Graphs, Adjacency, and Minimizing Algorithms

Student: Connor Robison

Instructor: Chad McDaniel

Course: CSC 307 H002

Date Submitted: November 17, 2023

Colloquially, there are numerous types of graphs, such as line graphs, bar graphs, scatter plots, pie charts, etc. Arguably, the most iconic type of graph is the plot of a function on the Cartesian plane. Most people likely think about the x-y coordinate plane whenever they hear the word "graph." However, in math and computer science, the term graph has a much more specific definition. And, due to centuries of Graph Theory, there are various algorithms and real-world applications associated with these graphs. This essay should be a brief introduction to graphs, implementations of graphs, and how procedures such as Dijkstra's and Prim's algorithms operate.

A simple graph is a pair (V, E) where V is a finite set of vertices and E is a finite set of edge pairs [1, pp. 9-10]. Vertices, or nodes, are fundamental units of a graph, and edges are used to connect two nodes. There are numerous subcategories of graphs, four of which are as follows: weighted, unweighted, directed, and undirected. The previously mentioned definition of a graph most accurately describes an unweighted, undirected graph. Weighted graphs and directed graphs include a few other aspects. For example, associated with each edge of a weighted graph is a number or a weight [2]. Each edge of a weighted graph essentially has a toll, and traversing any edge will bear a cost akin to the edge's weight. So, imaging the graph as a set of roads and intersections, if a person wanted to traverse from vertex A to vertex B on a weighted graph for the least amount of money, that would affect which path that person decides to take, i.e., the shortest past on a weighted graph may not match the shortest path on a similar, unweighted graph. On a directed graph, each edge has a starting vertex and an ending vertex [1, pp. 101)]. Sticking with the road analogy, traversing a directed edge is like driving down a one-way road. Once again, the shortest path on a directed graph may not match the shortest path on a similar, undirected graph.

More relationships can be defined regarding the vertices and edges of a graph. For example, two vertices are adjacent if they are connected by an edge. There are two common ways to represent adjacency. One way is using an adjacency list, which is an array of lists. Each index of the array corresponds to a vertex of the graph, and each node of the respective list represents an adjacent and traversable vertex [3]. An adjacency list is a relatively simple way to represent and manipulate a graph. An adjacency list can also be used to store additional data about a graph, such as weights and directed edges. Another way to represent a graph is with an adjacency matrix. An adjacency matrix is an $n \times n$ matrix, where n is the number of vertices in a graph. Each entry $A_{i,j}$ of the matrix represents the number of paths from vertex V_i to vertex V_j [1, pp. 115]. An adjacency matrix could also be used to store the weights of edges rather than the number of paths between two vertices. In general, a two-dimensional adjacency matrix requires less storage than a two-dimensional adjacency list, and runtime varies depending on the operation [4]. So, adjacency lists and matrices have various pros and cons. It is up to the user to decide which implementation is best suited for their project.

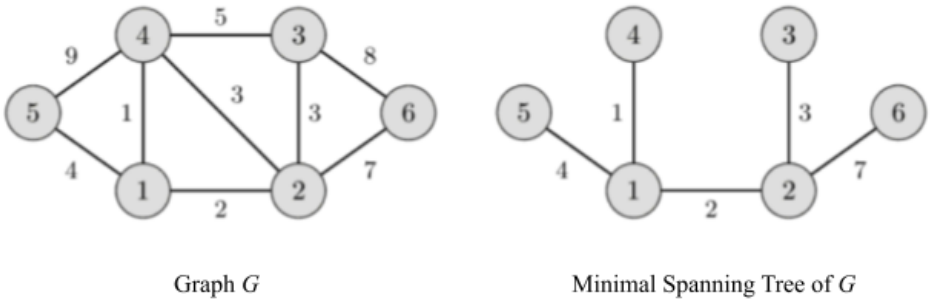

Figure 1: Minimal Spanning Tree from Prim's Algorithm. Adapted from [5]

Prim's algorithm produces a minimal spanning tree. Given a graph with n edges and m nodes, a minimal spanning tree will connect all m nodes with a subset of V where the sum of the edge weights has been minimized. Prim's algorithm starts at a random node and branches out from that vertex. Vertices and edges are added one at a time. Search all the edges for an edge that connects to the starting node and add the edge with the lowest weight to the minimal spanning tree. Once the tree has two nodes, search for edges that connect to the tree, and once again, add the lowest-weight edge and its accompanying node. Continue this process until the algorithm has visited every vertex or until the minimal spanning tree has $n - 1$ edges. See Figure 1 for an example of a minimal spanning tree of a graph. If Prim's algorithm is implemented by linearly searching an adjacency list or matrix, then the runtime will be $O(n^2)$ [5].

Dijkstra's Algorithm will produce the shortest path between two vertices on a graph. This algorithm starts by creating a list of unvisited nodes, a list of distances from the current node to the source node, and a shortest path list. Using a graph implementation, such as an adjacency list, this algorithm can check the distance from the current node to any neighboring nodes, mark the current node as visited, and add the node with the lowest-weight edge to the shortest path list. The shortest path will be complete once the target node is added to the list. See Figure 2 for an example of the shortest path between two nodes on a graph. Since this algorithm searches for unvisited nodes linearly and searches for edges using the adjacency list, the average runtime for Dijksta's Algorithm is $O(m \cdot log(n))$ [6].

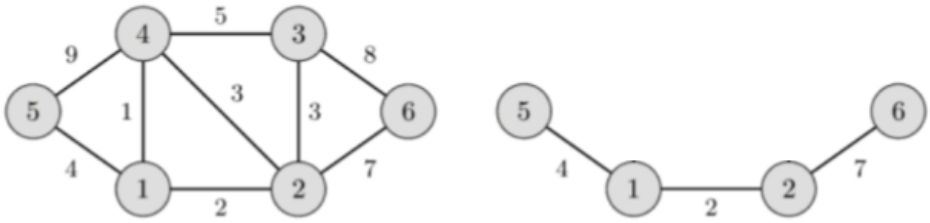

Figure 2: Shortest Path from Dijkstra's Algorithm. Adapted from [5]

Overall, graphs are a powerful data structure that can be utilized to solve a multitude of real-world issues. Prim's and Dijkstra's algorithms and only two of the numerous algorithms associated with graphs. Over the centuries since the conception of graph theory, an abundance of searching, shortest path, and minimal spanning tree algorithms have been invented. All of these algorithms help to solve issues regarding networking, social media mapping, resource allocation, GPS, producing search engine results, etc. Without graphs, these issues would likely be much more difficult to solve and understand conceptually.

References

[1] D. Grinberg, "An Introduction to Graph Theory," ArXiv.Org, 2023. Available: http://lynx.lib.usm.edu/working-papers/introduction-graph-theory/docview/2848591358/se-2.

[2] Weighted Graphs - Virginia Tech, https://courses.cs.vt.edu/~cs3114/Fall10/Notes/T22.WeightedGraphs.pdf

[3] “Adjacency List Representation of Graph,” Log2Base2, https://www.log2base2.com/data-structures/graph/adjacency-list-representation-of-graph.html.

[4] “Comparison Between Adjacency List and Adjacency Matrix Representation of Graph,” GeeksforGeeks, https://www.geeksforgeeks.org/comparison-between-adjacency-list-and-adjacency-matrix-representation-of-graph/.

[5] J. Kogler, *et al.*, “Minimum Spanning Tree - Prim’s Algorithm,” Algorithms for Competitive Programming, https://cp-algorithms.com/graph/mst_prim.html.

[6] A. Dey, “Dijkstra’s Algorithm,” The Algorists, https://www.thealgorists.com/Algo/ShortestPaths/Dijkstra.