Ece Alptekin
24156
Assignment 1: Solving Bloxorz using Search

States

The states are different coordinates of the block. If the block is on the “S” tile, it is on the start position. If the block is on the “0” tile, it is where the block is located. If the block is on the “G” tile, it reaches the goal. If the block is on the “X” tile, the block is out of the board, which means that the game is not completed. The coordinate of the block is stored in a list in my node definition. If the block covers two tiles, it has a horizontal alignment. If the block covers a tile, it has a vertical alignment.

Successor State Function

It is the all possible changings in the location of the block in a state. The block can be moved to up, down, right or left. The new coordinates of the block should be checked. If the new coordinates are out of the board, the game should be ended. If the new coordinates are possible, then the program calculates the cost of moving from the current state to the new state and add this cost to the priority queue.

Initial State

Initial state is the starting position of the block, which is indicated as the “S” tiles. The block can have a vertical or horizontal position.

Goal Test

Goal test is to check if the current state is equal to the goal state. If the block coordinate is equal to the goal coordinate, then the game is reached to its target.

Step Cost

Uniform Cost Search

The cost function is

$g(n) = g(n-1) + 1$

It takes 1 cost to move the block from one position to another. Each movement of the block is a step and each step costs 1. The total cost is calculated by adding the cost of each step.

A* Search

The heuristic function of the A* Search is

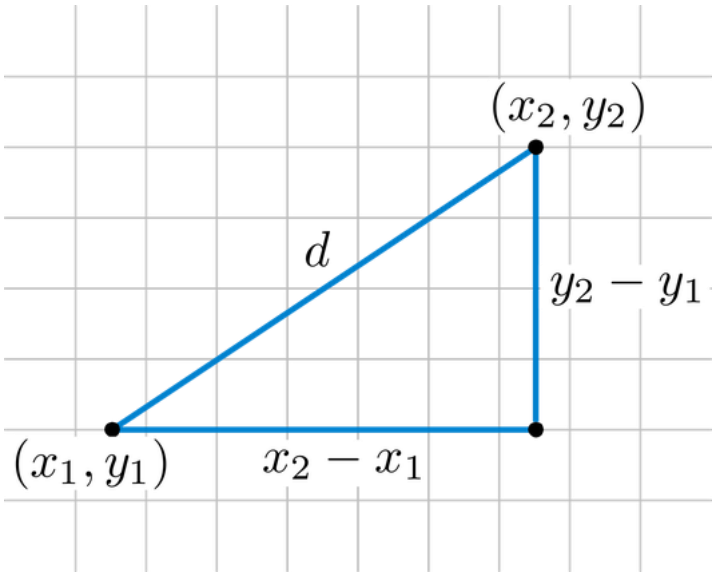
$f(n) = h(n) + g(n)$

h(n) is the heuristic estimate of the cost of getting to a goal node from n. g(n) is the cost of the path to node n. The f-value is an estimate of the cost of getting to the goal via this node (path). As the heuristic function, I compare the time consumption of Euclidian distance and Manhattan distance.

board	Euclidean Distance	Manhattan Distance
board1.txt	0.000667095	0.000526905
board2.txt	0.008721828	0.010604858
board3.txt	0.00099802	0.001024961
board4.txt	0.000648737	0.000567913
board5.txt	0.007189035	0.00770402

Since Euclidean distance as a heuristic function is more efficient than the Manhattan distance, I prefer to use Euclidean distance. According to the Euclidean distance, the distance between two points is calculated with the formula

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$



I calculated the distance between the goal coordinate and a node coordinate.

Admissible Heuristic Function

Let h*(n) be the cost of an optimal path from n to a goal node (∞ if there is no path). Then an admissible heuristic satisfies the condition

$h(n) \leq h^*(n)$

i.e., h never overestimates the true cost.

$h(g) = 0$ for any goal node g

If the goal coordinate is vertically or horizontally far from the coordinate of a node, the Euclidean distance is equal to the true distance. If the goal coordinate is diagonally far from the coordinate of a node, the Euclidean distance is smaller than the true distance. The Euclidean distance can be less than or equal to the true distance according to its formula. This property makes our heuristic function admissible, since it satisfies $h(n) \leq h^*(n)$.

Time & Memory Consumption

UCS

	Time Consumption	Memory Consumption (MiB)
Board1	0.1614	38.2539
Board2	0.3268	38.2773
Board3	0.2001	38.3047
Board4	0.5243	38.2891
Board5	0.3552	38.2617

A*

	Time Consumption	Memory Consumption (MiB)
Board1	0.0249	38.2891
Board2	0.2330	38.3008
Board3	0.0359	38.3281
Board4	0.0283	38.3242
Board5	0.2029	38.3086

The A* Search is much faster than the UCS. The memory consumptions are very close. Therefore, A* Search is more preferable. The heuristic function makes A* Search more efficient. Since, A* Search makes more logical decisions in compare with the UCS. The heuristic function provides decreasing in time consumption.

board1.txt

```
Please enter the filename for the game board: board1.txt
Filename: Checking.py

Line #    Mem usage    Increment  Occurences   Line Contents
=============================================================
   265  38.2539 MiB  38.2539 MiB           1   @profile(precision=4)
   266                                         def UCS(goal, start, empty_tiles, mSize, nSize):
   267
   268  38.2578 MiB   0.0039 MiB           1       frontier = []
   269  38.2578 MiB   0.0000 MiB           1       closed   = []
```

```
UCS
Duration: 0.16141295433044434
[[2, 3], [3, 3]]
[[2, 4], [3, 4]]
[[2, 5], [3, 5]]
[[2, 6], [3, 6]]
[[2, 7], [3, 7]]
[[4, 7]]
Filename: Checking.py

Line #    Mem usage    Increment  Occurences   Line Contents
=============================================================
   211  38.2891 MiB  38.2891 MiB           1   @profile(precision=4)
   212                                         def ASTAR(goal, start, empty_tiles, mSize, nSize):
   213
   214  38.2891 MiB   0.0000 MiB           1       frontier = []
   215  38.2891 MiB   0.0000 MiB           1       closed   = []
   216  38.2891 MiB   0.0000 MiB           1       start_node = Node(start.getCoordinate(), [], 0)
   217  38.2891 MiB   0.0000 MiB           1       frontier   = Queue(frontier, start_node)
   218
```

```
ASTAR
Duration: 0.024902820587158203
[[2, 3], [3, 3]]
[[2, 4], [3, 4]]
[[2, 5], [3, 5]]
[[2, 6], [3, 6]]
[[2, 7], [3, 7]]
[[4, 7]]
```

board2.txt

```
Please enter the filename for the game board: board2.txt
Filename: Checking.py

Line #    Mem usage    Increment  Occurences   Line Contents
=============================================================
   265  38.2773 MiB  38.2773 MiB           1   @profile(precision=4)
   266                                         def UCS(goal, start, empty_tiles, mSize, nSize):
   267
```

```
UCS
Duration: 0.32685208320617676
[[2, 3], [2, 4]]
[[2, 5]]
[[2, 6], [2, 7]]
[[3, 6], [3, 7]]
[[3, 8]]
[[4, 8], [5, 8]]
[[4, 7], [5, 7]]
[[6, 7]]
[[6, 5], [6, 6]]
[[6, 4]]
[[6, 2], [6, 3]]
[[7, 2], [7, 3]]
[[7, 4]]
```

```
Line #    Mem usage    Increment  Occurences   Line Contents
=============================================================
   211  38.3008 MiB  38.3008 MiB           1   @profile(precision=4)
   212                                         def ASTAR(goal, start, empty_tiles, mSize, nSize):
   213
   214  38.3008 MiB   0.0000 MiB           1       frontier = []
   215  38.3008 MiB   0.0000 MiB           1       closed   = []
   216  38.3008 MiB   0.0000 MiB           1       start_node = Node(start.getCoordinate(), [], 0)
   217  38.3008 MiB   0.0000 MiB           1       frontier   = Queue(frontier, start_node)
```

```
ASTAR
Duration: 0.23301291465759277
[[2, 3], [2, 4]]
[[2, 5]]
[[2, 6], [2, 7]]
[[3, 6], [3, 7]]
[[3, 8]]
[[4, 8], [5, 8]]
[[4, 7], [5, 7]]
[[6, 7]]
[[6, 5], [6, 6]]
[[6, 4]]
[[6, 2], [6, 3]]
[[7, 2], [7, 3]]
[[7, 4]]
```

board3.txt

```
Please enter the filename for the game board: board3.txt
Filename: Checking.py

Line #    Mem usage    Increment  Occurences   Line Contents
=============================================================
   265  38.3047 MiB  38.3047 MiB           1   @profile(precision=4)
   266                                         def UCS(goal, start, empty_tiles, mSize, nSize):
   267
   268  38.3086 MiB   0.0039 MiB           1       frontier = []
   269  38.3086 MiB   0.0000 MiB           1       closed   = []
   270  38.3086 MiB   0.0000 MiB           1       start_node = Node(start.getCoordinate(), [], 0)
   271  38.3086 MiB   0.0000 MiB           1       frontier   = Queue(frontier, start_node)
```

```
UCS
Duration: 0.200164794921875
[[0, 0], [1, 0]]
[[2, 0]]
[[2, 1], [2, 2]]
[[2, 3]]
[[2, 4], [2, 5]]
[[2, 6]]
[[2, 7], [2, 8]]
[[3, 7], [3, 8]]
[[4, 7], [4, 8]]
[[4, 9]]
```

```
Line #    Mem usage    Increment  Occurences   Line Contents
=============================================================
   211  38.3281 MiB  38.3281 MiB           1   @profile(precision=4)
   212                                         def ASTAR(goal, start, empty_tiles, mSize, nSize):
   213
   214  38.3281 MiB   0.0000 MiB           1       frontier = []
   215  38.3281 MiB   0.0000 MiB           1       closed   = []
   216  38.3281 MiB   0.0000 MiB           1       start_node = Node(start.getCoordinate(), [], 0)
   217  38.3281 MiB   0.0000 MiB           1       frontier   = Queue(frontier, start_node)
```

```
ASTAR
Duration: 0.03590202331542969
[[0, 0], [1, 0]]
[[2, 0]]
[[2, 1], [2, 2]]
[[2, 3]]
[[2, 4], [2, 5]]
[[2, 6]]
[[2, 7], [2, 8]]
[[3, 7], [3, 8]]
[[4, 7], [4, 8]]
[[4, 9]]
```

board4.txt

```
Please enter the filename for the game board: board4.txt
Filename: Checking.py

Line #    Mem usage    Increment  Occurences   Line Contents
=============================================================
   265  38.2891 MiB  38.2891 MiB           1   @profile(precision=4)
   266                                         def UCS(goal, start, empty_tiles, mSize, nSize):
   267
   268  38.2930 MiB   0.0039 MiB           1       frontier = []
   269  38.2930 MiB   0.0000 MiB           1       closed   = []
   270  38.2930 MiB   0.0000 MiB           1       start_node = Node(start.getCoordinate(), [], 0)
   271  38.2930 MiB   0.0000 MiB           1       frontier   = Queue(frontier, start_node)
```

```
UCS
Duration: 0.5243701934814453
[[2, 5], [3, 5]]
[[4, 5]]
[[5, 5], [6, 5]]
[[7, 5]]
[[8, 5], [9, 5]]
[[10, 5]]
[[11, 5], [12, 5]]
[[13, 5]]
[[14, 5], [15, 5]]
[[16, 5]]
```

```
Line #    Mem usage    Increment  Occurences   Line Contents
=============================================================
   211  38.3242 MiB  38.3242 MiB           1   @profile(precision=4)
   212                                         def ASTAR(goal, start, empty_tiles, mSize, nSize):
   213
   214  38.3242 MiB   0.0000 MiB           1       frontier = []
   215  38.3242 MiB   0.0000 MiB           1       closed   = []
   216  38.3242 MiB   0.0000 MiB           1       start_node = Node(start.getCoordinate(), [], 0)
   217  38.3242 MiB   0.0000 MiB           1       frontier   = Queue(frontier, start_node)
```

```
ASTAR
Duration: 0.02837395668029785
[[2, 5], [3, 5]]
[[4, 5]]
[[5, 5], [6, 5]]
[[7, 5]]
[[8, 5], [9, 5]]
[[10, 5]]
[[11, 5], [12, 5]]
[[13, 5]]
[[14, 5], [15, 5]]
[[16, 5]]
```

board5.txt

```
Please enter the filename for the game board: board5.txt
Filename: Checking.py

Line #    Mem usage    Increment  Occurences   Line Contents
=============================================================
   265  38.2617 MiB  38.2617 MiB           1   @profile(precision=4)
   266                                         def UCS(goal, start, empty_tiles, mSize, nSize):
   267
   268  38.2656 MiB   0.0039 MiB           1       frontier = []
   269  38.2656 MiB   0.0000 MiB           1       closed   = []
   270  38.2656 MiB   0.0000 MiB           1       start_node = Node(start.getCoordinate(), [], 0)


UCS
Duration: 0.35521531105041504
[[2, 3], [3, 3]]
[[1, 3]]
[[1, 1], [1, 2]]
[[2, 1], [2, 2]]
[[2, 3]]
[[2, 4], [2, 5]]
[[2, 6]]
[[2, 7], [2, 8]]
[[3, 7], [3, 8]]
[[3, 9]]
[[4, 9], [5, 9]]
[[6, 9]]
[[6, 7], [6, 8]]
[[6, 6]]
[[7, 6], [8, 6]]
[[7, 5], [8, 5]]
[[7, 4], [8, 4]]
[[9, 4]]
[[9, 2], [9, 3]]
[[9, 1]]


Line #    Mem usage    Increment  Occurences   Line Contents
=============================================================
   211  38.3086 MiB  38.3086 MiB           1   @profile(precision=4)
   212                                         def ASTAR(goal, start, empty_tiles, mSize, nSize):
   213
   214  38.3086 MiB   0.0000 MiB           1       frontier = []
   215  38.3086 MiB   0.0000 MiB           1       closed   = []
   216  38.3086 MiB   0.0000 MiB           1       start_node = Node(start.getCoordinate(), [], 0)
   217  38.3086 MiB   0.0000 MiB           1       frontier   = Queue(frontier, start_node)


ASTAR
Duration: 0.20295500755310059
[[2, 3], [3, 3]]
[[1, 3]]
[[1, 1], [1, 2]]
[[2, 1], [2, 2]]
[[3, 1], [3, 2]]
[[3, 3]]
[[3, 4], [3, 5]]
[[3, 6]]
[[3, 7], [3, 8]]
[[3, 9]]
[[4, 9], [5, 9]]
[[6, 9]]
[[6, 7], [6, 8]]
[[6, 6]]
[[7, 6], [8, 6]]
[[7, 5], [8, 5]]
[[7, 4], [8, 4]]
[[9, 4]]
[[9, 2], [9, 3]]
[[9, 1]]
```