/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Name: Abigail Payne

\*

\* Partner Name: Elaine Schutte

\* Partner NetID:

\*

\*

\* Hours to complete assignment (optional):

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**\* Describe concisely the data structure(s) you used to store the**

**\* information in synsets.txt. Why did you make this choice?**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

We decided to use a Hash map to keep track of connections within

We also decided to use an ArrayList to keep track of multiple inputs that correspond to one key. The main reason for using this instead of a HashSet is that, although HashSets prevent duplicates the data being inputted won’t contain duplicates. Considering the data inputted is already sorted, using an ArrayList then increases the range of functionality by allowing the use of binary search if needed.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**\* Describe concisely the data structure(s) you used to store the**

**\* information in hypernyms.txt. Why did you make this choice?**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ We had originally used another hash map to keep track of these connections as well but realized that it was possible to directly input the data into a directed graph. Seeing as the digraph was already required for the project we decided to only use this data structure to save space and time.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**\* Describe concisely the algorithm you use in the constructor of**

**\* ShortestCommonAncestor to check if the digraph is a rooted DAG.**

**\* What is the order of growth of the worst-case running times of**

**\* your algorithms as a function of the number of vertices V and the**

**\* number of edges E in the digraph?**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Description:

The constructor of the ShortestCommonAncestor copies the graph argument G to a private Digraph. If the graoh is null or have no verticies, we throw a NullPointerException or IllegalArgumentException, respectively

Order of growth of running time: Constant

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*  **Describe concisely your algorithm to compute the shortest common**

**\* ancestor in ShortestCommonAncestor. What is the order of growth of**

**\* the running time of your methods as a function of the number of**

**\* vertices V and the number of edges E in the digraph? What is the**

**\* order of growth of the best-case running time?**

**\***

**\* If you use hashing, you should assume the uniform hashing assumption**

**\* so that put() and get() take constant time.**

**\***

**\* Be careful! If you use a BreadthFirstDirectedPaths object, don't**

**\* forget to count the time needed to initialize the marked[],**

**\* edgeTo[], and distTo[] arrays.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Description:

First we created a BreadtFirstDirectedPath for v and w. Then for every vertex in in the graph we see if both v and w have a path to it. Then we check to see if the distances between v and w to that vertex is the smallest on yet. If it is we then update the shortest common ancestor. We finally return the shortest common ancestor.

running time

method best case worst case

------------------------------------------------------------------------

length(int v, int w) Θ(V+E) Θ(V+E)

ancestor(int v, int w) Θ(V+E) Θ(V+E)

length(Iterable<Integer> v, Θ(V+E) Θ(V+E)

Iterable<Integer> w)

ancestor(Iterable<Integer> v, Θ(V+E) Θ(V+E)

Iterable<Integer> w)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***\*\***

**\* Known bugs / limitations.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

There are no known bugs in the code.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**\* Describe whatever help (if any) that you received.**

**\* Don't include readings, lectures, and precepts, but do**

**\* include any help from people (including course staff, lab TAs,**

**\* classmates, and friends) and attribute them by name.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. <https://docs.oracle.com/javase/7/docs/api/java/util/package-summary.html>
2. <http://www.java67.com/2013/01/difference-between-set-list-and-map-in-java.html>
3. <https://introcs.cs.princeton.edu/java/stdlib/In.java>
4. <https://algs4.cs.princeton.edu/42digraph/Digraph.java.html>
5. <http://www.java67.com/2014/01/how-hashset-is-implemented-or-works-internally-java.html>
6. <https://stackoverflow.com/questions/3920602/get-specific-arraylist-item>
7. <https://stackoverflow.com/questions/2559527/non-static-variable-cannot-be-referenced-from-a-static-context>
8. <https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**\* Describe any serious problems you encountered.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

In the checklist/FAQ it says that we are not allowed to make call to the constructor of ShortestCommonAncestor. To deal with this restriction we MUST make the class (and/or) the methods static. However, this violates the API given to us in the project guidelines.

We have chosen to follow the API given in Programming Assignment 3 so that there are no compatibility errors at the expense of some efficiency.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**\* If you worked with a partner, assert below that you followed**

**\* the protocol as described on the assignment page. Give one**

**\* sentence explaining what each of you contributed.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Elaine: created rough code at the beginning, researched data types for code, javadocs

Abigail: Unit testing, finished and refined project code, general documentation, javadocs, integration testing

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* List any other comments here. Feel free to provide any feedback**

**\* on how much you learned from doing the assignment, and whether**

**\* you enjoyed doing it.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/