## D* Replanning by updating cost:

The **D* algorithm** primarily consists of two main functions: **PROCESS-STATE** and **MODIFY-COST**.

**PROCESS-STATE** is used to compute the optimal path cost to the goal.

**MODIFY-COST** is used to change the arc cost function and enter the affected states to the **OPEN** list.

Initially the **TAG** for all states is set to be **NEW** and the **Goal cost h(G)** is set to be zero and the **Goal** is placed on the **OPEN** list.

The **PROCESS-STATE** function is repeatedly called until the robot's state, **X**, is removed from the **OPEN** list, i.e, **TAG(X) = CLOSED** at which point either the sequence **{X}** is computed or a value of **-1** is returned if the sequence does not exist.

The robot then follows the back-pointers in the sequence **{X}** until it either reaches the goal or discovers an error in the arc cost function, due an obstacle detected by the sensors.

The function **MODIFY-COST** is then called to correct the arc cost function and propagate the cost to all the affected states and place them all on the **OPEN** list.

## Faster Replanning of D* as compared to A* or Dijkstra's:

A* algorithm is an optimal algorithm, but it is highly insufficient and impractical to use to use A* for replanning in expansive environments. It has to completely replan the path once the robot discovers any disparity in the real arc cost and the measured (by sensor) arc cost. This is the case since A* does not propagate the change in the arc cost to all the affected states in real-time and hence has to do complete replanning all over again. This can be prominently seen in the case when start and goal is significantly far apart and the information about only the half map is available.

D* algorithm is a better alternative to A* for replanning in a dynamically changing environment and it is also very useful in complex and expensive environment. This algorithm also provides optimal path form start to goal but and also efficiently replan whenever the sensor measured some disparity between the measured arc cost and the one at the beginning of the traversal. This algorithm specifically works by propagating the change in arc cost function to all the affected states in real-time and thus avoids complete replanning of the path and hence is faster and computationally very expensive.

The D* algorithm plans the optimal path form the goal to the start and during runtime it moves from the start position to the goal. If in between if the robot encounters some obstacles, this reverse planning enables D* to avoid planning the entire path all over again. Hence, because of these reasons, D* algorithm is faster as compared to A* and Dijkstra's.

## **Difference between RRT* and Informed RRT*:**

RRT* is asymptotically optimal everywhere but not necessarily for a single query-problem.

RRT* algorithm finds a path between the start and goal but does not guarantee about the optimality of the obtained path. Since sampling-based methods does not guarantee optimality, but the informed RRT* method ensured to improve the solution obtained by the regular RRT* algorithm by the limiting the search to a sub-problem.

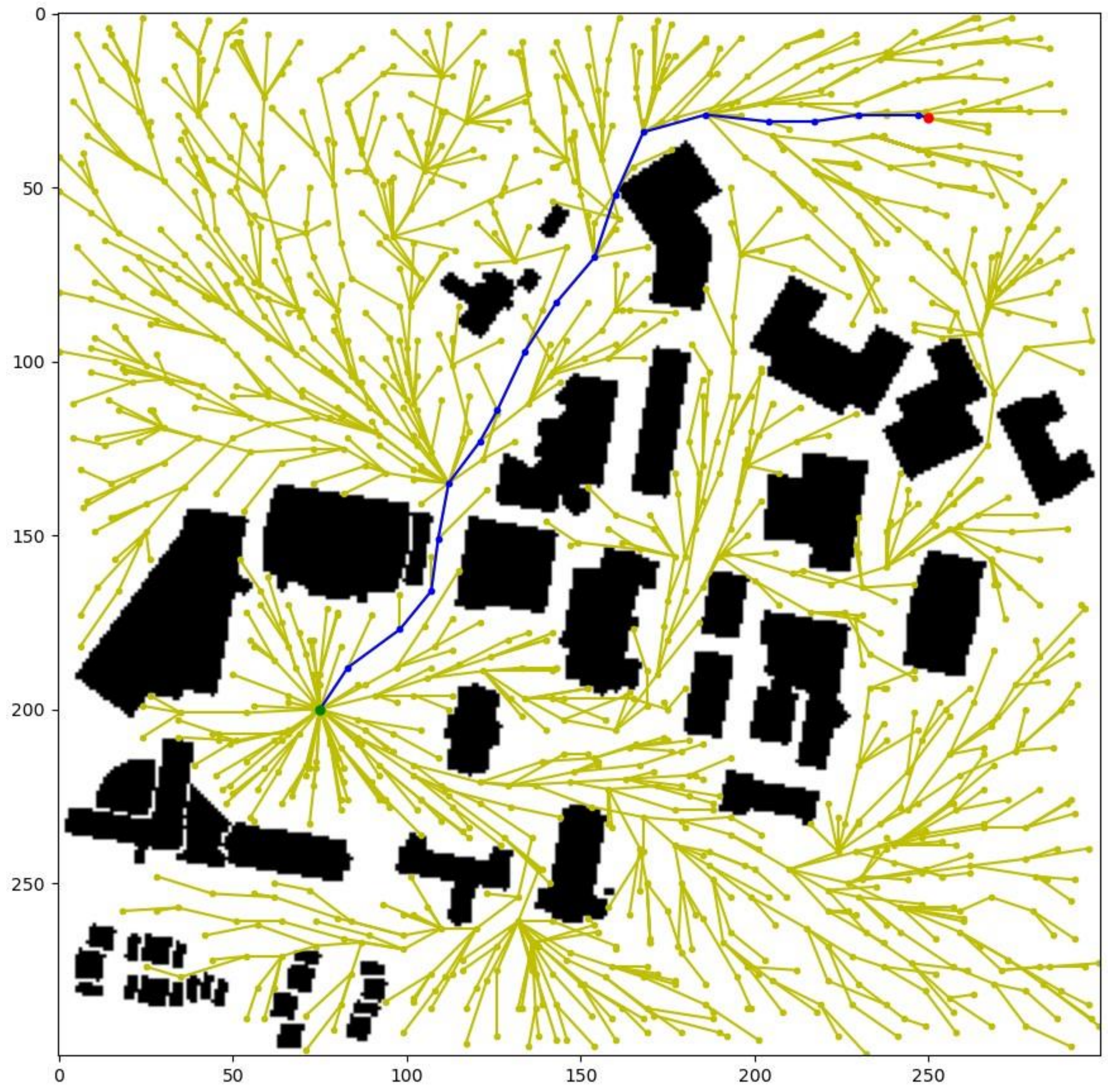
Informed RRT* algorithm tries to find an even shorter path from the start to the by sampling more points in an ellipse formed by keeping the start and goal as the focal points and using the cost of the path obtained by the regular RRT*. This enables searching in a sub-problem and trying to improve over the result obtained by RRT*.

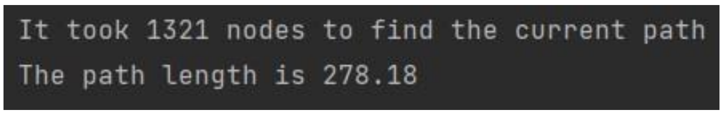
In case of the Informed RRT* algorithm the sub problem is defined as searching for a shorter path in an n-dimensional ellipse.

Informed RRT* uses more computational power to improve the quality of the path obtained and thus is computationally more expansive then the RRT* algorithm. However, RRT* algorithm is computationally cheaper but provides longer or more costly paths as compared to the Informed RRT* algorithm.

## Comparing Results for RRT* and Informed RRT*:

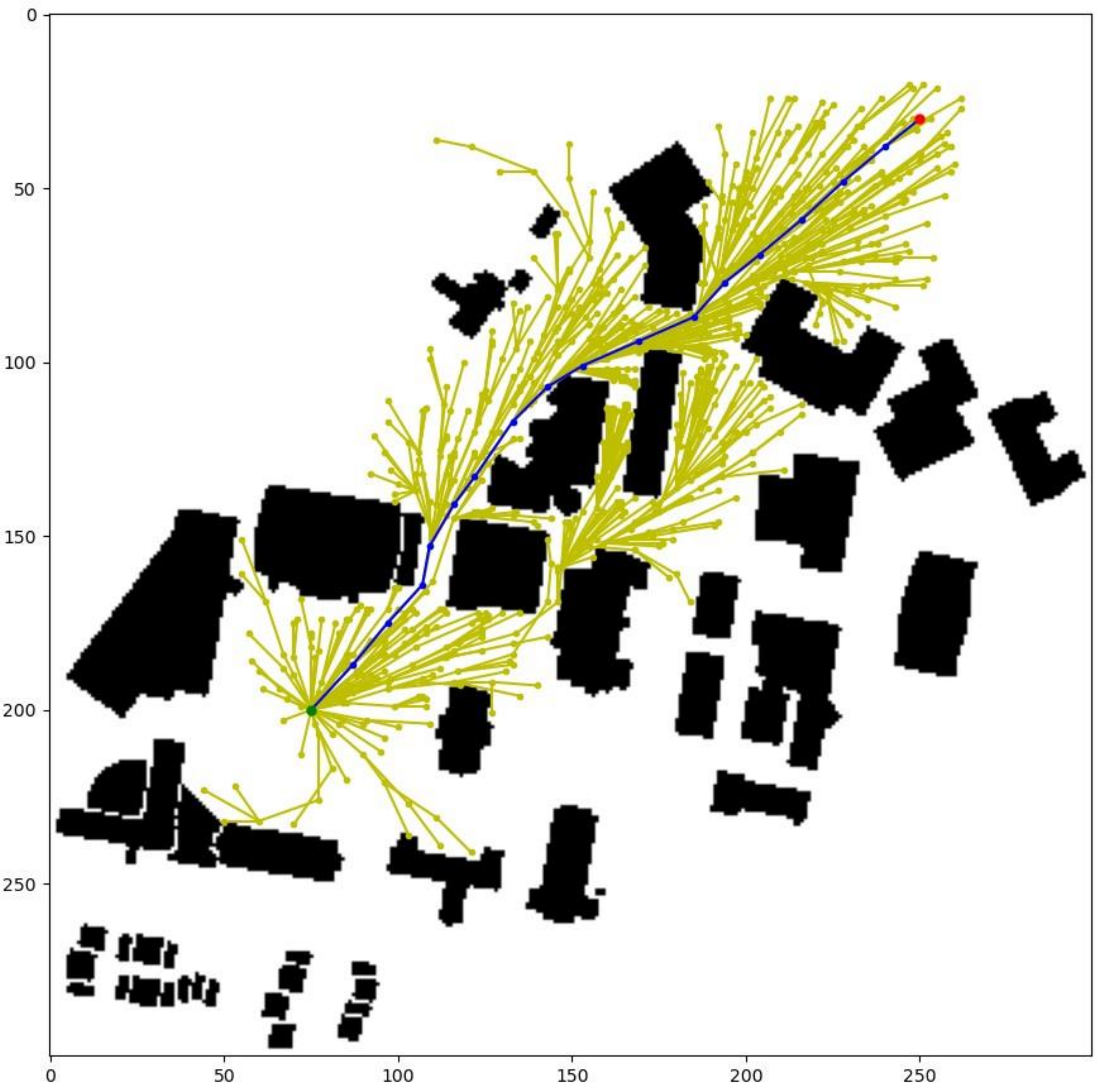
**RRT***

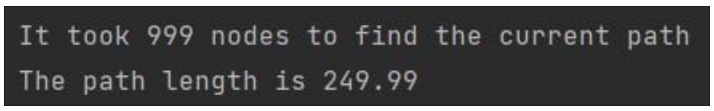




```
It took 1321 nodes to find the current path
The path length is 278.18
```

**Informed RRT***



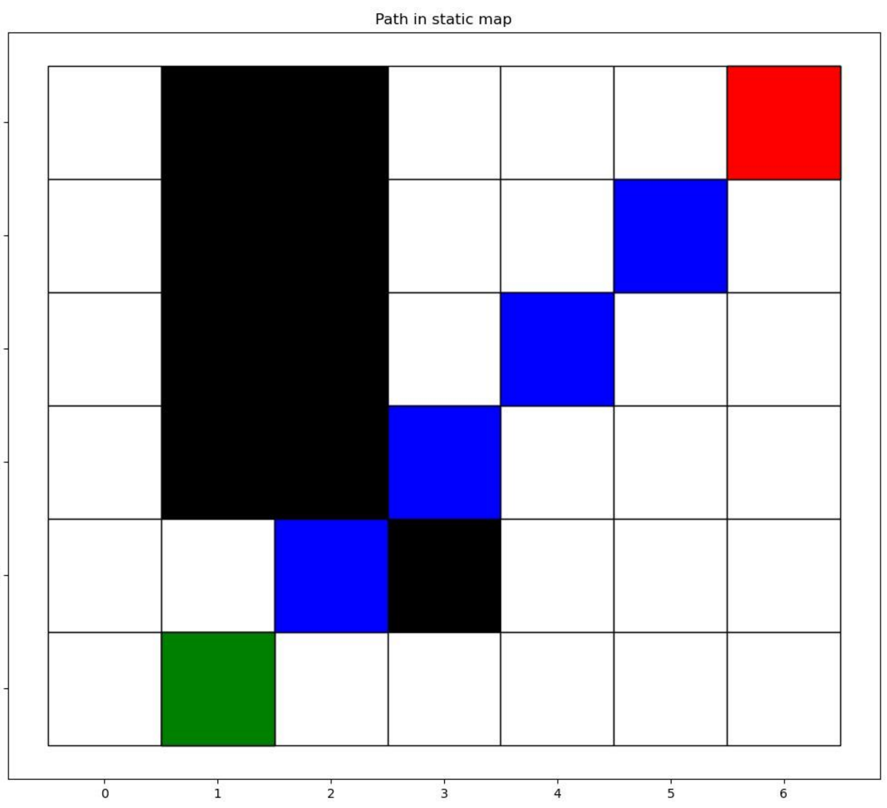


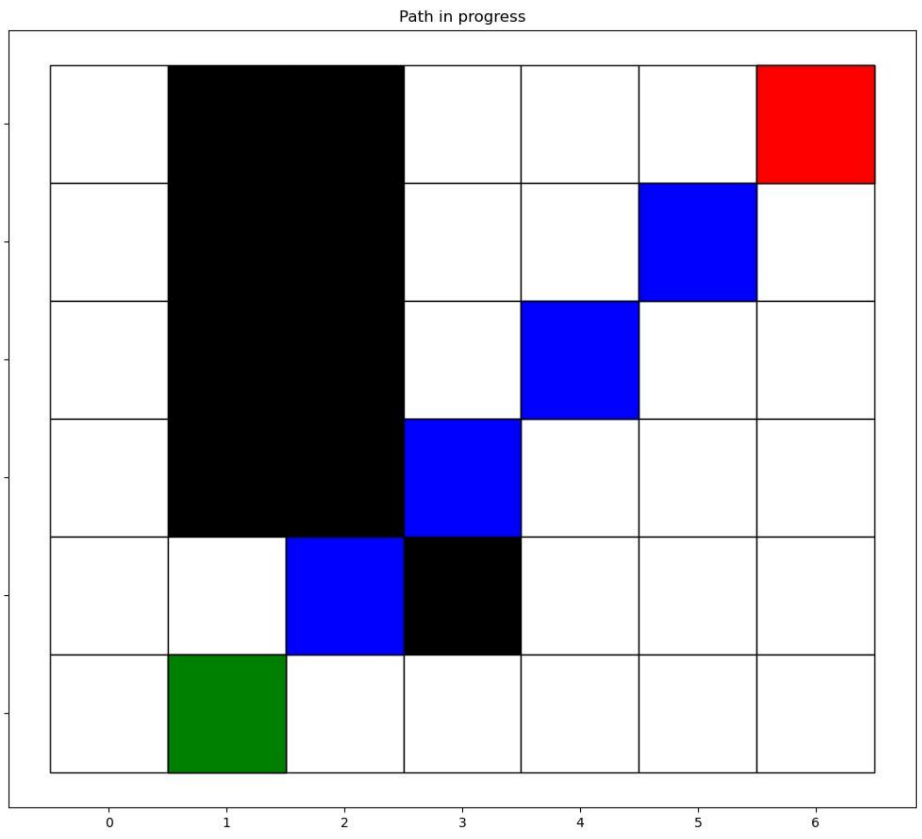
```
It took 999 nodes to find the current path
The path length is 249.99
```

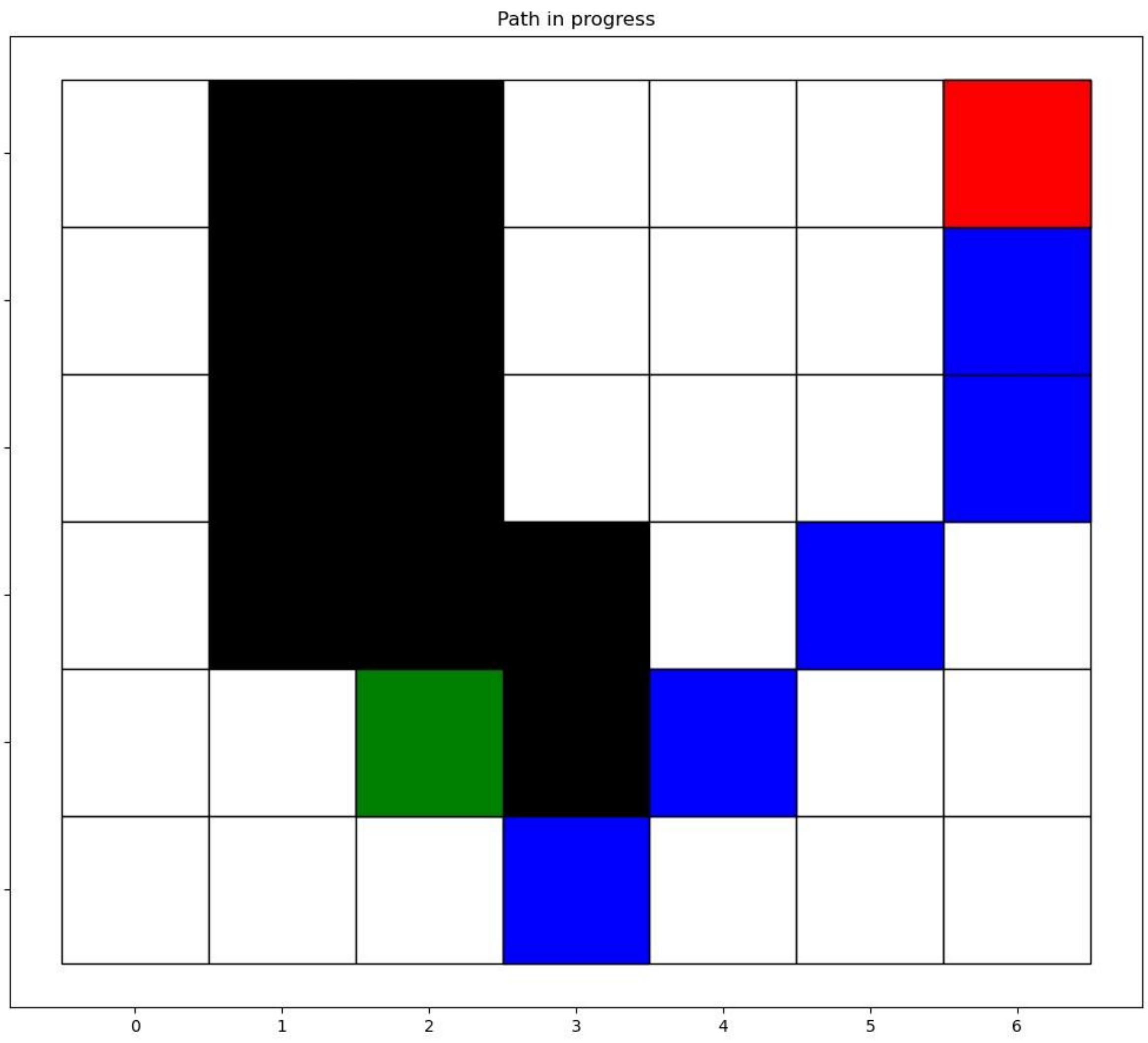
Here we can observe that in case of RRT* algorithm the points are randomly sampled all over the map and the path length found was 278.18. Whereas, in case of Informed RRT* algorithm, the points are randomly sampled within an ellipse formed between the start and the goal and not the entire map. It can also be seen that in this case the path length is only 249.99 which is shorter than in the case of RRT*. So we can say that Informed RRT* algorithm is able to find a shorter path as compared to RRT* even when same number of points (here,2000) were sampled in both the cases.

## <u>Results for D* Implementation:</u>

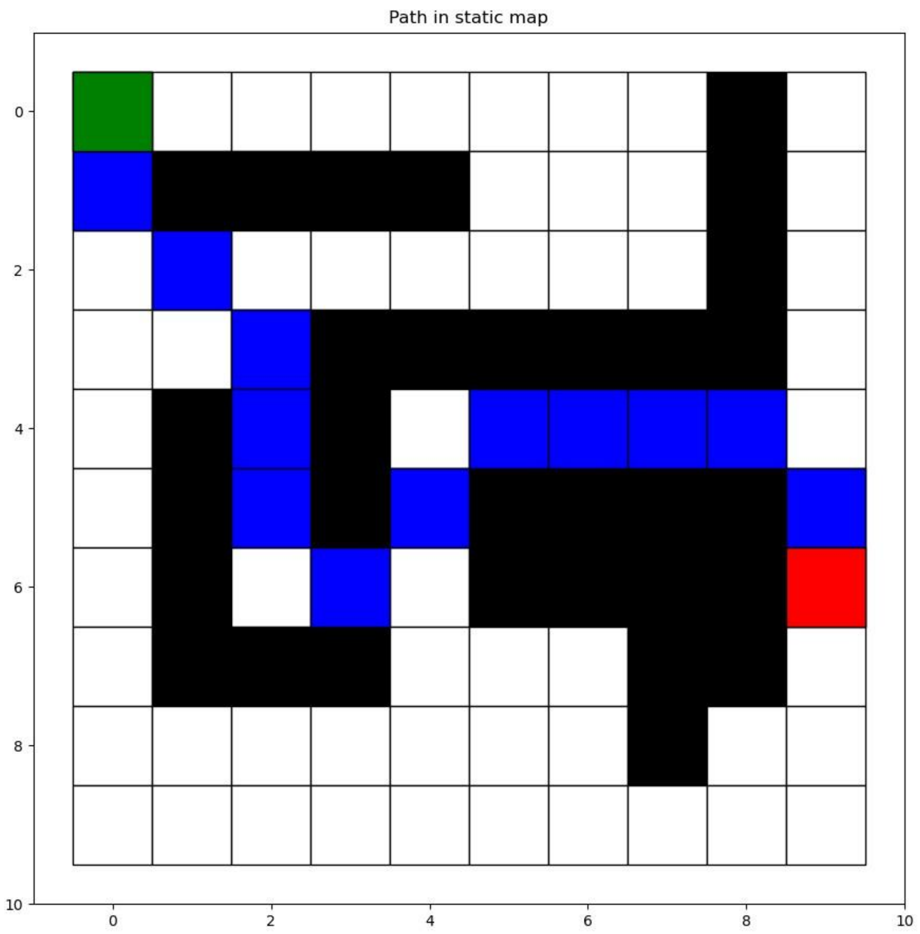
### MAP 1

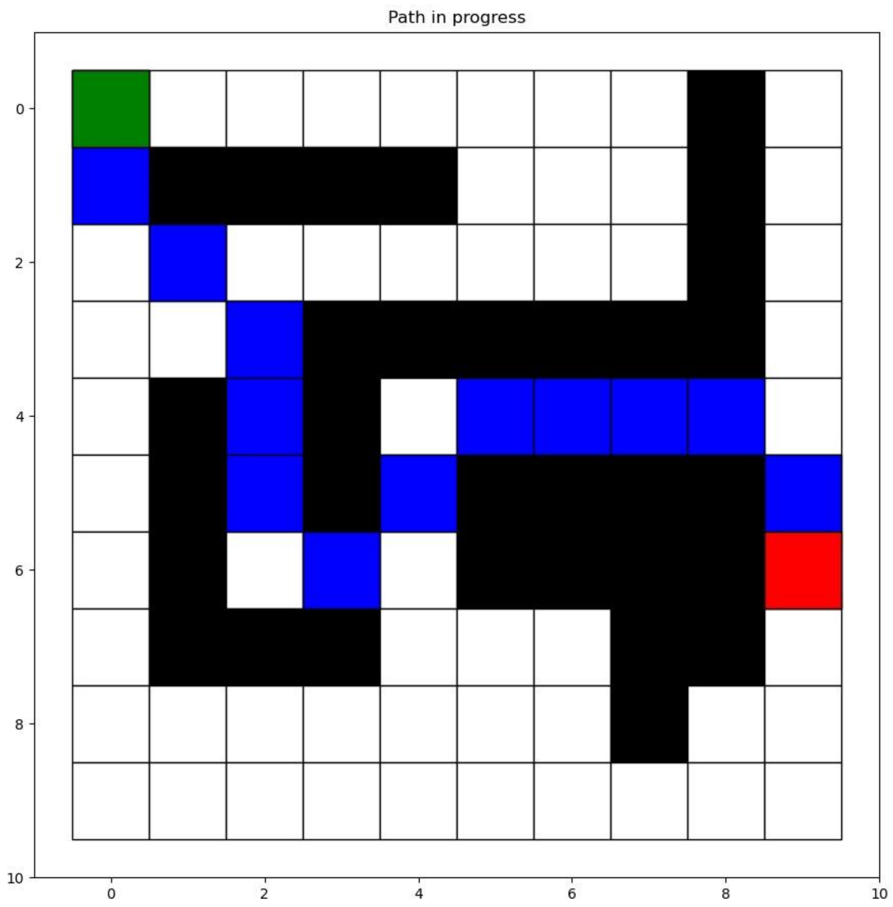


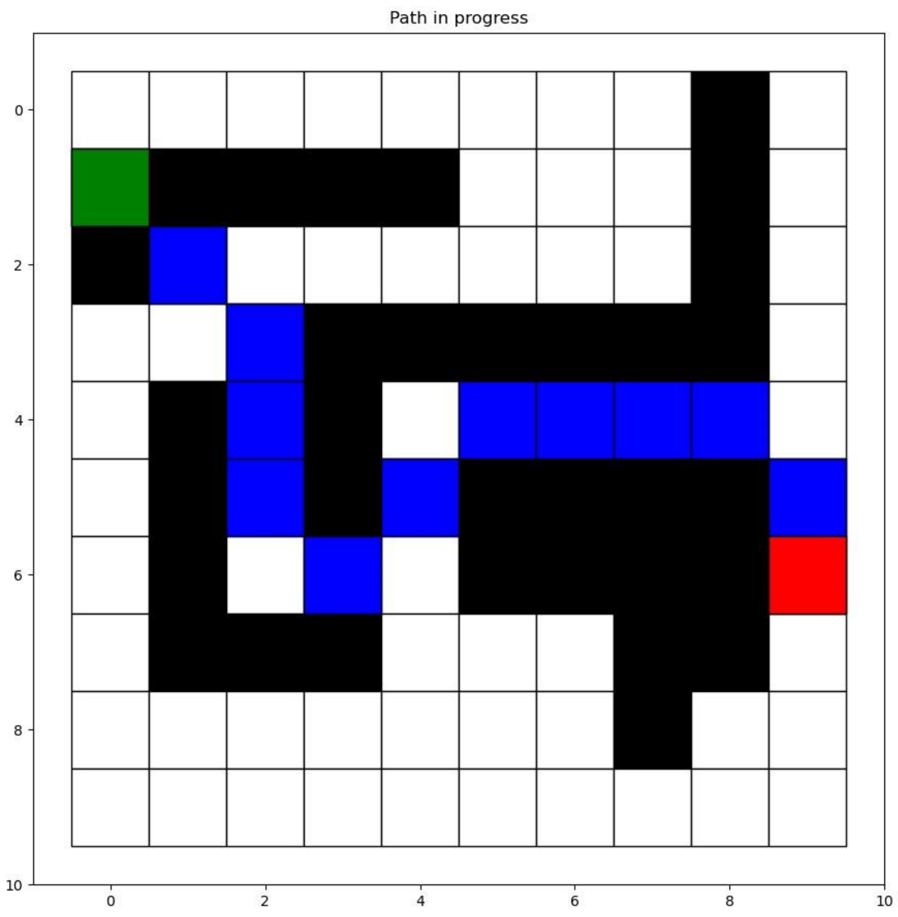


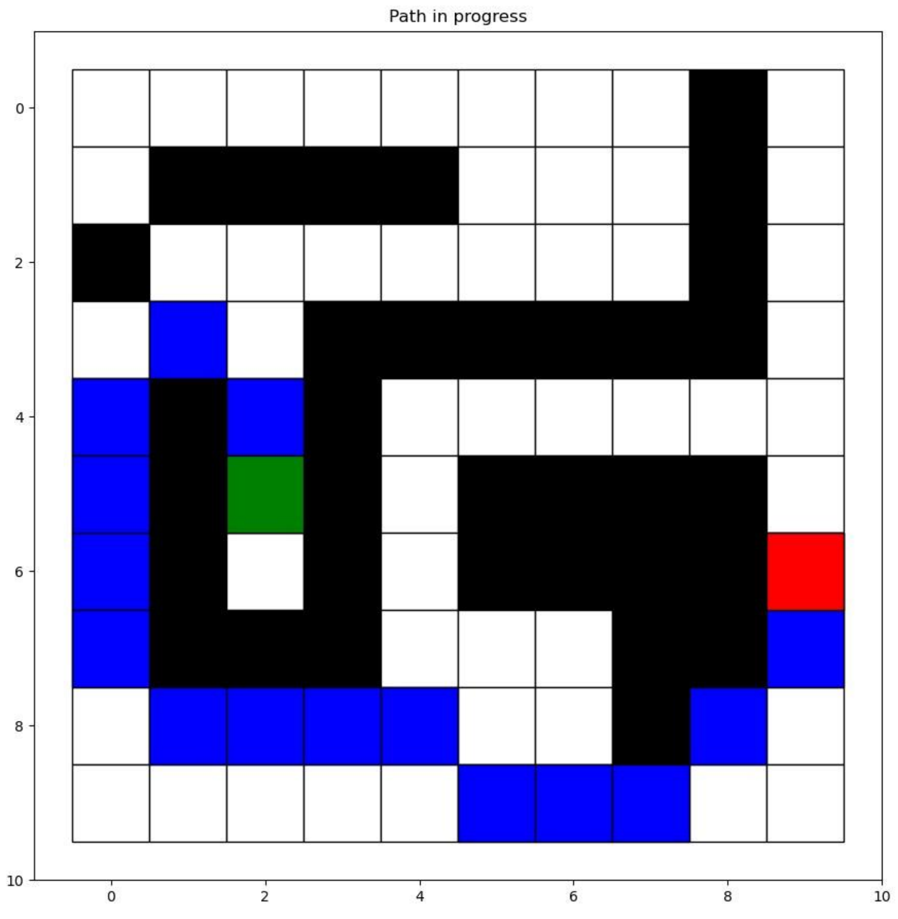

Path in progress
0
1
2
3
4
5
0
1
2
3
4
5
6

**MAP 2**

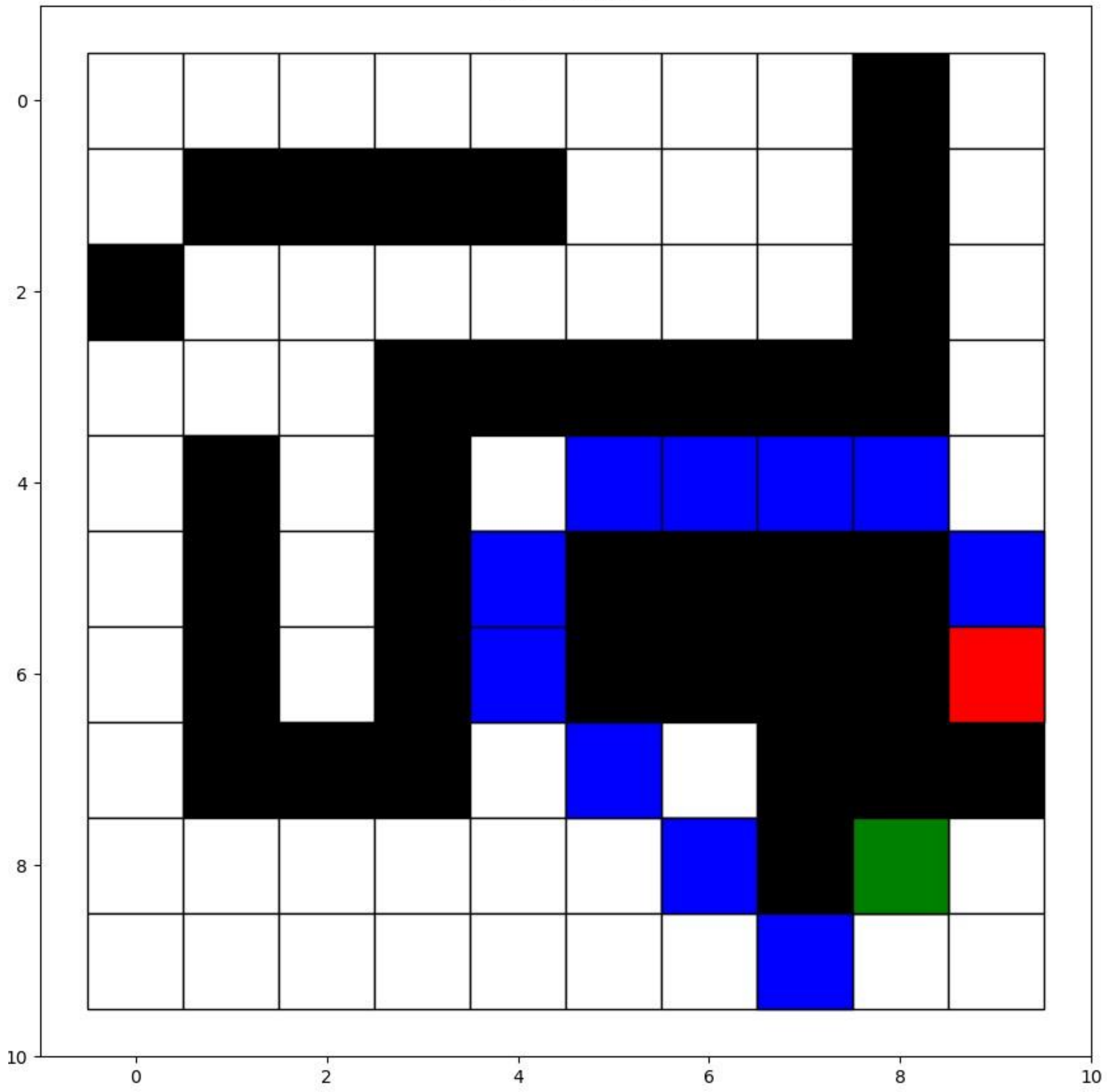

Path in static map
0
2
4
6
8
10
0
2
4
6
8
10


Path in progress
0
2
4
6
8
10
0
2
4
6
8
10

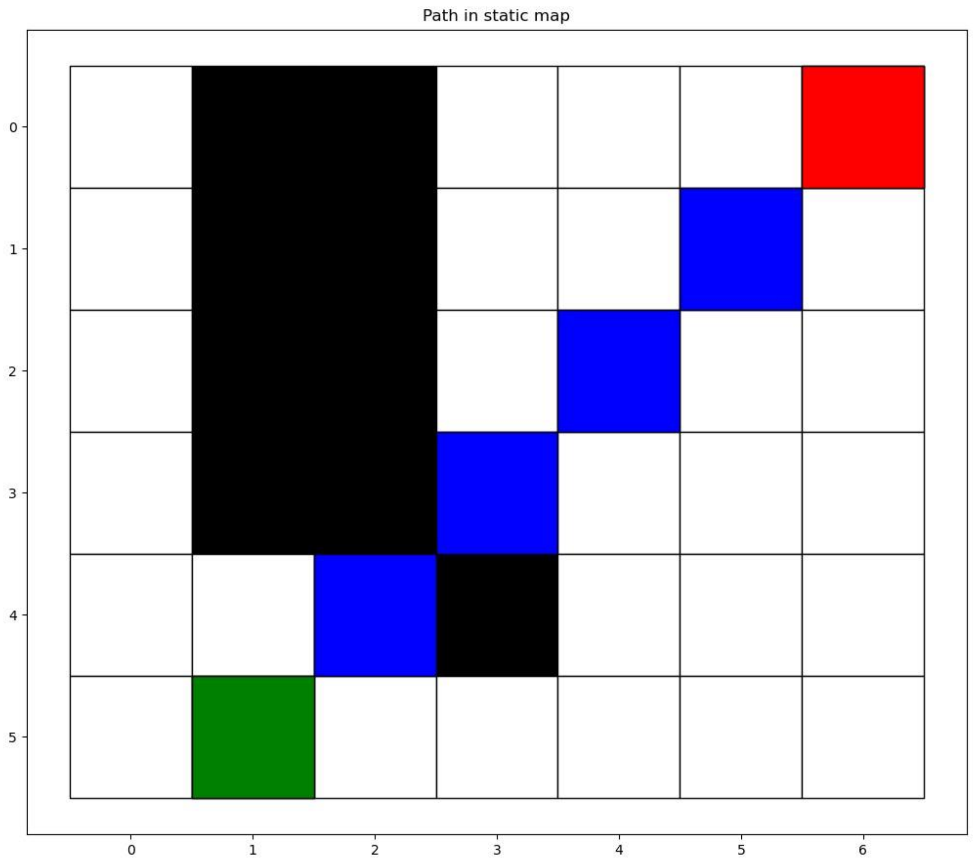

Path in progress
0
2
4
6
8
10
0
2
4
6
8
10

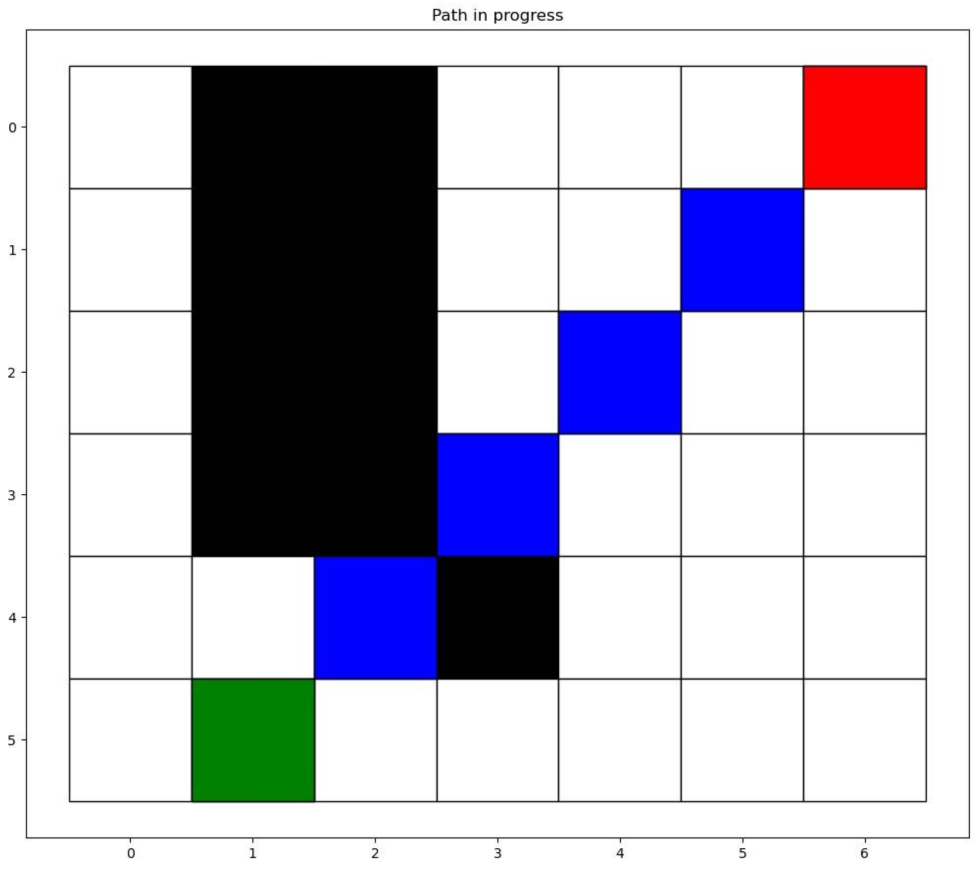

Path in progress
0
2
4
6
8
10
0
2
4
6
8
10

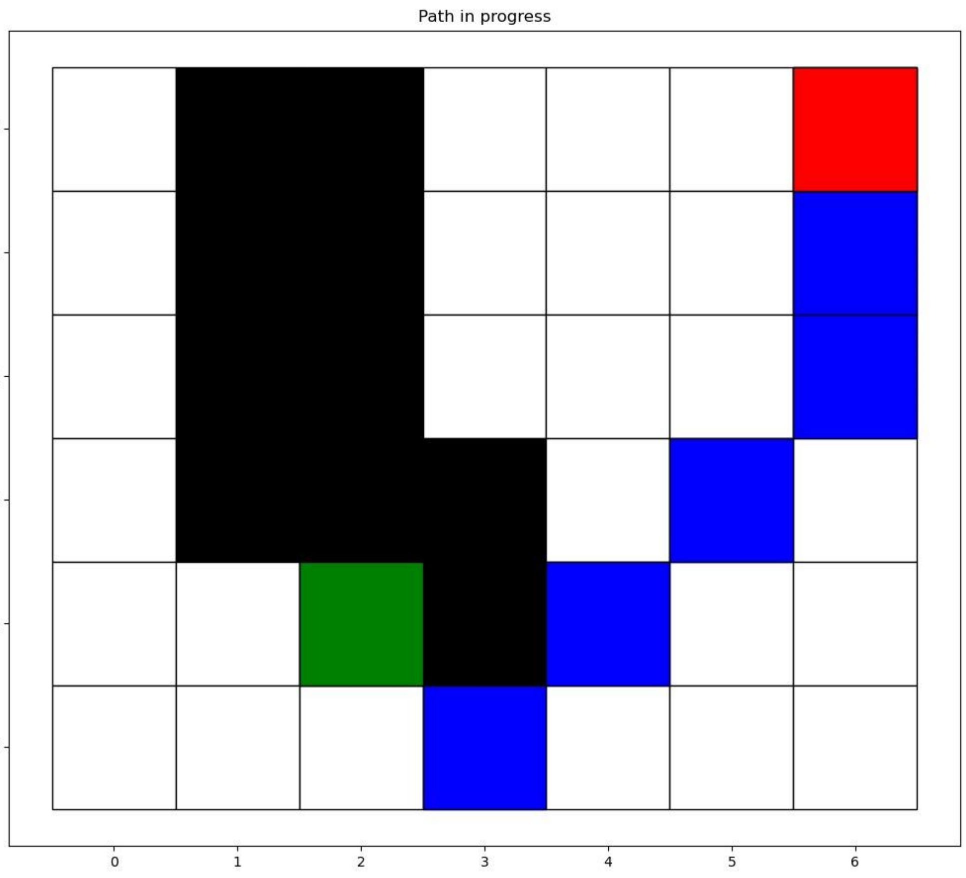
Path in progress

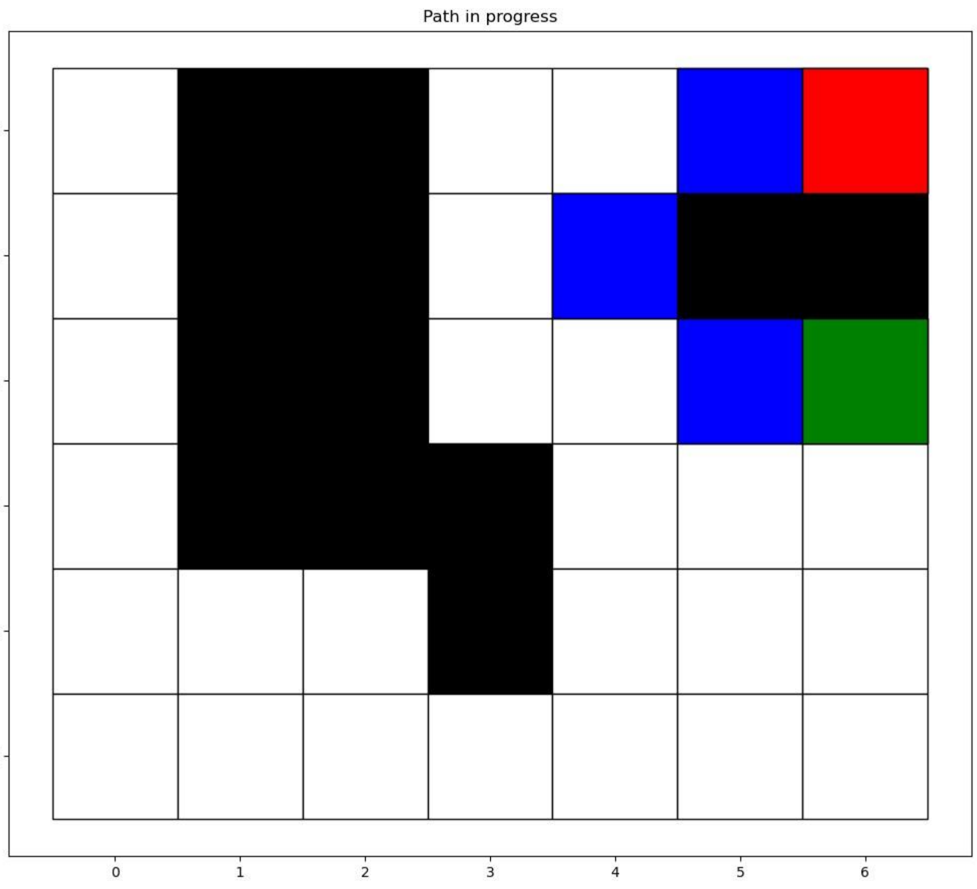


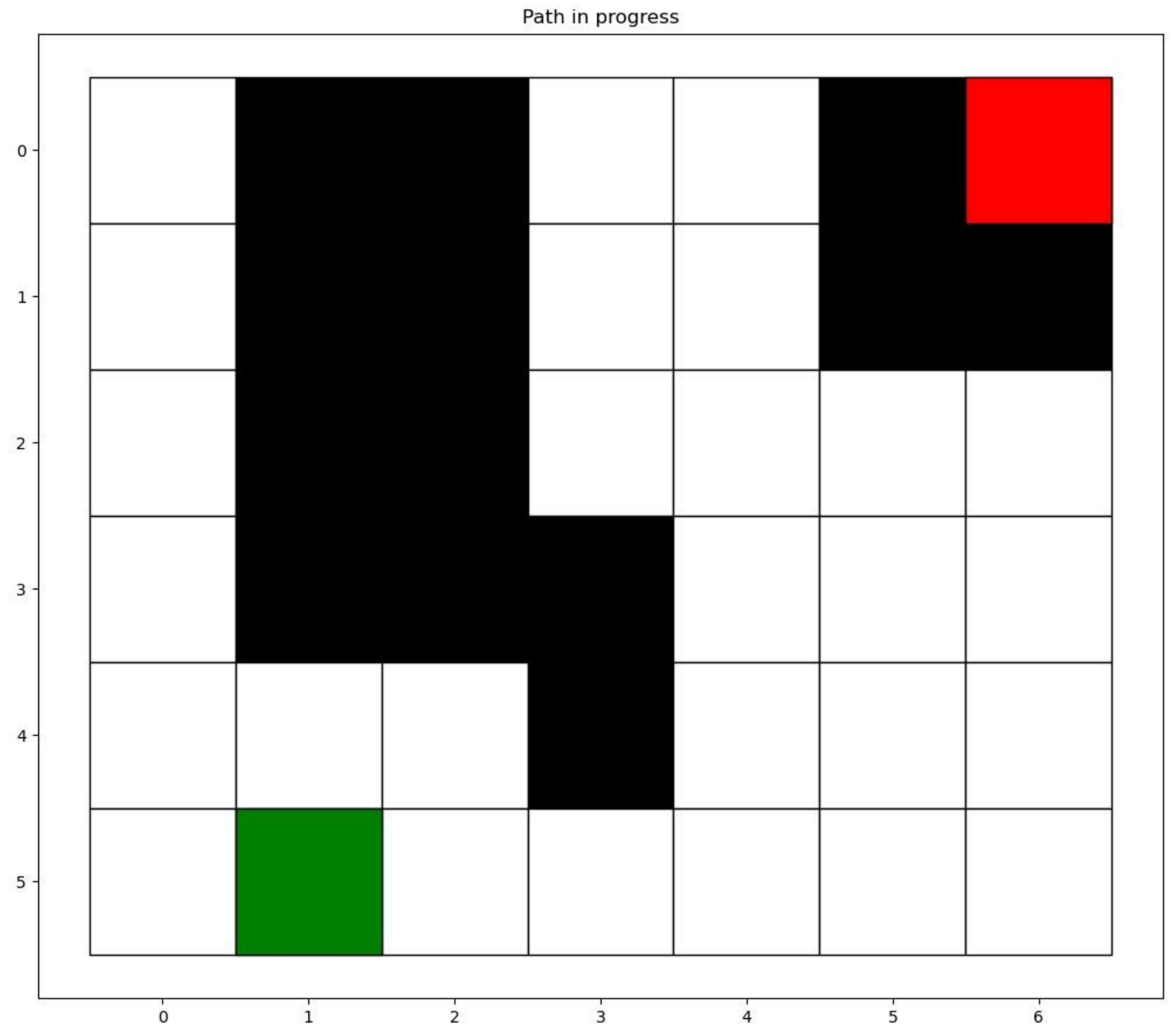
## MAP 3


Path in static map
0
1
2
3
4
5
0
1
2
3
4
5
6


Path in progress
0
1
2
3
4
5
0
1
2
3
4
5
6


Path in progress
0
1
2
3
4
5
0
1
2
3
4
5
6


Path in progress
0
1
2
3
4
5
0
1
2
3
4
5
6



END OF REPORT