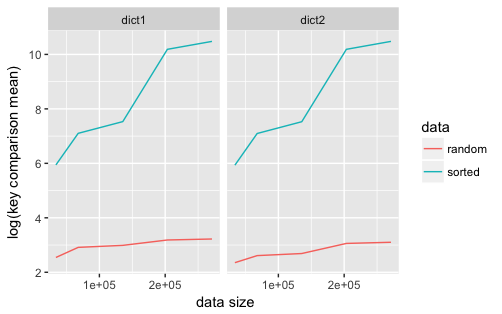
For the experimentation, data sets with 5 different data sizes were created: the same number as original data (271116 lines), three quarters (203337 lines), half (135558 lines), a quarter (67779 lines) and one eighth (33890 lines) of the original data. For each size, there are two data sets, one is sorted, and another one is randomised. Data analysis and visualisation of this report are conducted in R.

Looking at the mean of the number of key comparison for each data set with two different implementation of binary search trees provides the first insight into the correlation between the characteristic of data sets and the BST’s performance.

Figure 1: plot of data size vs. log(mean of number of key comparison)



As expected, the sorted data sets have a much higher mean of key comparison count for each key searched since in theory, a sorted data set creates a linked list rather than a balanced tree. Its growth of the mean is also much steeper than the shuffled data, since randomised data sets create BSTs that are more balanced and their search operation have a complexity of O(log(n)) while sorted data sets create linked lists and have a complexity of O(n), which grows much faster than log(n).

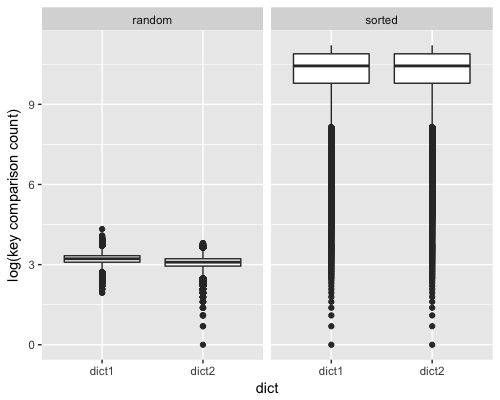


Figure 2: boxplot of key comparison count of the complete data set

Another distinctive difference between the sorted and shuffled data sets is the variance of key comparison count and it is the most noticeable in the data sets with all data. In figure 2, sorted data set’s range of key comparison count has a much bigger range, a higher number of outlier, and a higher variation. This is caused by the complexity of search operation. Since O(n) grows very fast, keys with lower alphabetical orders need to conduct a lot more key comparison comparing to the keys with higher alphabetical orders which are closer to the starting node of the linked list.

Figure 3: table of the mean of key comparison by different BSTs and data set sizes

In the design of the structure of the two BSTs, dict2 has an advantage that it needs less key comparison since data with the same key are not inserted in either left or right branch, rather a linked list is created and to access the linked list, key comparison is not needed after the first node with the desired key is found. However, for sorted data sets, the number of key comparison grow so much faster and become so big that this advantage of dict2 become insignificant and the difference between the means of key comparison of dict1 and dict2 are very close as it can be observed in figure 3. For the data sets with the most lines, the two BSTs even have the same mean (35541.9201).

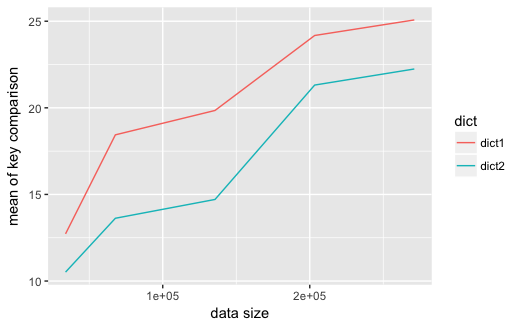


Figure 4:plot of data size vs. mean of key comparison of randomised data sets

Nonetheless, the advantage of dict2 is still visible in implementing BSTs for randomised data as the depth of the tree is much lower than the trees created by sorted data. This advantage can be more significant if there are a lot more data with the same keys in a randomised data set.

In conclusion, the most significant variations observed in the number of key comparison are caused by the characteristic of the data set (i.e. random or sorted). This highlight the importance of processing data before building a BST since a simple operation of randomisation can optimise the complexity of search operation, increase efficiency and make use of the data structure’s advantage.