Data Structures 4

Question 3:

|  |  |  |
| --- | --- | --- |
| Type | Usage | Space Complexity |
| Deque<Node> backtrackNodes | Saves the node before the insertion operation. Used for regular insertion and for insertion with split. | In regular only one added. In insertion with splits we add as many splits we have. |
| Deque<Node> backtrackParent | In split, saves the node that median value joins. | In regular only one added. In insertion with splits we add as many splits we have. |
| Deque<Integer> backtrackMidIndex | In split, saves the index of median value in it's new node. | In regular only one added. In insertion with splits we add as many splits we have. |
| Deque<T> backtrackInserted | Saves the value that was inserted. To know which value to delete in backtrack. | In regular only one added. In insertion with splits we add as many splits we have. |
| Deque<Character> backtrackType | Saves the type of the insertion, to know which backtrack type to use. | In regular only one added. In insertion with splits we add as many splits we have. |

Question 4:

|  |  |  |
| --- | --- | --- |
| Operation | Number of repetitions | Total time complexity |
| Backtrack without merge operation: Deletion of last inserted value from the node we saved. | 1 | In regular backtrack (no split involved), we use the deques we saved to simple delete. |
| Backtrack with merge operation:  For each split that was made we merge the nodes that were splitted. | As many splits we had, for most | As long as we see the same value in deque, we do a merge (that is "fixed" operation-קבוע) |