TDT4136 - Assignment 2
Applying the A* search

Arthur Testard - id: 105022

1 Overview

The A* algorithm problem is one of the best known algorithms used to find the shortest past on a graph. According to the lecture of [RN21], it evaluates nodes by combining $g(n)$, the cost to reach the node, and $h(n)$, the cost to get from the node to the goal, in order to get $f(n)$, the estimated cost of the cheapest solution through n:

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

We require that h is a admissible heuristic is one that never overestimates the cost to reach the goal. Manhattan distance or Euclidean distance works.

2 A* Implementation

Our implementation were mostly inspired by [PAT14]'s ones which can be find on the following link: `https://www.redblobgames.com/pathfinding/a-star/implementation.py`.

It basically works as following:

We initialize the frontier, which can be called flood fill as well. Its variable is a heap of next locations. We then do a loop on the following instructions until frontier is empty:

1. We pick and remove a location from the frontier
2. Looks on this location's neighbors, remove walls and non-reachable cases. We then add all of those neighbors which has not been reached and add them to both frontier heap and reached path.

To plot the graph, we then change all of the reached cells for the shortest path to the value 0 and plot it.

In the following parts we will see different situations on a map and how our A* algorithm resolve it.

1. In part 1, we resolve a situation with cells walkable or non-walkable.
2. In part 2, the cells have different costs. We will find the shortest path depending of the distance but also the type of each cell.
3. In part 3 (optionnal), we are in the same situation than part 2 but the goal is moving.

The map will be a 2D representation of Samfundet which can be find on Figure 1.

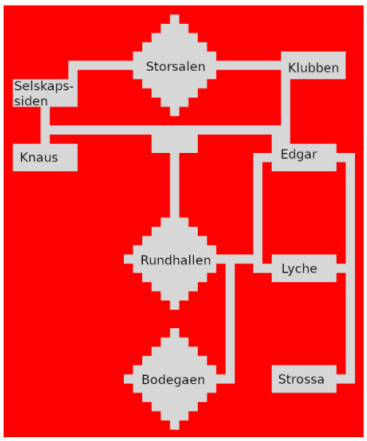

Figure 1: A 2D-world representation of Samfundet

3 Part 1 - Grid with obstacles

3.1 Task 1

In this first part we have to find the shortest path from Rundhallen to Strossa using our implementation of the A* algorithm. The shortest path found by A* algorithm is drawn on Figure 2, where pink and blue points are respectively start and goal positions, and the path is drawn in yellow. The cost of this path is 37.

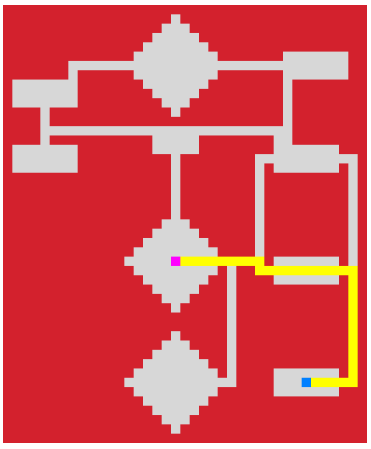

Figure 2: Shortest path from Rundhallen to Strossa by A* algorithm ($cost = 37$)

3.2 Task 2

For this task we want to find the shortest path from Strossa to Selskapssiden. Our implementation of A* algo-

rithm found the path drawn on Figure 3 using Manhattan distance for the heuristic function and Figure 4 using Euclidean distance for the heuristic function. The cost of both paths are 71.

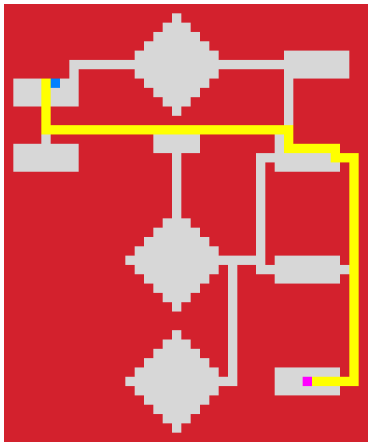

Figure 3: Shortest path from Strossa to Selskapssiden by A* algorithm with Manhattan distance ($cost = 71$)

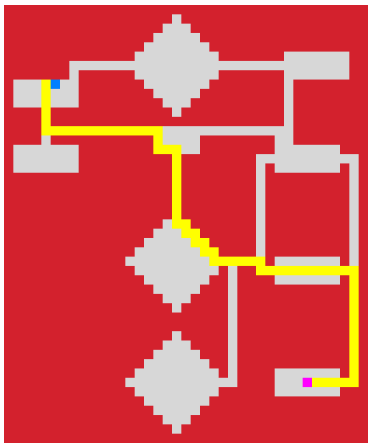

Figure 4: Shortest path from Strossa to Selskapssiden by A* algorithm with Euclidean distance ($cost = 71$)

4 Part 2 - Grids with different costs

In this part we introduce cost associated to cells. We can find they description on Table 1.

Description	Cost
Flat ground	1
Stairs	2
Crowded stairs	3
Crowded room	4

Table 1: Cell types and their associated costs

4.1 Task 3

In this situation we modelized crowded stairs by putting cost on those cells. We then want to fin the shortest path from Lyche to Klubben, taking into consideration those weights. Our implementation find that the shortest path cost 57.

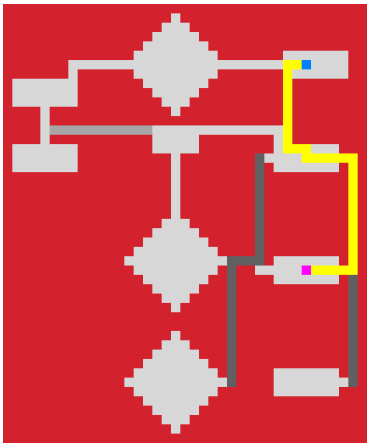

Figure 5: Shortest path from Lyche to Klubben by A* algorithm with Manhattan distance ($cost = 57$)

4.2 Task 4

We put ourself in a similar situation than the last one but we consider Edgar to be very crowded by people. So we recalculate the shortest path with our implementation taking those weights into consideration. We find the path plot on Figure 6 with a 64 cost (we can notice that this cost is higher than the last one as well, it would be an issue if it was not).

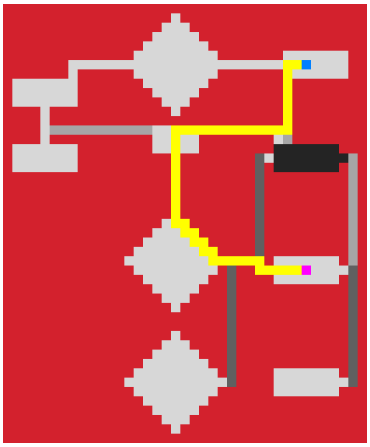

Figure 6: Shortest path from Lyche to Klubben, when Edgar is full, by A* algorithm with Manhattan distance ($cost = 64$)

5 Part 3 – Moving Goal (Optional)

5.1 Task 5

In that situation we consider a moving goal. We change a bit our A* algorithm to change the position every 4 of our loops. Depending of the norm used for it we find two different plots, Figure 7 and Figure 8 but both cost 27. However the second result is more satisfying so we could maybe add an other condition on our $f(n)$ cost function, which does not seems enough.

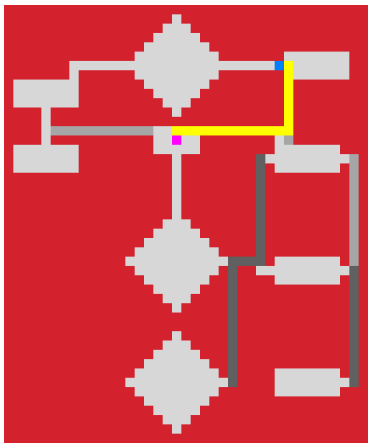

Figure 7: Shortest path to a moving goal with a designed norm ($cost = 27$)

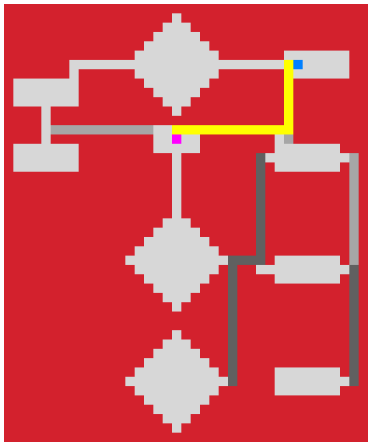

Figure 8: Shortest path to a moving goal with Manhattan distance ($cost = 27$)

6 Indication to run the code

The code works with the 'Map.py' file as it was given to do the assignment. It just have to be in the same directory to make the import works correctly. When you already have created your juyter kernel, you just have to run 'a_star.ipynb' to get the results presented in this report.

References

[PAT14] Amit PATEL. Introduction to the A* Algorithm. 2014. Available on 2023-09-21 at `https://www.redblobgames.com/pathfinding/a-star/introduction.html`.

[RN21] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach, 4th us ed. *University of California, Berkeley*, 2021.