

# Taxi Trajectory Analysis

EDAA - G04

Diogo Rodrigues  
Eduardo Correia  
João Sousa



# Context

**Taxis** are an important part of the **transportation** system, especially in **large** cities.

Over the years, the taxi industry has been **evolving** rapidly due to the increasing **efficiency needs**.

This resulted in installing mobile data terminals in each vehicle and provide information on **GPS** localization.






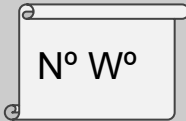



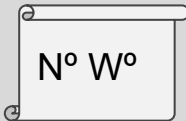



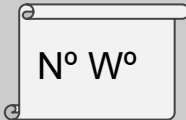
# Dataset

We're given a dataset from a **Kaggle** competition containing around **1.700.000** entries with information about **taxi trips**.

Coupled to this, a graph representing **Porto's** network of **roads** was retrieved from **OSM** and contains around **300.000** nodes.

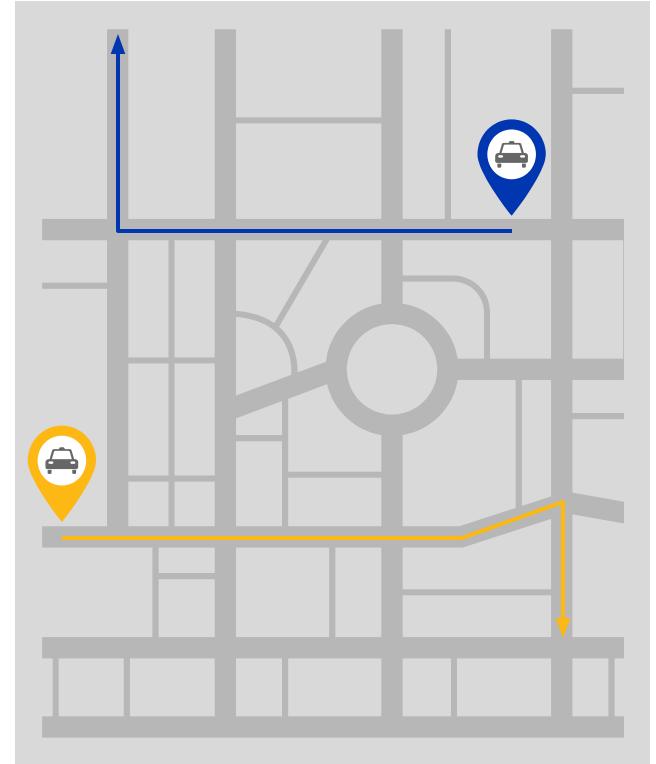


# Dataset sample

	Call Type	Missing Data	Timestamp	Polyline
Taxi A				
Taxi B				
Taxi C				

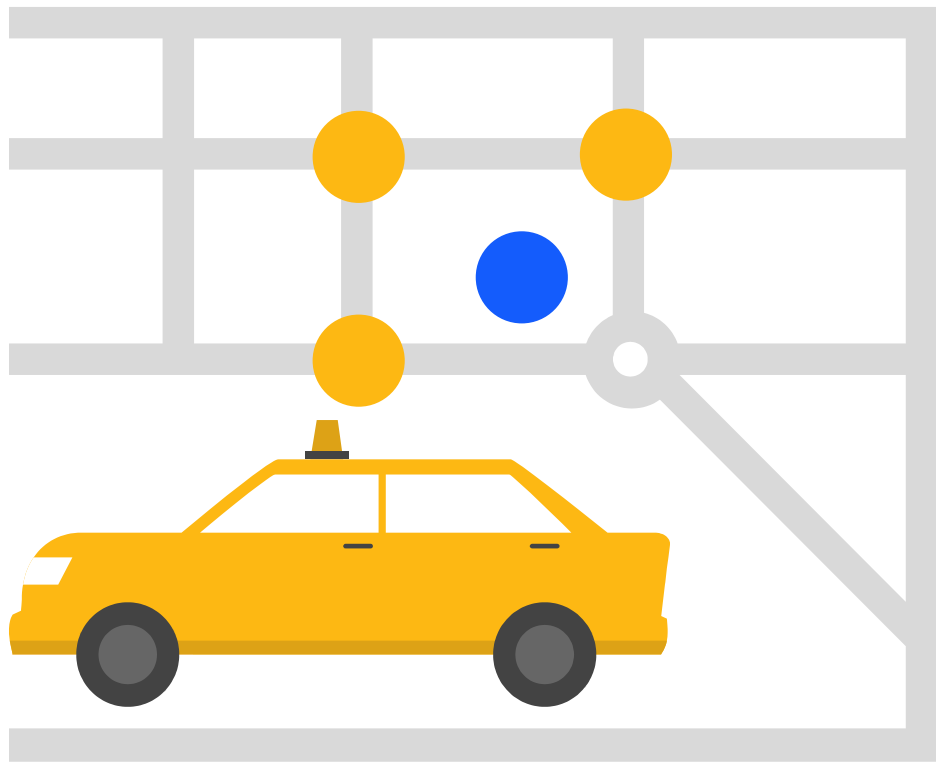
# Goals

The goal of this project would be to analyze **taxi trajectories**. Taking into account the **size** of the data we will be dealing with, we have to implement **efficient data structures** and **algorithms**.



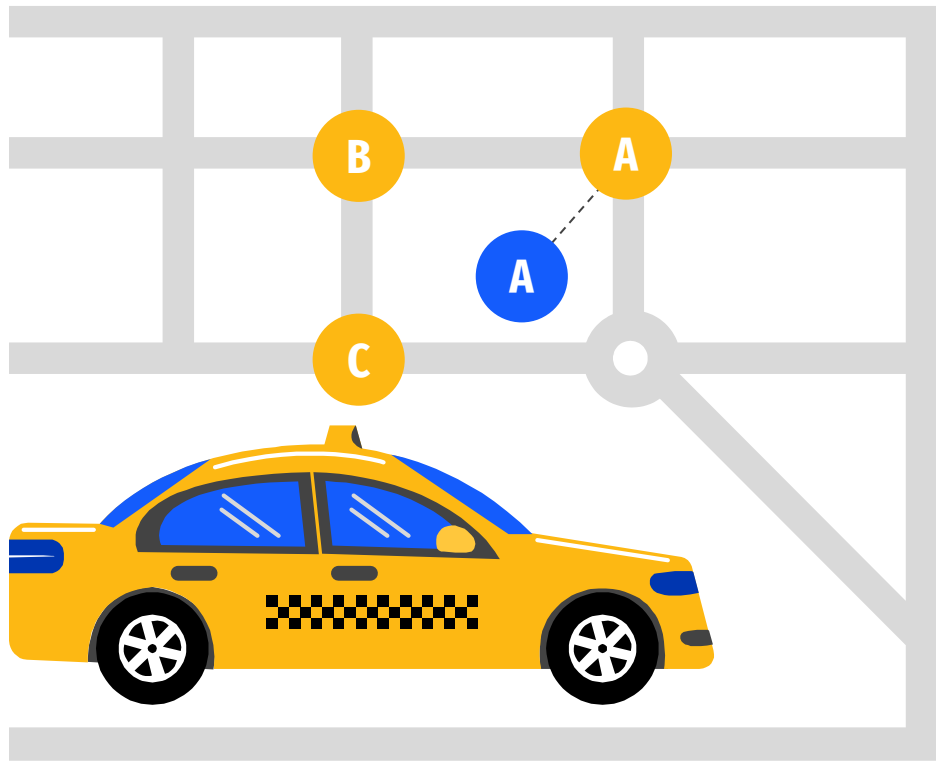
# Problem definition

However, we're struck with the problem of **mapping GPS coordinates** with **network nodes** that represent the real-life roads.



# Solution

Our problem might be reduced to a **nearest neighbour search (NNS)**, in which for each **taxi** position we locate the **nearest node** in the **graph** that represents the **roads network**.



# Quad-trees

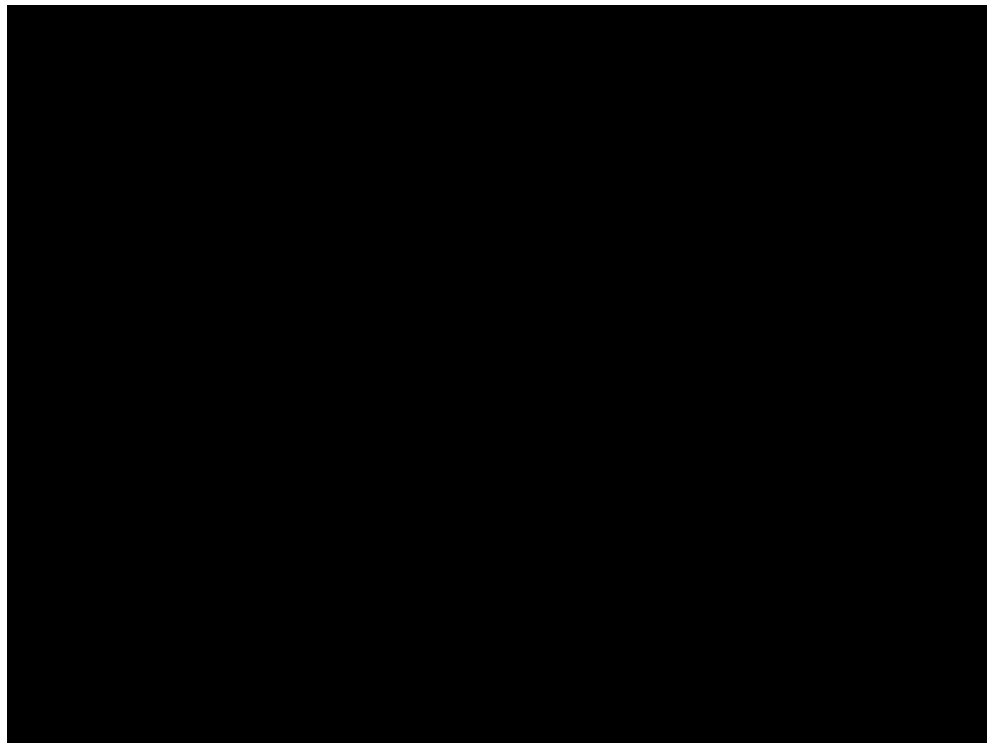
Since we're working with two-dimensional data, we might subdivide the plane into sections of four partitions.

This would allow us to locate the nearest neighbour of a point using a **NNS** by performing a search in the tree.





# NNS in Quad-Tree



# Voronoi Diagrams

The **Voronoi diagram** represents a set of regions in which a point, located in any of them, would have its closest node be the one located in the center of that region.

After constructing these diagrams, we only have to locate the region in which a point is located (point location).



# Slab decomposition

For that purpose, the **simplest data structure** we might use is slab **decomposition**.

It is based on subdividing  $S$  using vertical lines that pass through each vertex in  $S$ .

The region between two consecutive vertical lines is called a **slab**.

