Khyati Mahajan

8-puzzle Problem

Problem formulation

The problem formulation consists of moving each tile such that we can reach the goal state from the initial state using the A* search algorithm for exploring the nodes to traverse a solution.

Program Structure

The program consists of the Node class which includes the A star algorithm. This is the main class and houses the algorithms used to find a solution. There are helper functions in the program which allow taking in information about the initial and goal states.

Global variables

The explored nodes and expanded nodes are counted using global variables.

The initial_state and goal_state functions

The initial and goal state functions take in the inputs to the problem at hand. They explain the format in which the input needs to be formatted (9 digits never repeated and written one after the other with spaces between each digit).

The Node class

Generator

The Node class includes a generator for a node structure which contains the following variables:

1. State: An array of length 9 containing the configuration for the state
2. Action: Info about which action (up, left, right, down) led to the current node
3. g_cost: Path cost till node
4. h_cost: Heuristic cost at node
5. parent: Parent node

The structure hence allows to keep track of the tree structure.

Legal Move Functions

There are 4 functions which check whether the next action would lead to a legal move, and output the node obtained from said legal move. If the blank tile is in a state where a certain movement is restricted, the movement function returns a “None” node. If the blank tile can move through the action being checked for, the function returns a new state which copies the configuration of the current state but swaps the values according to the current move.

Heuristic Functions and Helper

The heuristic functions are included for 2 heuristics:

1. Misplaced tile heuristic: sum of misplaced digits, i.e. digits in the grid which are not in the position of the given goal state
2. Manhattan distance heuristic: sum of the steps each digit in the grid has to take in order to reach its final position as specified in the goal state

The helper functions determines which of these heuristics is being used according to the input taken from the user.

A* algorithm

The A* algorithm maintains a priority queue of the nodes ordered by the final cost (step cost + heuristic cost). It also maintains a set of visited nodes to check for repeated states.
While the priority queue has elements in it, the algorithm checks whether the current node is in the set of visited nodes. If not, the algorithm goes on to call the functions defined earlier to check each legal move - up, left, right, and down. In each of these possibilities, the algorithm checks whether the generated node is in the visited node, and if not the new node generated by performing this move is added to the priority queue along with the path cost and the heuristic cost.

Printing the path taken

This is a helper function which prints information about the step count, the state reached, the action taken to reach this state, the costs associated with this state, and finally information about how many nodes have been explored and how many have been expanded.

Main Function

The driver function for the program, main takes in inputs from the user, creates a dummy node, calculates the initial heuristic cost, and initialises the tree bsy putting the initial state in the root node.

Sample Execution Results

Initial: 1 2 3 7 4 5 6 8 0
Goal: 1 2 3 8 6 4 7 5 0
Misplaced tile heuristic:
Steps: 8
Explored nodes: 24
Expanded nodes: 43
Manhattan heuristic:
Explored nodes: 12
Expanded nodes: 22

Initial: 2 8 1 3 4 6 7 5 0
Goal: 3 2 1 8 0 4 7 5 6
Steps: 6
Misplaced Tile heuristic:
Explored nodes: 8
Expanded nodes: 14
Manhattan distance heuristic:
Explored nodes: 7
Expanded nodes: 12

Initial: 1 2 4 5 3 6 0 7 8
Goal: 1 2 3 4 5 6 7 8 0
Steps: 16
Misplaced tile:
Explored nodes: 657
Expanded nodes: 1071
Manhattan distance:
Explored nodes: 245
Expanded nodes: 406