School of Science/ Computer Science and IT

# COSC2636 Big Data Management
**Assignment 3**

Semester 1, 2019

## 1 Introduction
This is an individual assignment, to be submitted electronically using the Blackboard facility. A submission link will be enabled on blackboard closer to the submission date. It is due 23:59 Friday of Week 12, and contributes 15% towards the aggregate of 100 marks.

Assignment 3 is centered around the ACM SIGMOD paper “Searching trajectories by locations: an efficiency study” to build a simple efficient trajectory search engine. The paper can be found within the directory of Assignment 3. Please read this paper carefully and solve the problem introduced in later sections.

### 1.1 Plagiarism
All code or other material that is not original must be fully credited. That is, any material that is copied or derived from another source must be clearly identified as such and the original author must be identified. Sometimes students assist each other with an assignment, but end up working together too closely, so that the students' separate solutions have significant parts in common; unless the solutions were developed independently, they are regarded as plagiarised.

Plagiarism is a very serious offence. Any submissions determined to be a result of plagiarism will be deemed as an academic misconduct and harsh penalties apply. It is also an offence for students to allow their work to be plagiarised by another student. You should familiarize yourself with the university website for Academic Integrity Policy, Procedures and Guidelines. (https://www.rmit.edu.au/students/student-essentials/rights-and-responsibilities/academic-integrity) All work is to be done individually and plagiarism of any form will be dealt with according to the RMIT plagiarism policy.

### 1.2 What to Submit, When, and How

#### 1.2.1 When
This assignment is due at 23:59 Friday of Week 12.

#### 1.2.2 What
A zip file: which includes your complete code with the code provided by us. A query file showing some test cases to evaluate your code. Note: we will prepare test cases to evaluate the correctness and efficiency of your program.

Naming convention: StudentNumber_A3.zip

### 1.2.3 How

You are required to submit your solution electronically using the Canvas facility. A submission link will be enabled on Canvas closer to the submission date.

### 1.2.4 Penalties for late submissions

Late submissions of assignments will be penalised as follows. For 1 to 5 days late, a penalty of 10% (i.e. 10% out of total marks, not 10% out of your marks) per day.

### 1.2.5 Special Consideration

If unexpected circumstances affect your ability to complete the assignment you can apply for special consideration. If you seek a short extension, you can directly contact the lecturer. For longer extensions, you must follow instructions provided at
http://www1.rmit.edu.au/students/specialconsideration

## 1.3 Preparation Tasks

The code skeleton is provided in Canvas. Use the skeleton to implement the seven functions missing in the file "IKNN.java"

## 2 Task: Implement seven missing functions (key steps) (15 Marks)

**Problem Description:**
Given a set of existing routes in social network, a passenger wants to go from source A to destination B, your need to implement a method to return the k nearest direct routes (user can follow to travel from A to B) efficiently. The basic requirements are below:

1. k, A, B are arbitrary and can be input by users.

2. Using the searching idea from paper to achieve early termination, without checking all candidates in the dataset. Therefore, a brute-force method that scans all candidates is not acceptable.

**Dataset:**
We provide a dataset of real people's travel routes in Los Angeles. You can find it in the directory "/src/index/la_trips.txt" of the IKNN code package.

For example,
558483,33.595985412597656,-117.71573638916016
558483,34.05152130126953,-118.29872131347656
558483,33.70815658569336,-117.7821273803711
558483,33.74482345581055,-118.41141510009766
Stores a route of 4 points and the route id is 558483. Each line in the file represents a particular stop's latitude and longitude of this route.
Another file called "/src/index/la_points.txt" stores all the unique points covered by those routes after our pre-processing.

**Step-by-Step Guidance to complete this assignment:**

a. Read paper
  i. Section 1 to 4.1 of the paper.
  ii. Understand the input, output and search paradigm (Algorithm 1).
  iii. Familiar with the operations of R-tree (which you have learnt in lecture 2-3 and tutorial & lab in the first 5 weeks).
b. Have a look at the LA trajectory dataset (available at /index/la_trips.txt in the IKNN code package).
c. Understand how to store and index trajectories.
  i. Store trajectory as points.
  ii. Create R-tree index based on all the points (see /index/la_points.txt in the IKNN code package).
  iii. Building a mapping table to show the relationships between point and trajectory. (can be maintained in the main memory)
d. How to search k nearest routes by a pair of points?
  i. How to compute the distance from query to the route?
  ii. Build spatial index.
  iii. Filter impossible routes in advance based on bound comparison.
  iv. Refine the remaining candidate routes.

**Task Description for this assignment:**
Note that in this assignment, we support most part of the code, and what you need to

implement is actually d(iii) and d(iv) highlighted above (details are in Sec 4.1 of the paper). In particular, we delete 7 key steps (detailed tasks can be found in IKNN.java), and you need to implement these 7 key steps only to make the code runnable. The code and exemplar dataset can be found in the directory of Assignment 3 at Blackboard.

**Before implementing the code:**
You need to create a database to insert all trajectories to a database, for later use at "refinement" step (step d(iv)) the points of dataset. Here, we give the following guidance on how to do the insertion of data.

1. Install MySQL
2. Run SQL file in the db folder "src/db/tb_la_dataset.sql".
3. Change the setting in Settings.java.

**Sample Test Case:**
We provide some sample test cases for your reference during your implementation/test. Please refer to Sample – TestCase.txt file in the Assignment 2 folder.

**What you will learn:**
First, you need to understand the distance model between query and trajectory, and know how to compute the bound, and observe whether some trajectories can be skipped without checking, and see how different parameters may affect the performance, like the number of returned results. Further, you need to see how many routes can be discarded compared with the brute-force, and the effect on the result of various k.

## 3 Grading criteria

We will evaluate your program in terms of its correctness and efficiency on our prepared test cases.