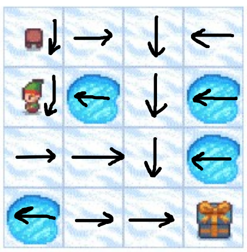
Exercise: Monte Carlo Methods and Temporal-Difference Learning

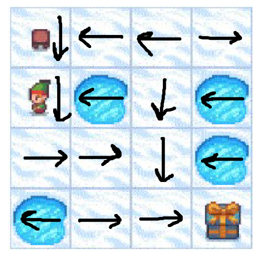
Task 1)
A) What is the difference between Monte Carlo (MC) methods and Dynamic Programming?
B) What is bootstrapping? Is MC using bootstrapping?
C) What is the difference between on-policy and off-policy?
D) What is the following equation describing?

$$\rho_{t:T-1} \doteq \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$$

Task 2) Frozen Lake



MC -> Policy in Bild eingezeichnet nach 2000 Episoden und Epsilon-Greedy Sequenz Generation



Q-Learning -> Policy in Bild eingezeichnet nach 20000 Episoden und Epsilon-Greedy Sequenz Generation

Solve the Open AI task frozen lake (desc=None,map_name="4x4", is_slippery=False) using MC and Q-Learning. You might either start your own implementation **or** complete the Python script provided in the lecture.

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:
  $\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
  $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
  $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):
  Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0
  Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
      Append G to $Returns(S_t, A_t)$
      $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
      $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s,a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$
  until S is terminal

Using Open AI Gym's Frozen Lake Environment
https://www.gymlibrary.ml/environments/toy_text/frozen_lake/
> pip install gym   # to install gym lib
> pip install pygame # to install pygame lib (needed for visualization)

Suggested further reading:
https://blog.paperspace.com/getting-started-with-openai-gym/
https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/

Bonus: solve the task using is_slippery=True

Task 3) Cliff Walk

Consider the gridworld shown below. This is a standard undiscounted, episodic task, with start and goal states, and the usual actions causing movement up, down, right, and left. Reward is −1 on all transitions except those into the region marked "The Cliff" Stepping into this region incurs a reward of −100 and sends the agent instantly back to the start. Movements outside the gridworld result in a reward of -1 and the state remains the same.

Use Sarsa or Q-learning methods with ε-greedy action selection (ε= 0.1) for 500 episodes. Draw the final walk of the agent in episode 500. Does it resemble the path shown below?