Luke Doukakis - 011847368
Professor Markus Eger
CS 4200.01

Lab 1: Pathfinding - Report

Implementation

As the basic structure for all four algorithms is similar, I had each function simply call another function named “search” to perform the graph search. This search algorithm works as a loop, popping off the first node in the frontier and adding its neighboring nodes that have not yet been expanded, doing this until it either finds the goal node or the frontier is empty (i.e. we’ve traversed the whole graph). The next node to be popped off the frontier depends on separate parameters used in the function call, determining whether we’re using a stack or queue, and if/how the frontier is to be sorted before popping the next node. For Depth-first Search, a stack is used, and for first breadth-first, a queue. Greedy and A* work as a modified breadth-first search, but sort the frontier before popping off the next node (greedy using the heuristic of the node, and A* using the sum of the heuristic and the path length to the node).

Once the goal node is found, there needs to be some way to retrace the path to that node in order to return the path and its length. I implemented this with the use of dictionaries - throughout the traversal of the graph, each node’s id is added as a key to two dictionaries, one whose value is the node that it came from, and the other whose value is those two nodes’ associated edge. Once we’ve reached the goal, we can then use these dictionaries to iteratively backtrack through the path we took to get to the goal node and add up their costs, ultimately getting the path and its total length.

Performance

The different search functions all performed pretty much as expected. Depth-first Search, in both the Austria graph and my custom graph (a map of Skyrim), seemed to perform better than Breadth-first Search, having visited less nodes and returning a shorter path. In the infinite search however, the depth-first search never found a path however, having to break after searching too long. This was expected, as an infinite graph causes this algorithm to spiral down a path forever, and you would have to get lucky for it to even find the goal node. The Greedy Search and A* search, when used with the heuristic, did perform considerably better than BFS and DFS, with A* performing the best. Interestingly enough though, in the Austria graph, Greedy Search visited and expanded far less nodes than A*, but returned a path just slightly longer than A*. This goes to show how much of a difference it makes when A* uses the length of the path to each node as a factor when sorting the frontier.

One additional test I performed was on my Skyrim graph for calculating the journey from Solitude to Riften, to see the difference in paths the algorithms find when I change up some edge lengths. On the way from Solitude to Morthal, there is a river with a bridge crossing it. The bridge allows the path to Morthal to be much shorter than if the path had to go around the end of the river. In my test, I changed the edge length from Solitude to Morthal to reflect the case where this bridge is no longer there, just to see if the algorithms responded, which indeed they did. A* changed its path to skip Morthal entirely, and instead travel along the southern border to Markarth to Falkreath to Riften.

Skyrim Graph

I used a measuring tool to come up with both the edge lengths and heuristic values for the graph. The heuristic simply uses the raw straight-line distance from the given city to the goal city, Riften. For the edge lengths, I used the measuring tool in conjunction with my knowledge of how the paths actually look from the Elder Scrolls game to come up with some rough edge lengths to represent the paths between the cities. The graph helps to visualize the test I described earlier. With the bridge connecting Solitude and Morthal as shown, the A* algorithm takes the path Solitude-Morthal-Whiterun-Riften. Without the bridge however, it decides it's better off to skip Morthal entirely and instead head Southwest to Markarth and travel along the border to get to the goal.