1. (Convergence of policy iteration) Given an MDP with a finite state space, action space, and reward function $R(s, a, s')$. We first define the V-value function and Q-value function with reward function corresponding to state s, action a and next state s'. Specifically, the V-value of policy π at state s is defined as

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R\left(s_t, a_t, s_t'\right) \mid \pi, s_0 = s\right],$$

and the Q-value of policy π at state s and action a is defined as

$$Q^\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R\left(s_t, a_t, s_t'\right) \mid \pi, s_0 = s, a_0 = a\right].$$

Corresponding to the above V-value and Q-value function, recall that the policy iteration algorithm is equivalent to

- **Policy Evaluation:** For fixed current policy π_i, compute the V values by iterating until values converge:

$$V_{k+1}^{\pi_i}(s) \leftarrow \sum_{s'} P\left(s, \pi_i(s), s'\right)\left[R\left(s, \pi_i(s), s'\right) + \gamma V_k^{\pi_i}\left(s'\right)\right],$$

where k denotes the iterating step when computing the values.

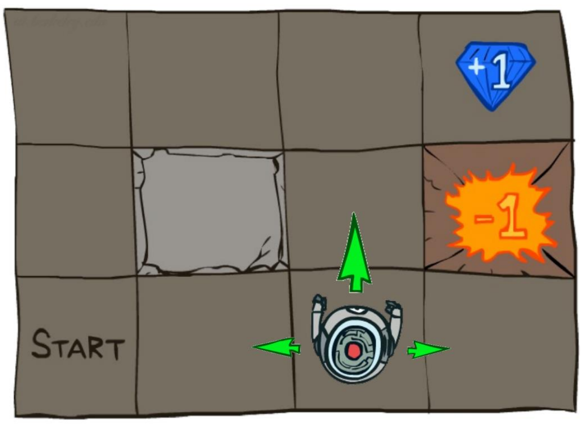
- **Policy Improvement:** For fixed values, get a better policy using policy extraction:

$$\pi_{i+1}(s) \in \arg\max_a \sum_{s'} P\left(s, a, s'\right)\left[R\left(s, a, s'\right) + \gamma V^{\pi_i}\left(s'\right)\right]$$

which is equivalent to $\pi_{i+1}(s) \in \arg\max_a Q^{\pi_i}(s, a)$.

Prove that a policy improvement step will always produce a new policy at least as good as the original one, and prove that policy iteration converges to an optimal policy.

2. (Grid World) Consider a known two-dimensional grid world environment. You will control an agent in the environment to make it to the `TERMINAL STATE`. Each action has a probability of 20% to not behave as expected, as specified in `getTransitionStatesAndProbs()`. When the agent enters the `TERMINAL STATE`, it must take the special 'exit' action to get the final reward (please see codes for more details).



(a) Recall the value iteration algorithm:

- **Init:** $\forall s$, $V_0(s) = 0$.

- **Iterate until converge:** $\forall s$,

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s' \mid s, a) \left[R(s, a, s') + \gamma V_k(s')\right] .$$

Implement a value iteration agent in `ValueIterationAgent`. Your value iteration agent is an offline planner, not a reinforcement learning agent, so the relevant training option is the maximum number of iterations of value iteration it should run (option -i) in its initial planning phase. You are also required to enable early stopping for value iteration by checking whether the maximum change of values among the states is smaller than ϵ (defined with option -e) in an iteration. Implement the following methods for `ValueIterationAgent`.

- Method `runValueIteration` computes the value function `self.values` by running the value iteration algorithm.
- Method `computeActionfromvalues(state)` computes the best action according to the value function given by `self.values`.
- Method `computeQvalueFromvalues(state, action)` returns the Q-value of the (state, action) pair given by the value function `self.values`.

(b) Recall the policy iteration algorithm in the first problem. Implement a policy iteration agent in `PolicyIterationAgent`. Again, your policy iteration agent is an offline planner and the maximum number of iterations of policy iteration is specified by option -i. Policy evaluation iterates until values converge (defined with option -e). If the policy does not change in the policy improvement phase, policy iteration stops early.

(c) Plot the utility estimates and policy actions of all states against the number of iterations, for both `ValueIterationAgent` and `PolicyIterationAgent`. Which algorithm converges faster? Explain your conclusion.