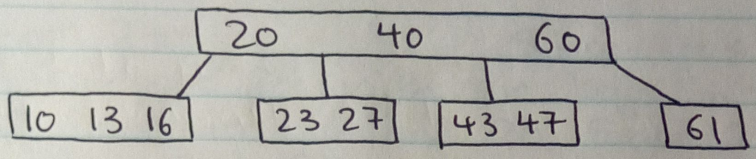
B-Tree Operations

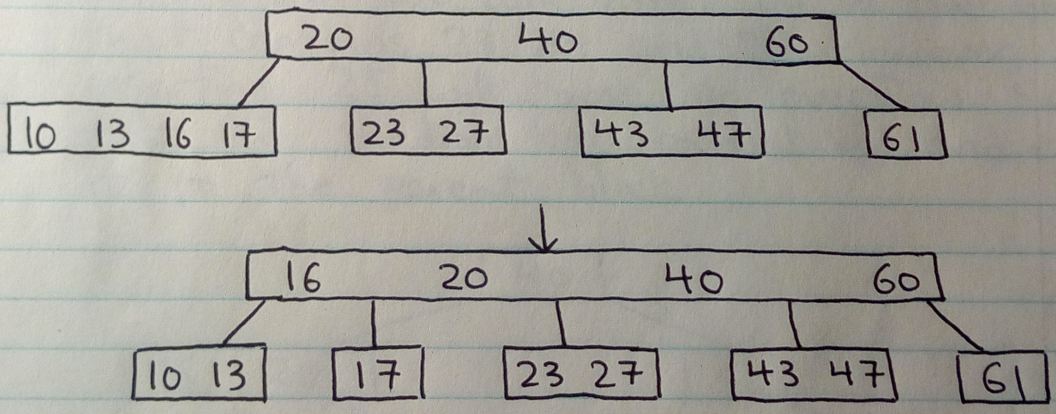
1. Insert

Suppose we were given this B-Tree

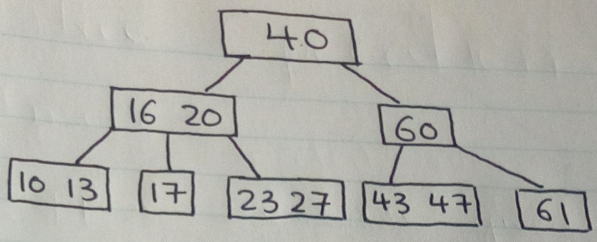


and we want to insert the following keys: 17, 24, 25 and 26.

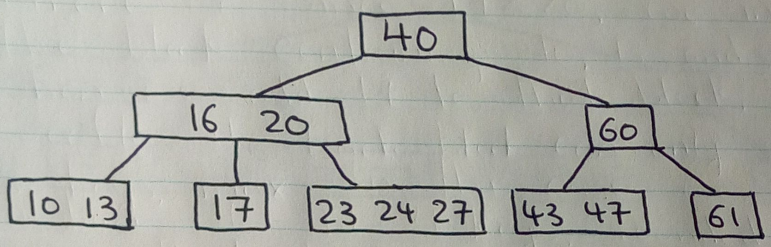
Insert 17: Since 17<20, we insert it in the block containing 10, 13 and 16. However, when we add 17 to that block, we will overflow it. Each block can only store up to 3 keys, max. In this case, we take the median of the 4 keys, rounded up, and move that to the parent node.



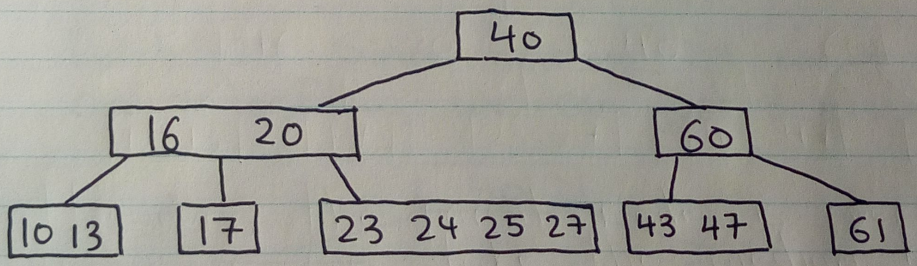
However, the parent node is overflowing now, so we take the median key, rounded up, and push that key up.

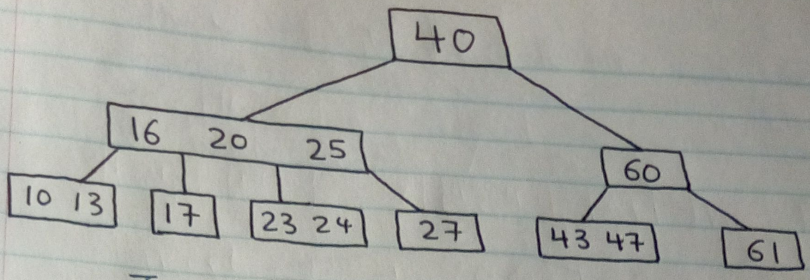


Insert 24: Since 24 < 40, we will compare 24 with the keys in the node that contains 16 and 20. Since 24 > 20, we will insert 24 in the node that contains 23 and 27. Since inserting 24 doesn't cause an overflow, we don't have to do anything else.

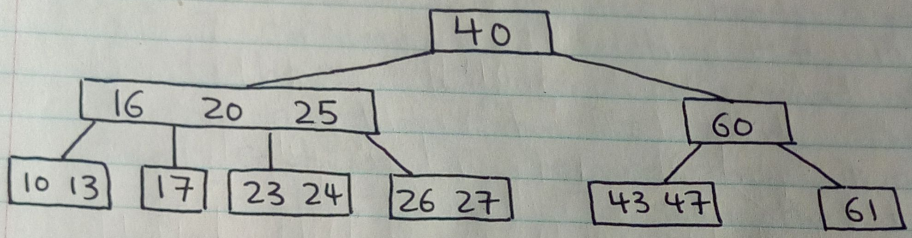


Insert 25: We will insert 25 in the block that contains 23, 24 and 27. However, inserting 25 will cause an overflow. So, we take the median key, rounded up, and push it to the parent block.





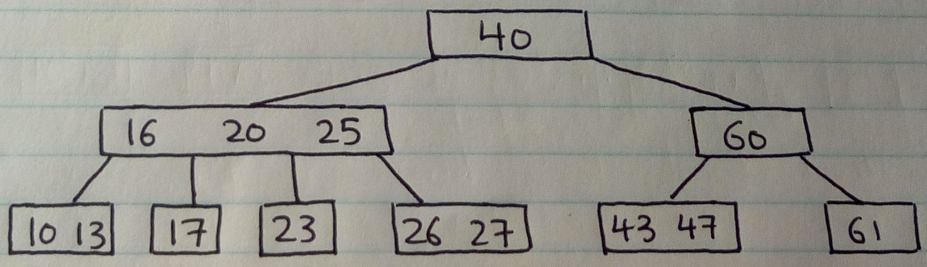
Insert 26: We will insert 26 in the block with 27. Since inserting 26 doesn't cause an overflow, we don't do anything further.



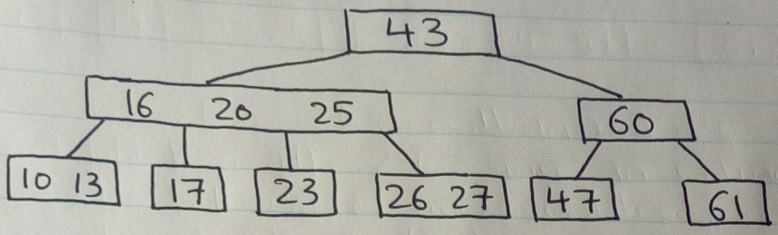
2. Delete

Given the B-Tree from above, right after inserting 26, delete 24, 40 and 47. For our purposes, we will replace a key with its successor, if necessary.

Delete 24: Since deleting 24 won't cause an underflow, we can just delete it.



Delete 40: To delete 40, we have to replace it with its successor, 43, since 40 has no sibling to borrow from or to merge with.



Delete 47: If we delete 47, we will cause an underflow. Every block needs a minimum of 2 children, and deleting 47 will cause 60 to have 1 children. Furthermore, we can't borrow anything from its sibling, the block that contains 61, because there is only 1 key in that block. To solve the issue, we merge 60 and 61, push 43 down and push 25 up.