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 A Recap on Finding Paths

We started our path-finding segment motivated by the problem of travel planning (the bus routes). Let's return to that: what benefits would you get from each of DFS, BFS, and Dijkstra's if you were implementing an application to compute driving directions?

- BFS would get you the path with the fewest number of hops
- Dijkstra's would get you the lowest-cost path
- DFS would ... ???

In practice, DFS is often faster than BFS, depending on the nature of your data (and sometimes, it is worse). If you are building an application involving graph search, test out both algorithms and see which does better on your data in practice.

Okay, so we're most likely going to use Dijkstra's. What weighting criterion should you use? Lowest cost, fastest driving time, lowest tolls, most scenic? There's a long list of options here, and some tools let the user decide which metric to use. But there's something to note about the options listed so far:

they were all from the perspective of the driver

Think back to our SRC lecture: there are many stakeholders affected by an algorithm. If we are going to configure an algorithm, we need to identify those stakeholders and determine which of their needs should be considered. For driving routes, there are issues such as:

- what activities occur along the selected edges? If a street passes a playground or near a school, do safety considerations suggest using a nearby alternative?
- what activities don't occur along the selected edges? When should traffic be directed through local areas to support local businesses?
- Should multiple queries for the same start/end poins always give the same route, or should traffic be distributed along multiple streets?

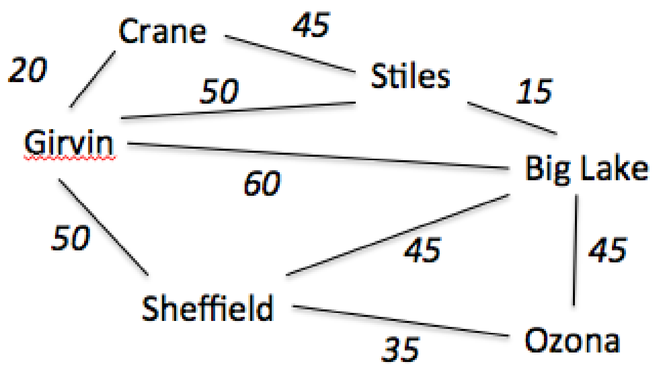
The list goes on. You could end up an intern or employee on a team that builds such software. You need to be prepared to raise questions about the stakeholders, good and bad impacts on them, and how technical choices affect the outcomes of that algorithm.

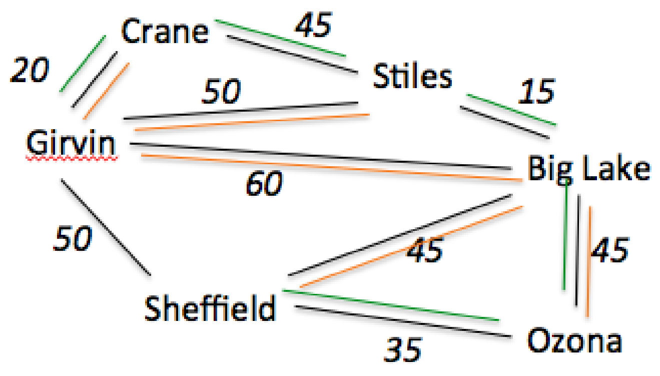
2 Minimum Spanning Trees

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`.