Duncan Fisher
4/22/17

HW4 Discussion

Finding real world data sets for weighted undirected graphs was harder than I expected so I decided to generate my own graphs. I used www.random.org to generate a set of random strings representing vertices, then passed these to a python3 script (generator.py) which outputs a random graph in the format specified by this assignment. I decided the graph could be thought of as a cluster of stars where vertices correspond to the name of each star and edge weights represent the distance in light years between them.

Now, imagine you are a spaceman with a spaceship capable of traveling up to one light year before refueling at the nearest star. You want to find the shortest path from star A to star B by jumping from star to star, but you can only travel distances less than one light year at a time or you will die in deep space. In the real world there is obviously an edge of some distance from any star to another but the distance is not necessarily less than one light year. The python script that generates each random graph therefore includes only edges with weights less than one, and each star has approximately 10 neighboring stars within this radius. The random graphs generated are not always completely connected, but the probability that there exists a path between any two vertices is relatively high. The analogy for our constellation is that there is a path between most pairs of stars that the spaceman can traverse but there are perhaps a few small clusters of stars which are isolated from the rest.

To test my shortest path finding algorithm I generated a graph in g.txt with 1000 vertices and ~5000 edges because each pair of vertices had about a one percent chance of sharing an edge and there are n(n-1)/2 distinct pairs of vertices where n is the number of vertices. I also implemented algorithms to determine if a graph is connected and exactly how many connected components it has. One optimization I made was implementing a hashtable data structure called StringMap which maps strings to integer values in O(1) time. I augmented my graph class with this data structure in order to make constant time accesses to elements of the adjacency list.

I would like to note that the analogy of the randomly generated graph representing a constellation in not completely accurate. Our universe is a metric space and therefore points in it obey the triangle inequality:

$$d(x,z) <= d(x,y) + d(y,z)$$

where d(a,b) represents the weight of edge (a,b). However there are subsets of this graph which do not obey the triangle inequality.

This algorithm was surprisingly fast even for a graph with 5000+ edges! It calculated the shortest path between two nodes in a few milliseconds, which is less time than it took to randomly generate the graph.