**University of California, Riverside**

**CS205-**

**Artificial Intelligence**

**Winter 2024**

**CS205 Assignment 1: The Eight Puzzle**

**05/07/2024**

|  |  |  |
| --- | --- | --- |
| **Name** | **SID** | **Email** |
| **Parth Shinde** | **862466119** | [**pshin022@ucr.edu**](mailto:pshin022@ucr.edu) |

I consulted the following in completing this assignment:

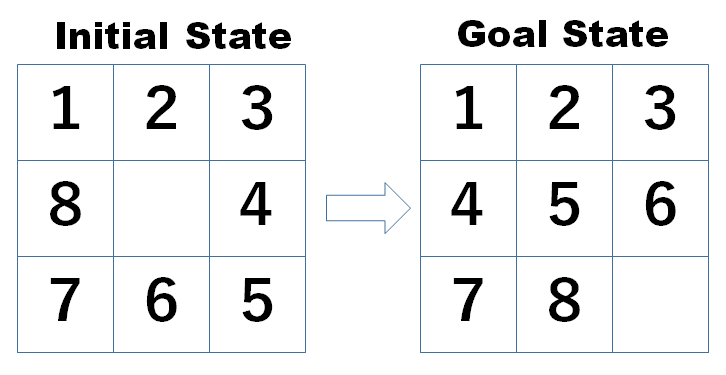
* CS 205 Lecture slides on [heuristic search](https://www.dropbox.com/sh/ftzvcnntl2j5eiu/AABcF7bSG1Na5cUrP3yICgyra/3__Heuristic%20Search.pptx?dl=0) by Dr. Eamonn Keogh
* [Content](https://realpython.com/python-heapq-module/) - I consulted for python3 library heapq (Unimportant – part of python in-built library)
* I also used python3 other in-built functions from the official documentation of python3.9((Unimportant)
* [Content](https://www.cs.princeton.edu/courses/archive/spring18/cos226/assignments/8puzzle/index.html#:~:text=The%208%2Dpuzzle%20is%20a,vertically%20into%20the%20blank%20square.) - I consulted for structuring the game format and the board(specifically the input) Npuzzle
* I also consulted the following links for understanding heuristic search and the different cost functions and their implementation.
  + <https://www.geeksforgeeks.org/8-puzzle-problem-using-branch-and-bound/>
  + <https://www.educative.io/answers/what-is-uniform-cost-search>

List of contents:

* + Cover page: (this page)
  + Description of assignment: Pages 3-4
  + Example Traceback: Pages 5 -7
  + Summary of results & Conclusion: Page 8
  + Code: Pages 9-10

**Introduction**

The sliding puzzle, particularly in its 3x3 (8 puzzle) form, presents a classic problem of rearranging shuffled tiles through sliding movements to reach a predefined goal state (for our case is the goal state is the normal ascending order in a row-major format in a the 3x3 matrix of tiles).



We address this problem using the A\* algorithm, enhanced with three distinct heuristic functions to evaluate the potential cost of moves. I have used Python 3 for this implementation, and it guides the puzzle towards the solution efficiently.

**Description of heuristics**

The A\* algorithm operates by selecting the state with the lowest associated cost. The cost can be described by the following:

The term *g*(*n*) represents the cost incurred in transitioning from the initial state to the current state, essentially capturing the cumulative expense of the operations performed to reach the current state. The term *h*(*n*) is a heuristic estimate of the remaining cost to reach the goal. For this project, we are required to implement three different heuristics.If the current considered state is not the goal state, then it is expanded and the next state with lowest cost is considered.

**Uniform Cost Search**

It is a part of A\* algorithm that has some with to the Djikstra Algorithm. Uniform cost search considers hardcoded as 0. It is an uniformed search that consists of simple cost function g(n) which is described as the total number of operations required to get to the current state. So , in our case ,it would be the cumulative number of moves that it takes to reach from the initial state to the current state(i.e the next state to choose from).

**Manhattan Heuristic**

It is considered an fully informed heurstic as it uses the - Manhattan heuristic function considers the cost as the sum of the Manhattan distances of location of each tile in the current state with the location of the corresponding tile in the goal state.

**Misplaced Tile Heuristic**

Misplaced tile heuristic function considers cost as the total number of tiles that are not in the correct location as per the desired goal state. Thus it is considered a partially informed heuristic as we do not know the exact location of each tile.

**Algorithm**

As per the assignment, the pseudocode of the algorithm is described in figure 3

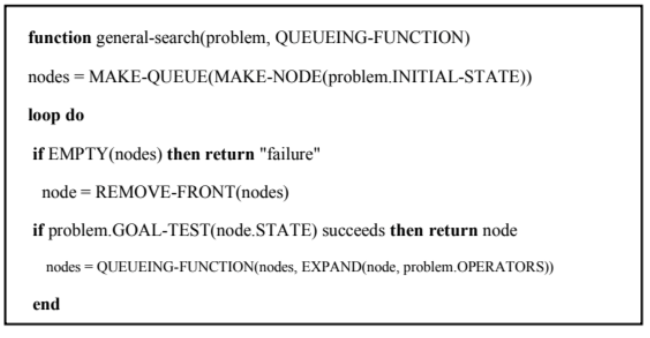


Figure 3: Pseudocode

In the assignment we have 7 puzzle inputs to verify and check the validity of our code. The code allows the user the following choices:-

1. It allows the user to input their NxN puzzle board (can be 3x3 or 4x4,etc)
2. It allows the user to input their initial state of the board .
3. It allows the user to do this in a easy to understand string format ,so the input is entered in a string format and represented in a row-major matrix format
4. The Initial and the output goal states are shown the reconstructed path with the depth of the steps is also shown in the output.
5. Trivial changes need to be done to solve the more than 8-puzzle(i.e Npuzzle problem) problem.

The following is the output of a depth 8 puzzle, the user inputs are highlighted:

Enter the puzzle size (e.g., 3 for 8-puzzle, 4 for 15-puzzle): 3

Enter the initial state as a string in a row-major format (e.g., '012345678' for 3x3 puzzle): 072461358

Enter heurstic cost function to be used:-

1:UCS

2:Manhattan

3:Misplaced Tile

3

This is your intial state of the board

\_72

461

358

This is your goal state of the board

123

456

78\_

---SEARCHER INTIALISED---

Solution found:

Depth of tree solver-->0:Board state given below

\_72

461

358

Depth of tree solver-->1:Board state given below

7\_2

461

358

Depth of tree solver-->2:Board state given below

762

4\_1

358

Depth of tree solver-->3:Board state given below

762

41\_

358

Depth of tree solver-->4:Board state given below

76\_

412

358

Depth of tree solver-->5:Board state given below

7\_6

412

358

Depth of tree solver-->6:Board state given below

716

4\_2

358

Depth of tree solver-->7:Board state given below

716

\_42

358

Depth of tree solver-->8:Board state given below

716

342

\_58

Depth of tree solver-->9:Board state given below

716

342

5\_8

Depth of tree solver-->10:Board state given below

716

3\_2

548

Depth of tree solver-->11:Board state given below

716

\_32

548

Depth of tree solver-->12:Board state given below

\_16

732

548

Depth of tree solver-->13:Board state given below

1\_6

732

548

Depth of tree solver-->14:Board state given below

136

7\_2

548

Depth of tree solver-->15:Board state given below

136

72\_

548

Depth of tree solver-->16:Board state given below

13\_

726

548

Depth of tree solver-->17:Board state given below

1\_3

726

548

Depth of tree solver-->18:Board state given below

123

7\_6

548

Depth of tree solver-->19:Board state given below

123

746

5\_8

Depth of tree solver-->20:Board state given below

123

746

\_58

Depth of tree solver-->21:Board state given below

123

\_46

758

Depth of tree solver-->22:Board state given below

123

4\_6

758

Depth of tree solver-->23:Board state given below

123

456

7\_8

Depth of tree solver-->24:Board state given below

123

456

78\_

TIME TAKEN FOR EXCUETION:- 1457.4253559112549 ms

**Summary of results:**

In the assignment, we have seven puzzle inputs to test our code. Users can choose from several options. I've run all the provided test cases and plotted the results using a script. Additionally, a CSV file is available for further analysis.

**A screenshot of a graph

Description automatically generated**

**Conclusion**

* The Manhattan heuristic is more efficient in guiding the search toward the goal state, likely due to its ability to provide a direct measurement of distance that correlates well with the actual cost of reaching the goal.
* Offers a middle ground, performing better than UCS but not as well as Manhattan. It provides a decent heuristic measure but lacks the spatial accuracy of Manhattan, which can lead to slightly more explorative paths.
* While thorough, its lack of heuristic guidance makes it the least efficient in terms of execution time and computational resources. It’s more suitable for scenarios where ensuring all possible paths are evaluated is more critical than efficiency.
* The results overall seem intuitive based on the content discussed in the lectures.

**Code Snippets:**

**A screen shot of a computer program

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

A screenshot of a computer program

Description automatically generated

The code ,the results and this report itself have been uploaded to this public [github](https://github.com/parthshinde1221/CS205-AI-8puzzle) repository.