# FIT5211: Programming Assignment No. 1

**This assignment contributes 20% towards your mark in FIT5211**

It is concerned with the implementation of Divide and Conquer Algorithms.

During Week 5 we have discussed Divide and Conquer and specifically a Divide and Conquer algorithm to find the closest pair of points in a set of points in 2-dimensional Euclidean space. We now want to solve the same problem using Manhattan Distance (this is simply the distance that you would have to walk on a grid where you can't take diagonal shortcuts as opposed to the Euclidean distance which measures the shortest straight line.). Your task is to adapt and implement the Divide and Conquer algorithm and to evaluate it empirically. If you need a refresher regarding the algorithm, please consult Kleinberg and Tardos, Section 5.4, or Levitin, Section 4. However, we

1. Use the ADT class for points in 2-dimensional Euclidean space discussed in the lecture in Week 1 (see lecture notes), and extend the class this with a distance function for pairwise Manhattan distance.
2. Based on this point class, implement an ADT class that models a point set. It will need a constructor as well as functions to insert and delete points in the set (you do not need to implement union, intersection, or other set operations at this point).
3. Extend the point set class with a method to find the closest pair of points. Your method should return the coordinates of the two points that have the smallest pairwise distance in the set as well as this distance. Bear in mind that we are now using Manhattan distance. Give a brief discussion what (if anything) you need to change in the algorithm to accomodate Manhattan distance and justify your answer_ precisely. Implement two diff erent versions as methods in the point set class you have implemented above:
    a. Implement the naive $O(n^2)$ *brute-force* algorithm.
    b. Implement the *divide-and-conquer* algorithm.

4. Perform an *empirical runtime evaluation* for both your algorithms using the Python timeit library. Plot runtime graphs for both algorithms for a reasonable rage of input sizes (you do *not* need to plot these using Python, but you are of course welcome to do so). Discuss briefly whether your plots substantiate your expectations.

5. Define the *runtime recurrence relation* that would arise if you would not pass sorted lists down the recursion but instead would sort the lists explicitly on each level of the recursion.

6. **For 10% bonus points** If you are interested in data visualisation in Python you may (sooner or later) want to have a look at Matplotlib or even special visualization libraries, such as Bokeh. Note that Matplotlib is an integral part of SciPy. You can use one of these packages to visualize how the algorithm searches (i.e. visualize the point sets, the splitlines, and ultimately the results) and even to let the user interactively (graphically) enter a point set (the latter is easier in Matplotlib than in Bokeh). There is no need to do any of this for this assignment, but if you do so voluntarily, a successful solution will attract up to 10% bonus points. Note that your tutors will not necessarily be able to support your work on this optional part of the assignment.

**Submission**:

Your submission will be due on Tuesday 12/9/2017 COB via Moodle Assignment 1 Dropbox. The only accepted format for the submission is as an iPython notebook. You must include the answers to all questions in a single notebook and send this as a zipped attachment. The only exception are the runtime plots which you may attach as additional PDFs if you do not plot them in Python (they may be created using other tools). Please zip the notebook file before attaching it. **Submissions not following moodle submission formats may not successfully upload and will therefore automatically receive zero marks.** Late penalties apply as per unit guide.

**Coversheet**: It is a [University requirement] (http://www.policy.monash.edu/policy-bank/academic/education
/conduct/student-academic-integrity-managing-plagiarism-collusion-procedures.html) for students to submit an assignment coversheet for each assessment item. Faculty Assignment coversheets can be found at http://www.infotech.monash.edu.au/resources/student/forms/.
You must attach a copy of the completed coversheet to your online submission.

**Late Penalties**: Assignments received after the due date will be subject to a penalty of 10% per day of late submission. No submissions will be accepted later than 1 week after the due date unless an extension has been granted in writing (email).

*Good luck!*