Prim’s vs Kruskal’s

Both of these algorithms are designed to generate a minimum spanning tree. Essentially, a tree that visits every node in a graph using the least amount of weights possible to do so. The main difference between these two algorithms is that Kruskal’s generates the tree by adding one edge at a time and Prim’s does so by adding one vertex at a time. Prim’s will generate a single tree each step of the way and Kruskal’s does not always since it can create two structures that will eventually be joined later. Essentially Prim’s is more efficient when you have a dense graph with more edges than vertices and Kruskal’s will perform better for sparser graphs.

I prefer Kruskal’s because for one I like how it works from a sorted list of edges and I like how instead of a tree growing it seems to be all over the place until it finally comes together at the end. Kruskal’s also seems to have simpler data structures.

Dijkstra’s Shortest Path

Practical examples for using Dijkstra’s Shortest Path:

* The most obvious example would be to use it on a map to determine the shortest distance from your destination to your location, such as with Google Maps.
* You could use it to route data over the internet. Using the algorithm to determine the route to your destination with the lowest latency.
* It is use in games (and I imagine robots) to determine how a computer controlled actor can get from pt A to pt B in the shortest way possible.