Nikoloz Muladze

1. How have you implemented the method for returning possible tile moves with given the empty space location?

I checked whether an empty space location could move UP, DOWN, RIGHT or LEFT

As it is completely random, depending on where the empty space location is.

1. What heuristic have you implemented for A\* search?

I used A\* Manhattan distance heuristic and Linear Conflict Heuristic

What is A\* Manhattan distance heuristic?

It is a method for calculating h(n), which ignores diagonal movement and any obstacles that might be in the way as it is computed by calculating the total number of squares moved horizontally and vertically to reach the target.

Formula: *h*=∣*xstart*​−*xdestination*​∣+∣*ystart*​−*ydestination*​∣

What is Linear conflict?

When 2 tiles are in a linear conflict if they are in the same row or column, and

their goal positions are in the same row or column and the goal position of one

of the tiles is blocked by the other tile in that row.

It is always combined with Manhattan distance to get the heuristic value.

Whenever linear conflict occurs it adds 2 moves to Manhattan distance so that heuristic value h(n) = Manhattan distance + 2 \* number of linear conflicts.

1. By using uniform cost search, how many moves are needed to solve the 8-puzzle problem and how long did it take to find the solution on the computer you used?

I was not able to implement it

1. By using A\* search, how many moves are needed to solve the 8-puzzle problem and how long did it take to find the solution on the computer you used?

It took me 30 moves and it compiled in about 0.2 second