

Minimum-Cost Spanning Trees

CSCI 5870

Goal

Implement Kruskal's algorithm for finding a minimum-cost spanning tree.

Details

This project has four main goals:

1. It must correctly implement Kruskal's MCST algorithm
2. It must use a heap and a union-find structure for efficiency
3. It must work with an arbitrary amount of data
4. Compare the algorithm to the brute-force (i.e., the $\Theta(e^2)$ search-for-min) algorithm

Data format

The input is an undirected graph of arbitrary size. Each node has a unique, arbitrary name, and distances between nodes are positive real numbers.

Input lines have the following format:

name₁/name₂/distance

Names are arbitrary, although they will not contain a / character.

Algorithm comparison

We know from the text and in-class discussion that using simple data structures (e.g., a simple list) leads to $\Theta(e^2)$ performance, while using more efficient structures leads to $\Theta(e \lg e)$ performance.

You must generate several test cases (at least ten), with between 10 and 10,000 nodes (one case must have 10, one must have 10,000). When creating edges, use the probability $P = \frac{1}{2^{\log_{10} n}}$ that two nodes are connected, where n is the number of nodes. If necessary, add just enough edges to connect the graph.

Compare the two implementations in terms of run-time (user + sys times from the UNIX `time` command).

What to turn in

Turn in an electronic copy of your code, and the results of your comparisons.