# Exercise 5

In this exercise, we discuss two different possible uses of Spark GraphX and Spark Spark MLib.

## Analysis of reachability by public transport: identifying missing links

With the threat of climate change, it is essential that cities move away from a personal automobile dominated plan to one where cities rely mostly on public transport. This also has benefits from the social justice point of view. Cars are expensive and not everyone can afford a car. If inability to afford a car affects accessibility of places of work and education, this has far reaching economic consequences for the individual as well as the society and the country at a large.

Ideally, a bus service for a city should be planned in such a way that people can confidently rely on it enough to avoid buying a car altogether. The environmental cost of a car comes not just from running it but also in producing it.

Trains and buses are, by far, the most economical modes of public transport as they can transport a large number of people at a very low cost. Trains are more economical and efficient than buses, however, running trains to places with low population density may not be the best approach. A combination of buses, trains and trams are an ideal solution to a transportation problem.

Research in developing countries have also shown that planning the last mile connectivity is essential for an effective public transport system. Buses form the ideal mode of transport for last mile connectivity.

The public transport infrastructure in Ireland is far from adequate, and while it is not quite as bad as in the USA, it leaves a lot to desire. In many places there are no bus routes, and where there are bus routes, there may not a good frequency of buses. So much so, that given two locations, it may not be practically possible to travel between the two with public transport because of the presence of missing links.

One exercise that may be done is to predict missing links in bus connectivity which prevent people from using public transport.

The assumptions here are these:

1. Commuters must be able to travel from any bus station in the city to any other bus station in the city.
2. Commuters may have to walk up-to a 500 metres to catch a connecting bus, but no more than that.
3. This must be true for the entire day 24/7

This knowledge can be used by the city planners to add or modify bus routes so that places where either 1 or 2 are invalid also come under the bus transport system. This can be done by adding new bus lines altogether, or by modifying existing bus routes to also serve some new stations along the route.

The third assumption is important here that any two places must be connected by the bus route 24/7. It is often the case that many bus routes reduce their frequency at night or completely stop plying. While the demand during those hours may be low, the unavailability of buses would mean that people cannot rely on them and are forced to buy personal vehicles, thereby incurring all the production cost of all the cars.

The problem essentially is a form of connected components. The idea is to find out subgraphs that are unconnected, and then connect them by changing the bus routes.

Both GraphX and GraphFrames facilities for finding connected components. However, we will prefer GraphFrames as it is supported on both Scala and Python, and will not tie us down to one language only. This is important as availability of talent is one important reason to prefer one one programming language over another, and python talent is easy to find as the language is very popular.

The steps to do so will be as follows:

1. Find a list of all stations by looking at the entire dataset in with regular spark RDDs. This can be found by looking at entries where atStop=1. Cache this set of stations for later use. We need to do this step separately because if we take a certain day some buses may not be running and serving some stations, and then we’ll miss those stations entirely. However, it is safe to assume that if a station is active, it will have some bus serving it in the entire dataset, and even if one bus serves it we will have made a note of it.
2. For every hour interval for both weekdays and weekends do the following:
   1. Add all stations as nodes
   2. Take all physical buses
      1. If a bus travels to two stations in the same line, then add an edge between them
   3. Take all pairs of stations (regardless of whether there is a bus between them or not).
      1. If the haversine distance between the two stations is less than 500m, then add an edge between them. This is because we have the assumption that commuters may have to walk upto 500 metres to catch a connecting bus. The haversine distance is not accurate as people will not walk as the crow flies, but we don’t have any other information in the dataset that will allow us to get the actual on-fot distance, so we will use this as a proxy.
   4. Find connected components using Graph Frames, and then print all the connected sub-graphs. Also, store all sub graph information with the nodes we had cached earlier. Now that we have all the nodes labelled with the connected sub-graph, we can move to the next phase of this algorithm

The next value addition would be to suggest stations which may be connected easily with the least cost. Two unconnected sub-graphs may be connected by adding just one edge, although adding more edges will be beneficial.