Surabhi Ravishankar
Email: surabhiravishankar2016@u.northwestern.edu
EECS 395 – Data Structures and Data Management

Alternative Program 4 Report

Execution:

```
$ make
$ ./wordladder
```
(The input word list is hard coded in the program)

Design and Implementation:

Implementation of Breadth First Search Algorithm:

I used the following pseudo code to implement the BFS algorithm. First I enqueue a vertex. If the word sought is found in this vertex, quit the search and return the resultant vector. Else, I enqueue any adjacent vertices that have not yet been discovered. If the queue is empty, every node on the graph has been examined – quit the search, else repeat the steps and store the path to the target word. This was a direct implementation of the BFS. The function to perform BFS is written in graph.h. The queue implementation is in queue.h.

Please refer to the comments in the code for a more detailed explanation of the design decisions.

Implementation of the graph:

I have maintained an adjacency list to store all the vertices in the graph. Each vertex in the graph represents a word. The adjacency list stores a key value pair where the key is a single word and the value is a list of words that vary only by one word. A function to compare each and every letter of a word to check the number of characters they defer by was written.

The Breadth first search is implemented on this adjacency list. All the functions to implement the graph and the searching are defined in the graph.h file. The graph class contains all these functions.

A detailed explanation of all the functions is written in the code. Please refer to it.

Finally, the program works by reading the inputs from the "Program4AlternativeFall2014words.txt". The file name is hard coded into the program. Then the word graph is created, after which the shortest path is found using BFS.