Assignment 5: Trains - An Adventure-Drama

1 Context

Suffering from writers block, the TAs have decided to embark on a journey of self discovery, to once again be able to write inspired introductions to the assignments they provide. With so many assignments to write, this period of deep contemplation can only last so long, as they have enlisted you, dear student, to help them plan this (rapid) spiritual wandering.

Also there are bombs disrupting the train networks ... you're going to need to take into account.

You will need to read in a finite number of train networks. Each train network will have a finite number of disruptions which you'll need to take into account. Each train network will also have a finite number of queries which you will have to process, a query being a pair of two cities. Each query will require you to return a path between the two cities, followed by the total travel time.

Figure 1: An example of the type of pensive journey one of your TAs might undertake

2 Simplifying Assumptions

- Changing trains takes no time
- The time to travel from A to B is the same as from B to A.
- The train networks are **completely independent**, in the sense that no two networks shares a station. However, two stations *may* have the same name across two networks. For example, two networks might have a station named "Central station", but obviously these two stations are not the same.

3 Input

- The number of stations.
- One line per station, containing a number and the name.
- The number of connections.
- One line per connection, listing two stations by their number and the distance in minutes.
- The number of disruptions, followed by two lines for each disruption.
- Any number of queries, each consisting of two lines.
- An exclamation mark after the last query for the current train network.

See Section 7 for an example.

4 Output

For each query, you should output:

- The path between the starting city to the ending city (including the starting / ending cities themselves)
- The total distance traveled
- If there is no possible path, your program should print `UNREACHABLE`.

5 Notes

- No code is provided for this assignment and no files are automatically included by Themis. You may use any data structures and typedefs you want.
- Efficiency is very important for this assignment. Using the wrong algorithm / data structure can result in not passing test cases with large networks.
- You must use Dijkstra's algorithm to find the shortest path (not including the bonus, which must use A*).
- For the main part of this assignment you can earn up to 3 points by passing the Themis tests and up to 2 points for simplicity, efficiency and clarity.
- Important: Dijkstra's Algorithm is famous and many implementations can be found online. To learn the most from this assignment, please write your own solution and do not search for existing code. We remind you that submitting any work that is not your own without references is plagiarism and all such cases will be forwarded to the Board of Examiners.

6 Report

For this assignment you should also write a programming report. You can find a template for this on Nestor. Please follow all guidelines from Appendix E of the lecture notes and **submit your report as a single PDF file on Themis**.

7 Input-Output Example

Explanation	Input	Output
# of networks	3	UNREACHABLE
# of stations	2	Berlin
	0 A	Paris
	1 B	Marseille
# of connections	1	683
	0 1 1	Amsterdam
# of disruptions	1	Paris
take out the only connection	A	Marseille
	B	420
query 1	A	Mora
	B	Ostersund
end of queries for network 1	!	Gallivare
# of stations	4	1112
	0 Amsterdam	Trondheim
	1 Berlin	Ostersund
	2 Paris	Mora
	3 Marseille	1918
# of connections	5	
	0 1 382	
	0 2 238	
	1 2 501	
	1 3 709	
	2 3 182	
# of disruptions	1	
disruption start	Berlin	
disruption end	Marseille	
query 1	Berlin	
	Marseille	
query 2	Amsterdam	
	Marseille	
end of queries for network 2	!	
# of stations	4	
	0 Gallivare	
	1 Ostersund	
	2 Mora	
	3 Trondheim	
# of connections	3	
	0 1 803	
	1 2 309	
	1 3 1609	
# of disruptions	0	
query 1	Mora	
	Gallivare	
query 2	Trondheim	
	Mora	
end of queries for network 3	!	

8 Extra: A* Implementation

Oh no! One of the TAs self-discovered that they were the twist villain all along! They have now Aku'd you and the other TAs back to the distant desolate future[1], forever leaving the first years with no good assignment introductions. As you all land on cold stone below you, you see the portal dissipating above you. All seems lost ...

Your faith holds steadfast however, and you travel to a nearby village, where you discover a map and rumors of other time portals. *Filled with determination*, you and the other TAs decide to split, and reach these portals, in the hope that at least one one of you can return to the past, and undo the future that is [REDACTED]!

Naturally these paths may be blocked off by the henchmen of [REDACTED] and the TAs may also have to recalculate the best path from different locations. Furthermore, the computers of the future are simply too slow to run Dijkstra. You need to implement a faster algorithm, that potentially explores less nodes: A*.

The heuristic you will use is the shortest distance, as the crow flies.

To simplify the situation a bit, you can assume a completely flat topography and that you only need to scan in 1 network (meaning that the initial digit indicating the number of networks is not needed).

To this end, you must slightly adjust the input to also take in geo-coordinates of the nodes. We write these geo-coordinates in a new line, below the station id/name. The output format remains the same.

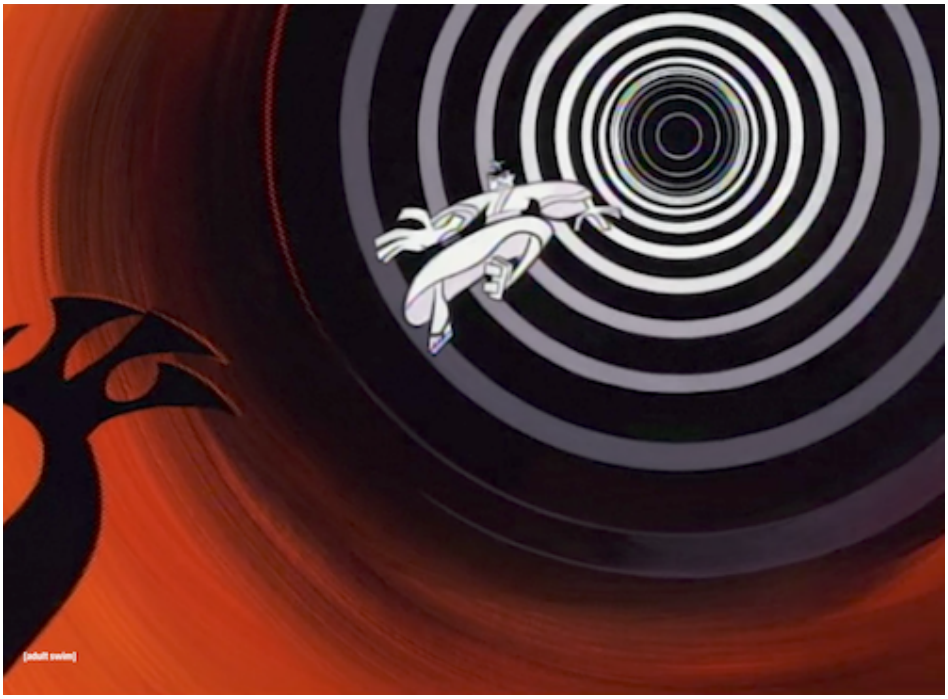

Figure 2: One of your TAs being thrown back in time, 2023 (colorized)

[1] A future in which the Netherlands and its provinces still exist

8.1 Input-Output Example

Explanation	Input	Output
# of stations	6	UNREACHABLE
station 1	0 Maastricht	
station 1 geo coords	50.85018/5.70525	
station 2	1 Eindhoven	
station 2 geo coords	51.44296/5.47953	
...	2 Nijmegen	
	51.84313/5.85309	
	3 Den Haag	
	52.08084/4.32455	
	4 Utrecht	
	52.08939/5.10982	
	5 Enschede	
	52.22236/6.88977	
# of connections	5	
connection 1	3 1 89	
connection 2	1 0 63	
...	1 2 55	
	1 4 47	
	0 2 111	
# of disruptions	1	
disruption start	Den Haag	
disruption end	Eindhoven	
query 1 start	Enschede	
query 1 end	Eindhoven	
good-bye	!	