To begin the data was collected using to 50 node data set, and timing is recorded in nanoseconds. I also could not figure out how to get the Adjacency List and Matrix onto the same scatter plot, so they are in two separate sets of plot. Before we get to the algorithms, regarding the graphs, it seems that the Adjacency List is more efficient than the Adjacency Matrix. This is probably due to the abstraction I had to do to make both data structures work with the same Graph interface methods. The Adjacnecy Matrix became more expensive data wise than it should’ve been had I been using it on its own. Vice versa for the Adjacency List with time. Also as the data is vastly different between the tables, I will analyze the algorithms using the Adjacency List scatter plots as they more accurately represent the data.

The most jarring thing I noticed with the Nodes in path is how vastly different the recursive versions of DFS and BFS were. It seems the recursive versions has drastically more nodes in their paths than the iterative versions. This is probably since connection nodes are being added in a different order in the recursive version, making the search run for longer on the same set of source and destination nodes. Dijkstra and A\* are less surprising, it makes sense that they have less nodes since they both find the best path available.

Nodes explored was the least surprising out of all the scatter plots. It makes sense that Dijkstra and A\* explored a ton more nodes than any of the BFS and DFS algorithms because they are trying to find the best path, which means searching more nodes until a cheap path is found. The recursive versions of DFS and BFS have more nodes searched than the iterative which tells us why the paths are larger for the recursive versions because more was searched.

The execution times seem reasonable, but a bit off. I thought that A\* would by far be the slowest as it is Dijkstra with an added heuristic, but it consistently performs quicker than Dijkstra. As for the other ones, it makes sense that Iterative DFS/BFS is faster than recursive because callbacks are expensive. Also, it makes sense Dijkstra and A\* are slower than DFS/BFS because they are checking more nodes.

There is little to talk about regarding cost. It seems that in both the List and Matrix A\* has a slightly higher cost than Dijkstra. For the cost of A\*. I used (1+weight)\*distance, and for the cost of Dijkstra I used (1+distance). It makes sense as A\* multiplies the distance times the weight as opposed to just the weight. I thought that A\*’s cost would be more though.

In all, it seems for pure speed you will want to stick with DFS/BFS Iteratively, but if you want the best path you will have to opt for the slower Dijkstra/A\*.