Jim is desperate for coffee. However, Jim likes a special coffee that is a combination of ingredients, and these ingredients are located at various coffee shops around Ames. Jim needs your to help collect these ingredients and deliver them to him as quickly as possible.

Project Description

This is a programming project that must be written in Java and turned in via Blackboard. The project has two parts. In the first part, you will implement a general graph interface using an adjacency list representation. This interface also requires you to implement Dijkstra's shortest path, topological sort, and a minimum spanning tree algorithm.

In the second part, you will use your code from part 1 to find a quick routes through the city of Ames to help get Jim his coffee as soon as possible. You will also build a minimum spanning tree of Ames so that Jim can estimate the cost to build direct coffee access lines throughout the city.

We provide you with an encoded graph representation of the city of Ames obtained from Open Street Map for part 2.

Part 1: (200pts)

You must create a class that implements the `Graph` interface provided to you on Blackboard. The name of your class must be `CSGraph` and it must have a constructor with one `boolean` parameter named `isDirected` which determines if the graph is to function as a directed or undirected graph.

Your grade for this part will be based entirely on the correctness of your implementation of this interface. Please read the documentation of the interface carefully before starting.

The implementation of topological sort algorithm must run in $O(|V| + |E|)$-time, shortest path must run in $O(|V|^2)$-time, and the minimum spanning tree must run in $O(|E| \log |E|)$-time.

City of Ames Data

You are provided with a file named `ames.txt` which encodes the city of Ames for use in part 2 of the project.

The first line of the file begins with

```
VERTICES: n
```

where `n` is the number of vertices in the graph.

After this first line, the following `n` lines enumerate all the vertices of the graph. Each vertex in the file corresponds to a business or an intersection in the city of Ames. A vertex is encoded as a line

```
id,lat,lon
```

where `id` is an integer id of the vertex and `lat` and `lon` are the latitude and longitude that define the location of this vertex in Ames. The `id` of a vertex is especially important. This id is used to specify edges later in the file as well as identify key locations within Ames.

After all the vertices, there is a line

```
EDGES: m
```

where `m` is the number of edges in the graph.

The rest of the file consists of `m` lines each specifying an *undirected* edge within the graph with the format

```
source,target,distance(,street)
```

where `source` and `target` are two vertex ids that specify the edge, `distance` is the distance (in meters) between the two vertices, and `street` specifies the name of the street. Note that the parentheses around `street` mean that this field is *optional*. There are some edges that do not have street names.

You may notice that the latitude, longitude, and street names of vertices and edges are never required to be used in this project. These fields are included just for fun.

Part 2: (100pts)

We now give you a more complete description of Jim's coffee situation. To make Jim's specialty drink, there are six ingredients (labeled A,B,C,D,E,F) that must be picked up and delivered to Jim. The location (i.e. vertex id) of each of the six ingredients is listed below.

A : 981	**B** : 1653	**C** : 524
D : 1864	**E** : 1119	**F** : 1104

To make things worse, due to the extra special recipe Jim uses, these ingredients must be picked up in a particular order. Below is a list of constraints on the order the ingredients can be picked up.

- **A** must be picked up before **C** and **F**.
- **B** must be picked up before **C** and **D**.
- **C** must be picked up before **D** and **E**.
- **F** must be picked up before **C** and **E**.

You and Jim are at the following locations.

You : 1067 **Jim** : 826

Your task to solve Jim's coffee problem is described below.

1. Create a graph that models the ingredients' dependencies and use topological sort to find a way to pick up the ingredients that satisfies the constraints above.
2. Parse the city of Ames data file and store it in an instance of your graph data structure.
3. Given an ordering of the ingredients, find the shortest route you could take from your location to pick up every ingredient in the order given and deliver them to Jim at his location. An edge's cost (or weight) is the distance in meters of the edge.

   Note that `ames.txt` uses undirected edges. Since Dijkstra's uses directed edges, be sure to construct a graph with every undirected edge added twice (one forward and one backward).
4. Create a minimum spanning tree of the whole city of Ames. Again, an edge's cost is the distance in meters of the edge.

   Minimum spanning trees use undirected edges. Therefore you will not need to double add edges for this part.

You are provided with an interface `CoffeeTask` with three methods that correspond to the tasks above. You must write a class that implements this interface named `CSCoffeeTask`. Your implementation must also have a default constructor.

Your grade for this part will be based entirely on your implementation of this interface. Please read the documentation of the interface clearly before you begin the project.