**Dijkstra’s Algorithm**

Your task is to implement Dijkstra’s Algorithm to solve the single source shortest paths problem. Use good object oriented design to write a general solution.

The test data will be road data from the United States. The Data Sets can be [found here](http://www.dis.uniroma1.it/challenge9/download.shtml). Information on the file formats is on [this page](http://www.dis.uniroma1.it/challenge9/format.shtml#coord). You are interest in two types of files, the .gr and .co file. The .gr files contain the list of edges and their weights. The .co files contain a list of nodes and the gps coordinates that match those nodes. There are two types of .gr files. One gives the weight of the edges as distance between the nodes and the other gives the weight as transit time between the nodes. Both of the weights are given as integers so it should not matter to your program which type of weight is used as input.

You are to write a class that performs Dikstra’s algorithm on a graph. You’ll want to write another class that reads in .gr files and returns a graph that Dikstra can run on.

I want you to provide a main method in a class titled FindShortestRoadPath. The main method will take in 4 command line arguments: the input data file, the source node id, the target node id, and the outputfile.

**Usage:**

java FindShortestRoadPath inputFile.gr sourceID targetID outputFileName

Given a dataset, a source node, and a target node, I want to know the path from the source to the target and total cost of that path. Your program will write out to the output file the total cost of the path and the path from source to target. The first line of your output file will be the cost of the path. After that will be the path, one node of the path per line, source node first and target node last.

**Upload your code to gitlab. You do not need to bring a print copy to class.**

**Bonus (50 points):**

Draw your path on a map. You can use the .co file to get the GPS coordinates of each node on your path. How you draw onto a map is up to you. [Google Maps API](https://developers.google.com/maps/documentation/javascript/tutorial) seems like a fruitful place to start. [How to draw lines on google maps](https://developers.google.com/maps/documentation/javascript/shapes#polylines)!