1. Install `simple/stile/15puzzle.py` and execute this command (last lines of output shown):

```bash
python3 15puzzle.py -s1 -p 15 14 13 12 11 10 9 8 7 6 5 4 3 1 2
... nodes searched 6
Moves: 102
```

i) For this execution, give the number of stages, the goal for each stage, the number of tile moves in the solution, and the total number of nodes searched over all stages. E.g. the goal for stage 1 is for tiles \{1, 2\} to be in their correct final locations. If you want to see all moves in the final solution, uncomment lines 109 and 110.

ii) Repeat the previous question for this command:

```bash
python3 15puzzle.py -s2 -p 15 14 13 12 11 10 9 8 7 6 5 4 3 1 2
```

iii) Repeat the previous question for this command:

```bash
python3 15puzzle.py -s3 -p 15 14 13 12 11 10 9 8 7 6 5 4 3 1 2
```

iv) What happens if you execute any of these commands with the permutation in reverse order, i.e. 15 . . . 1? Why?

v) For this problem, when the number of stages increases — e.g. from option `-s3` to `-s2` to `-s1` — the runtime decreases, the number of nodes searched decreases, and the number of moves in the returned solution increases. Why? Explain briefly.

2. For the map of cities in the webnotes, [https://webdocs.cs.ualberta.ca/~hayward/355/jem/tile.html](https://webdocs.cs.ualberta.ca/~hayward/355/jem/tile.html), detours have caused these changes: distance Sibiu – Fagaras now 199 (was 99), distance Rimnicu V – Pitesti now 297 (was 97). Trace Dijkstra’s algorithm: find a shortest path from Arad to Bucharest.

3. For the A* example from the webnotes (previous question), perform the next two steps of A*.

4. For each of these sliding tile positions, give i) the minimum number of moves to solve it (use `stile_search.py`), ii) its misplaced tiles score iii) its Manhattan heuristic score.

```
1 2 4 5 4 3 3 5 6 . 6 1 8 2 3
5 3 . 6 1 8 1 4 . 2 4 3 4 5 6
. 7 2 7 8 2 7 5 8 1 . 7
```

5. Find a sequence of sliding tile moves that transforms one position below to the other. Do you think your solution is shortest? Explain briefly.

```
4 * * 1 * *
1 * 4 *
```