Figure 1: Route suggested by google maps

0.1 Problem Area

There exist a plethora of route finding services for finding the shortest route between two points. These typically optimise for the shortest time taken to travel between two points. Provision for cyclists is however lacking, because it typically consists of an alteration to the existing code for cars, slightly modified to allow for cycle paths and different timings for cyclists.
However many of these route finding applications don't take into account safety. This is shown in Figure 1 which shows the route Google Maps suggests that I cycle to school by. The route includes the A4, which is a very dangerous stretch of road for cyclists as it is a 3 lane road. My plan is to combine accident statistics with traffic data to work out which roads are dangerous and then avoid them (finding the least dangerous instead of the shortest route).

0.2 Client

The client is me because I am personally not a fan of being involved in cycling accidents

0.3 Similar Systems

There exist similar systems such as google maps and Citymapper. These have advantage's over me such as being able to use live traffic data and other such things. Citymapper has a option to choose between "Quiet", "Regular", and "Fast" routes, which seems to be based on the type of road you are taken down. However they don't really take accident density into account which will be my aims.

0.4 Features

0.5 Map data

I need a data source which can provide data on roads that are legal to cycle on as well as being freely available for me to use. I settled on Open Street Map, a project which combines data gathered by volunteers into one massive freely available map. The map is downloadable in the form of a large XML file or PBF file. A PBF file is just a binary version of the same thing. I could write code to parse this myself, but instead I think that I am going to use the library libosmium which works in C++.

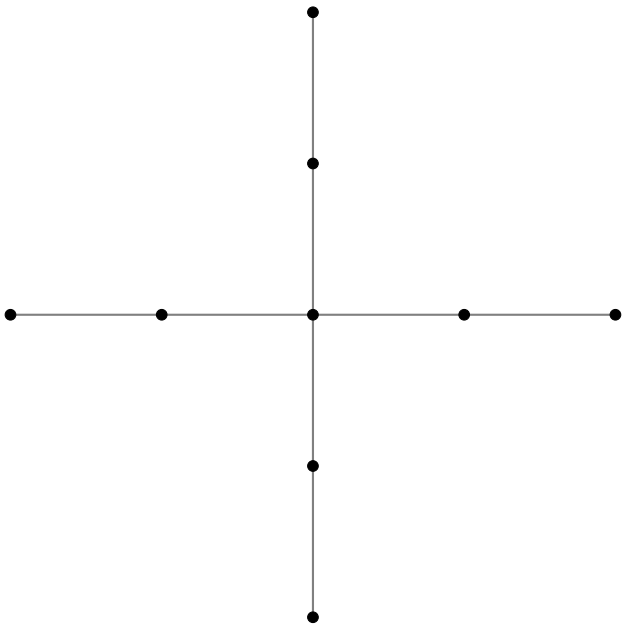

Figure 2: Typical OSM representation of an intersection

0.6 Accident Data

Transport for London has an excellent API which you can download accident data from. The data comes in JSON files and contains information about what type of vehicle was involved in each accident, the severity, and the coordinates of the accident. I need to find a way of mapping the coordinates onto the road network so that the danger of roads can be calculated as accurately as possible. Originally I was looking at adding the accident to the nearest edge in the graph but I decided against this after considering how intersections are typically represented in OSM. Road intersections typically look something like what is represented in Figure 2. If an accident were to occur at the intersection it would end up being added to one of the segments leading into the intersection, so the danger would not be properly calculated if not passing through that segment. Instead I thought about adding the danger to the nearest node, so that the accident would always be counted when a route passes through that intersection. The other problem that I have to deal with, if adding data to the nearest node is that the density of nodes is not constant. Straighter roads will not need to use as many OSM nodes as curved roads, so i might incorrectly add cost to the wrong node. My proposed solution to this is to interpolate in nodes to a very high density to deal with this problem.

0.7 Traffic Data

The main problem with this is that the accident data is absolute and can thus not be used to calculate probabilities. For example, more accidents happen on King's Street than the dangerous road I showed earlier, but this doesn't mean that King's Street is more dangerous merely that more cyclists travel on it. This means that I need to get accurate cyclist traffic data for the whole of London in order to turn my accident statistics into accident probabilities. The Department for Transport and the Office for National Statistics both keep data on traffic, but it isn't applicable because cycle data is only given as a total [1] and at specific count points. This means that I will

[1] 0.6 billion miles per year in London

either need to work out traffic data or get it from some dataset, such as Strava's Global Heatmap. Luckily the rest of the application can work without including traffic data so my plan is to deal with this problem at a later time.

0.8 Finding the shortest route

Dijkstra's algorithm and A* are both very slow. I will use preprocessing based algorithms such as ALT* to make my program run a lot faster.

0.9 Critical Path

- Import accidents
- Match accidents to nodes
- Build Graph
- Write pathfinder algorithms