SUBMISSION DEADLINE: 03.11.2023 @18.00

GIVEN

The provided folder structure:

```
quiz2
├── quiz2.pdf
└── code
    ├── route_search.py
    ├── search.py
    └── problems.py
```

TO DO

This is a two stage quiz.

1. Complete the search algorithms in "search.py" inside the "quiz2/code" folder to accomplish the tasks defined below. "search.py" is the only script that you need to complete!
2. Take the test in ItsLearning (**will be active from 01.11.2023 @08.00 to 03.11.2023 @18.00**)

Make sure that you are done with the coding before you start the test. The codes will help you to answer the questions in the test. If you directly jump into the test, you will NOT have enough time to complete the required functionality within the code!

1 Search Algorithms Analysis

This study analyzes performances of search algorithms in different routing problems over the map in Figure 1.

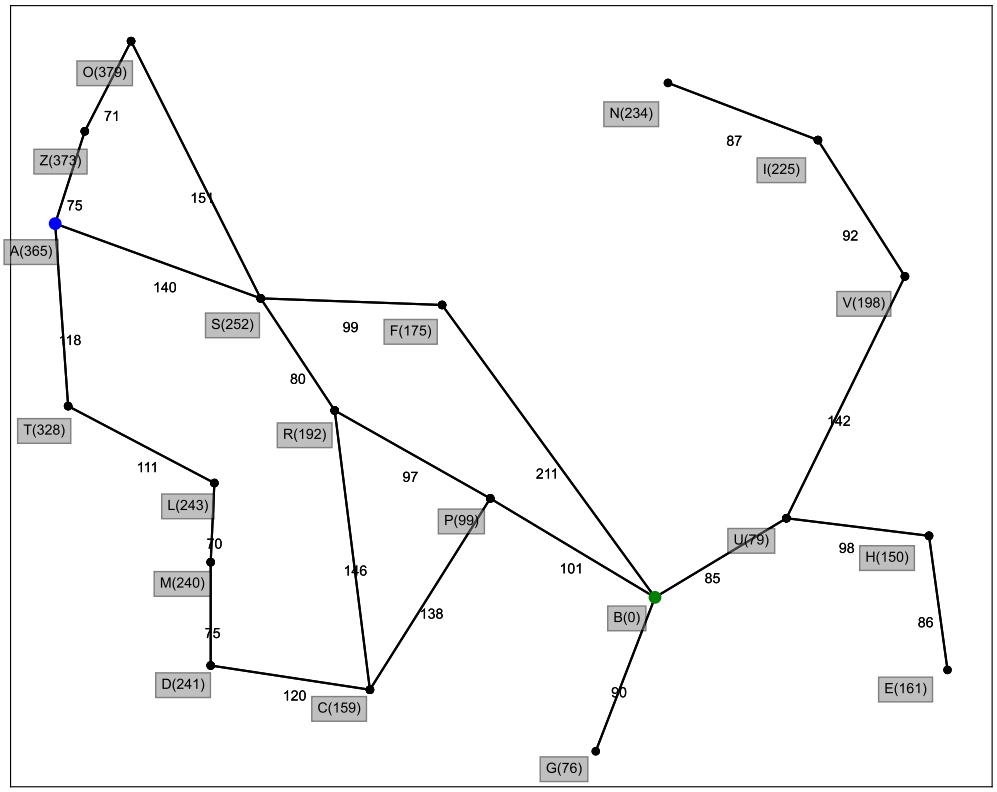

Figure 1: **Romania map.** The graph shows the map of Romania. Each vertex and edge correspond to a city and a road between two cities, respectively. Each vertex is labeled with the initial of the city (e.g., A: Arad, B: Bucharest, etc.) and each edge is tagged with the distance (cost of the road) between two cities. Initial and goal states are colored with blue and green, respectively. A value inside parentheses next to a vertex name (e.g., A(365), B(0), etc.) shows the heuristic value, i.e., estimated cost of the remaining path to the goal. We will use airline-distance of a city to the goal as our heuristic (h) in this exercise.

1.1 Memory and Computation Time Analysis

This section analyzes search algorithms in terms of the number of nodes created, the number of nodes extended, and how many times is-goal function is called. Besides, we will write down the solution path, the number of actions in the solution, and the cost of the solution.

We are given all necessary code to run required experiments inside 'quiz2/code' folder. For example, to collect required statistics for the problem of finding a solution from initial state 'A' to goal state 'B' using 'depth first graph search' algorithm, we will run the following code in the terminal:

"python route_search.py --initial_state A --goal_state B --search_algorithm depth_first_graph_search"

1.1.1 Sanity check: initial state 'A' and goal state 'B'

Once you complete search algorithms, you should check if your search algorithms work as expected by comparing your results with the results in Table 1.

Table 1: **Analysis of search algorithms on problem-1: initial state 'N' and goal state 'T'.**

search algorithm	# created	# extended	# is-goal	solution	# actions	cost
depth first graph search	9	3	4	['A', 'S', 'F', 'B']	3	450
breadth first graph search	30	12	13	['A', 'S', 'F', 'B']	3	450
astar	15	5	6	['A', 'S', 'R', 'P', 'B']	4	418