This is a pdf version of https://git.tu-berlin.de/lis-public/ai-student-workspace/-/blob/main/04/README.md

To obtain the code for this assignment, you will need to fetch and pull new commits from git@git.tu-berlin.de:lis-public/ai-student-workspace.git

As always, only modify the file `solution_??.py`. And even in `solution_??.py`, only modify what the functions do - don't change the function's names. Don't add any additional files, put all your code in `solution_??.py`.

You can run tests by navigating to the task folder `??`, and then simply typing `python3 -m pytest`. If you haven't yet, you will need to install pytest first: `sudo apt install python3-pytest`.

Assignment 4: Sequential Decisions – Bellman based

4.1: Vanilla Q-Learning

For this part of the exercise, you need to modify the function `q_learning(env)` in `solution_04.py`. This function's only input, `env`, is an object that simulates the domain, and has the following methods:

- `S = getNumStates()`, which returns the number of states $s \in 0, 1, \ldots, S-1$.
- `A = getNumActions()`, which returns the number of discrete possible actions $a \in 0, 1, \ldots, A-1$.
- `y = reset()`, which resets the simulator to a start state s_0, and returns $y_0 = s_0$ as observation.
- `(r, y, done) = step(a)`, which executes action a_t, leading to a new internal state s_{t+1}, and returns a real-valued reward r_t, an observation $y_{t+1} = s_{t+1}$, and `done`, a Boolean that indicates whether the new state is terminal.

As indicated above, states and actions are integer numbers between 0 and $S-1$ or $A-1$, respectively. The discount factor is $\gamma = 0.9$.

Implement standard Q-learning (for now without n-step updates, eligibilities, or replay) as described in the pseudo code (slide 12) of the lecture. If the argmax value (as in line 5 of the pseudo code) is not uniquely defined, because multiple actions have the same q-value, choose one of them randomly. Your function, `q_learning`, can call `env.step` up to 10,000 times (after that, an exception is raised). We suggest you start testing with $\alpha = 0.1$ and $\epsilon = 0.1$ However, potentially you will have to tune these parameters yourself to ensure efficient convergence.

Return a numpy array of size $S \times A$, containing all values of $Q^*(s, a)$. The tests will check whether the returned Q-values at selected states (esp. the start state) are indeed close to the optimal values.

The test domain used is FrozenLake: https://www.gymlibrary.ml/environments/toy_text/frozen_lake/. For debugging, you can call `env.render()` to get a refreshed rendering of the environment. However, make sure that your submitted `q_learning` function in the end does not contain any `env.render` calls, as this increases runtime dramatically. Your algorithm should complete the public tests in ca. 10 seconds.

4.2: n-step Q-learning

This exercise is not graded. Please use a separate file; do *not* write your code for this exercise in `solution_04.py`.

On slide 18, n-step updates are explained. To this end,implement a "data buffer" of fixed size , $D = [(s_\theta, r_\theta, a_\theta, s_{\theta+1})]_{\theta=t-(n-1)}^{t}$, where in each step you drop the oldest entry of the buffer and append the newest experience (s_t, a_t, r_t, s_{t+1}). In the beginning, when D is still smaller than n, just append experiences.

Based on this data you have two options:

- The n-step update: In each iteration, only update the Q-value $Q(s_\theta, a_\theta)$ of the oldest data entry using the n-step return (see slide 18)
- Replay: In each iteration, loop (e.g. backward) over all entries in D and perform a vanilla Q-learning update.

Implement both variants and for $n = 10$, and compare these two learning curves to the one using the vanilla Q-Learning algorithm from the previous exercise, so that your plot contains 3 learning curves.

"Learning curve" means: Plot over n the average value

$$\hat{J} = \frac{1}{S} \sum_s V(s) \quad \text{, where} \quad V(s) = \max_a Q(s, a)$$

Every 100th step, compute the number $\hat{J}$, append it to an array, and plot this array.