3.2.15

a. E Q
b. E Q J
c. E Q
d. E Q J
e. E D E Q J M J Q T S T if you implement it using keys(). Note that we include visits to the nodes when we need to count them. It can be different if you implemented differently.
f. E D E Q J M J Q T S T Note that we include visits to the nodes when we need to enqueue them. It can be different if you implemented differently.

3.2.20 The algorithm needs time at most proportional to the tree height in order to get a pointer to the leftmost node in the range, and then it enumerates all nodes/keys in the range (exactly one each and only them) in an inorder fashion, in total time proportional to the number of keys in the range.

3.2.24 Then we would have a sorting algorithm with fewer than $N \log N$ comparisons (to build the BST and traverse it inorder).

3.2.36 For range $[lo..hi]$ simply search for $lo$ and $hi$ in the tree, placing a new pointer to the next node in each node of the (two) search paths. Note that you need extra space for the pointers at most twice the height of the tree (for marking the two paths). Then do an (iterative) inorder traversal of the BST, using the extra pointers to not visit parts of it that do not belong to the range.

3.3.38 We can show (induction on the number of nodes) that any BST can be transformed into a right-hand chain using (at most $N - 1$, by the way) right-rotations. This is also reversible (we can go from the chain to the original BST reversing the right-rotations to left-rotations). So, we can go from BST $T_1$ to chain $C_1$ using right-rotations $R_1$, and from BST $T_2$ to chain $C_2$ using right rotations $R_2$. But $C_1 = C_2$ (why?); then doing $R_1$ and the reverse of $R_2$, we can get from $T_1$ to $T_2$.

3.4.15 In the worst-case, each element will probe all previously inserted elements; even if this happens only for the $N/4$ last elements before the next resizing (the previous one happened when the total number of elements was $N/4$), then the number of compares is

$$\sum_{i=0}^{N/4-1} i = N/8(N/4 - 1) = O(N^2).$$

3.5.24 You can build a red-black tree with the left boundaries of the intervals used as the node keys and the range width also stored in the corresponding node.

3.5.25 Can be implemented in many different ways, e.g., as in the previous exercise or using hashing.