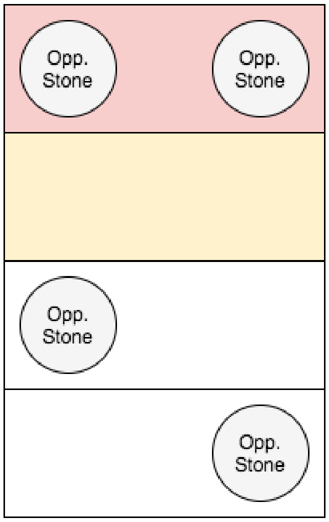
Curling

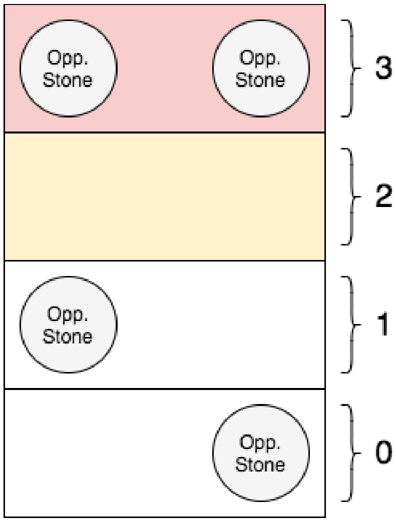
2. (18 points) You have designed a robot curling stone to enter a modified curling contest.[1] In an attempt to get your robot stone to perform well, you have designed a state and action space, a reward structure, and a transition model. The goal of the robot stone is to slide upwards on an ice sheet and stop in a target region. Your robot stone likes to show off; after each state transition, it displays the reward it receives. In addition to your robot stone, there will be a number of opponent stones on the ice, as shown below. For simplicity's sake, we will consider the opponent stones to be fixed.



Your model for the state and action spaces is as follows:

$$S \in \{t, 0, 1, 2, 3\}$$
$$a \in \{\text{“go”}, \text{“stop”}\}$$

where the states refer to the robot stone being in either a terminal state (denoted as t) or within one of the four regions below:



---

[1] Curling is an Olympic sport involving granite stones about 36 inches in circumference and 4.5 inches in height, that are directed toward target regions on an ice rink, and that might be defended or attacked by an opponent's stones. MIT has a Curling Club that you might enjoy.

You design the following reward function and (deterministic) transition model for your robot stone:

$R(s,a)$:

	action "go"	action "stop"
state 3	0	2
state 2	1	1
state 1	1	0
state 0	1	0

$$T(s,a,s'): \quad \begin{array}{c} T(0,\text{ “go”},\ 1) = 1 \\ T(1,\text{ “go”},\ 2) = 1 \\ T(2,\text{ “go”},\ 3) = 1 \\ T(3,\text{ “go”},\ \text{t}) = 1 \\ T(*,\text{ “stop”},\ \text{t}) = 1 \end{array}$$

and all other transition probabilities are 0. Here * indicates any state. Note that once the robot stone enters state t the game ends: there is no transition and zero reward out of state t (and hence no action to decide once in state t.) Together with this reward function and transition model, you specify a discount factor $\gamma = 1$.

(a) In order to enable decision making by your robot stone, you need to give it the optimal policy $\pi^*(s)$. For your reward and transition structure and discount factor $\gamma = 1$, what are the optimal Q-values, $Q^*(s,a)$? What is the optimal policy $\pi^*(s)$? Fill in the following two tables.

$Q^*(s,a)$:

	action "go"	action "stop"
state 3	0	2
state 2	3	1
state 1	4	0
state 0	5	0

$\pi^*(s)$:

	$\pi^*(s)$
state 3	stop
state 2	go
state 1	go
state 0	go

Unfortunately, your competitor has also designed a robot stone. You do not know your competitor's reward structure $R(s, a)$ or transition model $T(s, a, s')$; however, you do know they use the same state and actions spaces. Instead, you decide to use Q-learning to observe their robot stone and learn from it! For your Q-learning, use discount factor $\gamma = 1$ and learning rate $\alpha = 0.5$, with a Q table initialized to zero for all (s, a) pairs.

(b) Your competitor runs their robot through a first game, exhibiting the following experience:

step #	s	a	r	s'
1	0	"go"	1	1
2	1	"stop"	0	t

You perform Q-learning updates based on the experience above. After observing steps 1 and 2 (the first game), what is the learned $Q(0, \text{“go”})$?

Solution: We know $Q(s, a) := \alpha Q(0, a) + \alpha(r + \gamma \max_{a^i} Q(0, a_i)$ So step #1 causes the following update:

$$Q(0, \text{“go”}) = 0.5 \cdot 0 + 0.5(1 + 1 \cdot 0) = 0.5$$

What is the learned $Q(1, \text{“stop”})$?

Solution:

$$Q(1, \text{“stop”}) = 0.5 \cdot 0 + 0.5(0 + 1 \cdot 0) = 0$$

(c) Your competitor runs their robot through a second game, exhibiting the following **additional** experience:

step #	s	a	r	s'
3	0	"go"	1	1
4	1	"go"	1	2
5	2	"go"	1	3
6	3	"stop"	2	t

You perform additional Q-learning updates based on this additional experience. After completion of both games (all six steps), what are the full set of Q values you have learned for their robot? Fill in the following table.

$Q(s, a)$:

state s	action "go"	action "stop"
3	0	1
2	0.5	0
1	0.5	0
0	0.75	0

(d) We can think of learning the Q-value function for a given action as a regression problem with each state s mapped to a one-hot feature vector $x = \phi_A(s)$, where $x = [1\ 0\ 0\ 0]^T$ for state 0, $x = [0\ 1\ 0\ 0]^T$ for 1, etc., and $x = [0\ 0\ 0\ 0]^T$ for state t.

We'll focus on the action "go". We would like to come up with parameters θ, θ_0 such that $Q(s, \text{“go”}) = \theta \cdot \phi_A(s) + \theta_0 = \theta \cdot x + \theta_0$. Is there in general — for arbitrary values of our $Q(s, \text{“go”})$ — a setting of θ, θ_0 that enables representation of $Q(s, \text{“go”})$ with perfect accuracy? If so, provide the corresponding θ and θ_0. If not, explain why. (Note that we do not need to model $Q(t, a)$, since the game is over once state t has been reached.)

Solution: Yes; θ_i is simply the value for $Q(s = i, \text{“go”})$ and $\theta_0 = 0$.

Note: $\theta = [5\ 4\ 3\ 0]^T$ and $\theta_0 = 0$ would work for our optimal $Q^*(s, a)$, but we seek a more general θ corresponding to arbitrary or general $Q(s, a)$.

(e) Unfortunately, your robot's GPS system suddenly breaks, and it is no longer able to tell which of the four regions it is in. However, the robot has side cameras which can detect the opponent stones as it travels through the center of the ice, encoded as [(number of stones to immediate left) (number of stones to immediate right)]^T. You decide to use this information as state, giving the following feature transformation ϕ_B on your original state:

$$\begin{aligned}
\phi_B(3) &= [1\ 1]^T \\
\phi_B(2) &= [0\ 0]^T \\
\phi_B(1) &= [1\ 0]^T \\
\phi_B(0) &= [0\ 1]^T
\end{aligned}$$

We would still like to come up with parameters θ, θ_0 such that $Q(s, \text{“go”}) = \theta \cdot \phi_B(s) + \theta_0$, for general values of $Q(s, \text{“go”})$. Is there a setting of θ, θ_0 that enables representation of this encoding of $Q(s, \text{“go”})$ with perfect accuracy? If so, provide the corresponding θ and θ_0. If not, explain why this is not possible, and provide a feature transformation $\phi_C(\cdot)$ that does enable representation of $Q(s, \text{“go”}) = \theta \cdot \phi_C(\phi_B(s)) + \theta_0$ with perfect accuracy.

Solution: No. Let $[x_1\ x_2] = \phi_B(s)$, so $\theta_1 x_1 + \theta_2 x_2 + \theta_0 = Q(s, \text{“go”})$. $\phi_B(2)$ forces $\theta_0 = Q(2, \text{“go”})$; $\phi_B(1)$ forces θ_1; $\phi_B(0)$ forces θ_2; and we no longer have the ability to find θ for $\phi_B(3)$.

We can create ϕ_C as a one-hot encoding of state such that $\phi_C(\phi_B(s)) = \phi_A(s)$ to uniquely identify our four states (with corresponding θ and θ_0 as in the previous part) to regain perfect representational power.