Thomas “Andy” Archer

CSC 316

Project 4

There are two files for the submission: Proj4.java and Node.java.

Node.java contains a simple node structure to implement a singly linked list. It stores the string for the word entered into the dictionary, and it stores a link to the next word in the same bucket.

Proj4.java contains the main method and should be the file executed in order to run the program.

Compiling was tested on the EOS system through storing the Proj4 and Node files in the same directory, then compiling them together using openJDK and the command javac \*.java.

The program prompts the input of file names from the user through the terminal window. This includes the name of the dictionary file, then name of the file to be scanned, and finally the desired name of the output file.

The hashing formula used was one discussed in class:

In doing research for the hashing function, I came across a forum posting where someone mentioned using an r of 37 for a dictionary of 50,000 entries produced only 7 collisions. Unfortunately, I was not able to find this forum entry again to properly cite this source.

This was implemented using a for loop from 0 to the length of the string with the function:

hash += hash \* r + CharacterValue

Having determined the hash function and knowing the dictionary size, I worked backwards to find m based on an estimated load factor of .9. This gave an m of 27,940. While it is not a prime number, it does not divide evenly by 37.

Since a low number of collisions are expected, an array of Nodes were created to create a separate chaining table. The first value for the bucket is stored directly in the table, which also starts the linked list for each bucket. That Node then links to the other items in the bucket, if there are any.

Testing has normally produced probes per word checked at around 1.4 and probes per lookUp operation at round 1.2.