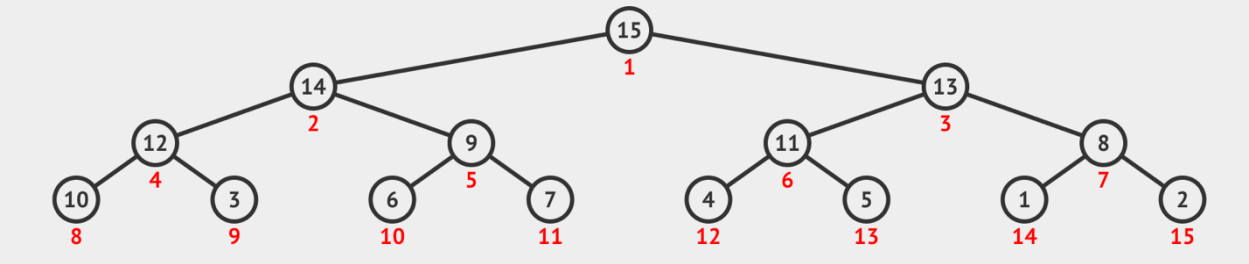
1. Node A must be greater than 30 and less than 35. This is because Node A is located to the left of a range boundary with the value of 35, but it is to the right of the node with a range boundary of 30, Node B must be greater than 35 but less than 50. This is because it is to the right of the 35 boundary and to the left of a node with a boundary of 50. The range boundary of the ? must be greater than 100 but not equal to 150, this is because the range boundary is to the right of the 100 boundary, and it can't be equal to 150 because each key is unique and 150 is already a value used.
2.
    a. The minimum number of keys that can be stored is 17.
    b. There can be a maximum of 8 nodes that stores keys including the root. Since the root needs at least 1 key and all other nodes need at least half of (m-1)/2 keys, which would equal 2 (rounded up to get at least half in each node). So, with 15 keys, 8 would be the maximum number of nodes that the tree can have that store keys.
3.
    a. The key functions used when building the heap in this order are shifting the root of the tree and subtrees to align with the balance of a max-heap. If the element inserted is larger than what is above it a shift will occur to ensure that the greatest value is the root of the heap.



    b. With the first delete function, we would replace the root with the rightmost leaf function and initiate 3 shiftDown() functions involving the left nodes to the root in order to rebalance the tree. The second delete function would initate one right shiftDown() followed by a left shift down. The final delete function would initiate 3 shiftDown() functions involving the nodes to the left of the root. All these initial start by replacing the root with the rightmost leaf.