Treasure Hunt Game Notebook

Read and Review Your Starter Code

The theme of this project is a popular treasure hunt game in which the player needs to find the treasure before the pirate does. While you will not be developing the entire game, you will write the part of the game that represents the intelligent agent, which is a pirate in this case. The pirate will try to find the optimal path to the treasure using deep Q-learning.

To Begin: Use this **TreasureHuntGame_starterCode.ipynb** file to complete your assignment.

You have been provided with two Python classes and this notebook to help you with this assignment. The first class, **TreasureMaze.py**, represents the environment, which includes a maze object defined as a matrix. The second class, **GameExperience.py**, stores the episodes – that is, all the states that come in between the initial state and the terminal state. This is later used by the agent for learning by experience, called "exploration". This notebook shows how to play a game. Your task is to complete the deep Q-learning implementation in the qtrain() function for which a skeleton implementation has been provided.

NOTE: The code block you will need to complete will have **#TODO** as a header.
First, read and review the next few code and instruction blocks to understand the code that you have been given.

Installations The following command will install the necessary Python libraries to necessary to run this application. If you see a "[notice] A new release of pip is available: 23.1.2 -> 25.2" at the end of the installation, you may disregard that statement.

In [2]:
```
!pip install -r requirements.txt
```

```
Requirement already satisfied: numpy==1.26.4 in /home/codio/.pyenv/versions/3.11.9/l
ib/python3.11/site-packages (from -r requirements.txt (line 1)) (1.26.4)
Requirement already satisfied: keras in /home/codio/.pyenv/versions/3.11.9/lib/pytho
n3.11/site-packages (from -r requirements.txt (line 2)) (3.4.1)
Requirement already satisfied: absl-py in /home/codio/.pyenv/versions/3.11.9/lib/pyt
hon3.11/site-packages (from keras->-r requirements.txt (line 2)) (2.1.0)
Requirement already satisfied: rich in /home/codio/.pyenv/versions/3.11.9/lib/python
3.11/site-packages (from keras->-r requirements.txt (line 2)) (13.7.1)
Requirement already satisfied: namex in /home/codio/.pyenv/versions/3.11.9/lib/pytho
n3.11/site-packages (from keras->-r requirements.txt (line 2)) (0.0.8)
Requirement already satisfied: h5py in /home/codio/.pyenv/versions/3.11.9/lib/python
3.11/site-packages (from keras->-r requirements.txt (line 2)) (3.11.0)
Requirement already satisfied: optree in /home/codio/.pyenv/versions/3.11.9/lib/pyth
on3.11/site-packages (from keras->-r requirements.txt (line 2)) (0.12.1)
Requirement already satisfied: ml-dtypes in /home/codio/.pyenv/versions/3.11.9/lib/p
ython3.11/site-packages (from keras->-r requirements.txt (line 2)) (0.4.0)
Requirement already satisfied: packaging in /home/codio/.pyenv/versions/3.11.9/lib/p
ython3.11/site-packages (from keras->-r requirements.txt (line 2)) (24.1)
Requirement already satisfied: typing-extensions>=4.5.0 in /home/codio/.pyenv/versio
ns/3.11.9/lib/python3.11/site-packages (from optree->keras->-r requirements.txt (lin
e 2)) (4.12.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in /home/codio/.pyenv/versions/
3.11.9/lib/python3.11/site-packages (from rich->keras->-r requirements.txt (line 2))
(3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /home/codio/.pyenv/version
s/3.11.9/lib/python3.11/site-packages (from rich->keras->-r requirements.txt (line
2)) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in /home/codio/.pyenv/versions/3.11.9/lib/
python3.11/site-packages (from markdown-it-py>=2.2.0->rich->keras->-r requirements.t
xt (line 2)) (0.1.2)
```

In [2]:
```
pip install --upgrade pip
```

```
Requirement already satisfied: pip in /home/codio/.pyenv/versions/3.11.9/lib/python
3.11/site-packages (24.1.2)
Collecting pip
  Downloading pip-25.3-py3-none-any.whl.metadata (4.7 kB)
Downloading pip-25.3-py3-none-any.whl (1.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 10.4 MB/s eta 0:00:0000:0100:
01
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 24.1.2
    Uninstalling pip-24.1.2:
      Successfully uninstalled pip-24.1.2
Successfully installed pip-25.3
Note: you may need to restart the kernel to use updated packages.
```

Tensorflow CPU Acceleration Warning

GPU/CUDA/Memory Warnings/Errors: You may receive some errors referencing that GPUs will not be used, CUDA could not be found, or free system memory allocation errors. These and a few others, are standard errors that can be ignored here as they are

environment based.

Example messages:

- oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders
- WARNING: All log messages before absl::InitializeLog() is called are written to STDERR

In [3]:
```
from __future__ import print_function
import os, sys, time, datetime, json, random
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import clone_model
from keras.models import Sequential
from keras.layers import Dense, Activation, PReLU
from keras.optimizers import SGD , Adam, RMSprop
import matplotlib.pyplot as plt
from TreasureMaze import TreasureMaze
from GameExperience import GameExperience
%matplotlib inline
```

Maze Object Generation

NOTE: The following code block contains an 8x8 matrix that will be used as a maze object:

In [4]:
```
maze = np.array([
    [ 1.,  0.,  1.,  1.,  1.,  1.,  1.,  1.],
    [ 1.,  0.,  1.,  1.,  1.,  0.,  1.,  1.],
    [ 1.,  1.,  1.,  1.,  0.,  1.,  0.,  1.],
    [ 1.,  1.,  1.,  0.,  1.,  1.,  1.,  1.],
    [ 1.,  1.,  0.,  1.,  1.,  1.,  1.,  1.],
    [ 1.,  1.,  1.,  0.,  1.,  0.,  0.,  0.],
    [ 1.,  1.,  1.,  0.,  1.,  1.,  1.,  1.],
    [ 1.,  1.,  1.,  1.,  0.,  1.,  1.,  1.]
])
```

Helper Functions and Global Variables

This **show()** helper function allows a visual representation of the maze object:

In [5]:
```
def show(qmaze):
    plt.grid('on')
    nrows, ncols = qmaze.maze.shape
    ax = plt.gca()
```

```
    ax.set_xticks(np.arange(0.5, nrows, 1))
    ax.set_yticks(np.arange(0.5, ncols, 1))
    ax.set_xticklabels([])
    ax.set_yticklabels([])
    canvas = np.copy(qmaze.maze)
    for row,col in qmaze.visited:
        canvas[row,col] = 0.6
    pirate_row, pirate_col, _ = qmaze.state
    canvas[pirate_row, pirate_col] = 0.3   # pirate cell
    canvas[nrows-1, ncols-1] = 0.9 # treasure cell
    img = plt.imshow(canvas, interpolation='none', cmap='gray')
    return img
```

The **pirate agent** can move in four directions: left, right, up, and down.

> **Note:** While the agent primarily learns by experience through exploitation, often, the agent can choose to explore the environment to find previously undiscovered paths. This is called "exploration" and is defined by epsilon. This value is the **EXPLORATION** values from the Cartpole assignment. The hyperparameters are provided here and used in the **qtrain()** method. You are encouraged to try various values for the exploration factor and see how the algorithm performs.

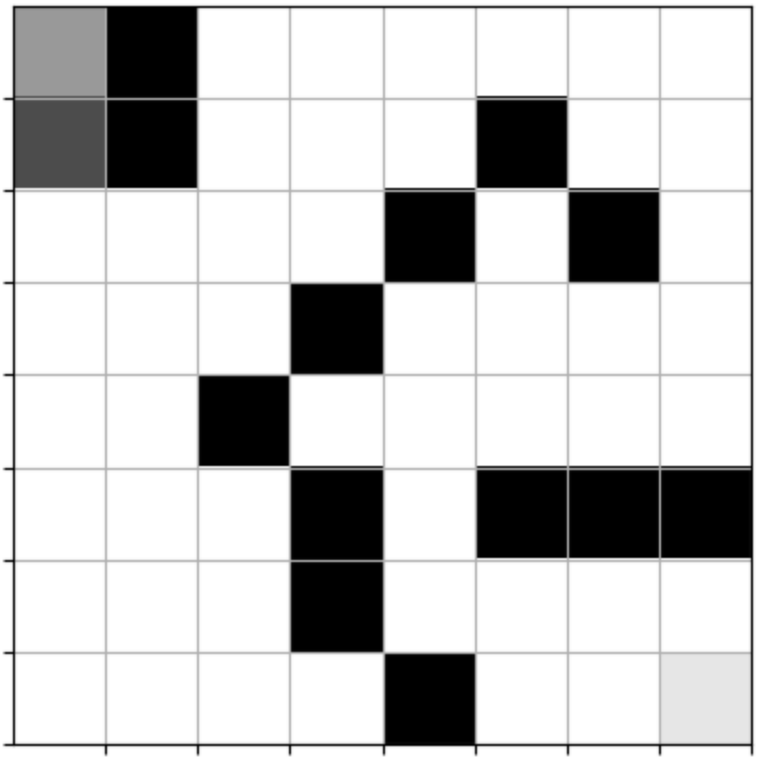
In [6]:
```
LEFT = 0
UP = 1
RIGHT = 2
DOWN = 3


# Exploration factor
epsilon = 1.0
epsilon_min = 0.05
epsilon_decay = 0.995
patience = 10

# Actions dictionary
actions_dict = {
    LEFT: 'left',
    UP: 'up',
    RIGHT: 'right',
    DOWN: 'down',
}

num_actions = len(actions_dict)
```

The sample code block and output below show creating a maze object and performing one action (DOWN), which returns the reward. The resulting updated environment is visualized.

In [7]:
```
qmaze = TreasureMaze(maze)
canvas, reward, game_over = qmaze.act(DOWN)
print("reward=", reward)
show(qmaze)
```

```
reward= -0.04
```

Out[7]: `<matplotlib.image.AxesImage at 0x7d36757bd390>`



NOTE: This **play_game()** function simulates a full game based on the provided trained model. The other parameters include the TreasureMaze object, the starting position of the pirate and max amount of steps to make sure the code does not get stuck in a loop.

In [8]:
```python
def play_game(model, qmaze, pirate_cell, max_steps=None):
    qmaze.reset(pirate_cell)
    envstate = qmaze.observe()
    steps = 0
    if max_steps is None:
        max_steps = qmaze.maze.size * 4  # safety cutoff

    while steps < max_steps:
        state = np.asarray(envstate, dtype=np.float32)
        if state.ndim == 1:
            state = np.expand_dims(state, axis=0)

        q_values = model(state, training=False).numpy()
        action = np.argmax(q_values[0])

        envstate, reward, game_status = qmaze.act(action)
        steps += 1

        if game_status == 'win':
            return True
        elif game_status == 'lose':
            return False
```

```
    return False  # timed out with no result
```

Note: This **completion_check()** function helps you to determine whether the pirate can win any game at all. If your maze is not well designed, the pirate may not win any game at all. In this case, your training would not yield any result. The provided maze in this notebook ensures that there is a path to win and you can run this method to check.

In [9]:
```
def completion_check(model, maze_or_qmaze, max_steps=None):
    # Accept either raw numpy maze or TreasureMaze instance
    if isinstance(maze_or_qmaze, TreasureMaze):
        qmaze = maze_or_qmaze
    else:
        qmaze = TreasureMaze(maze_or_qmaze)

    for cell in qmaze.free_cells:
        if not qmaze.valid_actions(cell):
            continue
        if not play_game(model, qmaze, cell, max_steps=max_steps):
            return False
    return True
```

Note: The **build_model()** function in the block below will build the neural network model. Review the code and note the number of layers, as well as the activation, optimizer, and loss functions that are used to train the model.

In [10]:
```
def build_model(maze):
    model = Sequential()
    model.add(Dense(maze.size, input_shape=(maze.size,)))
    model.add(PReLU())
    model.add(Dense(maze.size))
    model.add(PReLU())
    model.add(Dense(num_actions))
    model.compile(optimizer='adam', loss='mse')
    return model
```

Note: This **train_step()** helper function in the block below is used to help predict Q-values (quality values) in the current modelto see how good each action is in a given state and improve the Q-network by reducing the gap between what is predicted and what should have been predicted.

If you're interested in reading up on the *@tf.function*, which is a decorator for Tensorflow to run this code into a TensorFlow computation graph, please refer to this link: https://www.tensorflow.org/guide/intro_to_graphs

Tensorflow GPU Warning

> You will see a **warning in red** "INTERNAL: CUDA Runtime error: Failed call to cudaGetRuntimeVersion: Error loading CUDA libraries. GPU will not be used.". This is simply coming from **Tensorflow skipping using GPU for this assignment.**

In [11]:
```
loss_fn = tf.keras.losses.MeanSquaredError()
optimizer = tf.keras.optimizers.Adam()

@tf.function
def train_step(x, y):
    with tf.GradientTape() as tape:
        q_values = model(x, training=True)
        loss = loss_fn(y, q_values)
    grads = tape.gradient(loss, model.trainable_variables)
    optimizer.apply_gradients(zip(grads, model.trainable_variables))
    return loss
```

```
WARNING: All log messages before absl::InitializeLog() is called are written to STDE
RR
I0000 00:00:1764052025.677538     354 cuda_executor.cc:1015] successful NUMA node re
ad from SysFS had negative value (-1), but there must be at least one NUMA node, so
returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Do
cumentation/ABI/testing/sysfs-bus-pci#L344-L355
I0000 00:00:1764052026.370819     354 cuda_executor.cc:1015] successful NUMA node re
ad from SysFS had negative value (-1), but there must be at least one NUMA node, so
returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Do
cumentation/ABI/testing/sysfs-bus-pci#L344-L355
I0000 00:00:1764052026.374277     354 cuda_executor.cc:1015] successful NUMA node re
ad from SysFS had negative value (-1), but there must be at least one NUMA node, so
returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Do
cumentation/ABI/testing/sysfs-bus-pci#L344-L355
I0000 00:00:1764052026.379170     354 cuda_executor.cc:1015] successful NUMA node re
ad from SysFS had negative value (-1), but there must be at least one NUMA node, so
returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Do
cumentation/ABI/testing/sysfs-bus-pci#L344-L355
I0000 00:00:1764052026.382407     354 cuda_executor.cc:1015] successful NUMA node re
ad from SysFS had negative value (-1), but there must be at least one NUMA node, so
returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Do
cumentation/ABI/testing/sysfs-bus-pci#L344-L355
I0000 00:00:1764052026.385560     354 cuda_executor.cc:1015] successful NUMA node re
ad from SysFS had negative value (-1), but there must be at least one NUMA node, so
returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Do
cumentation/ABI/testing/sysfs-bus-pci#L344-L355
I0000 00:00:1764052026.621909     354 cuda_executor.cc:1015] successful NUMA node re
ad from SysFS had negative value (-1), but there must be at least one NUMA node, so
returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Do
cumentation/ABI/testing/sysfs-bus-pci#L344-L355
I0000 00:00:1764052026.623857     354 cuda_executor.cc:1015] successful NUMA node re
ad from SysFS had negative value (-1), but there must be at least one NUMA node, so
returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Do
cumentation/ABI/testing/sysfs-bus-pci#L344-L355
I0000 00:00:1764052026.625670     354 cuda_executor.cc:1015] successful NUMA node re
ad from SysFS had negative value (-1), but there must be at least one NUMA node, so
returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Do
cumentation/ABI/testing/sysfs-bus-pci#L344-L355
```

#TODO: Complete the Q-Training Algorithm Code Block

This is your deep Q-learning implementation. The goal of your deep Q-learning implementation is to find the best possible navigation sequence that results in reaching the treasure cell while maximizing the reward. In your implementation, you need to determine the optimal number of epochs to achieve a 100% win rate.

Pseudocode:

For each epoch: Reset the environment at a random starting cell agent_cell = randomly

select a free cell
Hint: Review the reset method in the TreasureMaze.py class.

```
    Set the initial environment state
    env_state should reference the environment's current state
    Hint: Review the observe method in the TreasureMaze.py class.

    While game status is not game over:
       previous_envstate = env_state
        Decide on an action:
            - If possible, take a random valid exploration action
and
              randomly choose action (left, right, up, down)
              and assign it to an action variable
            - Else, pick the best exploitation action from the
model and assign it to an action variable
              Hint: Review the predict method in the
GameExperience.py class.

       Retrieve the values below from the act() method.
       env_state, reward, game_status = qmaze.act(action)
       Hint: Review the act method in the TreasureMaze.py class.

        Track the wins and losses from the game_status using
win_history

       Store the episode below in the Experience replay object
       episode = [previous_envstate, action, reward, envstate,
game_status]
       Hint: Review the remember method in the GameExperience.py
class.

       Train neural network model and evaluate loss
       Hint: Call GameExperience.get_data to retrieve training data
(input and target)
       and pass to the train_step method and assign it to
batch_loss and append to the loss variable

  If the win rate is above the threshold and your model passes the
completion check, that would be your epoch.
```

Note: A 100% win rate **DOES NOT EXPLICITLY MEAN** that you have solved the maze. It simply indicates that during the last evaluation, the pirate *happened* to get to the treasure. Be sure to utilise the **completion_check()** function to validate your pirate found the treasure at every starting point and consistently!

You will need to complete the section starting with #START_HERE. Please use the pseudocode above as guidance.

In [26]:

```
# Step 1: Build the model
from tensorflow import keras
from tensorflow.keras import layers
import tensorflow as tf
import numpy as np
import datetime
import random

def build_model(maze, learning_rate=0.001):
    state_size = maze.size
    num_actions = 4

    model = keras.Sequential([
        keras.Input(shape=(state_size,)),
        layers.Dense(32, activation='relu'),
        layers.Dense(32, activation='relu'),
        layers.Dense(num_actions, activation='linear')
    ])

    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate=learning_rate),
        loss='mse'
    )
    return model

# Step 2: Training step wrapper

def train_step(model, inputs, targets):
    return model.train_on_batch(inputs, targets)

# Step 3: qtrain loop
def qtrain(model, maze, **opt):
    global epsilon

    n_epoch = opt.get('n_epoch', 15000)
    max_memory = opt.get('max_memory', 1000)
    data_size = opt.get('data_size', 50)
    target_update_freq = opt.get('target_update_freq', 50)
    max_steps_per_episode = opt.get('max_steps', 200)

    start_time = datetime.datetime.now()
    qmaze = TreasureMaze(maze)

    target_model = clone_model(model)
    target_model.set_weights(model.get_weights())

    experience = GameExperience(model, target_model, max_memory=max_memory)

    win_history = []
    hsize = qmaze.maze.size // 2
    win_rate = 0.0

    loss = 0.0
    n_episodes = 0
```

```
    for epoch in range(n_epoch):
        start_cell = random.choice(qmaze.free_cells)
        qmaze.reset(start_cell)
        env_state = qmaze.observe()
        game_status = 'not_over'
        steps = 0

        max_steps_per_episode = 150
        while game_status == 'not_over' and steps < max_steps_per_episode:
            steps += 1
            previous_env_state = env_state

            valid_actions = qmaze.valid_actions()
            if (np.random.rand() < epsilon) and valid_actions:
                action = random.choice(valid_actions)
            else:
                q_values = experience.predict(previous_env_state)
                masked_q = np.full_like(q_values, -np.inf, dtype=np.float32)
                masked_q[valid_actions] = q_values[valid_actions]
                action = int(np.argmax(masked_q))

            env_state, reward, game_status = qmaze.act(action)
            done = (game_status != 'not_over')

            if done or steps >= max_steps_per_episode:
                n_episodes += 1
                if game_status == 'win':
                    win_history.append(1)
                else:
                    win_history.append(0)

            experience.remember([previous_env_state, action, reward, env_state, don

            if len(experience.memory) >= data_size and steps % 4 == 0:
                inputs, targets = experience.get_data(batch_size=data_size)
                batch_loss = train_step(model, inputs, targets)
                loss = 0.9 * loss + 0.1 * float(batch_loss)

        win_rate = sum(win_history[-hsize:]) / hsize if len(win_history) >= hsize e

        if (epoch + 1) % target_update_freq == 0:
            target_model.set_weights(model.get_weights())

        dt = datetime.datetime.now() - start_time
        t = format_time(dt.total_seconds())
        print("Epoch: {:03d}/{:d} | Loss: {:.4f} | Episodes: {:d} | Win count: {:d}
            epoch, n_epoch-1, loss, n_episodes, sum(win_history), win_rate, t))

        if win_rate > 0.9:
            epsilon = 0.05
        else:
            epsilon = max(epsilon * epsilon_decay, epsilon_min)

        if win_rate >= 0.999 and completion_check(model, maze):
            print(f"Reached 100% win rate at epoch {epoch}")
            break
```

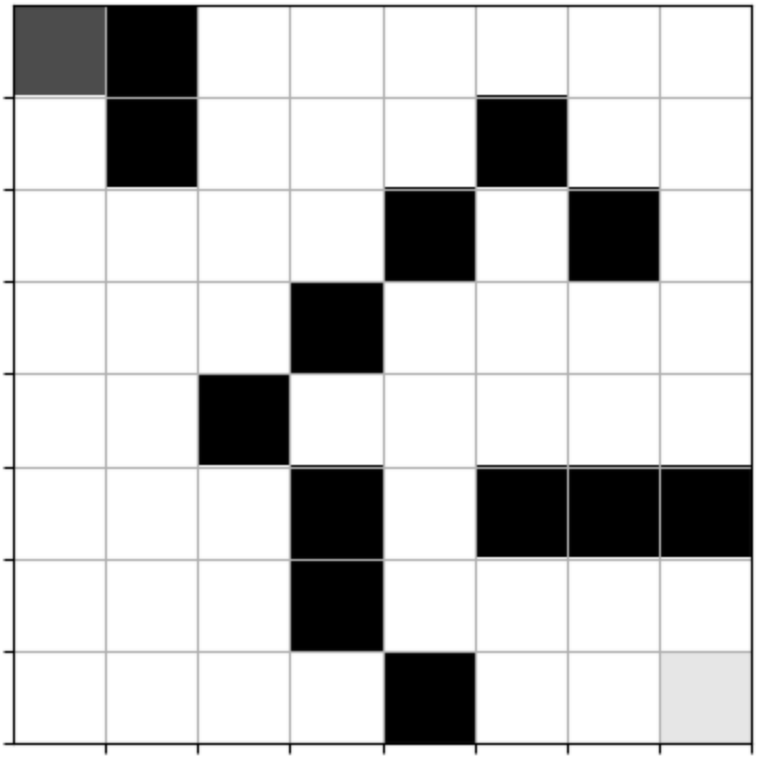
```
    total_time = format_time((datetime.datetime.now() - start_time).total_seconds())
    print("Training complete in:", total_time)


def format_time(seconds):
    if seconds < 400:
        return "%.1f seconds" % float(seconds)
    elif seconds < 4000:
        return "%.2f minutes" % (seconds / 60.0)
    else:
        return "%.2f hours" % (seconds / 3600.0)
```

Test Your Model

Now we will start testing the deep Q-learning implementation. To begin, select **Cell**, then **Run All** from the menu bar. This will run your notebook. As it runs, you should see output begin to appear beneath the next few cells. The code below creates an **instance** of TreasureMaze. This does not show your actual training done.

In [27]:
```
qmaze = TreasureMaze(maze)
show(qmaze)
```

Out[27]:
```
<matplotlib.image.AxesImage at 0x7d366c232d50>
```



In the next code block, you will build your model using the **build_model** function and train it using deep Q-learning. Note: This step takes several minutes to fully run.

WARNING If you did not attempt the assignment, the code **will** error out at this section.

In [29]:
```
model = build_model(maze)
qtrain(model, maze, n_epoch=100, max_memory=2*maze.size, data_size=8, target_update
```

```
Epoch: 000/99 | Loss: 0.0030 | Episodes: 1 | Win count: 0 | Win rate: 0.000 | time:
5.8 seconds
Epoch: 001/99 | Loss: 0.0028 | Episodes: 2 | Win count: 1 | Win rate: 0.000 | time:
6.3 seconds
Epoch: 002/99 | Loss: 0.0037 | Episodes: 3 | Win count: 2 | Win rate: 0.000 | time:
9.1 seconds
Epoch: 003/99 | Loss: 0.0040 | Episodes: 4 | Win count: 2 | Win rate: 0.000 | time:
13.2 seconds
Epoch: 004/99 | Loss: 0.0036 | Episodes: 5 | Win count: 3 | Win rate: 0.000 | time:
17.7 seconds
Epoch: 005/99 | Loss: 0.0036 | Episodes: 6 | Win count: 4 | Win rate: 0.000 | time:
17.9 seconds
Epoch: 006/99 | Loss: 0.0037 | Episodes: 7 | Win count: 5 | Win rate: 0.000 | time:
21.4 seconds
Epoch: 007/99 | Loss: 0.0041 | Episodes: 8 | Win count: 5 | Win rate: 0.000 | time:
26.2 seconds
Epoch: 008/99 | Loss: 0.0042 | Episodes: 9 | Win count: 6 | Win rate: 0.000 | time:
26.7 seconds
Epoch: 009/99 | Loss: 0.0041 | Episodes: 10 | Win count: 6 | Win rate: 0.000 | time:
31.9 seconds
Epoch: 010/99 | Loss: 0.0039 | Episodes: 11 | Win count: 7 | Win rate: 0.000 | time:
35.3 seconds
Epoch: 011/99 | Loss: 0.0040 | Episodes: 12 | Win count: 7 | Win rate: 0.000 | time:
41.1 seconds
Epoch: 012/99 | Loss: 0.0038 | Episodes: 13 | Win count: 7 | Win rate: 0.000 | time:
47.1 seconds
Epoch: 013/99 | Loss: 0.0038 | Episodes: 14 | Win count: 8 | Win rate: 0.000 | time:
47.1 seconds
Epoch: 014/99 | Loss: 0.0037 | Episodes: 15 | Win count: 9 | Win rate: 0.000 | time:
49.1 seconds
Epoch: 015/99 | Loss: 0.0037 | Episodes: 16 | Win count: 10 | Win rate: 0.000 | tim
e: 49.2 seconds
Epoch: 016/99 | Loss: 0.0037 | Episodes: 17 | Win count: 11 | Win rate: 0.000 | tim
e: 49.2 seconds
Epoch: 017/99 | Loss: 0.0038 | Episodes: 18 | Win count: 11 | Win rate: 0.000 | tim
e: 55.2 seconds
Epoch: 018/99 | Loss: 0.0037 | Episodes: 19 | Win count: 11 | Win rate: 0.000 | tim
e: 70.8 seconds
Epoch: 019/99 | Loss: 0.0035 | Episodes: 20 | Win count: 11 | Win rate: 0.000 | tim
e: 80.7 seconds
Epoch: 020/99 | Loss: 0.0035 | Episodes: 21 | Win count: 11 | Win rate: 0.000 | tim
e: 89.6 seconds
Epoch: 021/99 | Loss: 0.0035 | Episodes: 22 | Win count: 12 | Win rate: 0.000 | tim
e: 92.2 seconds
Epoch: 022/99 | Loss: 0.0034 | Episodes: 23 | Win count: 12 | Win rate: 0.000 | tim
e: 100.2 seconds
Epoch: 023/99 | Loss: 0.0033 | Episodes: 24 | Win count: 12 | Win rate: 0.000 | tim
e: 108.1 seconds
Epoch: 024/99 | Loss: 0.0032 | Episodes: 25 | Win count: 12 | Win rate: 0.000 | tim
e: 116.8 seconds
Epoch: 025/99 | Loss: 0.0031 | Episodes: 26 | Win count: 12 | Win rate: 0.000 | tim
e: 125.1 seconds
Epoch: 026/99 | Loss: 0.0030 | Episodes: 27 | Win count: 12 | Win rate: 0.000 | tim
e: 134.3 seconds
Epoch: 027/99 | Loss: 0.0029 | Episodes: 28 | Win count: 12 | Win rate: 0.000 | tim
e: 143.1 seconds
```

```
Epoch: 028/99 | Loss: 0.0028 | Episodes: 29 | Win count: 12 | Win rate: 0.000 | tim
e: 152.4 seconds
Epoch: 029/99 | Loss: 0.0027 | Episodes: 30 | Win count: 12 | Win rate: 0.000 | tim
e: 162.0 seconds
Epoch: 030/99 | Loss: 0.0027 | Episodes: 31 | Win count: 12 | Win rate: 0.000 | tim
e: 172.4 seconds
Epoch: 031/99 | Loss: 0.0026 | Episodes: 32 | Win count: 12 | Win rate: 0.375 | tim
e: 182.4 seconds
Epoch: 032/99 | Loss: 0.0025 | Episodes: 33 | Win count: 12 | Win rate: 0.375 | tim
e: 192.3 seconds
Epoch: 033/99 | Loss: 0.0025 | Episodes: 34 | Win count: 13 | Win rate: 0.375 | tim
e: 194.6 seconds
Epoch: 034/99 | Loss: 0.0026 | Episodes: 35 | Win count: 13 | Win rate: 0.344 | tim
e: 204.4 seconds
Epoch: 035/99 | Loss: 0.0025 | Episodes: 36 | Win count: 14 | Win rate: 0.375 | tim
e: 212.0 seconds
Epoch: 036/99 | Loss: 0.0026 | Episodes: 37 | Win count: 14 | Win rate: 0.344 | tim
e: 222.5 seconds
Epoch: 037/99 | Loss: 0.0026 | Episodes: 38 | Win count: 14 | Win rate: 0.312 | tim
e: 233.4 seconds
Epoch: 038/99 | Loss: 0.0026 | Episodes: 39 | Win count: 14 | Win rate: 0.281 | tim
e: 244.6 seconds
Epoch: 039/99 | Loss: 0.0026 | Episodes: 40 | Win count: 14 | Win rate: 0.281 | tim
e: 256.5 seconds
Epoch: 040/99 | Loss: 0.0026 | Episodes: 41 | Win count: 14 | Win rate: 0.250 | tim
e: 268.2 seconds
Epoch: 041/99 | Loss: 0.0026 | Episodes: 42 | Win count: 15 | Win rate: 0.281 | tim
e: 268.2 seconds
Epoch: 042/99 | Loss: 0.0026 | Episodes: 43 | Win count: 15 | Win rate: 0.250 | tim
e: 280.0 seconds
Epoch: 043/99 | Loss: 0.0025 | Episodes: 44 | Win count: 15 | Win rate: 0.250 | tim
e: 292.7 seconds
Epoch: 044/99 | Loss: 0.0025 | Episodes: 45 | Win count: 15 | Win rate: 0.250 | tim
e: 305.6 seconds
Epoch: 045/99 | Loss: 0.0025 | Episodes: 46 | Win count: 15 | Win rate: 0.219 | tim
e: 319.9 seconds
Epoch: 046/99 | Loss: 0.0025 | Episodes: 47 | Win count: 16 | Win rate: 0.219 | tim
e: 332.6 seconds
Epoch: 047/99 | Loss: 0.0025 | Episodes: 48 | Win count: 16 | Win rate: 0.188 | tim
e: 345.4 seconds
Epoch: 048/99 | Loss: 0.0025 | Episodes: 49 | Win count: 16 | Win rate: 0.156 | tim
e: 359.0 seconds
Epoch: 049/99 | Loss: 0.0025 | Episodes: 50 | Win count: 17 | Win rate: 0.188 | tim
e: 359.7 seconds
Epoch: 050/99 | Loss: 0.0025 | Episodes: 51 | Win count: 17 | Win rate: 0.188 | tim
e: 372.8 seconds
Epoch: 051/99 | Loss: 0.0025 | Episodes: 52 | Win count: 17 | Win rate: 0.188 | tim
e: 386.3 seconds
Epoch: 052/99 | Loss: 0.0025 | Episodes: 53 | Win count: 17 | Win rate: 0.188 | tim
e: 6.68 minutes
Epoch: 053/99 | Loss: 0.0025 | Episodes: 54 | Win count: 17 | Win rate: 0.156 | tim
e: 6.91 minutes
Epoch: 054/99 | Loss: 0.0025 | Episodes: 55 | Win count: 18 | Win rate: 0.188 | tim
e: 6.92 minutes
Epoch: 055/99 | Loss: 0.0024 | Episodes: 56 | Win count: 18 | Win rate: 0.188 | tim
e: 7.16 minutes
```

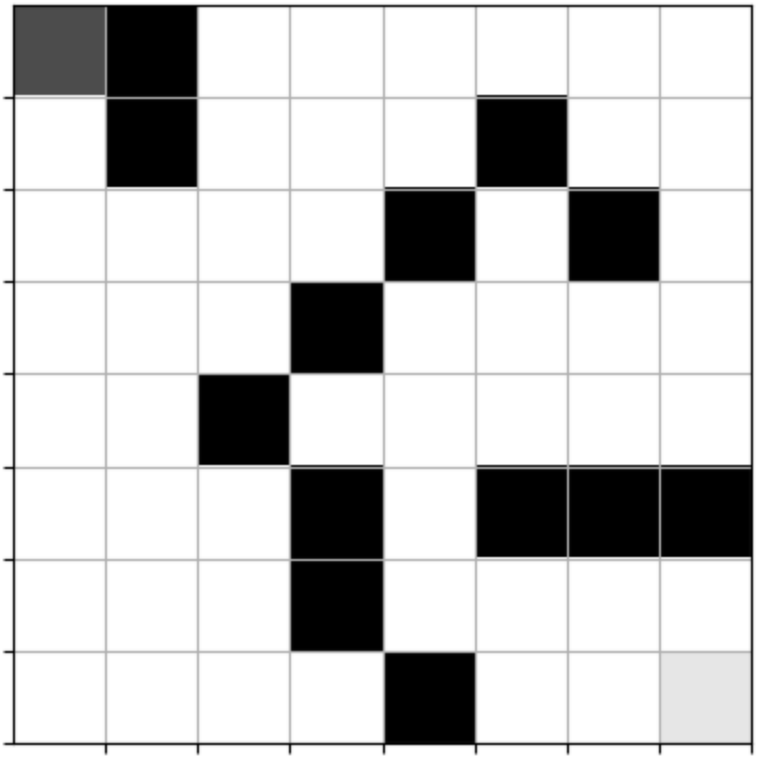
```
Epoch: 056/99 | Loss: 0.0024 | Episodes: 57 | Win count: 19 | Win rate: 0.219 | tim
e: 7.32 minutes
Epoch: 057/99 | Loss: 0.0024 | Episodes: 58 | Win count: 20 | Win rate: 0.250 | tim
e: 7.43 minutes
Epoch: 058/99 | Loss: 0.0024 | Episodes: 59 | Win count: 21 | Win rate: 0.281 | tim
e: 7.43 minutes
Epoch: 059/99 | Loss: 0.0024 | Episodes: 60 | Win count: 22 | Win rate: 0.312 | tim
e: 7.43 minutes
Epoch: 060/99 | Loss: 0.0025 | Episodes: 61 | Win count: 23 | Win rate: 0.344 | tim
e: 7.50 minutes
Epoch: 061/99 | Loss: 0.0028 | Episodes: 62 | Win count: 24 | Win rate: 0.375 | tim
e: 7.78 minutes
Epoch: 062/99 | Loss: 0.0030 | Episodes: 63 | Win count: 24 | Win rate: 0.375 | tim
e: 8.07 minutes
Epoch: 063/99 | Loss: 0.0030 | Episodes: 64 | Win count: 24 | Win rate: 0.375 | tim
e: 8.39 minutes
Epoch: 064/99 | Loss: 0.0029 | Episodes: 65 | Win count: 24 | Win rate: 0.375 | tim
e: 8.66 minutes
Epoch: 065/99 | Loss: 0.0029 | Episodes: 66 | Win count: 25 | Win rate: 0.375 | tim
e: 8.91 minutes
Epoch: 066/99 | Loss: 0.0029 | Episodes: 67 | Win count: 26 | Win rate: 0.406 | tim
e: 8.92 minutes
Epoch: 067/99 | Loss: 0.0029 | Episodes: 68 | Win count: 27 | Win rate: 0.406 | tim
e: 8.98 minutes
Epoch: 068/99 | Loss: 0.0031 | Episodes: 69 | Win count: 27 | Win rate: 0.406 | tim
e: 9.26 minutes
Epoch: 069/99 | Loss: 0.0031 | Episodes: 70 | Win count: 27 | Win rate: 0.406 | tim
e: 9.57 minutes
Epoch: 070/99 | Loss: 0.0031 | Episodes: 71 | Win count: 27 | Win rate: 0.406 | tim
e: 9.88 minutes
Epoch: 071/99 | Loss: 0.0030 | Episodes: 72 | Win count: 28 | Win rate: 0.438 | tim
e: 10.02 minutes
Epoch: 072/99 | Loss: 0.0031 | Episodes: 73 | Win count: 28 | Win rate: 0.438 | tim
e: 10.31 minutes
Epoch: 073/99 | Loss: 0.0031 | Episodes: 74 | Win count: 28 | Win rate: 0.406 | tim
e: 10.63 minutes
Epoch: 074/99 | Loss: 0.0031 | Episodes: 75 | Win count: 29 | Win rate: 0.438 | tim
e: 10.70 minutes
Epoch: 075/99 | Loss: 0.0032 | Episodes: 76 | Win count: 29 | Win rate: 0.438 | tim
e: 11.00 minutes
Epoch: 076/99 | Loss: 0.0031 | Episodes: 77 | Win count: 29 | Win rate: 0.438 | tim
e: 11.29 minutes
Epoch: 077/99 | Loss: 0.0031 | Episodes: 78 | Win count: 29 | Win rate: 0.438 | tim
e: 11.60 minutes
Epoch: 078/99 | Loss: 0.0031 | Episodes: 79 | Win count: 30 | Win rate: 0.438 | tim
e: 11.62 minutes
Epoch: 079/99 | Loss: 0.0031 | Episodes: 80 | Win count: 30 | Win rate: 0.438 | tim
e: 11.93 minutes
Epoch: 080/99 | Loss: 0.0031 | Episodes: 81 | Win count: 30 | Win rate: 0.438 | tim
e: 12.26 minutes
Epoch: 081/99 | Loss: 0.0031 | Episodes: 82 | Win count: 30 | Win rate: 0.406 | tim
e: 12.58 minutes
Epoch: 082/99 | Loss: 0.0031 | Episodes: 83 | Win count: 31 | Win rate: 0.438 | tim
e: 12.76 minutes
Epoch: 083/99 | Loss: 0.0031 | Episodes: 84 | Win count: 32 | Win rate: 0.469 | tim
e: 12.80 minutes
```

```
Epoch: 084/99 | Loss: 0.0031 | Episodes: 85 | Win count: 33 | Win rate: 0.500 | tim
e: 12.88 minutes
Epoch: 085/99 | Loss: 0.0032 | Episodes: 86 | Win count: 34 | Win rate: 0.531 | tim
e: 13.18 minutes
Epoch: 086/99 | Loss: 0.0032 | Episodes: 87 | Win count: 34 | Win rate: 0.500 | tim
e: 13.53 minutes
Epoch: 087/99 | Loss: 0.0032 | Episodes: 88 | Win count: 34 | Win rate: 0.500 | tim
e: 13.84 minutes
Epoch: 088/99 | Loss: 0.0032 | Episodes: 89 | Win count: 34 | Win rate: 0.469 | tim
e: 14.18 minutes
Epoch: 089/99 | Loss: 0.0031 | Episodes: 90 | Win count: 34 | Win rate: 0.438 | tim
e: 14.55 minutes
Epoch: 090/99 | Loss: 0.0031 | Episodes: 91 | Win count: 34 | Win rate: 0.406 | tim
e: 14.90 minutes
Epoch: 091/99 | Loss: 0.0031 | Episodes: 92 | Win count: 34 | Win rate: 0.375 | tim
e: 15.24 minutes
Epoch: 092/99 | Loss: 0.0031 | Episodes: 93 | Win count: 34 | Win rate: 0.344 | tim
e: 15.60 minutes
Epoch: 093/99 | Loss: 0.0030 | Episodes: 94 | Win count: 34 | Win rate: 0.312 | tim
e: 15.96 minutes
Epoch: 094/99 | Loss: 0.0030 | Episodes: 95 | Win count: 35 | Win rate: 0.344 | tim
e: 16.20 minutes
Epoch: 095/99 | Loss: 0.0031 | Episodes: 96 | Win count: 35 | Win rate: 0.344 | tim
e: 16.56 minutes
Epoch: 096/99 | Loss: 0.0031 | Episodes: 97 | Win count: 36 | Win rate: 0.375 | tim
e: 16.71 minutes
Epoch: 097/99 | Loss: 0.0031 | Episodes: 98 | Win count: 37 | Win rate: 0.375 | tim
e: 16.76 minutes
Epoch: 098/99 | Loss: 0.0033 | Episodes: 99 | Win count: 37 | Win rate: 0.344 | tim
e: 17.12 minutes
Epoch: 099/99 | Loss: 0.0033 | Episodes: 100 | Win count: 37 | Win rate: 0.312 | tim
e: 17.48 minutes
Training complete in: 17.48 minutes
```

Note: This cell will check to see if the model passes the completion check. Note: This could take several minutes.

In [30]:
```
completion_check(model, qmaze)
show(qmaze)
```

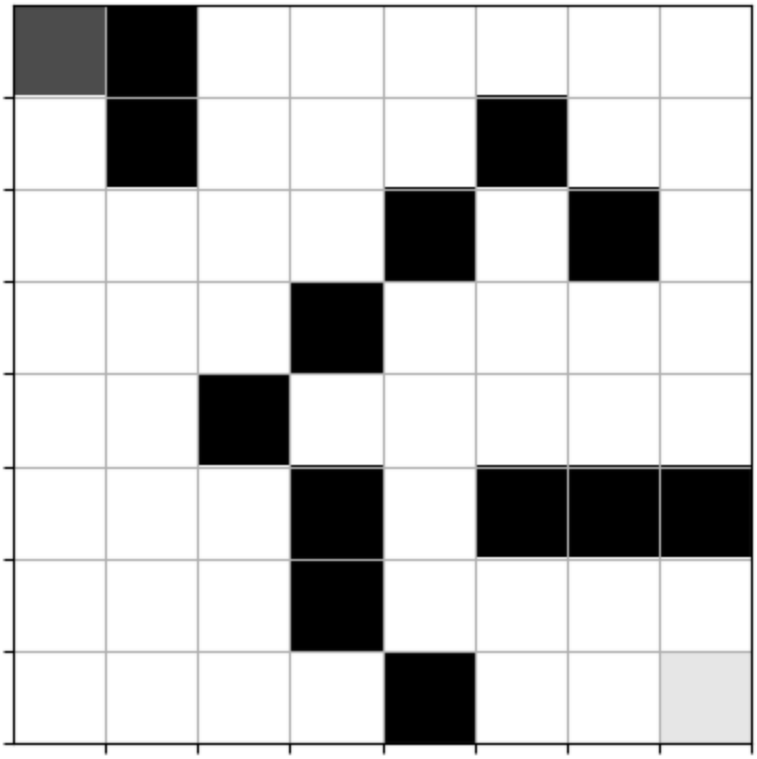
Out[30]:
```
<matplotlib.image.AxesImage at 0x7d320d1ad710>
```



This cell will test your model for one game. It will start the pirate at the top-left corner and run **play_game()**. The agent should find a path from the starting position to the target (treasure). The treasure is located in the bottom-right corner.

In [31]:
```
pirate_start = (0, 0)
play_game(model, qmaze, pirate_start)
show(qmaze)
```

Out[31]:
```
<matplotlib.image.AxesImage at 0x7d320b4c1e90>
```



Save and Submit Your Work

Hint: To use the markdown block below, double click in the **Type Markdown and LaTeX: α2** block below, to turn it back to html, Run the cell.

After you have finished creating the code for your notebook, save your work. Make sure that your notebook contains your name in the filename (e.g. Doe_Jane_ProjectTwo.html). Download this file as an .html file clicking on ***file*** in *Jupyter Notebook*, navigating down to ***Download as*** and clicking on ***.html***. Download a copy of your .html file and submit it to Brightspace.