CITS3210 Algorithms Project

Voronoi Tessellations and Visualisations
Due: week 13, Thursday, November 2, 2012, 5pm

The algorithms project in 2012 will focus on Voronoi Diagrams and Visualisations.

Suppose we are given a set of points in the two dimensional plane. A Voronoi Diagram (or tessellation) is a partitioning of the plane into regions where each given partition is the region closest to a single point (a site) in the set. These are used extensively in graphical information systems, meteorology, economics, anthropology and so on.

The Delaunay triangulation is the dual graph of the Voronoi tessellation. It is also the triangulation of the points (i.e. the sites are vertices with as many non-crossing edges are possible) that maximizes the minimum of all angles, or the triangulation such that no vertex appears within the circle with three mutually connected vertices on its circumference. It is reasonably easy to compute the Delauney Triangulation from the Voronoi Tesselation, or vice-versa.

Your task is to research, implement and report on an algorithm that computes the Voronoi Tessellation of a set of points. You may chose which algorithm, and which programming language to use (although you will only recieve marks for work you implement, not work already available in libraries). Furthermore, you may also choose which metric to use (for the Voronoi tessellation): the Manhattan metric possibly simplifies some of the mathematics, at the cost of producing a less elegant diagram. You may also consider heuristic or probabalistic implementations (these may be easier to implement, but can be harder to verify).
As a motivating application for this project, we will suppose that the Voronoi Tesselation is used to visualize data that is sampled from a two dimensional plane. Given an image we select a set of points in that image, and then compute the Voronoi tesselation. Then we colour in the cell of the Voronoi Tesselation with the colour of that single point. This (should) provide a reasonable visualisation

You are encouraged to work in pairs, but individual efforts are also acceptable. However the same standards and marking scheme will be applied to pairs and individuals. You should submit:

- An electronic copy of your commented and neatly formatted source code, directions to compile and run your algorithm, and some test data, sample outputs, demonstrating both the correctness and complexity of the algorithm. **40%**

- A report of at most 6 pages (**60%**) including:
  1. An introduction outlining the variant of the problem you have chosen to address **10%**
  2. A description of the algorithm you have chosen to implement, including pseudocode, explanations and references for any resources you have used. **10%**
  3. A careful explanation of how the algorithm works. **15%**
  4. A derivation of the asymptotic complexity of the algorithm. **15%**
  5. Empirical data gathered from running your implementation, demonstrating correctness (or not) and complexity. **10%**

The report should be **both** submitted to cssubmit, and submitted as a hard-copy by the due date.

This is a very challenging assignment. The efficient \( O(n \lg n) \) algorithms are extremely complicated, and even the most basic algorithm \( O(n^2 \lg n) \) is far from trivial. Hopefully you will find it interesting and rewarding, but remember to start early and ask for as much help as you need. However, students wishing to avoid the complications of computational geometry may choose to do a simpler version that only uses integer points in the plane (so, for example, graph traversal becomes a feasible approach).

Resources will be made available through the unit web-page:
http://undergraduate.csse.uwa.edu.au/units/CITS3210/material.html,
and you are strongly encouraged to use the discussion forum.

Good luck.