Report

1. Pseudo codes for methods add, and search(Tuple t) methods from HashTable.

Add(Tuple t)

*Get and save hash for tuple key  
In the hash table array, go to the hash position - arr[ hash ]  
If arr[ hash ] is null, create a new array at arr[ hash ] and add the tuple to the list  
Otherwise add the tuple to the front of the list.  
Increment the remembered number of elements*

Search(Tuple t)

1. Derive and state asymptotic run-times of above methods. Express run-time as a function of n, and k, where n is the number of elements on the Hash Table and k is the length of the String.
2. For each of the classes BruteForceSimilarity, HashStringSimilarity, and HashCodeSimilarity

* Describe the data structures used.
* Pseudo code for all three methods
* Derive and state asymptotic run-time of all three methods. Express run-times as functions of n, m and k. Where n and m are lengths of the strings and k is the shingle length parameter.

1. You are provided two test files. Convert the files into strings and run the method similarity from all three classes. Use 8 as shingle length. Report the similarities returned and the run-times.

* Brute Force
  + Similarity: 0.41189092
  + Time (seconds): 109.786
* Hash Code
  + Similarity: 0.31048927
  + Time (seconds): 8.407
* Hash String
  + Similarity: 0.41189092
  + Time (seconds): 49.324

1. Compare all three run times. Which one is smallest? Which one is largest? Does one run time equal (or very close to) another run time? Explain why one run-time equals (or very close) /smaller/larger than the other run times.

The hash code function runs with the least time. The Brute Force function takes the longest.

1. Do all three methods return the same value for similarity? If not, explain the reason.