Last name:_____ First name:_____ SID:_____ Class account login:_____

Collaborators: _____

CS188 Spring 2011 Written 2: Minimax, Expectimax, MDPs

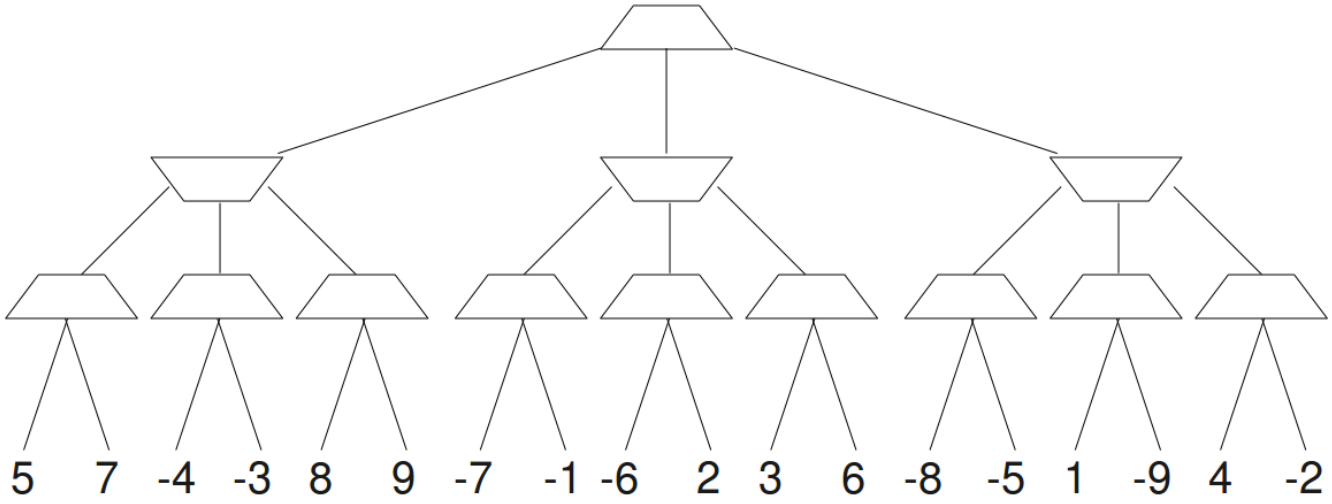
Due: Monday 2/28 at 5:29pm either in lecture or in 283 Soda Drop Box (no slip days).

Policy: Can be solved in groups (acknowledge collaborators) but must be written up individually. Recall to make a photo-copy of your solutions to allow you to resubmit for partial credit recovery. See course webpage for details.

1 [11 pts] Minimax Search and Pruning

Consider the zero-sum game tree shown below. Trapezoids that point up, such as at the root, represent choices for the player seeking to maximize; trapezoids that point down represent choices for the minimizer.

[1 pt] (a) Assuming both opponents act optimally, carry out the minimax search algorithm. Write the value of each node inside the corresponding trapezoid and highlight the action the maximizer would take in the tree.



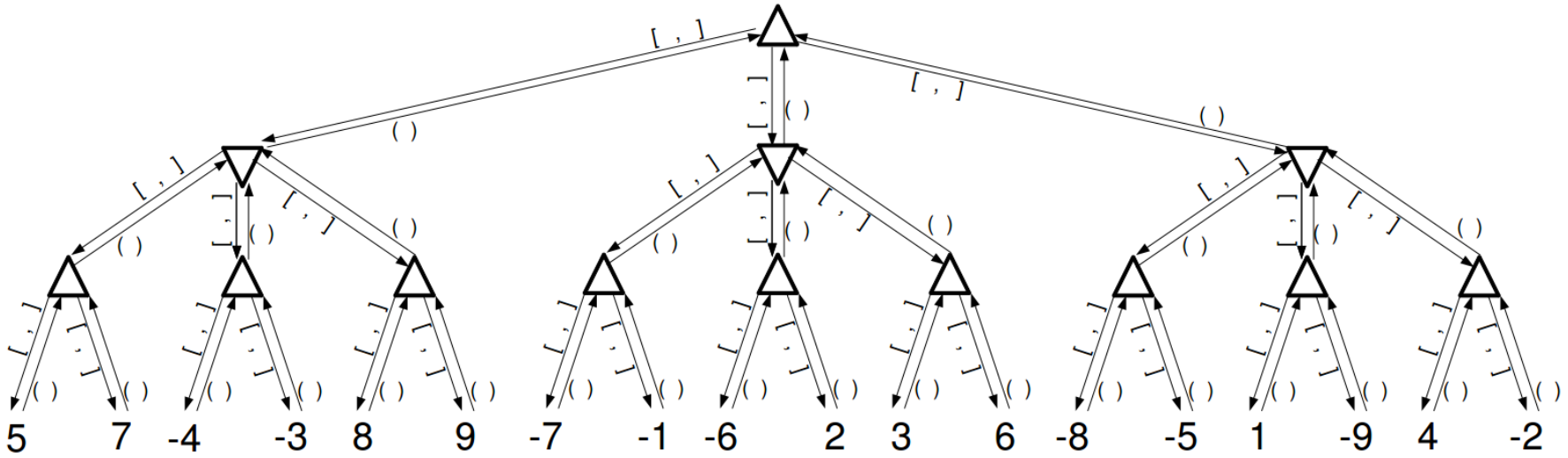
[3 pt] (b) Now reconsider the same game tree, but use α-β pruning (the tree is printed on the next page). Expand successors from left to right. In the brackets [,], record the $[\alpha, \beta]$ pair that is passed down that edge (through a call to MIN-VALUE or MAX-VALUE). In the parentheses ( ), record the value (v) that is passed up the edge (the value returned by MIN-VALUE or MAX-VALUE). Circle all leaf nodes that are visited. Put an 'X' through edges that are pruned off.

[1 pt] (c) **True / False.** Minimax and α-β pruning are guaranteed to find the same value of the top node.

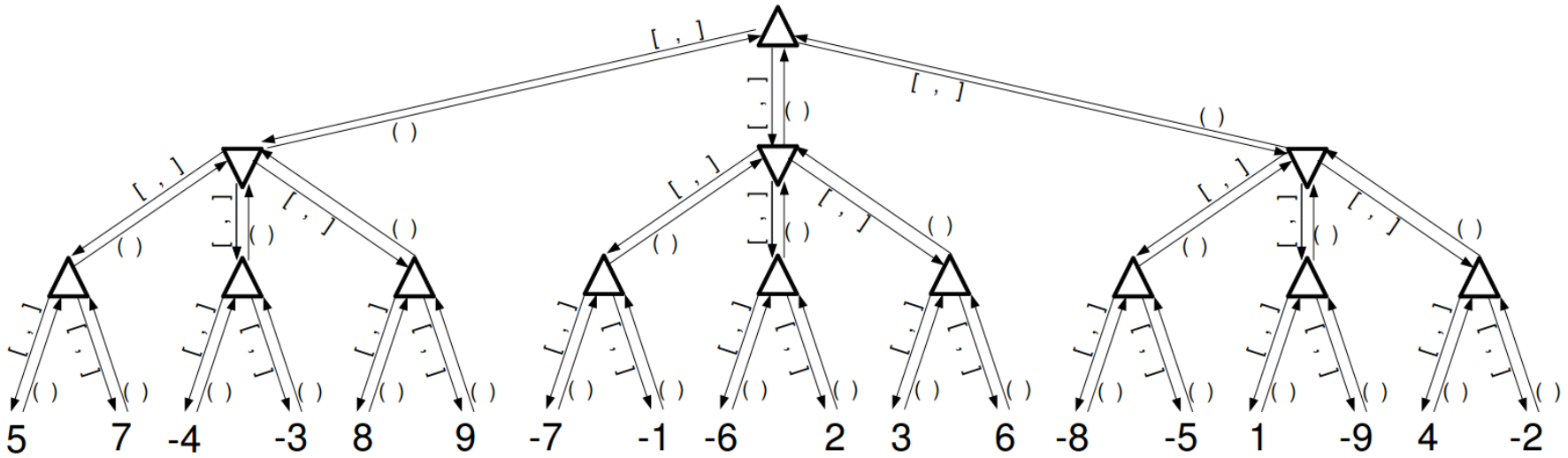
[4 pt] (d) Consider again the same game tree, searched using α-β pruning. This time, rather than expanding successors from left to right assume you can decide the order in which you expand successors. Find the order that results in exploring as few nodes as possible for this particular game. As in part (b), record the $[\alpha, \beta]$ values passed down the tree, and the (v) return values passed up. Circle all leaf nodes that are visited. Put an 'X' through edges that are pruned off.

[2 pt] (e) Assume you have an evaluation function which for each node can provide an estimate of the minimax value (though the estimate will not be perfect). How can you use these minimax value estimates to guide the order in which successors are expanded, with the goal of minimizing the number of leaf nodes visited while running the α-β pruning algorithm?

(b)



(d)



2 [8 pts] Expectimax for cs188-Blackjack

Blackjack is the most widely played casino betting game in the world. The goal of the game is to be dealt a hand whose value is as close to 21 as possible without exceeding it. If the current value of a player's hand is less than 21, the player can "hit", or be dealt a single card, in hopes of acquiring a hand with higher value. However, the player runs the risk of "busting", or going over 21, which results in an immediate loss. In casino play, players bet independently against a dealer, who plays according to a fixed set of rules that govern when he should hit or stay.

In this problem set, we consider a simplified variant called cs188-Blackjack.

- There are only 3 cards in the deck: 5's, 10's and 11's. Each card appears with equal probability.
- The casino has invented an infinite deck. The probability of being dealt any given card is independent of the cards already dealt.
- To model the action of a dealer, we assume the casino gives fixed payoffs according to the following schedule (in dollars)

Hand Value	Payoff
0-14	0
15	3
16	3
17	3
18	3
19	3
20	9
21	12
Bust	-6

- There are two actions available: Hit, which draws a card uniformly at random and adds its value to your current score, and Stay, which ends the game and yields the above payoff. If your score goes above 21 the game ends immediately with a payoff of -6. It is not possible to hit on 21. Thus if you ever arrive at a hand value of ≥ 21, there are no actions possible.

You are playing a hand of cs188-Blackjack. You have been dealt 1 card, and its value is 11.

[2 pts] (a) Build the expectimax tree for this game, starting from your current hand and including all chance and max nodes. In your tree, you should put "hit" actions to the left of "stay" actions, and you should order max nodes below the same chance node in increasing order of the *hand's* value (from left to right). Write the value of each state next to the given node. What is your optimal strategy? Specify your actions at all max nodes in the tree.

[2 pt] (b) Unfortunately, you are playing at a table with an unscrupulous dealer who is rigging the deck. Every time he deals a card, instead of dealing you a random card, he gives you the worst possible card you could get at that moment. What is the value of the game now and what is your optimal strategy?

[2 pt] (c) When you complain about the cheating dealer to the pit boss, a new dealer is brought in. This dealer is extremely nice: half of the time, when his boss is watching, he deals you a random card. The other half of the time, he deals you the best possible card you could get at that moment. Draw out the game tree for this (using the same instructions as (a)). What is the value of the game now and what is your optimal strategy?

[2 pt] (d) The casino owner, anxious about dwindling interest in cs188-Blackjack, asks you to help him rework the game. He would like to increase the payouts for a value of 21 to $x. What is the minimal value of $x so that the optimal strategy for a player holding 16 changes? Assume fair dealers (as was assumed in part (a)).

3 [12 pts] Mission to Mars

You control a solar-powered Mars rover. It can at any time drive `fast` or `slow`. You get a reward for the distance crossed, so `fast` gives +10 while `slow` gives +4. Your rover can be in one of three states: `cool`, `warm`, or `off`. Driving `fast` tends to heat up the rover, while driving `slow` tends to cool it down. If the rover overheats, it shuts `off`, forever. The transitions are shown to the right. Because critical research depends on the observations of the rover, there is a discount of $\gamma = 0.9$.

s	a	s'	$T(s, a, s')$
cool	slow	cool	1
cool	fast	cool	1/4
cool	fast	warm	3/4
warm	slow	cool	1/4
warm	slow	warm	3/4
warm	fast	warm	7/8
warm	fast	off	1/8

[1pt] (a) How many possible deterministic stationary policies are there?

[1 pt] (b) What is the value of the state `cool` under the policy that always goes `slow`?

[1 pt] (c) Fill in the following table of depth-limited values from value iteration for this MDP. Note that this part concerns (optimal) value iteration, not evaluation of the always-`slow` policy.

s	$V_0(s)$	$V_1(s)$	$V_2(s)$
cool	0		
warm	0		
off	0	0	0

[1 pt] (d) How many rounds of value iteration will it take for the values of all states to converge to their exact values? (State infinitely many if you think it will only have converged after infinitely many.)

[1pt] (e) What is the optimal policy for $\gamma = .9$?

s	$\pi^*(s)$
cool	
warm	

[1pt] (f) What are the optimal values for the optimal policy when $\gamma = .9$?

s	$V^*(s)$
cool	
warm	
off	0

[1pt] (g) Central command, demanding results `faster`, tells you that they don't care about the future of the rover. In particular, they say that your discount parameter γ should be .5. What are the optimal policy and values now?

s	$\pi^*(s)$
`cool`	
`warm`	

s	$V^*(s)$
`cool`	
`warm`	
`off`	0

[2pt] (h) Now imagine that you do not know in advance what the thermal responses of the rover will be, so you decide to do Q-learning. You observe the following sequence of transitions:

1. (`cool`, `slow`, 4) → `cool`
2. (`cool`, `fast`, 10) → `cool`
3. (`cool`, `fast`, 10) → `cool`
4. (`cool`, `fast`, 10) → `warm`
5. (`warm`, `slow`, 4) → `cool`

Give the Q-values for each step in this sequence as it is processed by Q-learning, assuming a learning rate (α) of 0.5 and a discount factor $\gamma = 0.9$. For example, $Q_3(s, a)$ should be the Q-values after processing transitions 1, 2, and 3.

s	a	$Q_0(s,a)$	$Q_1(s,a)$	$Q_2(s,a)$	$Q_3(s,a)$	$Q_4(s,a)$	$Q_5(s,a)$
`cool`	`slow`	0					
`cool`	`fast`	0					
`warm`	`slow`	0					
`warm`	`fast`	0					

[3pt] (i) An ε-greedy policy may not be the right choice for Q-learning in this situation given that the rover, once `off`, is lost forever. On the other hand, it may not be optimal to never risk going `fast` from a `warm` state – perhaps the planet is very cold and there is little risk. Imagine that you know that T(`cool`,`fast`,`warm`) = T(`warm`,`fast`,`off`) for all environments. *Note:* this property is not true for the transitions above!

State a modified Q-learning update and procedure that exploits this knowledge and from which you will learn all optimal Q-values without ever visiting the Q-state (`warm`,`fast`), assuming you do visit all other Q-states infinitely often. Be precise (i.e. use math).