Assignment 1

Skip to Part 1 if you don't want to read AI generated text.

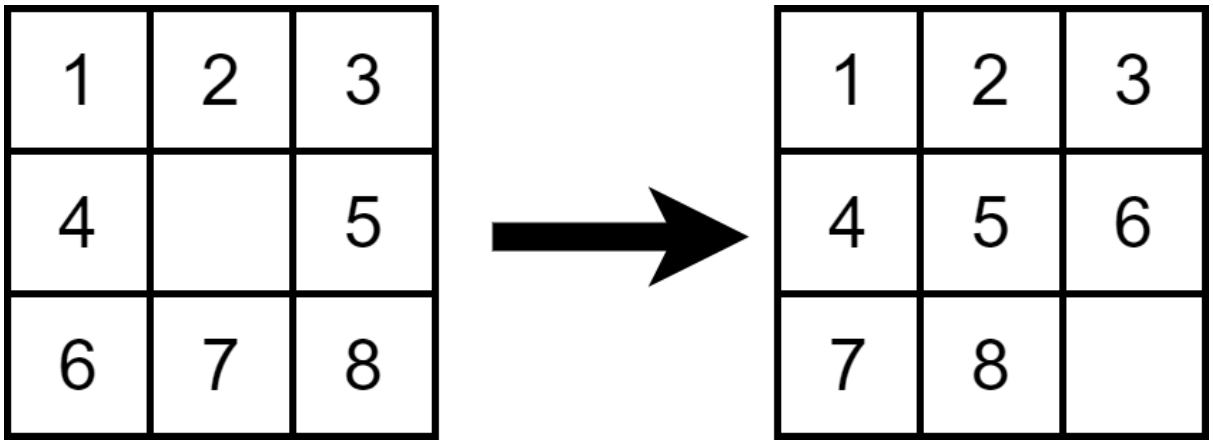
The 8-puzzle, also known as the sliding puzzle, is a classical problem in artificial intelligence, computer science, and recreational mathematics. It offers an excellent platform to explore various problem-solving techniques, search algorithms, and heuristics, making it an ideal subject for this project.

The 8-puzzle has historical importance dating back to the 19th century. It was created by Noyes Palmer Chapman, a postmaster in Canastota, New York, who patented the puzzle in 1879. The original version, known as the "15-puzzle," featured a 4x4 grid with 15 numbered tiles and one empty space.

In the 8-puzzle, a 3x3 grid that contains eight numbered tiles (numbered from 1 to 8) and one blank space are used. The goal is to transform the initial configuration of the puzzle into a specified goal configuration through a series of moves while adhering to specific constraints. These constraints include:

1. You can move a tile into the adjacent empty space either horizontally or vertically, effectively "sliding" the tile.
2. The puzzle must stay within the confines of the 3x3 grid, meaning there are limits on where each tile can move.
3. The objective is to reach a goal configuration by finding the minimum number of moves.

Part 1



Solve the 8-puzzle problem using

1. Breadth First Search
2. Depth First Search
3. Uniform Cost Search
4. Greedy Search
5. A* Seach

For each algorithm, find

1. Number of expanded nodes
2. Path
3. Path-cost

There are 4 actions: up, right, down, left. Use them in this order (search the tree in this order). Otherwise, although your code can find a solution, it won't be the solution we expect, and you will get **no points**. While grading, we will only check whether the results in the output files your code produce are the same as the ones we want. The empty tile will be represented with 0 in the inputs.

Your script should take the initial system state from a file, and write the number of expanded nodes, path, and path-cost to another file. Write each value to a different line. We provided some example input files. Your code should take the name of the input and output files as arguments. We will run your code using the following:

- python3 part1.py <input-filename> <output-filename>

Output Format:

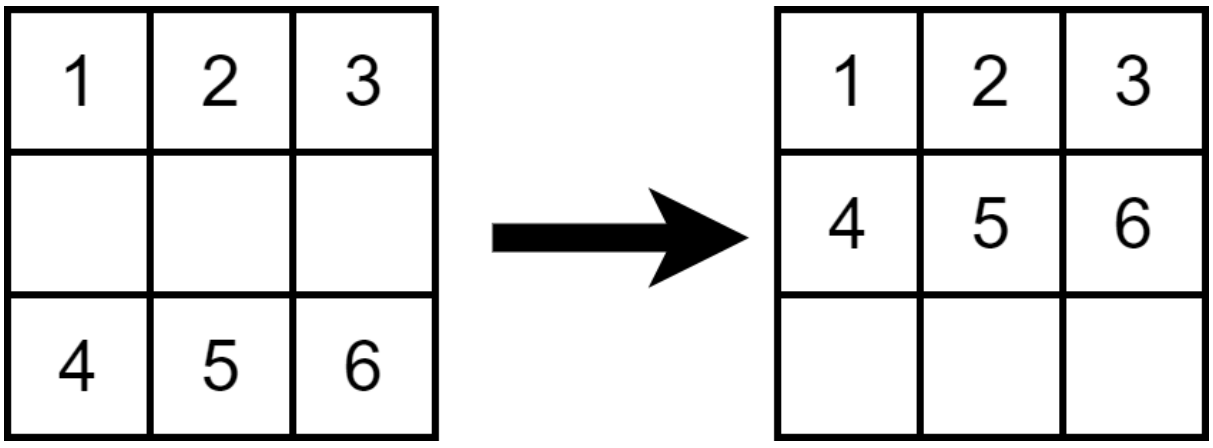
- Number of expanded nodes: Integer
- Path-cost: Integer
- Path: sequence of actions taken by the blank tile. Each action is separated by a single space and shown with the first letter of the actions:
    - Example path: U L R U D R D → the blank tile goes up, left, right, up, down, right, down respectively.

Results of all algorithms should be written to the output file in the order given above. So, your code should generate an output file with 15 lines.

Important Remarks:

- For the A* search, use Total Manhattan Distance as heuristic.
- For Greedy and Uniform Cost Search, if the action and the path-cost are the same, execute the nodes in the order they are expanded (the order they went into the queue).
- Your system is expected to detect repeated states and make plans accordingly. Check Graph Search in the slides (page 128 in 01-02-agents-seach.pdf)
- Hint: Please keep in mind that the puzzle is not always solvable. The examples we provide are solvable, but if you want to create your own test cases, make sure they are solvable.
- For greedy search, sort them by their path cost first, then their actions (U, R, D or L), and then finally by the order in which they are added to the fringe.
- For BFS, add the child nodes to the fringe in the given order. Fringe is FIFO.

Part 2



In this part, there are 3 empty tiles. Your objective is to find a good heuristic for the A* search**. Explain your heuristic in a pdf file (couple lines are enough)**. Report Number of expanded nodes, path, and path-cost. The best heuristic (the ones with the lowest path-cost) will get the most points. If there are multiple equally good solutions, whoever submitted earlier will get bonus points. Write the number of expanded nodes and path-cost to the output file as you did in part1. For the path, print each state after each action separated by an empty line. Path won't be auto graded, so just make sure it is readable.

- python3 part2.py <input-filename> <output-filename>

NOTES:

- If you have any questions, you can ask using the forum on Moodle. If you think there is anything ambiguous in the description (that can affect the output), please ask. However, before asking, please read the description again. Obvious questions will not be answered.
- In case of an objection, readability of your code is important. If we can't understand your code, we can't accept your objection. Although not mandatory, documenting your code with inline comments is recommended.

Cheating Policy

Don't cheat. Don't use ChatGPT generated code.

Any sign of cheating will be penalized, and you will get -50 points for the project, and you will get F grade in case of recurrence in any other project.

Submission Details

The zip file you submit should include 3 files: part1.py , part2.py and heuristic.pdf . There should not be another folder in the zip file, only these files. You will zip them up and submit on Moodle as a single .zip file. The name of the zip file should be Cmpe480_Assignment1_<studentid>.zip. **No other type of submission will be accepted.**