**Section 3: Theoretical questions**

Question 1:

|  |  |  |
| --- | --- | --- |
| Type | Usage | Space Complexity |
| ArrayDeque<Object> | 3 to 5 objects (nodes and enum – ImbalanceCases) are inserted, that are used to delete the node, and undo rotations (for each insertion) | *for each insertion* |

Question 2:

|  |  |  |
| --- | --- | --- |
| Operation | Number of repetitions | Total Time Complexity |
| Deletion | 1 |  |
| Undo – Rotation (reverse side) | 0 or 1 or 2 |  |

Question 3:

|  |  |  |
| --- | --- | --- |
| Type | Usage | Space Complexity |
| ArrayDeque<Object> | For each insertion – up to 2-cell arrays were inserted to the deque, each first cell is the value of the inserted object, and the second is the value (if exists) of the median value of a split node.  There could only be up to splits. | *for each insertion* |

Question 4:

|  |  |  |
| --- | --- | --- |
| Operation | Number of repetitions | Total Time Complexity |
| Deletion | 1 |  |
| Merge | Up to |  |

Question 5:

The time complexity of Danny’s implementation of the backtrack operation is because he is simply using a single line assigning the copy’s root to being the current root we have in the tree.

Question 6:

No, it is not, it’s true that the time complexity of the backtrack operation is , but at the same time, he increased the time complexity of the insertion operation from to , and also his space complexity for the entire implementation went from for each insertion, to .