BLG 335E - Analysis of Algorithms I

Project 3 Report

Onat Şahin - 150150129

Compilation instructions: My code requires C++ 11 since I used the stoi() function to convert strings to integers. Therefore the compilation command should be "g++ -std=c++11 main.cpp structures.cpp"

1) The values here are the results of running the program in the ssh server.

	Dictionary	**List**
Block Insertion	3.32 seconds	0.01 seconds
Block Lookup	3.59 seconds	41.61 seconds

2) There is a noticable difference between the performances of the two data structures. Performance of the dictionary is higher for block lookup. However, when it comes to block insertion, list is much more efficient. These results are inline with the complexities of these structures. The average complexity of dictionary for both insertion and lookup is $O(1+\alpha)$ while for list, the average complexity of insertion is $O(1)$ and the average complexity of lookup is $O(n)$.

3) The average number of collisions increase as more items are inserted in the dictionary. The increase is close to linear. The average number of collisions increase because as more items are inserted, the amount of empty spaces in the hash table decreases. Therefore, the probability for a collision increases. Since every insertion decreases the number of empty spaces by one, the increase in the average number of collisions is linear.

4) If the hashing and/or key generations are badly written, every key might be hashed to the same location. This would be the worst case for the dictionary. In this case, in every insertion, hashing and probing must be done for every other location occupied by a character. The corresponding time complexity for this case is $O(n)$. In this case, hashing and probing become slow enough to dominate the complexity since it is done so many times.