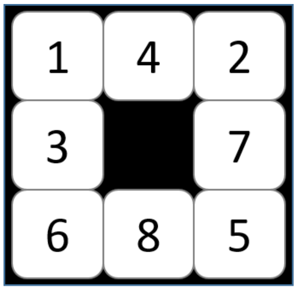
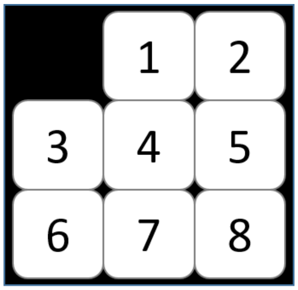
Target Topic: A* Search, Iterative Deepening A* (IDA*) Search

Case Study A : 8-PUZZLE





INPUT: A permutation of {0, .. , 8} arranged in 3x3 format. 0 represents the hole.

OUTPUT: The sequence of moves required to reach the goal state, with the minimum number of moves

Provided materials:

- 8puzzle_idastar_template.py : the incomplete program for IDA*.
    - 8puzzle_idastar_template_EZer.py is the easier version, for students struggling with programming
- 8-puzzle testcases.zip from Week 1

Task:

1. Study to make sure that you understand A*.

2. A* solution for 8-puzzle is a very easy adaptation of the GBFS program developed last week. Modify the GBFS from the last week so that it solves 8-puzzle with A* technique.

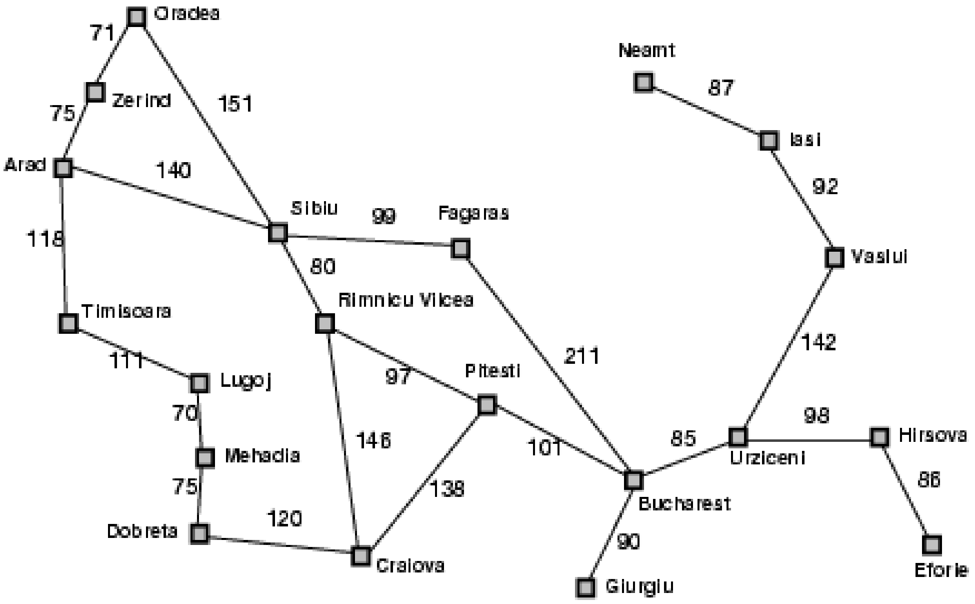
3. Study to make sure that you understand how IDA* works.

4. Study the provided 8puzzle_idastar_template.py program for what are required to complete the program.
    - As IDA* utilizes a variance of Depth-first Search (DFS), a function for DFS is required for implementing IDS. However, the EZer version includes a complete modified DFS function.

5. Complete the program so that it solves the 8-puzzle problem with IDA* technique.
    - The “print_path” function should be the developed from previous weeks

6. Test the program with the provided test cases. Note the limitation of the program.

CASE STUDY B : ROMANIA MAP



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

1. Study the map data file provided. The distance is obtained from an online site and will be different from those in the figure above.

2. Read the input data to store the map as a collection of adjacency lists. This will be the most efficient graph representation to support the search algorithm.

3. First, attempt a Uniform Cost Search (UCS) solution. This will be a simple adaptation of Greedy Best-first Search program, by changing the key value (for priority queue handling) from h to g.

4. Once the UCS works correctly, add the use of straight-line distance to Bucharest as heuristic value in order to transform the search algorithm to A*.