Algorithmics I – Assessed Exercise

Status and Implementation Reports

Peter Dodd
2308057D

November 16, 2020

Status report

I believe both my programs to be fully functional. With ladders that were possible to complete by hand, the programs both gave expected results, and they only didn't compute when a ladder wasn't possible. Both programs also successfully compiled from the command line. I think that I could've improved the efficiency of which the graphs were created, but I believe the ladder algorithms themselves to suffice.

Implementation report

(a) To create the graph, I stored the data in an ArrayList structure, then used that to input the data into the graph. I believe that this was an efficient method, as the constructor was able to take the ArrayList as an argument, so therefore could process all the information at once.

I implemented a breadth-first traversal in order to find the shortest distance between each point on the graph. I did this as it is a typical algorithm used to find shortest paths between vertices on an undirected, unweighted graph. This was done by firstly adding the vertex representing the start word to an initialised queue. From there, the first vertex in the queue is selected and removed, and the program iterates through each unvisited, adjacent node for that vertex. Each of these is added to the queue, with its predecessor set to the selected vertex. Once that is completed, the process repeats for the next vertex in the queue. This is repeated until either all possible vertices have been visited and the queue is empty, or the end word is found. If the latter is true, then the queue ends at that point, as the target vertex has been found, and thus, the shortest path to it.

(b) I implemented the graph for this program in much the same way, although a variable weight was added for nodes in adjList for each vertex.

I implemented dijkstras algorithm with the use of two ArrayLists, settled and unsettled. The former represents all vertexes that have been visited as the main evaluation vertex, and the latter represents vertices with a distance that have yet to be evaluated. Each vertex is only evaluated once, although can be assigned distance values before that. Initially, the vertex representing the start word is added to the unsettled list. Then, the smallest vertex with the smallest distance value in the unsettled list is selected, and removed from the list. The program then iterates through the nodes in adjList for the vertex that aren't in settled, and

calculates the minimum distance and path for that node, based on its source vertex. If the vector representing the node isn't already present in unsettled, then it is added. Once this loops ends, the selected vertex is then added to settled, and the process repeats. Although the use of two ArrayLists may improve readability and ease of understanding, it might create space problems when handling much larger dictionary files. However, I believe it worked well for the data supplied in this task.

Empirical results

This section is part of the marking scheme "Outputs from test data: 2 marks".

Program 1

```
################################
Word Ladder Finder 1

Start Word: print
End Word: paint

Ladder found in 1 steps!

paint <-- print

Elapsed time: 310 milliseconds
################################

################################
Word Ladder Finder 1

Start Word: forty
End Word: fifty

Ladder found in 4 steps!

fifty <-- fifth <-- firth <-- forth <-- forty

Elapsed time: 335 milliseconds
################################

################################
Word Ladder Finder 1

Start Word: worry
End Word: happy

No ladder found.

Elapsed time: 315 milliseconds
```

```
################################


################################
Word Ladder Finder 1

Start Word: smile
End Word: frown

Ladder found in 12 steps!

frown <-- crown <-- croon <-- crook <-- crock <--
clock <-- click <-- slick <-- slice <-- spice <--
spite <-- smite <-- smile

Elapsed time: 407 milliseconds
################################


################################
Word Ladder Finder 1

Start Word: small
End Word: large

Ladder found in 16 steps!

large <-- marge <-- merge <-- verge <-- verse <--
terse <-- tease <-- cease <-- chase <-- chasm <--
charm <-- chard <-- shard <-- share <-- shale <--
shall <-- small

Elapsed time: 391 milliseconds
################################


################################
Word Ladder Finder 1

Start Word: black
End Word: white

Ladder found in 8 steps!

white <-- whine <-- thine <-- trine <-- brine <--
brink <-- blink <-- blank <-- black

Elapsed time: 338 milliseconds
```

```
##################################


##################################
Word Ladder Finder 1

Start Word: greed
End Word: money

No ladder found.

Elapsed time: 338 milliseconds
##################################
```

Program 2

```
##################################
Word Ladder Finder 2

Start Word: blare
End Word: blase

Ladder found with a distance of 1!

blare --> blase

Elapsed time: 365 milliseconds
##################################


##################################
Word Ladder Finder 2

Start Word: blond
End Word: blood

Ladder found with a distance of 1!

blond --> blood

Elapsed time: 351 milliseconds
##################################


##################################
Word Ladder Finder 2

Start Word: allow
End Word: alloy
```

```
Ladder found with a distance of 2!

allow --> alloy

Elapsed time: 396 milliseconds
################################


################################
Word Ladder Finder 2

Start Word: cheat
End Word: solve

Ladder found with a distance of 96!

cheat --> chert --> chart --> charm --> chasm -->
chase --> cease --> lease --> leave --> heave -->
helve --> halve --> salve --> solve

Elapsed time: 409 milliseconds
################################


################################
Word Ladder Finder 2

Start Word: worry
End Word: happy

No ladder found.

Elapsed time: 375 milliseconds
################################


################################
Word Ladder Finder 2

Start Word: print
End Word: paint

Ladder found with a distance of 17!

print --> paint

Elapsed time: 361 milliseconds
```

```
##################################


##################################
Word Ladder Finder 2

Start Word: small
End Word: large

Ladder found with a distance of 118!

small --> shall --> shale --> share --> shard -->
chard --> charm --> chasm --> chase --> cease -->
tease --> terse --> verse --> verge --> merge -->
marge --> large

Elapsed time: 353 milliseconds
##################################


##################################
Word Ladder Finder 2

Start Word: black
End Word: white

Ladder found with a distance of 56!

black --> slack --> shack --> shank --> thank -->
thane --> thine --> whine --> white

Elapsed time: 373 milliseconds
##################################


##################################
Word Ladder Finder 2

Start Word: greed
End Word: money

No ladder found.

Elapsed time: 360 milliseconds
##################################
```