

# Project: 1 CS 205. Introduction to Artificial Intelligence

## Due Date: May 8<sup>th</sup> 2021 at 11:59pm

### The Eight Puzzle

For this project I want you to write a program that solves the eight-puzzle. You will solve it using

- 1) Uniform Cost Search<sup>1</sup>
- 2) A\* with the Misplaced Tile heuristic.
- 3) A\* with the Manhattan Distance heuristic.

You may use any language, but the variable and function names for main “driver” program should approximately match the pseudocode in the book. If you have a good reason to ignore this directive, let me know in advance.

```
function general-search(problem, QUEUEING-FUNCTION)
  nodes = MAKE-QUEUE(MAKE-NODE(problem.INITIAL-STATE))
  loop do
    if EMPTY(nodes) then return "failure"
    node = REMOVE-FRONT(nodes)
    if problem.GOAL-TEST(node.STATE) succeeds then return node
    nodes = QUEUEING-FUNCTION(nodes, EXPAND(node, problem.OPERATORS))
  end
```

*The general search algorithm.*

I expect the code to be meaningfully commented, nicely indented etc. This is not a software engineering class, but if I cannot clearly review and understand your code. I will strongly lean towards a lower grade.

The code should be kept general as possible. In other words, your code should require only a modicum of effort to change to solve the 15-puzzle, or the 25-puzzle etc.

You can hardcode an initial state for testing purposes. But I want to be able to enter an arbitrary initial state. So sometime along the lines of the interface on the next page would be nice.

You may use some predefined utility routines, for example sorting routines or queue manipulation functions. However, I expect all the major code to be original. You must document any book, webpage, person or other resources you consult in doing this project (see the first day's handout).

You may consult colleagues at a high level, discussing ways to implement the tree data structure for example. But you may **not** share code. At most, you might illustrate something to a colleague with pseudocode.

You will hand in a two to five-page report which summaries your findings. I have appended a sample report to this file.

You **must** keep the evolving versions of your code, so that, if necessary you can demonstrate to the course staff how you went about solving this problem (in other words, we may ask you to prove that you did the work, rather than copy it from somewhere).

If you use any LLM/ChatGPT etc for the code or the report, you will get a failing grade in the class,

I will give you directions of how to hand in your report later.

---

<sup>1</sup> Note that Uniform Cost Search is just A\* with  $h(n)$  hardcoded to equal zero.

Dear Students

Below is sample of an eight-puzzle project report. This looks like a nice report, it would earn the student an A or A-. I am *not* claiming this report is perfect, or that it is the *only* way to do a high-quality project. It is simply an example of what high-quality work might look like.

Notes:

- For every Figure or Table in a report, there needs to be some text in the body of the report that explicitly points to it, and interprets it. The next sentence is a sample. *As we can see in Figure 1, the Fowl Heuristic is much faster than the Fish heuristic, especially as we consider harder problems.*

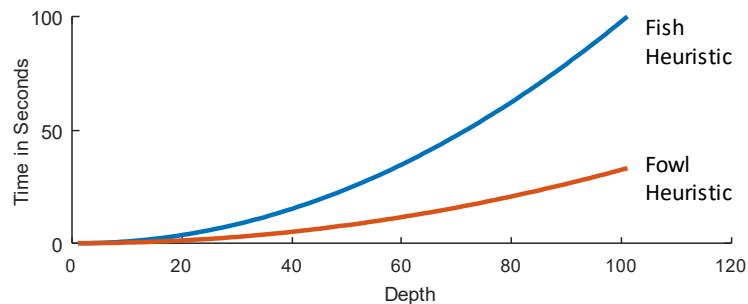


Figure 1: A comparison of two heuristics on the Rubix Sphere Problem, for increasingly hard problems.

- Look at your figures carefully. Did you label the X-axis and the Y-axis? Would a log Y-axis help?
- Do you have an *explicit* conclusion to your report? *As we have discovered empirically, the cost of owning a dog is approximately 240% the cost of owning a cat, over the life of the animal. However, this cost gap is closer for smaller dog breeds.*
- Do you have spurious precision? *“The algorithm took 23.798332322 seconds”* No! *“The algorithm took 23.7 seconds”* Yes
- If you copy a sentence from the internet or a book, without attribution, you will get a failing grade in this class.
  - **NOT ALLOWED:** Dr. Keogh asked us to create a program to solve a 3 by 3 sliding tile puzzle. A sliding puzzle is a combination puzzle that challenges a player to slide pieces along certain routes to establish a certain end-configuration. I begin by... This student will get a failing grade in the class. If you want to use someone else words or even long phrases, you must cite them. Put the text in quotation marks, and put a pointer (like this [1]) to where you found it.
  - **ALLOWED:** Dr. Keogh asked us to create a program to solve a 3 by 3 sliding tile puzzle. According to Edward Hordern *“A sliding puzzle, is a combination puzzle that challenges a player to slide pieces along certain routes to establish a certain end-configuration”* [1]. I begin...

[1] Sliding Piece Puzzles (by Edward Hordern, 1986, Oxford University Press, ISBN 0-19-853204-0)

Sue Mee

SID 12344321

Email sue@hmail.com

Date: May-06-2025

In completing this assignment I consulted:

- The Blind Search and Heuristic Search lecture slides and notes annotated from lecture.
- Python 2.7.14, 3.5, and 3.6 Documentation. This is the URL to the Table of Contents of 2.7.14: <https://docs.python.org/2/contents.html>
- For the randomly generated puzzles: <http://www.puzzlopi.com/puzzles/puzzle-8/play>

All important code is original. Unimportant subroutines that are not completely original are...

- All subroutines used from **heapq**, to handle the node structure of states.
- All subroutines used from **copy**, to deepcopy and correctly modify states.

## Outline of this report:

- Cover page: (this page)
- My report: Pages 2 to 7.
- Sample trace on an easy problem, page 8.
- Sample trace on a hard problem, page 9.
- My code pages 10 to 11. Note that in case you want to run my code, here is a URL that points to it online *<http://Github.ucr.joe/cs150>*

# CS170: Assignment 1: The eight-puzzle

Sue Mee, SID 12345678 May-07-2025

## Introduction

Sliding tile puzzles, as shown in Figure 1, are familiar mechanical toys. The 8-puzzle is a smaller version of the slightly better known 15-puzzle. The puzzle consists of an area divided into a grid, 3 by 3 for the 8-puzzle, 4 by 4 for the 15-puzzle. On each grid square is a tile, except for one square which remains empty. Thus, there are eight tiles in the 8-puzzle and 15 tiles in the 15-puzzle. A tile that is next to the empty grid square can be moved into the empty space, leaving its previous position empty in turn. Tiles are numbered, 1 through 8 for the 8-puzzle, so that each tile can be uniquely identified.



Figure 1: A picture on a 15-puzzle I bought to help build my intuition for sliding tile puzzles.

This assignment is the first project in Dr. Eamonn Keogh's Introduction to AI course at the University of California, Riverside during the quarter of Spring 2025. The following write up is to detail my findings through the course of project completion.

It explores Uniform Cost Search, and the Misplaced Tile and Manhattan Distance heuristics applied to A\*. My language of choice was Python (version 3), and the full code for the project is included. Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur

## Comparison of Algorithms

The three algorithms implemented are as follows: Uniform Cost Search, A\* using the Misplaced Tile heuristic, and A\* using the Manhattan Distance heuristic.

## Uniform Cost Search

As noted in the initial assignment prompt, **Uniform Cost Search** is simply A\* with  $h(n)$  hardcoded to 0, and it will only expand the cheapest node, whose cost is in  $g(n)$ . In the case of this assignment, there are no weights to the operators, and each expanded node will have a cost of 1. This reflects the fact that it takes the same amount of "finger effort" to move the tile in any direction.

## The Misplaced Tile Heuristic

The second algorithm implemented is A\* with the **Misplaced Tile Heuristic**. The heuristic looks to the number of “misplaced” tiles in a puzzle. ipsam voluptatem quia voluptas sit aspernatur performed the best, followed by the enim ipsam voluptatem, For example consider Figure 2:

```
A puzzle:
[[1, 2, 4],
 [3, 0, 6],
 [7, 8, 5]]

goal state:
[[1, 2, 3],
 [4, 5, 6],
 [7, 8, 0]]
```

Figure 2: A worked example of the misplaced tile heuristic

Not counting 0 (the placeholder for the blank/missing tile),  $g(n)$  is set to the number of tiles not in their current goal state position are counted; in this example,  $g(n) = 3$ . This assigns a number, where ipsam voluptatem quia voluptas sit aspernatur performed the best, followed by the enim ipsam voluptatem, Uniform Cost Search would.

## The Manhattan Distance Heuristic

The **Manhattan Distance Heuristic** is similar to the Misplaced Tile Heuristic such that enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est.

Using the example initial state shown in Figure 2, not counting the position of 0, it can be seen that tiles 4, 3, and 5 are out of place. Based on enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est Comparison of Algorithms on Sample Puzzles

As shown in Figure 3, Dr. Keogh provided use with the following test cases, sorted by depth of optimal solution. Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est.

Depth 0	Depth 2	Depth 4	Depth 8	Depth 12	Depth 16	Depth 20	Depth 24
123 456 780	123 456 078	123 506 478	136 502 478	136 507 482	167 503 482	712 485 630	072 461 358

Figure 3: Samples of nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni.

It was found that the difference between the three enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est ut perspiciatis unde,

Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est.

Dmnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. quasi architecto beatae vitae dicta sunt explicabo.

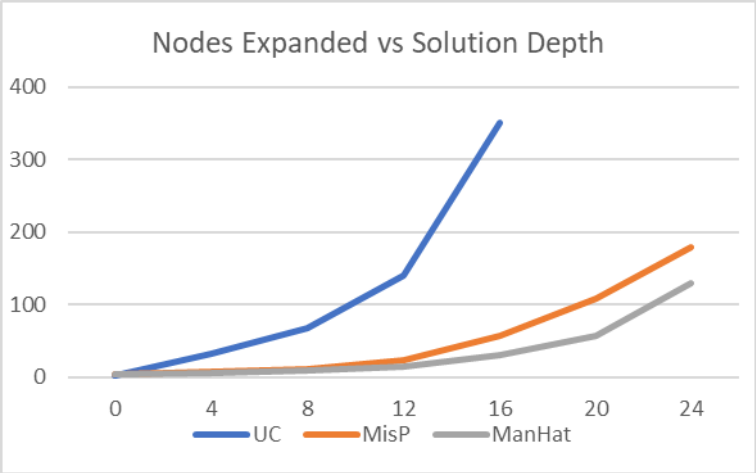


Figure 4: omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo

In Figure 5 we compute some accusantium Dmnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architect.

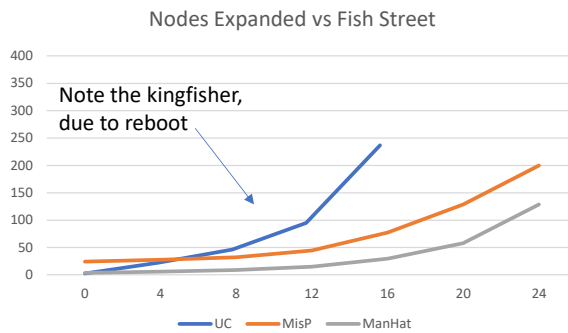


Figure 5: omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo

enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est.

Dmnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architect enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est.

## Additional Examples

I also create some accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi, This examples show that if the input is enim ipsam voluptatem quia voluptas.

## Conclusion

Considering the list of the three algorithms and the comparisons between them: Uniform Cost Search, Misplaced Tiles, and Manhattan Distance, it can be said that:

- It can be seen that out of the three algorithms, the ipsam voluptatem quia voluptas sit aspernatur performed the best, followed by the enim ipsam voluptatem, followed by voluptatem quia voluptas (or in this case, effectively also called Breadth-First Search).
- The Misplaced Tile and Manhattan Distance heuristics improve the voluptatem quia of algorithms. Uniform Cost Search,  $h(n)$  having been hardcoded to 0, became Breadth First Search, which has a time complexity of  $O(bob^{sue})$  and also a space complexity of  $O(van^{tan})$ , where *van* is the color factor and *tan* is the flavor of the solution in the magni dolores.
- While both the sed quia consequuntur Heuristic and voluptatem Distance Heuristic improved sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi, This examples show that if the input is enim ipsam voluptatem.

## The following is a traceback of an easy puzzle

Welcome to my 8-Puzzle Solver. Type '1' to use a default puzzle, or '2' to create your own.

2

Enter your puzzle, using a zero to represent the blank. Please only enter valid

8-puzzles. Enter the puzzle demilimiting the numbers with a space. RET only when finished.

Enter the first row: 1 2 3

Enter the second row: 4 0 6

Enter the third row: 7 5 8

Select algorithm. (1) for Uniform Cost Search, (2) for the Misplaced Tile Heuristic, or (3) the Manhattan Distance Heuristic.

3

The best state to expand with a  $g(n) = 0$  and  $h(n) = 3$  is...

[1, 2, 3]

[0, 4, 6]

[7, 5, 8]

The best state to expand with a  $g(n) = 1$  and  $h(n) = 3$  is...

[1, 2, 3]

[4, 5, 6]

[0, 7, 8]

The best state to expand with a  $g(n) = 3$  and  $h(n) = 0$  is...

[1, 2, 3]

[4, 0, 6]

[7, 5, 8]

Goal state!

Solution depth was 4

Number of nodes expanded: 13

Max queue size: 8

<EK says, The numbers above are just random numbers I made up>



The following is a traceback of a **hard** (depth 16) puzzle.

Welcome to my 170 8-Puzzle Solver. Type '1' to use a default puzzle, or '2' to create your own.

2

Enter your puzzle, using a zero to represent the blank. Please only enter valid

8-puzzles. Enter the puzzle demilimiting the numbers with a space. RET only when finished.

Enter the first row: 1 6 7

Enter the second row: 5 0 3

Enter the third row: 4 8 2

Select algorithm. (1) for Uniform Cost Search, (2) for the Misplaced Tile Heuristic, or (3) the Manhattan Distance Heuristic.

3

The best state to expand with a  $g(n) = 0$  and  $h(n) = 3$  is...

[1, 2, 3]

[0, 4, 6]

[7, 5, 8]

The best state to expand with a  $g(n) = 2$  and  $h(n) = 12$  is...

[1, 2, 3]

[4, 5, 6]

[0, 7, 8]

:::: // Here I deleted about 17 pages of the trace to save space

The best state to expand with a  $g(n) = 4$  and  $h(n) = 3$  is...

[1, 2, 3]

[4, 6, 0]

[7, 5, 8]

The best state to expand with a  $g(n) = 34$  and  $h(n) = 0$  is...

[1, 2, 3]

[4, 0, 6]

[7, 5, 8]

Solution depth was 34

Number of nodes expanded: 45553

Max queue size: 534334

<EK says, The numbers above are just random numbers I made up>

URL to my code is <http://Github.ucr.SueMEE/cs205>

### Special note for students that have taken CS170 at UCR

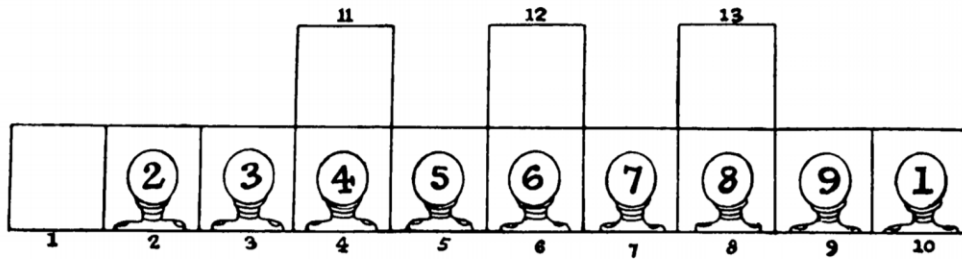
Since you have already solved the 8-puzzle, it does not make sense for me to ask you to solve it again. So, I am going to ask you to solve a different, but similar problem. Below I list some problems you can solve. The only one I have solved was *Nine Men in a Trench*, and I found it easy to adapt my 8-puzzle code to do this. The others seem like they should be doable, but I have never actually tried them ;-)

Please solve *one* of the following problems

- **Nine Men in a Trench**, this is most similar to the 8-puzzle. I \*think\* a good heuristic is simply the Manhattan distance between square '1' and the sergeant.
- **RAILWAY SHUNTING** [a] this is similar to the 8-puzzle.
- **ADJUSTING THE COUNTERS** [a], I have never solved this myself, but think it should be doable.
- **THE ANGELICA PUZZLE** [a] this is similar to the 8-puzzle.

**Nine Men in a Trench** is a math puzzle invented in 1917 by H. E. Dudeney.

Here are nine men in a trench. Number 1 is the sergeant, who wishes to place himself at the other end of the line—at point 1—all the other men returning to their proper places as at present. There is no room to pass in the trench, and for a man to attempt to climb over another would be a dangerous exposure. But it is not difficult with those three recesses, each of which will hold a man.



How is it to be done with the fewest possible moves? A man may go any distance that is possible in a move.

[a] [http://jnsilva.ludicum.org/HMR13\\_14/536.pdf](http://jnsilva.ludicum.org/HMR13_14/536.pdf)