Assignment-1 Problem solving agents

Your objective is to implement SIMPLE-PROBLEM-SOLVING-AGENT(s) based on the pseudocode in **Figure 3.1 of AIMA 3rd edition book chapter 3 page 67**. Please read and understand the pseudocode before proceeding further. If you haven't seen the pseudocode, most of the things below won't make any sense.

There are two problems in this assignment. For implementing the solutions, please do the following:

1. Hardcode the goal (goal-test), instead of automatically formulating it based on a performance measure. The best way to hardcode the goal is as follows:
    a. Design a function (python) to take as input a node and test if the state represented by the node passes the goal conditions.
    b. Pass this python function as an argument to the SEARCH method along with other arguments such as the initial-sate, precondition-action-effect and path-cost.
2. The SEARCH method can take in an additional argument "search_type" which could take values from the set { BFS, DFS, GBFS, ASTAR, etc}. This argument determines which algorithm will be used for the search.
3. Just printing the action-sequence as output is sufficient. You don't have to iterate through the sequence one at a time, as shown in the pseudocode. But, make sure to print the action sequence in way that is easier to understand and evaluate.

Careful planning will allow you to implement a general solution and reduce the total amount of code you have to write. We will measure how much of the code has been reused/shared between the two problems and incentivize accordingly.

Problem-1: Path finding

I have assigned a data collection competition. The top three solutions of the competition will be shared with the entire class. Please use that to create your graph. **If, unfortunately, no one submits a valid solution by Sep 10th, then assume the map of Romania in the textbook as your graph.** For this problem, you need to build a basic taxi-agent. The search algorithm(s) should find path between two cities (provided as input). Implement the following search algorithms: BFS, DFS, Greedy Best First search and A*

Problem-2: Eight Puzzle

N-Puzzle or sliding puzzle is a popular puzzle that consists of N tiles where N can be 8, 15, 24 and so on. In our example **N = 8 (Hence called Eight Puzzle)**. The Eight Puzzle problem will have 3 rows and 3 columns. The puzzle consists of 8 tiles and one empty space where the tiles can be moved **(see the book for visualization and more details)**. Start and Goal configurations (also called state) of the puzzle will be provided. The puzzle can be solved by moving the tiles one by one in the single empty space and thus achieving the Goal configuration. You have to implement an agent that can take in initial and goal configurations. The search algorithm(s) should find the sequence of actions that can achieve the goal. Implement BFS, DFS, Greedy Best First Search and A*. Come up with your **own heuristic function** to solve the problem.

I will provide sample test cases for the problems at a later date