Status report

This report is written for the Algorithmics assessed exercise. Two different programs had to be implemented, first one being the wordladder algorithm and the second one being the dijkstra algorithm. The programs both compile just fine with no errors. I am confident that the word ladder gives out the correct results and I believe I have managed to produce an efficient code. The dijkstra algorithm gives out the expected results in certain of the test data, where in the case of some other tests I am not sure if they are the expected results. My main consideration about the dijkstra code is that what I have written seems too long to me, and if I had more time I may have been able to make it shorter at least.

Implementation report

- In order to write the dijkstra algorithm, I tried to implement the algorithm from the lecture that was given to us. Firstly I wrote down what I expected from the algorithm to do, which looked like this:

  dijkstras algorithm //program takes start word and end word as parameters //program locates vertex v of startword //program locates vertex u of endword //store v in S array, attribute in vertex class //declare distances array // set distance to infinity for everything except the start vertex, set predecessor to be the start vertex //for each vertex x in adjacency list of v //compute distance i of x from v //store i in distance array // while S not empty //find minimum value in distance array not in S with min function //add it to S //while vertex y is not the end word //for each vertex y in adjacency list of v and not in S //perform relaxation function //add result in distance array

  The actual algorithm is not working excactly as the text above, but it follows the same logic. For example the relaxation function is not a different function but a part of the dijkstra algorithm. It may have been better if I broke it down to more functions which may have made it easier to read. I have created though an external function for detecting the characters that differs between two words. In addition several functions were created in the Vertex class, for handling the name of a Vertex and a distance variable for distance between vertices.

- For the wordladder, I once again made sure that I fully understood the material provided by the lectures. I used the material that was given to us during the lab session and extended it to solve the problem. I once again wrote down what I expected from the algorithm to do as I did in the dijkstra algorithm. Unfortunately though I lost the description of the algorithm that I used. Functions were also added in the Vertex.java file for handling the distance from a Vertex's predecessor. The Main.java file was written first for this problem and was slightly modified and used again in the dijkstra algorithm.

Empirical results

The text below are the command line arguments along with the results for the wordladder algorithm. The elapsed time is between 85-95 milliseconds which I believe I am satisfied with.

```
java Main words5.txt greed money No ladder. Elapsed time: 94 milliseconds

java Main words5.txt print paint Distance is 1. Elapsed time: 85 milliseconds

java Main words5.txt smile frown Distance is 12. Elapsed time: 90 milliseconds

java Main words5.txt forty fifty Distance is 4. Elapsed time: 87 milliseconds

java Main words5.txt black white Distance is 8. Elapsed time: 90 milliseconds

java Main words5.txt greed money No ladder. Elapsed time: 95 milliseconds

java Main words5.txt worry happy No ladder. Elapsed time: 85 milliseconds
```

The text below are the command line arguments along with the results for the dijkstra algorithm. The elapsed time once again is between 85-95 milliseconds which I believe I am satisfied with, but as I did not always get the expected results, I know that there is something wrong with the code but did not have enough time unfortunately, to figure out what it was.

```
java Main words5.txt blare blase Distance is 1. Elapsed time: 83 milliseconds

java Main words5.txt blond blood Distance is 1. Elapsed time: 83 milliseconds

java Main words5.txt allow alloy Distance is 2. Elapsed time: 84 milliseconds

java Main words5.txt cheat solve No path Elapsed time: 84 milliseconds

java Main words5.txt worry happy No path Elapsed time: 84 milliseconds

java Main words5.txt print paint No path Elapsed time: 86 milliseconds

java Main words5.txt small large No path Elapsed time: 83 milliseconds

java Main words5.txt black white No path Elapsed time: 84 milliseconds

java Main words5.txt greed money No path Elapsed time: 84 milliseconds
```