Optimal Policy Generation for Doorkey Environement Using Dynamic Programming

Trivikram Choudhury

I. Introduction

Optimal policy generation and planning are key components of achieving in autonomous systems. An optimal policy is one which minimizes the cost associated with some actions taken by an agent over a timeperiod. Here, we discuss the motion of an agent in a `minigrid` environment, part of OpenAI's gymnasium suite of environments. Our agent is spawned at a predetermined location in each environment, and we have to find the optimal set of actions the agent should take to reach to the goal. Environments also have features like walls, doors and keys, and the agent must unlock a door with a key before it can cross it. Finding the optimal path is now also a function of more than just the position of the agent in the environment, and thus we redefine the the state variable. We model our state transitions as a deterministic Markov decision process (MDP), and once we have the dynamics and states of the environment, we use the dynamic programming algorithm to find the value function and optimal policy for each state at each time step. Given a start location, we follow the optimal policy to each subsequent state to find the optimal trajectory in the open loop case.

II. Problem Formulation

In a Doorkey environment, we are in a simplistic world model where we have an agent which can occupy some positions in a grid, and there are obstacles like doors and walls which the agent navigates around. The eventual mission is to reach the goal state, and once we do that, we claim that we stay there.

A. Agent Description and Objective

An agent is an entity which can occupy a position, has a forward direction, can take actions and can carry objects. When the agent reaches the goal, we say we have achieved our mission. **The objective of our exercise is to complete the mission in the shortest time possible.**

B. Grid Elements

Each grid location (i, j), where i is the x-offset for the position from the leftmost grid column and y is the y-offset from the top row, can be a one of 3 types of element - 1. Wall, 2. Door, 3. Floor

1) Wall - The wall is a location which cannot be occupied by any other object or agent. If the agent tries to move to a wall location, the agent cannot move and returns to the same location as it was in at the beginning of that action
2) Door - The door is a dynamic object, which can be either be open or closed. If the door is closed, it behaves like a wall. Each door has color. To open a closed door, the agent needs to possess something called a Key, that too of the same color. Once the agent has the key and opens the door, it behaves like a floor location.
3) Floor - A floor location is one which can be occupied by an agent or an object, like a Key. A floor location can also have special attributes, like being the goal.

Additionally there are other objects and attributes which the locations can have.

1) Key - A key object can either be placed on a floor location, or can be carried by an agent. A key has a color attribute, and can open doors of the same color. A floor location occupied by a key cannot be occupied by any other object or agent.
2) Goal - A floor location can be a goal, and once the agent reaches the goal, termination flag is raised.

C. Action Space

We know that our action space has 5 elements -

1) Move Forward (MF) - The agent tries to take a step in the direction it is facing. If possible, the agent moves to the new location, else it stays in the same place with no change in the environment. If the agent tries to move outside the grid. There is an error.
2) Turn Left (TL) - The agent's direction changes from the current one to a 90 degree turn to the left.
3) Turn Right (TR) - The agent's direction changes from the current one to a 90 degree turn to the right.

4) Pick Key (PK) - The agent tries to pick a key which it expects in the floor location it is facing. If there is a key there, the location now becomes devoid of a key and the agent is now carrying the key. If there is no key there, then there is no change in the agent or the environment.
5) Unlock Door (UD) - The agent tries to unlock a door in the floor location it is facing. If the agent has the key and the door is locked, then the door becomes open. Else there is no change in the environment.

III. Technical Approach

We recognize that out problem is one of optimal control. We want to reach the goal with the lowest cost (here, the cost is time).

We model our mission as a Markov Decision Process, by modelling our knowledge of our environment as a state, and creating a deterministic motion model between states based on the actions we can take.

A. Markov Decision Process

A Markov decision process (MDP) is a formulation where we change between states of a system with time depending on some actions. Each state-action pair generates a cost, and after some finite or infinite time T, the state we are at generates some termination cost. We also introduce a discount factor γ to discount costs incurred at a later time versus cost incurred sooner. An MDP is given by the tuple

$$
\begin{aligned}
&(\mathcal{X}, \mathcal{U}, p_0, p_f, T, \mathrm{l}, \mathrm{q}, \gamma) \\
\text{where } \mathcal{X} &= \text{ The state space (possible states)} \\
\mathcal{U} &= \text{ The action space (possible actions)} \\
p_0 &= \text{ The prior probability(Prior estimate of state, this is deterministic for each environment)} \\
p_f &= \text{ The transition probability (probability of reaching any state from a given action-state pair)} \\
T &= \text{ The horizon, could be a whole number (finite) or infinite} \\
\mathrm{l} &= \text{ The stage cost} \\
\mathrm{q} &= \text{ The terminal cost} \\
\gamma &= \text{ Discount factor}
\end{aligned}
$$

B. Known Environments

Here we see that our environment could be one of many, and for each environment we know the position and state of the door, the goal and the key. We can also see the state. Using this, we create a state definition as the tuple

$$
\begin{aligned}
x_t &= (p_t, d_t, k_t, door_t) \\
\text{where } p_t &= \text{ position of the agent in grid} \\
d_t &= \text{ Direction of the agent} \in \{0-right, 1-down, 2-left, 3-up\} \\
k_t &= \text{ Whether we are carrying key (1 if carrying, 0 ow)} \\
door_t &= \text{ Whether the door is locked (True if unlocked)}
\end{aligned} \tag{1}
$$

C. Random Environments

Here we see that our environment could be one of many, and for all environments we know the position and state of the doors, the goal and the keys. The initial position of the key and the goal are not fixed, and there are two door, any, all, or none of which could be open. We can only see the state after landing into the environment, with no knowledge a priori. We also have to use the superset of these environments as a common environment. Using this, we create a state definition as the tuple

$$
\begin{aligned}
x_t &= (p_t, d_t, k_t, door_t) \\
\text{where } p_t &= \text{ position of the agent in grid} \\
d_t &= \text{ Direction of the agent} \in \{0-right, 1-down, 2-left, 3-up\} \\
k_t &= \begin{cases} 0 \text{ if key is at (1,1)} \\ 1 \text{ if key is at (2,3)} \\ 2 \text{ if key is at (1,6)} \\ 3 \text{ if the agent is carrying the key} \end{cases} \\
door_{t,i} &= \begin{cases} True \text{ if } door_i \text{ is open} \\ False \text{ ow} \end{cases}
\end{aligned} \tag{2}
$$

We see that this complex state encodes all information needed by our MDP to find the optimal cost

D. Cost

For each action we take when we are not in the goal position, we consider our cost to be 1. After reaching our goal, any action we take will give a cost 0. The terminal cost of all states where the agent is not at the goal is ∞, and the terminal cost of all states where the agent is at the goal is 0. We notice that the environment raises an error whenever we take any action from a state at the edge of the grid, while the agent is facing outwards. We model this as a stage cost of ∞ as well, as this is essentially telling us that we have ended the MDP without reaching the goal. From here on, we can calculate the optimal value function at each timestep as

$$\begin{aligned} V_t^*(x) &= \min_{u \in \mathcal{U}} \mathrm{q}(\mathrm{x_T}) + \sum_{\tau=t}^{\tau=T-1} l(x_\tau, u_\tau) \\ \text{where } l(x_t, u_t) &= \begin{cases} 1 \text{ if } p_t \neq \text{ goal} \\ 0 \text{ ow} \end{cases} \\ \mathrm{q}(x_T) &= \begin{cases} 0 \text{ if } p_T = goal \\ \infty ow \end{cases} \end{aligned} \tag{3}$$

Here x_t is the state of the agent and environment at time t, u_t is the action taken at time t, $l(x_t, u_t)$ is the stage cost, and $\mathrm{q}(x_t)$ is the terminal cost we take by reaching state x_T at time t. We define an optimal policy as the following

$$\begin{aligned} \pi^* &= \arg\min_{\pi} V_0^{\pi}(x_0) \\ \pi(x) &\in \mathcal{U} \quad \forall x \in \mathcal{X} \end{aligned} \tag{4}$$

A key point to note is that the time horizon T here is $|\mathcal{X}| - 1$. As we can see the our problem is an optimal control problem now. Hence we use the dynamic programming method now.

E. Dynamic Programming

This is a method which takes a finite horizon optimal control problem, and tries to estimate the value function at each state at each timestp, starting at the termination and moving backwards. This is done using the following algorithm

Algorithm 1 Dynamic Programming Algorithm

Initialize $V_T(x) = \mathrm{q}(x) \forall x \in \mathcal{X}$
for $t \leftarrow (T-1)$ to 0 **do**
$Q_t(x, u) = l(x, u) + \mathbf{V_{t+1}}(\mathbf{f}(\mathbf{x}, \mathbf{u})), \quad \forall x \in \mathcal{X}, u \in \mathcal{U}$
$V_t(x) = \min_{u \in \mathcal{U}} Q_t(x, u) \quad \forall x \in \mathcal{X}$
$\pi_t(x) = \arg\min_{u \in \mathcal{U}} Q_t(x, u) \quad \forall x \in \mathcal{X}$

end for
return π, V_0

IV. Results

We divide the results into two parts, the open loop deterministic map case, and the open loop stochastic map case.

A. Known Map

We train our policy separately for each map using our known dynamics and dynamic programming function. As the state space is small, we calculcate the value function $V_t(x)$ at each time step $t \in [0, T]$, without checking for convergence. The code outputs the optimal sequence for each environment, which are enumerated in table 1(a).

We consider two scenarios of interest. The first, 'Doorkey-8x8-shortcut' (in fig 1a)we see that this route which could either go take the key or take the shortcut to the goal, and it chooses to take the path through the door, even minimizing the number of turns taken as each turn is also an additional cost.

The second case, 'doorkey-8x8-direct' (in fig 1b) shows a scenario where there is a much better route than the door, and the program exploits to to completely ignores the door and the key.

A thing to note is that in this scenario, we did not try to handcraft a motion model based on our knowledge of the dynamics of the environment, but rather we discovered the motion model by making the agent achieve each state possible, and pass actions as arguments to the environment to take a step, and polled the result as the next state.

Similarly, we didn't use our experience that the value function and policy become stationary, but created a dynamic value function and policy and policy which are time dependent. The result seems to have converged to the a correct result. Having

said that, we can expect that the time taken to achieve this population of the value function for each time-state pair could be done faster if we realize that the time is not a factor. This is done in the next section, where we leverage the speedup in the large statespace.

B. Random Map

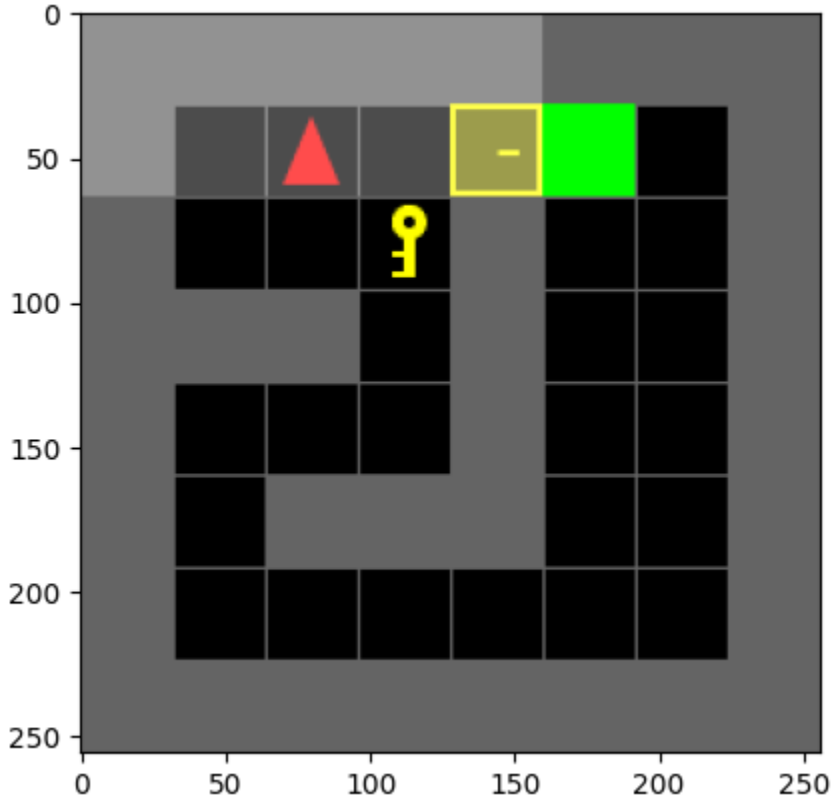
In the scenario where we cannot predict a priori which map we end up in, but all maps have a similar structure, we see that our optimized model achieves faster convergence than the deterministic case for the value function and policy calculation. We have found one common policy all maps by defining appropriate states. While creating the value function and policy, as they are stationary, we see that even though our state is large ($|\mathcal{X}|$) = 10800) we still converge within 20 timesteps, avoiding looping over $T = |\mathcal{X}| - 1$ states. Also, as our motion model didn't use the environment for every transition, we saved time which would have been used to first generate, then update and finally poll the environment by creating a much simpler motion model. This effectively shows that having a good (and cheap) simulator can and should be preferred over running experiments.

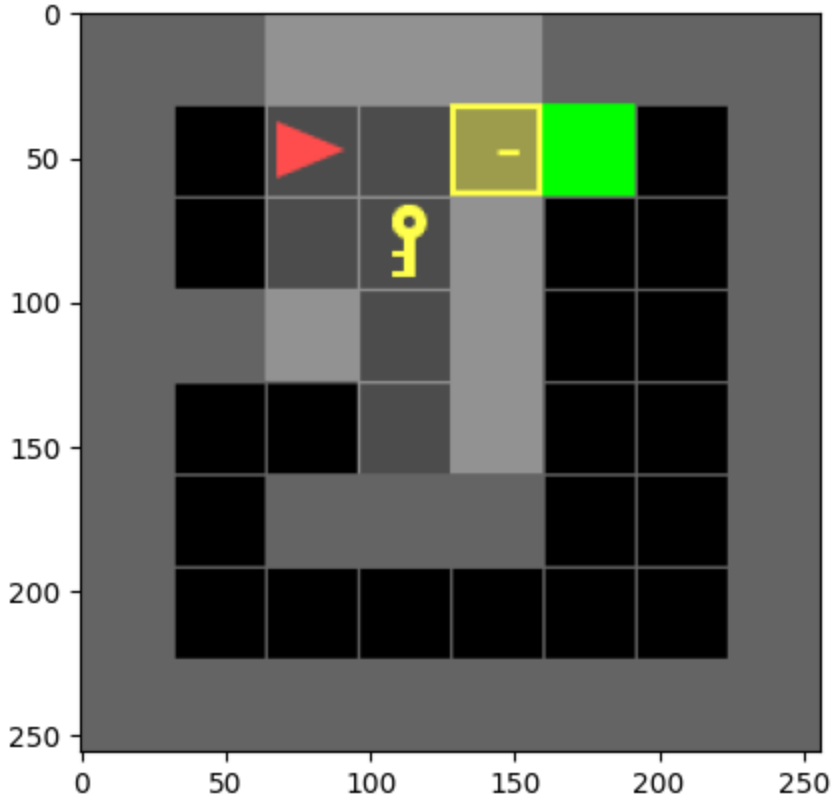
Some of the optimal paths are highlighted in figure 2. We see that in both these cases, the agent moves to the shortest path, whether that means moving towards the door or towards the key, depending on the state of the door and whether it is carrying the key or not. These results match our expected values.

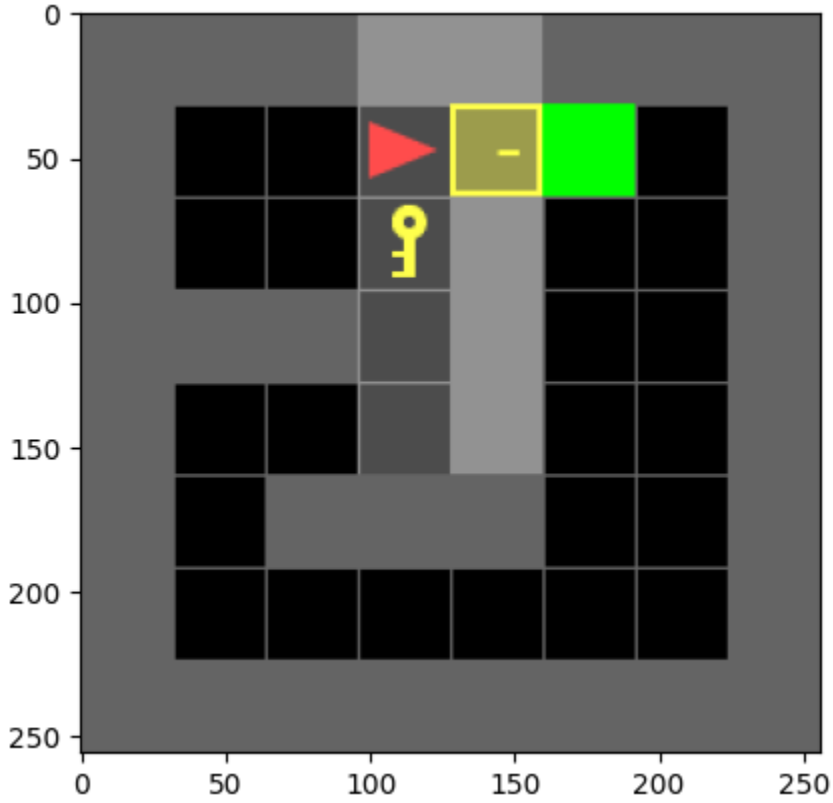
Figure 1 - Known Environments

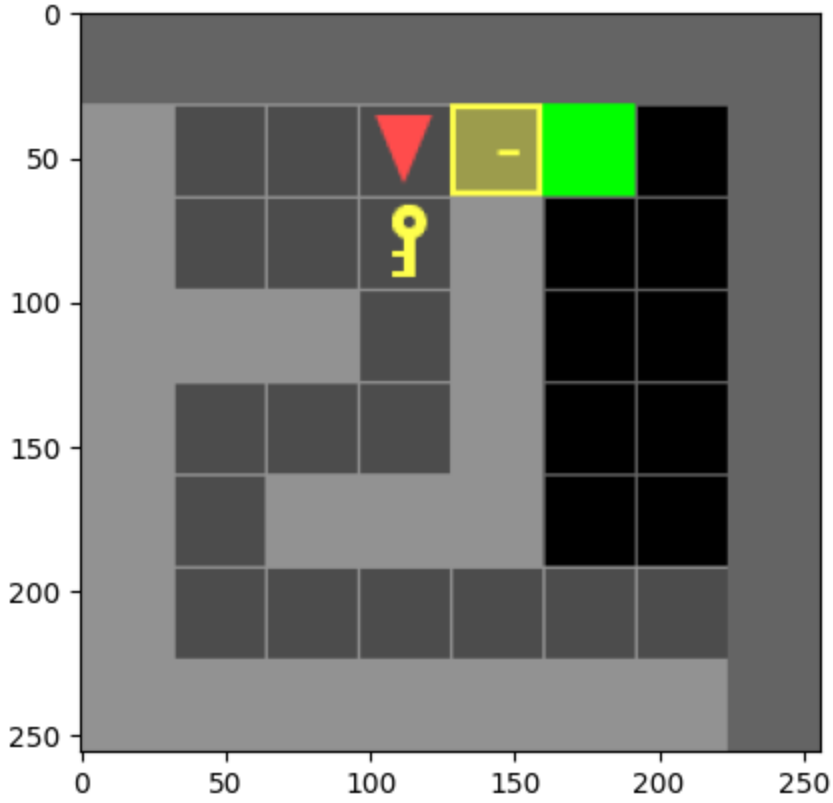
Doorkey-8x8-shortcut

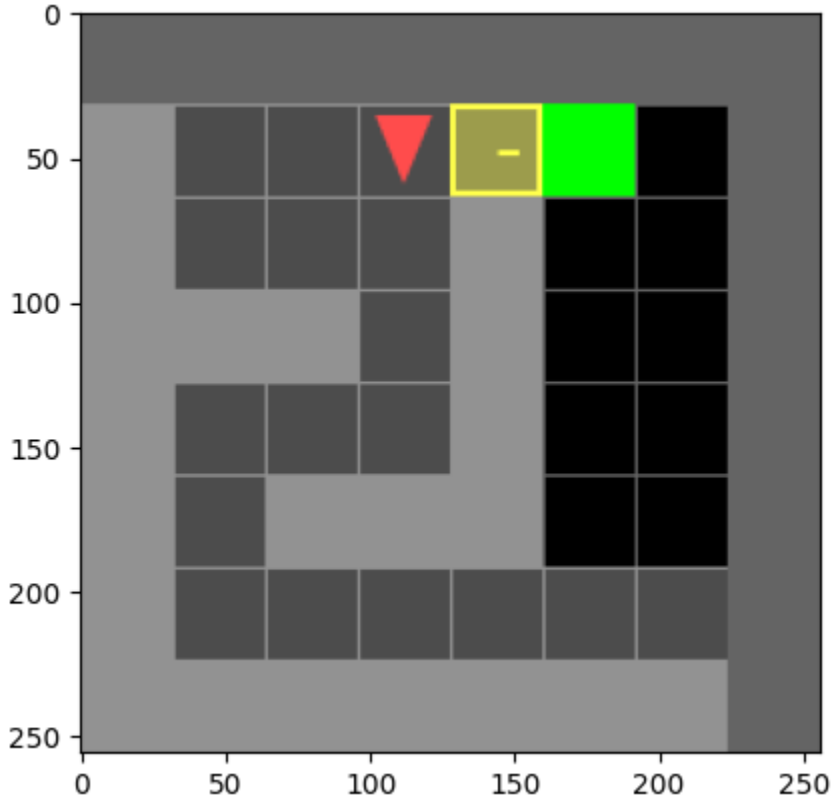
Optimal Sequence - [2, 0, 2, 3, 1, 4, 0, 0]

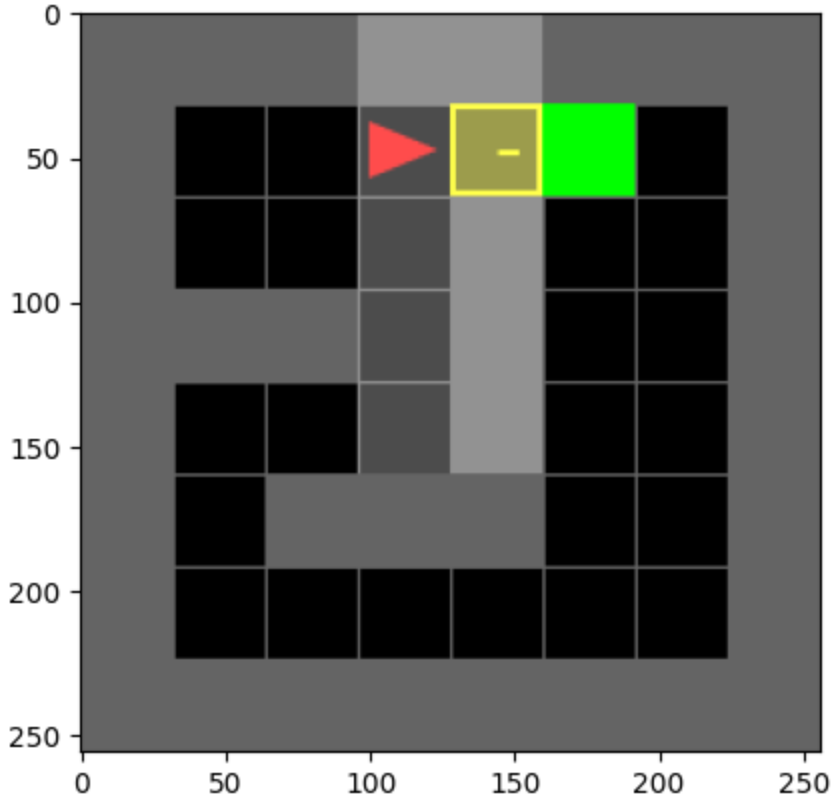


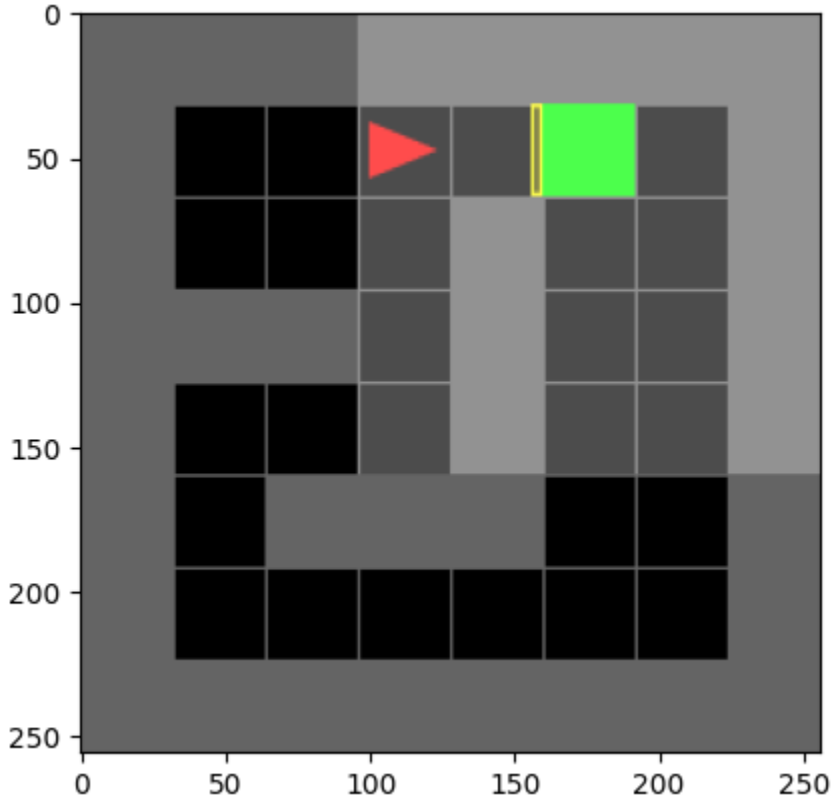

0
50
100
150
200
250
0
50
100
150
200
250

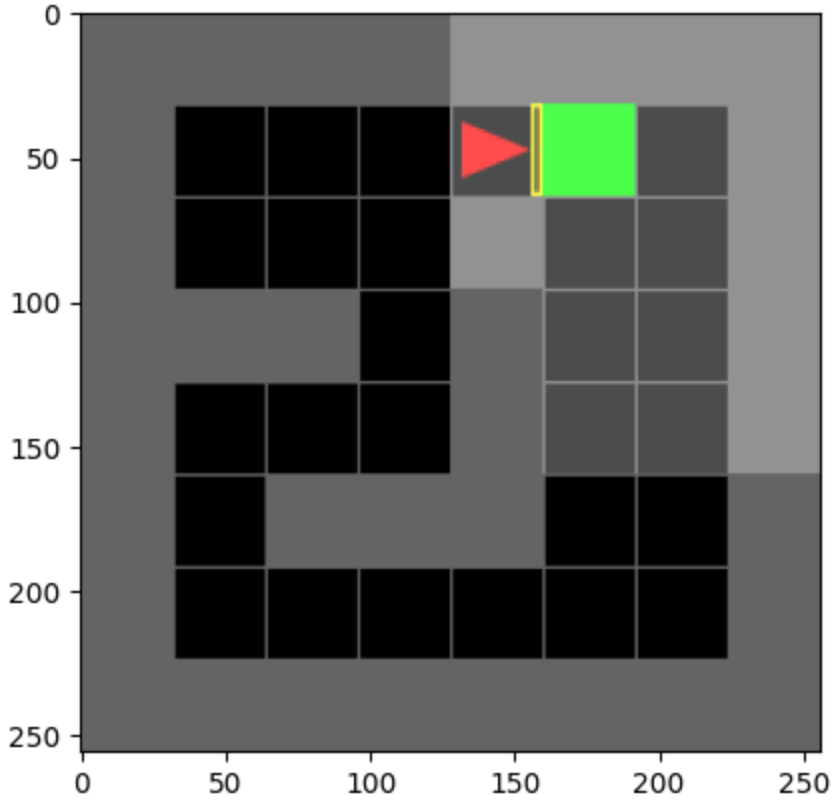

0
50
100
150
200
250
0
50
100
150
200
250

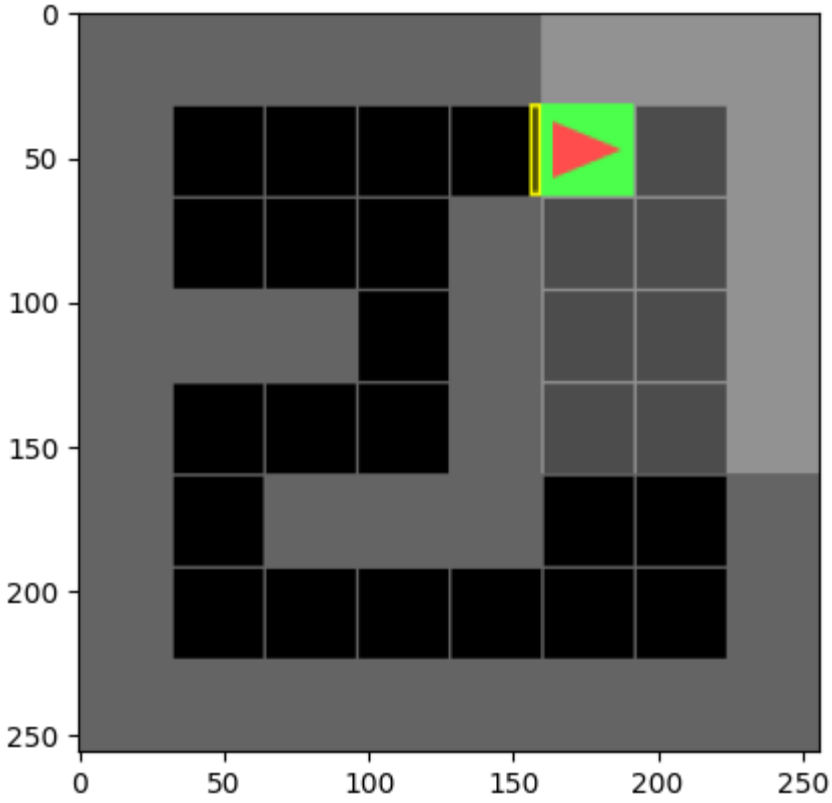

0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250

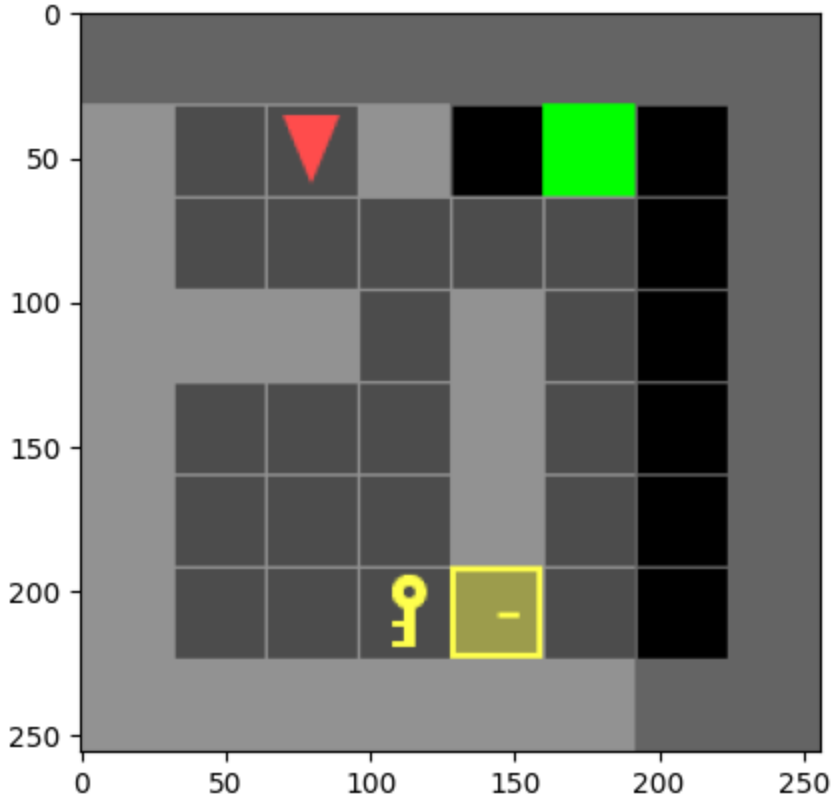

0
50
100
150
200
250
0
50
100
150
200
250

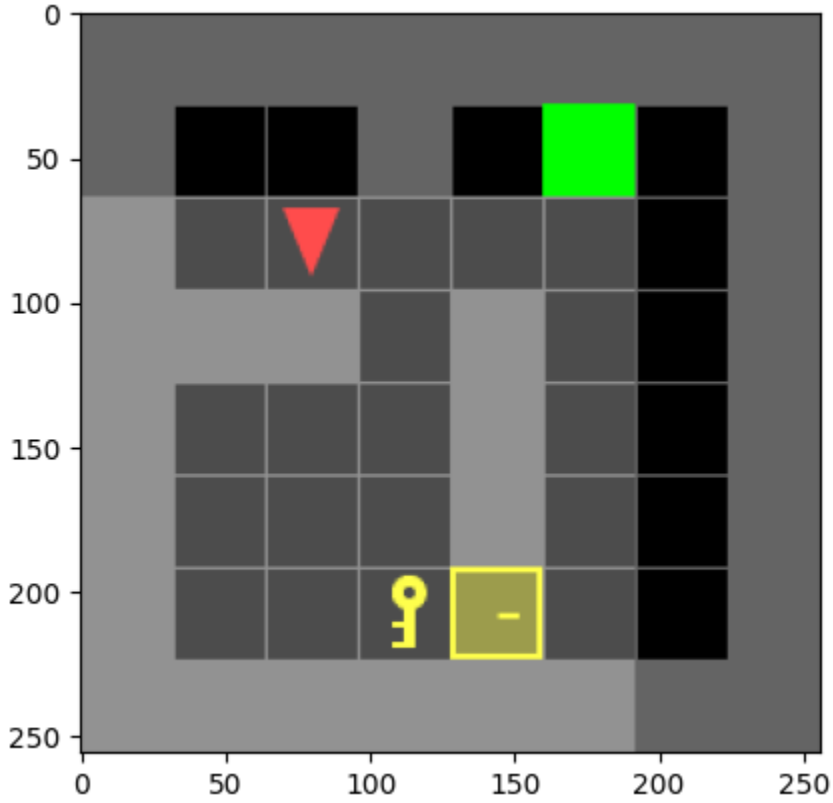

0
50
100
150
200
250
0
50
100
150
200
250

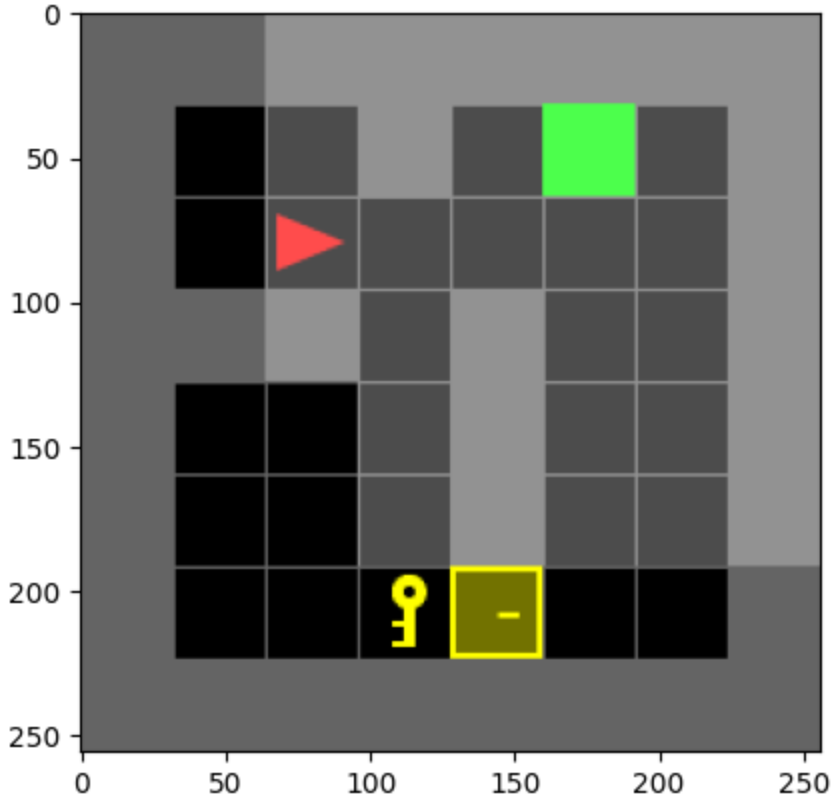


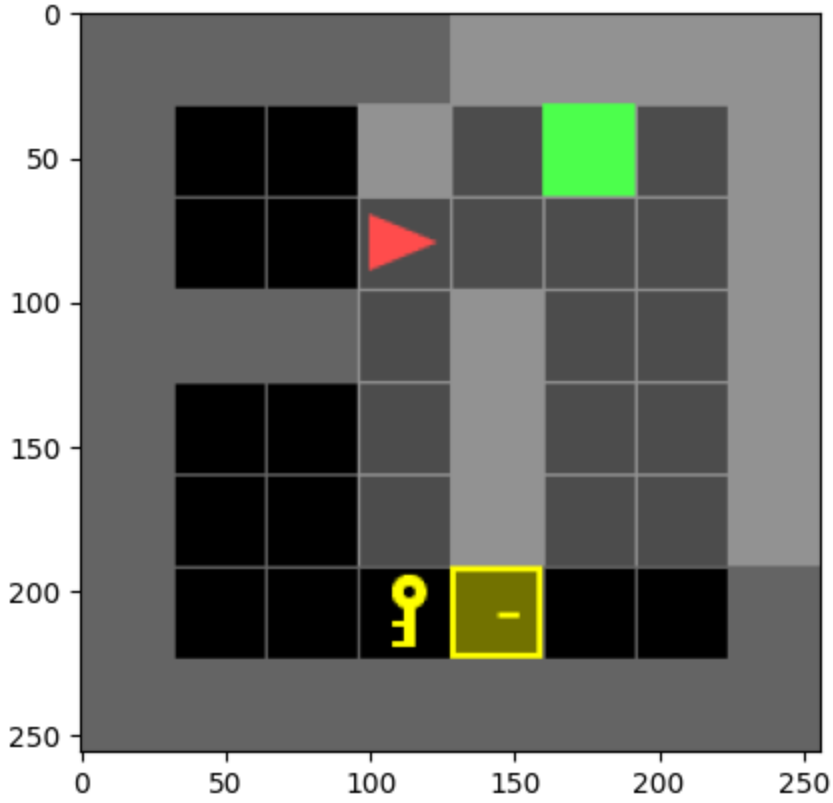
Doorkey-8x8-direct

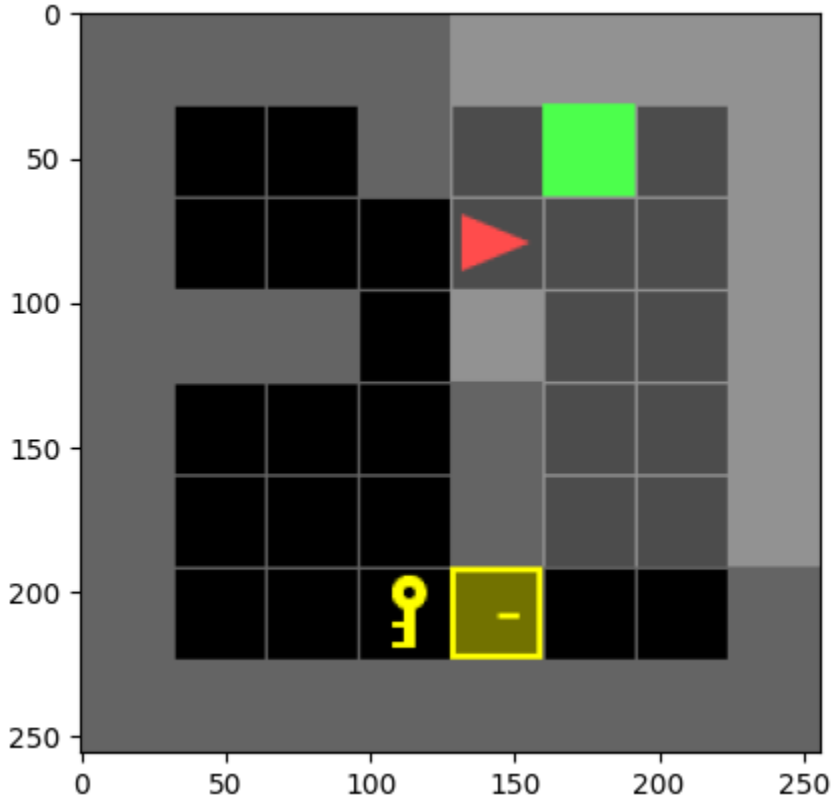
Optimal Sequence - [0, 1, 0, 0, 0, 1, 0]

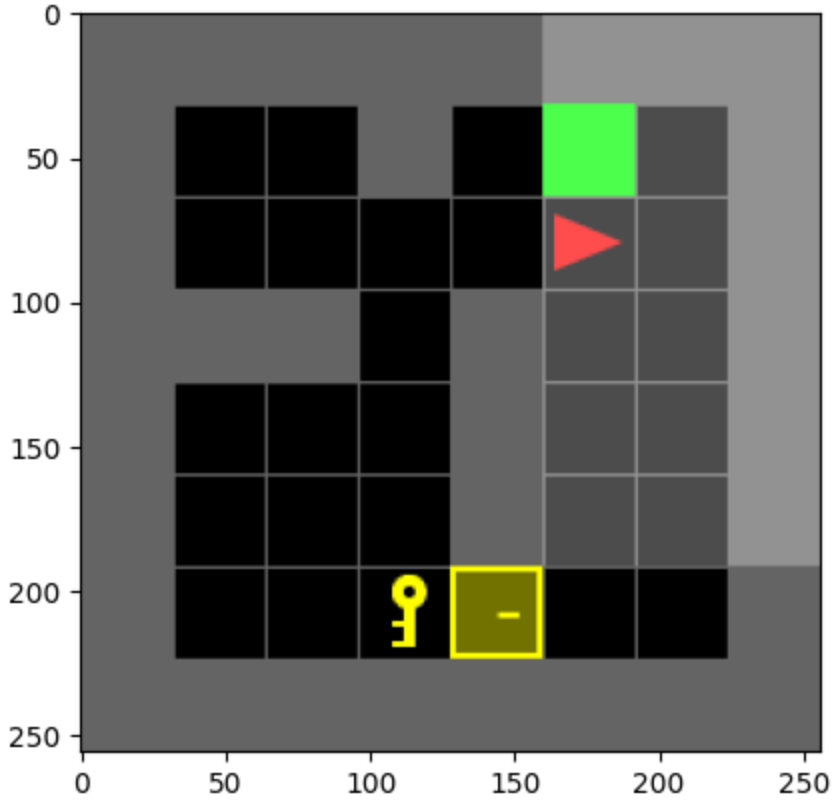

0
50
100
150
200
250
0
50
100
150
200
250

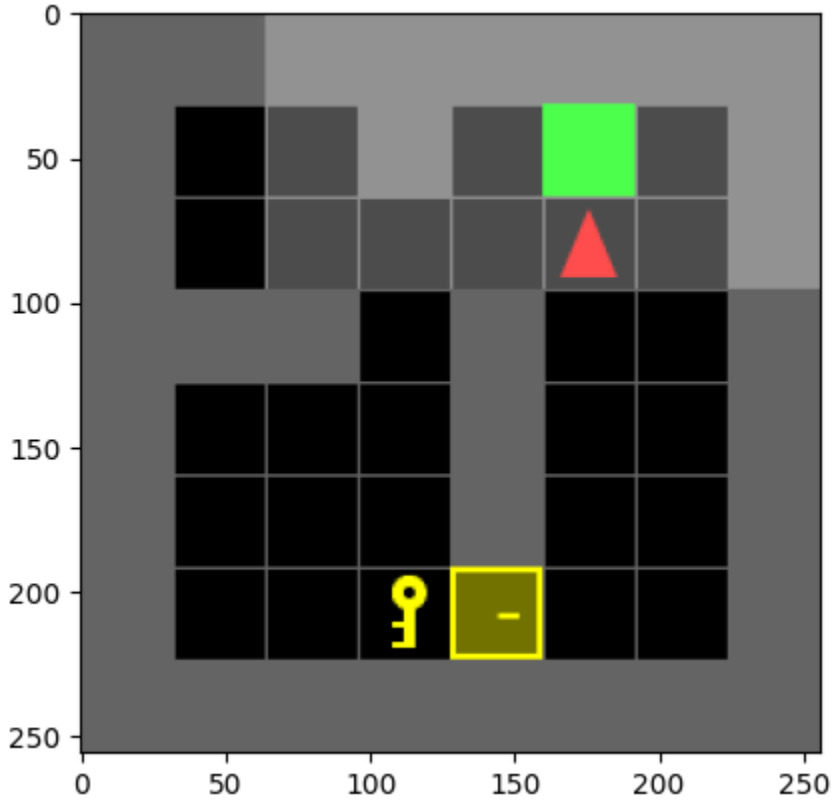

0
50
100
150
200
250
0
50
100
150
200
250

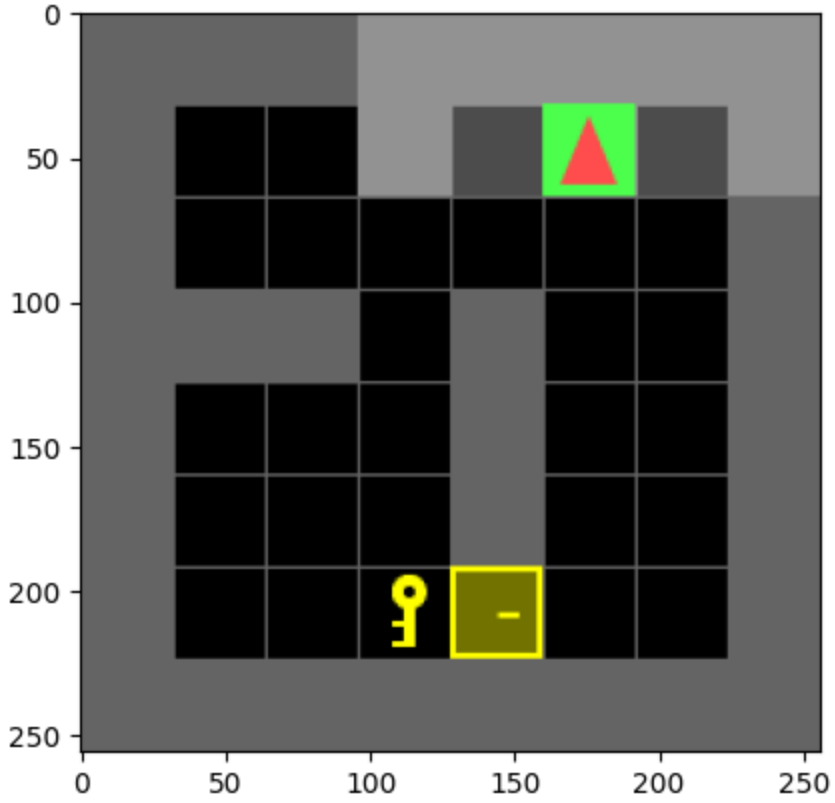

0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250

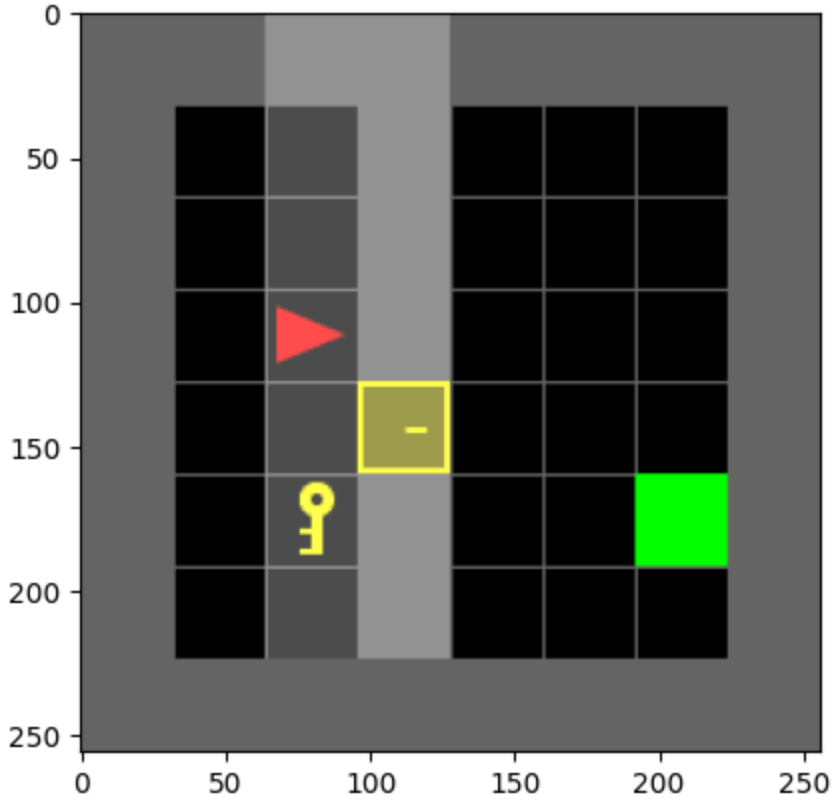

0
50
100
150
200
250
0
50
100
150
200
250

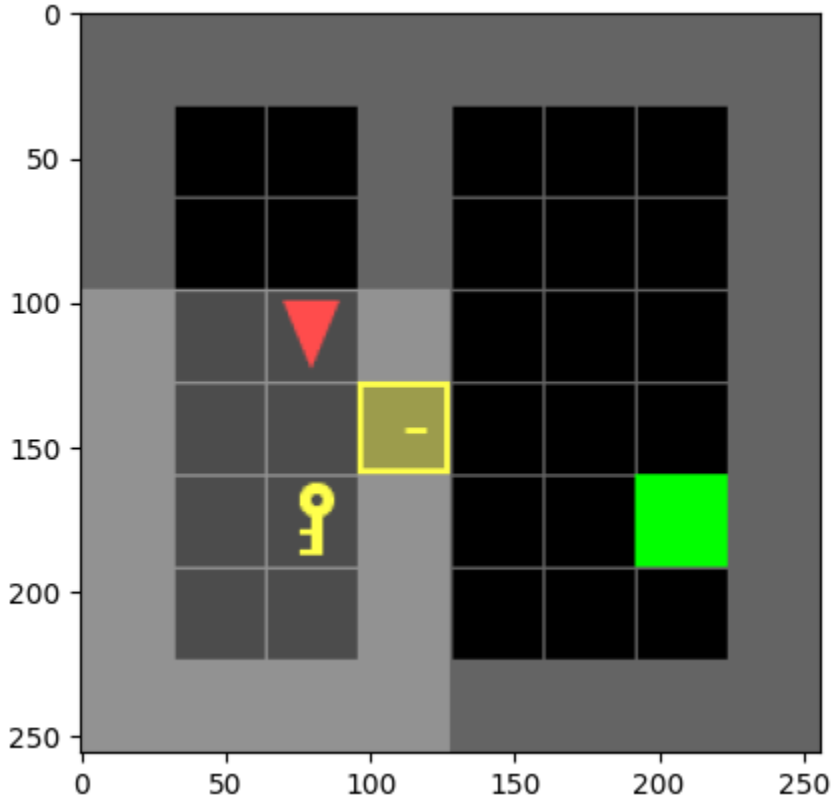

0
50
100
150
200
250
0
50
100
150
200
250

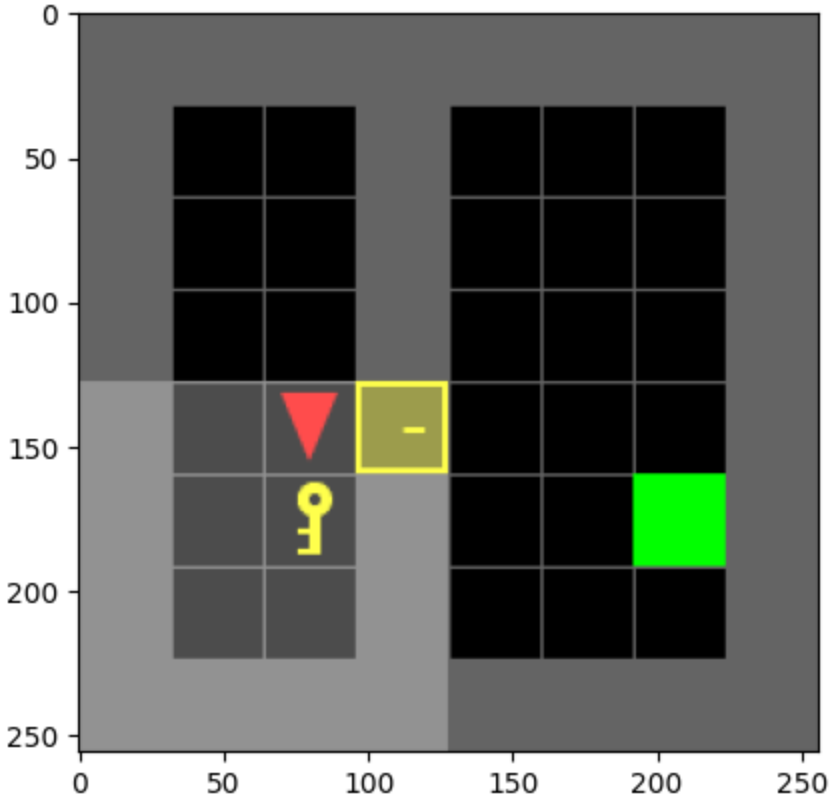


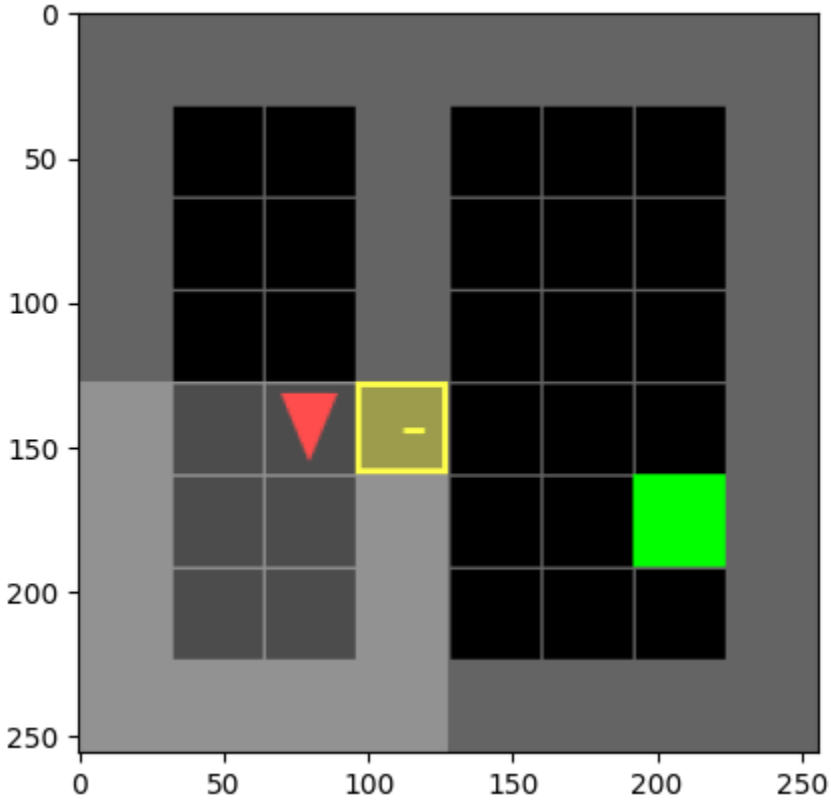
Example-8x8

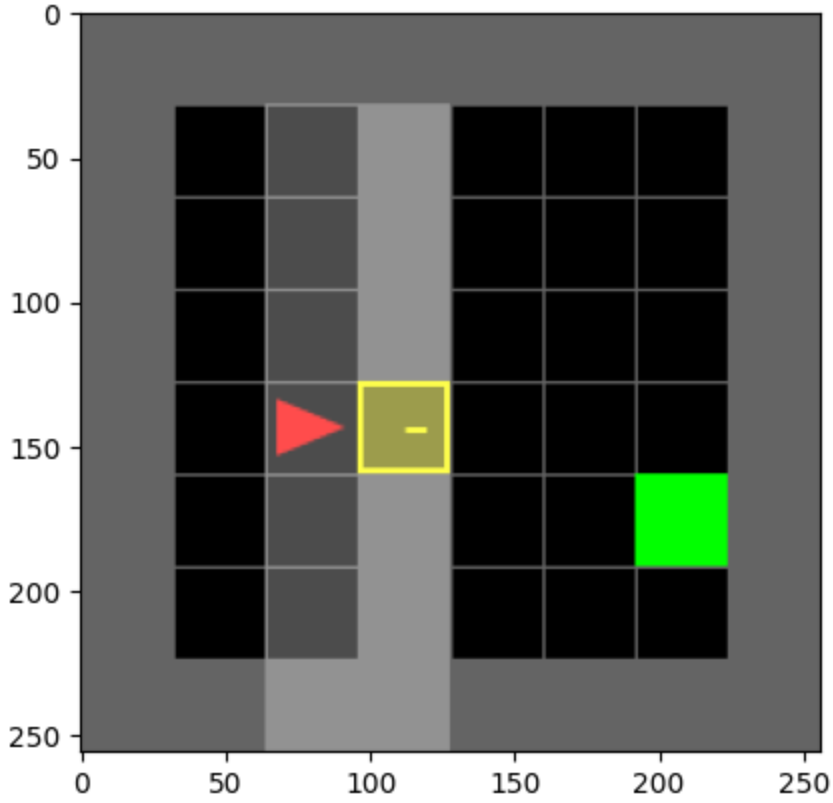
Optimal Sequence - [2, 0, 3, 1, 4, 0, 0, 0, 0, 2, 0]

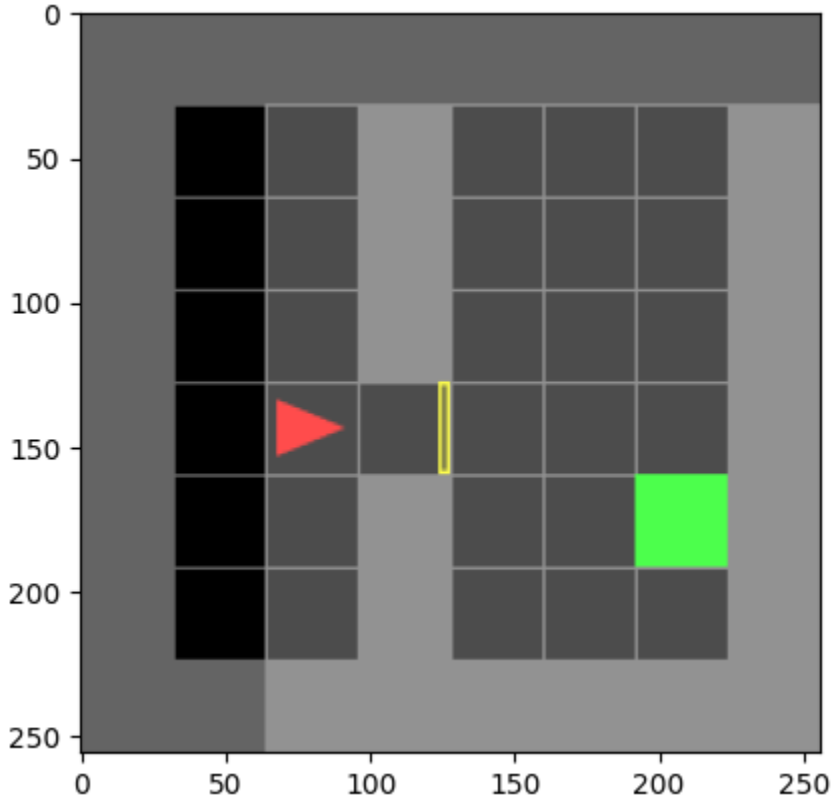

0
50
100
150
200
250
0
50
100
150
200
250

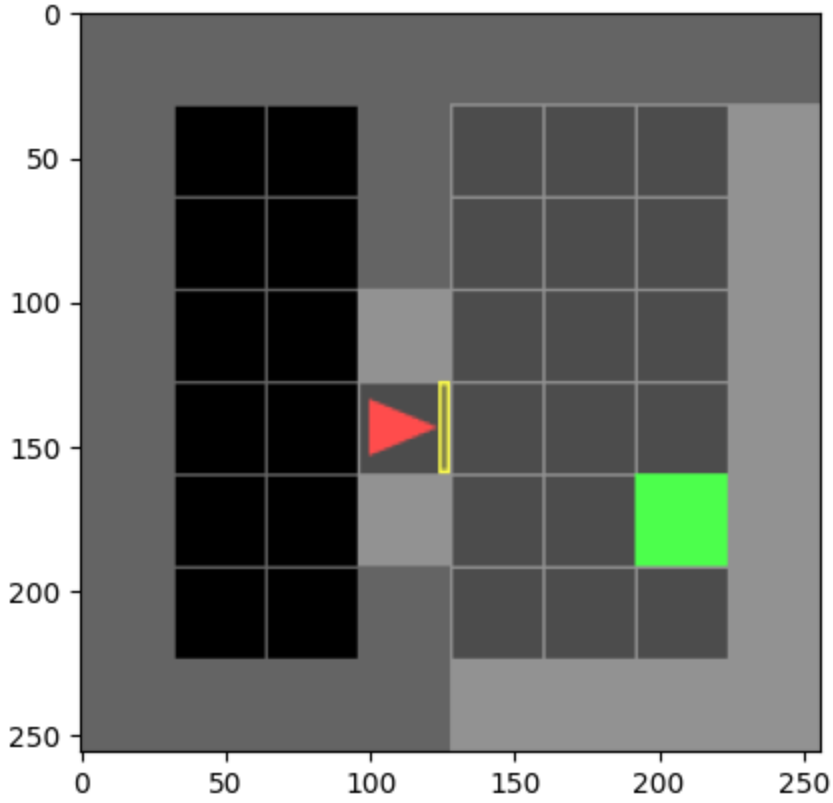

0
50
100
150
200
250
0
50
100
150
200
250

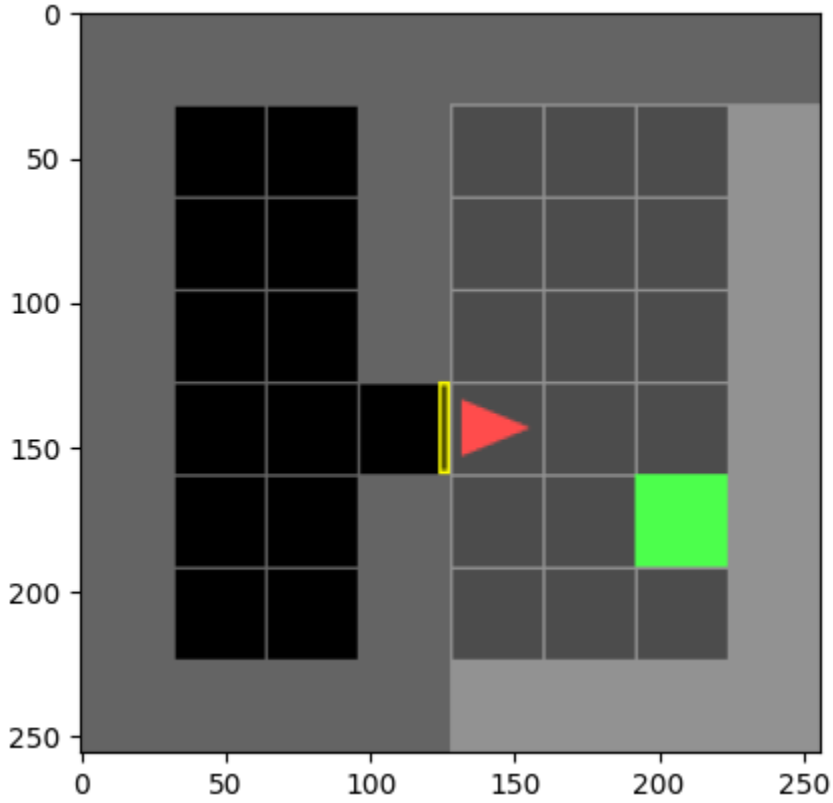

0
50
100
150
200
250
0
50
100
150
200
250

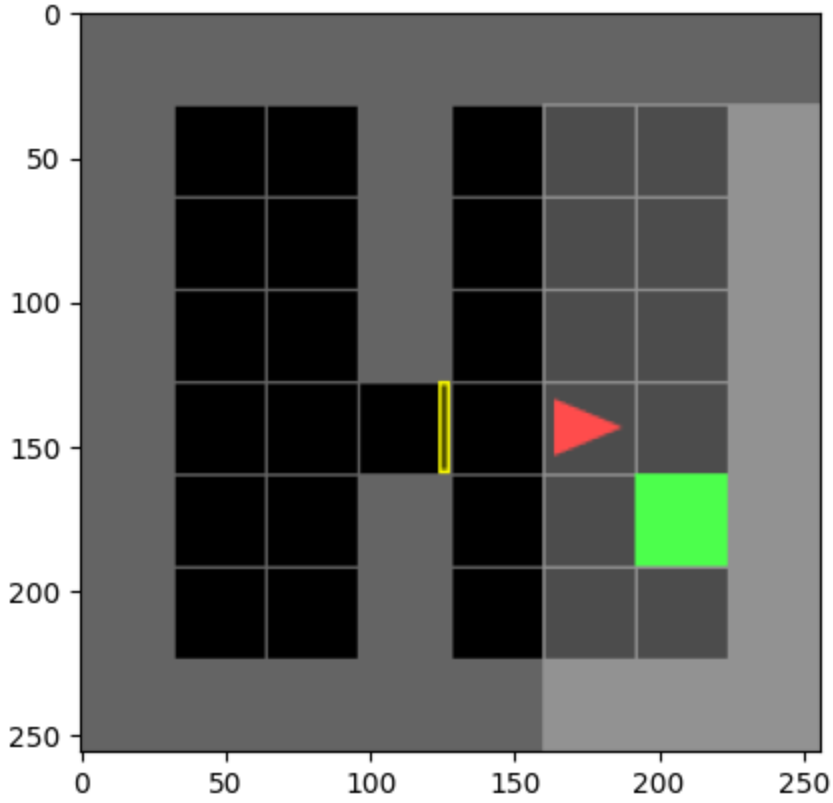

0
50
100
150
200
250
0
50
100
150
200
250

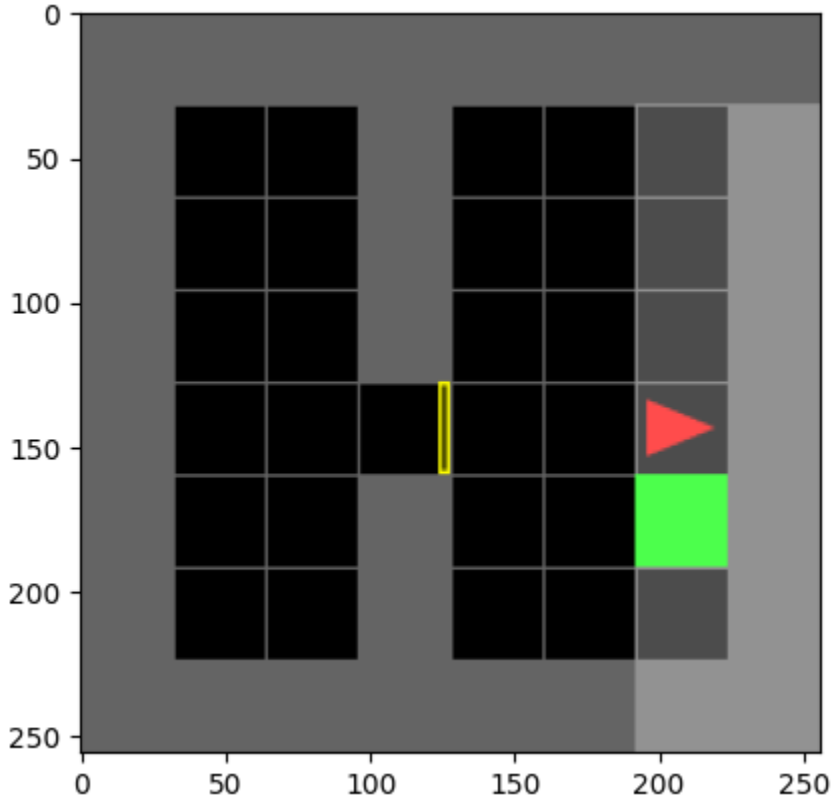

0
50
100
150
200
250
0
50
100
150
200
250

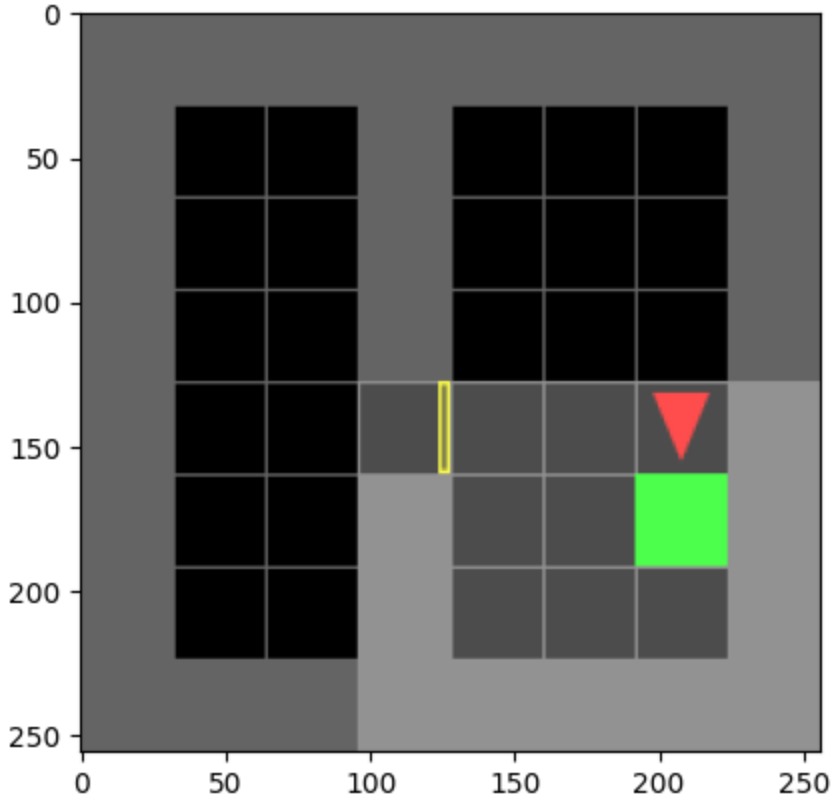

0
50
100
150
200
250
0
50
100
150
200
250

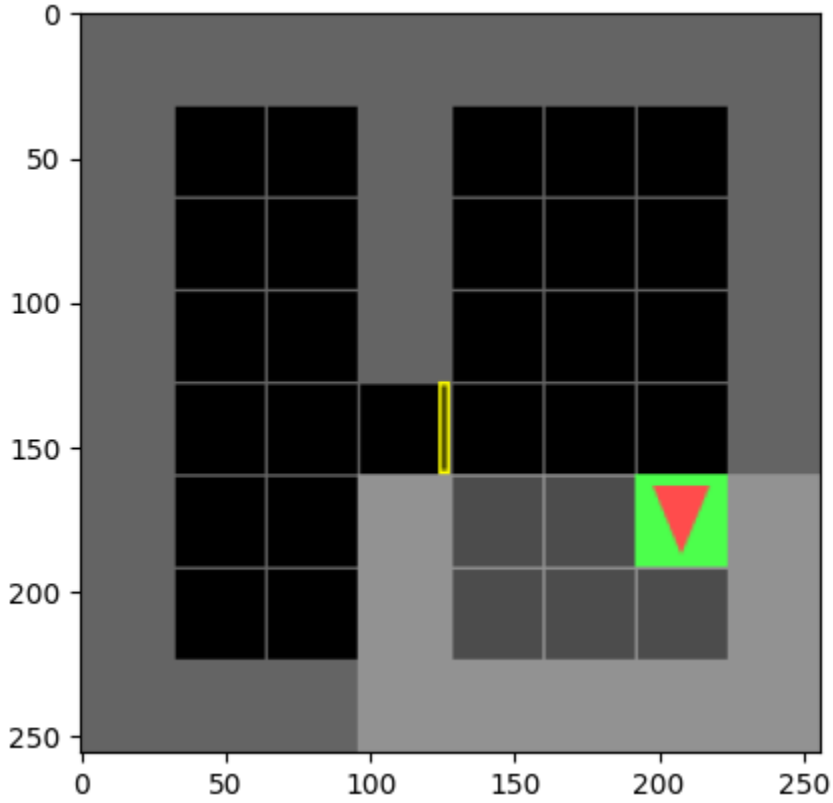

0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250

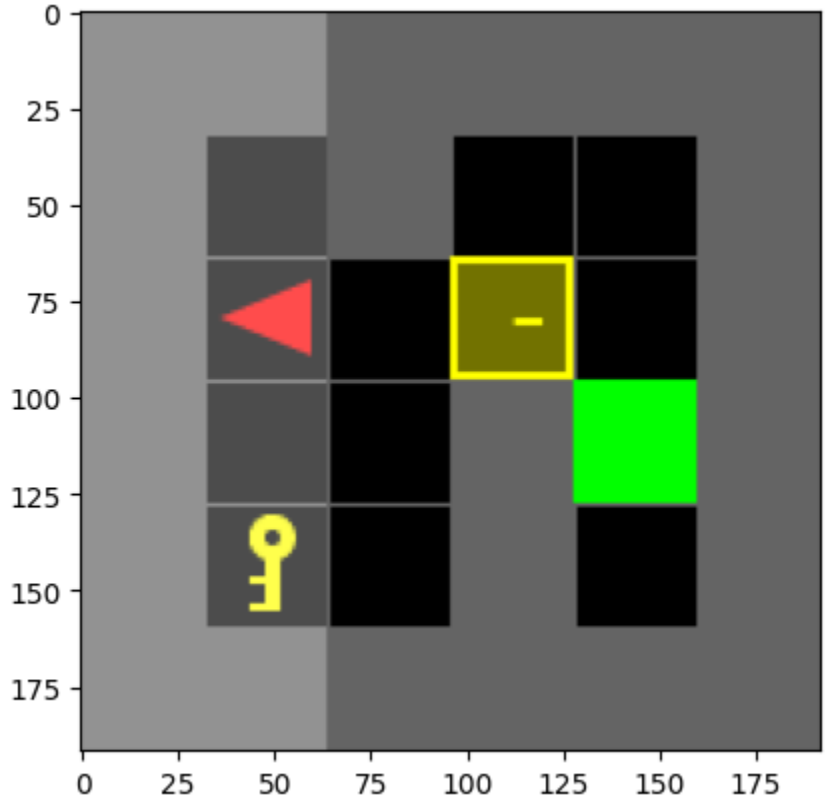

0
50
100
150
200
250
0
50
100
150
200
250

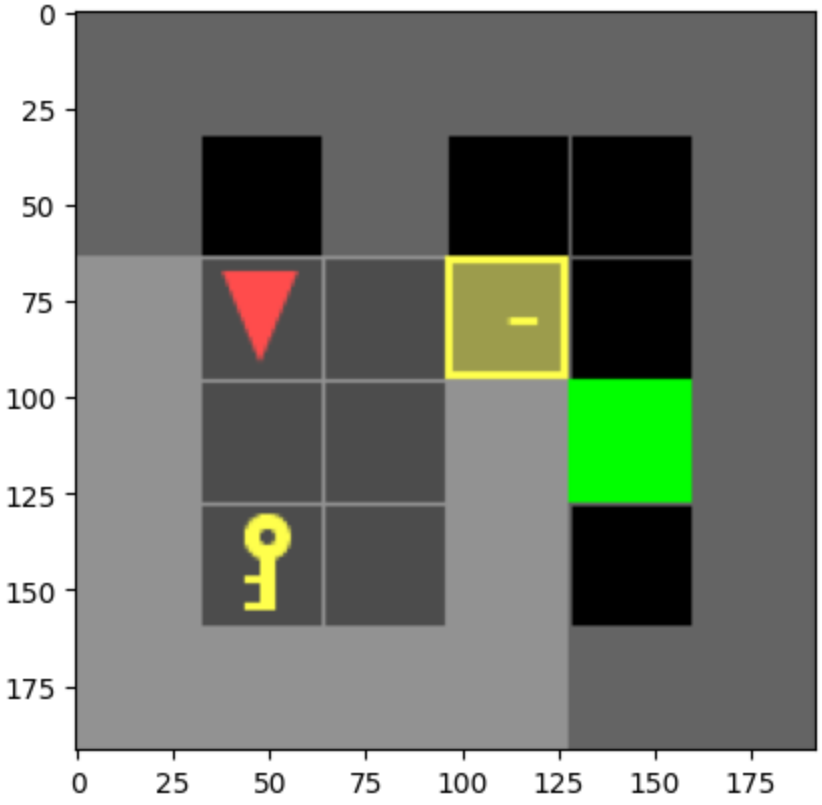

0
50
100
150
200
250
0
50
100
150
200
250

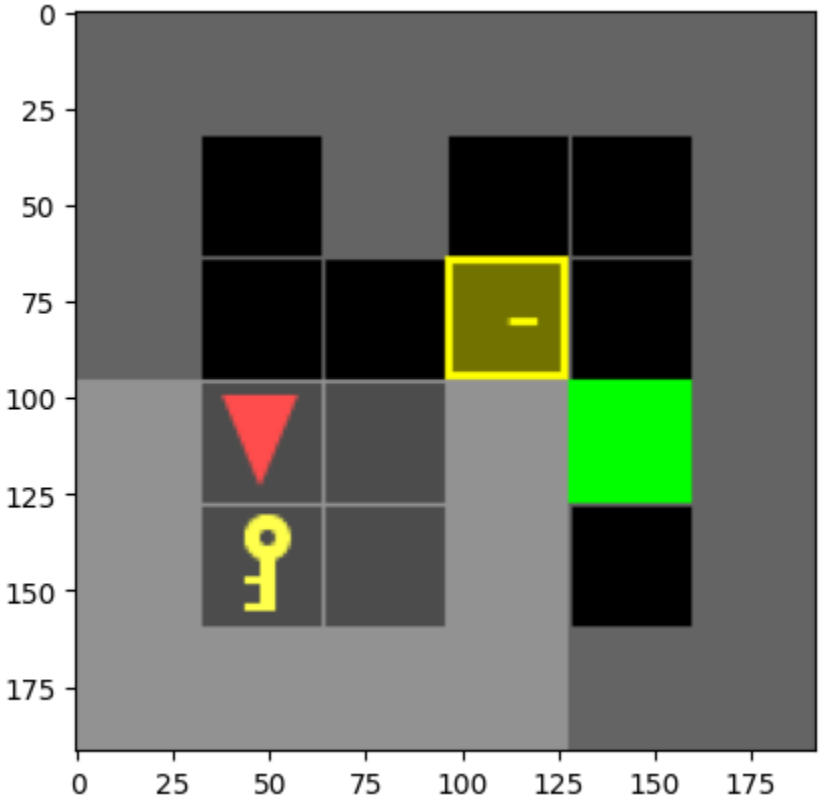

0
50
100
150
200
250
0
50
100
150
200
250

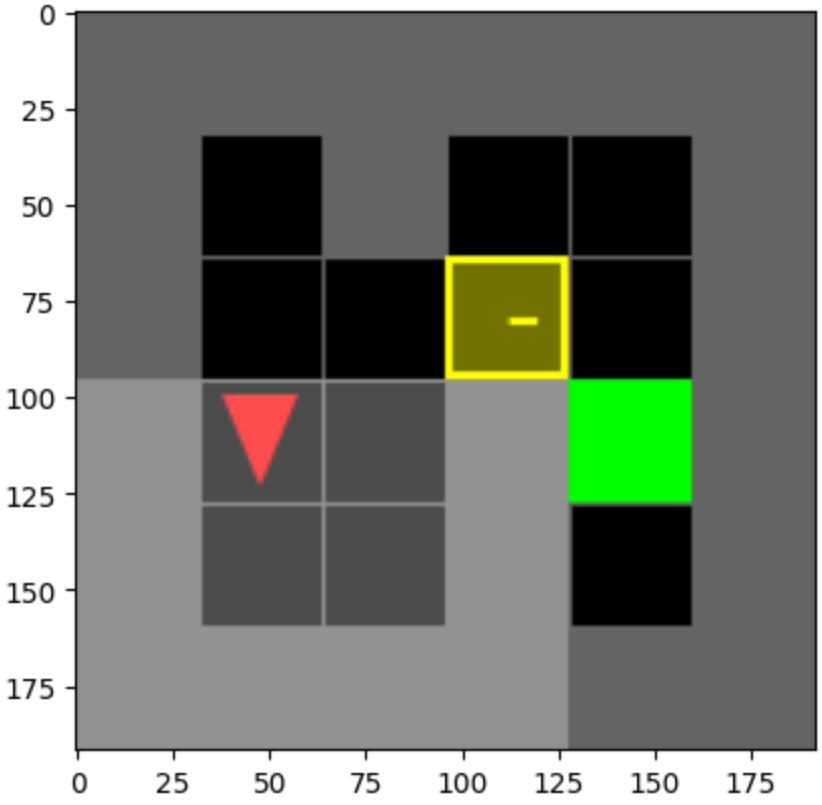
Doorkey-6x6-normal

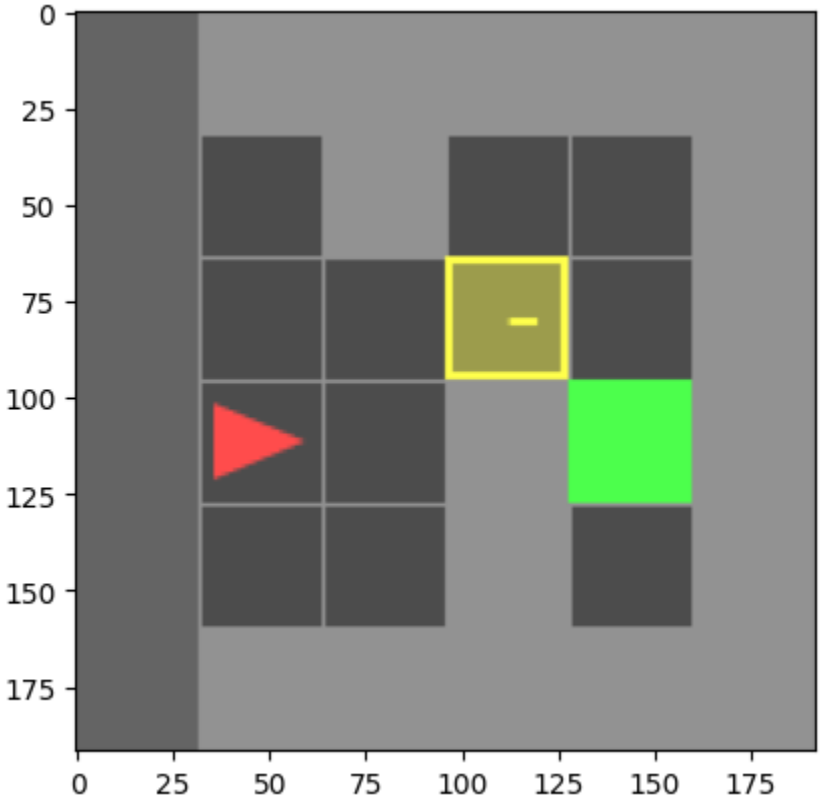
Optimal Sequence - [1, 0, 3, 1, 0, 1, 0, 2, 4, 0, 0, 2, 0]

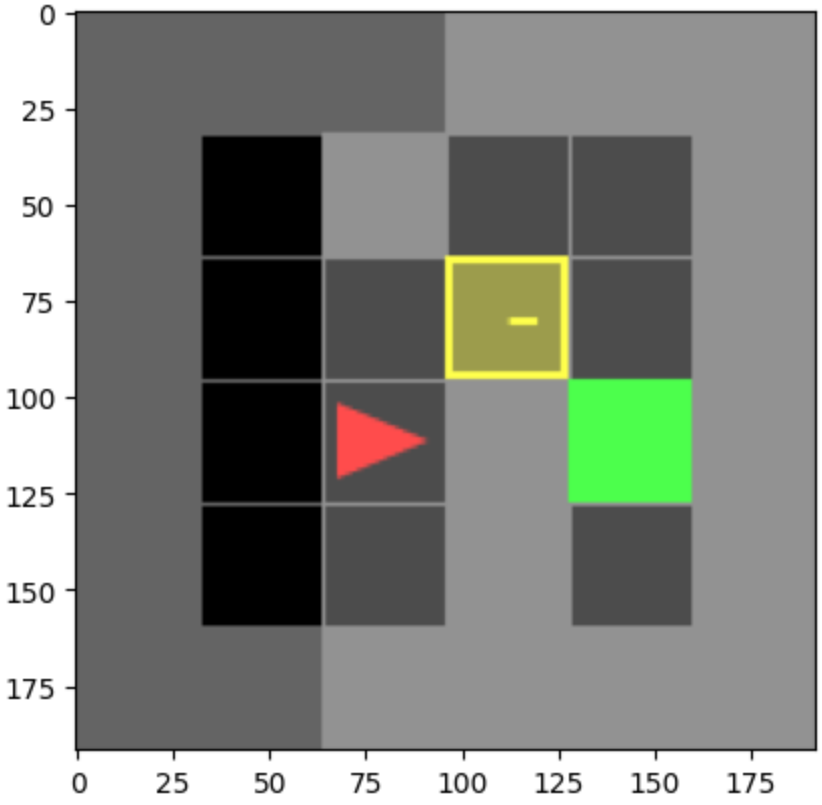


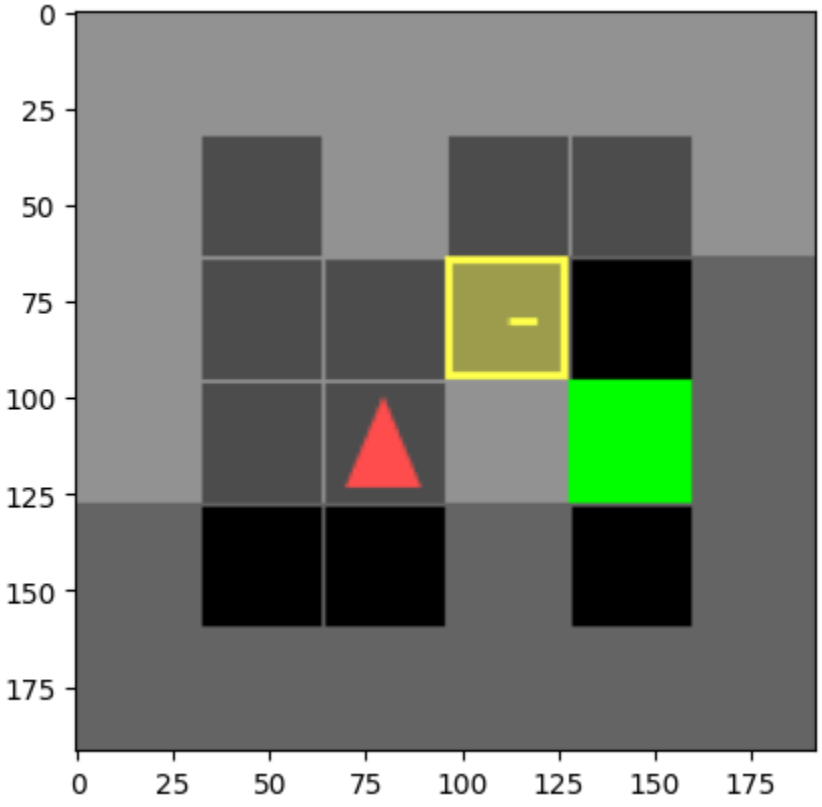

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

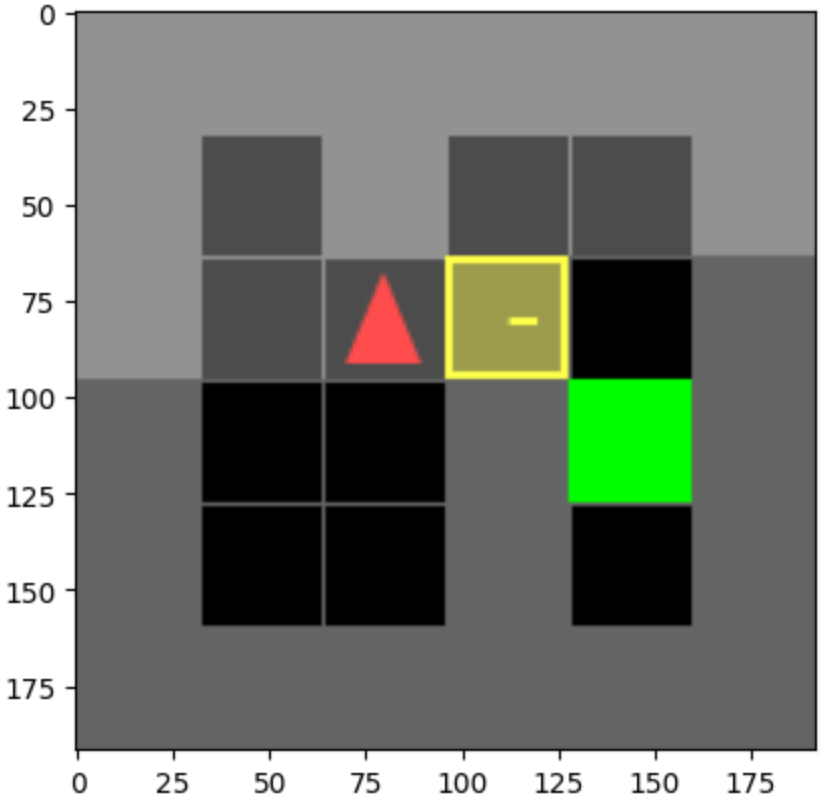

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

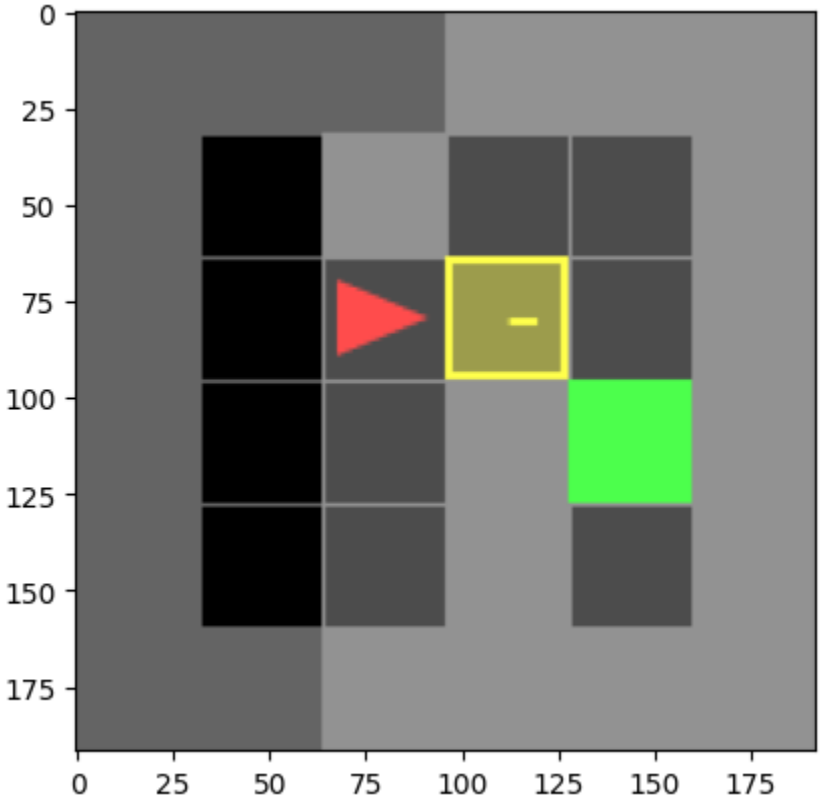

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

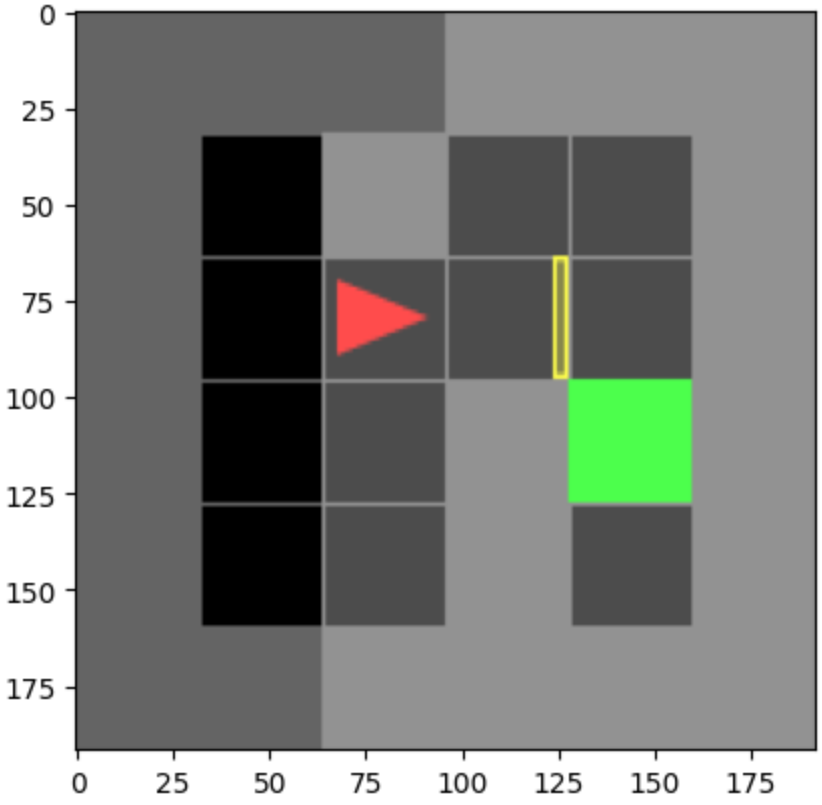

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

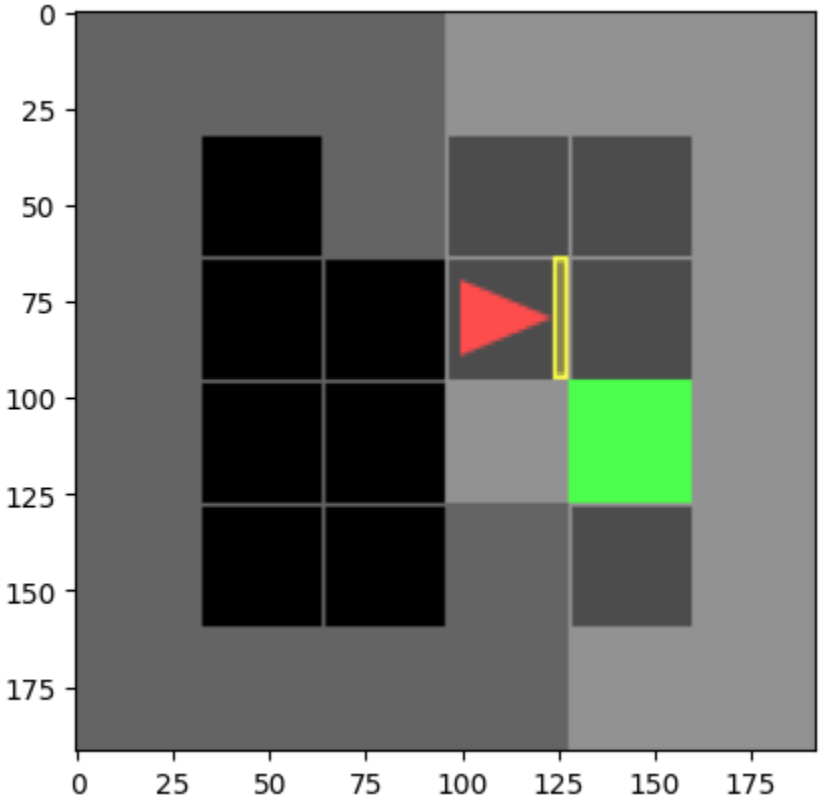

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

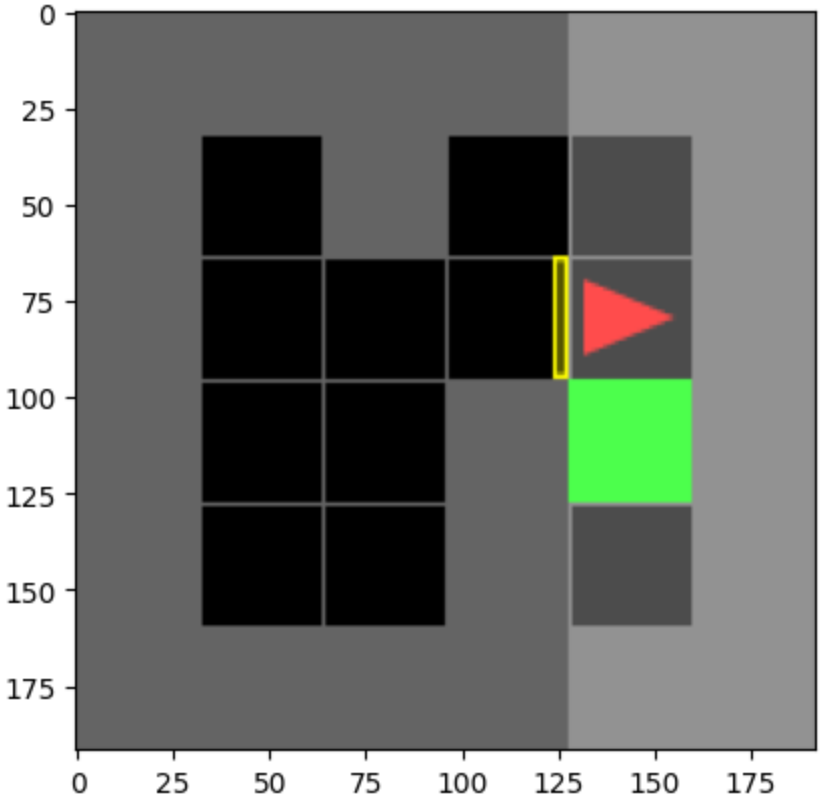

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

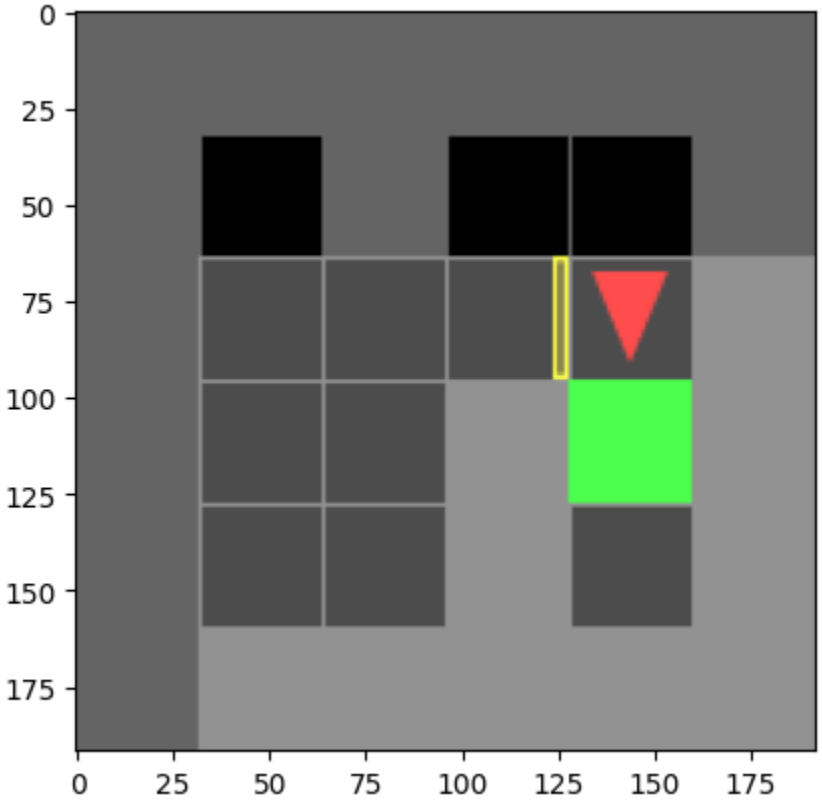

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

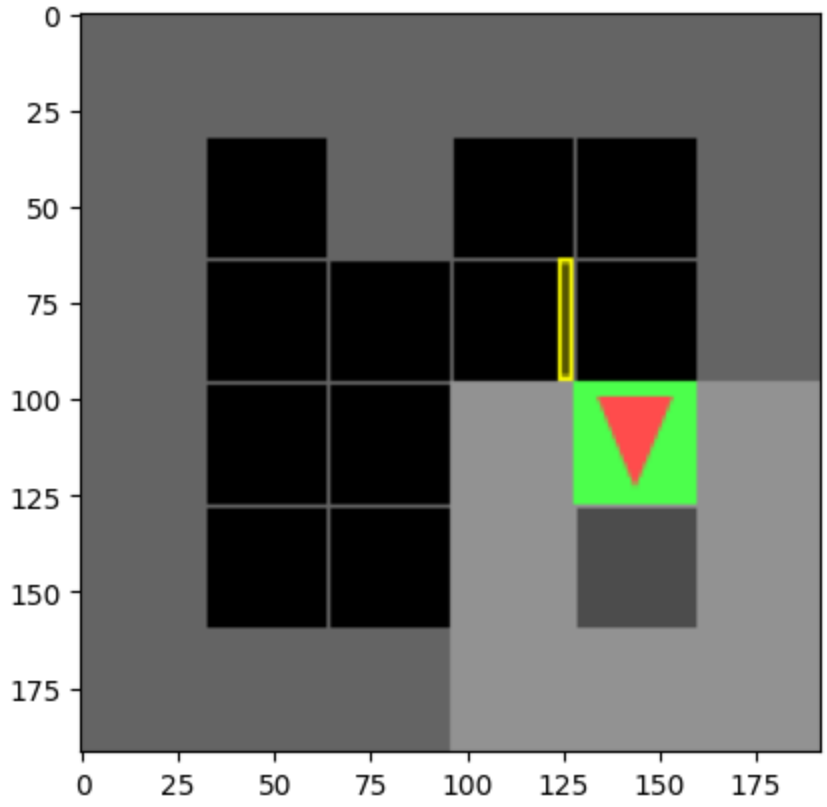

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175


0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175


0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

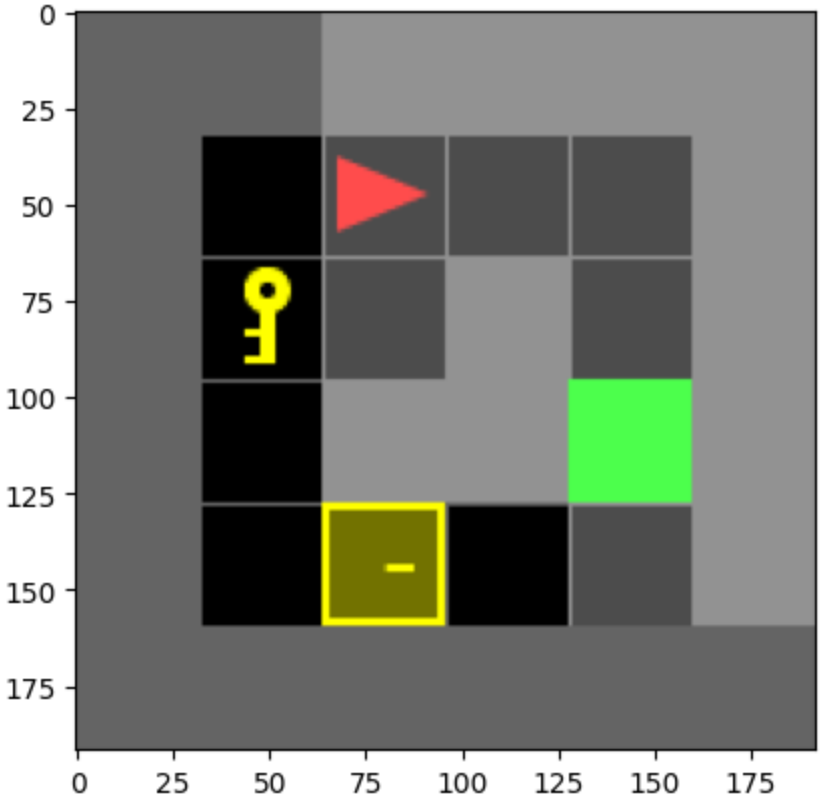

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

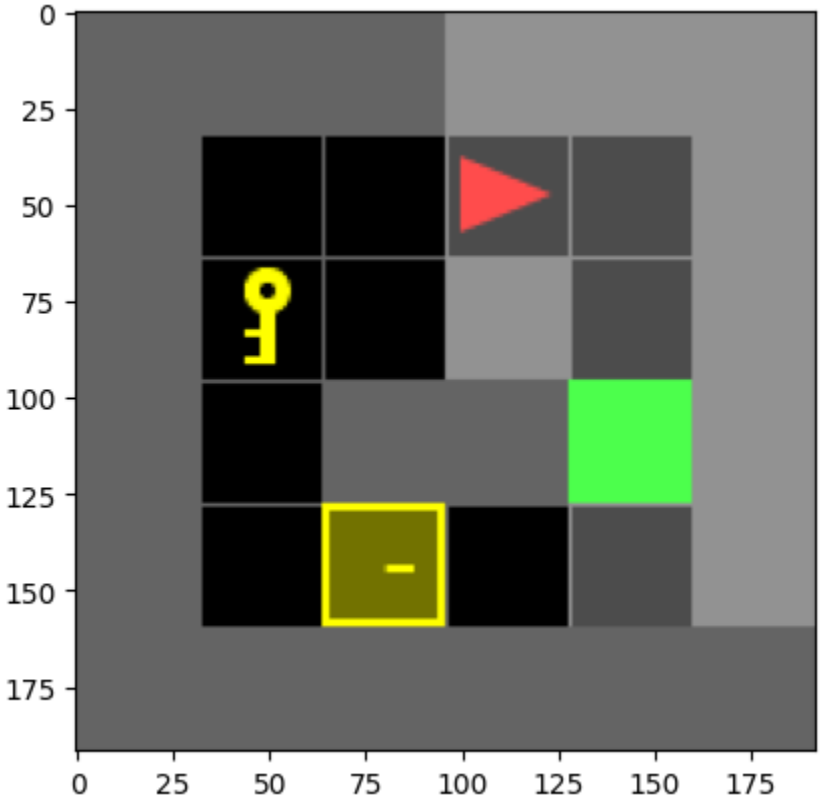

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

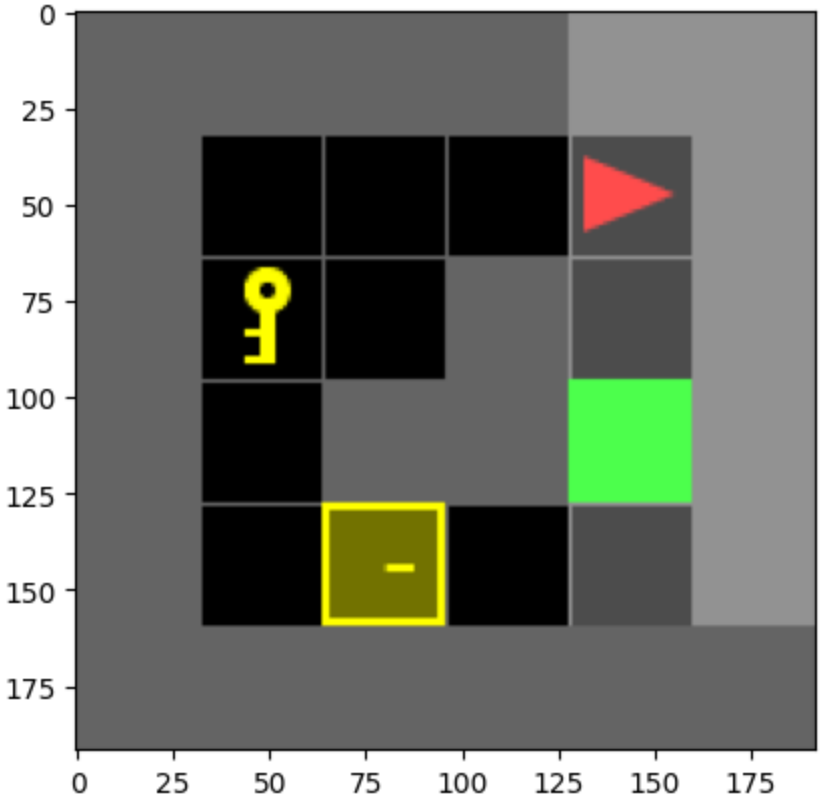


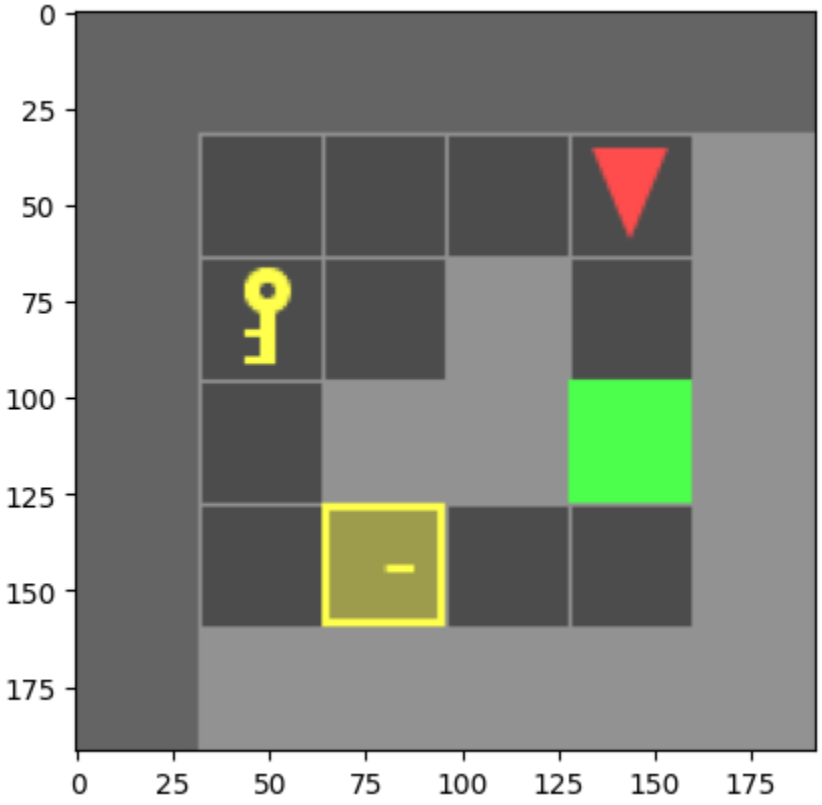
Doorkey-6x6-direct

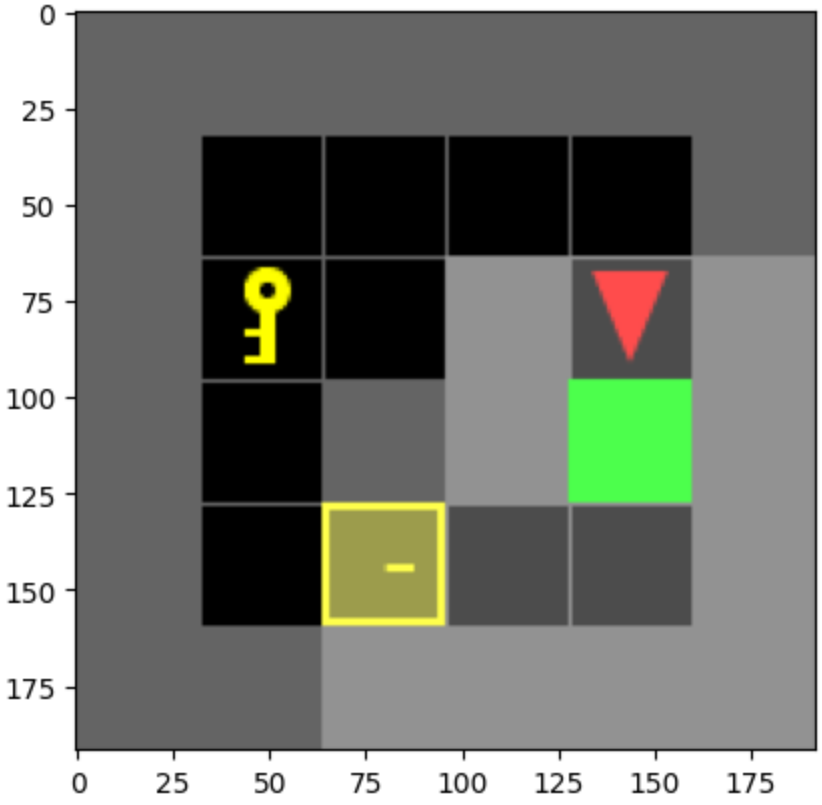
Optimal Sequence - [0, 0, 2, 0, 0]

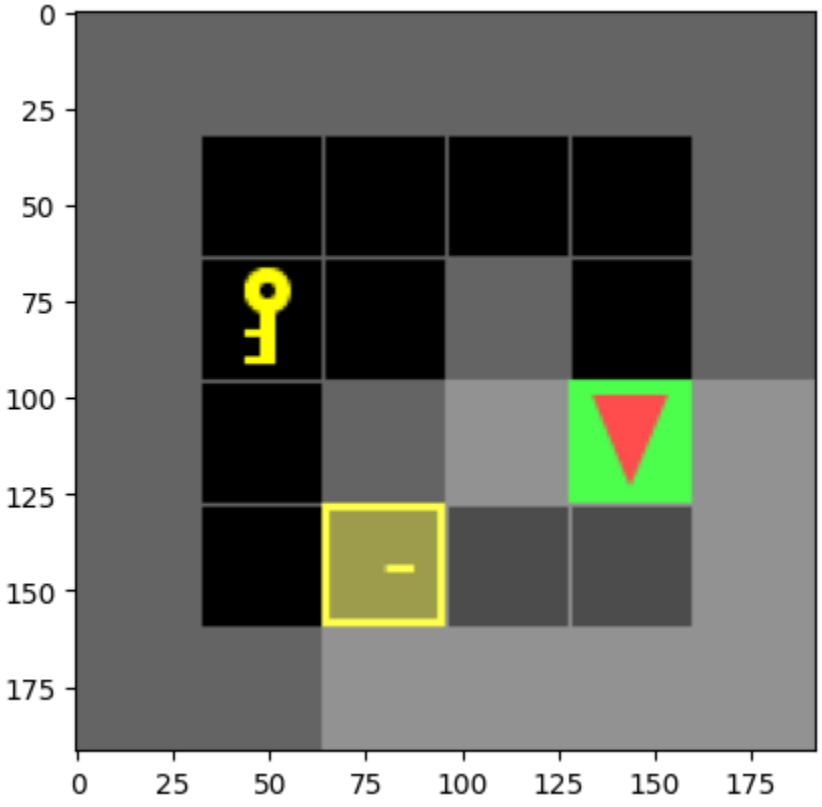

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175


0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175


0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

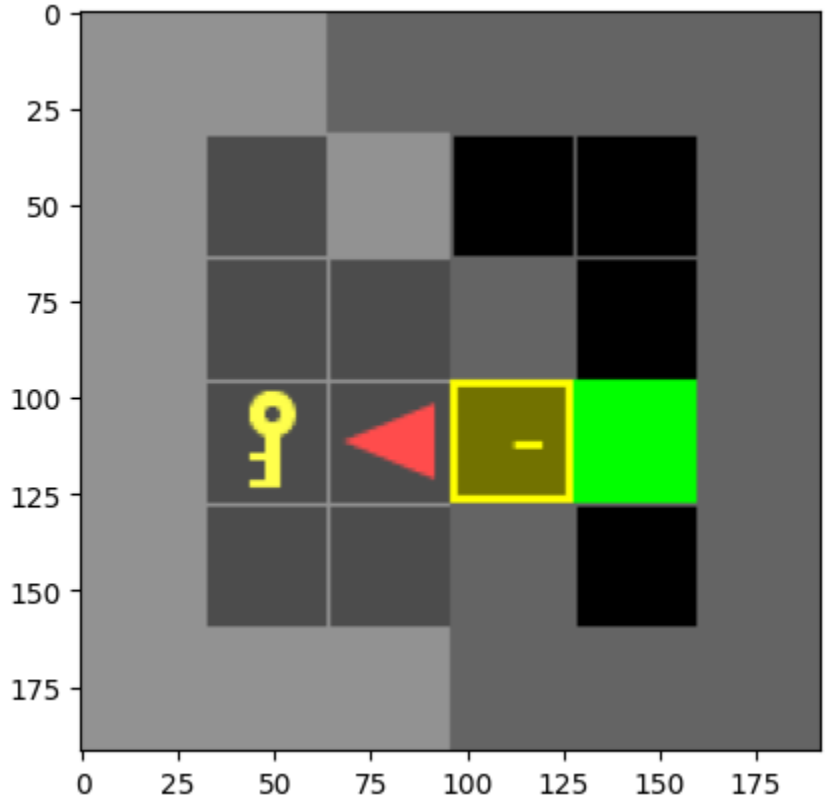

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

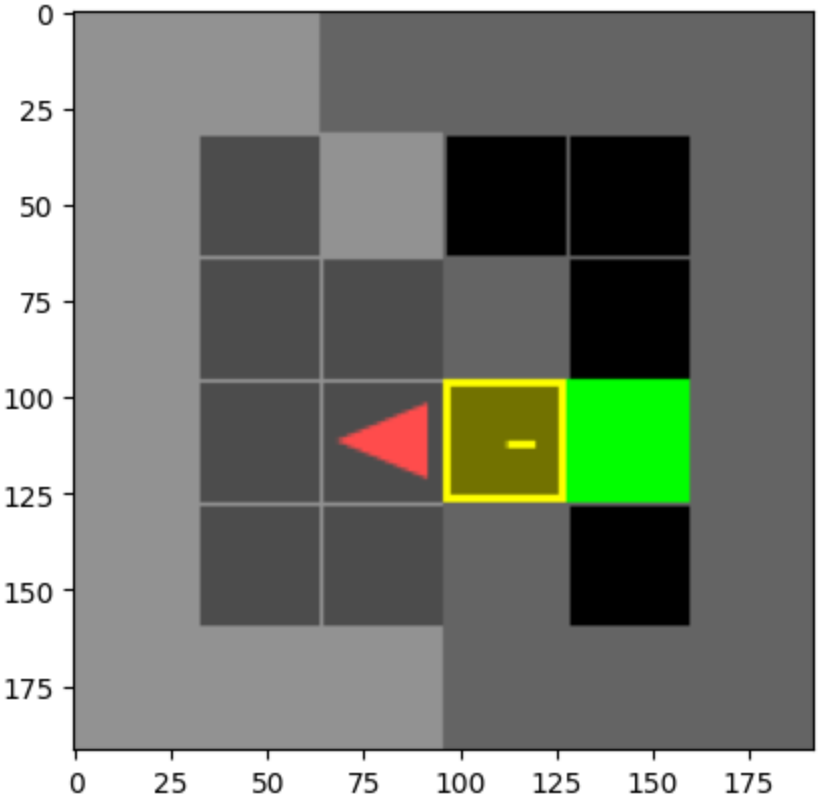

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

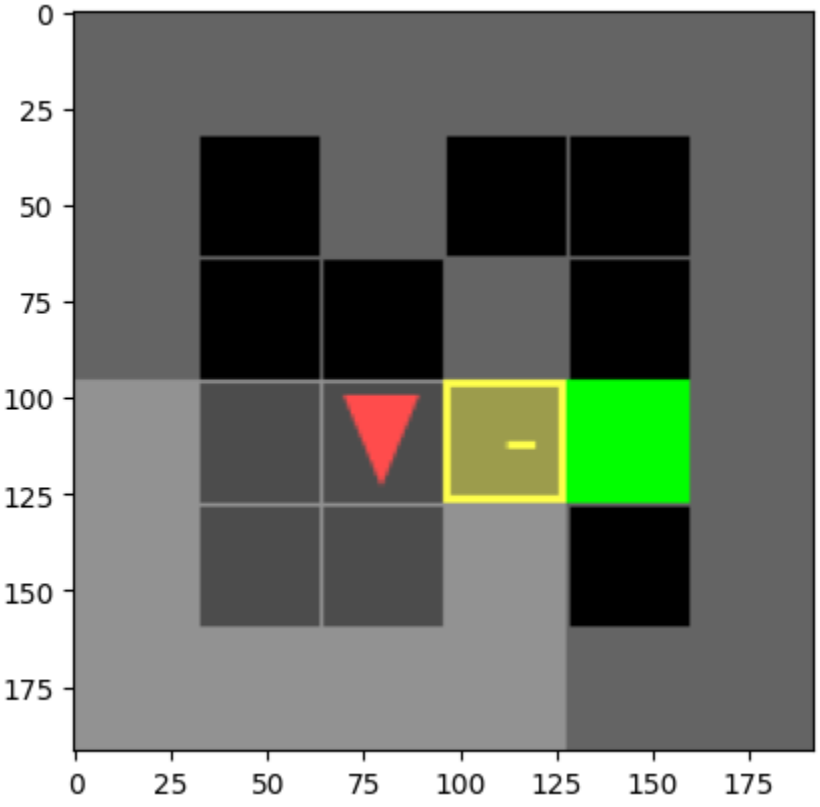

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

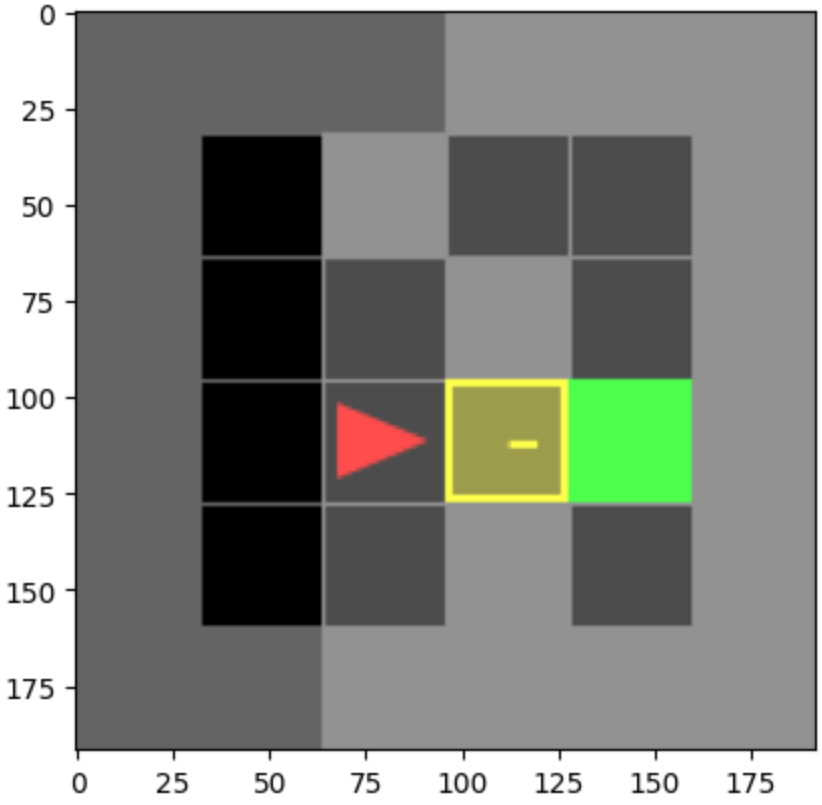
Doorkey-6x6-shortcut

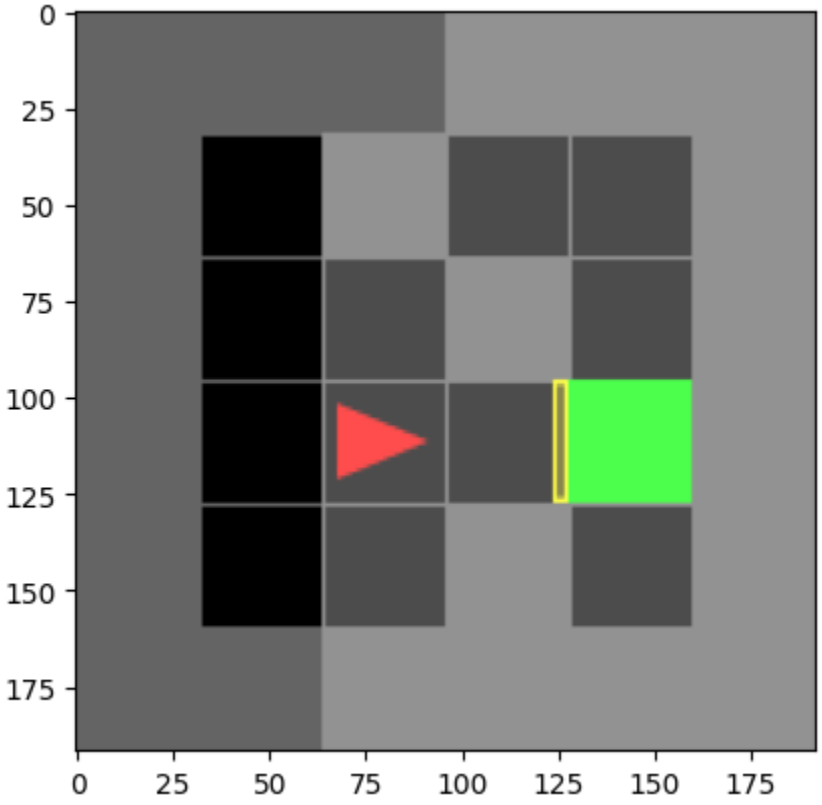
Optimal Sequence - [3, 1, 1, 4, 0, 0]

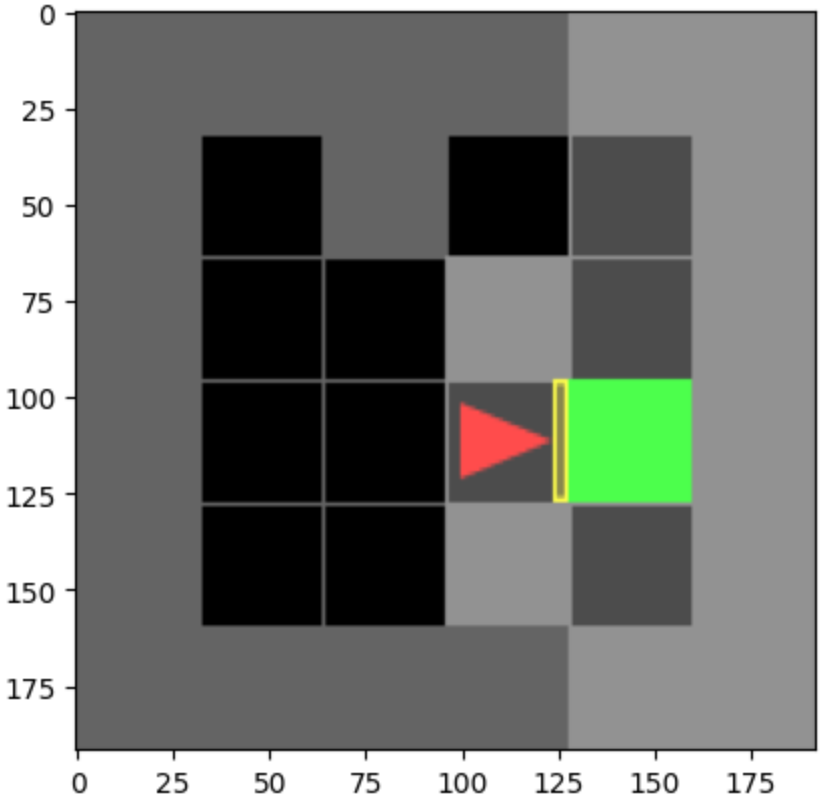


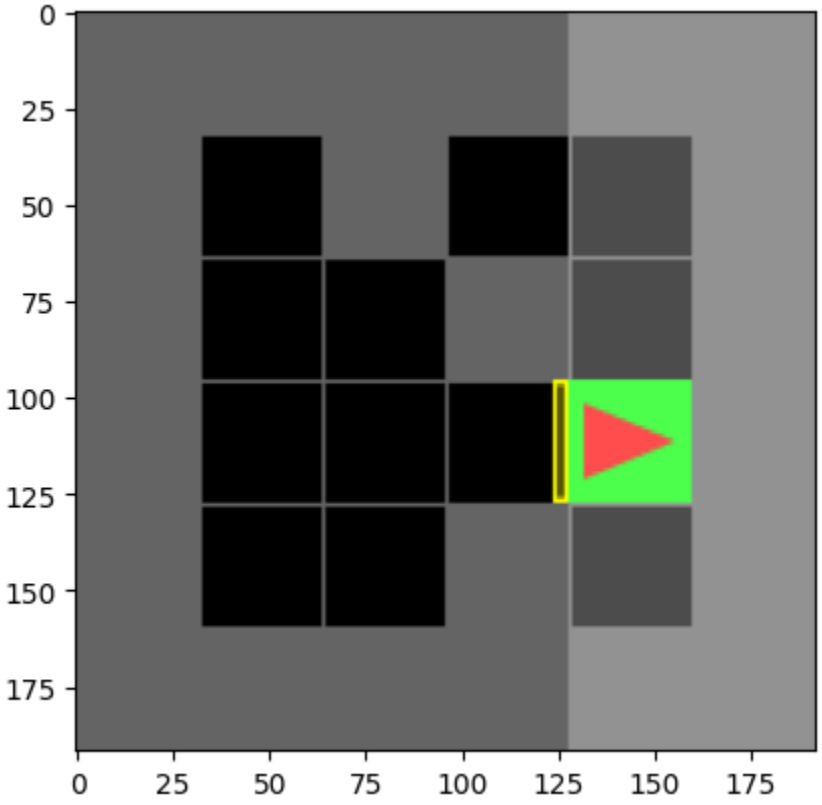

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175


0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175


0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

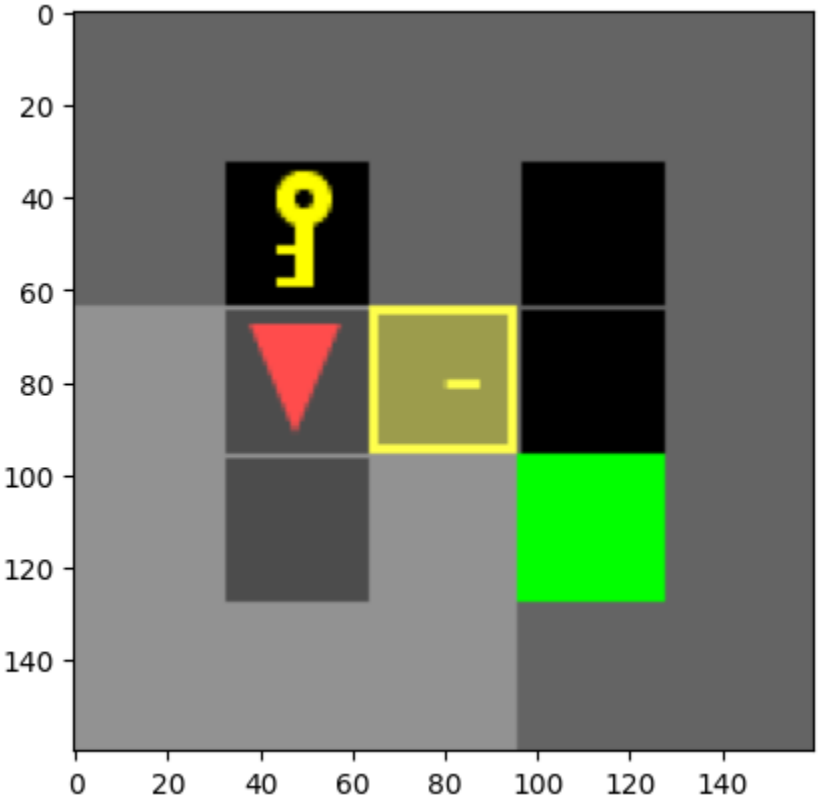

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

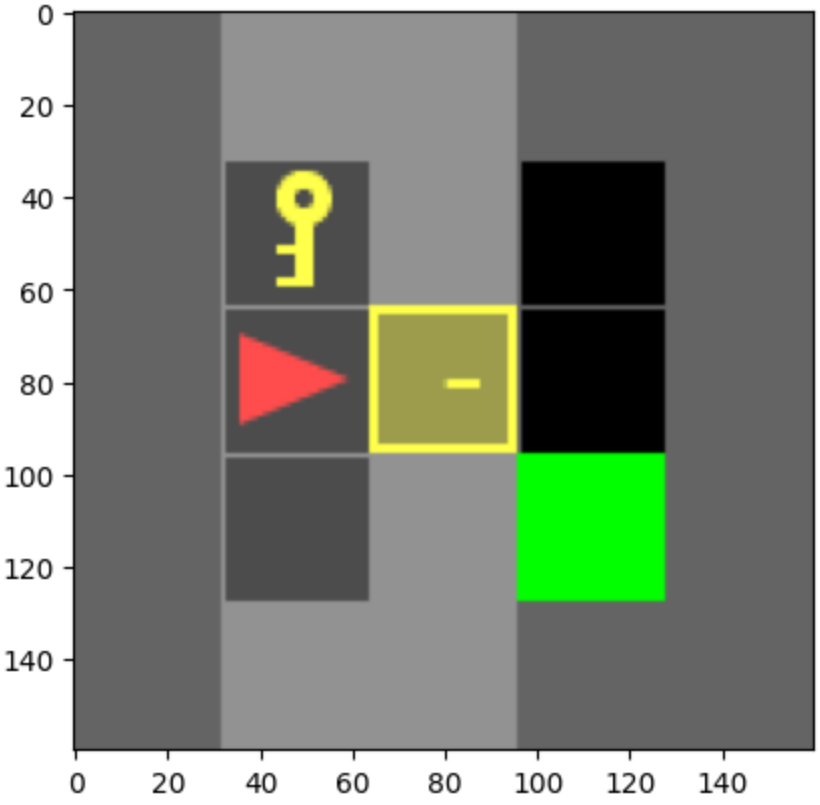

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

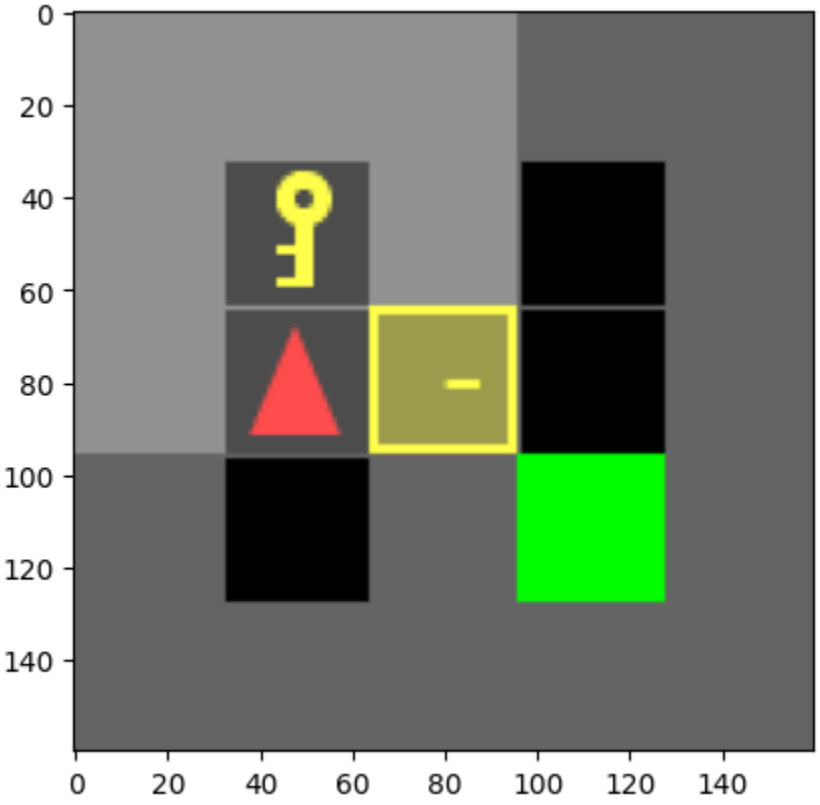

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

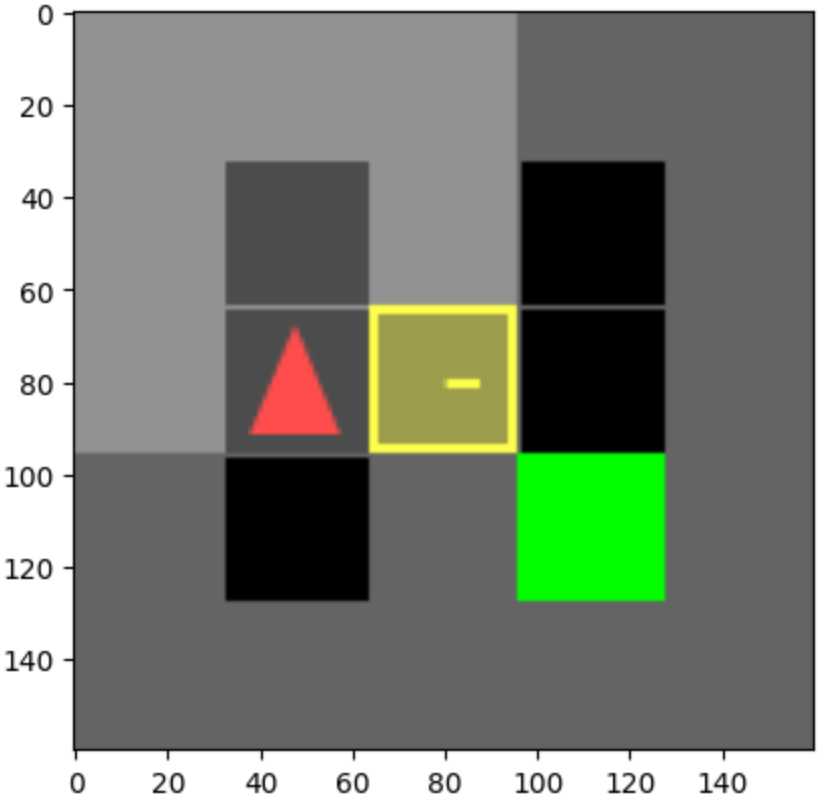
Doorkey-5x5-normal

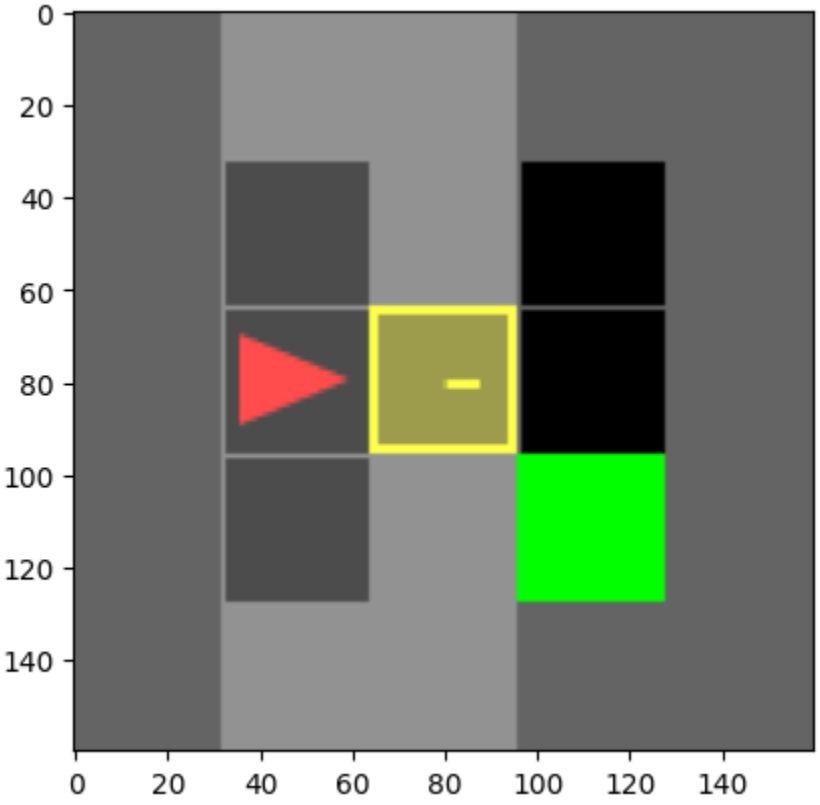
Optimal Sequence - [1, 1, 3, 2, 4, 0, 0, 2, 0]

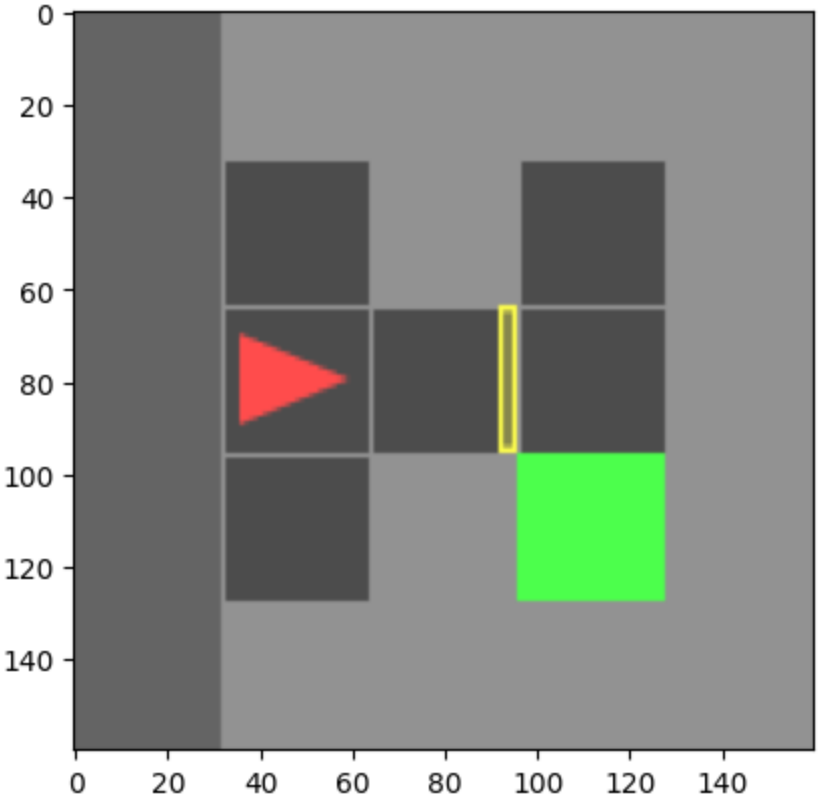


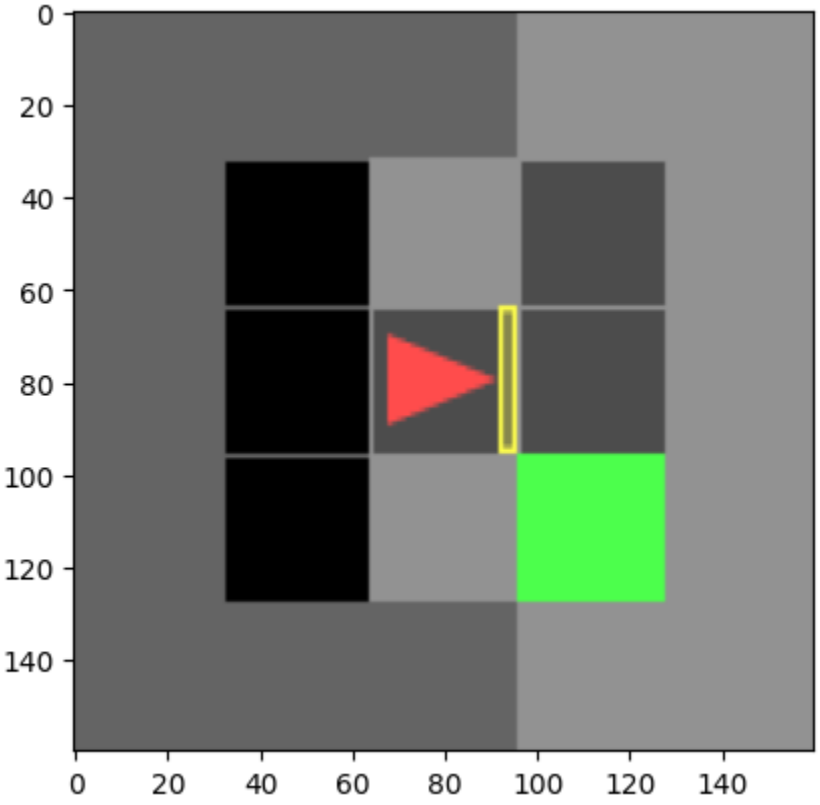

0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140

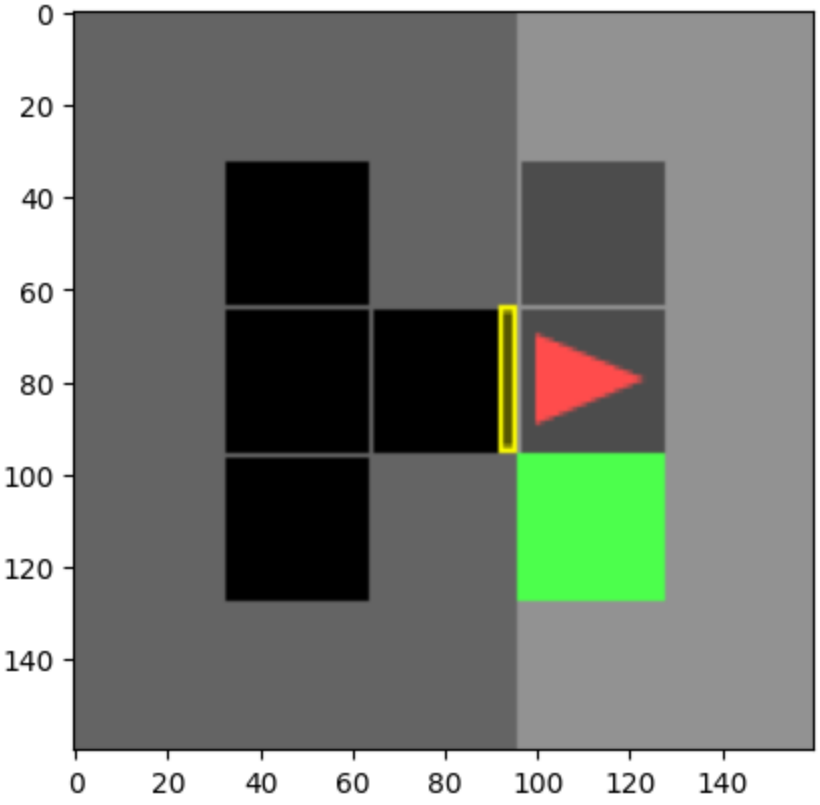

0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140

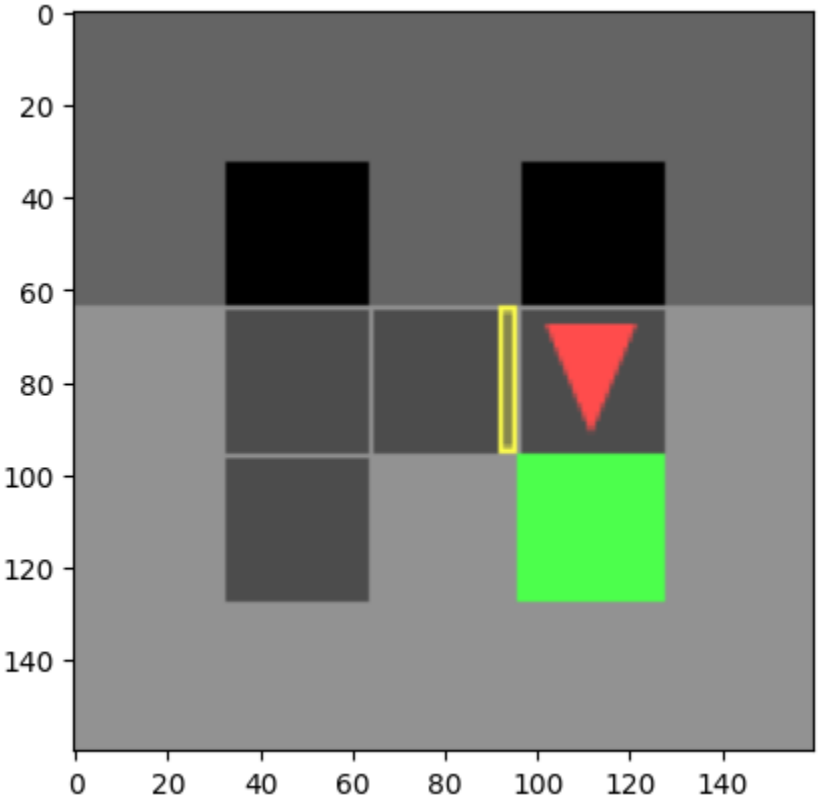

0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140

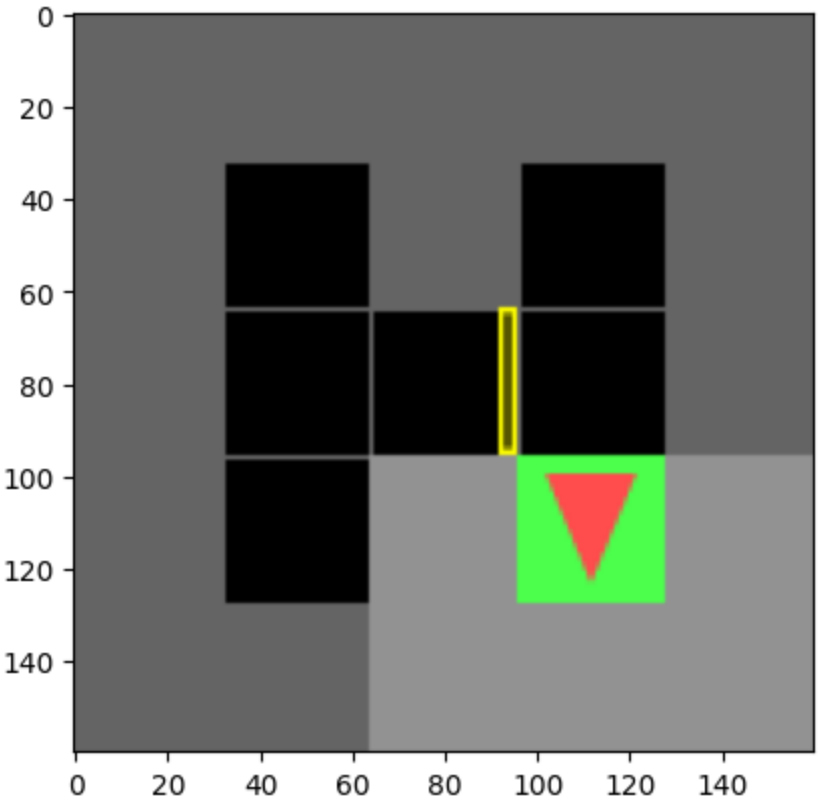

0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140


0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140


0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140

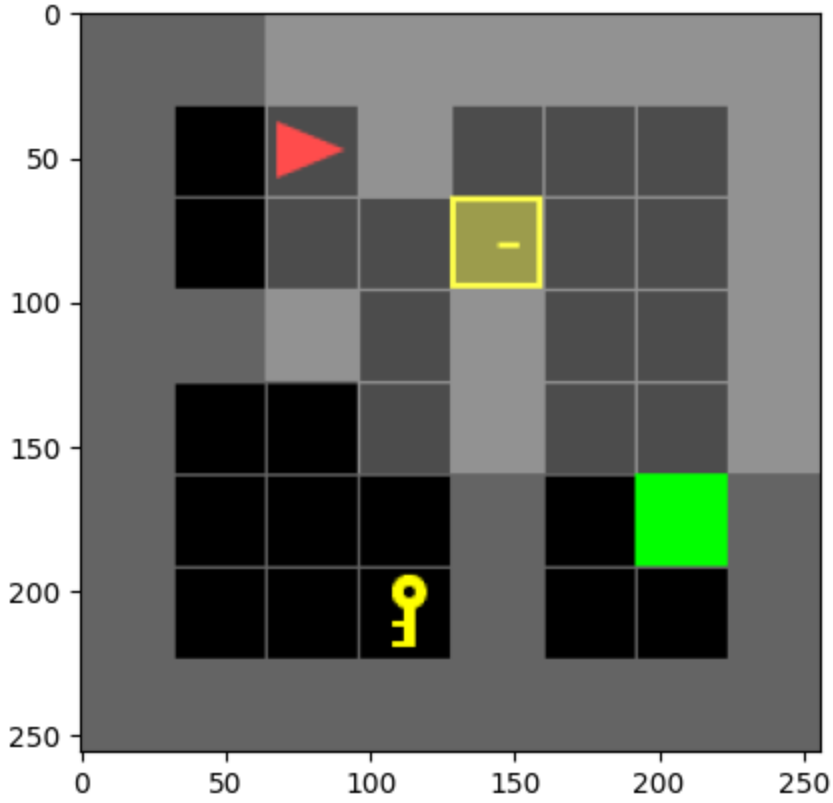

0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140

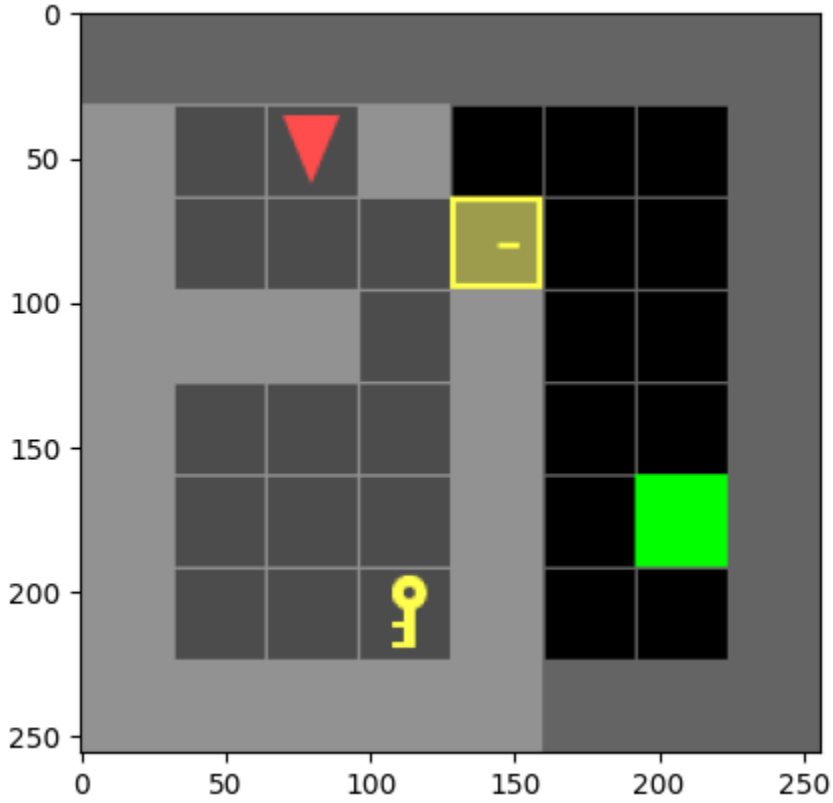

0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140

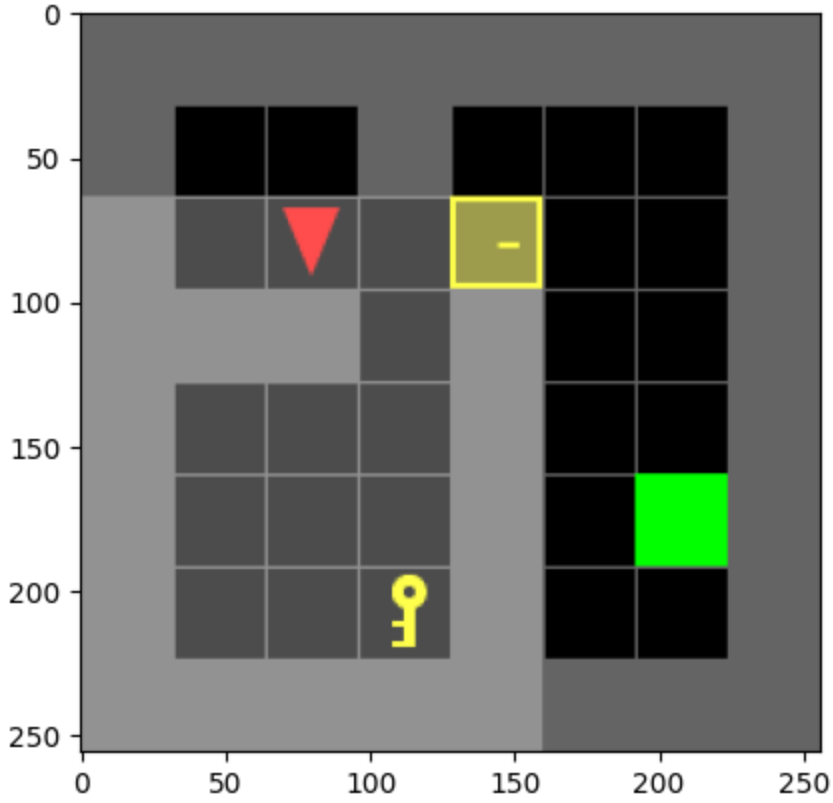


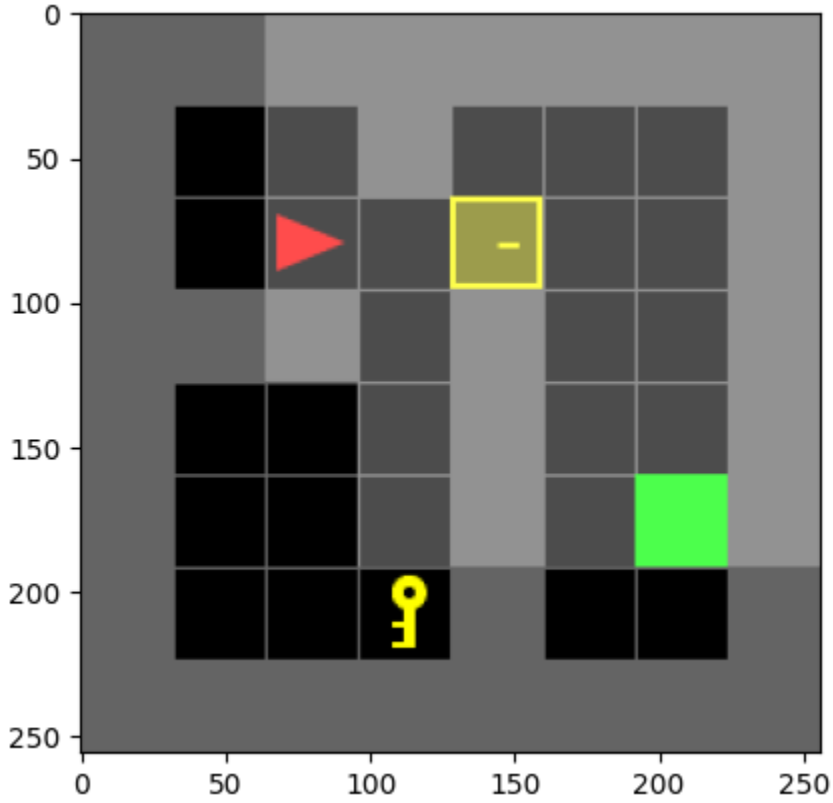
Doorkey-8x8-normal

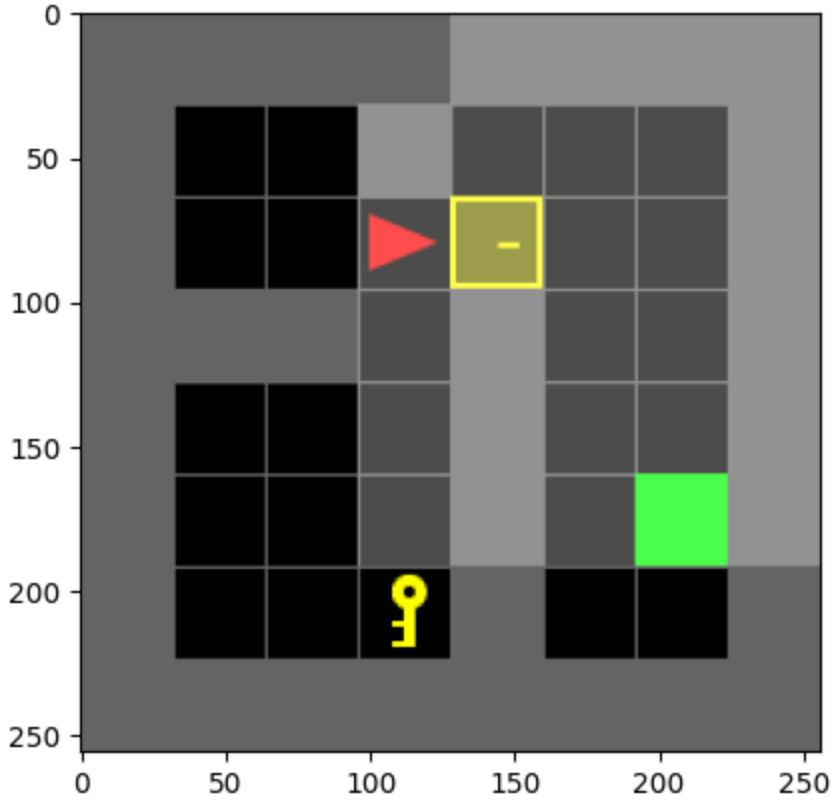
Optimal Sequence - [2, 0, 1, 0, 2, 0, 0, 0, 3, 1, 1, 0, 0, 0, 2, 4, 0, 0, 0, 2, 0, 0, 0]

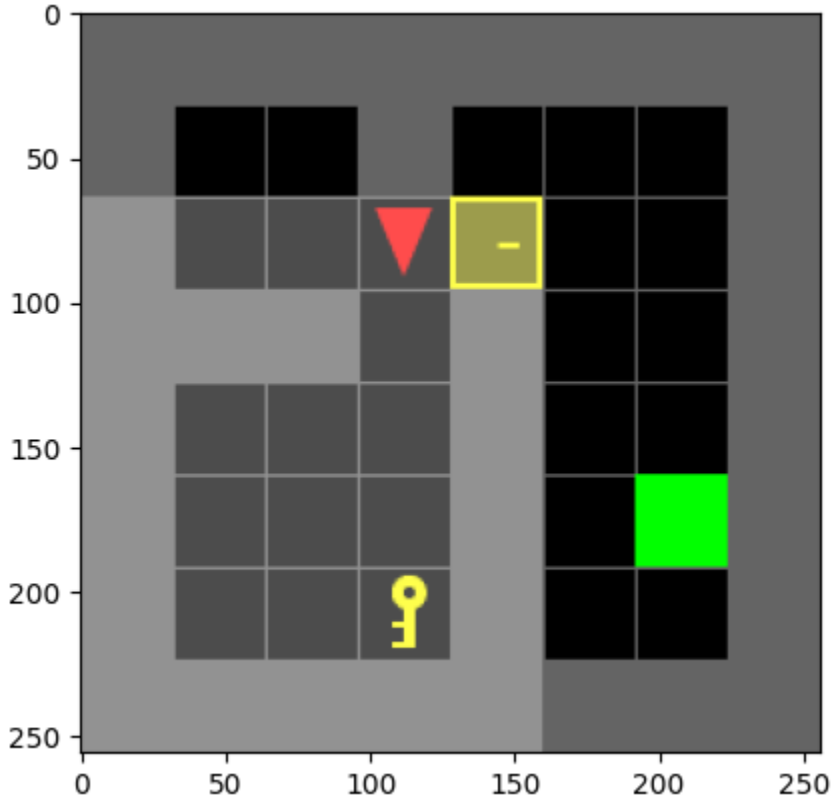

0
50
100
150
200
250
0
50
100
150
200
250

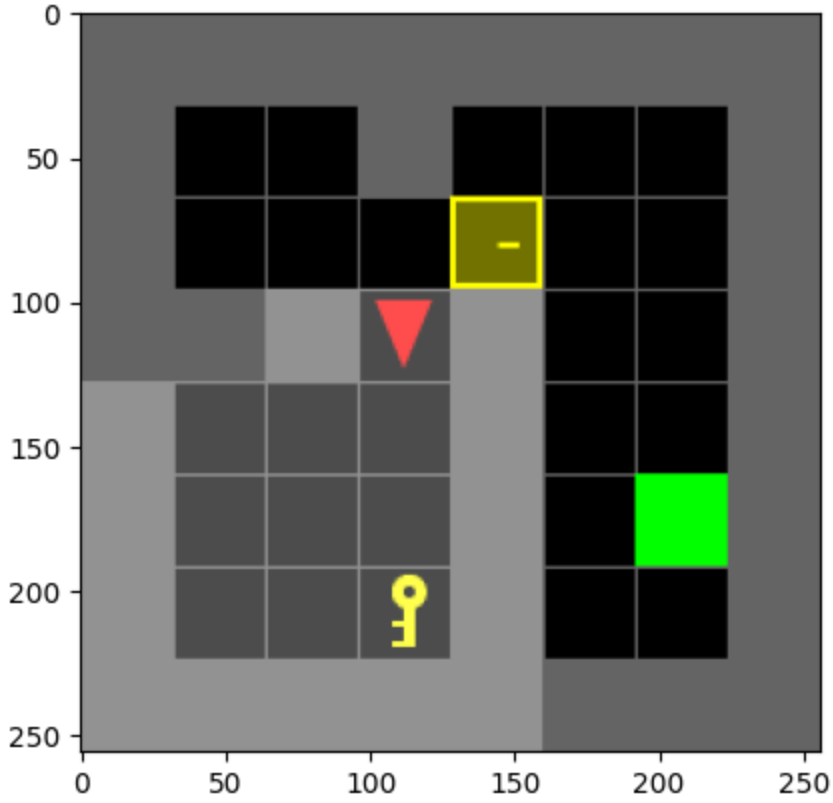

0
50
100
150
200
250
0
50
100
150
200
250

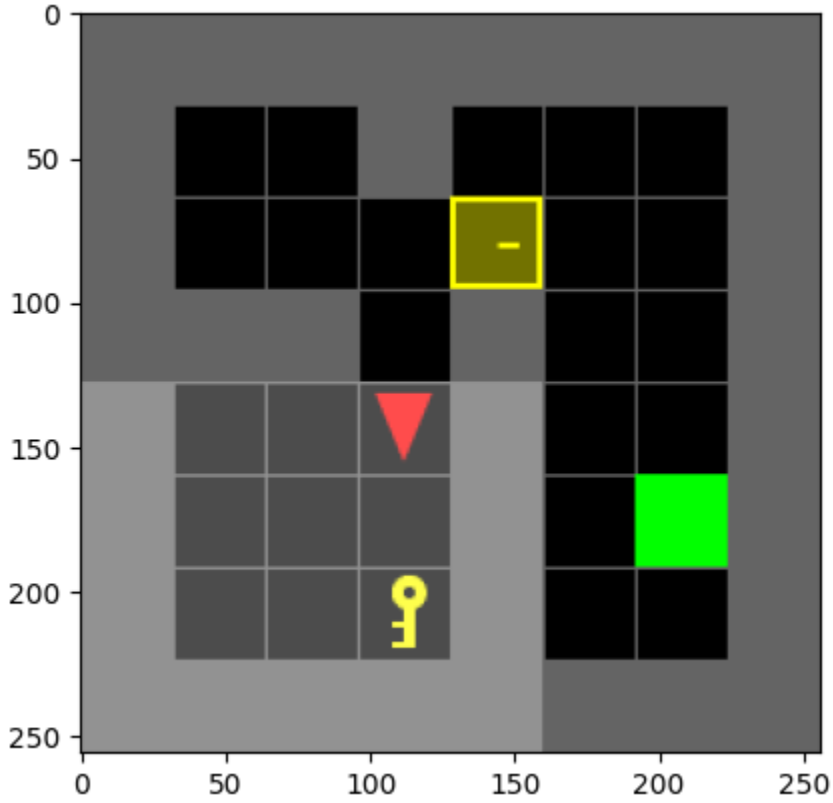

0
50
100
150
200
250
0
50
100
150
200
250

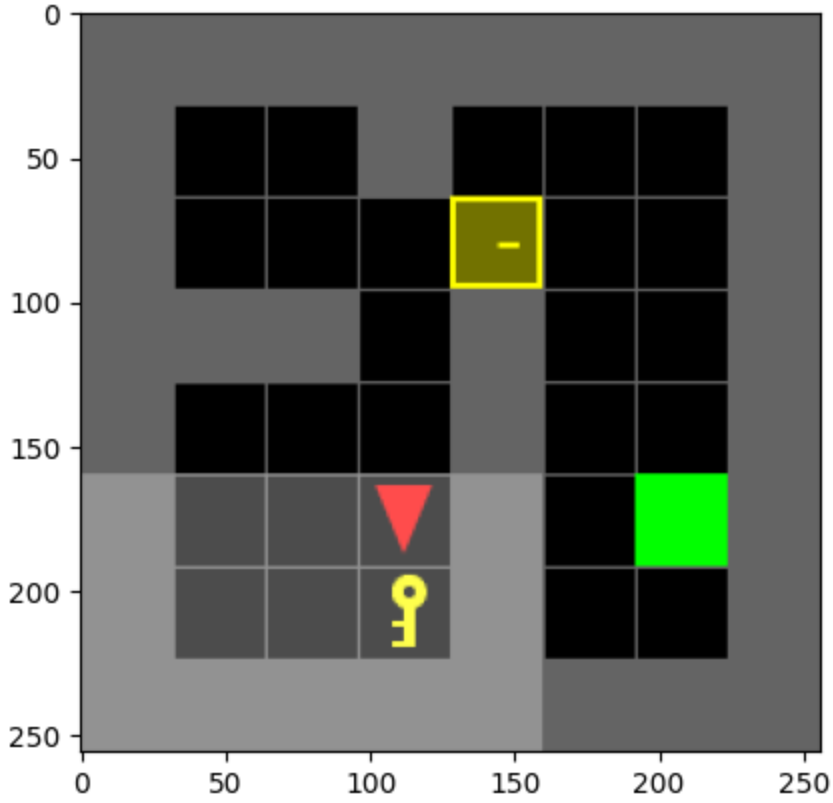

0
50
100
150
200
250
0
50
100
150
200
250

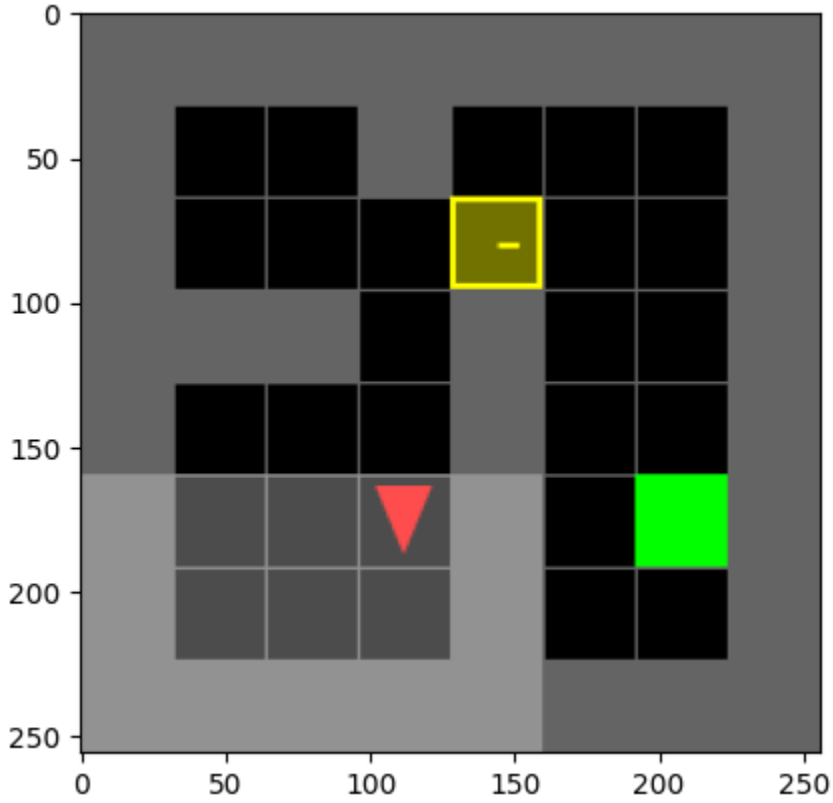

0
50
100
150
200
250
0
50
100
150
200
250

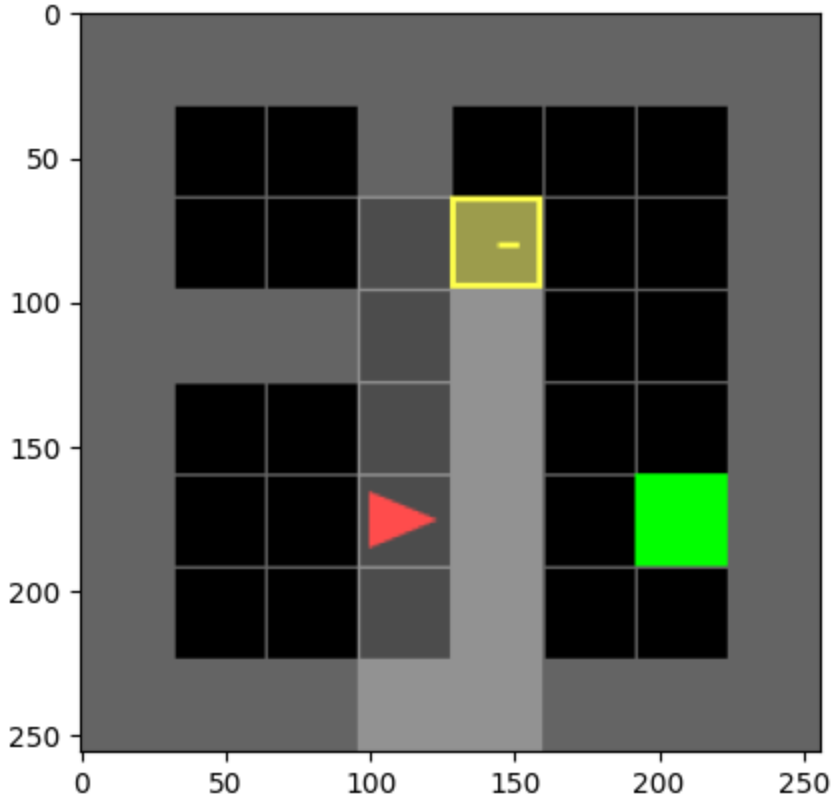

0
50
100
150
200
250
0
50
100
150
200
250

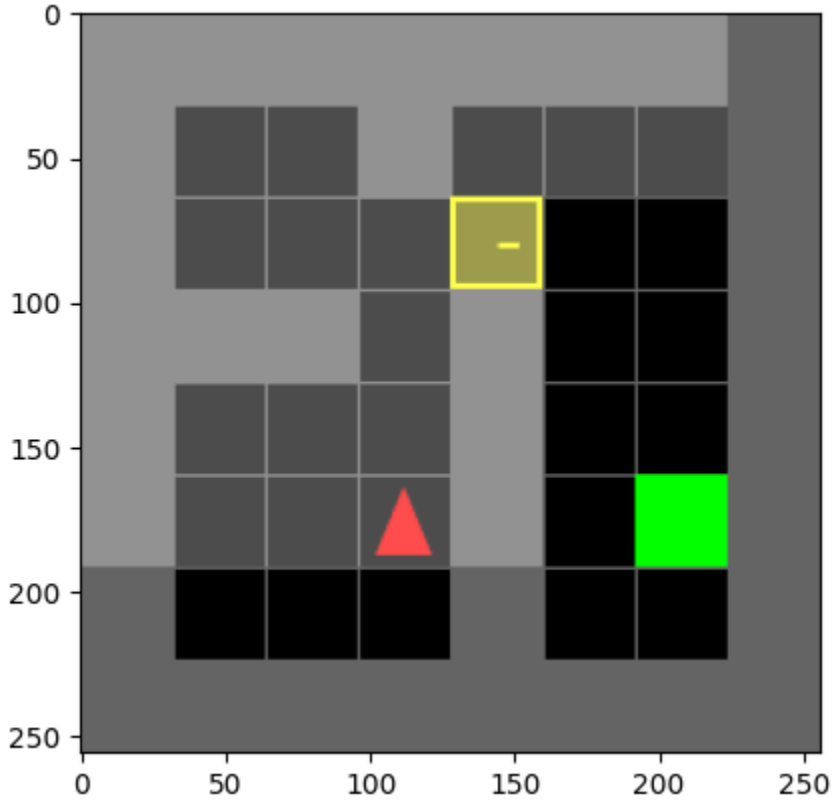

0
50
100
150
200
250
0
50
100
150
200
250

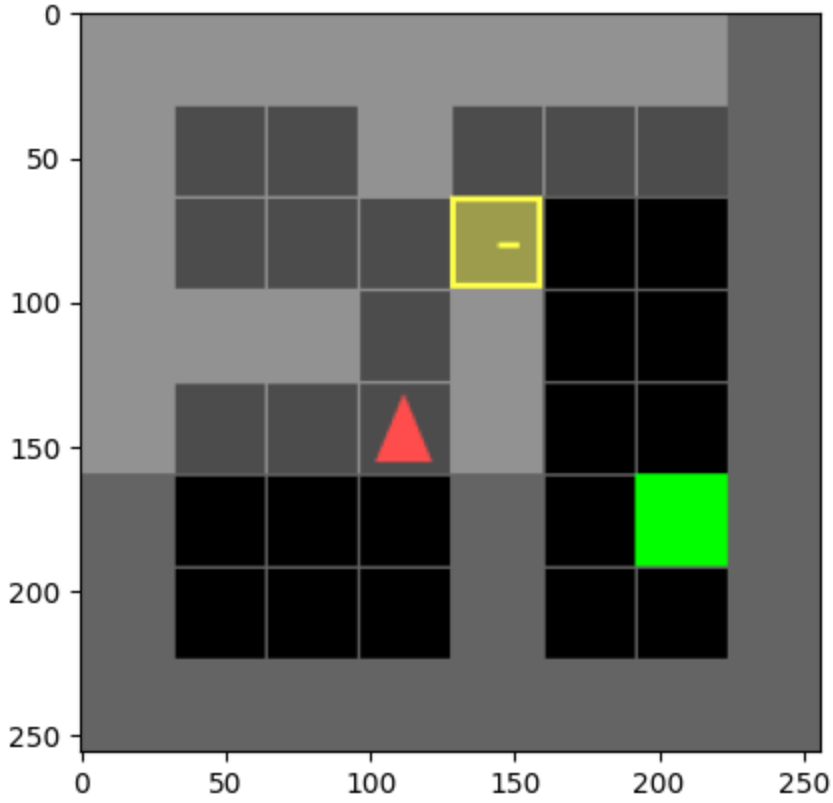

0
50
100
150
200
250
0
50
100
150
200
250

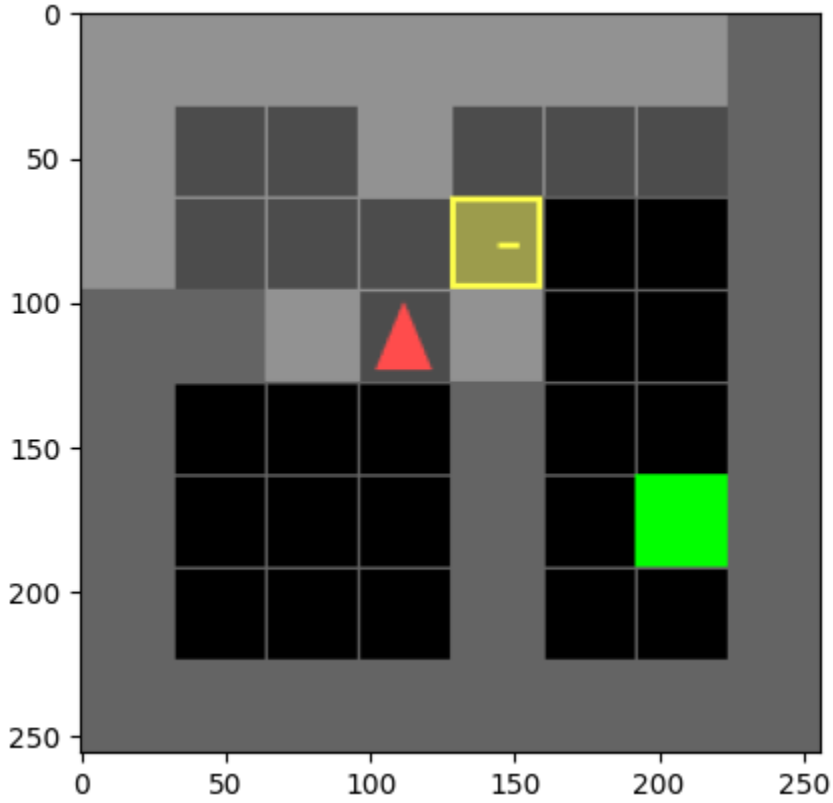

0
50
100
150
200
250
0
50
100
150
200
250

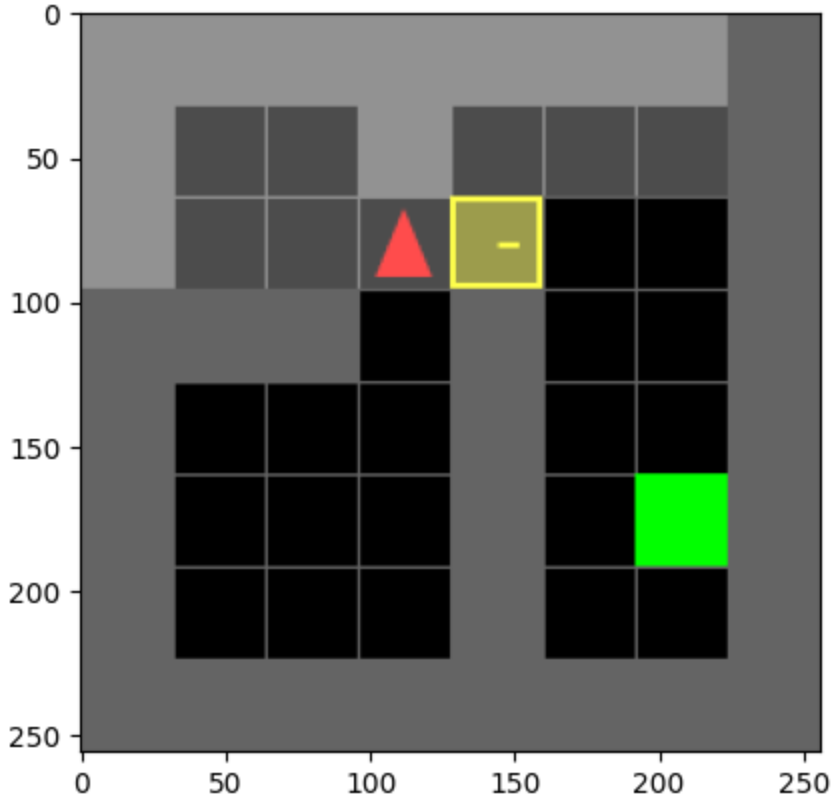

0
50
100
150
200
250
0
50
100
150
200
250

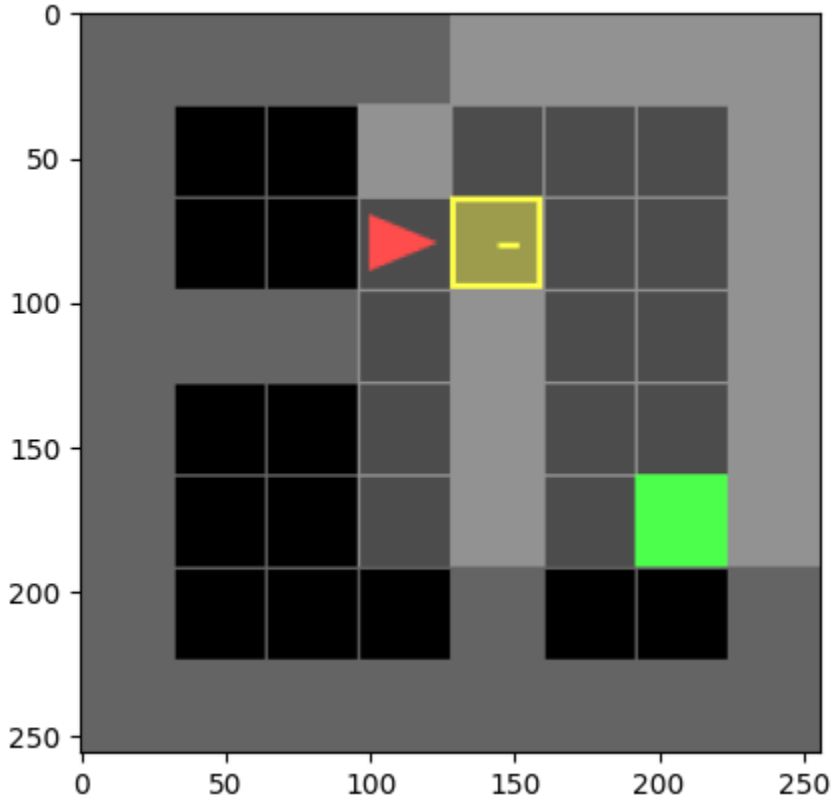

0
50
100
150
200
250
0
50
100
150
200
250

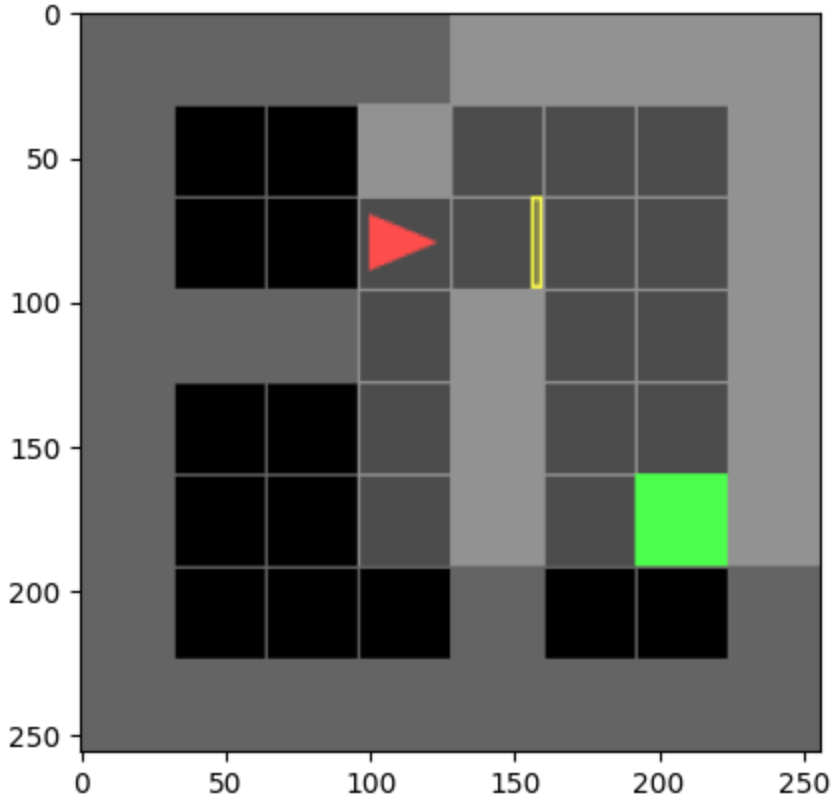

0
50
100
150
200
250
0
50
100
150
200
250

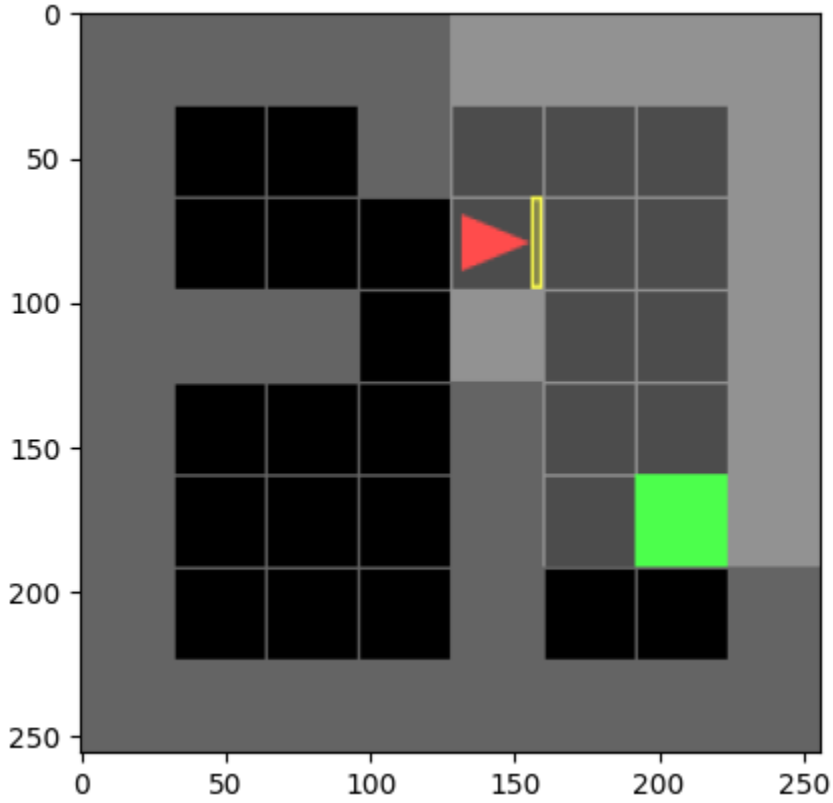

0
50
100
150
200
250
0
50
100
150
200
250

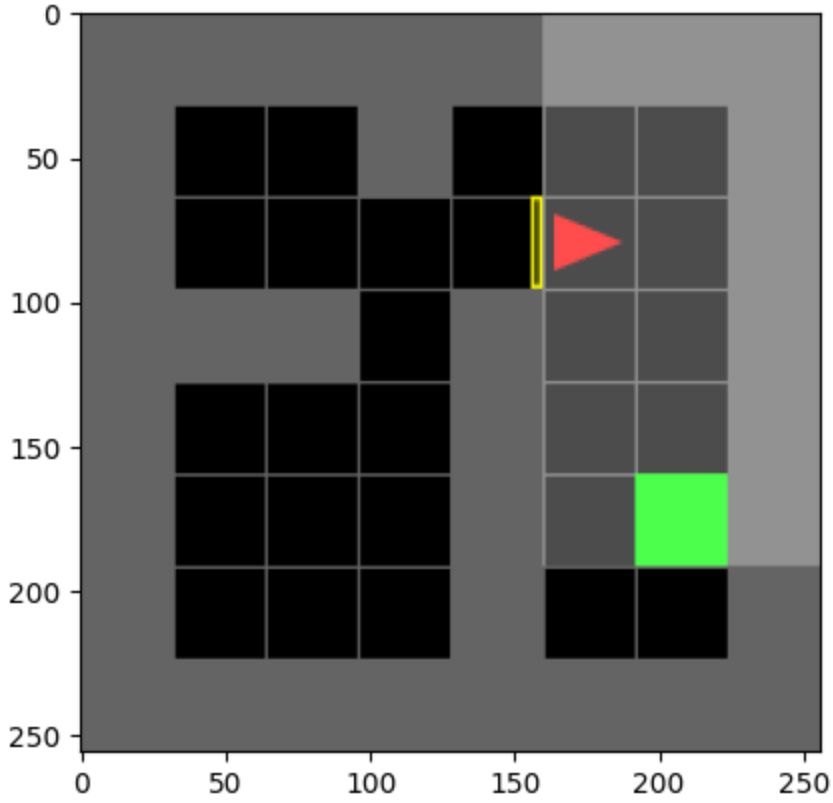

0
50
100
150
200
250
0
50
100
150
200
250

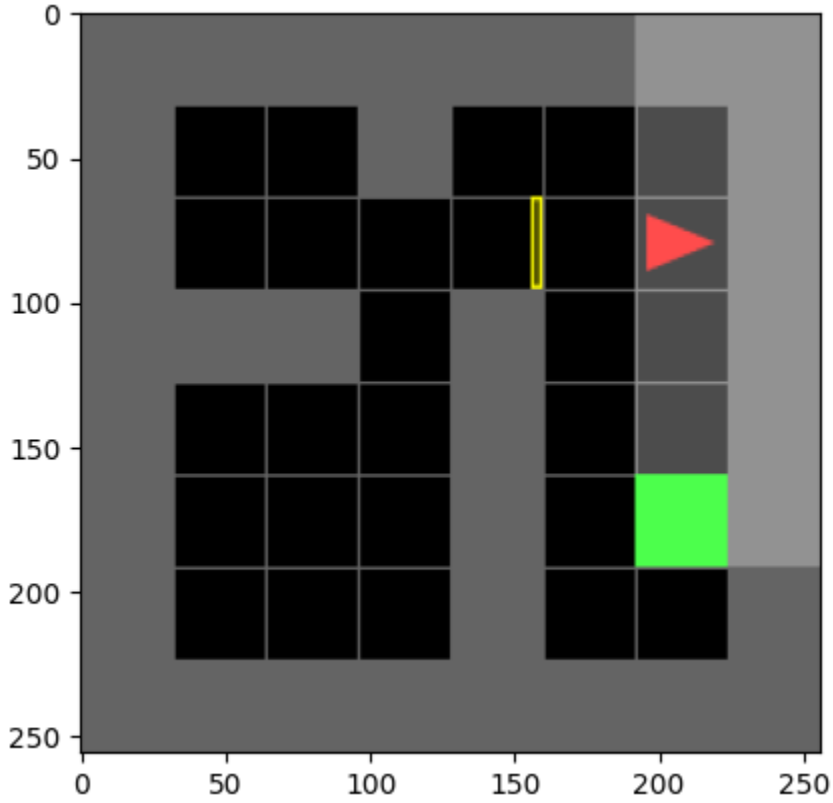

0
50
100
150
200
250
0
50
100
150
200
250

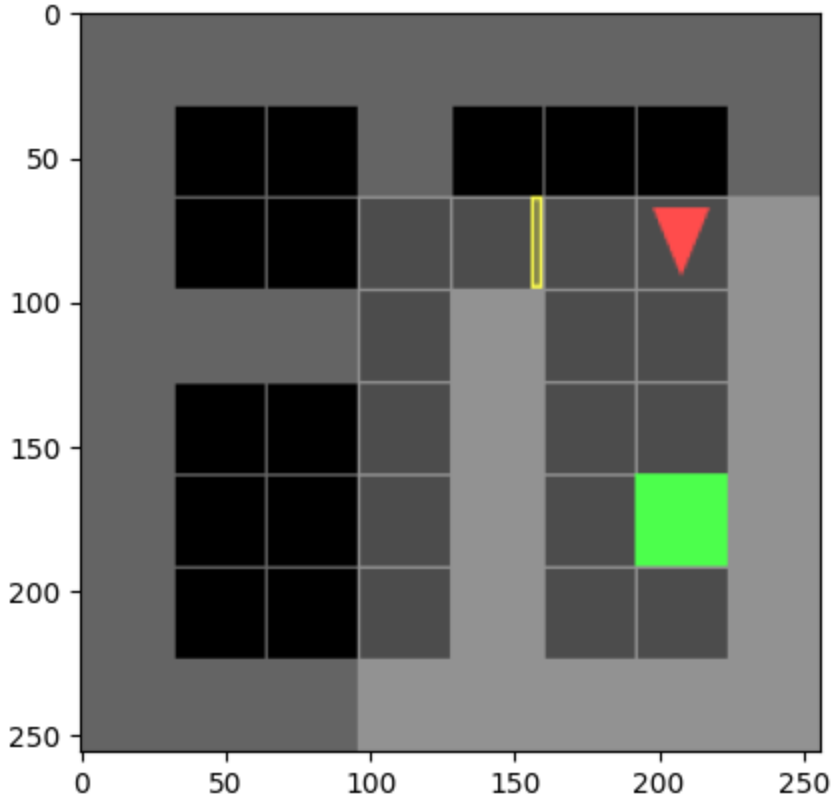

0
50
100
150
200
250
0
50
100
150
200
250

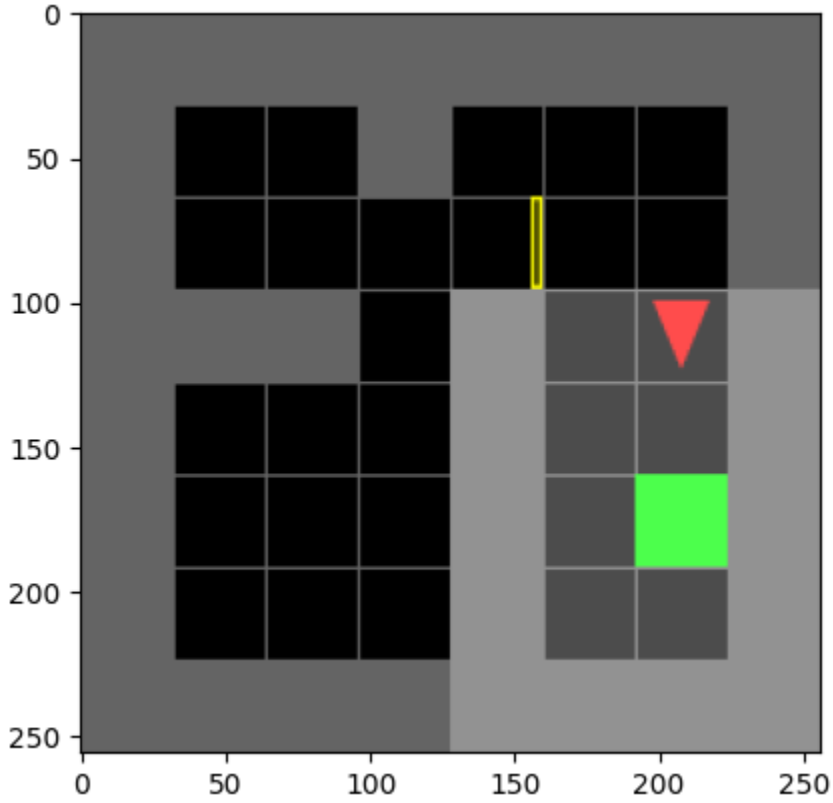

0
50
100
150
200
250
0
50
100
150
200
250

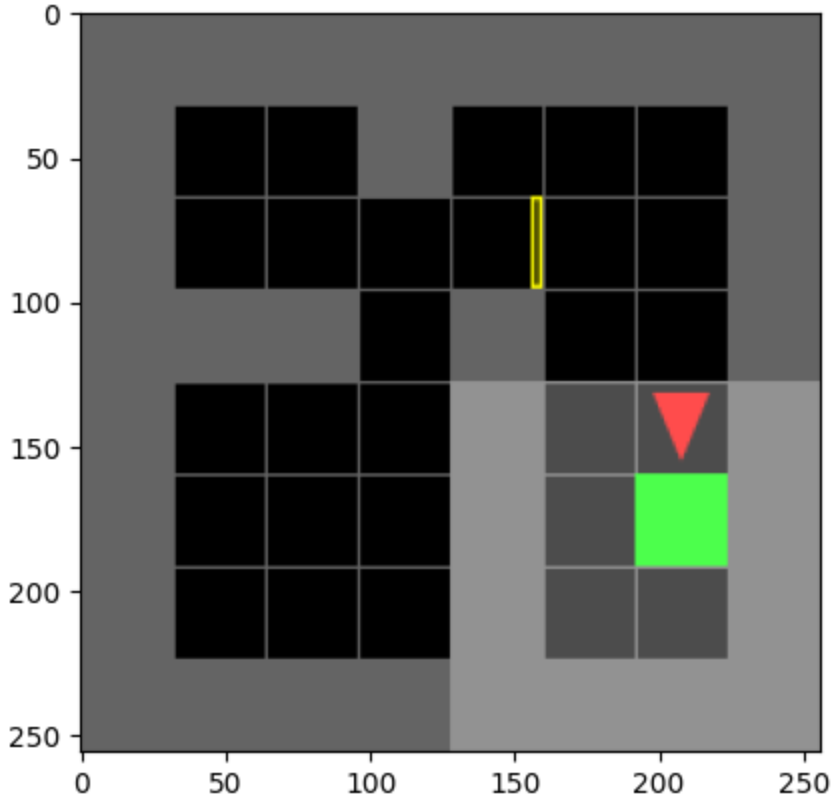

0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250

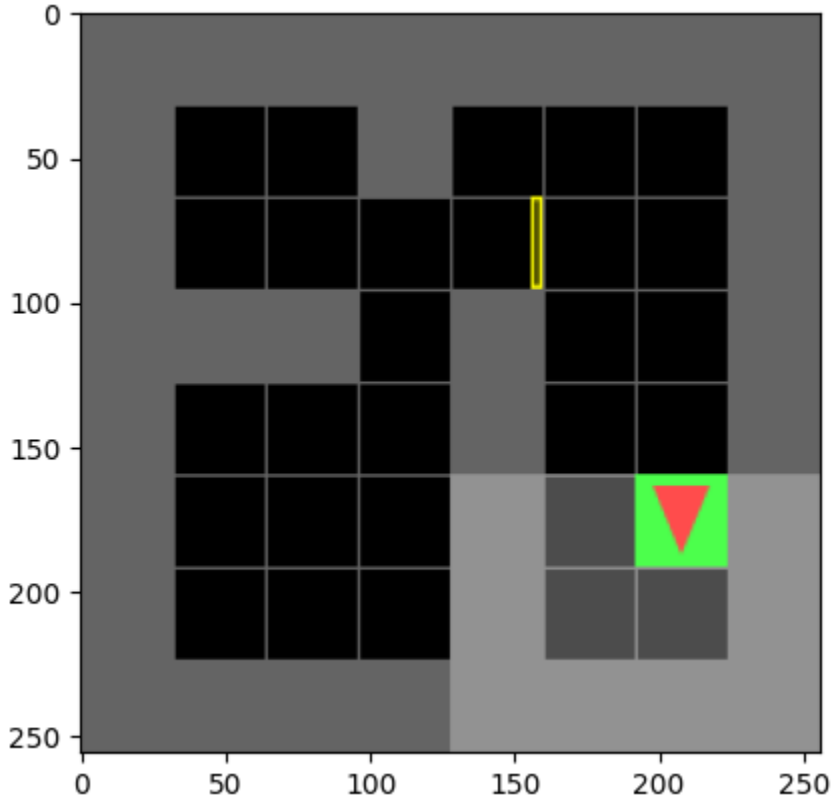
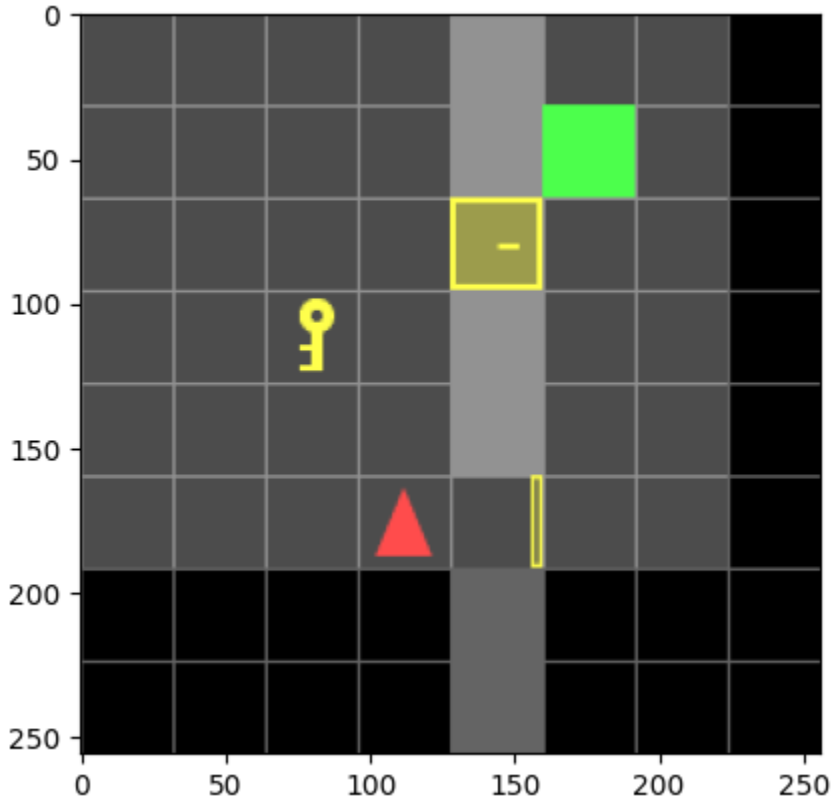
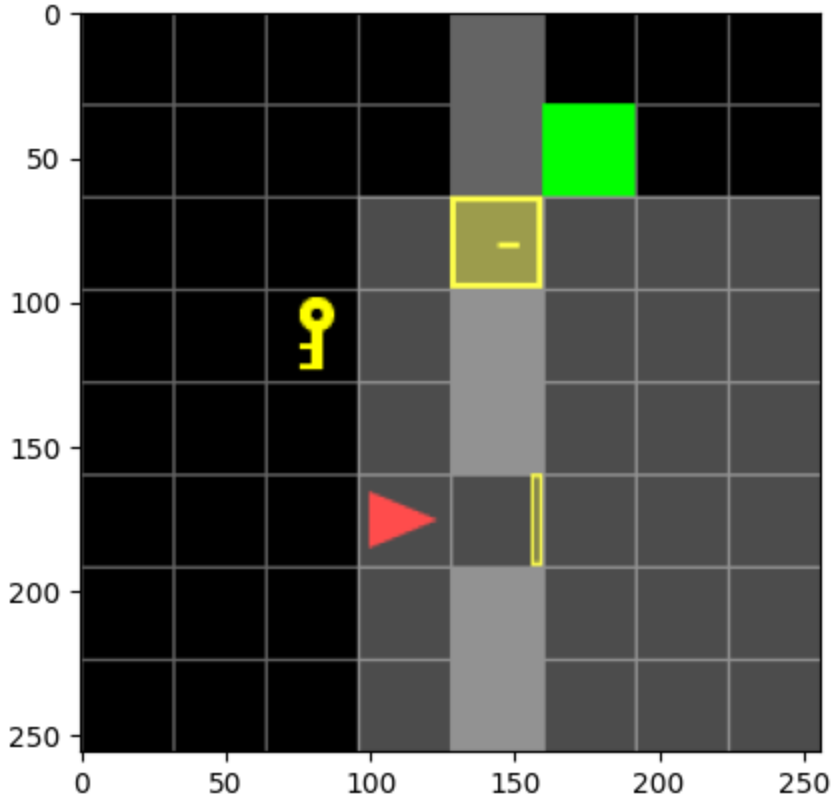

0
50
100
150
200
250
0
50
100
150
200
250

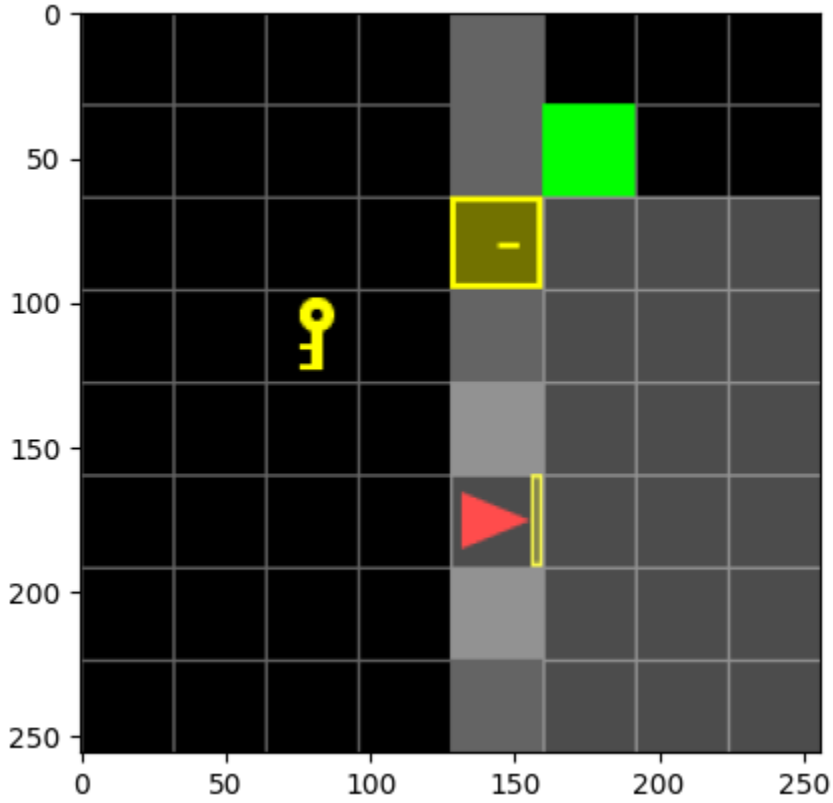
Figure 2 - Random Environments

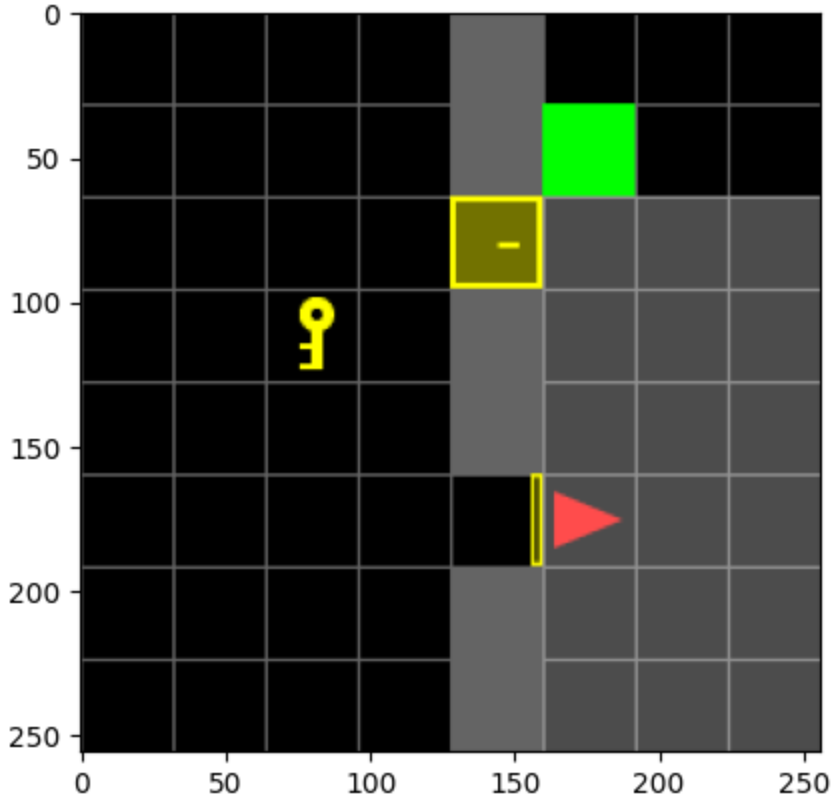
DoorKey-8x8-15.env

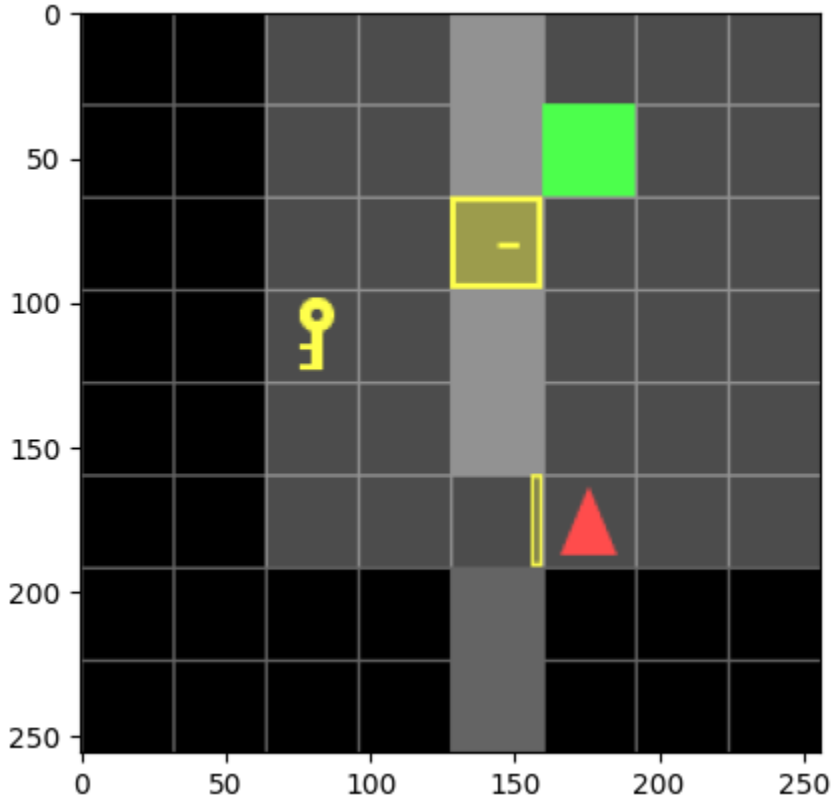
Optimal Sequence - [2, 0, 0, 1, 0, 0, 0, 0]

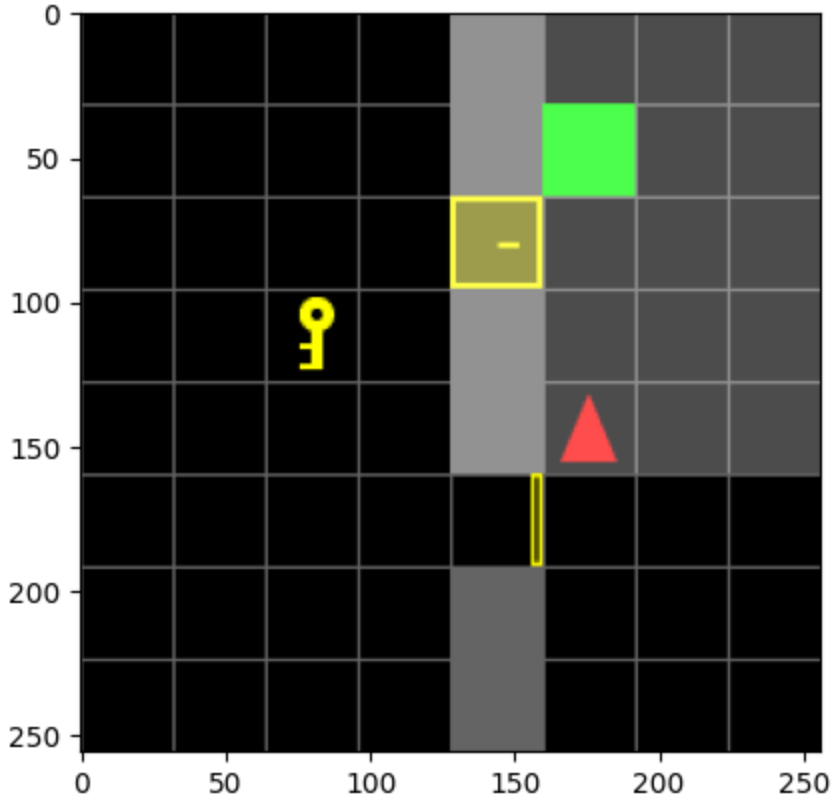


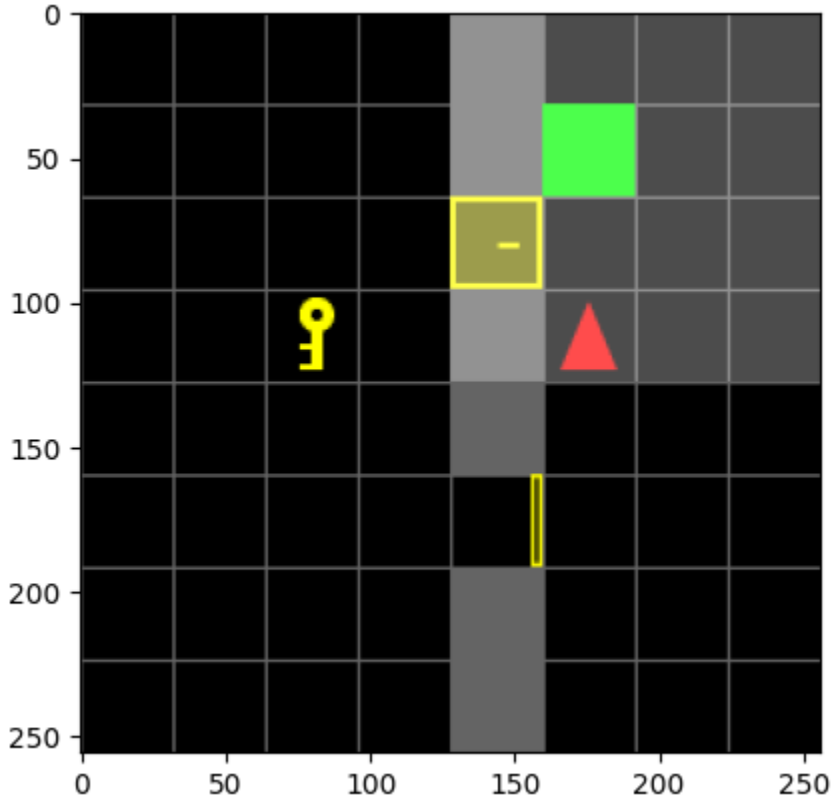

0
50
100
150
200
250
0
50
100
150
200
250

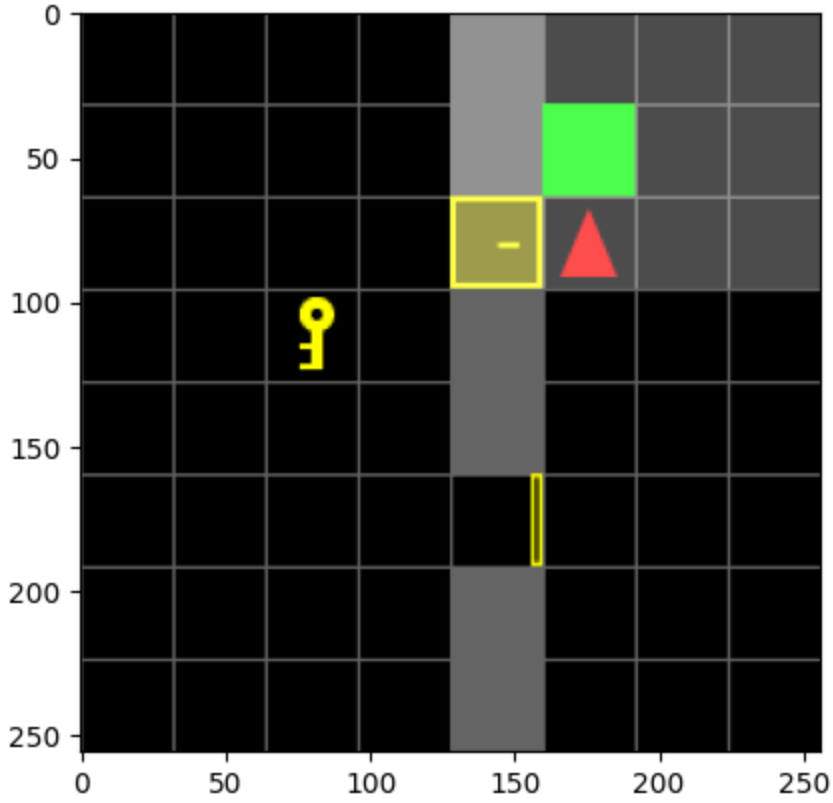

0
50
100
150
200
250
0
50
100
150
200
250

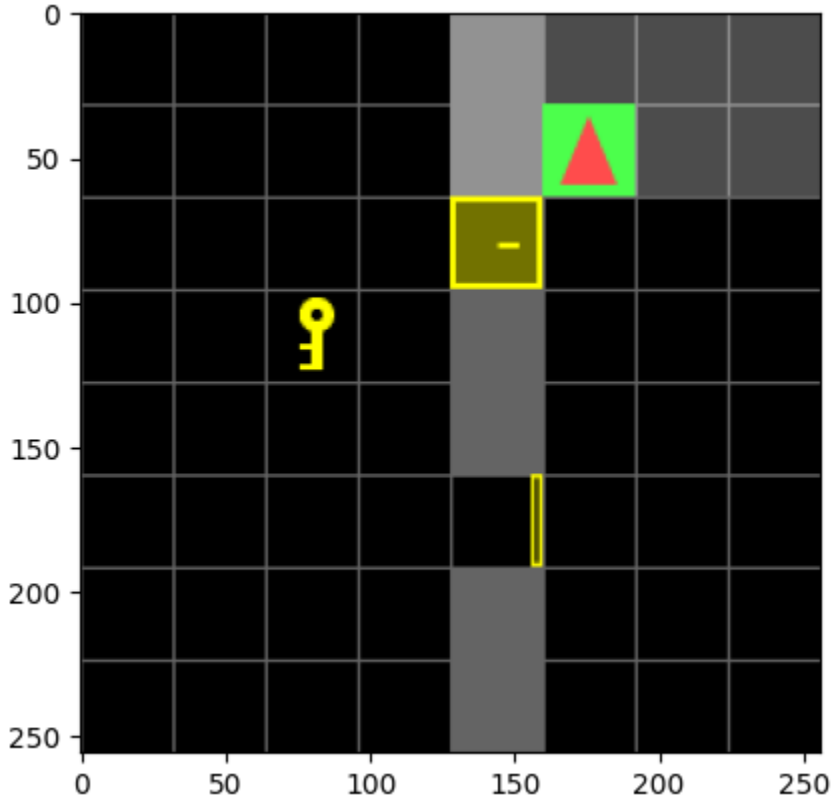

0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250

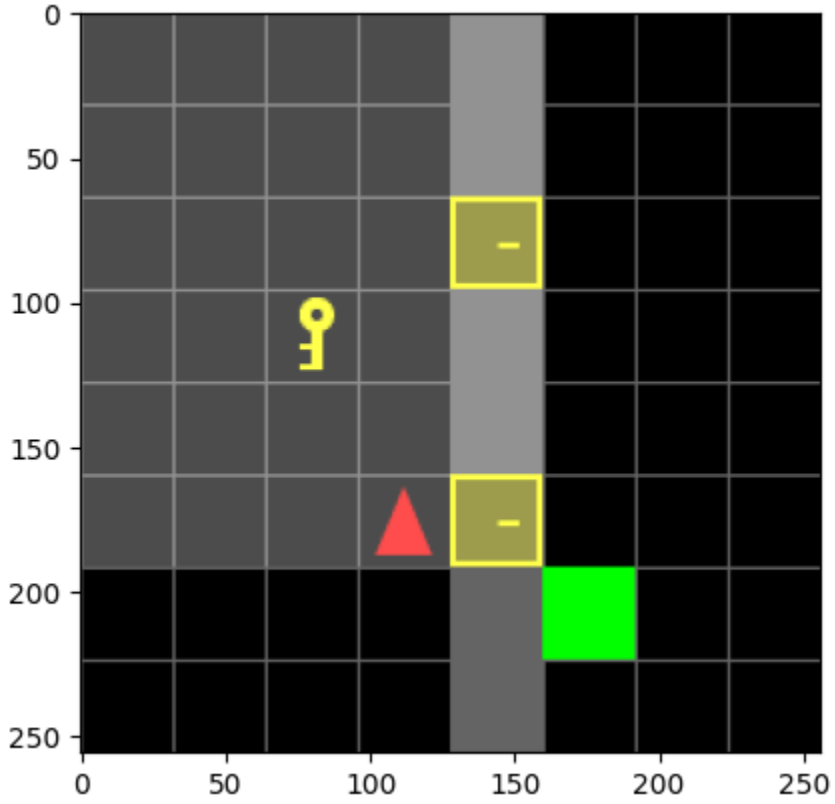

0
50
100
150
200
250
0
50
100
150
200
250

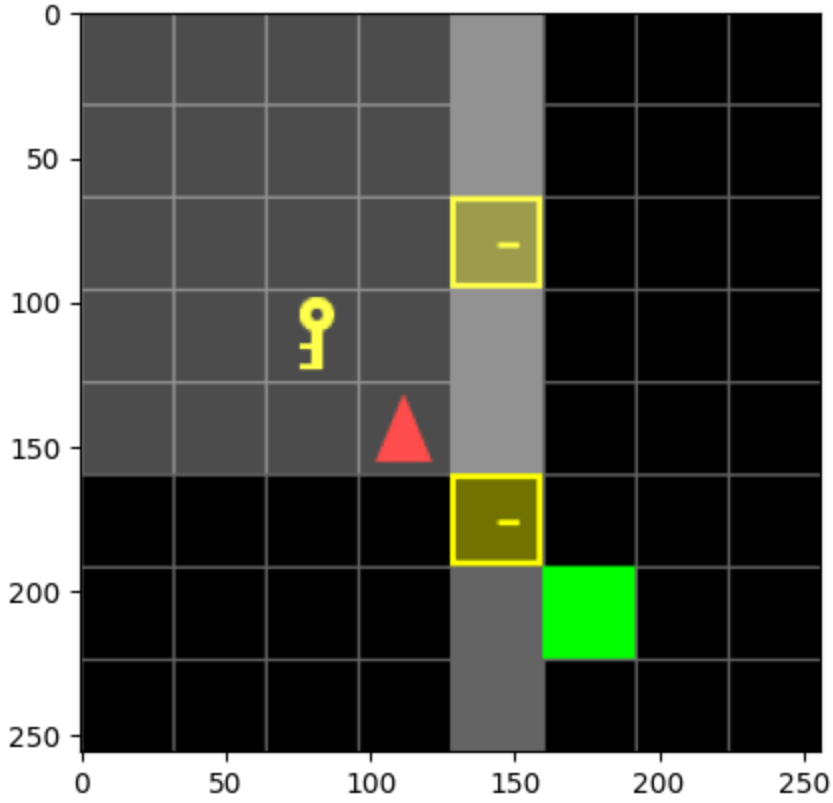

0
50
100
150
200
250
0
50
100
150
200
250

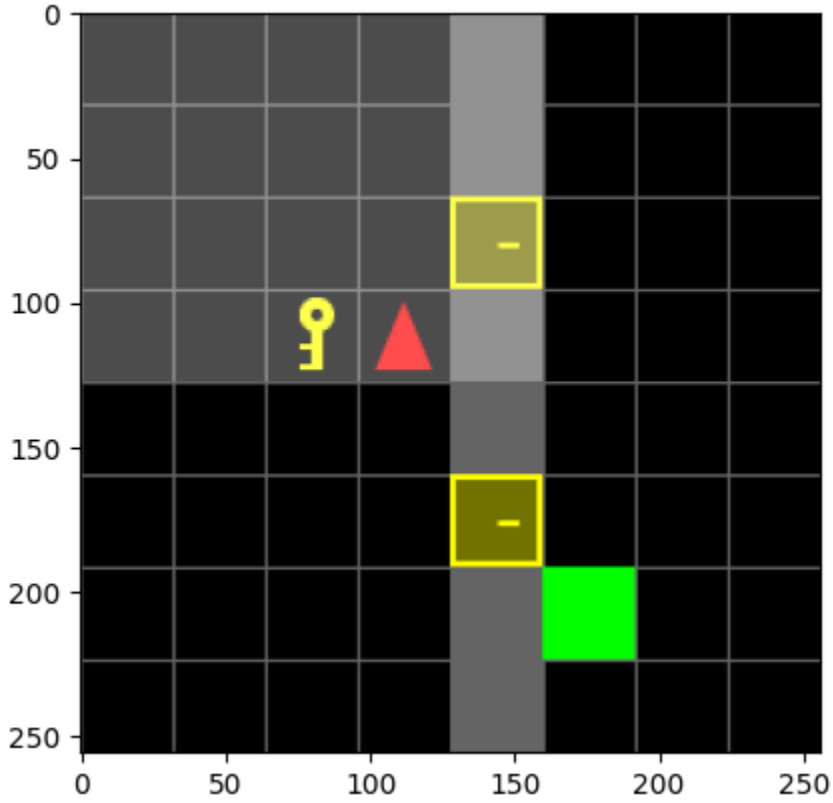

0
50
100
150
200
250
0
50
100
150
200
250

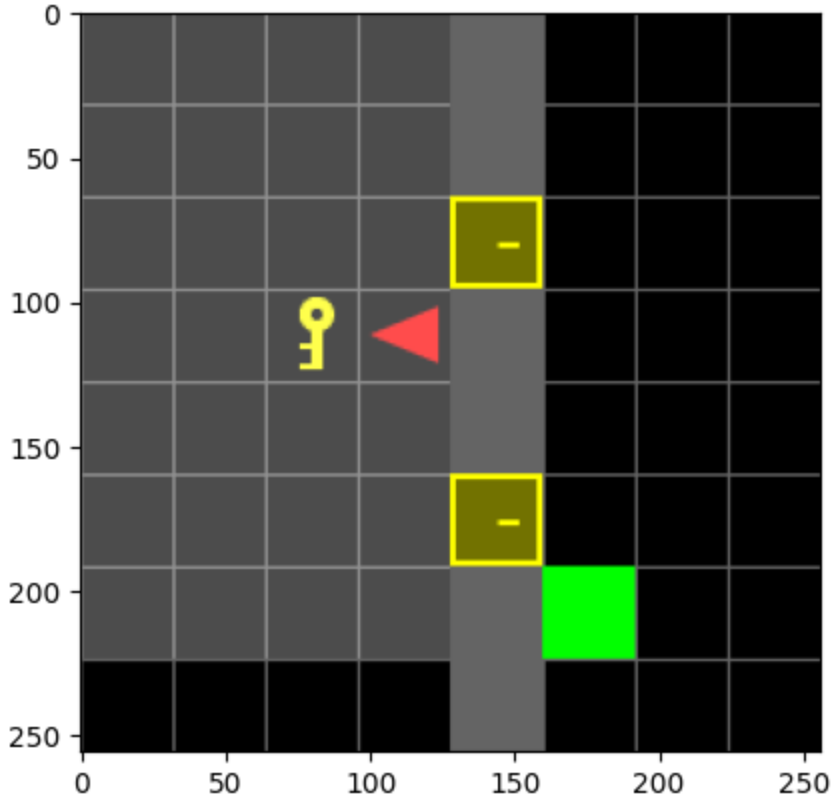
DoorKey-8x8-24.env

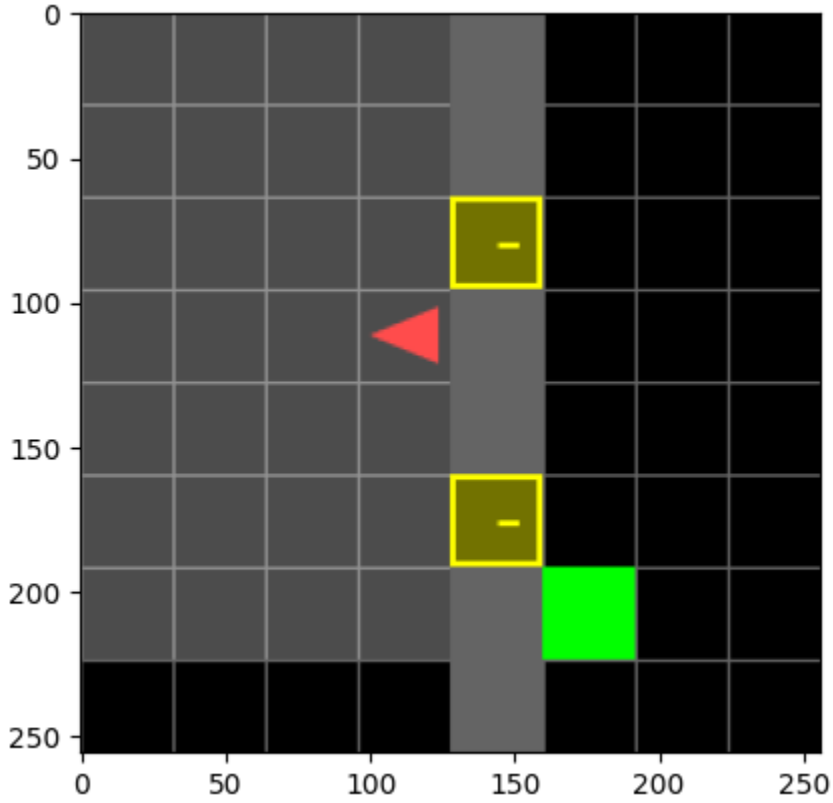
Optimal Sequence - [0, 0, 1, 3, 1, 0, 0, 1, 4, 0, 0, 2, 0]

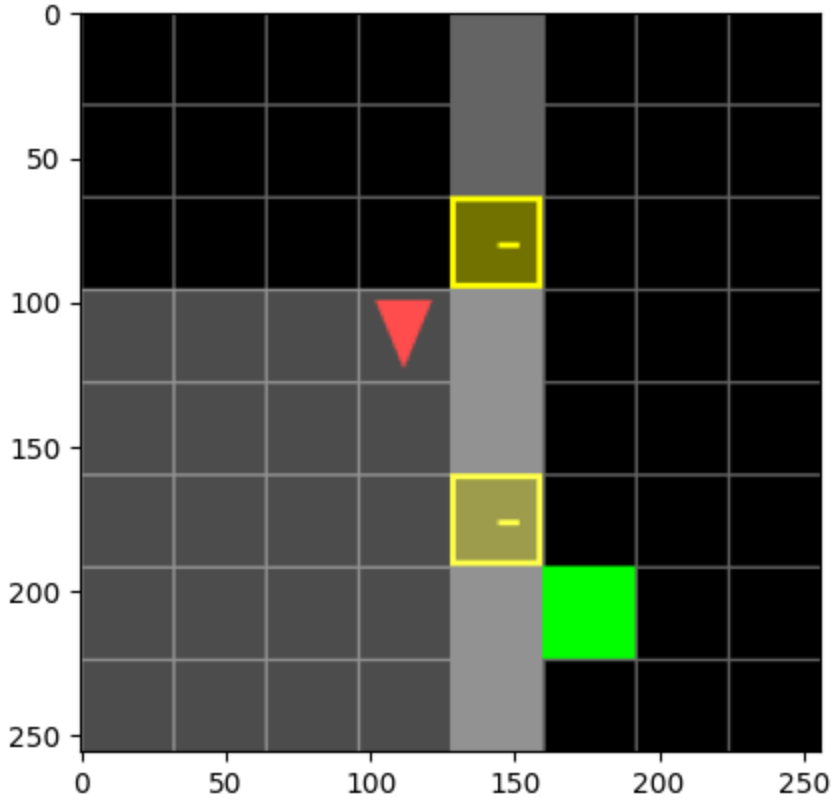


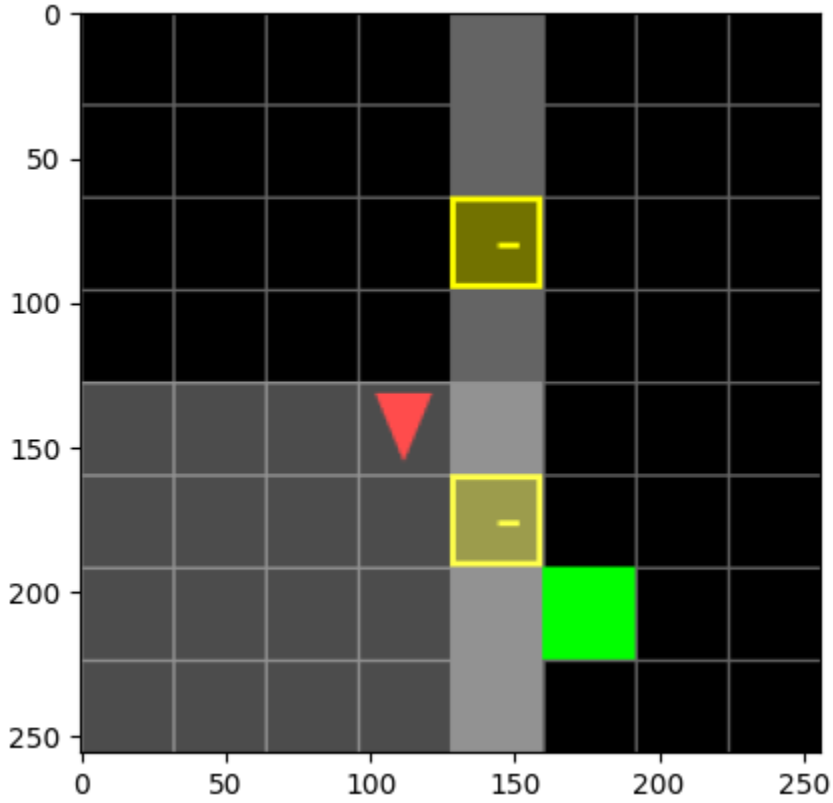

0
50
100
150
200
250
0
50
100
150
200
250

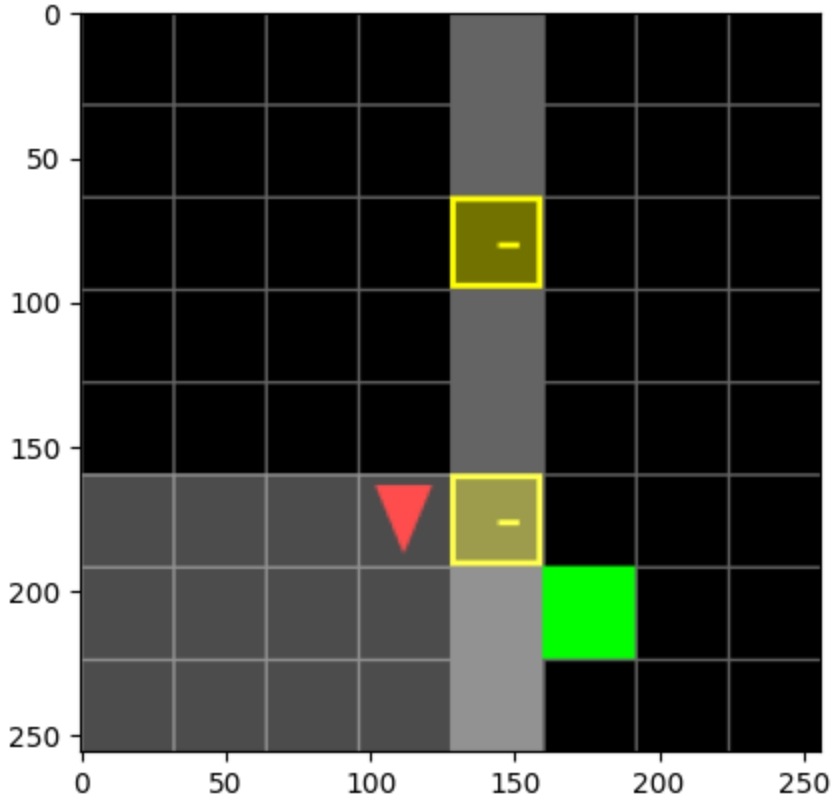

0
50
100
150
200
250
0
50
100
150
200
250

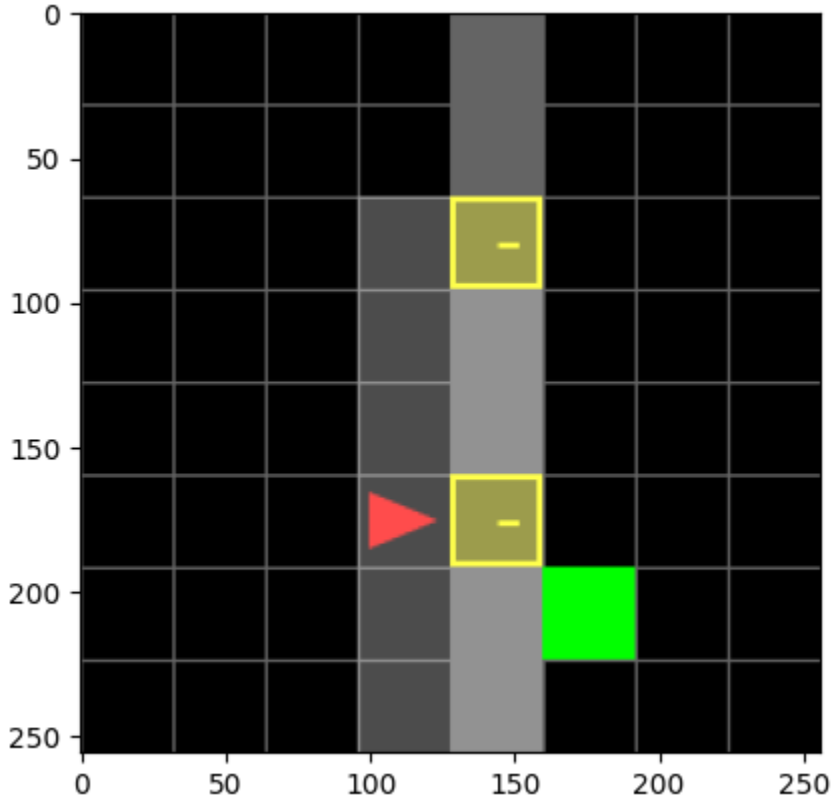

0
50
100
150
200
250
0
50
100
150
200
250

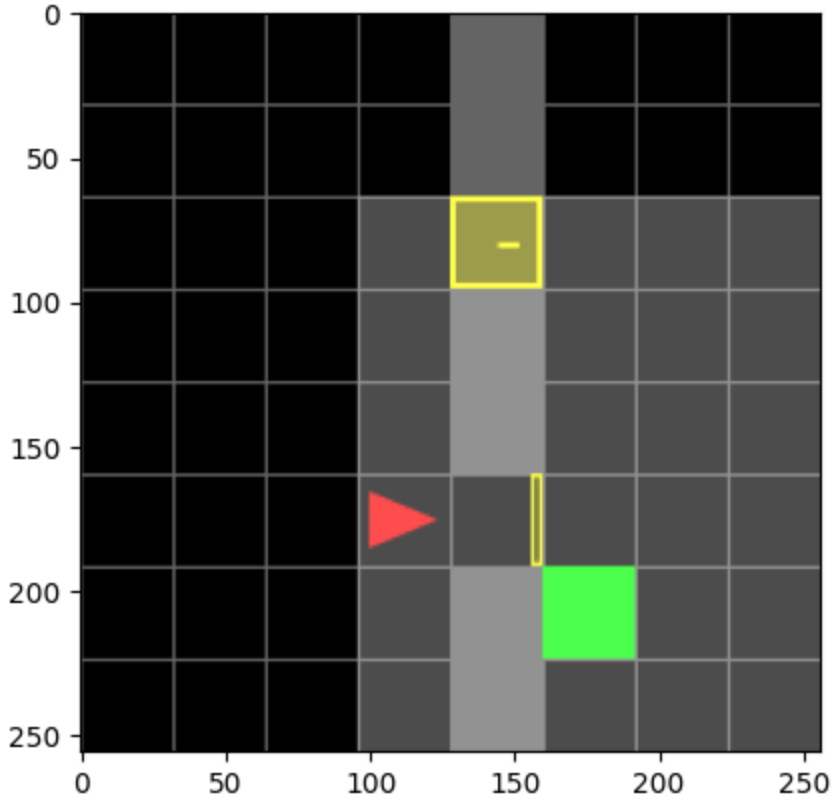

0
50
100
150
200
250
0
50
100
150
200
250

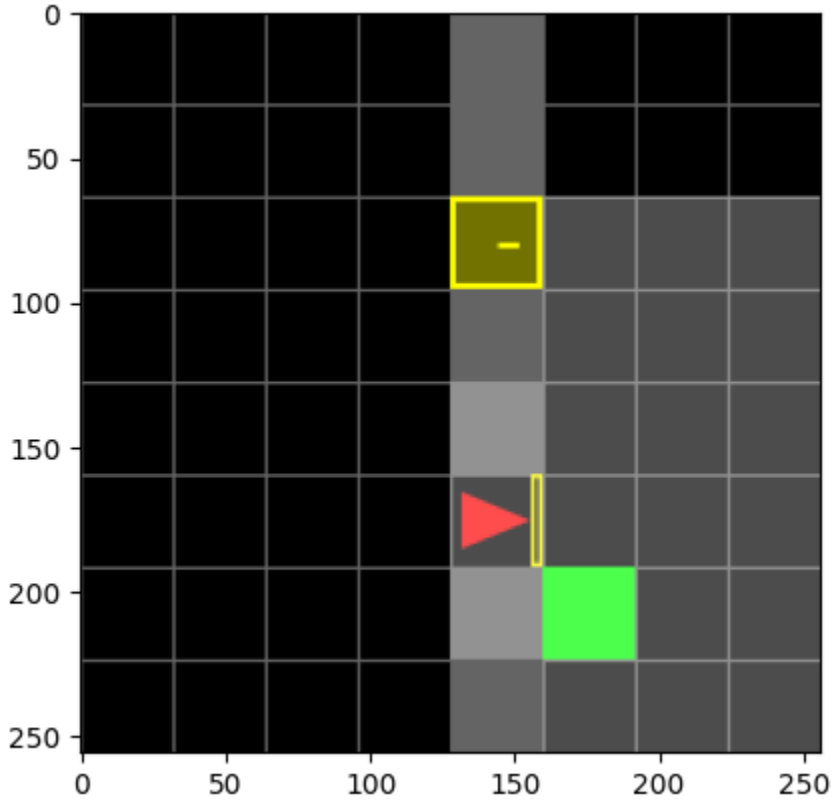

0
50
100
150
200
250
0
50
100
150
200
250

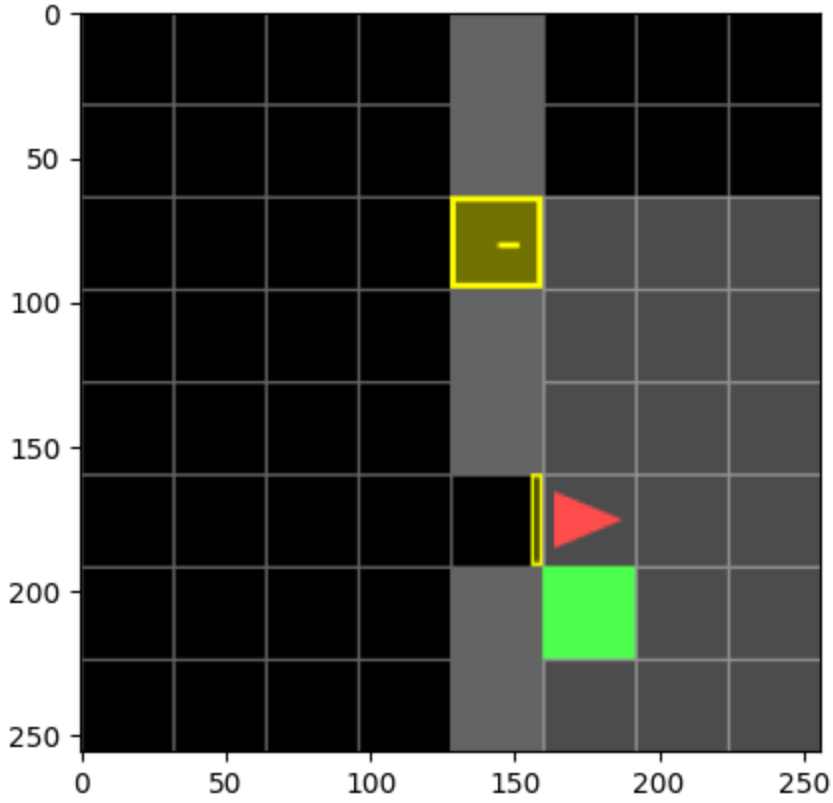

0
50
100
150
200
250
0
50
100
150
200
250

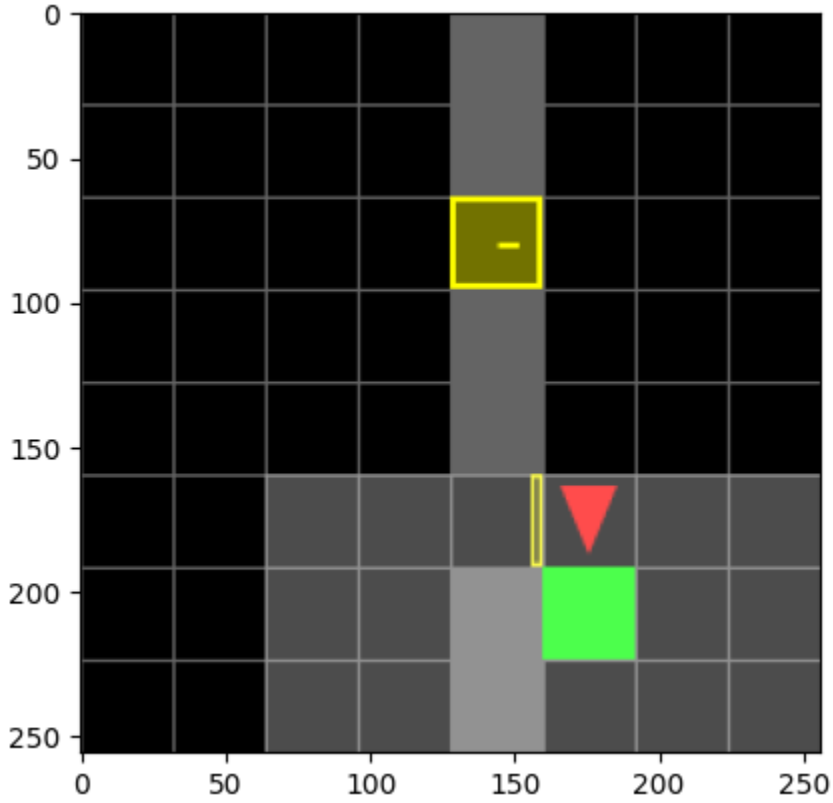

0
50
100
150
200
250
0
50
100
150
200
250

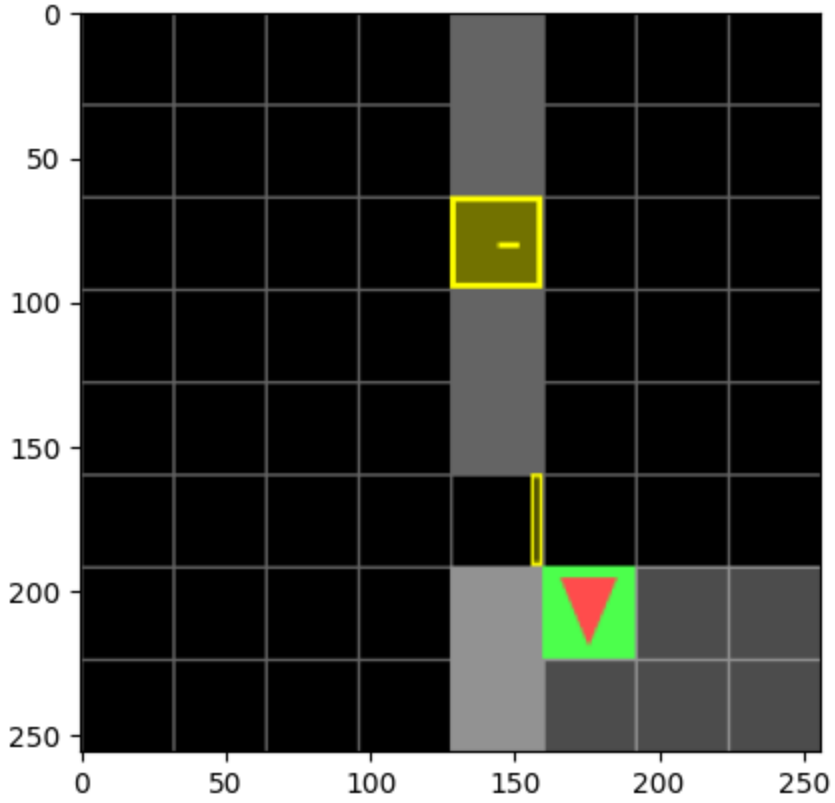

0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250