CS 577: Intro to Algorithms
Program 2
Spring 2011

Minimum Spanning Trees

The assignment deals with implementing algorithms for finding minimum spanning trees. It is multi part and you can do those parts individually without doing the entire assignment. But, if you do all the parts, it will count as 2 programming assignments and can give you as many (or more) points as an exam.

Due date: If you want to do this assignment, you should have at least one part completed by April 12th. You may have until April 28th to complete additional parts and until the end of the semester to complete the extra credit.

To begin, here is a link to an excellent set of slides produced by Kevin Wayne at Princeton University.
http://www.cs.princeton.edu/~wayne/cs423/lectures/mst-4up.pdf

Throughout this assignment description, I have provided links to places where you can find additional information. In addition to these sources our textbook is a good source.

Part 1. (20 points) Implement Prim's algorithm for building minimum spanning trees.
http://en.wikipedia.org/wiki/Prim%27s_algorithm

Recall, that in this algorithm you will be adding the shortest edge that adds a new node to the tree. Your algorithm should begin with the tree that contains 2 nodes connected by the shortest edge in the graph.

The algorithm is heap based. But, you do not need to implement a heap – if you are programming in Java, you can simply use the heap structure provided.

Part 2. (15 points) Implement a Fibonacci heap data structure. This data structure has better amortized complexity than the binomial heap. Here is a link to a nice Java applet for animating the operations on a Fibonacci heap and the pseudo-code for its implementation.

Part 3. (20 points) Implement Kruskal's algorithm for building minimum spanning trees.
http://en.wikipedia.org/wiki/Kruskal%27s_algorithm

This is algorithm requires maintaining a sorted list of edges. A variety of data structures are possible. Often a heap is used. But, for this part of the assignment you do not need to implement a heap data structure. If you are programming in Java, you may use the heap structure provided. You can also simply presort the edges and then maintain a list (or array) in sorted order.

However, to ensure that you don't introduce cycles you will need to implement a Union-Find data structure. These are also called disjoint set data structures. Here is a link with more information: http://en.wikipedia.org/wiki/Disjoint-set_data_structure

Part 4. (20 points) Implement Boruvka's algorithm for building minimum spanning trees. This was the first algorithm given for the problem. It dates to 1926 and was developed to help design an efficient electricity network for Moravia. There is a modern description of the algorithm included on Kevin Wayne's slides (link above).

Part 5. (20 points extra credit) Implement the Karger-Klein-Tarjan randomized algorithm for constructing minimum spanning trees.

Whether there is a linear time method to construct minimum spanning trees is an open question in computer science. Such a method would need to run in O(m + n), where m is the number of edges and n is the number of nodes. The Karger-Klein-Tarjan algorithm runs in expected linear time. It is randomized and its worst case could be nonlinear.

This algorithm is described in Kevin Wayne's slides. Before you implement this algorithm you should have done Parts 1, 3 and 4 of this assignment. In other words, no credit for this part until you get the 3 deterministic algorithms for minimum spanning tree running.