Tut-Lab Week 7

Aims:

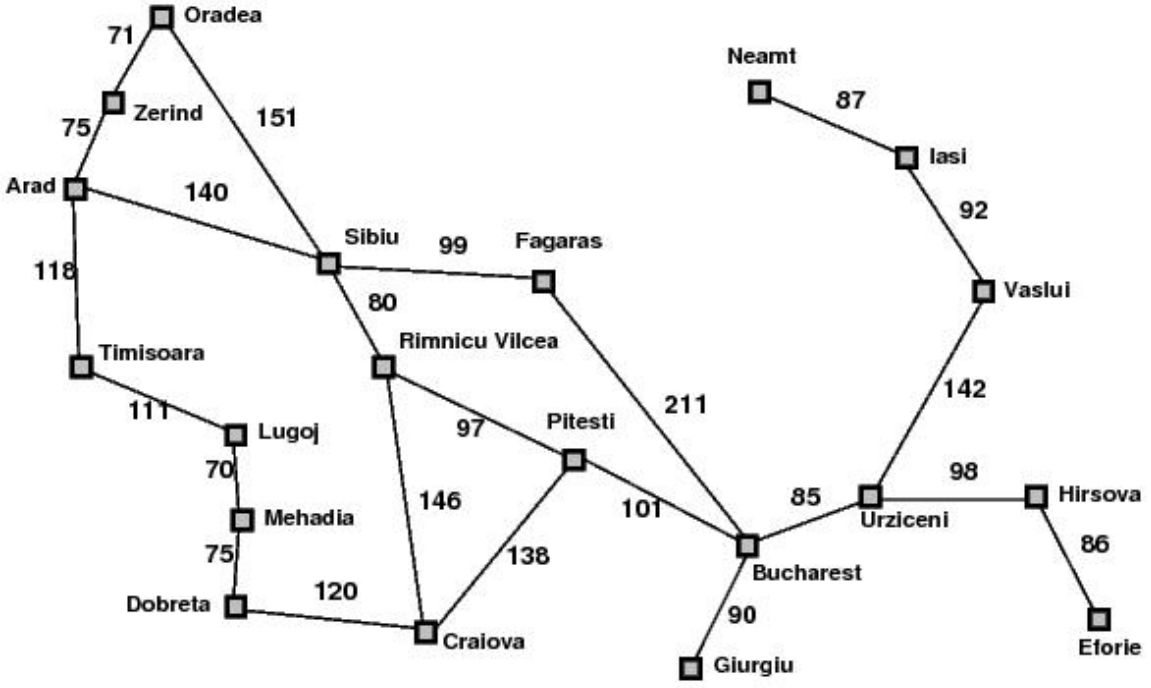
- Understand the behaviour of heuristic search algorithms
- Gain practical experience implementing a search algorithm

Preparation:

- Review the material on A* search
- Review the `Strategy` pattern

Problem-Solving Algorithms

- Consider again the route-finding problem using the Romania map



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

- What order are nodes in the state space expanded for each of the following algorithms when searching for a (shortest) path between Arad and Bucharest (when there is a choice of nodes, take the one earliest in the alphabetical ordering)?
    - A* search with straight-line distance heuristic (inefficient, guaranteed optimal solution)
- Modify your implementation of breadth-first search to implement A* search for route-finding in the Romania map
    - Use a class that implements the **Graph<E>** interface type to store the map
    - Use a **PriorityQueue<E>** to store the nodes generated in the search
    - Store the successors of a node in an **ArrayList** and make sure they are added into the **PriorityQueue** in order of *f* value

- Refactor your design
    - Apply the `Strategy` pattern so that the heuristic used by A* search is passed as a parameter to an appropriate method – are there any tradeoffs associated with the choice of class or method?
    - Draw a UML class diagram of your program, making sure your design and code conforms to the `Strategy` pattern