Project 2 Fifteen Puzzle

Demonstration: https://youtu.be/dcZP8kCLAC8

For this project we were to build an AI algorithm that could solve the 15 puzzle game using A* Search. To accomplish this I made 2 classes which would be used to conduct a* Search on the list of puzzle pieces, the Node, and the Environment.

The Node was used to create a tree of board states based off of moves. Each Node consisted; of a board (Representing the current board state); a parent (Representing the previous Node); a cost (representing the moves taken before reaching that state); and a heuristic (Representing how close the puzzle is to be solved). The Node class works by calling a function called generate_moves, which instantiates more Nodes to be branched off of the current Node, each of these Nodes will have a boardstate with exactly one other piece swapped with the “-1 Piece” (The Blank Piece), simulating a board move, each of the Nodes will also assign their Parent to the current Node. A Heuristic is a function that estimates how close a state is to its goal state, I decided to use the Manhattan distance, an equation that goes through all the tiles and calculates how far they are from their correct position. This is being calculated in check heuristic.

The Environment class manages the overall search process for solving the 15-puzzle game using A* search. It initializes the search by setting up the starting board configuration as the initial_node and adding it to a priority queue (or "frontier") based on its priority, which is the sum of its cost (the number of moves taken so far) and heuristic (the estimated moves needed to reach the goal). During the search, the algorithm repeatedly explores nodes with the lowest priority, popping nodes from the frontier and expanding their possible moves. Each new move generates a "neighbor" node representing a new board state, which is added back to the frontier if it hasn't been visited before. By using a priority queue and the Manhattan distance heuristic, the algorithm efficiently navigates toward the goal state. Once a solution is found, the

reconstruct_path method traces the series of moves back to the starting configuration, allowing the steps to be displayed in order from the initial board to the solved state.

https://www.geeksforgeeks.org/deque-set-1-introduction-applications/