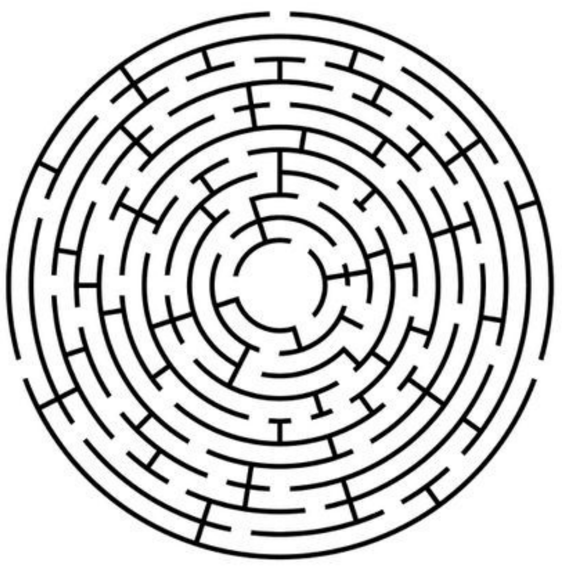
A* Algorithm Maze Solver



FALL 2020 – COMP 3958 FINAL PROJECT

Jon Andre Briones & Brian Li

Table of Contents

Summary

This program solves a maze using the A* Search Algorithm. A maze can be thought of as a 2D grid with cells with differing weights or obstacles. We start from a starting point and use the A* Search Algorithm to determine the shortest path to a target point.

A* Search Algorithm

The algorithm used in this program to find the best path to solve a maze is the A* search algorithm. It is a best-first search algorithm that attempts to find the path from a starting node to a target node that would have the smallest cost.

The path that is chosen is the path that minimizes: $f(n) = g(n) + h(n)$

Where n is the next node on the path, $g(n)$ is the cost of the path from the starting node to n, and $h(n)$ is a heuristic function that estimates the cost of the minimal path from n to the target.

Getting Started

This section provides instructions on how to begin using the program.

Requirements

Ocaml is required to run this program. All remaining required files to run this program are included in the same .zip file this documentation was provided with.

Building the Program

To build this program, in your command line interface, navigate to the folder that the program's files are located in. Then, execute the following command:

```
corebuild astar.native
```

Running the Program

To run this program with the provided maze, in your command line interface, navigate to the folder that the program was built in. Then, execute the following command:

```
./astar.native maze.txt
```

Once the program is running, you will be prompted to enter the starting coordinate and the target coordinate for the maze. Ensure to input the coordinates in the following format: x y

NOTE: maze.txt is the provided maze, but this program can be executed with any named .txt file containing a maze.

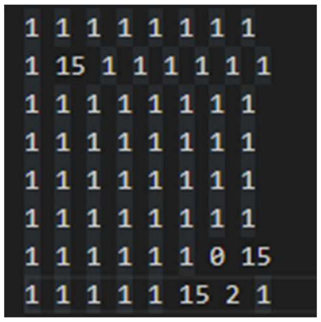
The Maze

The maze in this program is represented as a matrix. Each element is a room represented as an integer that indicates the weight of the room. The meanings of each element are described in Table 1.

Table 1: Values of elements of a maze matrix and their meanings

Integer	**Meaning**
0	A wall that cannot be passed
1	A room with "normal" terrain
2 – 4	A room with "rough" terrain
5 +	A room with "very rough" terrain

The example maze provided in the text file is an 8 x 8 maze with one wall, three rooms with "very rough" terrain, one room with "rough" terrain, and fifty-four rooms with "normal" terrain. The example maze is shown in Figure 1 below.



```
1 1 1 1 1 1 1 1
1 15 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 0 15
1 1 1 1 1 15 2 1
```

Figure 1: Example maze provided for this program.

As shown in Figure 1, mazes are represented as a square matrix, with columns being separated by ' ' and rows separated by '\n'.

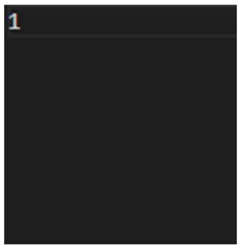
Creating Your Own Maze

These steps outline how to create your own maze to be used for this program.

1. Create a new .txt file, it can have any name.

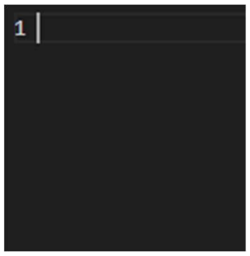
2. Decide on the dimensions of your maze.

3. Type in an integer representing the weight of the first room of your maze.



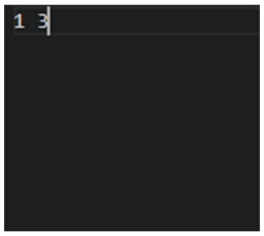
```
1
```

4. Press spacebar to move onto the next room (column).



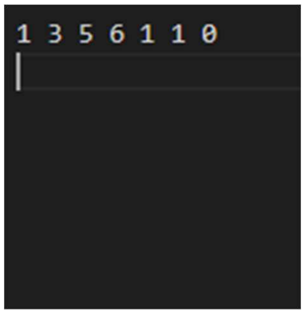
```
1 
```

5. Repeat steps 3 – 4 until you wish to move onto the next row.



```
1 3
```

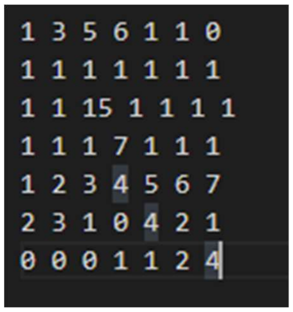
6. Press enter to indicate a new row.



```
1 3 5 6 1 1 0

```

7. Repeat steps 3 – 6 until your maze is complete.



```
1 3 5 6 1 1 0
1 1 1 1 1 1 1
1 1 15 1 1 1 1
1 1 1 7 1 1 1
1 2 3 4 5 6 7
2 3 1 0 4 2 1
0 0 0 1 1 2 4
```

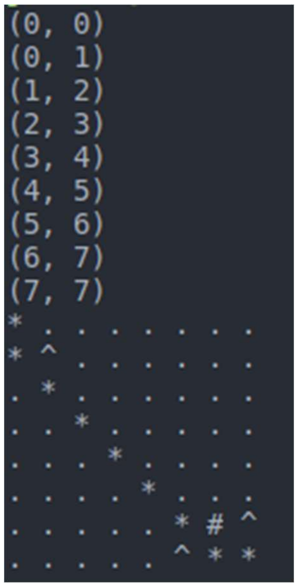
8. Save the .txt file into the same directory as the astar.ml file.

Results

This section will outline how to interpret the results from this program.

Output

This program will print out the coordinates that the resulting path follows. It will then print a graphical representation of the maze, also showing the path. The output from the provided maze to this program is displayed in Figure 2.



```
(0, 0)
(0, 1)
(1, 2)
(2, 3)
(3, 4)
(4, 5)
(5, 6)
(6, 7)
(7, 7)
* . . . . . . .
* ^ . . . . . .
. * . . . . . .
. . * . . . . .
. . . * . . . .
. . . . * . . .
. . . . . * # ^
. . . . . ^ * *
```

Figure 2: The output from the program using the provided maze.

Interpreting the Graphical Output

To interpret the graphical maze and path outputted by the program, one needs to understand the meanings of the symbols found in Table 2.

Table 2: Symbols in the graphical output of the program with their meanings

Symbol	Meaning
#	A wall
.	"Normal" terrain
-	"Rough" terrain
^	"Very rough" terrain
*	The path our maze solver has followed