Boston University

METCS 526: Final Project

By: Aidan Duffy

Section I, Pseudocode:

**Algorithm One (G, exclude):**

Input: an undirected, weighted, connected graph G and a Node exclude, if needed, otherwise it is null

Output: the node *v* that, among all neighboring nodes of *n*, has the smallest return value of *dd(v)*, or *direct distance* to the destination node Z*.*

Minimum\_distance = Integer.MAX\_VALUE

Best\_node = null

**For** each node *v* which shares an edge with node *n* **do**:

**if** *dd(v)* < minimum\_distance and v is not exclude **then:**

minimum\_distance = *dd(v)*

best\_node = v

return best\_node

**Algorithm Two(G, exclude):**

Input: an undirected, weighted, connected graph G and a Node exclude, if needed, otherwise it is null

Output: the node *v* that, among all neighboring nodes of *n*, has the smallest return value of *dd(v)* + *w(n,v)*, or *direct distance* to the destination node Zplus the weight of the edge shared by node *n* and node *v.*

Minimum\_distance = Integer.MAX\_VALUE

Best\_node = null

**For** each node *v* which shares an edge with node *n* **do**:

**if** *w(n,v) + dd(v)* < minimum\_distance and v is not exclude **then:**

minimum\_distance = *dd(v) + w(n,v)*

best\_node = v

return best\_node

Both of these algorithms will run iteratively until the current node *n* is Z.

Section II, Data Structures Used:

I created two new classes, a Node and a Graph.

For the Node, it contained:

1. **identifier:** a character for its letter value (ex: ‘A’, ‘B’,…, or ’Z’)
2. **neighbors:** a HashMap where the key is the Node that neighbors this Node and the value is the weight of their shared edge
3. **direct\_distance:** an integer which is the Node’s direct distance to destination Node Z.

For the Graph, it contained:

1. **nodes:** an ArrayList of all of the Nodes (or vertices) in this graph
2. **edges:** a HashMap of all of the edges in this graph where the key is one Node, and the value is another Node; the construction ensures no duplicates exist (ie. A:B and B:A will not both be in there)
3. **position:** a Node which is the current position of our graph traversal

In my project file, it contained:

1. **characterNode**: a HashMap with the character ID as the key and the actual Node as the value
   1. this was used for the two setup methods, Graph.constructGraph and Node.setDirectDistances since the input files contain just the character with the associated node.
2. **path**: an ArrayList which stores the characters that shows the full path from the given algorithms
3. **shortestPath:** an ArrayList which stores the characters of the nodes that shows the final, shortest/ideal path from the given algorithms