BLG35E – Analysis of Algorithms 1, Fall 2017

Project 3

Yunus Güngör 150150701

Compiling: compiled with command: `g++ main.cpp -std=c++11`

a. Comparing Insertion and Lookup Processes for Dictionary and List Structure

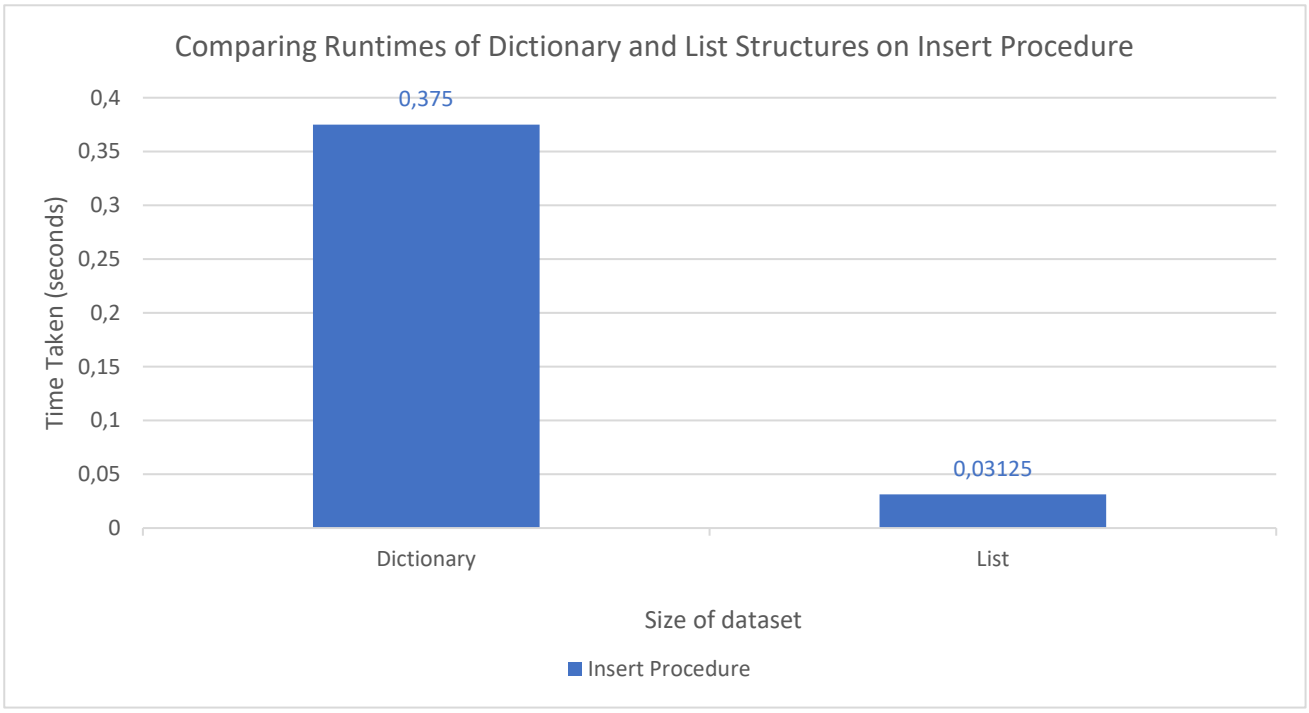

Figure 1: Comparison of runtimes for insert procedure.

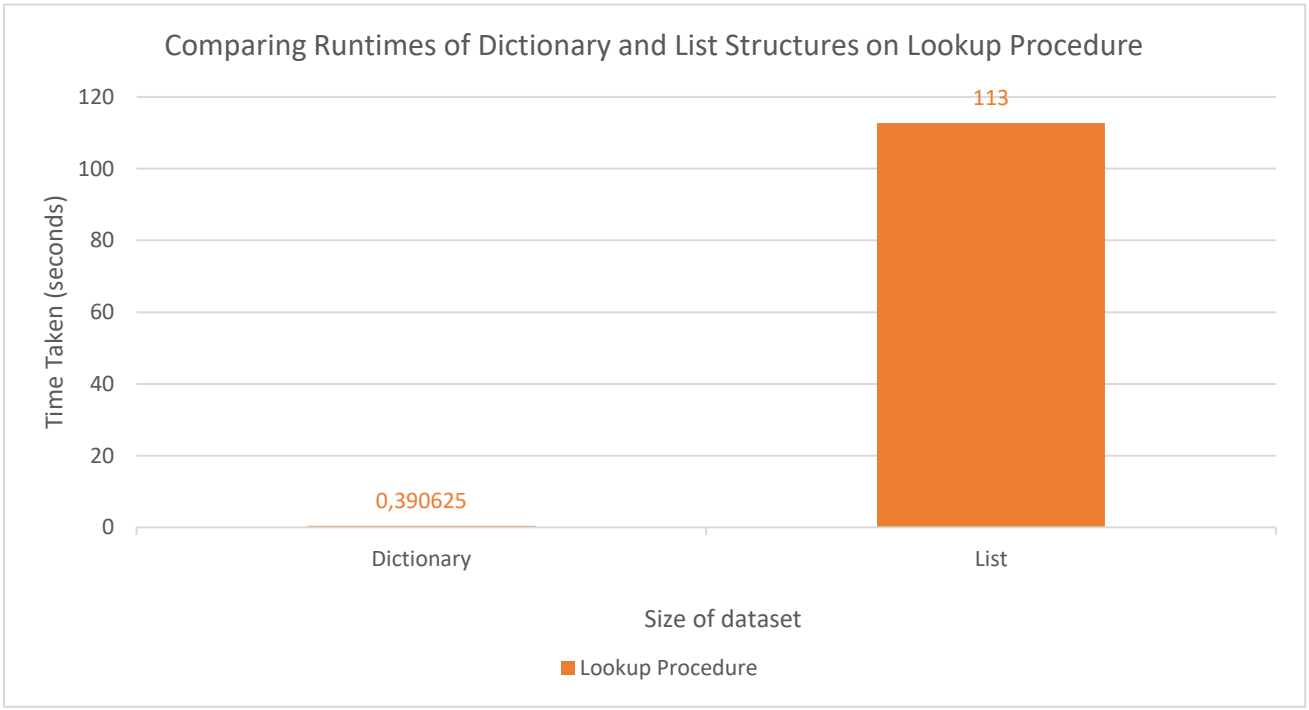

Figure 2: Comparison of runtimes for lookup procedure.

b. Analysis of data

Graphs in figure 1 and figure 2 shows that, list structure has an advantage while inserting data to structure. However, searching for a key requires much more time in list structure. Dictionary structure has a clear advantage while acquiring required data.

List structure is 10 times faster while performing insert procedure. Because dictionary structure has conflicts while inserting which requires more calculation. List structure just adds new data to end of the structure.

Dictionary structure approximately 300 times faster while performing lookup procedure. Because list structure must traverse on each data it has, until required data found. List structure's look up process has $O(1)$ in best case, $O(\frac{n}{2})$ in average and $O(n)$ on worse case. However, dictionary structure usually doesn't have to traverse that much. Dictionary structure has $O(1)$ for both lookup and insert in theory considering that no collisions occur. But consider that dictionary structure has conflicts that increases required time. This conflicts much more less than n or $n/2$, but more than 1 . Therefore, Dictionary structure is much faster than list structure while performing lookup, and slower while performing insert.

Theoretical complexity for directory is not applicable to this experiment because hash table size and dataset size are the same. This situation increases conflicts and causes more processing to be done. Theoretical complexity for list is applicable to this experiment, because there is no more process required.

c. Collisions

Average number of collisions increases exponentially as more items are inserted into directory. This means that, as much as items inserted into the directory, slower the insert procedure gets. Collision count increases because empty space on the hash table decreases as more items inserted. Less space available for new items increases possibility of a collision with an existing item. Increments in collisions can be observed in figure 3. Consider that graph in figure 3 is logarithmic and shows exponential increment of collisions.

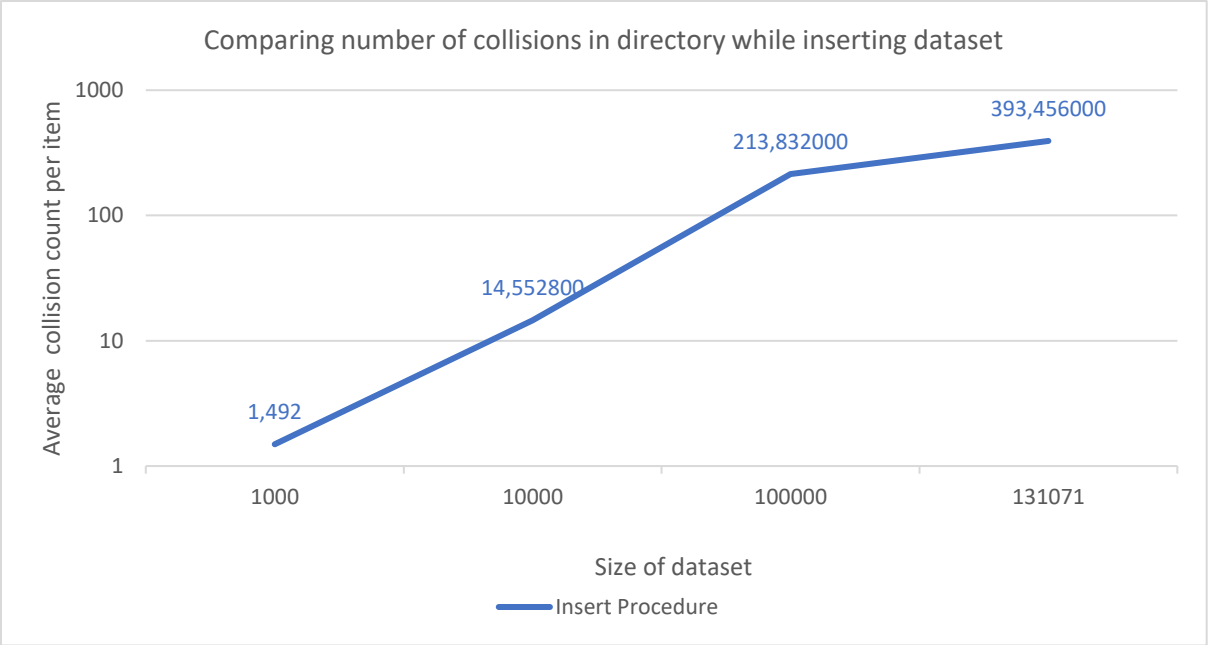

Figure 3: Comparison of average collision number per item. Consider that graph is logarithmic.

d. Worst Case for Looking Up in Dictionary

Worst case while performing lookup procedure in dictionary is the case where maximum number of collisions happened. Maximum number of collisions can be as much as N which is size of the hash table. In this case complexity of the procedure would be $O(n)$. Calculating next hash each time a collision occurred is slowing the process at most. Using a bigger hash table or using a more evenly distributed hash function might help eliminating worst cases.