FIT5211: Programming Assignment No. 2

This assignment contributes 20% towards your mark in FIT5211

The assignment uses the same kind of Partition ADT (union-find data structure) and the minimum spanning tree algorithms that we discussed in Weeks 9 – 10.

Context

Electricity grids can be designed using a 'minimum cost spanning tree' algorithm, by taking into account the minimal cost of installing transmission links and structures, rather than simply using the shortest distance from a node to the power station. To achieve this, one can start by assigning each node in the network its own partition and calculate the distances between all pairs of nodes. Nodes with the lowest cost edges can be connected first in the grid, forming spanning trees. These trees can then be successively merged in a forest (partial spanning trees) until we obtain a full single spanning tree. In other words, we are successively joining clusters of connected data points or nodes until there is only a single big cluster left.

Sometimes, the electricity providers prefer splitting the electricity network into fragments or micro-grids (also known as 'Islands'). This helps improve the self-healing, reliability, and resiliency of the electrical system of remote towns or communities. It also reduces overloading of the micro-grid and helps save the cost of installing and maintaining lengthy and expensive high voltage transmission links over fragmented communities.

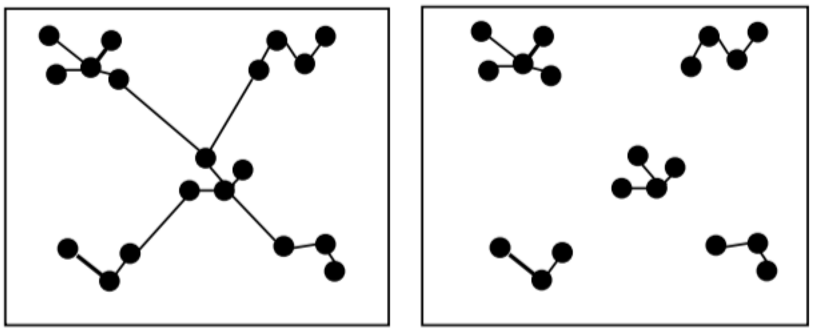
Figure 1: (a) The MCST connecting all the nodes (b) Micro-grids without some of the expensive links (with k=5 clusters)

A particularly simple approach to achieve the above form of spanning tree using clustering is by using the Kruskal's algorithm we have studied earlier in the course. In Kruskal's algorithm, we successively merge trees in a forest (partial spanning trees) until we obtain a full single spanning tree. This type of clustering is referred to as single-link clustering and is a specific type of hierarchical clustering. You can read more about hierarchical clustering here.

In this task, we start by assigning each data point (representing location of customers) its own partition and successively join clusters of data points or customer nodes until we end up with a single big cluster. To do so we can use the Partition ADT (union-find data structure).

Task:

Suppose you work for an electricity generation and retailing company which wants to setup micro-grids within a rural region. The company has done its research and knows all the geo-coded customer locations (nodes) in the region. For the purpose of this assignment, we will simply consider the customer location as points $p_i = (x_i, y_i)$ in two-dimensional space R^2 with no further information attached to them.

1. Using the ADT class for points in 2-dimensional Euclidean space from before, implement a function to generate a test set of n random nodes/points, where n is a user-definable parameter.
2. Implement an ADT class for Partition (union-find). It must support the operations for generating a new partition (i.e. a set of sets), for generating a new set within the partition, for merging two sets (union) and for finding a set to which an element belongs (find). You should use the tree-based implementation. If you need a refresher regarding the Partition ADT, consult the lecture notes for spanning trees. Wikipedia also has a good and compact explanation here. (*****Note****: In the pracs, we implemented the union-find procedure. You now have to implement it as an ADT class*)
3. Using the Partition ADT and test data function that you wrote for the first two tasks, implement the clustering procedure described above. If you did not manage to implement the tree structure or the forest structure in the earlier tasks, you may do so using the simple list-based union-find implementation, but this will only attract partial marks.
4. Extend your forest-based Partition ADT class from above with path compression.
5. Ultimately, we are aiming to find k-clusters i.e. we want to the electricity providers to see the structure of the micro-grid if they decide to commission k micro-grids such that all the nodes are covered in the vicinity of these sub-stations. Extend your implementation such that it stops when k clusters have been achieved and return those clusters, where k is a user-definable parameter.
6. Implement functions/methods to let the user query whether two given points (nodes) belong to the same cluster (micro-grid).
7. Define a function to compute the Dunn index, a measure for the quality of the clustering. You can find the definition of the Dunn index in the Wikipedia entry on clustering.
8. Compare the forest that you have generated for the k-clustering to the full minimum cost spanning tree. Give a brief, but precise characterization of the sets of edges that are in the MCST but not in the forest of the -clustering.

You may choose to do the following task for bonus points.

Bonus Point (10%)

For 10% bonus points as in the last assignment, you could use Matplotlib or Bokeh to visualize how the algorithm works (i.e. visualize the point sets, the cluster connections as they emerge, 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 Friday 20/10/2017 at 11:55pm via Moodle Assignment 2 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. **Submissions not following moodle submission format may not successfully upload and will therefore automatically receive zero mark**. Late penalties apply as per unit guide.

Coversheet:

It is a [University requirement] (http://www.policy.monash.edu/policybank/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 Penalty:

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!