Compare the heights of the resultant trees- how do they compare with a Binary Search Tree (BST) for the same input values?

|  |  |  |  |
| --- | --- | --- | --- |
| **Tree** | **Height of set 1** | **Height of set 2** | **Height of set 3** |
| Red-Black | 3 | 3 | 4 |
| 2-3-4 | 1 | 1 | 2 |
| B | 1 | 1 | 1 |
| BST | 4 | 7 | 10 |

Compare the complexity of the algorithms, how much work would be required for the main operations: insert(), find(), delete()? Compare this to a BST.

* BST and Red-Black trees have the same complexity for find.
* 2-3-4 and B trees have a slightly more complex find operation due to multiple keys at each node.
* BST has a much simpler delete and insert function when compared to all Red-Black, 2-3-4 and B-trees as it does not perform any operations to balance the tree

Compare the understandability of the algorithms, which would be easier to implement?

* The hardest algorithm to understand and implement is the Red-Black trees due to the multiple different operations it uses for balancing (colour swap, left swap and right swap)
* Both the 2-3-4 and B-trees share a similar ease of understanding since they only use a split function to balance.

Describe how an in-order traversal would work on each type of tree.

In-order traversal searches the tree by visiting the left child branch then the parent then the right child branch.

* For a Red-Black tree this algorithm would be the same as the one implemented for the BST
* For the 2-3-4 and B-tree this algorithm would work by:
  + checking the left child branch of the left key
  + then the left key
  + then its right child branch
  + then the next parent key
  + then that keys child and so on for each key in the node