Carolyn Kao (chkao831@stanford.edu)

CME 241, Winter 2021

HW15

ref: https://github.com/coverdrive/MDP-DP-RL/blob/master/src/examples/exam_problems/mrp_tdmc.py

1.Implement get_mc_value_function.

```
def get_mc_value_function(
    state_return_samples: Sequence[Tuple[S, float]]
) -> vf:
    sorted_samples = sorted(state_return_samples, key=itemgetter(0))
    return s: np.mean([r for _, r in i])
            for s, i in groupby(sorted_samples, itemgetter(0))
```

2.Implement get_probability_and_reward_functions and using its output, implement get_mrp_value_function

```
def get_probability_and_reward_fntions(srs: Sequence[Tuple[S, float, S]]) -> Tuple[ProbFunc, RewardFunc]:
    d = s: [(r, s1) for _, r, s1 in i] for s, i in
         groupby(sorted(srs, key=itemgetter(0)), itemgetter(0))

    reward_fn = s: np.mean([r for r, _ in i]) for s, i in d.items()
    prob_fn = s: s1: len(list(l1)) / len(i) for s1, l1 in
                     groupby(sorted(i, key=itemgetter(1)), itemgetter(1))
                     if s1 != 'T' for s, i in d.items()

    return prob_fn, reward_fn

def get_mrp_value_function(
    prob_fn: ProbFunc,
    reward_fn: RewardFunc
    ) -> vf:

    states = list(reward_fn.keys())
    rewards = np.array([reward_fn[s] for s in states])
    prob_matrix = np.array([[prob_fn[s][s1] if s1 in prob_fn[s] else 0.
                             for s1 in states] for s in states])
    calc = np.linalg.inv(np.eye(len(states)) - prob_matrix).dot(rewards)
    return states[i]: calc[i] for i in range(len(states))
```

3.Implement get_td_value function.

```
def get_td_value_function(
    srs: Sequence[Tuple[S, float, S]],
    iterations: int = 300000,
    lr: float = 0.3,
    decay: int = 30
) -> vf:
    return = s: [0.] for s in set(x for x, _, _ in srs)
    srs_samples = len(srs)
    for updates in range(iterations):
        s, r, s1 = srs[randint(srs_samples, size=1)[0]]
        return[s].append(return[s][-1] + lr *
                      (updates / decay + 1) ** -0.5
                      * (r + (return[s1][-1] if s1 != 'T' else 0.) - return[s][-1]))
    return s: np.mean(v[-int(len(v) * 0.9):]) for s, v in return.items()
```

4. Implement get_lstd_value function .

```
def get_lstd_value_function(
    srs: Sequence[Tuple[S, float, S]]
) -> vf:
    next_states = list(set(x for x, _, _ in srs))
    num_nt_states = len(next_states)
    phi = np.eye(num_nt_states)
    a = np.zeros((num_nt_states, num_nt_states))
    b = np.zeros(num_nt_states)
    for s, r, s1 in srs:
        p1 = phi[next_states.index(s)]
        p2 = phi[next_states.index(s1)] if s1 != 'T' else np.zeros(num_nt_states)
        a += np.outer(p1, p1 - p2)
        b += p1 * r
    return next_states[i]: v for i, v in
            enumerate(np.linalg.inv(a).dot(b))
```