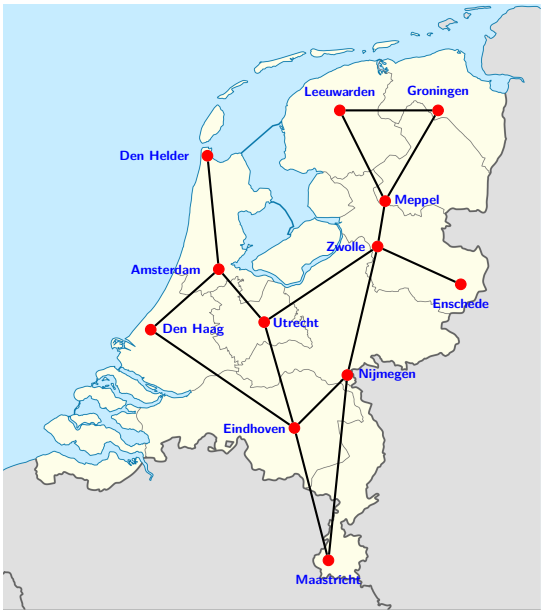
Assignment 5: Trains in the Storm

It is Saturday afternoon and you have to be at home before the curfew. The strongest snow storm in decades just passed the country. Due to the storm the Dutch railways can only offer a limited number of trains. Will you be able to make it back home in time?

Below you can see a map of the train lines that are still available.



Map based on https://commons.wikimedia.org/wiki/File:Netherlands_location_map.svg — CC BY-SA Lencer.

Travel time in minutes:

Amsterdam	Den Haag	46
Amsterdam	Den Helder	77
Amsterdam	Utrecht	26
Den Haag	Eindhoven	89
Eindhoven	Maastricht	63
Eindhoven	Nijmegen	55
Eindhoven	Utrecht	47
Enschede	Zwolle	50
Groningen	Leeuwarden	34
Groningen	Meppel	49
Leeuwarden	Meppel	40
Maastricht	Nijmegen	111
Meppel	Zwolle	15
Nijmegen	Zwolle	77
Utrecht	Zwolle	51

In addition to the above, you can get information about additional disruptions, as shown at `https://www.ns.nl/en/travel-information/current-situation-on-the-tracks`. This information is included at the beginning of each test case as shown below.

We make some simplifying assumptions:

- We only care about the duration, not the time of the day.
- Changing trains never takes any time, all connections are instant.
- The time required to travel from A to B is the same as from B to A.

Input

- The input starts with a line containing the number n of current disruptions.
- The first line is followed by n many disruptions. Each disruption consists of two lines which describe a direct connection that is no longer available.
- All remaining lines are *queries*. Each query consists of two lines: the start and the goal.
- The input ends with an exclamation mark.

Output

For each query your program should use Dijkstra's Algorithm to find the fastest connection and then output the list of all stations along that route, including the starting and ending station. Moreover, your program should print the total time (in minutes) this connection will take. In case there is no connection for a given query, your program should print 'UNREACHABLE'.

Input-Output Examples

input	resulting **output**
0 Leeuwarden Meppel Groningen Amsterdam Eindhoven Zwolle !	Leeuwarden Meppel 40 Groningen Meppel Zwolle Utrecht Amsterdam 141 Eindhoven Utrecht Zwolle 98

input	resulting **output**
2 Utrecht Zwolle Enschede Zwolle Amsterdam Groningen Enschede Eindhoven Leeuwarden Eindhoven !	Amsterdam Utrecht Eindhoven Nijmegen Zwolle Meppel Groningen 269 UNREACHABLE Leeuwarden Meppel Zwolle Nijmegen Eindhoven 187

Notes

- No code is provided for this assignment and no files are automatically included by Themis. You may use any data structures and `typedef`s you want.
- **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.
- 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.

Report (5 points)

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.

Extra 1: Optimization with A* (up to 1 bonus point)

The NS appreciates your program, but it is not efficient enough. Can you optimize it further by using the A* algorithm instead of Dijkstra's Algorithm? For the heuristic you will need geo-coordinates of each station. See for example `https://osm.org/node/1112410297`.

The input/output format for this part is the same as for the main part of the assignment.

Extra 2: Going International (up to 1 bonus point)

Imagine the COVID-19 crisis is over and it is time to go on holiday again. Adapt your program to read in an arbitrary train network. The first line of the input is the number of different train networks. For each train network, the input then consists of:

- 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.

explanation	**input**	resulting **output**
`# of networks`	`2`	`Amsterdam`
`# of stations`	`4`	`Paris`
	`0 Amsterdam`	`Marseille`
	`1 Berlin`	`420`
	`2 Paris`	`Mora`
	`3 Marseille`	`Ostersund`
`# of connections`	`5`	`Gallivare`
	`0 1 382`	`1112`
	`0 2 238`	`Trondheim`
	`1 2 501`	`Ostersund`
	`1 3 709`	`Mora`
	`2 3 182`	`1917`
`# of disruptions`	`1`	
`disruption`	`Berlin`	
	`Marseille`	
`query`	`Amsterdam`	
	`Marseille`	
`end of queries`	`!`	
`# of stations`	`4`	
	`0 Gallivare`	
	`1 Ostersund`	
	`2 Mora`	
	`3 Trondheim`	
`# of connections`	`3`	
	`0 1 803`	
	`1 2 309`	
	`1 3 1608`	
`# of disruptions`	`0`	
`first query`	`Mora`	
	`Gallivare`	
`second query`	`Trondheim`	
	`Mora`	
`end of queries`	`!`	