Recitation 9 Solutions

1.

a. For U_A we have

$$U_A(s) = R(s) + \max_a \sum_{s'} P(s'|a,s) U_B(s')$$

and for U_B we have

$$U_B(s) = R(s) + \min_a \sum_{s'} P(s'|a,s) U_A(s') \ .$$

b. To do value iteration, we simply turn each equation from part (a) into a Bellman update and apply them in alternation, applying each to all states simultaneously. The process terminates when the utility vector for one player is the same as the previous utility vector *for the same player* (i.e., two steps earlier). (Note that typically U_A and U_B are not the same in equilibrium.)

c. The state space is shown in Figure S17.2.

d. We mark the terminal state values in bold and initialize other values to 0. Value iteration proceeds as follows:

	(1,4)	(2,4)	(3,4)	(1,3)	(2,3)	(4,3)	(1,2)	(3,2)	(4,2)	(2,1)	(3,1)
U_A	0	0	0	0	0	**+1**	0	0	**+1**	**–1**	**–1**
U_B	0	0	0	0	–1	**+1**	0	–1	**+1**	**–1**	**–1**
U_A	0	0	0	–1	+1	**+1**	–1	+1	**+1**	**–1**	**–1**
U_B	–1	+1	+1	–1	–1	**+1**	–1	–1	**+1**	**–1**	**–1**
U_A	+1	+1	+1	–1	+1	**+1**	–1	+1	**+1**	**–1**	**–1**
U_B	–1	+1	+1	–1	–1	**+1**	–1	–1	**+1**	**–1**	**–1**

and the optimal policy for each player is as follows:

	(1,4)	(2,4)	(3,4)	(1,3)	(2,3)	(4,3)	(1,2)	(3,2)	(4,2)	(2,1)	(3,1)
π^*_A	(2,4)	(3,4)	(2,4)	(2,3)	(4,3)		(3,2)	(4,2)			
π^*_B	(1,3)	(2,3)	(3,2)	(1,2)	(2,1)		(1,3)	(3,1)			

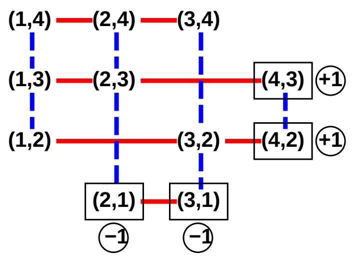

Figure S17.2 State-space graph for the game in Figure 5.17.

2.

This wording is a bit ambiguous. If we display the values after each action in one trail, we get:

	iter=0	iter=1	iter=2	iter=3
Q(Move,A)	0	0	0	0
Q(Stay,A)	0	0	0	0
Q(Move,B)	0	0	0	0
Q(Stay,B)	0	0	0	0
Q(Move,C)	0	0	0	1
Q(Stay,C)	0	0	0	0

If we think about the 3 actions as a trial and show what happens after each full trial, we get:

	iter=0	iter=1	iter=2	iter=3
Q(Move,A)	0	0	0	0.81
Q(Stay,A)	0	0	0	0
Q(Move,B)	0	0	0.9	0.9
Q(Stay,B)	0	0	0	0
Q(Move,C)	0	1	1	1
Q(Stay,C)	0	0	0	0

2. (2 pts) Characterize the weakness of Q-learning demonstrated by this example. Hint. Imagine that the chain were 100 states long.

   The Q-learning updates are very local. In a chain of 100 states we would have to go around 100 times before we had non-zero Q values in each state.

3. (3 pts) Why might a solution based on ADP (adaptive dynamic programming) be better than Q-learning?

   ADP updates all the states to make optimal use of whatever information we have discovered. So, for example, all states leading to a good state would have their values increased.

4. (2 pts) On the other hand, what are the disadvantages of ADP based approaches (compared to Q-learning)?

   - *ADP is global. If there are lots of states then the cost of these updates will be very large.*
   - *If we use ADP to learning the utility/value function directly (instead of the Q-function), this will require also learning the transition function so that we can compute the action with the best expected utility.*