Report 2 - Dynamic Data Structures

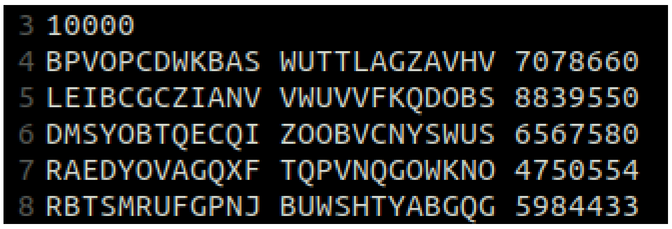
Bogna Kilanowska 148252

Dependence of insertion time of the number n (OL vs. BST vs. BBST)

Data:

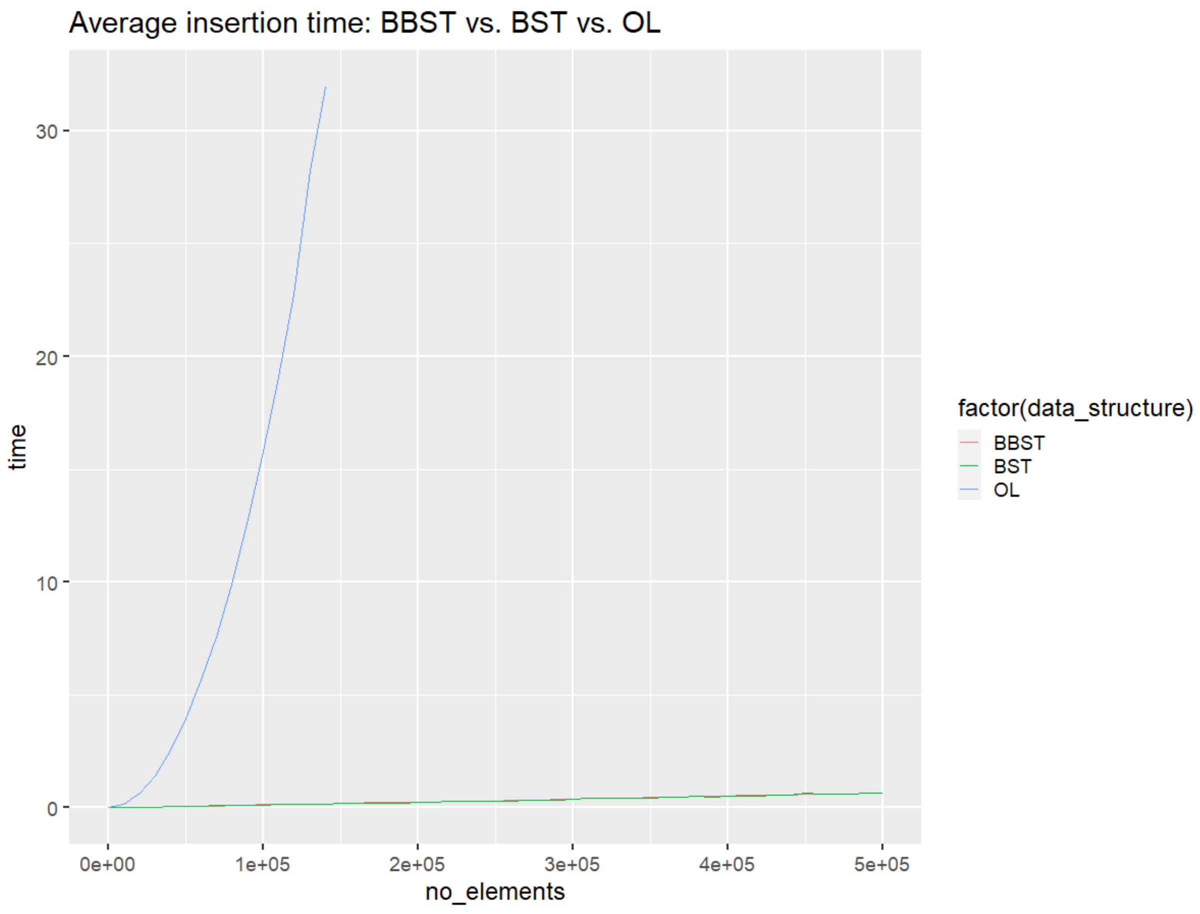
The average time of inserting the data into the following data structures was measured for random sets of data, consisting of 0, 10000, 20000, …, 500000 records (BST and BBST) and 0, 10000, 20000, …,150000 records (OL)

Sample of the data file:



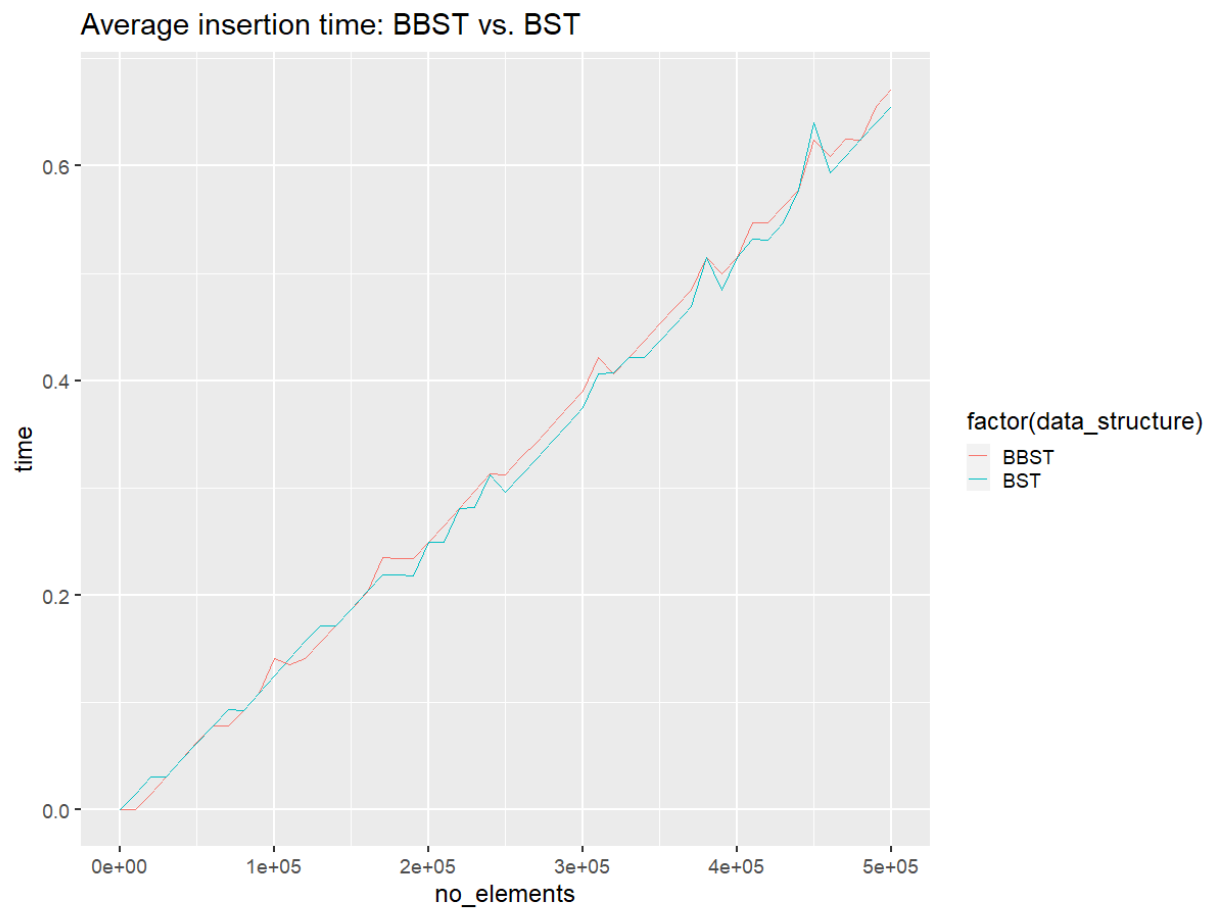
```
10000
BPVOPCDWKBAS WUTTLAGZAVHV 7078660
LEIBCGCZIANV VWUVVFKQDOBS 8839550
DMSYOBTQECQI ZOOBVCNYSWUS 6567580
RAEDYOVAGQXF TQPVNQGOWKNO 4750554
RBTSMRUFGPNJ BUWSHTYABGQG 5984433
```

Chart:



Conclusions:
The average time of inserting new records to the Ordered List is much worse than the average time of inserting the same data to BST and BBST. It is because to add a new element to the Ordered List, the algorithm must check all the elements one by one, till it finds a bigger element than the new one (or smaller if we put elements to OL in descending order). The time complexity of adding a new element to OL equals O(n). To see how BST and BBST performed, we must compare times for only those two data structures.

Dependence of insertion time of the number n (BST vs. BBST)

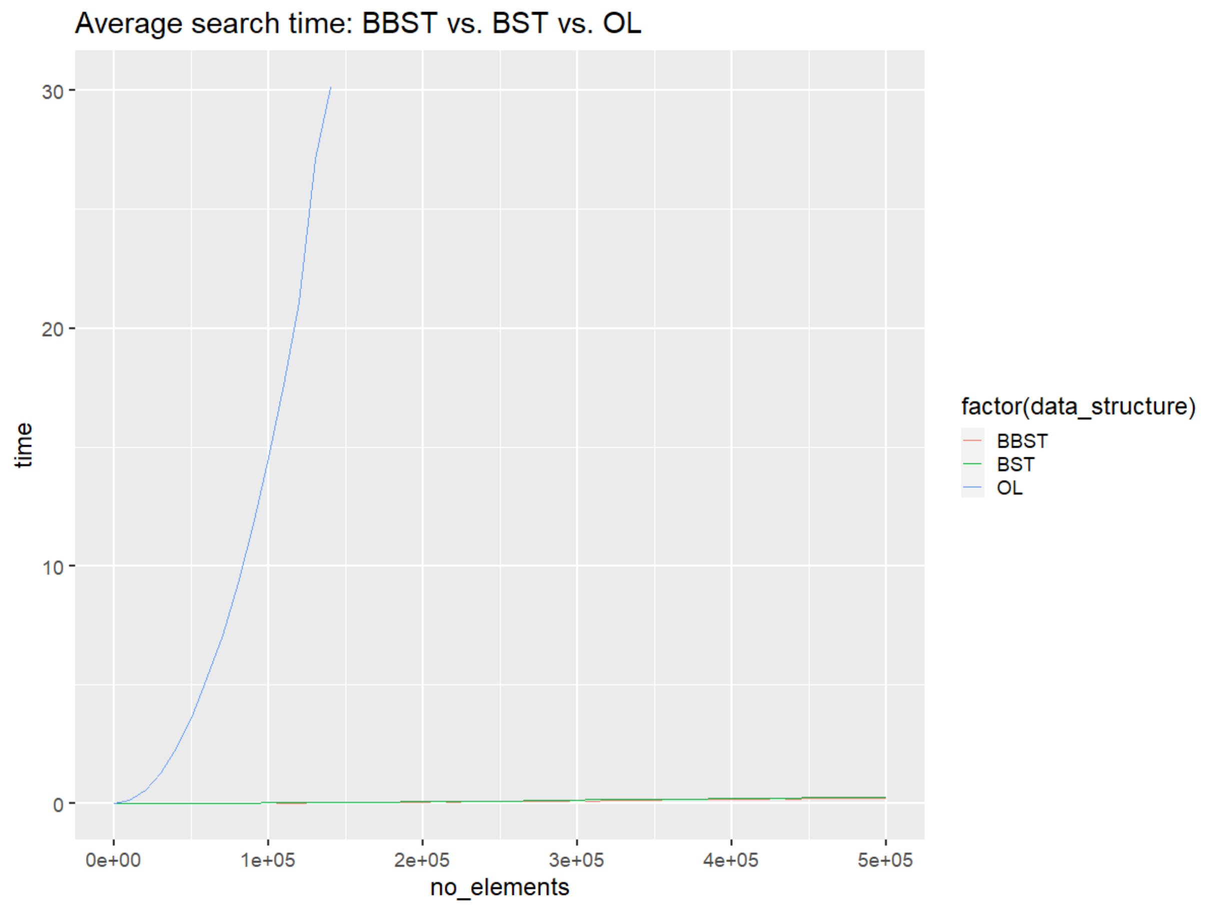


Conclusions:
The average times of inserting new records to the Binary Search Tree and Balanced Binary Search Tree are quite similar and much better than the average time of adding new records to the Ordered List. It is because of the tree structure of BST and BBST. In average time complexity of adding a new element to one of those two structures of size n, equals O(log(n)). So what is the difference between them? The unbalanced structure of BST makes it not optimal - in the worst-case scenario BST could look like an OL, then the average time of adding a new element would be O(n). On the other hand, to create BBST we must first sort the data, which also takes some time.

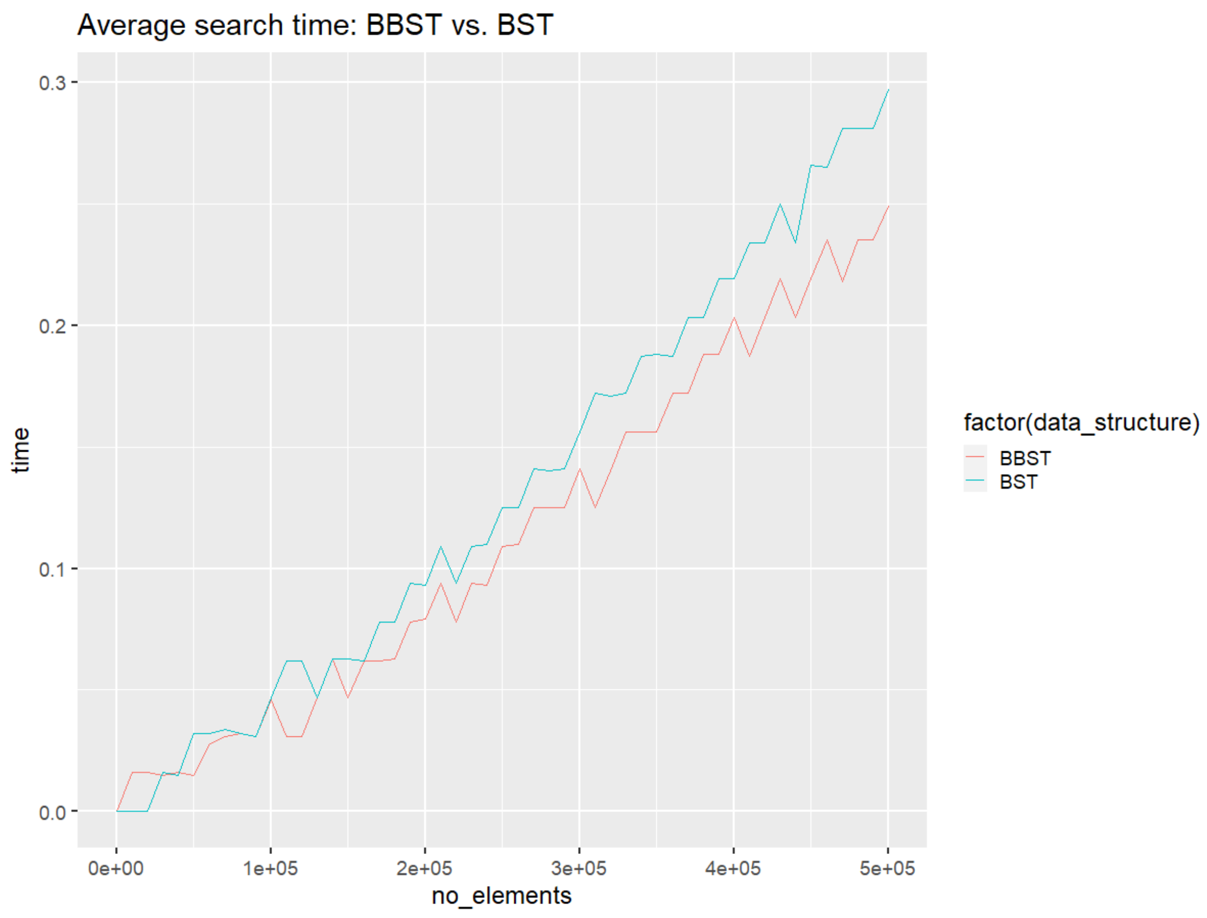
Dependence of search time of the number n (OL vs. BST vs. BBST)

Data:
The average time of searching the data in the following data structures was measured for random sets of data, consisting of 0, 10000, 20000, …, 500000 indexes (BST and BBST) and 0, 10000, 20000, …,150000 indexes (OL)

Charts:



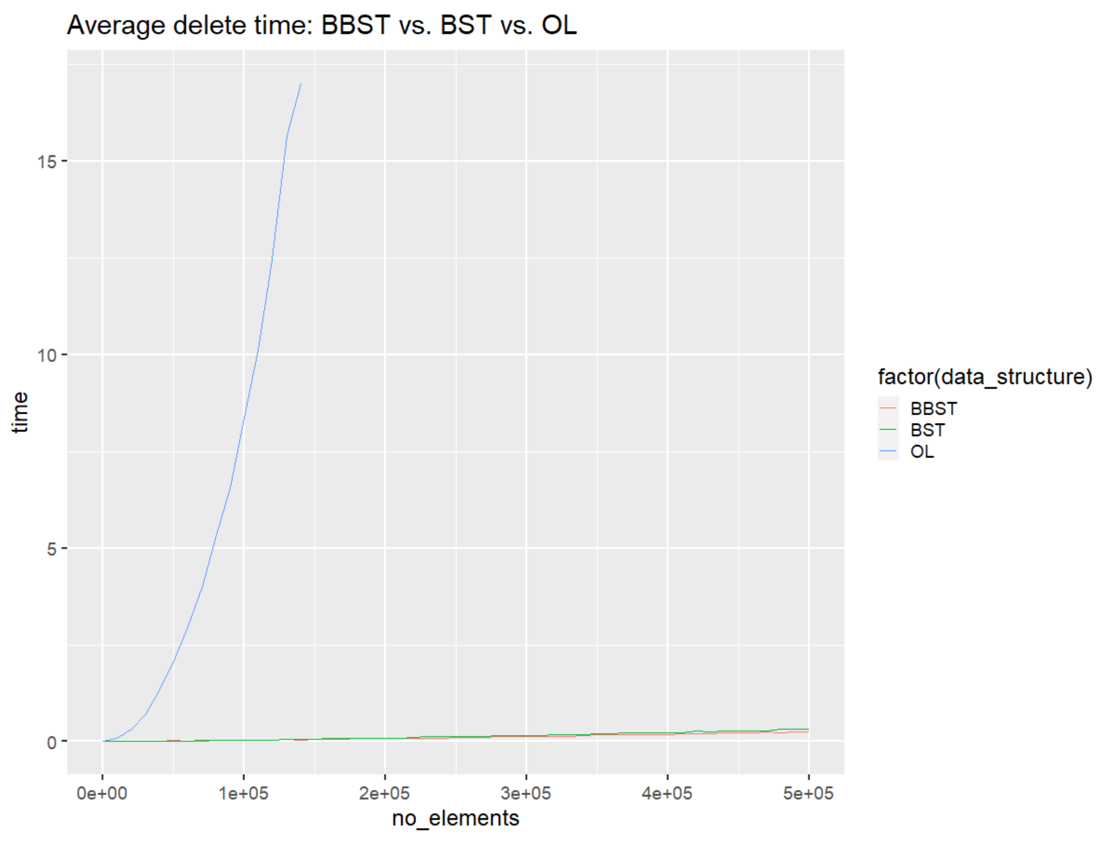
Conclusions:
The average time of searching for a record in the Ordered List is much worse than the average time of searching for an element in the BST or BBST. To perform a search operation on the OL, we must check separately each element one by one, so the average time complexity of this procedure is O(n). Searching for an element in BST and BBST is much faster because we only check some part of the elements (in average case log(n)), what's more, the balanced structure of BBST guarantees, that also in the worst-case scenario the number of checks will equal log(n). It can be seen on the chart below, that the searching operation performed on BBST is slightly faster than on BST.

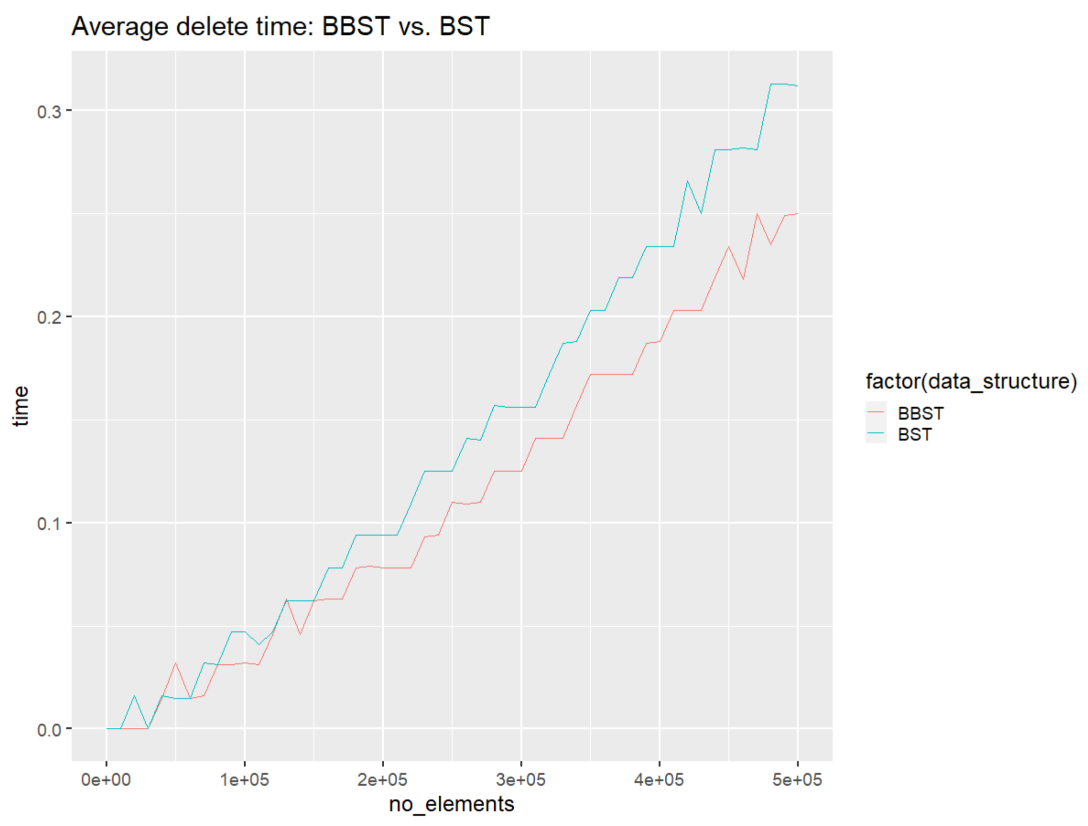


Average time of deleting an element from the structure (OL vs. BST vs. BBST)

Data:
The average time of deleting the data from the following data structures was measured for random sets of data, consisting of 0, 10000, 20000, …, 500000 indexes (BST and BBST) and 0, 10000, 20000, …,150000 indexes (OL)

Conclusions:
The charts presented above are quite similar to those concerning average time of searching for an element: deleting an element from the OL consumes much more time than deleting the same element from the BST or BBST, which performed the best.


Average delete time: BBST vs. BST vs. OL
time
15
10
5
0
0e+00
1e+05
2e+05
3e+05
4e+05
5e+05
no_elements
factor(data_structure)
BBST
BST
OL


Average delete time: BBST vs. BST
time
0.3
0.2
0.1
0.0
0e+00
1e+05
2e+05
3e+05
4e+05
5e+05
no_elements
factor(data_structure)
BBST
BST

Conclusions

Operations done on the Balanced Binary Search Tree are the least time-consuming. However, BBST requires more time to be created than BST.

The BST method of storing the data doesn’t require sorting elements before creating the tree, but search and delete operations performed on this structure are less efficient than on BBST. In the worst-case scenario, the time of searching for the element in BST is as long as for searching in the Ordered List.

The Ordered List structure is the most “naive” of all of them. Searching for an element in OL is the most time-consuming because in the worst case it requires checking all elements in the list.