ECE457A Assignment Q6 Short Paper

Christopher Chung, 20883463

Description of Implementation

Breath-First Search (BFS)

Implemented in Python, BFS was implemented using a queue and *visited* set to keep track of nodes that have been visited. The queue contains a tuple, containing the number of moves taken so far (cost), a node on the fringe to be explored, and an array containing the path taken to the fringe node from the start node.

Items are continuously dequeued until the queue is empty, at which point the search could not find the goal node. For each dequeued item, it is checked against the goal node. If the dequeued item is not already in the visited set, it is added, and its children (valid moves) are enqueued.

Depth-First Search (DFS)

DFS was implemented the exact same way as BFS but implemented with a stack instead of a queue.

A* Search

Implemented in Python, A* was implemented using a priority queue and *visited* set to keep track of nodes that have been visited. The priority queue contains a tuple holding the next node to be explored, the node's f(n) value, the node's g(n) value, and an array containing the path taken to the fringe node from the start node.

Items are continuously dequeued from the priority queue, based on the lowest f(n) = h(n) + g(n) value, until the priority queue is empty, at which point the search could not find the goal node. For each dequeued item, it is checked against the goal node. If the dequeued item is not already in the visited set, it is added, and its children's f(n) value and current g(n) value are calculated. The item is then enqueued.

The heuristic function, h(n), was implemented using the Manhattan distance. The equation is shown below.

$$h(n) = \left|x_{goal} - x_n\right| + \left|y_{goal} - y_n\right|$$

This implementation was chosen because it gives the shortest amount of moves to a goal node if diagonal moves are not allowed on a 2D grid-based game. Although it does not consider blocked paths, it gives an accurate estimation, in most scenarios, of the distance to the goal at the current node. Based on this, it is easy to give priority to nodes in the priority queue.