b has no more room, so it splits creating node d.

Insert 7, 10



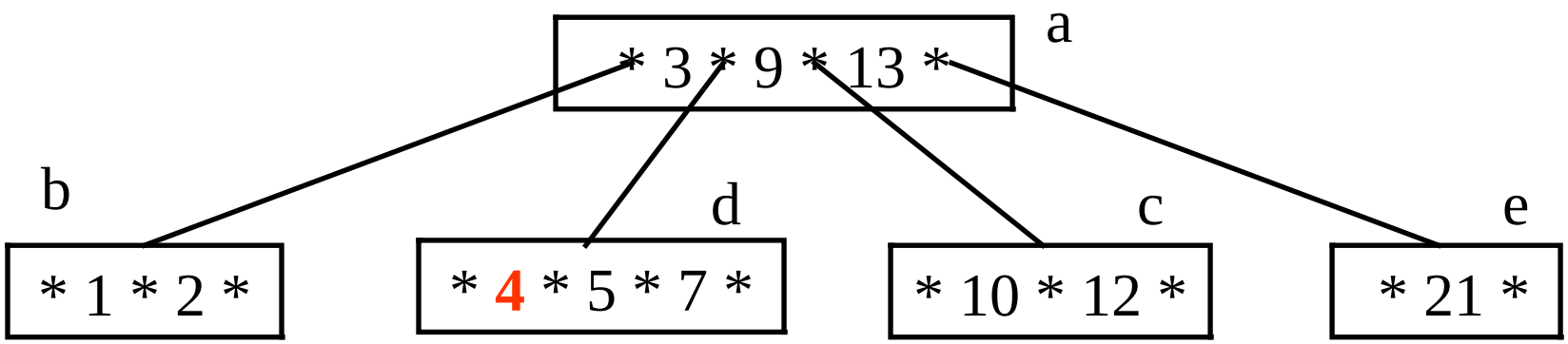
Nodes d and c have room to add more elements

Insert 12



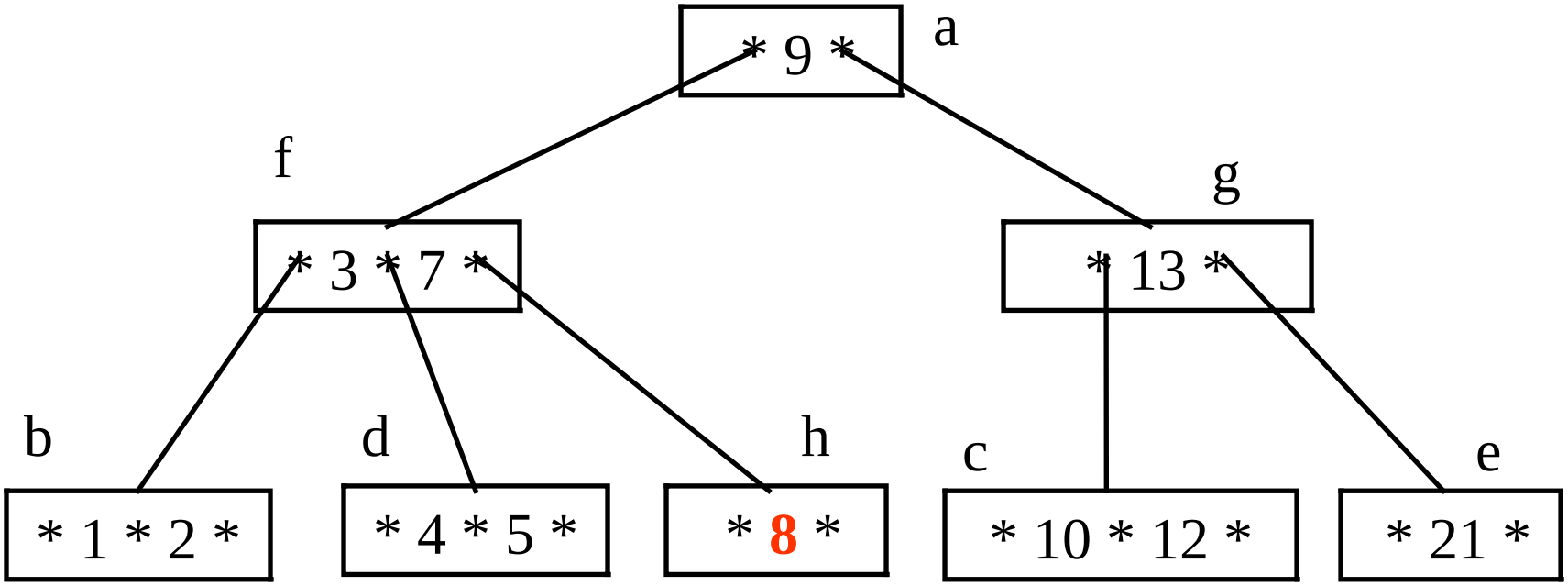
Nodes c must split into nodes c and e

Insert 4



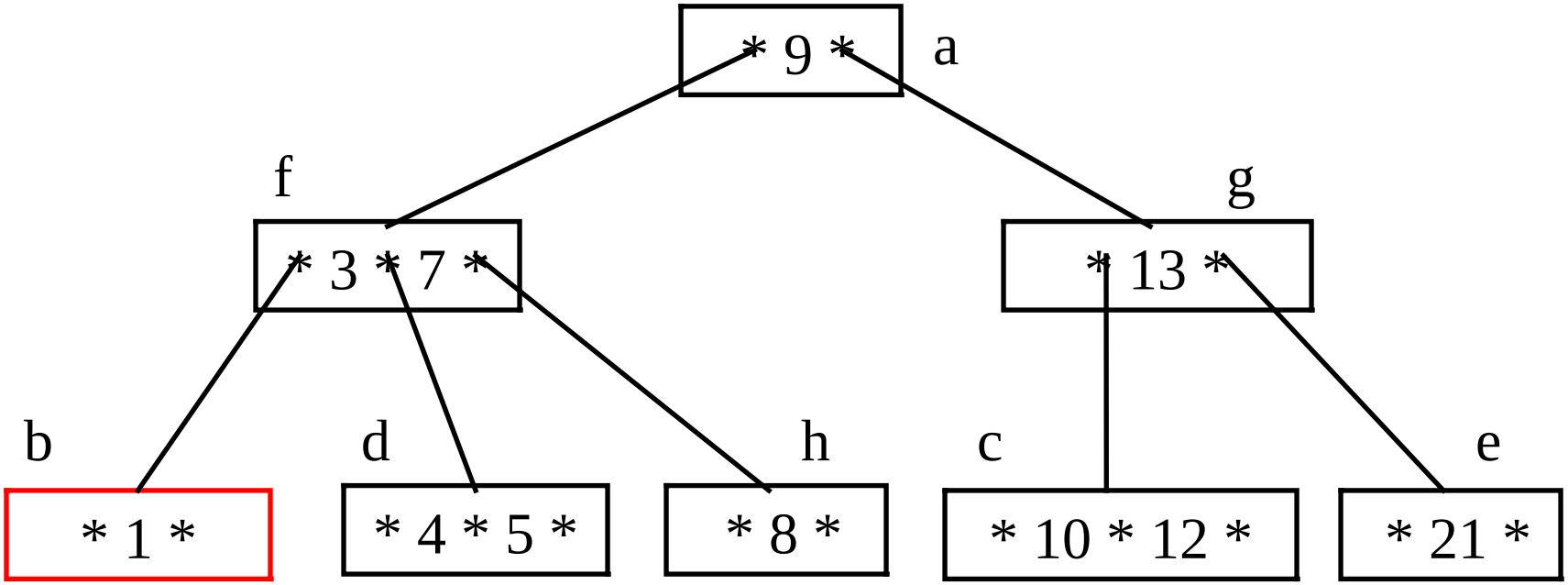
Node d has room for another element

Insert 8



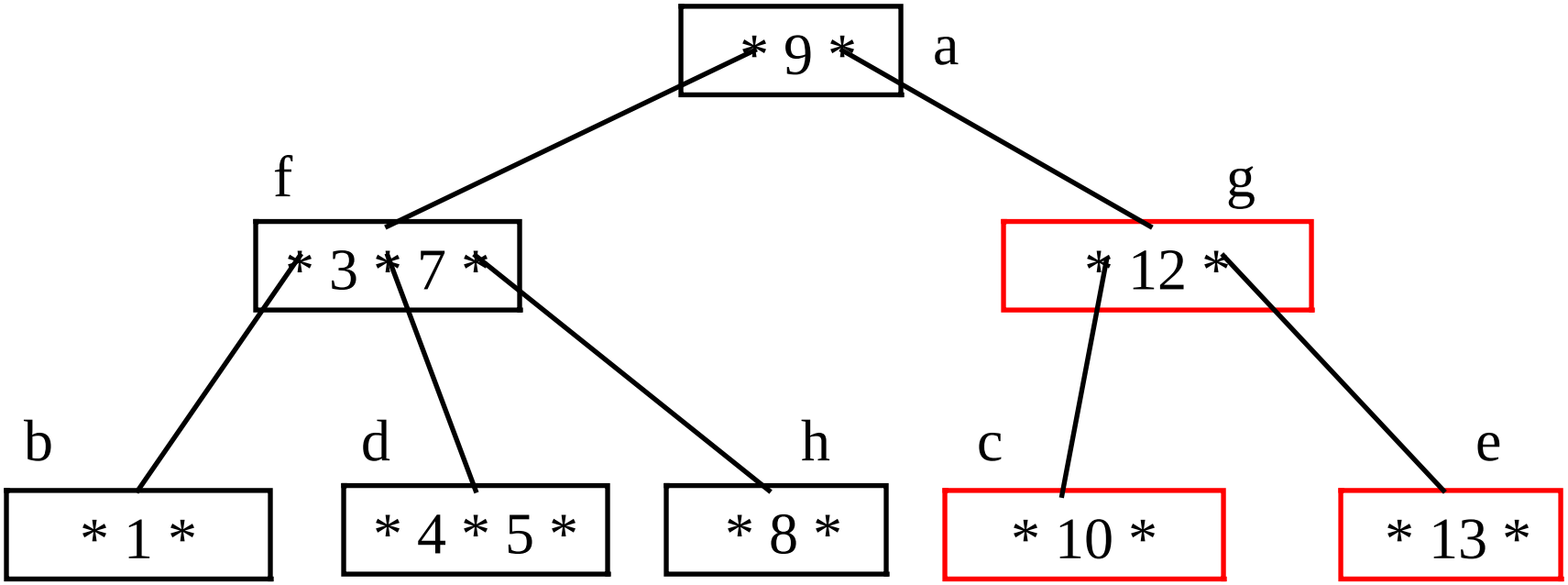
Node d must split into 2 nodes. This causes node a to split into 2 nodes and the tree grows a level.

Delete 2



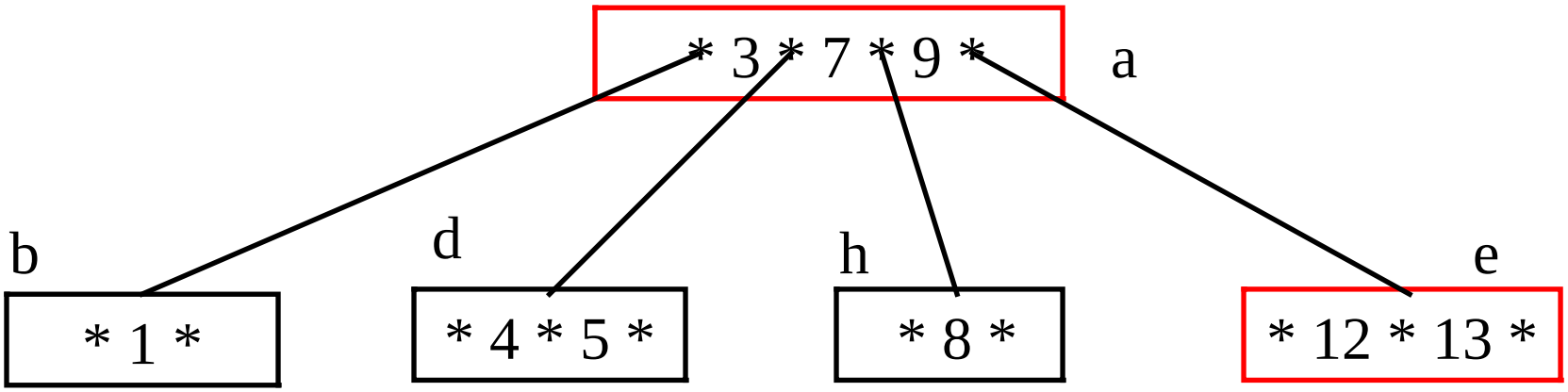
Node b can loose an element without underflow.

Delete 21



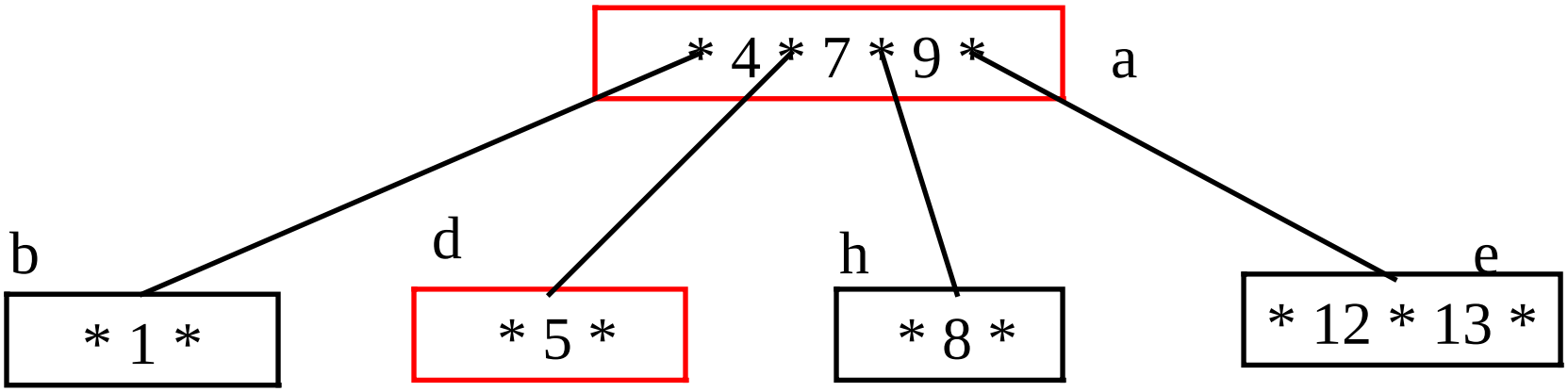
Deleting 21 causes node e to underflow, so elements are redistributed between nodes c, g, and e

Delete 10



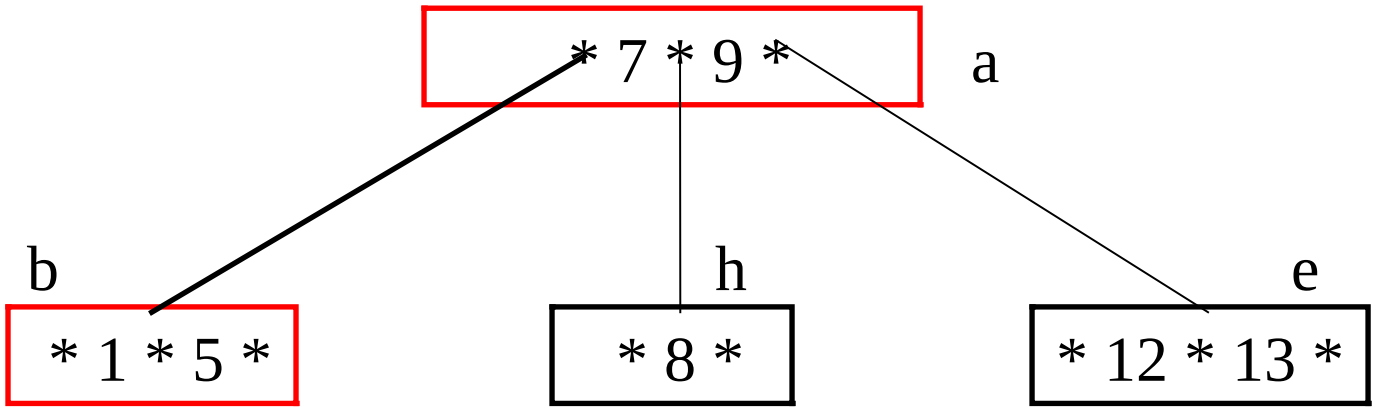
Deleting 10 causes node c to underflow. This causes the parent, node g to recombine with nodes f and a. This causes the tree to shrink one level.

Delete 3



Because 3 is a pointer to nodes below it, deleting 3 requires keys to be redistributed between nodes a and d.

Delete 4



Deleting 4 requires a redistribution of the keys in the subtrees of 4; however, nodes b and d do not have enough keys to redistribute without causing an underflow. Thus, nodes b and d must be combined.