Classic Search: 8-Puzzle

Due Sep 25, 2020 by 11:59pm **Points** 50 **Submitting** a file upload

Goals

The goals of this assignment are:

- Understand efficient state representation, including a successor state function
- Implement iterative deepening
- Build basic heuristics
- Implement A* search

Background

The 8-Puzzle is a type of sliding tile puzzle, the most common of which is the **15-puzzle (https://en.wikipedia.org/wiki/15_puzzle)**

The 8-Puzzle consists of a 3x3 grid of numbered tiles, with one tile (#9) missing. The object of the puzzle is to get the tiles in a particular order, subject to the constraints of physically sliding one tile at a time into the open space.

For our puzzle, we will consider that the following state is our goal:

1	2	3
8	.	4
7	6	5

Note that this is different from the way the book describes the solution!

Tasks

State Representation

One important thing you need to think about is how to represent a state of the tile puzzle. A simple way is to use a list or tuple to keep track of what number is in each of the nine positions. Because the 8-puzzle is a simple toy problem, this solution will work. However, for more complex problems, memory might become an issue. If you want to stretch yourself a bit, consider how to represent each state of the 8-puzzle as a single integer. (There are a few reasonable ways to do this.)

Additionally, you will want to write a function that translates from your internal state representation to a visual representation (like the grid above) for debugging purposes. Name this function *visualize* and have it take a state and display a simple grid of the numbers that corresponds to the given state.

Problem Setup

I recommend (but do not require) using the book's structure for building the problem:

- Build a Node class with state, parent, action, and path_cost as member variables
- Write a function or method *child_node* that takes a node and an action and returns the child that is generated when the given action is taken
- Use the actions Right, Left, Up, and Down, which correspond to how a tile is moved into the blank space to generate the next state

Uninformed Search

Depth-First Search

Write a version of the book's graph_search function that implements depth-first search for this problem. Note that testing this might be tricky... how big is the state space?

Depth-Limited Search

Adapt your depth-first search to perform depth-limited search.

Iterative Deepening

Finally, write a function *iterative_deepening* that takes a state as input and returns a solution using iterative-deepening depth-first search by repeatedly calling your depth-limited search function. The solution should be returned as a list of a sequence of actions, starting from the start state, to arrive at the goal state.

Informed Search

Heuristics

Write two heuristics, implemented as functions named *num_wrong_tiles* (which counts the number of tiles in the wrong location) and *manhattan_distance* (which calculates the total manhattan distance for all tiles to move to their correct locations), each of which take a state as input and return an integer.

A* Search

Write A* search for this problem. Name the function *astar* and have it take which heuristic function to use as an argument.

Running your program

Your program should be runnable from the command line and accept exactly one command-line argument: The initial state, given as a single integer, where "0" corresponds to the blank. Thus, the integer 120843765 would

correspond to the initial state

1	2	.
8	4	3
7	6	5

For which the solution is: Up, Right

Your program should then solve from this given state using iterative deepening, A* with num_wrong_tiles, and A* with manhattan_distance, reporting its answer and time taken for each.

Grading

Submission

Using Canvas, submit your files

Rubric

Grades will be given approximately as follows:

State Representation - 10 pts

Iterative Deepening - 10 pts

Heuristics - 10 pts

A* Search - 10 pts

Command line and output - 10 pts

HW1 - Grades

Criteria	Ratings						Pts
State representation	**10 to >9.0 pts Full Marks**	**9 to >8.0 pts Minor Mistakes**	**8 to >6.0 pts major mistakes**	**6 to >4.0 pts incorrect answer but somewhat meaningful**	**4 to >1.0 pts Incorrect answer with major mistakes**	**1 to >0 pts No Marks**	10 pts
Iterative Deepening	**10 to >9.0 pts Full Marks**	**9 to >8.0 pts Minor Mistakes**	**8 to >6.0 pts major mistakes**	**6 to >4.0 pts incorrect answer but somewhat meaningful**	**4 to >1.0 pts Incorrect answer with major mistakes**	**1 to >0 pts No Marks**	10 pts
Heuristics	**10 to >9.0 pts Full Marks**	**9 to >8.0 pts Minor Mistakes**	**8 to >6.0 pts major mistakes**	**6 to >4.0 pts incorrect answer but somewhat meaningful**	**4 to >1.0 pts Incorrect answer with major mistakes**	**1 to >0 pts No Marks**	10 pts
A* search	**10 to >9.0 pts Full Marks**	**9 to >8.0 pts Minor Mistakes**	**8 to >6.0 pts major mistakes**	**6 to >4.0 pts incorrect answer but somewhat meaningful**	**4 to >1.0 pts Incorrect answer with major mistakes**	**1 to >0 pts No Marks**	10 pts
Command line and output	**10 to >9.0 pts Full Marks**	**9 to >8.0 pts Minor Mistakes**	**8 to >6.0 pts major mistakes**	**6 to >4.0 pts incorrect answer but somewhat meaningful/ missing sub portion of the requested answer**	**4 to >1.0 pts Incorrect answer with major mistakes**	**1 to >0 pts No Marks**	10 pts
							Total Points: 50