**Diana Mann**

**CSC1 211**

**Fall 2023**

**Dijkstra vs. Kruskal**

**Dijkstra’s and Kruskal’s traversals**

Both Dijkstra’s and Kruskal’s algorithm are a form of traversing acyclic directed graphs. This kind of graph allows us to express information in two dimensions and is particularly useful for describing maps in ways that can be computationally solved. The traditional algorithm for this usage is Dijkstra’s.

This is not the only use of directed graphs however – many other applications exist where it is more important to connect *all the nodes* in the graph using the absolute lowest cost possible. Think of applications like computer networks and telecom, the communications backbone of the internet is at it’s core, a spanning tree design for network traffic.

For these applications, we use *spanning tree* type algorithms, such as Kruskal’s algorithm. In comparison to Dijkstra’s, Kruskal’s will always find the absolute best path, where the “best” path is the least expensive. (This cost can represent many things: Distance for physical systems, latency or bandwidth for communications, etc.)

In many ways these algos are similar. But there are differences in their implementation and ideal applications that should be kept in mind when choosing to use Dijkstra’s or Kruskal’s:

**Similarities:**

* They both select a path in an acyclic graph (or map, network tree, etc)
* They are both **greedy algorithms:** These are characterised by following the classic strategy of selecting the *best possible option* at every local step of the problem. The idea being; that local optimisation will ultimately yield the best possible global solution.
* This also means that ***these algorithms will only operate correctly on problems that have optimally-possible solutions!***

**Differences:**

* **Dijkstra’s** will always choose **the *shortest path possible to connect two paths, v, and u***in the acyclic graph.
* **Dijkstra’s** cannot be run in parallel in most implementations.
* **Kruskal’s** will always connect ***all the nodes using the shortest possible routes.***
* **Kruskal’s** uses a divide and conquer approach enabling parallel processing.

**Applications:**

* **Dijkstra’s is used for path finding** where the starting and ending nodes are known. It will always select the most efficient route between two points without knowing anything about the intermediate nodes, choosing the shortest distance at every stage.
* **Kruskal’s** is useful for **connecting all nodes** of a acyclic graph into spanning trees**.** This is useful in network protocol and other communication systems. It is particularly suitable also for creating Circuits in EDA applications, min/max optimisations of process controls, image analysis and cluster analysis.