The 5 search algorithms were each ran through 7 random samples of Start and Goal town pairs.

The metrics tracked were the length of the final reported path, the number of nodes searched, and the physical time elapsed running the function (measured in processor ticks).

It should be noted that the time metric is extremely unreliable for algorithm analysis. It is practically non-deterministic from my perspective, and there are many hardware factors that have a much stronger influence on time elapsed than the scale of search done. Furthermore, the overhead from my implementations of the algorithms seems to be the dominant contributor to time. All trials were ran in the same instance of the program. The first 2 trials for each algorithm had magnitudes more time elapsed, I suspect as I did more trials, efficiency was increased due to things like cacheing.

Undirected search is obviously not a contender for efficiency in path length or nodes visited. It holds no memory of previously visited nodes, and therefore will waste a lot of time revisiting places and wandering aimlessly. It always searches exactly as many nodes as the final path -1. Interestingly, undirected search is competitive in the time metric. I suspect this is due to the low scale of this problem and the lack of overhead in running the algorithm.

Depth first search consistently found shorter paths than Breadth First search. Sometimes Breadth first visited fewer nodes, sometimes Depth first visited fewer nodes. It makes sense for these algorithms to perform similarly in this context, since the environment has no implicit concept of depth hierarchy between towns- it is simply something that is constructed from the perspective of the algorithms. It should be noted that the memory appetite of breadth first search is trivial in a problem this size.

IDDFS was the most expensive in terms of time and nodes visited. This makes sense to me given how much work it repeats. Again memory is not a concern in a problem of this scale, so none of the advantages of IDDFS meaningfully show in this context. On some trials, IDDFS found a path with a higher length than simple depth first found. This is baffling to me- as far as I understand IDDFS will find the solutions with less depth first. I believe this points to the presence of a bug in my implementation. [IDDFS makes the same arbituary choices of what successors from a node to follow first that depth first does].

Surprisingly, depth first was slightly better at finding the shortest path than best first search. I suspect that this is a sign that my heuristic needs to be improved- or that adjacency data is composed of many short physical distance connections between towns. Best First search excels in visiting few nodes, putting all other algorithms tested to shame. For reasons beyond my comprehension, the first 2 trials of best first used ~100,000 ticks. I can only guess that this is skill issue from my implementation (it is very cursed that I had to iterate through the priority queue constantly to remove bad duplicates), but then I am at a complete loss for why the other trials have a tick counter in the hundreds.