

Professor: For each sub-task, mention the complexity order of your algorithm and justify it:

readCSVFile - calls arrayList add method inside of it.

Overall time: 148 seconds to build the binary search trees

Complexity: readCSVFile $O(n)$ * arrayListAddMethod $O(n)$ overall.

In a loop, readCSVFile reads $O(n)$ lines into a Report object using createReport, then calls arrayListAddMethod, which uses a loop to check if there is an $O(n)$ BST with state equal to report state. If there is, insert it into the BST $O(1)$. If not, create a new BST, set state to report state, insert report into BST, add BST into arrayList $O(4)$ -> $O(1)$.

Total Complexity: $O(n) * O(n+4+1+1) \rightarrow O(n^2)$

countChildren - recursive counting method based on input date

Overall time: 0 seconds to count this using children count fields

Complexity: $O(\log_2(n))$

My intention with countChildren was initially to calculate the number of children per node while the BST was being read with a separate method, in-Order traversal. With modifications to match what was asked in our deliverables, I accidentally created the exact same method as countRecords.

The following explanation will be in accordance to countChildren and countRecords since they do the same thing:

- return 0 if root is null $O(1)$. Otherwise, run a precondition (if input date is after or equal to report date) that if passed, adds 1 to a counter $O(1)$. The return statement adds the counter along with a recursive call to the node's left and right child $O(1 + n/2)$. Similar to a binary search, this call splits the tree in half (left, right node), making them the roots with this call. This call is then made again/2, again/2, logarithmically approaching the limit n . Therefore,

Total Complexity: $O(1) * O(1) * O(1+(n/2 * (n \text{ times}))) \rightarrow O(\log_2(n))$

countRecords - $O(\log_2(n))$

Overall time: 0 seconds to calculate this using recursive method