Water Jug Problem

To solve this problem i used BFS and DFS from the provided code skeleton. A state is defined by 2 values, representing the quantity of water in the 2 jugs. Initial state is (0, 0) and final state is (2, 0)(in the beginning, both jugs are empty, and in the end the problem asks that the value of the second jug to be 2 and the second 0). We define the class for this problem in P1.py.

The actions defined in the class are : filling a jug, emptying a jug, empty first jug in second, empty second jug in first. In main.py we see the used actions in order. In main is also measured the time of both BFS and DFS.

Fifteen Puzzle Problem

For this problem I used a\* search from the provided code skeleton with 4 heuristics(2 of them were discussed at the laboratory). A state is defined by 16 values, representing the numbers in all the board positions.(Since we have a 4X4 board, we have 16 numbers. The initial state is a random ordering of the numbers from 0 to 15, and the ends state is the ascending order of the numbers from 1 to 15, with 0 in the end.

There are 4 defined actions, represented by the movement of the empty square(the 0 value) in each directions(UP, DOWN, LEFT, RIGHT), although some of them might not be available.(if you are in the first row of the board, you cannot could UP, because the square would “fly” off the board.

As mentioned earlier, I implemented 4 heuristic(2 of them are discussed at the lab) : misplaced tiles heuristic – counts how many numbers are not on the right tile, manhattan distance heuristic– the manhattan distance from the current tile of a number, to the goal tile of the same number, Euclidian distance heuristic – the euclidian distance from the current tile of a number, to the goal tile of the same number, misplaced rows&column heuristic – number of numbers that are on tiles that are not on the same row as the tile on the goal state of the same number + number of numbers that are on tiles that are not on the same column as the tile on the goal state of the same number.

At the end of the main function I also called the compare\_searchers to compare the 4 heuristics.