Lecture 32: Graphs: Mininum Spanning Trees

11:00 AM, Apr 4, 2021

Contents

Objectives

By the end of this lecture, you will know:

- What a minimum spanning tree is

By the end of this lecture, you will be able to:

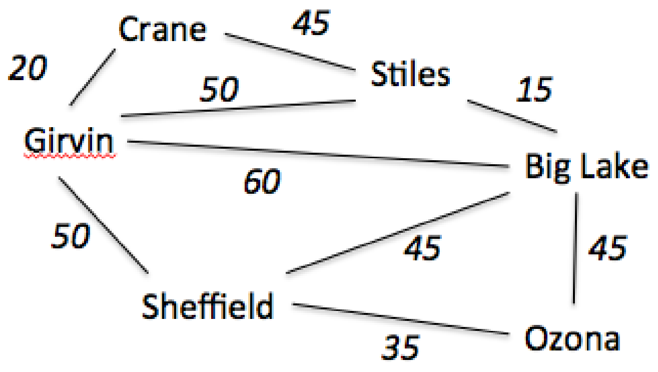
- Use Jarnik/Prim's algorithm to construct a minimum spanning tree
- Use Kruskal's algorithm to construct a minimum spanning tree

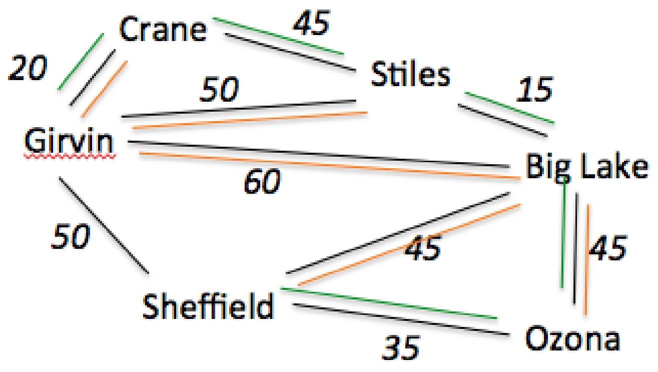
1 Minimum Spanning Trees

We've been talking about how to find shortest paths between two nodes in a graph. What if instead we needed to connect all nodes using the minimum total edge weight?

A typical motivating example for this is laying electrical wires among towns: given an undirected connected graph in which nodes are towns and weighted edges show the distance between pairs of towns, we need to select a subset of the edges such that (a) every town is included, (b) the subset of edges forms a connected graph, and (c) the total weight of the selected edges is minimal (relative to other sets of edges we might choose). Such a subset of edges is called a *minimum spanning tree*.

As an example, consider the following graph (using a collection of towns in rural Texas – the edge weights are only approximate). The upper figure shows the original graph. The lower figure shows two spanning trees for the graph: the orange has weight 220 and the green has weight 160. The green one is minimal.





For the rest of today's notes, we refer to section 19.8 of a textbook called Programming and Programming Languages (written by Brown CS Professor Shriram Krishnamurthi)

`https://papl.cs.brown.edu/2019/graphs.html`

We covered through 19.8.4 in this lecture, and will cover 19.8.5 in the next lecture.

Please let us know if you find any mistakes, inconsistencies, or confusing language in this or any other CS18 document by filling out the anonymous feedback form: `https://cs.brown.edu/courses/cs018/feedback`.