# Pathfinding with Dijkstra's Algorithm and Heaps

Diana Tarazi and Sawyer Welden

A graph G is a pair of sets (V,E) where V is nonempty, and E is a set of unordered pairs of elements of V. The elements of V are called the vertices of G and the elements of E are called the edges of G. Sometimes we will write V(G) for the vertices of G and E(G) for the edges of G. Usually we represent the vertices by points on a plane, but the vertices can be any objects that are connected quantifiably. An edge might be a road between two cities, a telephone line between two houses, or a preference between a person and their favorite food.

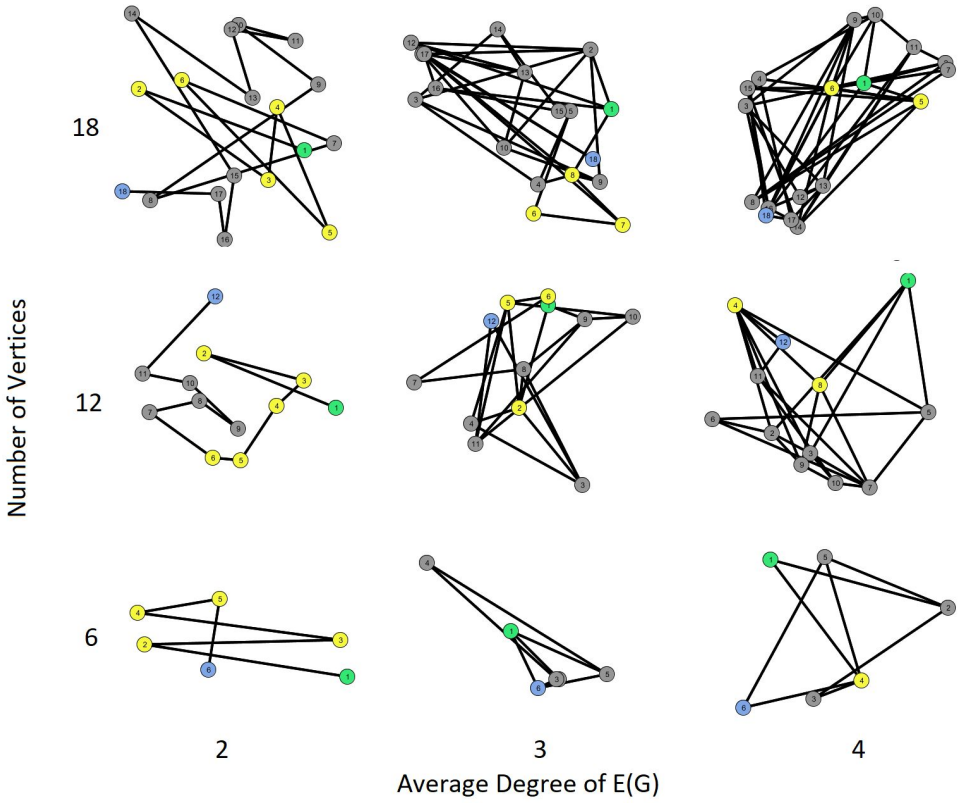
The diagram below is an example of a graph. There are 20 vertices, represented by Circles, and 30 edges, represented by line segments. The graph has a few vertices colored for non-obvious reasons. The colored vertices are the result of a pathfinding algorithm, which finds the shortest path  one vertex to another, often called the start and goal vertex respectively. In this diagram the green circle represents the start vertex and the blue vertex represents the goal vertex. The pathfinding algorithm has determined that the shortest path from green to blue is along the yellow vertices, and has labeled them accordingly.
p
But how do you define a shortest path? This graph is an example of a weighted graph, meaning there is a weight mapped to each edge. This is the cost of travelling along that edge. In the case of the example below and all the graphs used in this experiment, the weights are defined by the cartesian distance between the vertices. The shortest path from start to goal is the path with the shortest total cartesian distance travelled.

**Problem and Solution:**

- In a graph with a variety of vertices in different locations, we wanted to find the shortest path between two points 7in the graph. We used Dijkstra’s algorithm to solve this problem. Instead of using 7 p which comes with Dijkstra’s algorithm by default, we decided to use the Binary Heap. We think that this data structure is the most effective since we are only interested in finding the shortest path. The way Binary Heap works is it puts the least or the smallest element at the top of the tree or at the beginning of the array and it does not care about ordering the other members of the array. Thus, it uses its time efficiently for finding the smallest element which is what we are interested in when implementing Dijkstra’s algorithm.

**Correctness:**

We tested this algorithm on a wide range of inputs and made sure the implementation is correct. The shortest path was verified to be correct for all graphs. Degree is how many edges are connected to a vertex. Because each edge connects to two vertices, the number of edges is 2 times the total number of edges.



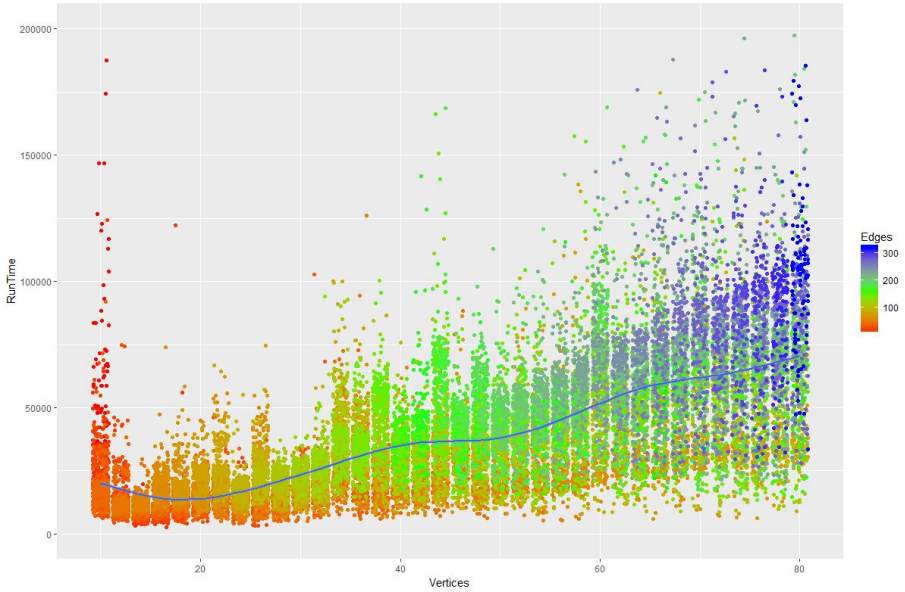
**Analysis:**

**Runtime by |V(G)| and |E(G)|**

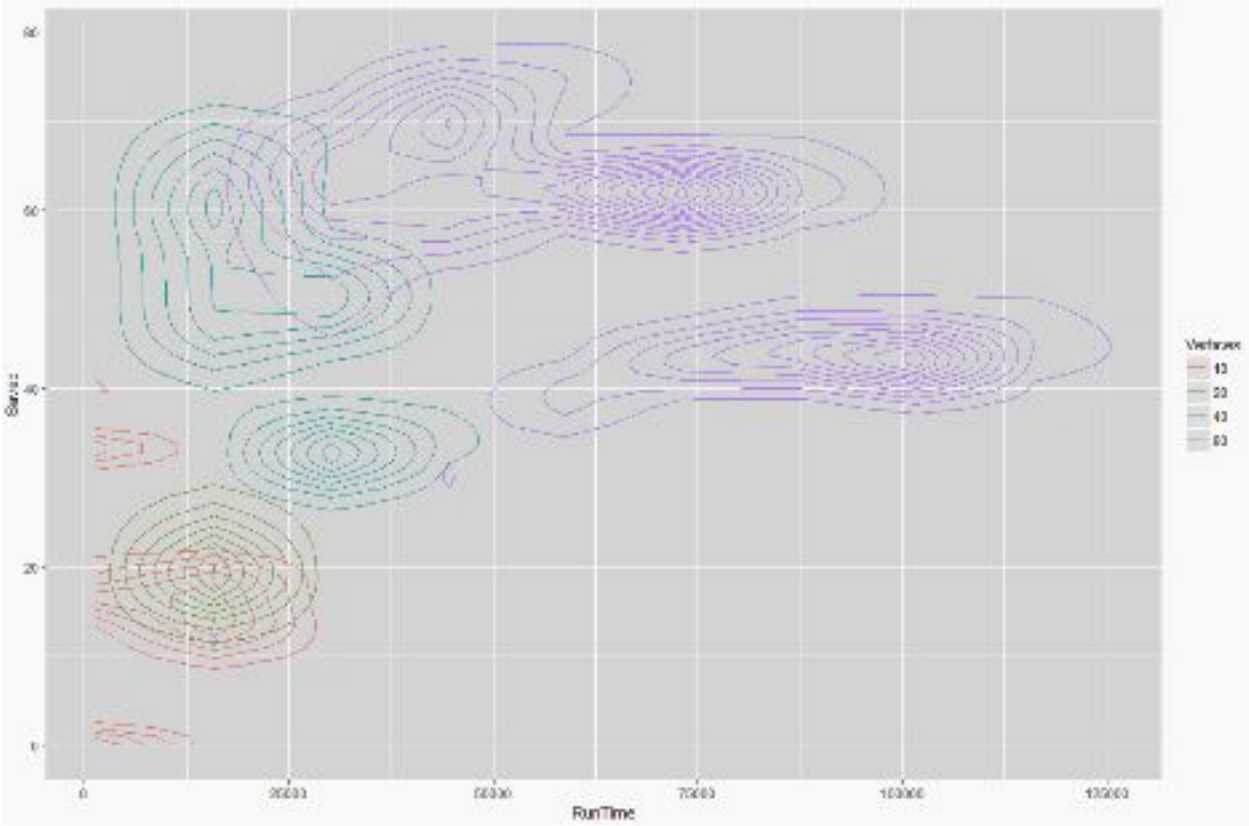
time in nanoseconds, 10,000 runs per line averaged

| Vertices | Edges | ArrayList | Time Saved with Heap | Time Saved |
|---|---|---|---|---|
| 10 | 10 | 4876.564 | 933.2075 | 19.14% |
| 10 | 20 | 4095.638 | 776.5309 | 18.96% |
| 10 | 40 | 6001.640 | 675.2440 | 11.25% |
| 20 | 20 | 7035.823 | 2206.891 | 31.37% |
| 20 | 40 | 10354.353 | 2478.826 | 29.06% |
| 20 | 80 | 13546.003 | 2478.826 | 18.30% |
| 40 | 40 | 16078.72 | 9325.477 | 58.00% |
| 40 | 80 | 25357.53 | 12508.336 | 49.33% |
| 40 | 160 | 36236.39 | 11644.812 | 32.14% |
| 80 | 80 | 42718.04 | 28503.94 | 66.73% |
| 80 | 160 | 67462.39 | 40906.01 | 60.64% |
| 80 | 320 | 93657.51 | 40167.00 | 42.88% |

As we can see, the time taken increases quickly based on the number of Vertices, increasing by around double every time the number of vertices is increased. This leads us to believe that the runtime of Dijkstra's Algorithm may contain the term O(V). Similarly, as the number of Edges doubles, the runtime increases by about 2 / 3, so this may indicate a runtime of O(V + ⅓E).



It is also interesting to note that the time saved using the binary heap appears logarithmic. It increases as the number of vertices increases, but decreases as the number of edges increases. The benefit of the binary heap is that the runtime of finding the smallest element in a set is O(1). This is used often, as the algorithm considers the vertex with the shortest path length from the start vertex. In contrast, to find the smallest element in an ArrayList, we have to sort the list, which is O(nLogn).




**Conclusion:**

Dijkstra's algorithm is a very simple and relatively fast algorithm for pathfinding in a graph. The use of a binary heap decreases the runtime of Dijkstra's algorithm logarithmically, but the runtime is so fast that it is almost unnecessary. For example, the 120,000 graphs made for the data tabulated in the analysis section took less than 1 second to generate and pathfind.