

# Public Transport Connections

Aurélie Martin & David Girou

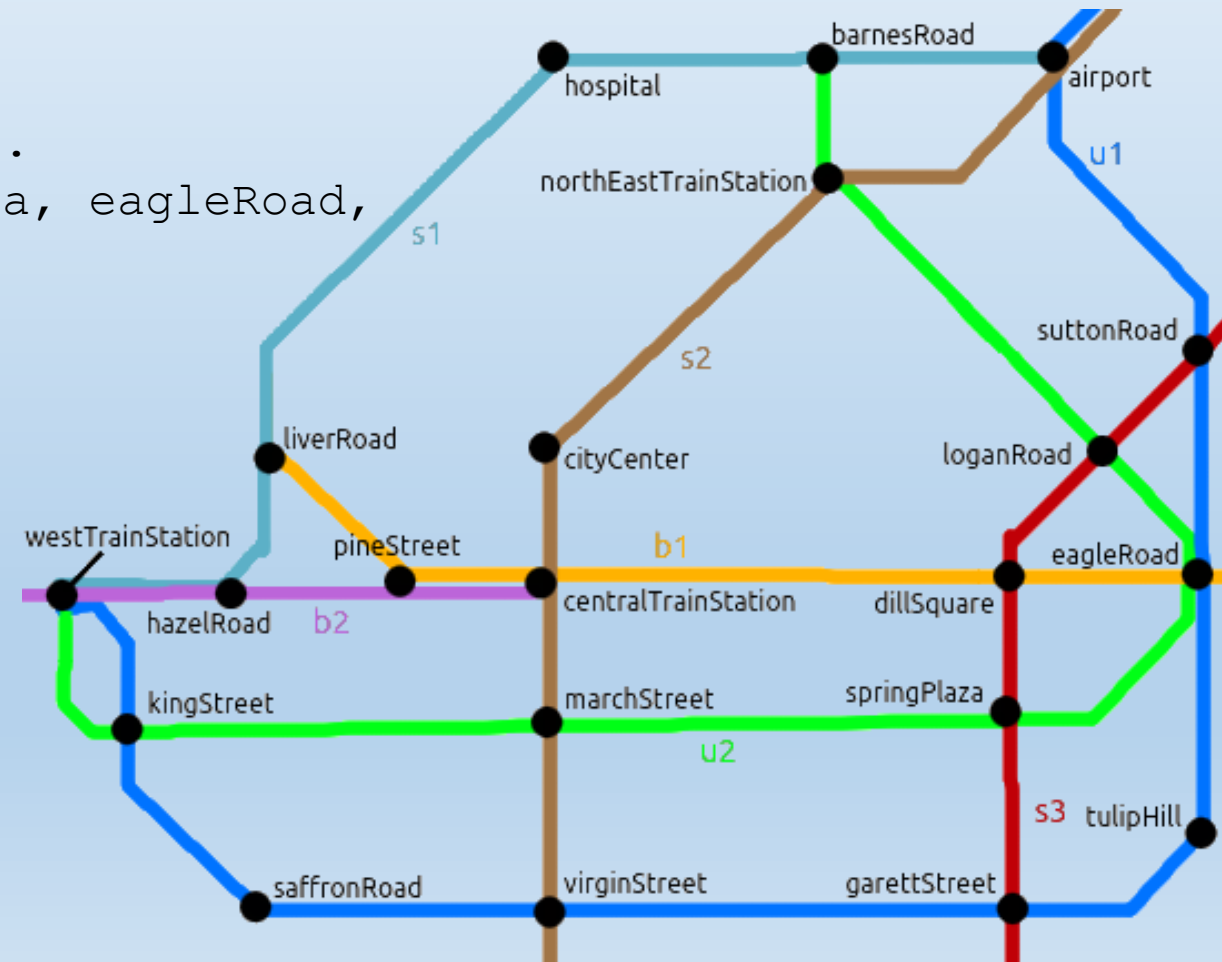
# Main Usage

## Public Transport Connections

Girou - Martin

```
?- route(pineStreet, airport, Route).  
Route = [pineStreet, liverRoad, hospital, barnesRoad,  
airport].
```

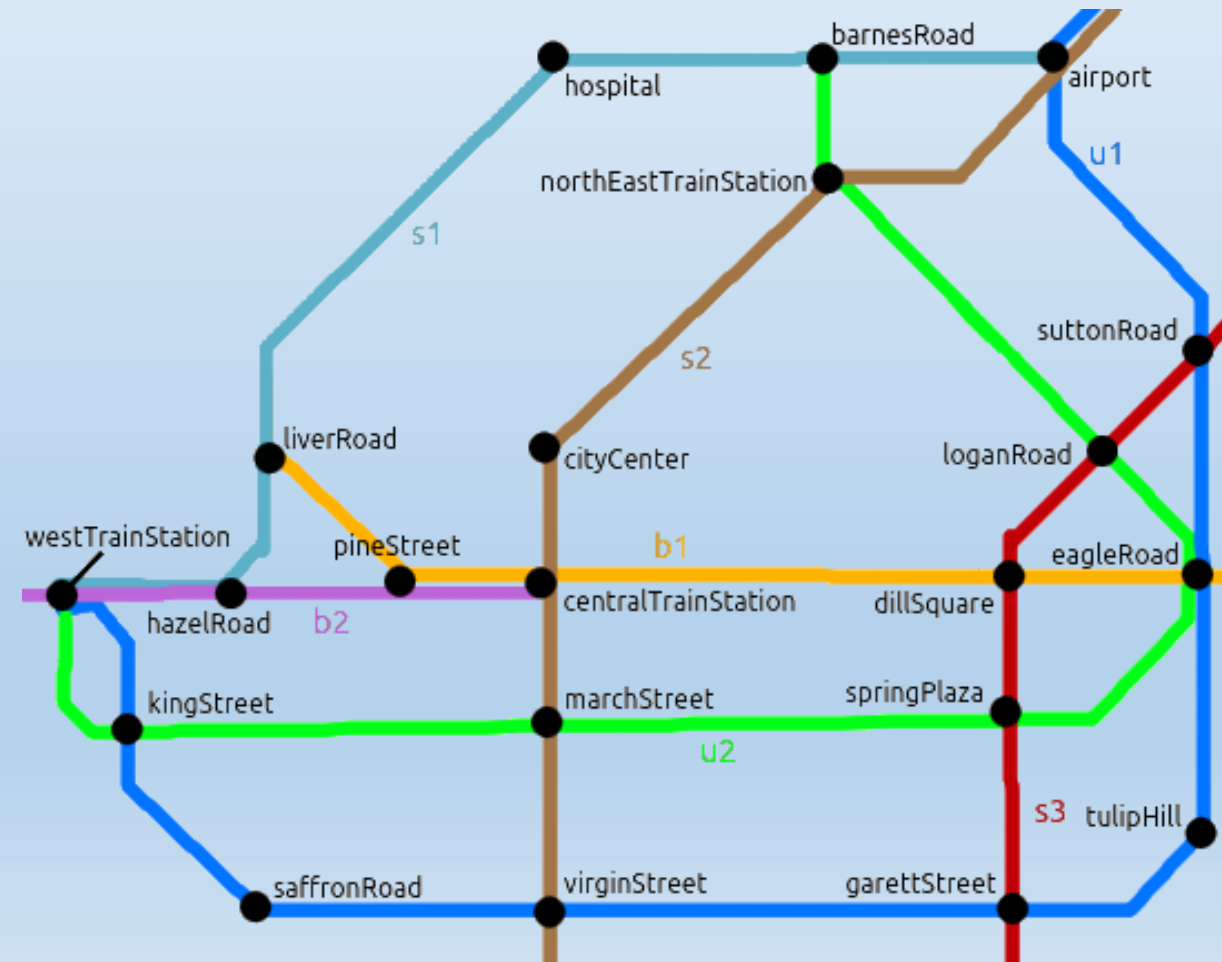
```
?- route(kingStreet, loganRoad, Route, Time).  
Route = [kingStreet, marchStreet, springPlaza, eagleRoad,  
loganRoad],  
Time = 24.
```



# The code

## Declaring the stops

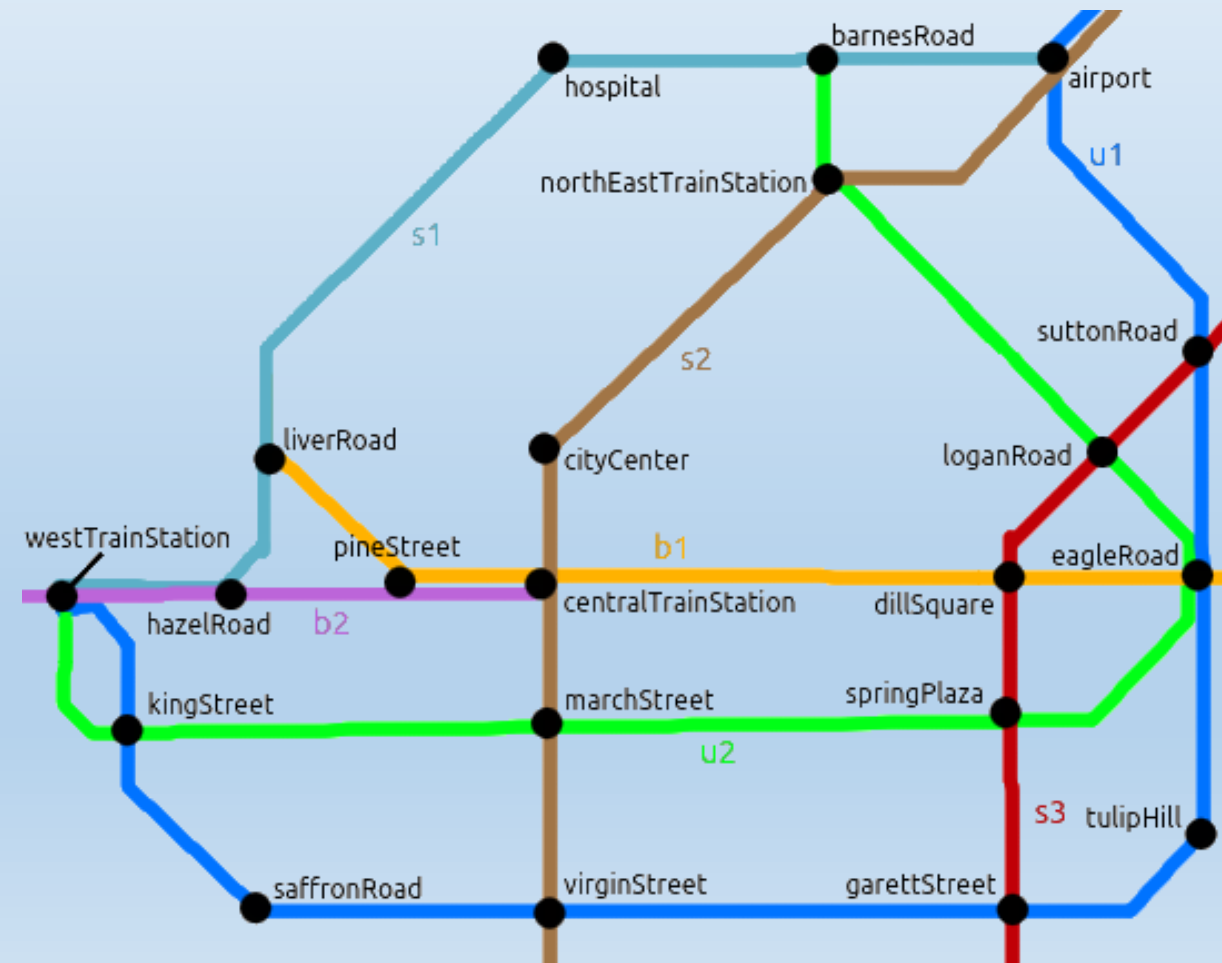
```
stop(airport, [u1,s1,s2]).
stop(suttonRoad, [u1,s3]).
stop(eagleRoad, [u1,u2,b1]).
stop(tulipHill, [u1]).
stop(garettStreet, [u1,s3]).
stop(virginStreet, [u1,s2]).
stop(saffronRoad, [u1]).
stop(kingStreet, [u1,u2]).
stop(westTrainStation, [u1,u2,s1,b2]).
stop(barnesRoad, [u2,s1]).
stop(northEastTrainStation, [u2,s2]).
stop(loganRoad, [u2,s3]).
stop(springPlaza, [u2,s3]).
stop(marchStreet, [u2,s2]).
stop(hospital, [s1]).
stop(liverRoad, [s1,b1]).
stop(hazelRoad, [s1,b2]).
stop(cityCenter, [s2]).
stop(centralTrainStation, [s2,b1,b2]).
stop(dillSquare, [s3,b1]).
stop(pineStreet, [b1,b2]).
```



# The code

## Declaring adjacent stops

```
adjacent(X,Y):- adjacent_stops(X,Y).  
adjacent(X,Y):- adjacent_stops(Y,X).  
  
%u1  
adjacent_stops(airport,suttonRoad).  
adjacent_stops(suttonRoad,eagleRoad).  
adjacent_stops(eagleRoad,tulipHill).  
adjacent_stops(tulipHill,garettStreet).  
adjacent_stops(garettStreet,virginStreet).  
adjacent_stops(virginStreet,saffronRoad).  
adjacent_stops(saffronRoad,kingStreet).  
adjacent_stops(kingStreet,westTrainStation).
```



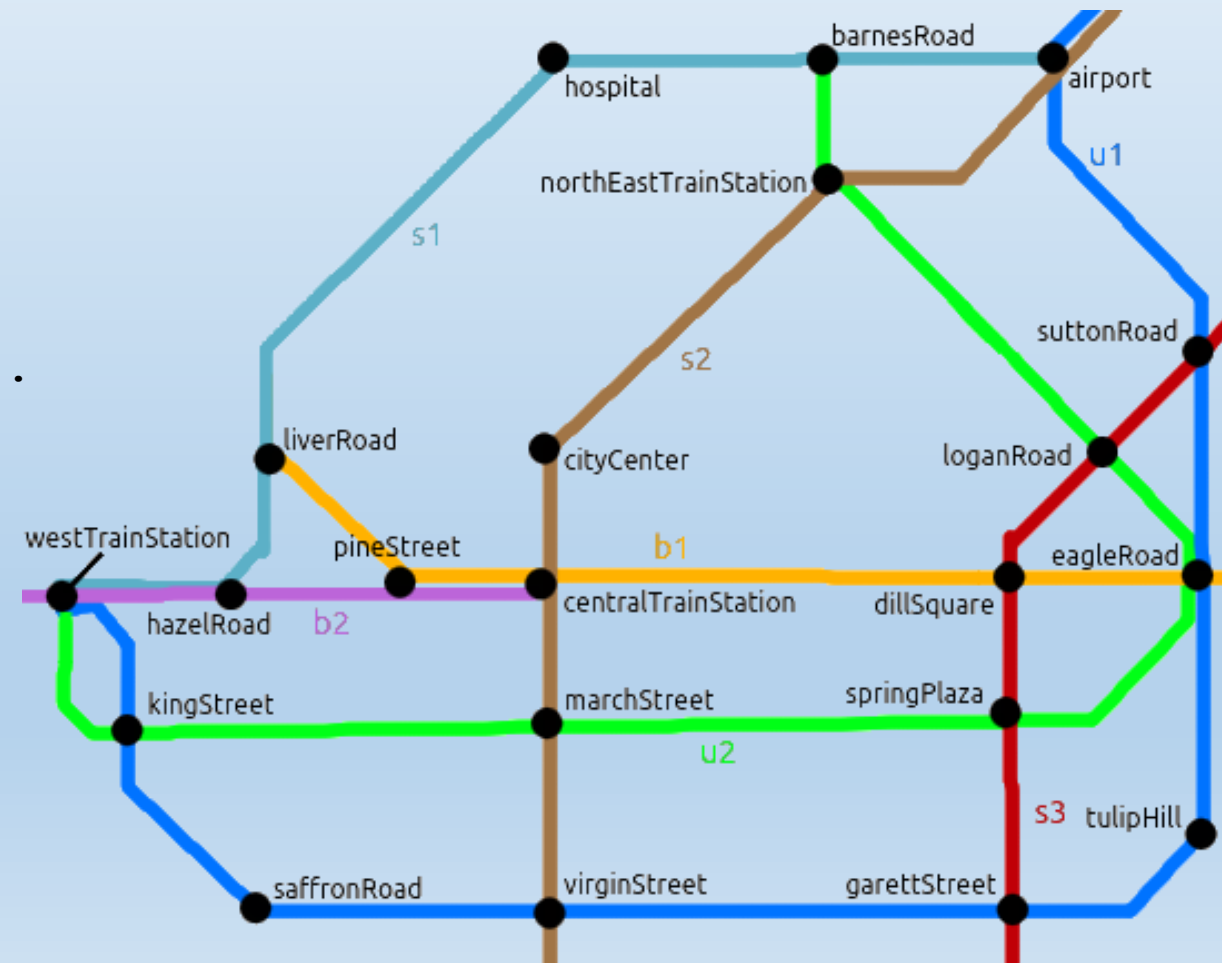
# The code

## The sameLine rule

```
sameLine(Stop1, Stop2, Line):-  
    stop(Stop1, Line1),  
    stop(Stop2, Line2),  
    member(Line, Line1),  
    member(Line, Line2).
```

```
?- sameLine(saffronRoad, suttonRoad, Line).  
Line = u1.
```

```
?- sameLine(airport, pineStreet, Line).  
false.
```

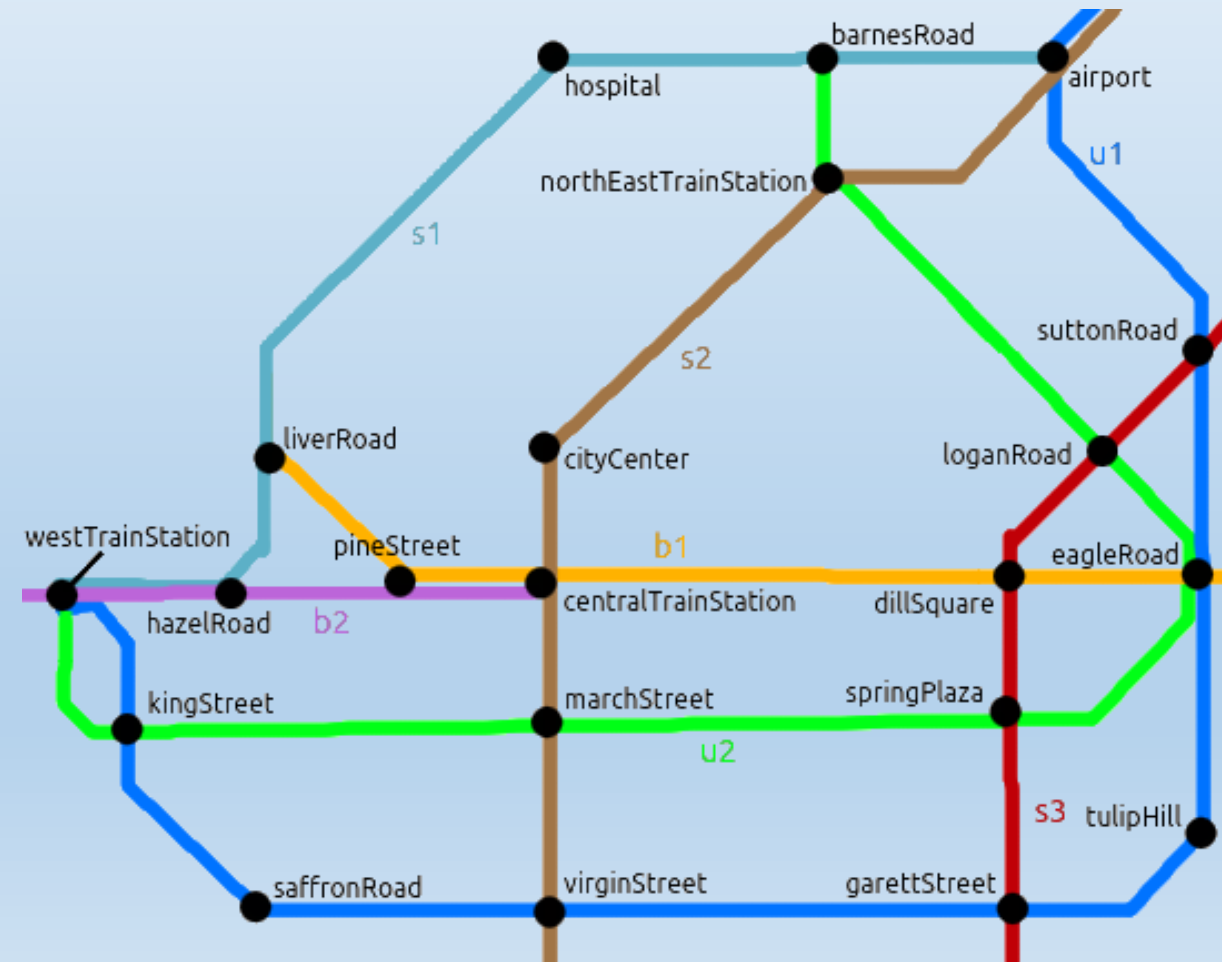


# The code

## The findAllStops rule

```
findAllStops(Line, ListOfStops):-  
    findall(Stop,(stop(Stop,NewLine),  
        member(Line, NewLine)), ListOfStops).
```

```
?- findAllStops(b2, Stops).  
Stops = [westTrainStation, hazelRoad,  
centralTrainStation, pineStreet].
```

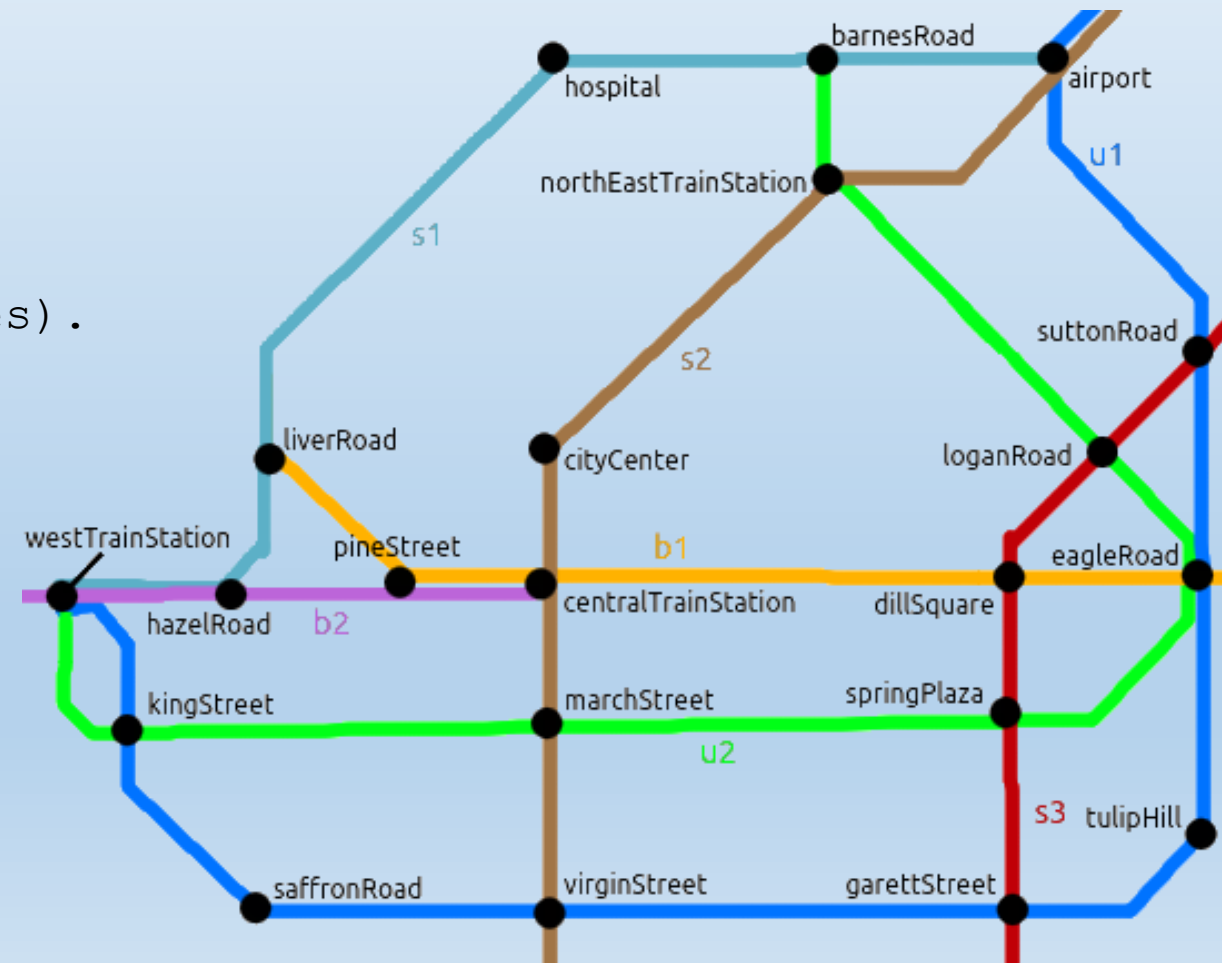


# The code

The numberOfLines rule

```
numberOfLines(Stop, NumberOfLines) :-  
    stop(Stop, Line),  
    length(Line, NumberOfLines).
```

```
?- numberOfLines(centralTrainStation, Lines).  
Lines = 3.
```



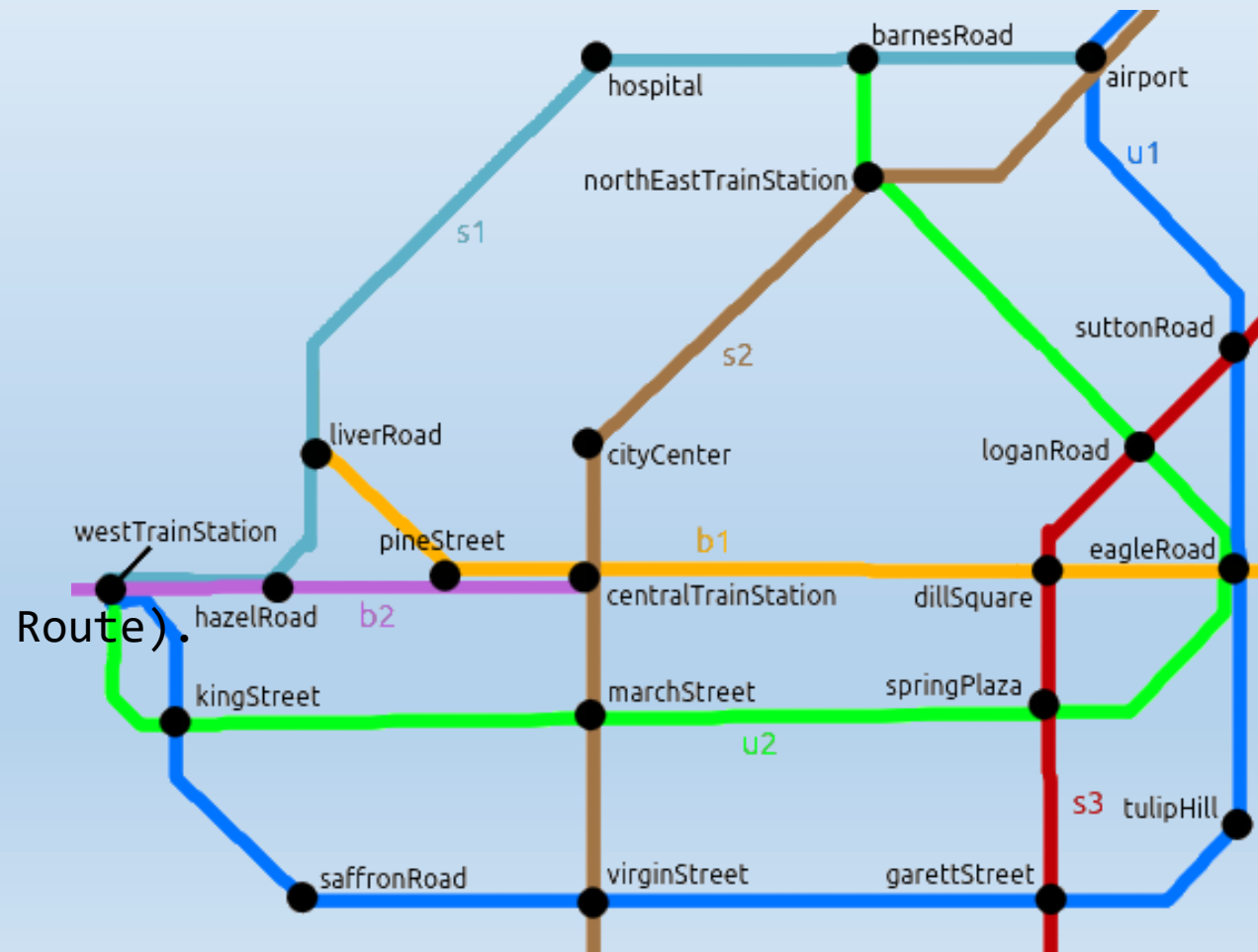
# The code

## The route calculation

```
calcRoute(Stop1, Stop2, Route):-  
    tempRoute(Stop1, Stop2, [], Return),  
    reverse([Stop2|Return],Route).
```

```
tempRoute(Stop1, Stop2, Temp, Route):-  
    adjacent(Stop1, Stop2),  
    \+member(Stop1, Temp),  
    Route = [Stop1|Temp].
```

```
tempRoute(Stop1, Stop2, Temp, Route):-  
    adjacent(Stop1,Next),  
    Next \== Stop2,  
    \+member(Stop1, Temp),  
    tempRoute(Next, Stop2, [Stop1|Temp], Route).
```



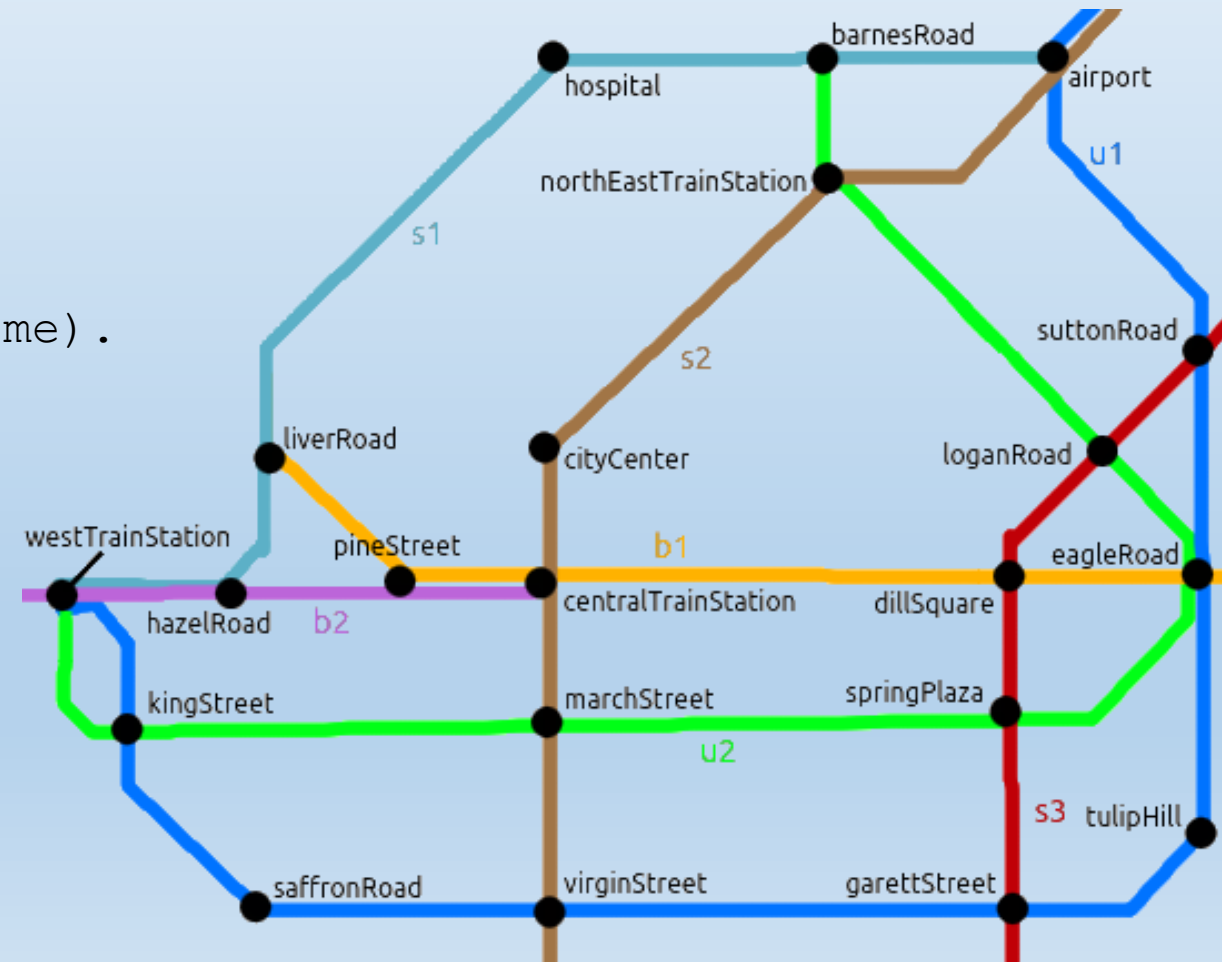


# The code

## The routeTime rule

```
routeTime(Stop1, Stop2, Route, RouteTime):-  
    calcRoute(Stop1, Stop2, Route),  
    length(Route, Time),  
    RouteTime is (Time -1) * 6.
```

```
?- routeTime(airport, eagleRoad, Route, Time).  
Route = [airport, suttonRoad, eagleRoad],  
Time = 12 .
```



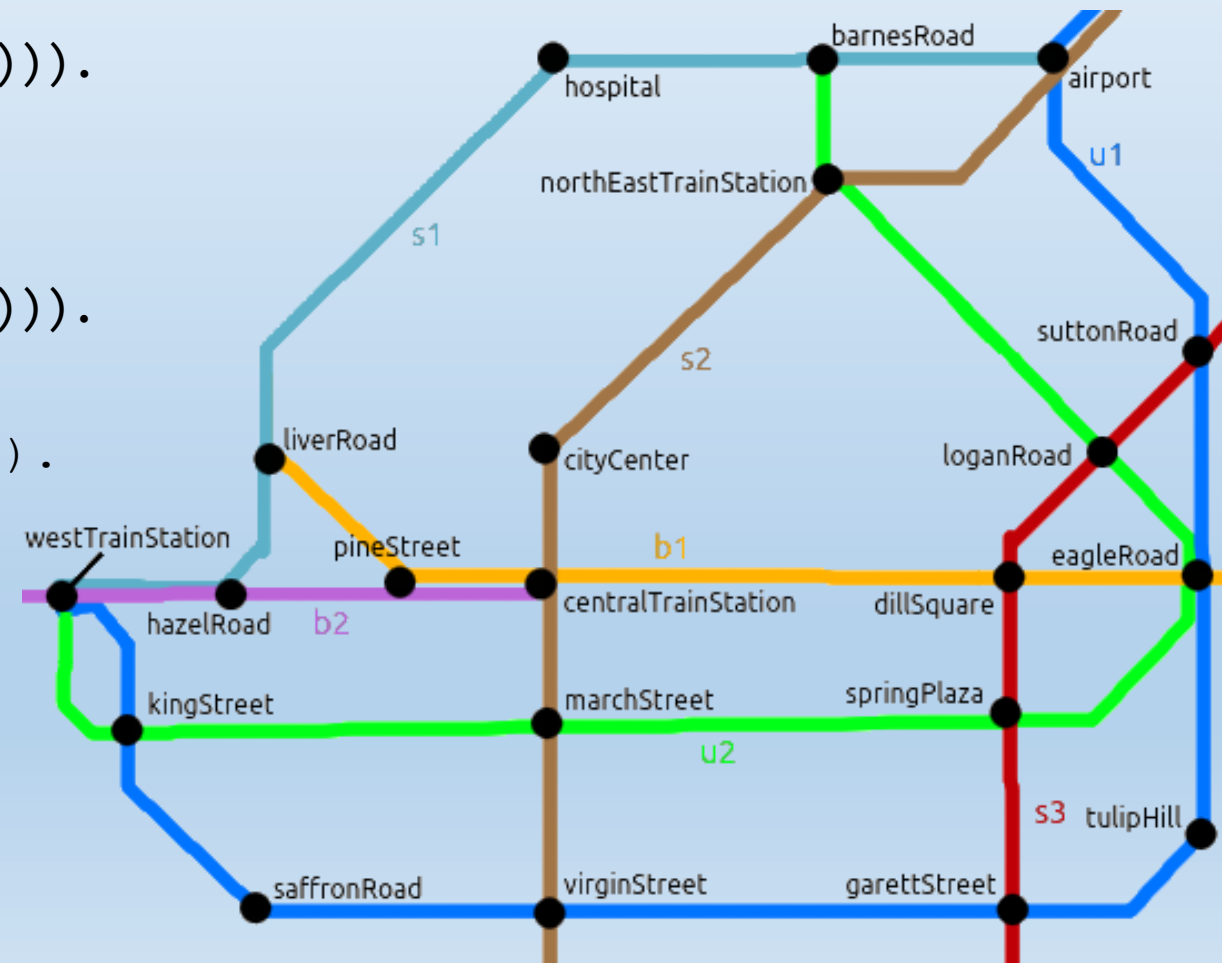
# The code

## The route rule

```
route(Stop1, Stop2, Route):-  
    limit(1, (order_by([asc(Time)],  
        (routeTime(Stop1, Stop2, Route, Time))))).
```

```
route(Stop1, Stop2, Route, Time):-  
    limit(1, (order_by([asc(Time)],  
        (routeTime(Stop1, Stop2, Route, Time))))).
```

```
?- route(hospital, pineStreet, Route, Time).  
Route = [hospital, liverRoad, pineStreet],  
Time = 12.
```



Thank you