

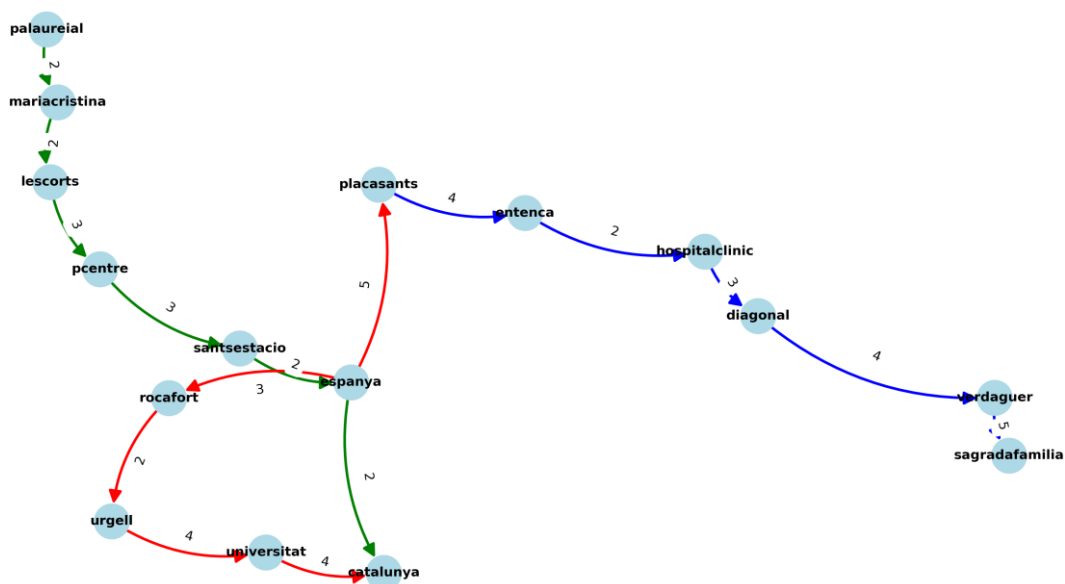
Homework 5: Logic Programming

Introduction	1
Testing.....	2
Connected.....	2
Number of steps	4
Route Distance.....	6
Same Line.....	8

Introduction

This report explores the application of logic programming to model and analyze the Barcelona underground map. Four predicates—`connected/2`, `numberofsteps/3`, `routedistance/4`, and `sameline/3`—are developed to facilitate path finding, step counting, route distance calculation, and line identification within the network. Each predicate is tested with various scenarios to ensure correctness and effectiveness using the Prolog interpreter available on the "Learn Prolog Now!" website. For each of the predicates implemented, some testing examples are provided.

The piece of underground map selected is the following one. We tried to include different lines, represented with the edge colors; different weights, represented with edges labels; and a multiple path example for testing purposes (espanya – catalunya).



Testing

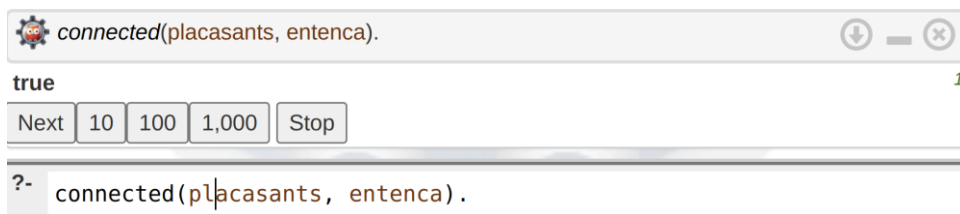
Connected

Recursive function which checks if there exists a path from 2 stations.

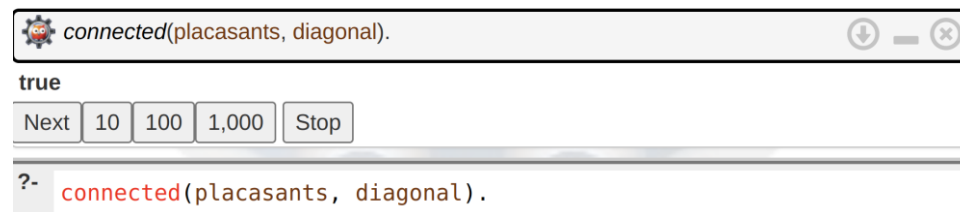
```
% Base Case: Every station is self-connected
connected(B, B).

connected(A, B) :-
    link(A, X, _, _),      % There is a link from A to X
    connected(X, B).        % X is connected to B
```

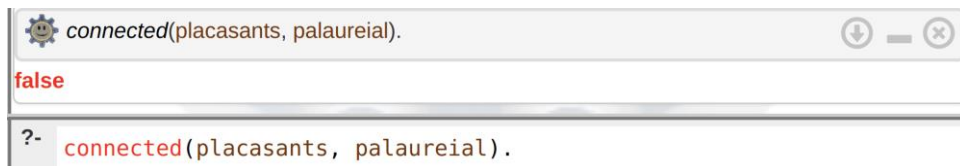
Successful Query for Direct Connection. Expected: true.






Successful Query for Indirect Connection. Expected: true.



Unsuccessful Query for No Connection. Expected: false.






Query for Self-Connection. Expected: true.

 `connected(placasants, placasants).`  

true 1

?- `connected(placasants, placasants).`

Query for Indirect Connection with No Direct Link (opposite direction). Expected: false.

 `connected(espanya, palaureial).`  

false

?- `connected(espanya, palaureial).`

Number of steps

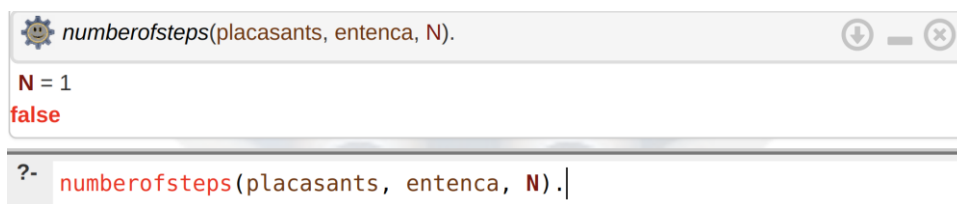
Number of steps needed to go from A to B following a certain path.

```
% Base case: If they are directly connected, 1 step
numberofsteps(A, B, 1) :-
    link(A, B, _, _).

numberofsteps(A, B, N):-
    link(A, X, _, _),
    numberofsteps(X, B, P),
    N is P+1.
```

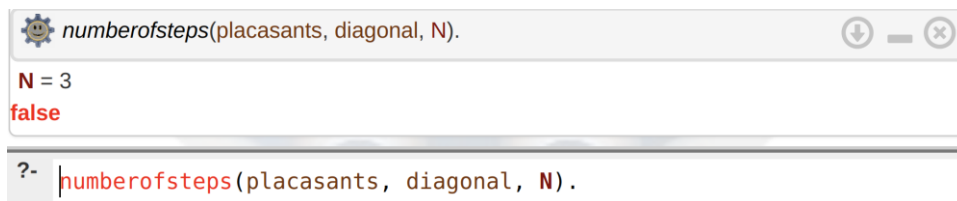
Successful Query with 1 Step. Expected: N = 1.

This tests if two stations are directly connected.



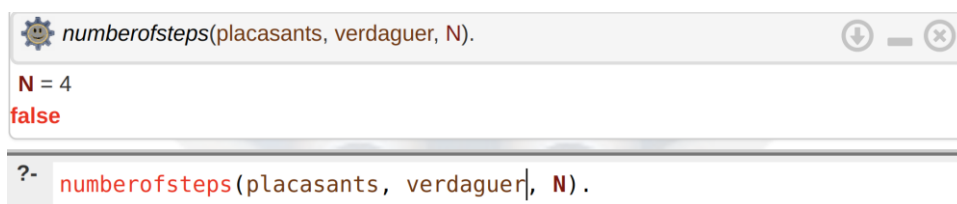
```
numberofsteps(placasants, entenca, N).
N = 1
false
?- numberofsteps(placasants, entenca, N).
```

Successful Query with 2 Steps. Expected N = 3.



```
numberofsteps(placasants, diagonal, N).
N = 3
false
?- numberofsteps(placasants, diagonal, N).
```

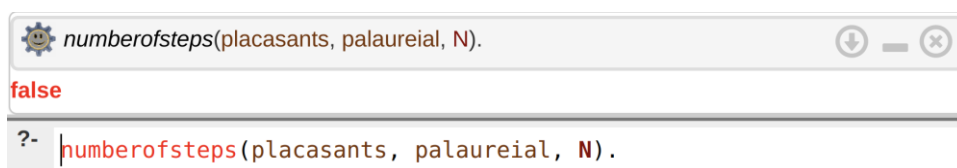
Successful Query with 3 Steps. Expected N = 4.



```
numberofsteps(placasants, verdaguer, N).
N = 4
false
?- numberofsteps(placasants, verdaguer, N).
```

Unsuccessful Query. Expected: false.

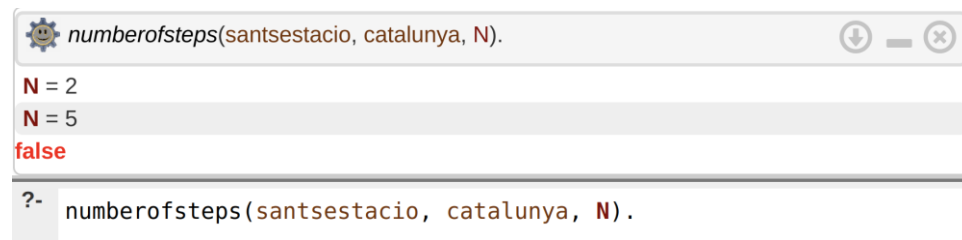
This tests if two stations are not connected.



```
numberofsteps(placasants, palaureial, N).
false
?- numberofsteps(placasants, palaureial, N).
```

Query with More than One Possible Answer. Expected: N=2; N=5.

This tests if there are multiple ways to reach a destination station.



```
numberofsteps(santsestacio, catalunya, N).  
N = 2  
N = 5  
false  
?- numberofsteps(santsestacio, catalunya, N).
```

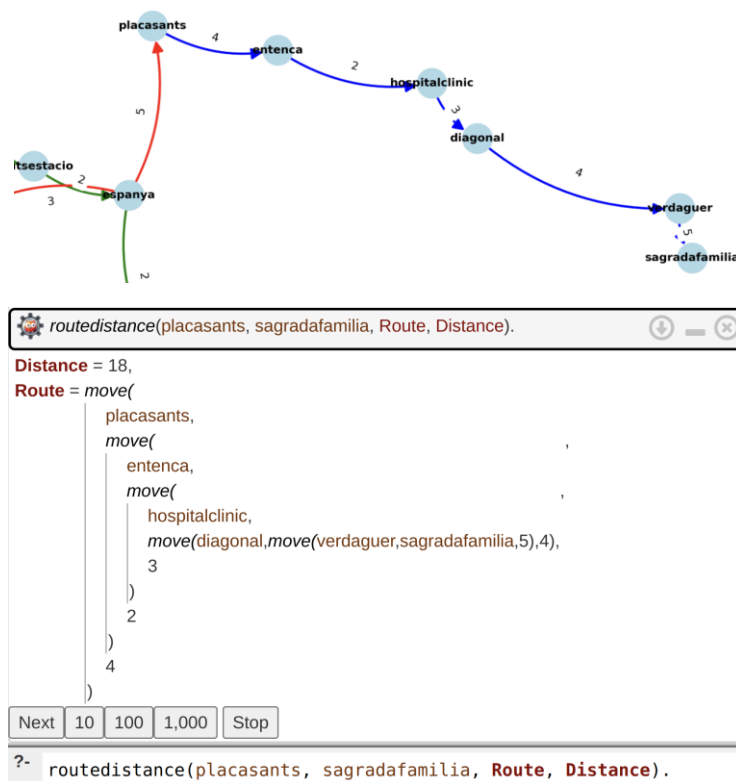
Route Distance

Given two stations A and B, returns in D the time needed to move from A to B following a path in the graph, and R returns a representation of each step to be made on that path and its time length.

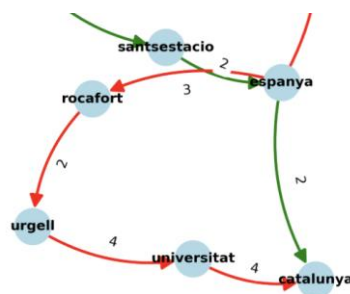
```
% Base Case: NO intermediate connection
routedistance(A, B, move(A, B, D), D) :-
    link(A, B, D, _).

routedistance(A, B, move(A, C, D1), D) :-
    link(A, Z, D1, _),
    routedistance(Z, B, C, D2),
    D is D1 + D2.
```

Query for Route Distance with Intermediate Stations and a Single Possibility.



Query for Route Distance with Intermediate Stations and Multiple Possibilities.



```
⚙️ routedistance(santsestacio, catalunya, Route, Distance).  
Distance = 4,  
Route = move(santsestacio,move(espanya,catalunya,2),2)  
Distance = 15,  
Route =  
move(santsestacio,  
move(espanya,move(rocafort,move(urgell,move(universitat,catalunya,4),4),2),3),2)  
false  
?- routedistance(santsestacio, catalunya, Route, Distance).
```

Unsuccessful Query: Opposite Direction.

```
⚙️ routedistance(catalunya, espanya, Route, Distance).  
false  
?- routedistance(catalunya, espanya, Route, Distance).
```


Same Line

Given two stations A and B, returns in C the colour of the line, if there exists a path from A to B in which only links of that line are used.

```
% Base Case: No intermediate connections
sameline(A, B, C) :-
    link(A, B, _, C).

sameline(A, B, C) :-
    link(A, X, _, C),
    sameline(X, B, C).
```


Query with Same Line Connection

 `sameline(placasants, entenca, Line).`

`Line = blue`
`false`

`?- sameline(placasants, entenca, Line).`

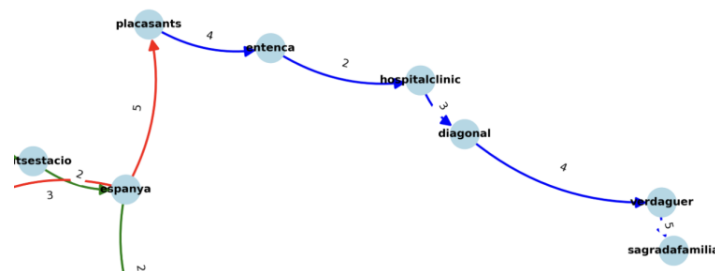
Query with Same Line Connection. Multiple Options.


 `sameline(espanya, catalunya, Line).`

`Line = green`
`Line = red`
`false`

`?- sameline(espanya, catalunya, Line).`

Query with Metro Commuting.



 `sameline(espanya, sagradafamilia, Line).`

`false`

`?- sameline(espanya, sagradafamilia, Line).`