COMS 3110: Homework 4
Due: November 18^{th}, 11:59pm
Total Points: 80

Late submission policy. An assignment that is submitted one day late will get a penalty of 10%. An assignment that is submitted two days late will get a penalty of 20%. Assignments submitted after two days will not be graded and will get no points. For example, if an assignment is due on Friday by midnight, a submission on Saturday will be penalized by 10%, and a submission on Sunday will be penalized by 20%. A submission on Monday will get no points.

Submission format. Your submission must be a `zip` file consisting of only all necessary source files and any other files explicitly mentioned. You should *not* include compiled files, such as those ending in `.class`, unless otherwise explicitly stated. You should *not* include other files such as project and IDE configurations. Name your submission file: `<Your-net-id>-3110-hw4.zip`. For instance, if your netid is `asterix`, then your submission file will be named `asterix-3110-hw4.zip`. Your submission `zip` should unpack to folder called `asterix-3110-hw4`. Inside that folder should be a `src` folder containing the source code for the assignment. All `java` source files should be in a folder hierarchy that matches the sources' packages. For example, if you are submitting `MyCode.java`, and it is in the package `course.assignment`, your submission should unzip to `./asterix-3110-hw4/src/course/assignment/MyCode.java`. Each student must submit their own assignment. If you discussed the homework or solutions with others, a list of collaborators must be included with each submission. Each of the collaborators has to write the solution(s) themselves (copies are not allowed).

General Requirements

- When proofs are required, do your best to make them both clear and rigorous. Even when proofs are not required, you should justify your answers and explain your work.
- When asked to present a construction, you should show the correctness of the construction.

0 Preamble

This description of a programming assignment is not a linear narrative. It may require multiple readings before things start to click. You are encouraged to consult the teaching staff for any questions/clarifications regarding the assignment. Your programs must be written in **Java**.

1 Overview

Your friend just uncovered a trove of lost manuscripts during an archaeological dig at the site of an ancient library. Amongst these were various chronologies of ancient cities and villages in the region. Upon examination, your friend observed potential inconsistencies with the dates given for some of these communities. The documents are to be digitally restored and cataloged. Even so, the sheer volume of data makes analyzing these inconsistencies by hand a difficult task. The friend has asked you to help with this task.

1.1 Details

Access to the raw, restored, digital documents is currently restricted. You are, however, provided text files with sufficient information for you to find any discrepancies. The files contain statements of two kinds. These statements are found in the second and third segments of the file, respectively. The first is "city A and city B both existed at the same time". Such a statement is included even if they were said to have coexisted for only a mere moment. The second is that "city B was established after the fall of city A". Here, the fall of the earlier city is taken to mean that it no longer existed and the establishment of the later city means that it began to exist.

The format of the file is given below. It consists of 3 segments. Each one begins with a single integer specifying the number of data points in the segment. Each statement is newline terminated. Different entries on the same line are whitespace separated. City names do not contain any whitespace or non-printable characters. The final line may or may not be terminated by a newline character. The first segment of the file is the set of all cities present in the original data. Any city named in the latter two segments will be present in the first segment, although there may be cities listed in the first segment that do not appear in the following two.

```
<number of cities: integer>
<city 1: string>
<...>
<city n: string>
<coexistence statements count: integer>
<city name: string> <city name: string>
<...>
<city name: string> <city name: string>
<exclusive statements count: integer>
<earlier city name: string> <later city name: string>
<...>
<earlier city name: string> <later city name: string>
```

You've decided that you can use your knowledge of graph algorithms to solve the problem, and you intend to design a general graph class to assist you in your task. You will also

implement a separate class for analyzing the data. It will be responsible for reading in the data from a file and utilizing your graph class to determine the consistency of the data. The data is said to be consistent, or free of conflict, if there are no contradictions within it. A trivial example of a conflict is "city B first existed after the passing of city A" and "city A and city B both existed at the same time ". But, due to the sheer volume of data you will be processing, it is important that your algorithm is efficient.

2 Implementation

You are required to submit two Java classes, `Graph` and `Analyzer`. They are described below. They must be in the package `coms3110.hw4`, and no other subpackage or superpackage. None of the described functionality in these classes should throw a checked exception unless mentioned explicitly in this document.

2.1 Graph Implementation

The `Graph` class is generic and contains many public methods providing basic graph functionality.

- A constructor that takes a `Collection` of `T` as input. Each string is the name of a vertex in the graph.
- A method `addEdge` takes two `T` arguments. If either argument is not a pre-existing vertex in the graph, or if the edge already exists in the graph, the method returns false. Otherwise, it adds a directed edge from the first argument to the second argument and returns true.
- A method `vertexCount` that takes no arguments and returns an `int` equal to $|V|$.
- A method `edgeCount` that takes no arguments and returns an `int` equal to $|E|$.
- A method `reachable` that takes to two `T` parameters and returns a `boolean`. It returns true if

  1. Both parameters are vertices in the graph, and
  2. The second given vertex is reachable from the first.

  It returns false in any other case.
- A method `bfs` that takes a single argument of type `T`. It returns a `Map` from the vertex objects (`T`) to the layer they belong to (`Integer`) resulting from a breadth-first search starting at the input parameter. All vertices of the graph should be present as keys in the map. The value for unreachable vertices should be -1. If the starting vertex is not present in the graph, it returns `null`.

- A method `diameter` returning `int`. This method returns the diameter of the graph held by this object. If there exist vertices $u, v \in E$ such that u cannot reach v, it returns -1. Otherwise, the value returned is the longest shortest path between any pair of vertices in the graph.

- A method `dfs` that takes no arguments. It returns a `Map` from the vertex objects (`T`) to their finish time (`Integer`) resulting from a depth-first search on the graph. The search should explore the entire graph.

- A method `hasCycle`. It takes no arguments, and returns true if the graph has one or more cycles. It returns false otherwise.

2.2 Analyzer

The `Analyzer` class is required to have a single static public method called `analyze`. It takes a single `String` argument indicating the path to a file containing the city data described earlier. It returns a `boolean` indicating whether the statements are free of conflict (true) or not (false). This class may throw an `Exception` if the file is not readable or malformed.

3 Miscellanea

- You are required to include a README file with your submission. It should include a brief description of how you used graphs to solve the given problem. If you use an LLM or other AI tools (e.g. chatgpt) to assist in writing your project, you are required to state this in the README, including any queries you posed.

- The classes you are required to implement may be tested independently of one another.

- You are not allowed to use any external libraries or JARs, especially those providing graph functionality.