CS480 – PROGRAMMING EXERCISE 1 – CHAPTER 3

In this programming exercise, you'll get experience with some of the algorithms discussed in chapter 3. You are provided three files:

- search.py contains the search algorithms and necessary data structures.
- problem.py contains the definition of a few problems, such as N-Queens, 8-Puzzle, and Travel.
- test-search.py has a few examples on how to run these search algorithms.

1. Complete the implementation of breadth-first-search on line 225 of search.py. This requires only a single line of code.
    a. Hint: It's one of the following lines:
        i. return search_type(problem, FIFOQueue())
        ii. return search_type(problem, LIFOQueue())
2. Complete the implementation of depth-first-search on line 228 of search.py. This requires only a single line of code.
    a. Hint: See the hint for question 1.
3. Complete the implementation of astar-search on line 282 of search.py. This requires only a single line of code.
    a. Hint: see the implementations of uniform_cost_search and greedy_best_first_search.
4. Replicate Homework 1 solutions. Print the frontier and the node expanded at each step of the search. Are the DFS results different for Homework 1 question 1? Why?
5. Create the map of Romania (slide 15 of chapter 3) and try various algorithms, like Breadth-first search, uniform cost search, greedy-best-first search, A* search using tree search and graph search for traveling from Arad to Bucharest. For h(n) use slide 30 of chapter 3. Print the frontier and the node expanded at each step. Compare your results with ch3-nodes.pdf file on blackboard.
    a. Hint: see the travel problem in test-search.py.
6. Implement the manhattan-distance heuristic for the eight puzzle problem.
    a. Hint: see the misplaced_tiles_heuristic implementation on line 96 of problem.py.
7. Compare the number of misplaced tiles heuristic and manhattan distance heuristic using A* search for 8-puzzle problems of various difficulty. Count how many nodes are generated by A* when using these heuristics.
    a. Extra: replicate the A*(h1) and A*(h2) results presented on slide 43 of chapter 3. For this exercise, you need to write a function that can return a random 8-puzzle problem.
8. Generalize the 8-puzzle problem to be the N-puzzle problem where (N+1) is a square of an integer.
9. Implement a problem of your interest which can be solved using a search algorithm described in class. For example, you can implement a simple version of the Sokoban puzzle, the Rubik's cube, or any smartphone game where you can clearly specify the current state, the goal state, the allowed actions at each state, and an admissible heuristic. Remember that for chapter 3, the environment needs to be single-agent, deterministic, discrete, known, and observable.