Code and Plot for Q1C

```
import numpy as np
import pandas as pd

# Define the four-rooms environment structure (from hw0)
# 0 = empty cell, 1 = wall
four_room_space = np.array([
    [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    [1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1],
    [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
])

# Find all valid states (convert from array coordinates to state
coordinates)
empty_cells = np.where(four_room_space == 0.0)
state_space = [[col, 10 - row] for row, col in zip(empty_cells[0],
empty_cells[1])]

print(f"Total valid states: {len(state_space)}")
print(f"Example states: {state_space[:5]}")


empty_cells = np.where(four_room_space == 0.0)
state_space = [[col, 10 - row] for row, col in zip(empty_cells[0],
empty_cells[1])]
print(f"Total valid states: {len(state_space)}")
print(f"Example states: {state_space[:5]}")

Total valid states: 104
Example states: [[0, 10], [1, 10], [2, 10], [3, 10], [4, 10]]
Total valid states: 104
Example states: [[0, 10], [1, 10], [2, 10], [3, 10], [4, 10]]
```

define the actions fo the state and other helper functions

```
#define actiıns
actions = {
    'LEFT': [-1, 0],
    'RIGHT': [1, 0],
    'DOWN': [0, -1],
    'UP': [0, 1]
}

#define goal and start state
GOAL_STATE = [10, 10]
START_STATE = [0, 0]

def is_valid_state(state):
    #check if its a valid state so it doesnt go over
    return state in state_space

def get_next_state(state, action):
    """
    like step function we get the next state if its valid
    """
    #restart
    if state == GOAL_STATE:
        return START_STATE

    #calculate the next step
    next_state = [state[0] + action[0], state[1] + action[1]]

    #check if calculated state is valid
    if is_valid_state(next_state):
        return next_state
    else:
        return state  #stay in that state if its not

def get_reward(state, next_state):
    #get the  immediate reward
    if next_state == GOAL_STATE:
        return 1
    else:
        return 0

#couple print statements to test
print("Test: Moving right from [0, 0]:", get_next_state([0, 0],
actions['RIGHT']))
print("Test: Moving up from [0, 0]:", get_next_state([0, 0],
actions['UP']))
print("Test: From goal [10, 10]:", get_next_state([10, 10],
actions['UP']))
```

generate table

```
#create empty dynamics table  -> p(s',r/s,a)
dynamics_table = []

#loop over eeach state
for state in state_space:
    #from those each actiıns
    for action_name, action_vec in actions.items():
        #get the next state given state and action
        next_state = get_next_state(state, action_vec)

        # get the reward(r)
        reward = get_reward(state, next_state)

        #assuming deterministic not stochastic
        probability = 1.0

        #generate the table
        dynamics_table.append({
            's': tuple(state),          #current state
            'a': action_name,            #actiın
            's_prime': tuple(next_state), #next state
            'r': reward,                 #reward
            'p': probability             #probabbility
        })

print(f"Total non-zero entries in p(s', r|s, a):
{len(dynamics_table)}")
print(f"This equals: {len(state_space)} states × {len(actions)}
actions = {len(state_space) * len(actions)}")
```

display the table

```
#convert to pandas df
df = pd.DataFrame(dynamics_table)

print("\n" + "="*80)
print("SAMPLE ENTRIES FROM THE DYNAMICS TABLE")
print("="*80)
print("\nFirst 20 entries:")
print(df.head(20).to_string(index=False))

print("\n" + "="*80)
print("SPECIAL CASES")
print("="*80)

#
print("\nTransitions FROM goal state [10, 10] (all teleport to [0,
0]):")
goal_df = df[df['s'] == (10, 10)]
print(goal_df.to_string(index=False))

#transtion to goal state
print("\nTransitions TO goal state [10, 10]:")
to_goal_df = df[df['s_prime'] == (10, 10)]
print(f"Number of (s, a) pairs that reach goal: {len(to_goal_df)}")
print(to_goal_df.to_string(index=False))
```

```python
#rewards
print("\n" + "="*80)
print("SUMMARY STATISTICS")
print("="*80)
print(f"\nReward distribution:")
print(df['r'].value_counts().sort_index())
print(f"\nReward = 1: {len(df[df['r'] == 1])} transitions")
print(f"Reward = 0: {len(df[df['r'] == 0])} transitions")
```