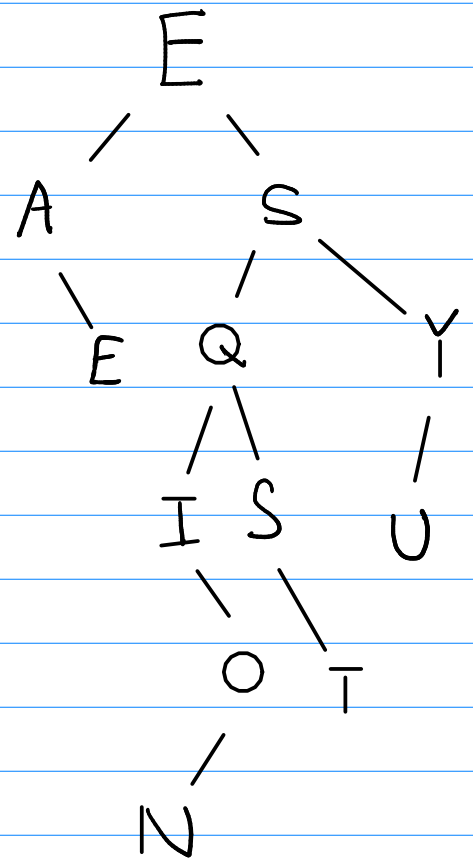
3.2.1 Draw the BST that results when you insert the keys `E A S Y Q U E S T I O N`, in that order (associating the value `i` with the `i`th key, as per the convention in the text) into an initially empty tree. How many compares are needed to build the tree?

EASYQUESTION



3.2.5 Suppose that we have an estimate ahead of time of how often search keys are to be accessed in a BST, and the freedom to insert them in any order that we desire. Should the keys be inserted into the tree in increasing order, decreasing order of likely frequency of access, or some other order? Explain your answer.

The keys should be inserted in decreasing order of likely frequency, where the keys that have a high frequency of access will be placed near the root of the tree. This means that it will take much less time to access the keys since the algorithm will not need to traverse the tree for long.

Imagine having to walk the distance to your fridge to get a snack in order to watch the Broncos whoop the Raiders. Ideally, you would want the fridge to be closer, so you wouldn't have to walk so far each time you need a snack.

This is the same case with search keys in a BST. Place the search keys of highest frequency of access near the root so that the algorithm does not need to search very far to find them.