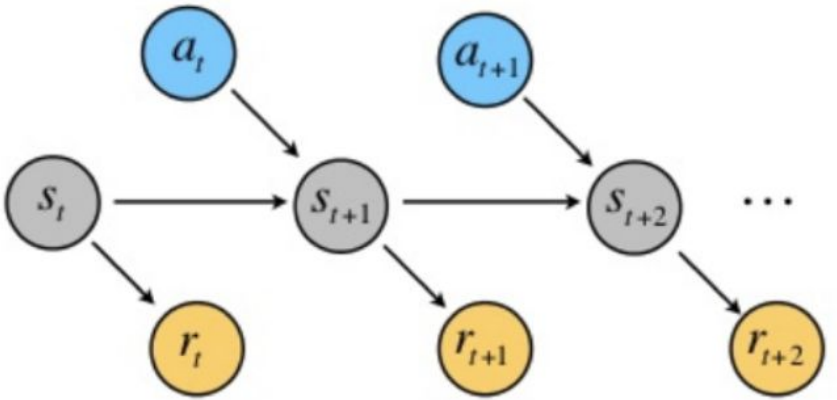
General Principles of Human and Machine Learning

Tutorial 3: Introduction to RL

RL Framework

- In RL, problems can be described using the MDP formality
  - MDP is a 4-tuple (S, A, P, R)
    - S: state space
    - A: action space
    - P: state transition probability
      - $P(S_{t+1}=s_{t+1}|S_t=s_t,A_t=a_t)$
    - R: state transition returns
      - $R(s_t,s_{t+1}) = r_t$
  - Markov Decision Process because of the Markov property
    - $P(S_{t+1}|S_t,...,S_1,A_t) = P(S_{t+1}|S_t,A_t)$



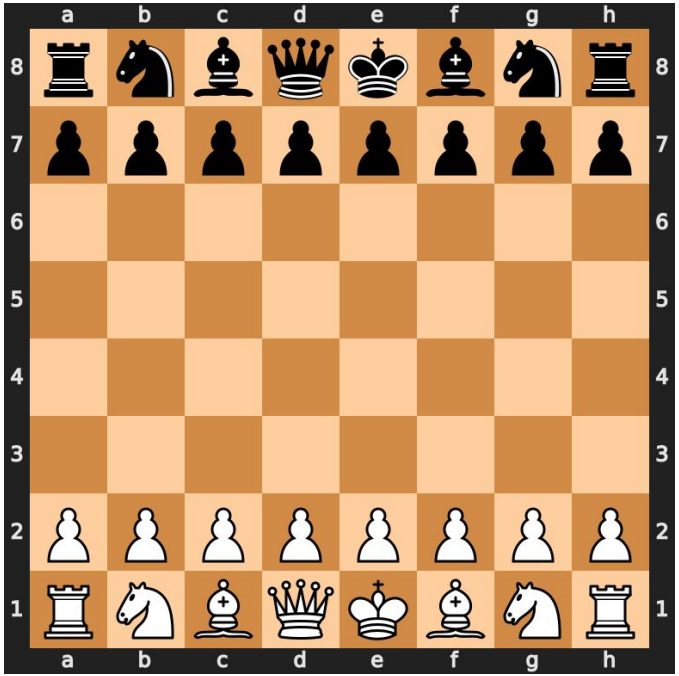
Andrey Markov

Tutorial Questions

Devise three example tasks of your own that fit into the reinforcement learning framework, identifying for each its states, actions, and rewards. Make the three examples as different from each other as possible. The framework is abstract and flexible and can be applied in many different ways. Stretch its limits in some way in at least one of your examples.

Example 1: Chess

- S: board position of the pieces
- S_0 = {Black rook: a8, Black knight: b8, …., White rook: h1}



Example 1: Chess

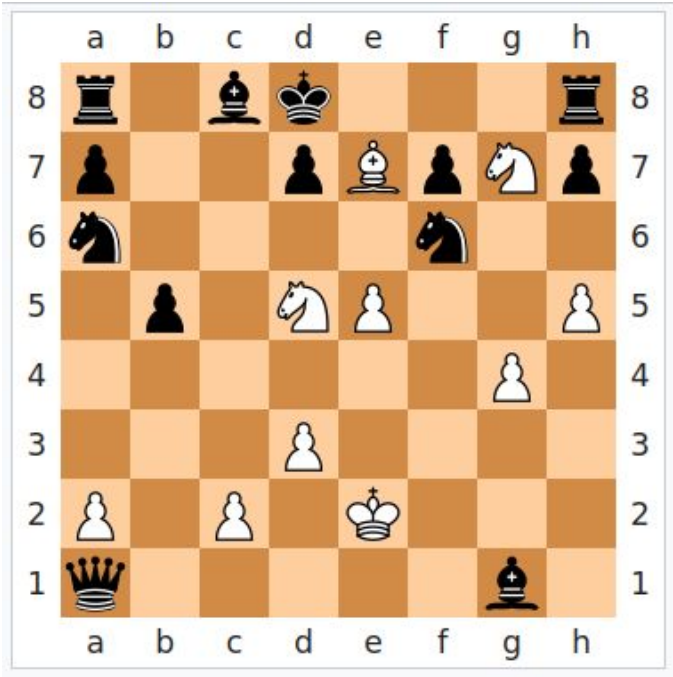
- A: all the legal moves (whites) given the current board configuration
  - For example, on the board to the right:
    - White pawn: e2 →e4

Example 1: Chess

- P: the probability of each board configuration given player's move and the current configuration.
  - For example, on the board to the right:
    - Given White pawn: e2 →e4
      - Black pawn: e7 →e5
      - But could have also played:
        - Black pawn: d7 →d5
        - Black pawn: e7 →e6
        - Black knight: b8 →c6
        - Etc
- P(S_1 = {Black rook: a8, Black knight: b8, …., Black pawn: e5,White pawn: e6, …, White rook: h1} | S_0 = {Black rook: a8, Black knight: b8, …., White rook: h1}, A_0 = White pawn: e2 →e4) ∈ [0,1]
- Note that the Markov property is preserved:
  - Previous board configurations don't affect the transitions:
    - All the necessary information is incorporated into the current board configuration

Example 1: Chess

- R:
  - Endgame:
    - Win: +1

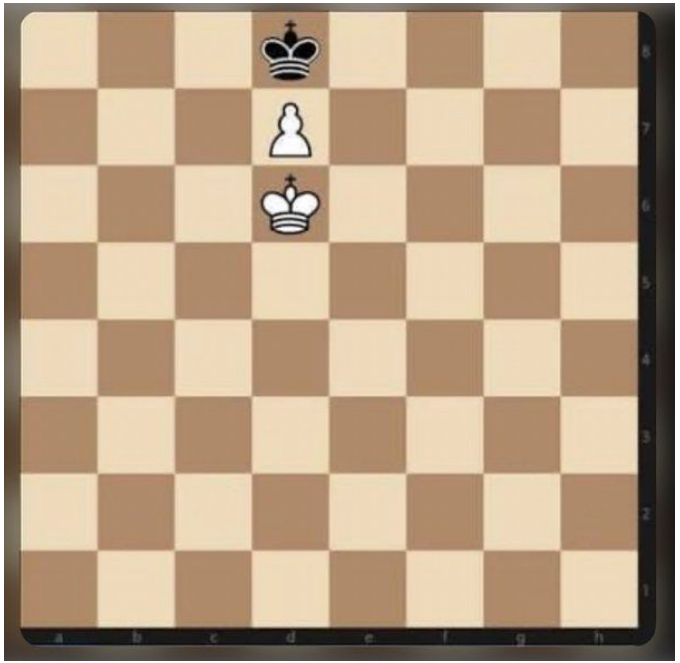


Example 1: Chess

- R:
  - Endgame:
    - Win: +1
    - Loss: -1

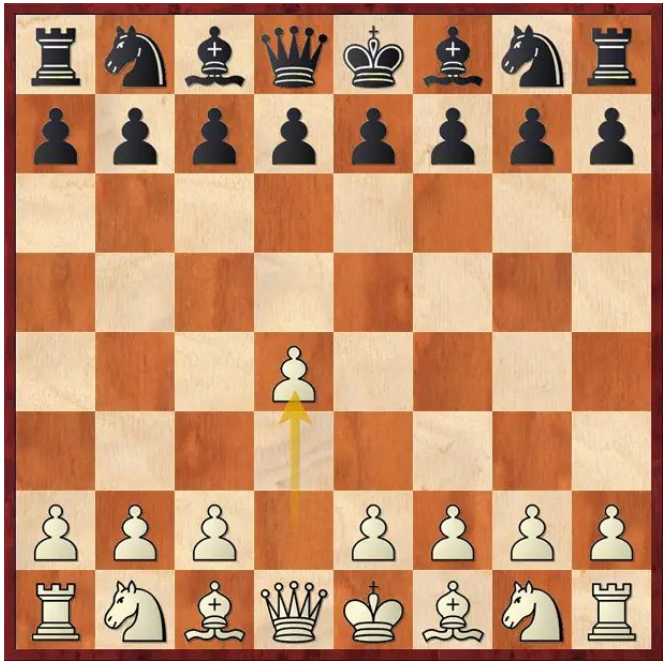
Example 1: Chess

- R:
  - Endgame:
    - Win: +1
    - Loss: -1
    - Draw: 0



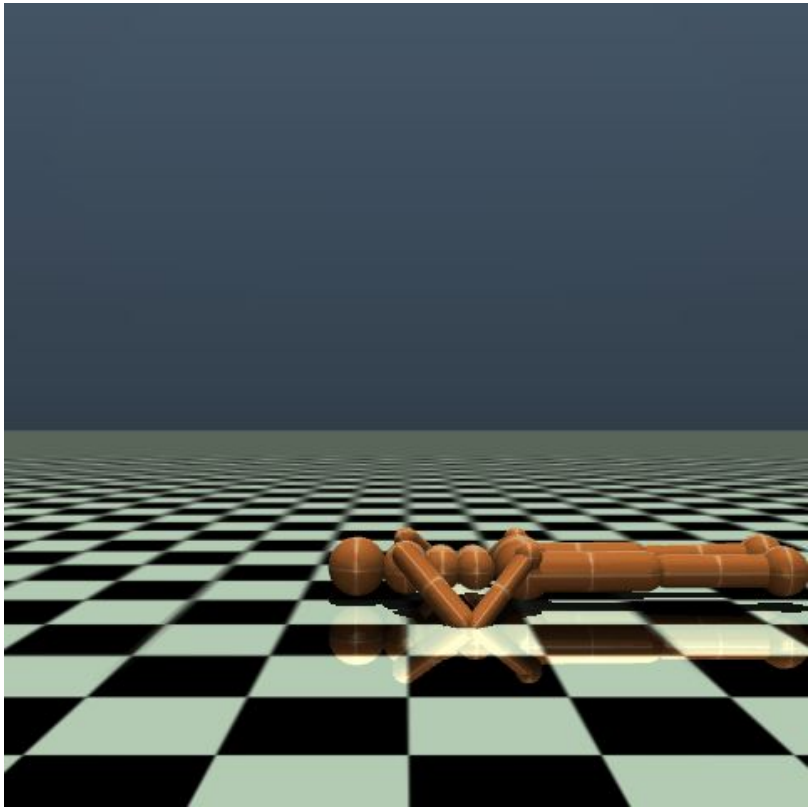
Example 1: Chess

- R:
  - Endgame:
    - Win: +1
    - Loss: -1
    - Draw: 0
  - Otherwise:
    - If a move doesn't lead to an endgame situation: 0
      - This is a major problem in RL called sparsity of reward
      - It leads to the credit assignment problem



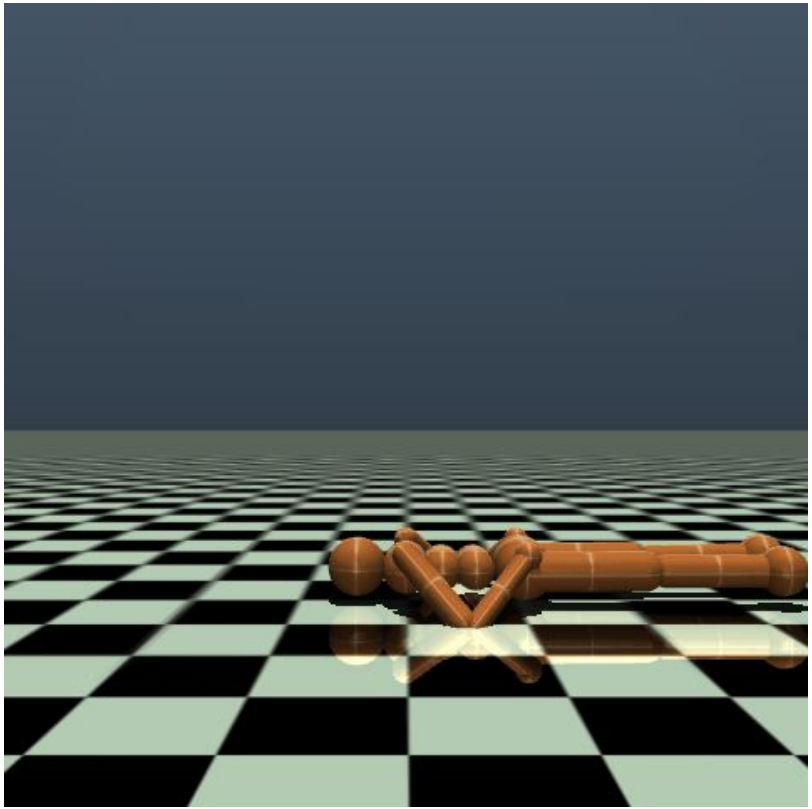
Example 2: OpenAI Gym (Humanoid Standup)

- S: The state space consists of positional values of different body parts of the Humanoid, followed by the velocities of those individual parts (their derivatives) with all the positions ordered before all the velocities.
  - 1: z-coordinate of the torso (centre)
  - 2: x-orientation of the torso (centre)
  - …
  - 45: angular velocity of the angle between left upper arm and left_lower_arm



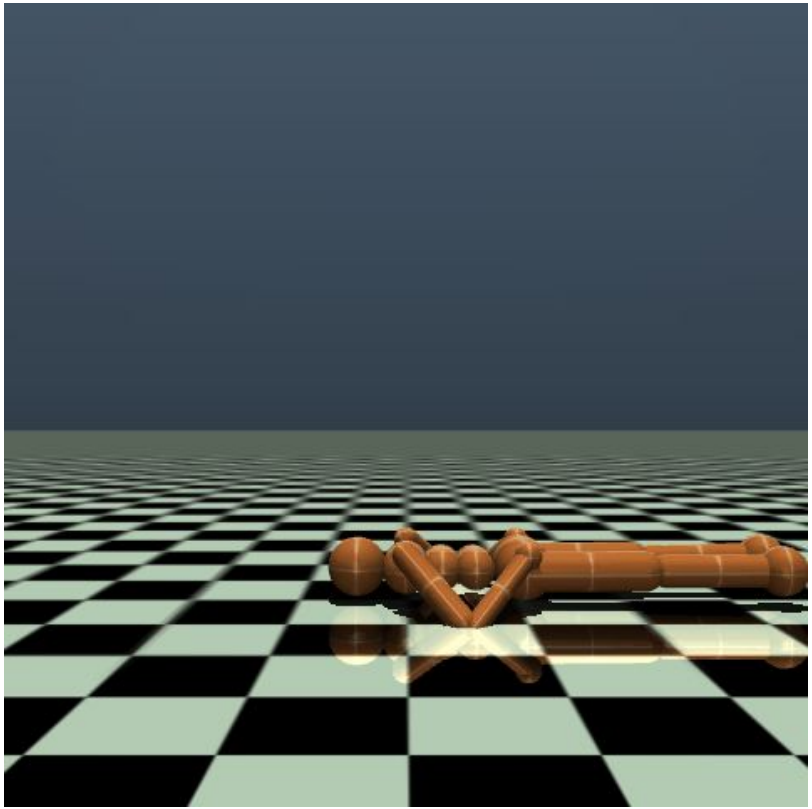
Example 2: OpenAI Gym (Humanoid Standup)

- A: The agent take a 17-element vector for actions representing the numerical torques applied at the hinge joints.
  - 0: Torque applied on the hinge in the y-coordinate of the abdomen
  - 1: Torque applied on the hinge in the z-coordinate of the abdomen
  - …
  - 16: Torque applied on the rotor between the left upper arm and left lower arm



Example 2: OpenAI Gym (Humanoid Standup)

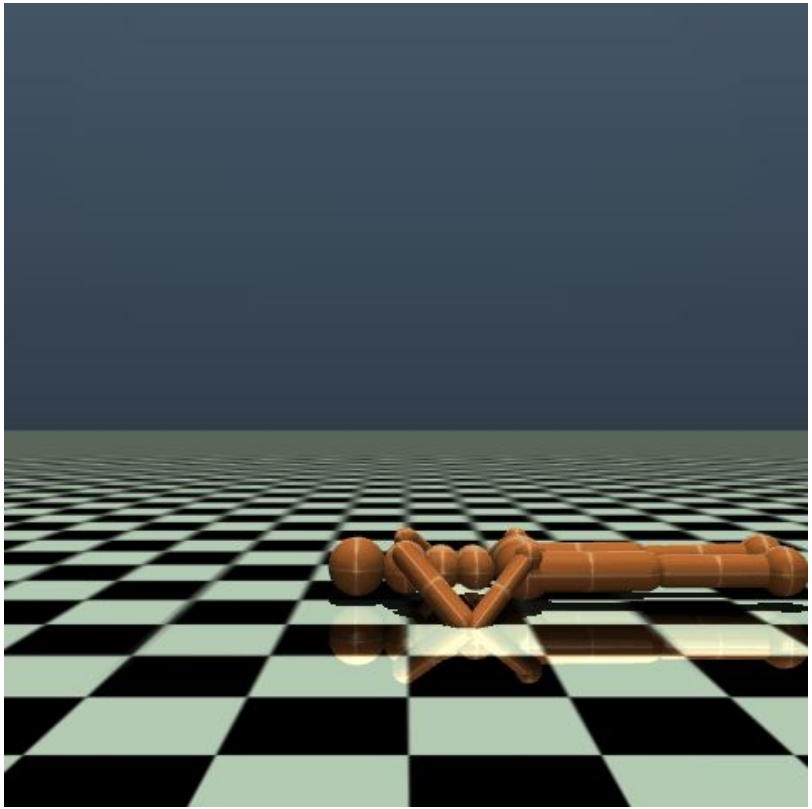
- P: the probability of the next positional values of different body parts given their velocities and applied torques at the joints
  - The environment is inherently noisy which makes the transitions non-deterministic



Example 2: OpenAI Gym (Humanoid Standup)

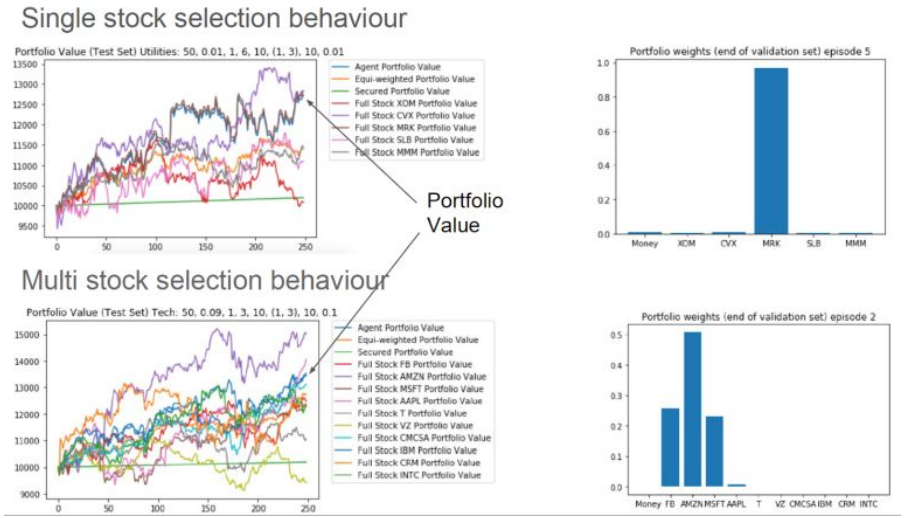
- R: a reward for moving upward
  - Technically there are other rewards as well, but it's irrelevant for our purposes

https://www.youtube.com/shorts/K-pzg5nw7us



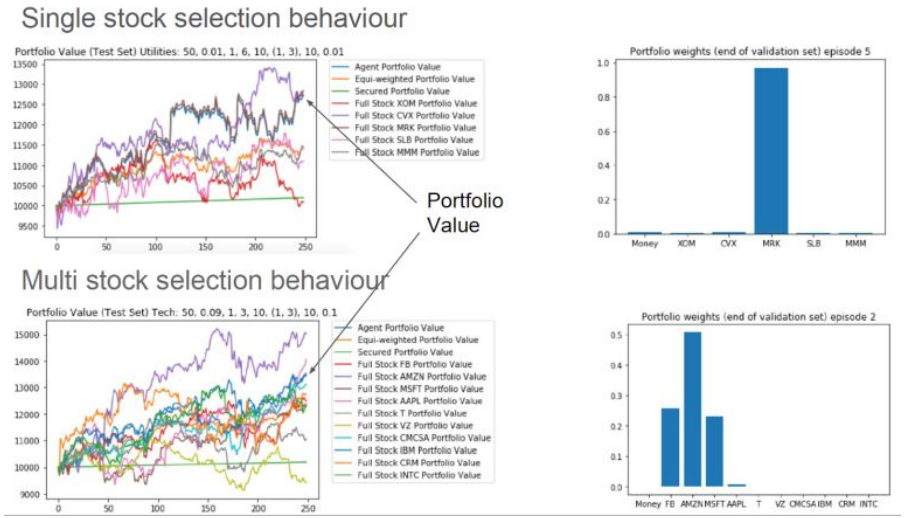
Example 3: Portfolio management (Finance)

- S: Number of each stock in the portfolio and their current value
  - Full Stock FB (Number: 100, Price: 235.79)
  - Full Stock AMZN (Number: 50, 112.18)
  - …



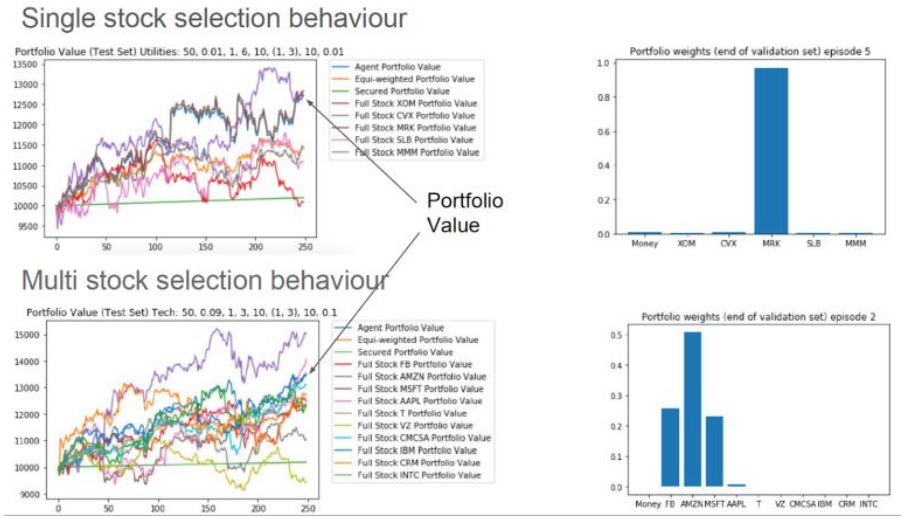
Example 3: Portfolio management (Finance)

- A: Buy #stocks available in the market, Sell #stocks in your portfolio:
  - Sell Full Stock FB (Number: 10)
  - Buy Full Stock BABA (Number: 20)
    112.18)
  - etc



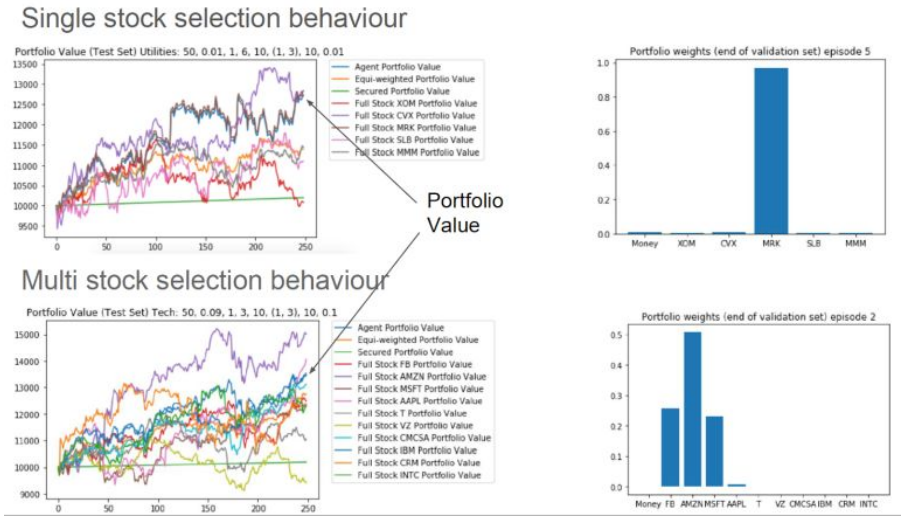
Example 3: Portfolio management (Finance)

- P: the probability of the next time step values of the stocks in the market and the number of stocks the agent has in the portfolio
  - The values are inherently stochastic because they depend on market forces that aren't fully predictable by the agent



Example 3: Portfolio management (Finance)

- R: the change in portfolio valuation



Tutorial Questions

Is the reinforcement learning framework adequate to usefully represent all goal-directed learning tasks? Can you think of any clear exceptions?

REMINDER:

- Goal directed: one has an objective in mind
    - Involves planning
- Habitual:
    - Actions that are automatic

Partially observable states

- The MDP framework fails when the state cannot be fully observed.
  - Imagine trying to control the temperature of the house
    - States: temperature of the house
    - Actions: controlling the knob of the heater
    - State transitions: given the current temperature of the house and the angle of the knob turns, the house temperature will transition to a new state
      - It's Markovian as the the house heating state from >1 prior time steps doesn't affect the current one
      - It's probabilistic due to physics of heat
    - Reward: The change in distance between the target and current temperature
    - There's a thermometer in a room that lets you observe that room's temperature
      - However that thermometer reading is not an accurate proxy for the whole house
        - The agent's state space will correspond to the temperature readings of the room as opposed to the actual temperature of the house
  - POMDPs have been introduced to deal with this kind of problems

Non-Markovian Stochastic process

- Not all stochastic processes have the Markov property:
  - $P(S_{t+1}|S_t,...,S_1,A_t) = P(S_{t+1}|S_t,A_t)$
  - In these stochastic processes, states do not contain full information needed for transitions
- An urn contains two red balls and one green ball. One ball was drawn yesterday, one ball was drawn today, and the final ball will be drawn tomorrow. All of the draws are "without replacement".
  - Suppose you know that today's ball was red, but you have no information about yesterday's ball. The chance that tomorrow's ball will be red is 1/2. That's because the only two remaining outcomes for this random experiment are "r,r,g" and "g,r,r".
  - On the other hand, if you know that both today and yesterday's balls were red, then you are guaranteed to get a green ball tomorrow.
  - This discrepancy shows that the probability distribution for tomorrow's color depends not only on the present value, but is also affected by information about the past.
  - This stochastic process of observed colors doesn't have the Markov property.

https://math.stackexchange.com/questions/89394/example-of-a-stochastic-process-which-does-not-have-the-markov-property

Tutorial Questions

Give an example of situations in which habitual behaviour is enough and will work reliably, and give an example of a task in which you really need a goal directed approach.

Habitual behavior

- the environment or outcome is stable
  - Tooth brushing
  - making tea
  - Riding a bicycle
  - etc

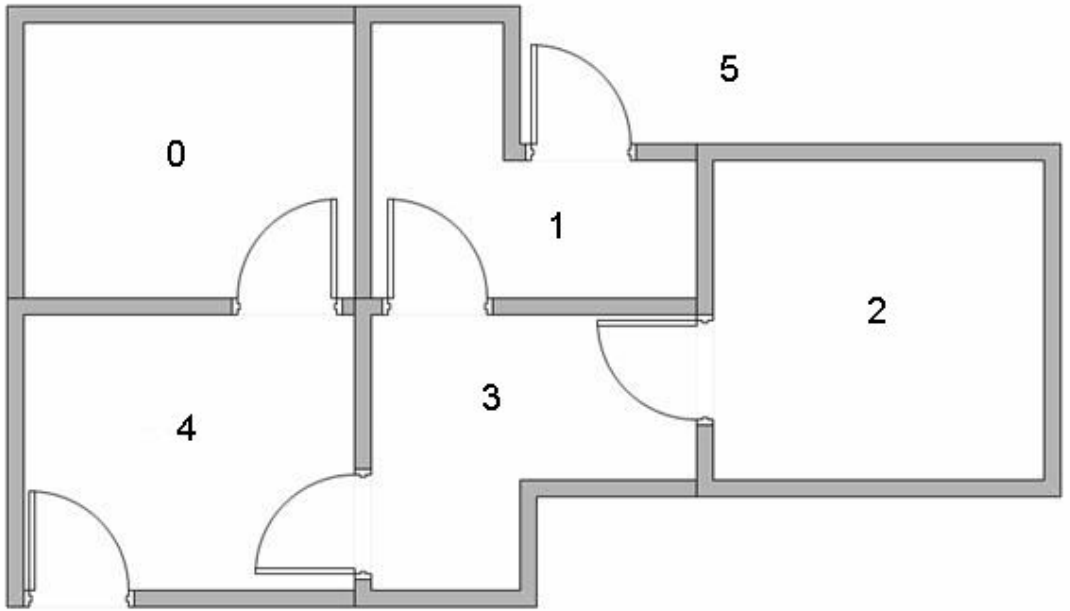
Goal-directed behavior

- the environment necessitates to go somewhere new, or to reach a location when the transitions are constantly changing; planning is necessary
  - Arranging travel (visa, tickets, hotels, etc)
  - Chess playing
  - Planning your education/career path
  - etc

Tutorial Questions

Let's consider a situation in which a robot is placed inside a building that has a floorplan like that shown in the following image.

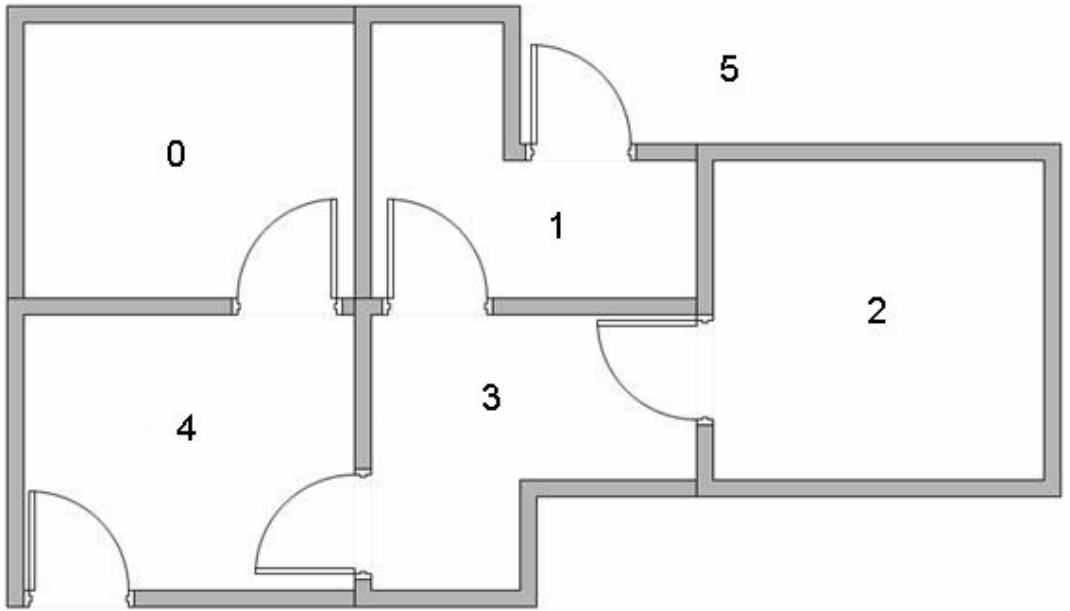
We can characterize this space as an MDP, where each state represents one room in the building (or outside, e.g., room 5) and where the agent can transition between rooms by moving either north, south, east, or west. The agent cannot stay in the same state from time step to time step, except once it is outside.



Tutorial Questions

Let's consider a situation in which a robot is placed inside a building that has a floorplan like that shown in the following image.

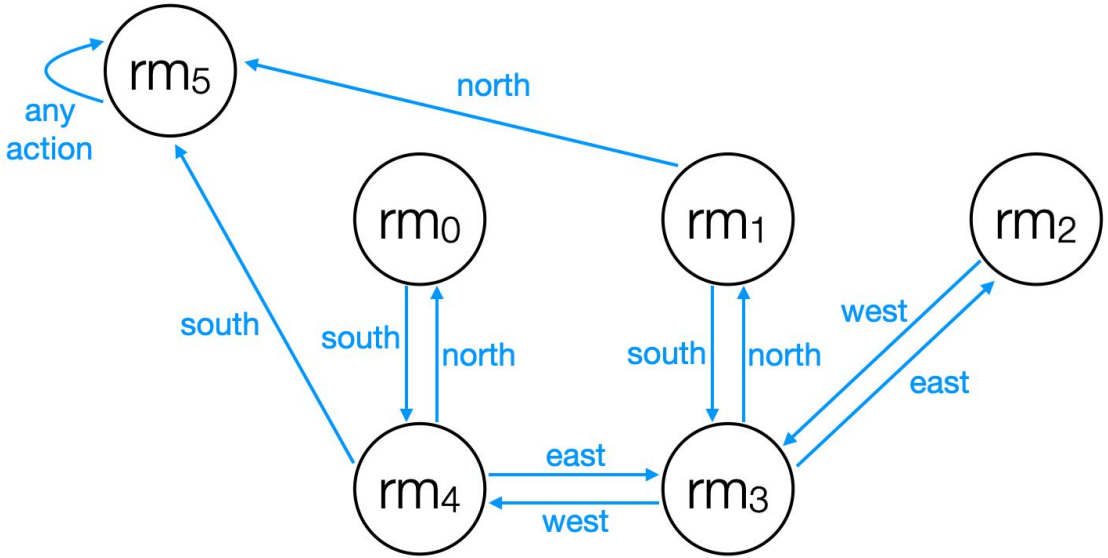
Draw a graph with nodes corresponding to states and edges to - state transitions.



Tutorial Questions

Let's consider a situation in which a robot is placed inside a building that has a floorplan like that shown in the following image.

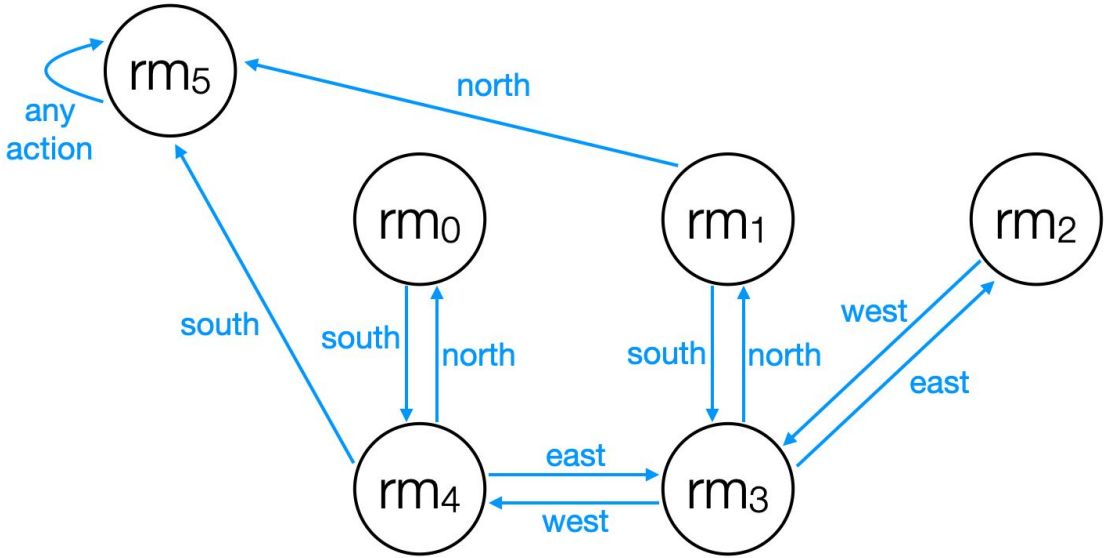
Draw a graph with nodes corresponding to states and edges to - state transitions.



Tutorial Questions

The agent receives a reward of 100 when it transitions outside from either room 1 or room 4. Additionally, the agent continues reaping rewards once it's already outside every time step.

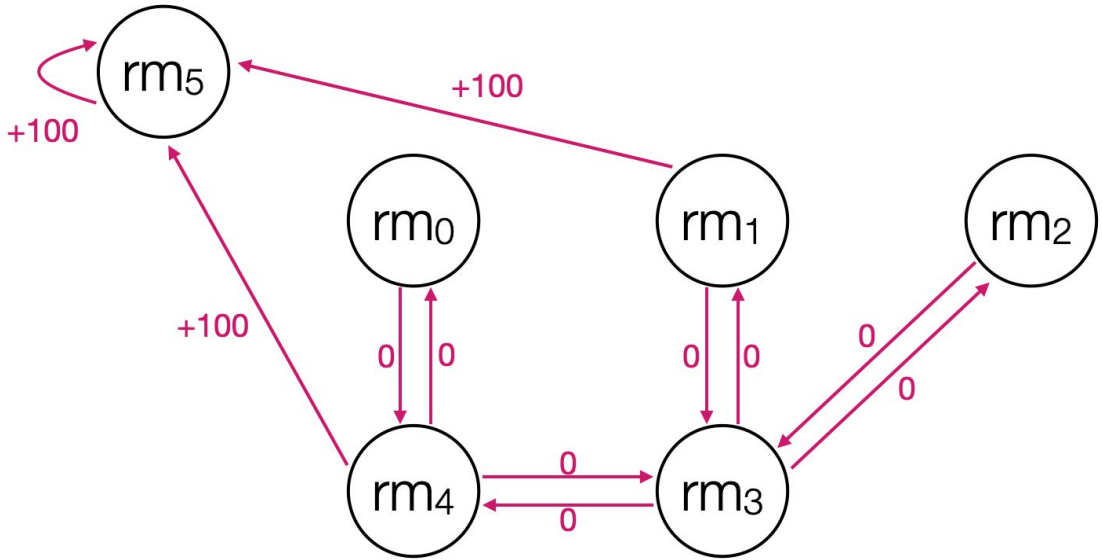
Add the rewards to the transitions



Tutorial Questions

The agent receives a reward of 100 when it transitions outside from either room 1 or room 4. Additionally, the agent continues reaping rewards once it's already outside every time step.

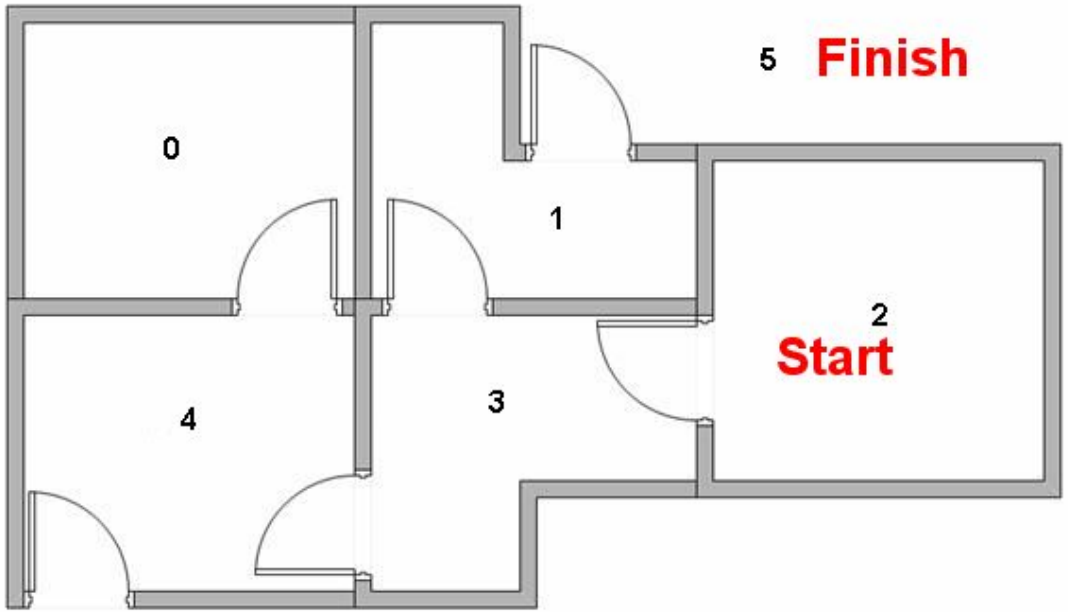
Add the rewards to the transitions



Tutorial Questions

Our robot's goal in this world is to get outside (room 5) from its starting position in room 2.

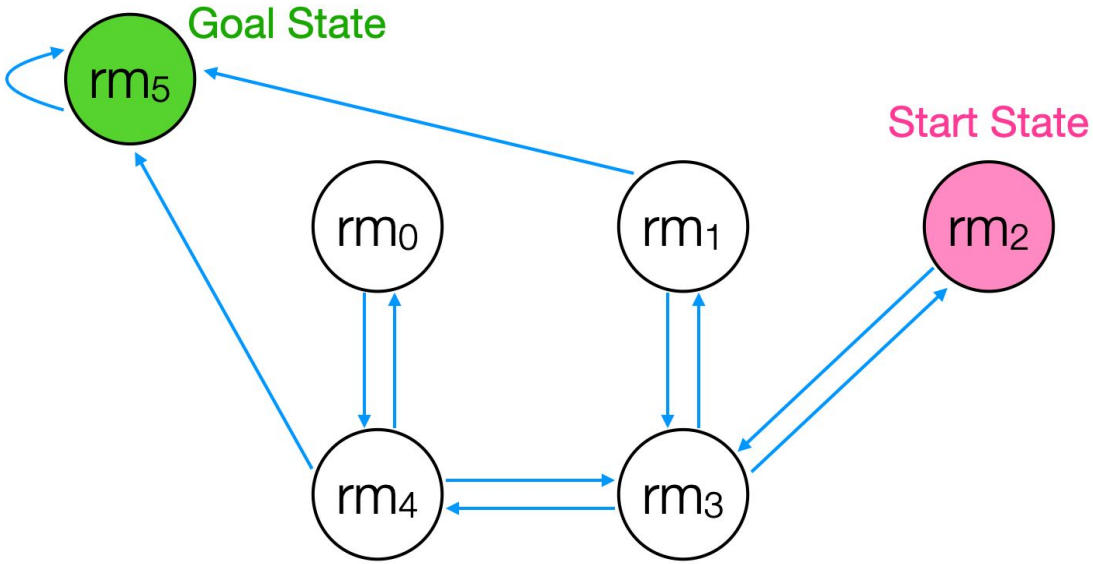
Make the necessary changes to the previously drawn graph



Tutorial Questions

Our robot's goal in this world is to get outside (room 5) from its starting position in room 2.

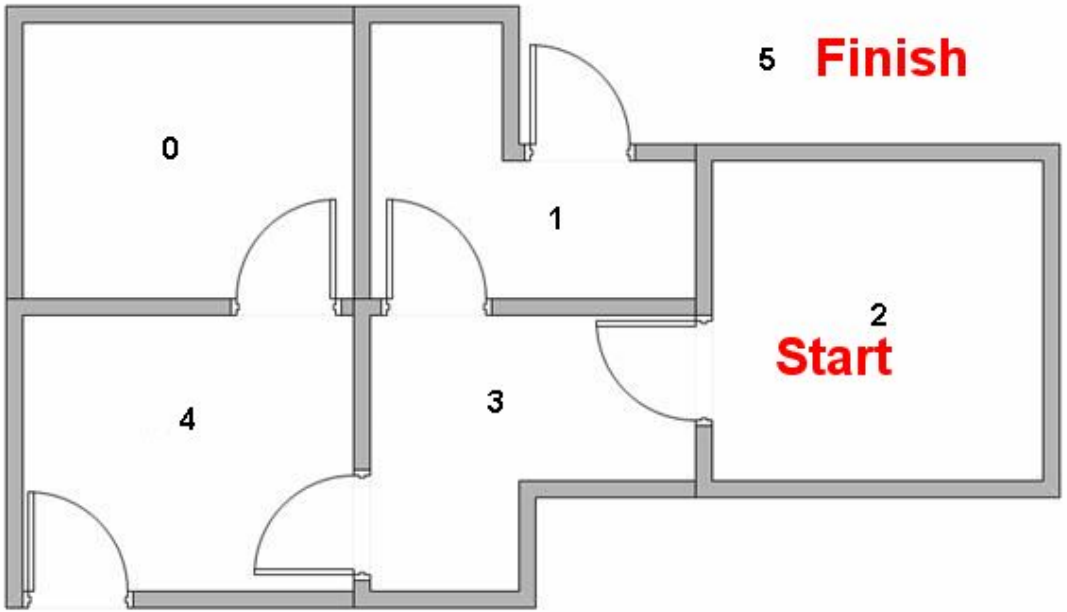
Make the necessary changes to the previously drawn graph



Tutorial Questions

Build a table / matrix to represent $Q(s_t,a_t)$

- rows represent the world states (rooms)
- columns represent actions that the robot can take.
- all valid state-action pairs are initialized to 0
- invalid state-action pairs are initialized to -1



Tutorial Questions

Your goal in this exercise is to update $Q(s_t,a_t)$ according to the Q-learning algorithm. In this exercise we will make the following assumptions:

- $\alpha = 1$
- $\gamma = 0.8$
- Transitions are always successful
- $Q(s_t,a_t) \leftarrow \alpha\,(r_t + \gamma \max_a Q(s_{t+1},a)) = r_t + 0.8 \max_a Q(s_{t+1},a)$

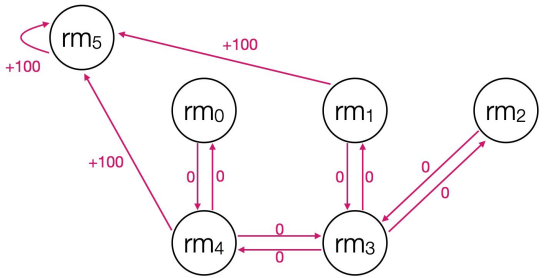
	North	South	East	West
rm0	-1	0	-1	-1
rm1	0	0	-1	-1
rm2	-1	-1	-1	0
rm3	0	-1	0	0
rm4	0	0	0	-1
rm5	0	0	0	0

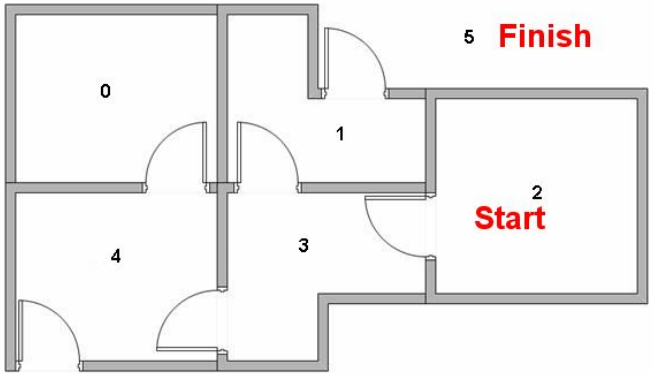
Tutorial Questions

For each action within each trajectory, update according to the Q-learning algorithm after each trajectory (assume that your agent always starts in room 2 at the beginning of each trajectory).

- Trajectory 1: west, west, south, north

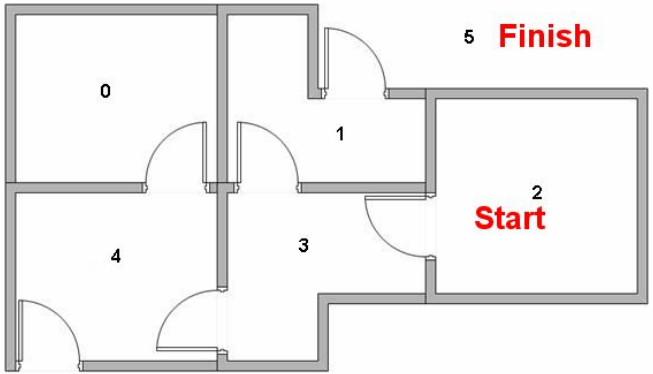
$Q(s_t,a_t) \leftarrow \alpha\,(r_t + \gamma \max_a Q(s_{t+1},a)) = r_t + 0.8 \max_a Q(s_{t+1},a)$





	North	South	East	West
rm0	-1	0	-1	-1
rm1	0	0	-1	-1
rm2	-1	-1	-1	0
rm3	0	-1	0	0
rm4	0	0	0	-1
rm5	0	0	0	0

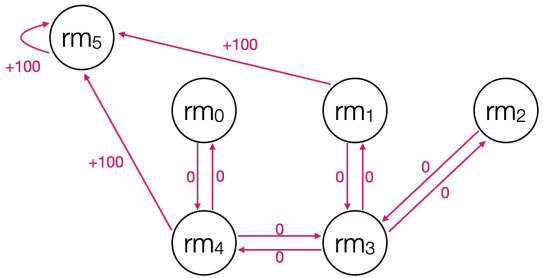
Tutorial Questions



For each action within each trajectory, update according to the Q-learning algorithm after each trajectory (assume that your agent always starts in room 2 at the beginning of each trajectory).

- Trajectory 1: west, west, south, north

$Q(s_t,a_t) \leftarrow \alpha\,(r_t + \gamma \max_a Q(s_{t+1},a)) = r_t + 0.8 \max_a Q(s_{t+1},a)$



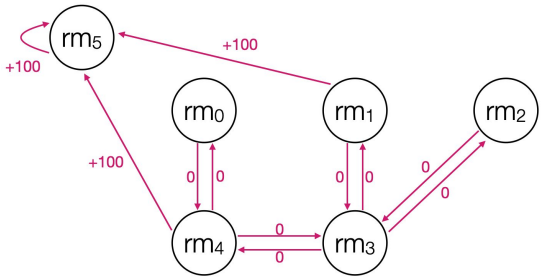
	North	South	East	West
rm0	-1	0	-1	-1
rm1	0	0	-1	-1
rm2	-1	-1	-1	**0**
rm3	0	-1	0	**0**
rm4	0	**100**	0	-1
rm5	**100**	0	0	0

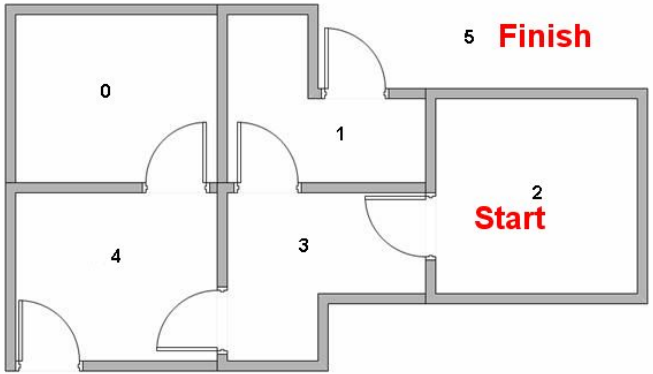
Tutorial Questions

For each action within each trajectory, update according to the Q-learning algorithm after each trajectory (assume that your agent always starts in room 2 at the beginning of each trajectory).

- Trajectory 2: west, north, north, south

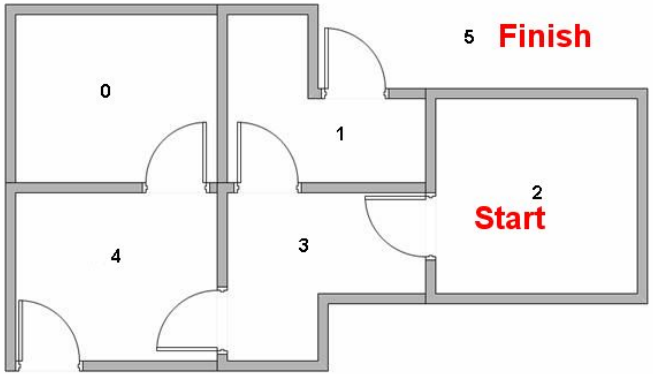
$Q(s_t,a_t) \leftarrow \alpha\,(r_t + \gamma \max_a Q(s_{t+1},a)) = r_t + 0.8 \max_a Q(s_{t+1},a)$





	North	South	East	West
rm0	-1	0	-1	-1
rm1	0	0	-1	-1
rm2	-1	-1	-1	0
rm3	0	-1	0	0
rm4	0	100	0	-1
rm5	100	0	0	0

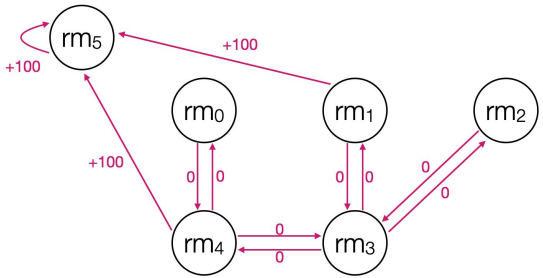
Tutorial Questions



For each action within each trajectory, update according to the Q-learning algorithm after each trajectory (assume that your agent always starts in room 2 at the beginning of each trajectory).

- Trajectory 2: west, north, north, south

$Q(s_t,a_t) \leftarrow \alpha\,(r_t + \gamma \max_a Q(s_{t+1},a)) = r_t + 0.8 \max_a Q(s_{t+1},a)$



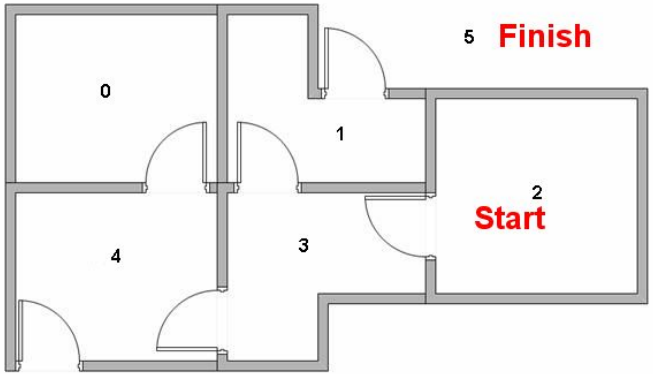
	North	South	East	West
rm0	-1	0	-1	-1
rm1	**180**	0	-1	-1
rm2	-1	-1	-1	**0**
rm3	**0**	-1	0	0
rm4	0	100	0	-1
rm5	100	**180**	0	0

Tutorial Questions

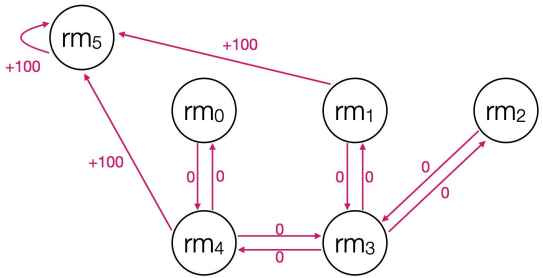
For each action within each trajectory, update according to the Q-learning algorithm after each trajectory (assume that your agent always starts in room 2 at the beginning of each trajectory).

- Trajectory 3: west, west, north, south, south

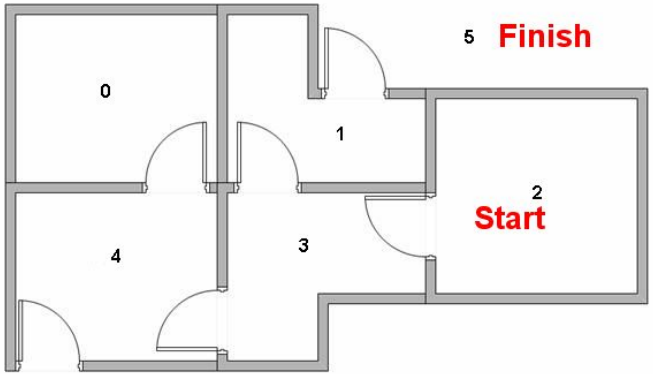
$Q(s_t,a_t) \leftarrow \alpha\,(r_t + \gamma \max_a Q(s_{t+1},a)) = r_t + 0.8 \max_a Q(s_{t+1},a)$



	North	South	East	West
rm0	-1	0	-1	-1
rm1	180	0	-1	-1
rm2	-1	-1	-1	0
rm3	0	-1	0	0
rm4	0	100	0	-1
rm5	100	180	0	0



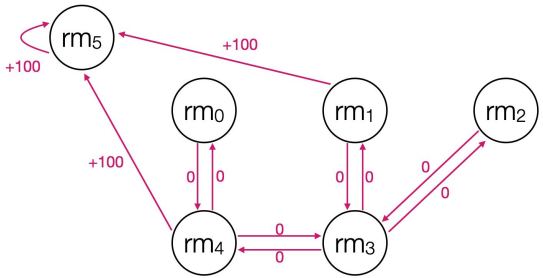
Tutorial Questions



For each action within each trajectory, update according to the Q-learning algorithm after each trajectory (assume that your agent always starts in room 2 at the beginning of each trajectory).

- Trajectory 3: west, west, north, south, south

$Q(s_t,a_t) \leftarrow \alpha\,(r_t + \gamma \max_a Q(s_{t+1},a)) = r_t + 0.8 \max_a Q(s_{t+1},a)$



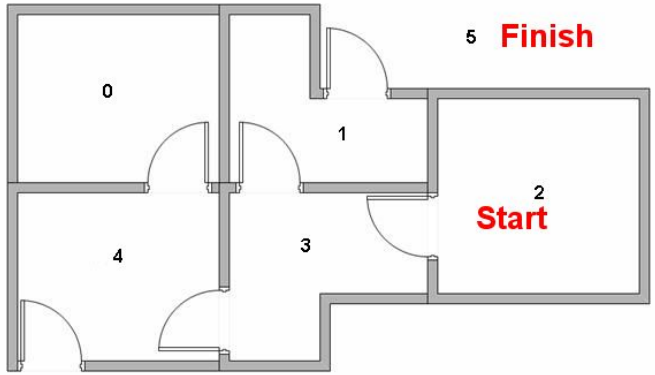
	North	South	East	West
rm0	-1	**80**	-1	-1
rm1	180	0	-1	-1
rm2	-1	-1	-1	**0**
rm3	0	-1	0	**80**
rm4	**0**	**244**	0	-1
rm5	100	180	0	0

Tutorial Questions

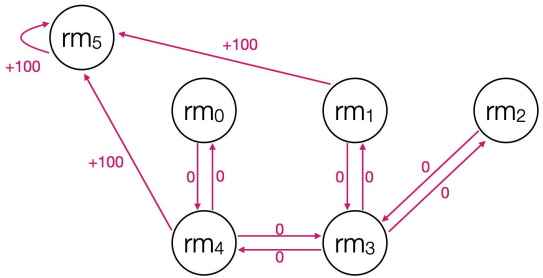
For each action within each trajectory, update according to the Q-learning algorithm after each trajectory (assume that your agent always starts in room 2 at the beginning of each trajectory).

- Trajectory 4: west, west, east, north, north

$Q(s_t,a_t) \leftarrow \alpha\,(r_t + \gamma \max_a Q(s_{t+1},a)) = r_t + 0.8 \max_a Q(s_{t+1},a)$



	North	South	East	West
rm0	-1	80	-1	-1
rm1	180	0	-1	-1
rm2	-1	-1	-1	0
rm3	0	-1	0	80
rm4	0	244	0	-1
rm5	100	180	0	0

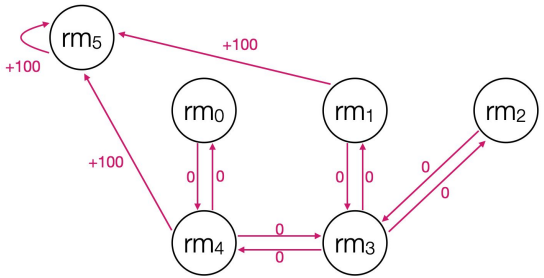


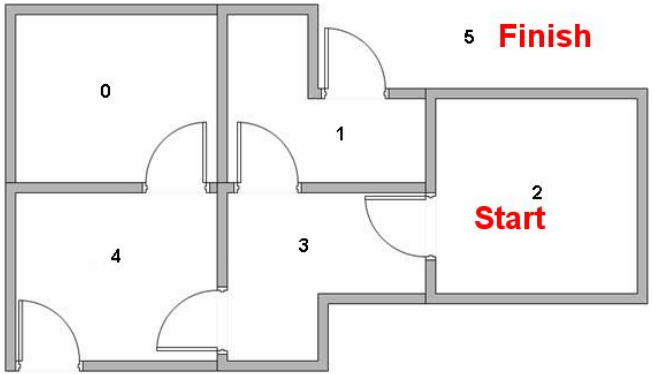
Tutorial Questions

For each action within each trajectory, update according to the Q-learning algorithm after each trajectory (assume that your agent always starts in room 2 at the beginning of each trajectory).

- Trajectory 4: west, west, east, north, north

$Q(s_t,a_t) \leftarrow \alpha\,(r_t + \gamma \max_a Q(s_{t+1},a)) = r_t + 0.8 \max_a Q(s_{t+1},a)$





	North	South	East	West
rm0	-1	80	-1	-1
rm1	**244**	0	-1	-1
rm2	-1	-1	-1	**64**
rm3	**144**	-1	0	**195.2**
rm4	0	244	**156.16**	-1
rm5	100	180	0	0