CS451/551 – Introduction to Artificial Intelligence
Assignment-1

Solving N-Puzzle Problem with Search Algorithms
Due: 30 March 2021 @ 23.55

created by Furkan Kınlı, for help → e-mail: furkan.kinli@{ozu, ozyegin}.edu.tr

1. Definition

In this assignment, you will implement different search algorithms to solve N-Puzzle Problem where N is 8. We will provide some baseline code for N-Puzzle environment (on LMS), and you need to incrementally implement well-known search algorithms. Mainly, we expect you to implement breadth-first search (BFS), depth-first search (DFS), uniform cost search (UCS) and A* search (A*) algorithms for this assignment.

In code baseline, you will have 7 different script files as follows:

- main.py → Runner script, you are NOT allowed to change this file.
- Node.py → Class file for Node implementation, we strongly encourage you to check over this file to understand what a node can do. NOT allowed to change.
- Graph.py → Class file for Graph implementation, we strongly encourage you to check over this file to understand what the graph can do. NOT allowed to change, and you do not want to change it ☺
- BFS.py → Class file for breadth-first search algorithm. In this file, you need to implement "run" method to design your algorithm for BFS.
- DFS.py → Class file for depth-first search algorithm. In this file, you need to implement "run" method to design your algorithm for DFS.
- UCS.py → Class file for uniform cost search algorithm. In this file, you need to implement "run" method to design your algorithm for UCS.
- AStar.py → Class file for A* search algorithm. In this file, you need to implement "run" method to design your algorithm for A* search.

2. N-Puzzle

N-Puzzle is a popular puzzle problem consisting of N tiles where N can be 8, 15, 24 and so on (M_2-1). In this assignment, N = 8. The puzzle is divided into square root of N+1 rows and columns (*e.g.* 15-Puzzle will have 4 rows and 4 columns, and an 8-Puzzle will have 3 rows and 3 columns). The puzzle consists of N tiles and one empty space where the tiles can be moved. **Start** and **Goal** configurations (also called **state**) of the puzzle are provided. The puzzle can be solved by moving the tiles one by one in the single empty space and thus achieving the Goal configuration. For more information about the problem, you may check here.

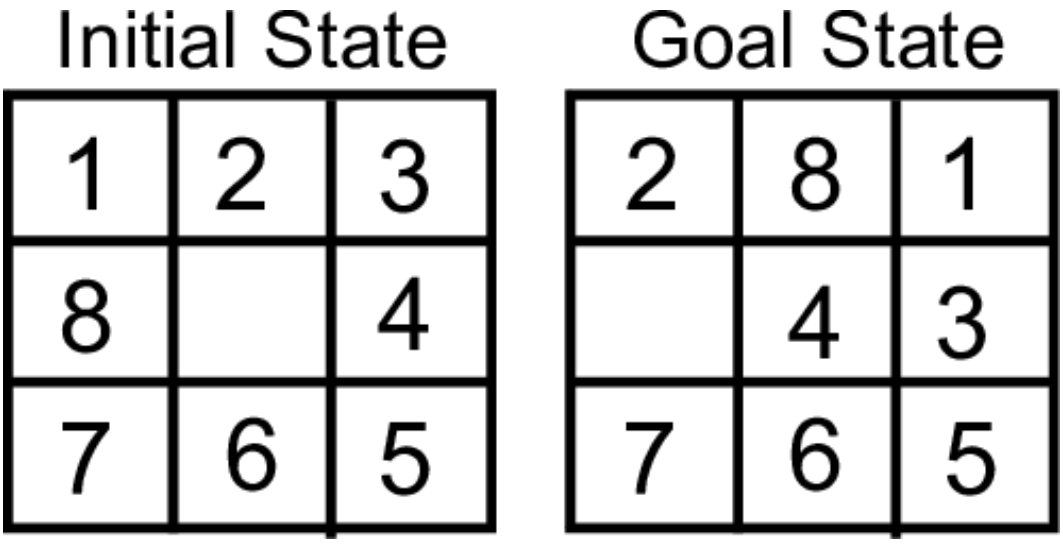


Figure 1. Example states for N-Puzzle problem where N=8.

3. Implementation Details

As we provided a code baseline, you have to be strict on this baseline. You are not allowed to propose any other implementation for this assignment. Node and Graph implementations are given to you to deeply understand what they actually do, and you are not allowed to change these files. Also, you will have a runner script which is named as "main.py", and this file is basically responsible for running your script and making possible to select the algorithm that the user wants to apply to the problem. After entering the grid size as 8, you need to enter your initial and goal states, and then you need to select the algorithm type. For your services, the grid size and particular states are kept as parameter if you may want to change them, and investigate the different levels of this problem with your designed algorithms.

You need to implement your algorithms on "run" method of corresponding class files (*e.g.* BFS.py, DFS.py, UCS.py and AStar.py), and report your results accordingly. Please first try to understand the main structure of the environment, and then start to implement the algorithms. Please do NOT change any code block unless it has the comment line as

"YOUR CODE HERE"

If you are not comfortable to write code in Python, you may follow this tutorial to improve your Python programming skills. Even if you do not know anything about Python, you only need 2-3 days to learn Python from scratch to be able to complete this assignment. If you have any question about the implementation, please do not hesitate to send an e-mail to the assistant.

You are allowed to implement your algorithms on Java, but you need to create such an environment on your own, and make sure that all properties aforementioned before should work properly.

4. Criterion

Runtime: Please do NOT include any third-party library to your project, because you do not need them. Including different third-party library may cause a problem for running your code in different local machines (*e.g.,* dependencies etc.). For any dependency on runtime that the assistant can solve→ 20 points penalty, for any dependency on runtime that the assistant cannot solve →100 points penalty.

Compilation: Please make sure that your code is compiled properly. Not compiling code (*e.g.,* any error etc.)→ 100 points penalty.

Report: You need to write a report to explain your design in detail. 1 paragraph code explanation is not a report, and the submission with such reports will be IGNORED. Please follow the rules of technical report writing (*e.g.,* Introduction, Algorithms, Implementation Details, Results, Conclusion etc.).

Submission: All project files (*.PY) and report file (.PDF) should be zipped (.ZIP) together, and the filename of ZIP file should be in the format of "NAME_SURNAME_ID_hw1.zip". Please follow this structure on your submission. **Not following→ 10 points penalty.**

Due date: 31 March 2021 – 23.55 via LMS.

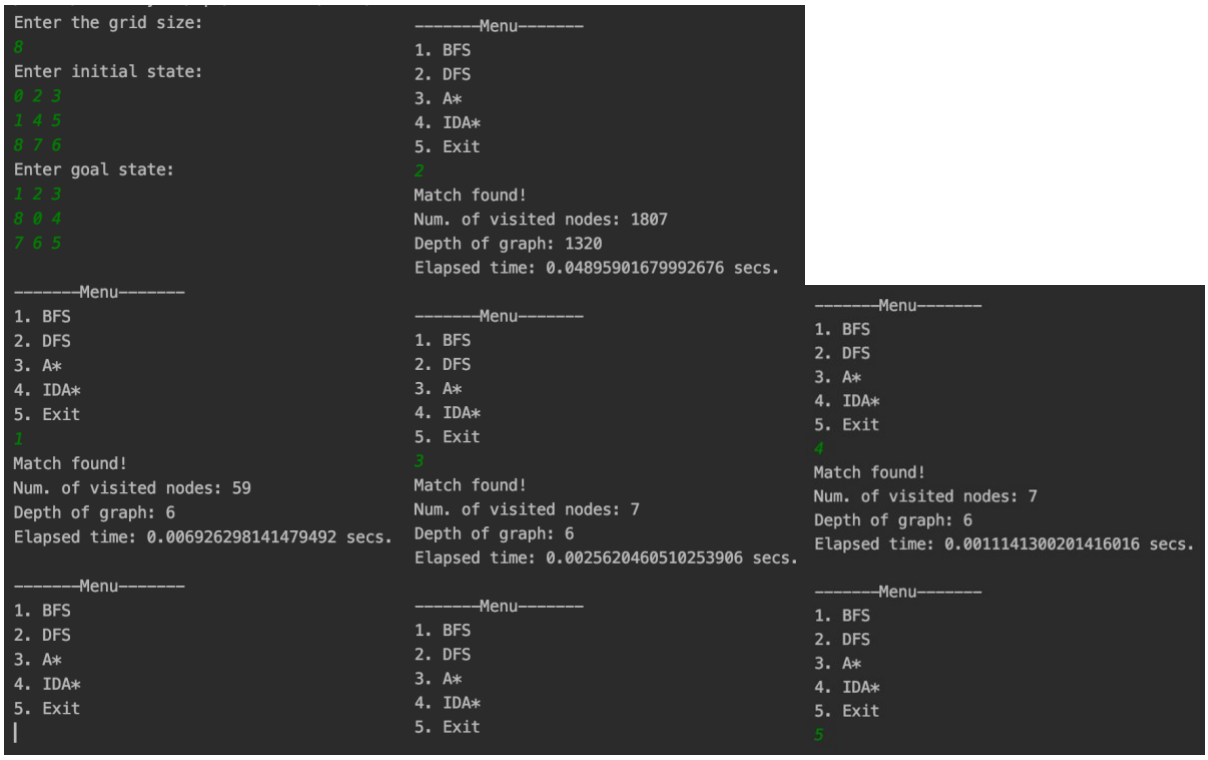
Late submission: Allowed for 3 extra days, with -10 for 1st day, -15 for 2nd day, -20 for 3rd day.

Group effort: You may discuss the algorithms and so forth with your friends, but this is and individual work, so you have to submit your original work. **In any circumstances of plagiarism, first, you will fail the course, and the necessary actions will be taken immediately.**

Grading criteria:

- BFS: 20 points
- DFS: 20 points
- UCS: 20 points
- A*: 25 points
- Report: 15 points

EXAMPLE SCENARIO:



However, you may want to check the other possible N-puzzle cases from online resources, we will check your algorithms with different test cases. Runtime may depend on the local machine that you run the code, but you must consider the runtime of each algorithm (there are some facts some of them works better than the others).

Note: The output of 3rd option: 365 visited nodes with depth of 6.