Exercise – *Shortest Paths (Bonus Task)*

This task is a bit different from the others. There is no intriguing story here and no hidden algorithm to discover. It is plane and simple: We give you the street graph of a city or of a country and ask you to compute the lengths of some shortest paths.

Coding a working solution for this task should really not take you long as you can mostly reuse what is in the code snippet provided for the first BGL lecture. What is nice about this task though is that you can see a graph algorithm running on *"real world data"*. We provide you with the street graphs of the country of Liechtenstein, the city of Bern and the greater area of New York City. We also prepared a little visualization so that you can see how the shortest path that you computed looks like and you can play with your algorithm interactively.

Input

- The input starts with a line that contains three integers `n m q`, separated by a space. They denote
  - n, the number of vertices in the graph ($0 \leq n \leq 500'000$);
  - m, the number of directed street lanes in the graph ($0 \leq m \leq 1'000'000$);
  - q, the number of shortest path queries that you should answer ($0 \leq q \leq 1000$);
- The following n lines describe the positions of the vertices. Each line contains two integers `lon lat`, separated by a space and such that $-180'000'000 \leq lon \leq 180'000'000$ and $-90'000'000 \leq lat \leq 90'000'000$. Such a line describes the position of the vertex on our planet in longitude (East/West direction) and latitude (North/Sourth direction).
- The following m lines define the street lanes in the graph. Each line contains three integers `a b c`, separated by a space and such that $0 \leq a \leq n-1$, $0 \leq b \leq n-1$, $1 \leq c \leq 10^6$. Such a line describes a lane of a street of length c (in some unknown unit) that leads from vertex a to b (and only in this direction).
- The final q lines describe the queries. Each line contains two integers `s t`, separated by a space and such that $0 \leq s, t \leq n-1$. Such a line describes a query for a shortest path from vertex s to vertex b (and only in this direction).

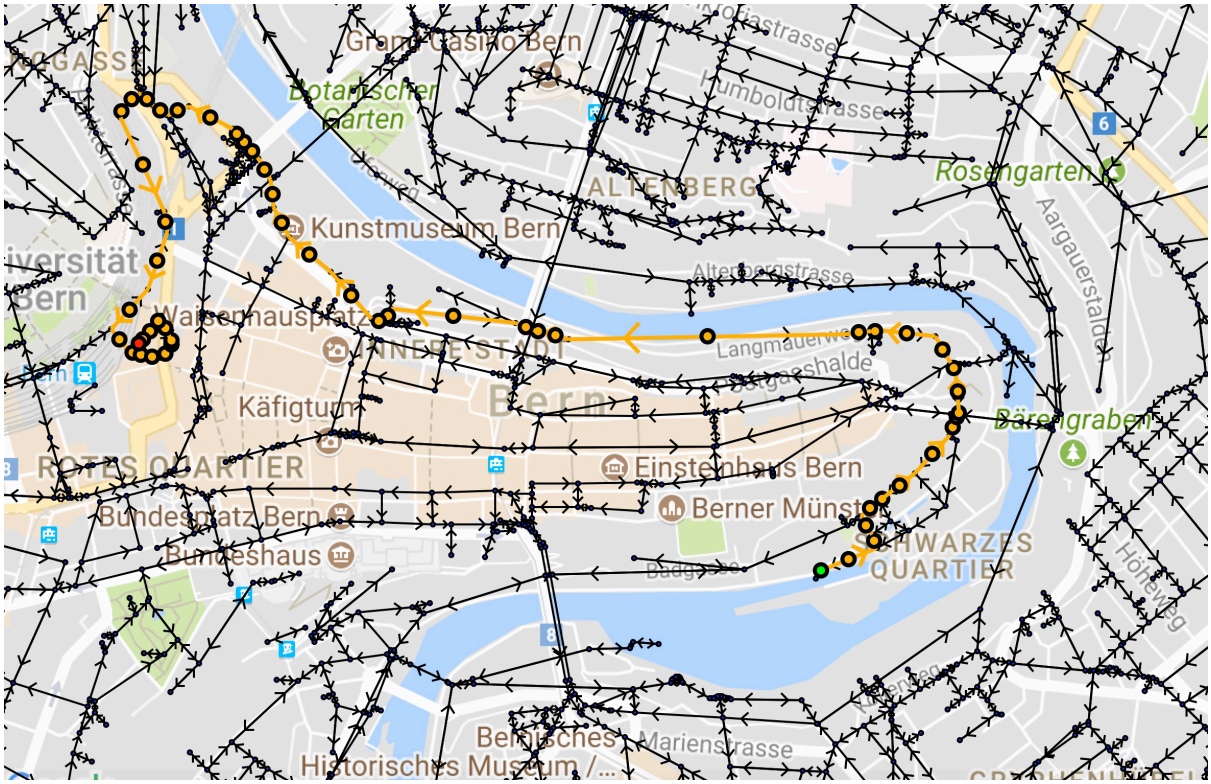
Output For each query output one line with a single integer that denotes the length of the shortest path from s to t, or the single word `unreachable` if there is no path from s to t.

Points There are three test sets, worth 30, 30, and 40 points. They contain the exact same graphs of Liechtenstein, Berne and New York City as we made available for you, just with different query pairs.

Visualization The visualization is provided in the file `visualization.py`. To run it you should note the following

- It requires python3 on your machine. Start it with "`python3 visualization.py`"
- Extend your solution so that it also prints the edges of the shortest path of each query in the following format on a second line: First the number of edges in the shortest paths, then the indices (according to the input order) of these edges, all separated by spaces.
- Set the path to your compiled solution and to the input file that you want to use (the remaining `TODO` comments).
- (Optional) To show Google Maps images behind the graph, uncomment the corresponding lines in the code (marked with `TODO` comments). It requires a few simple things to work
  - PIL (the python image library) or its successor Pillow. If it is not part of your python distribution, you can load it with `pip install Pillow` (preferably in a `venv`)
  - An API token for the Google Maps Static Image API which you can get for free here.

If it all works out you should end up with some nice images like this one:



Data Sources The graphs of Liechtenstein and Bern were created using dumps of Open Street Map (Geofabrik and Mapzen) and then extracted using RoutingKit. The graph of New York City was created from the benchmark test set of the 9th DIMACS Implementation Challenge.