Demonstration Task 3 ($A^\star$-Algorithmus)

1 Objective and problem description

The aim of this task is to write a program that can solve the well-known 8-slide puzzle using the $A^\star$-algorithm.
Most of you are probably already familiar with the 8-slide puzzle, since it is a popular as a children's game. It consists of a frame in which 8 puzzle pieces can be moved. It is possible to move one of the pieces to the free space (shown in white in the picture), as long as the piece is horizontally or vertically adjacent to this free square. (i.e. the white square swaps with one of the maximum four horizontally or vertically adjacent squares). The picture below shows a typical problem. You find the puzzle in the configuration shown on the left. Moves to achieve the goal shown on the right.

Start state

2		3
1	6	4
8	7	5

game moves $\longrightarrow$

Goal state

1	2	3
8		4
7	6	5

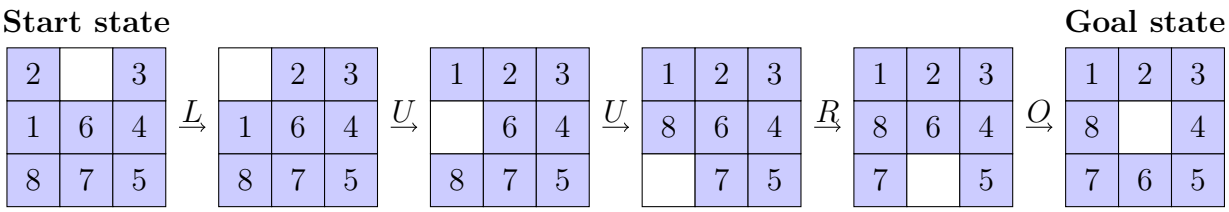
You can consider how you would proceed here.

2 Description of the problem as a graph

The problem can be modelled as a simple search in a graph.

- Each node of the graph consists of a two-dimensional array, where the individual positions are assigned the digits 0 to 8. The 0 corresponds to the empty field.
- A neighbouring node is the field that can be created by swapping the 0 with any horizontally or vertically adjacent field.
- Moves can easily be "noted" unambiguously. You simply specify in the notation for a move, in which direction the white square (i.e. the 0) is moved. L for left, R for right, U for down, and O for up.
- The problem shown above can thus be solved by the move sequence $L \rightarrow U \rightarrow U \rightarrow R \rightarrow O$.

Solution path in graphical representation:



3 Heuristic for the $A^\star$ algorithm

You know the $A^\star$ algorithm from the lecture. You need a value for $g(x)$. This is simply the number of steps you have made so far. Then you need a heuristic estimation function $h^\star(x)$ that estimates how many moves are left from position x to the target.
For this you can take the sum of the so-called Manhattan distances of each stone from its target position. The manhattan distance of a piece is calculated the sum of the horizontal and vertical differences to the target position. target position.
You can see below an example of the calculation of $h^\star(x)$ with Manhattan distances for a given example configuration:

6	2	3
8	4	
7	5	1

Here is the calculation of the distance of each puzzle piece to the target position according to the Manhattan distance.
$h^\star(x) = 9$ for the pattern shown.

Puzzle Piece	Distance
1	4
2	0
3	0
4	1
5	1
6	3
7	0
8	0
Sum:	9

4 Help for programming

The best thing to do is to divide your programme into corresponding sub-problems and implement suitable functions/procedures for them, depending on the programming language you use. For the $A^\star$-algorithm you need the two lists $OPEN$ and $CLOSED$, to keep track of the nodes of the graph to be expanded.
As partial functionalities would be e.g.:

- Definition of the data structure for a node of the graph (e.g. 2-dimensional field).
- Implementation of a function that computes all "'neighbours"' for a given node (jigsaw puzzle) i.e. how this node is expanded. These are all allowed actions.
- Implementation of a function to calculate the Manhattan distance to the target for a given puzzle position.
- A custom function for implementing the $A^\star$ algorithm. It is also important to consider how the successful path is determined/carried along in the implementation so that it can be output later.
- The output of the path after the calculation of a solution is obligatory in order to be able to understand the to be able to reconstruct the calculated solution.
- You must also provide suitable functionalities for the input of the start position, as well as the output of the solution.

Temporary page!

LaTeX was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it.
If you rerun the document (without altering it) this surplus page will go away, because LaTeX now knows how many pages to expect for this document.