**Chapter 4: Graph Algorithms (Needham & Hodler, 2019):**

*Pathfinding and Graph Search*

**Pathfinding algorithms:** built on top of graph search algorithms; explore routes between nodes; used to identify optimal routes through a graph for logistics planning, least cost call or IP routing and gaming simulation

Examples

1. Shortest Path (A\* and Yen’s) - find the shortest path between two nodes
   1. Calculated by relationship/edge weights



1. All Pairs Shortest Path and Single Source Shortest Path - for finding the shortest path between all pairs or from a chosen node to all others
   1. All Pairs: optimized calculation of shortest paths from all nodes to all other nodes



* 1. Single Source: shortest path from a route node to all other nodes (accumulating the least weight)



1. Minimum Spanning Tree - find a connected tree structure with the smallest cost for visiting all nodes from a chosen node by starting at a given node and traversing ALL nodes via lowest-weight paths



1. Random Walk - useful pre-processing / sampling step for ML workflows and other graph algorithms, select which direction to go / how to traverse randomly; aka drunkard’s walk

See Jupyter Notebook on my GitHub exploring the phenomenon of Random Walks and the CLT in python:

<https://github.com/krashr-ds/framingham-ms/blob/main/Random%20Walks.ipynb>

Explore a graph for:

* General discovery
* Explicit search

**NOT** necessarily computationally optimal

**Breadth Search**

Traverse across nodes at a level before descending to the next sub-level

**Depth Search**

Traverse down to the bottom of a subtree before moving to the next child or sibling node