Assignment 2

TDT4136 - Introduction to Artificial Intelligence
Fall 2023

Elena Filippini
Francesca Grimaldi

1 Part 1

We adapted the implementation of the A* algorithm available at "redblobgames" as the foundational framework for our own implementation, that can be found at [red]. We instantiate a Map object from the code provided during the assignment lecture. The heuristic `h` uses Manhattan distance, although it is possible to choose the Euclidean one. We decided to represent the Manhattan distance, as it better suits the scenario described in the handout.

The cost function is composed of `h` and `g`, where `g` is `cost_to_node`. We utilized the Python `PriorityQueue` library to create the frontier, where we store the current node along with its corresponding cost. Additionally, we implemented the `neighbors` method, which calculates the coordinates to the north, east, south, and west of a given node, while also ensuring that they remain within the map boundaries.

The A* algorithm is executed inside the `find_path` method, which in the end draws the optimal path from the goal node retracing its steps back to the starting node, by following the predecessor nodes. Each node along this path is marked in purple by updating its cell value to the specified character "`-`".

The first and the second task both revolve around finding the shortest path between two locations without encountering obstacles, regardless of the cell values (being set to 1 in Task 1), as they do not impact the cost function.

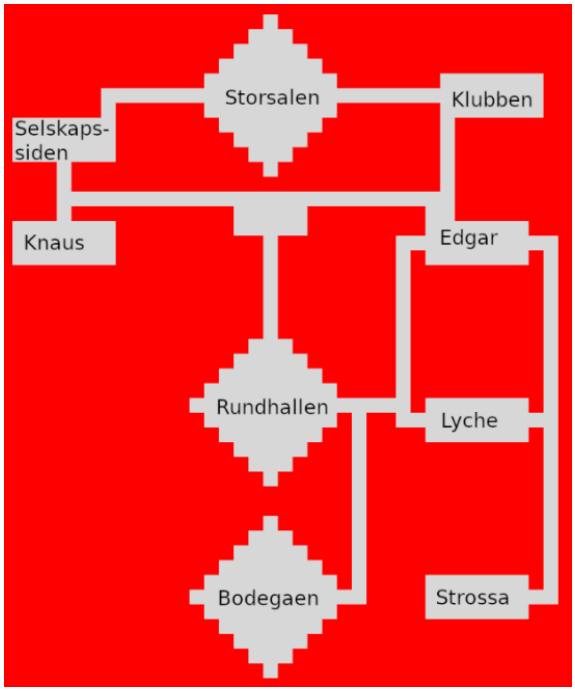
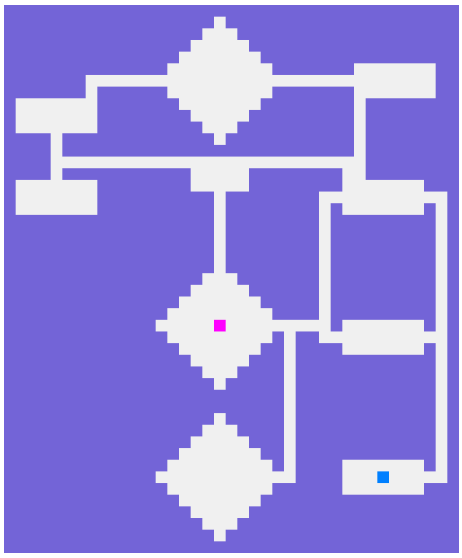


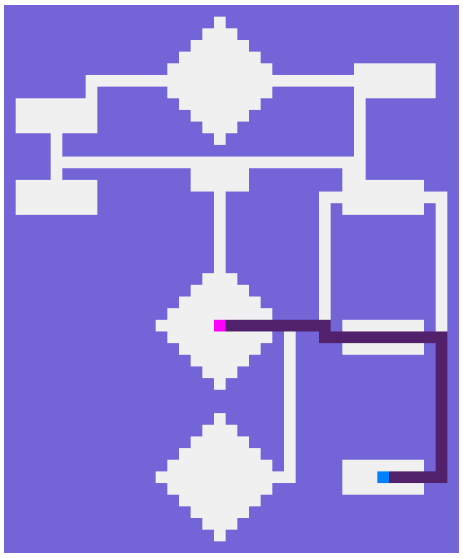
Figure 1: 2-D map of the building

1.1 Task 1

In task 1, as shown in 2a, we are starting from Rundhallen and we are reaching Strossa. All the cells have the same costs, therefore are all displayed with the same colour. It is visible from figure 2b that the shortest path has been selected.



(a) Initial scenario with start node in magenta and ending node in blue

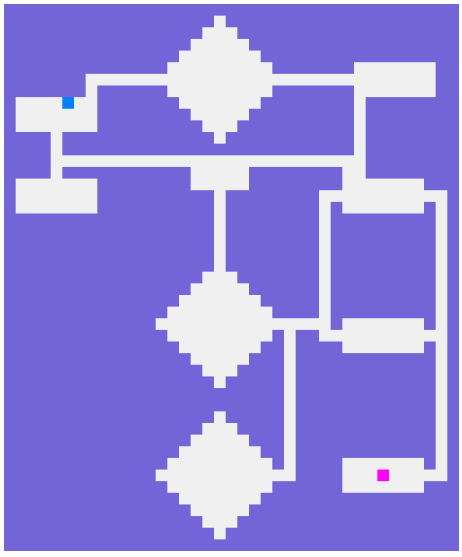


(b) Scenario with shortest path displayed

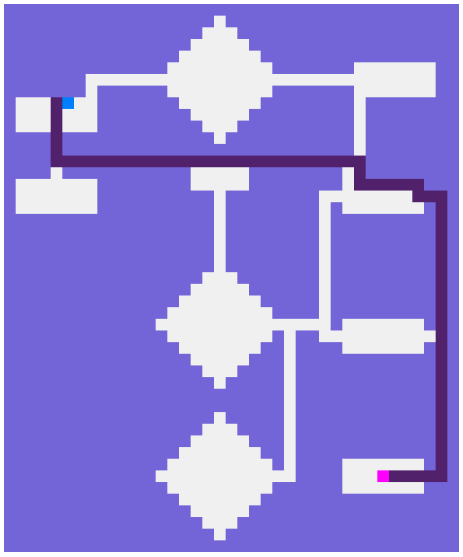
Figure 2: Task 1: initial scenario and final scenario with path displayed

1.2 Task 2

In this second task the start node is in Strossa and the goal is in Selkstapssiden, as shown in 3a. Also in this case it is evident from figure 3b that the path selected is the shortest one among the one possible ones.



(a) Initial scenario with start node in magenta and ending node in blue



(b) Scenario with shortest path displayed

Figure 3: Task 2 initial and ending state

2 Part 2

As mentioned before, in Part 1 all cell values are set to 1, and the only requirement is that the cell is not an obstacle (a condition that is verified within the `neighbors` method). Instead, in this

task each cell has a different cost, described in figure 4.

Since we make use of the `get_cell_value()` method from the Map object, our code remains consistent regardless of whether we are working on Part 1 or Part 2, as their settings are automatically determined when selecting the task.

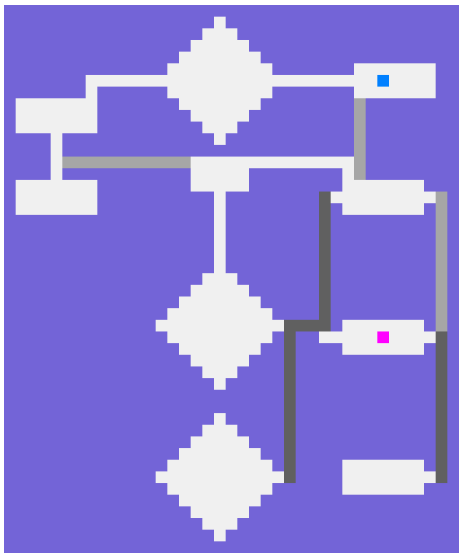
Char	Description	Cost
.	Flat ground	1
,	Stairs	2
:	Crowded Stairs	3
;	Crowded room	4

Figure 4: Different cell costs

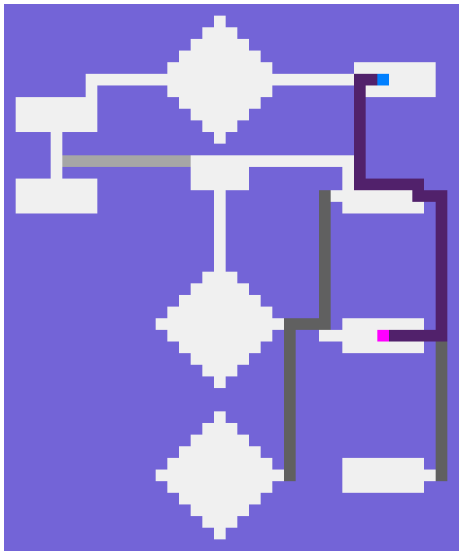
The maps 5a and 6a illustrate that areas shaded in darker grey represent higher costs.

2.1 Task 3

The figure 5 demonstrates that the chosen path considers the fewest grey cells while still being the shortest route.



(a) Initial scenario with start node in purple and ending node in blue

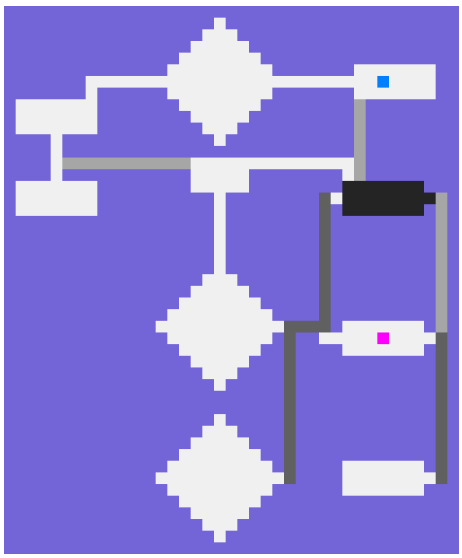


(b) Scenario with shortest path displayed

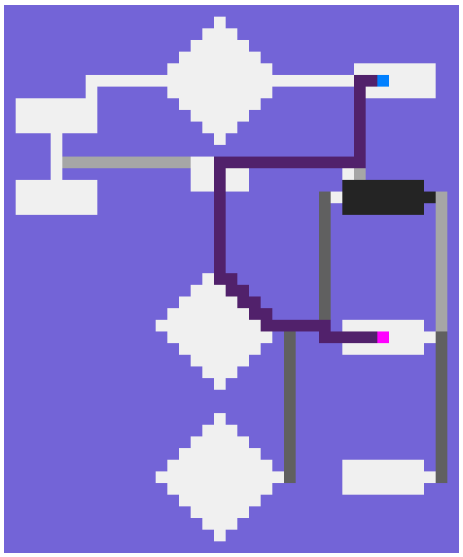
Figure 5: Task 3: initial scenario and final scenario with path displayed

2.2 Task 4

The goal of this fourth scenario is the same as the one in the third, but the "Edgar" room, which was previously walked through, experiences a cost increase, reaching its maximum value. Therefore, the selected path is changed, as it becomes more expensive to navigate through the dark building and follow the grey path, compared to circumventing it by passing through Rundhallen.



(a) Initial scenario with start node in purple and ending node in blue



(b) Scenario with shortest path displayed

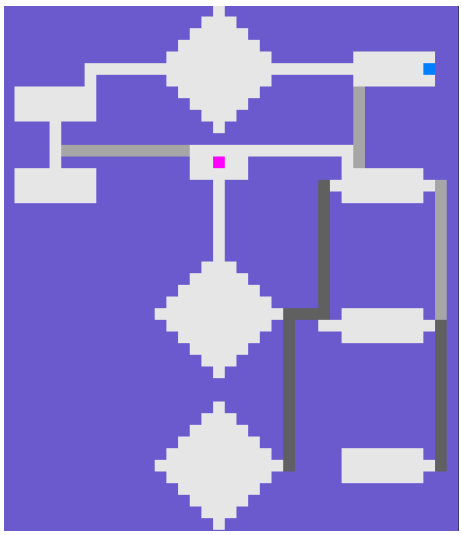
Figure 6: Task 4: initial scenario and final scenario with path displayed

3 Part 3

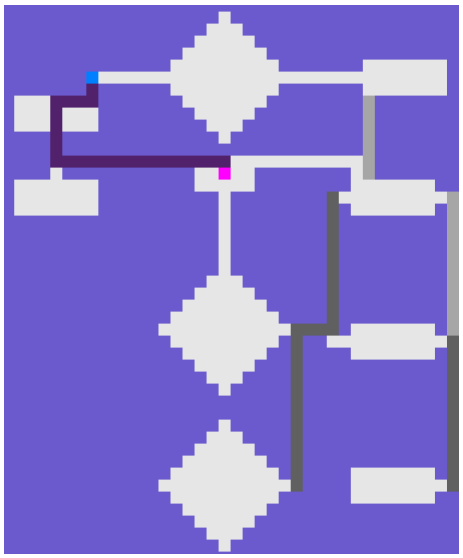
3.1 Task 5

The fifth task distinguishes itself from the previous ones by incorporating a dynamic objective. The goal initially resides in Klubben and gradually shifts towards Selskapssiden, moving one cell every 4 iterations.

To enable this behavior, we introduced the variable `task` as an input parameter to the `find_path` function, which in turn transfers it to the A* algorithm. Consequently, the algorithm assesses whether the goal is in motion. If it is, the algorithm invokes the `ticks` method, which is part of the provided code_handout [cod], responsible for updating the goal's position by one cell for every four calls.



(a) Initial scenario with start node in purple and ending node in blue



(b) Final scenario, where the goal has moved

Figure 7: Task 5: initial scenario and final scenario with path displayed

References

[cod] *code handout available on the Blackboard page of the course.*

[red] https://www.redblobgames.com/pathfinding/a-star/implementation.h.