**Hash map:**

The difference of time complexity between these two algorithms are almost the same, while the size of data is increasing, the first algorithm needs longer time to solve the problem.

In the first algorithm, a while-loop is used to read the data, but the difference with the second algorithm is that the split method is used in this algorithm. The space complexity of using split() instead of using for-loop is O(m+n). The time-complexity for reading the dataset should be O(4n\*n). We expect it needs longer time to contribute the hash map when the size of data is large. In the algorithm, one while-loop is needed to read and store the dataset. In the function for searching abbreviations, the split method is also used. The time-complexity for this part should be O(4m).

In the second algorithm, we use a while-loop to read the data in the dataset, and then read the individual abbreviations and full names through two for-loops that is not tessellated with each other. Comparing to the first algorithm, it’s more efficient than algorithm that is using split() method. The space complexity of using for-loop is O(1). The time-complexity for this part should be O(3n\*n). The read data is then stored in a hash map via StringBuilder. The punctuation is first removed from the sentences entered by the user, and then a for-loop is used to search for abbreviations in the input. Finally, replace the abbreviation with the selected full name. The time-complexity for this part should be O(3m).

Therefore, when the data structure is the same, the first algorithm needs more time to solve the same problem than the second algorithm needs.

**Array:**

Using array to contribute the dataset takes longer time than using hash map. When the first algorithm using array, it takes longer time than it using hash map. Therefore, different data structure will change time running time of this algorithm significantly.