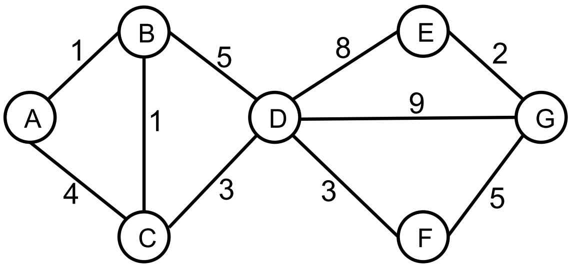
Exam 1 - Practice Problems

CS 182 - Artificial Intelligence

October 25, 2018

1 Search



Node	h_1	h_2	h_3
A	9.5	10	14
B	9	12	13
C	8	10	10
D	7	8	8
E	1.5	1	2
F	4	4.5	4
G	0	0	0

Consider the state space graph shown above. A is the start state and G is the goal state. The costs for each edge are shown on the graph. Each edge can be traversed in both directions. The table on the right shows three heuristics h_1, h_2 and h_3.

(a) Are h_1, h_2 and h_3 consistent? Are they admissible?

(b) For each of the following graph search strategies, mark which, if any, of the listed paths it could return. Note that for some search strategies the specific path returned might depend on tie-breaking behavior. In any such cases, make sure to mark *all* paths that could be returned under some tie-breaking scheme.

Search Algorithm	**A-B-D-G**	**A-C-D-G**	**A-B-C-D-F-G**
Depth first search			
Breadth first search			
Uniform cost search			
A* search with heuristic h_1			
A* search with heuristic h_2			
A* search with heuristic h_3			

(c) Suppose you are completing the new heuristic function h_4 shown below. All the values are fixed except $h_4(B)$.

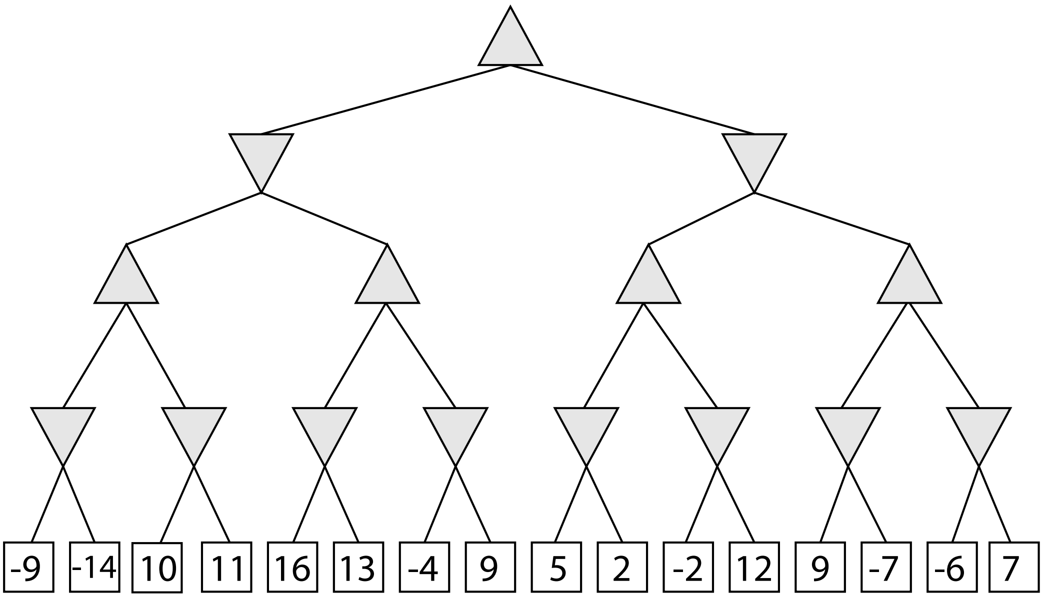
Node	A	B	C	D	E	F	G
h_4	10	?	9	7	1.5	4.5	0

For each of the following conditions, write the set of values that are possible for $h_4(B)$. (1) What values of $h_4(B)$ make h_4 admissible? (2) What values of $h_4(B)$ make h_4 consistent?

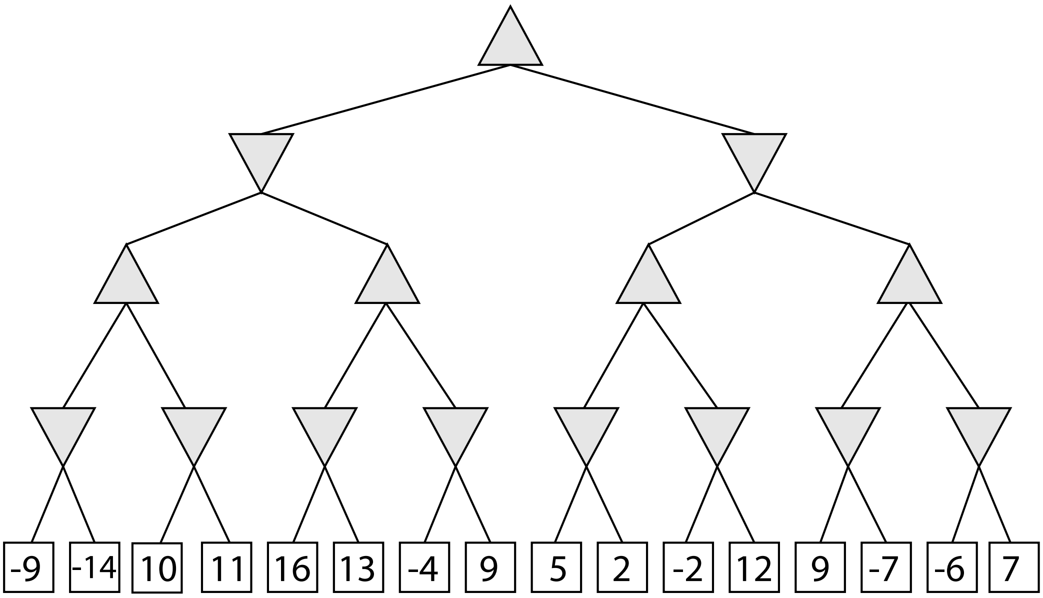
(d) What values of $h_3(B)$ will cause A* graph search to expand node A, then node C, then node B, then node D in order?

2 Adversarial Search

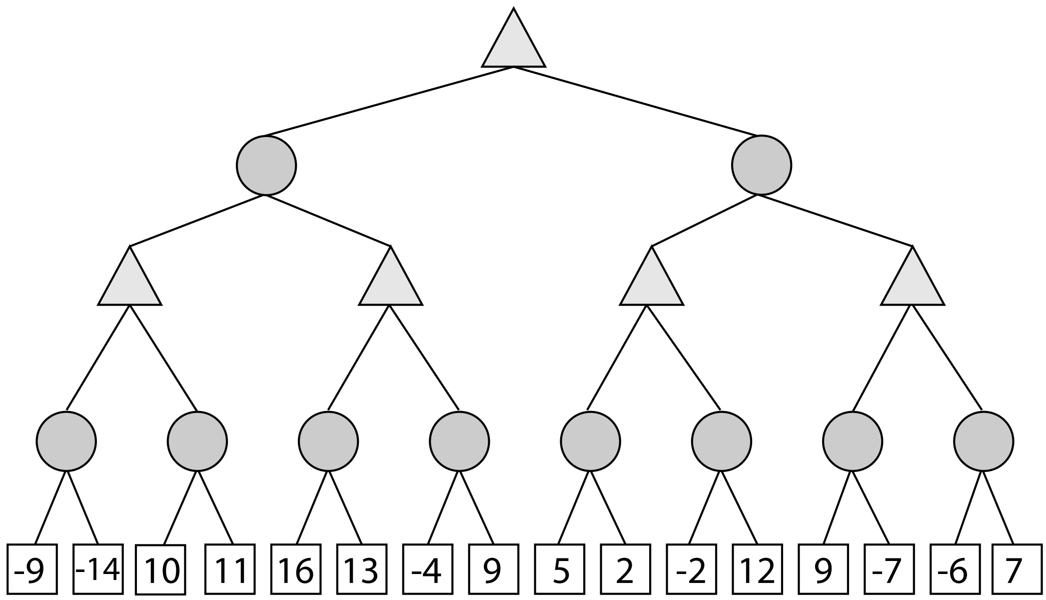
(a) Below you can see a search tree. Fill out the values of the nodes according to minimax search. What action would the agent take at the root?



(b) Mark the edges that would be pruned when using Alpha-Beta pruning? Also write the alpha and beta value as well as the value on each node.



(c) After running the algorithm, you notice that the other player is not playing optimally. Instead, the player picks their action uniformly at random. Recompute the values of the nodes. What is the action at the root now?



(d) Your computer can only use 1MB of memory while running the algorithm. A single node takes 10B to store. Given a branching factor of 3, Compute an upper bound on the depth of a tree that your computer can conduct AB-pruning on.

(e) Your computer can check 10,000 nodes per second Is it feasible to solve the problem for the maximum depth within 2 minutes? If not, what strategy do you recommend instead?

(f) True or false. For every game tree, the utility obtained by MAX using minimax decisions against a suboptimal MIN will be never be lower than the utility obtained playing against an optimal MIN. Justify your response.

Can you come up with a game tree in which MAX can do still better using a suboptimal strategy against a suboptimal MIN?

3 CSP

You are designing a menu for a special event. There are several choices, each represented as a variable: (A)ppetizer, (B)everage, main (C)ourse, and (D)essert. The domains of the variables are as follows:

- A: (v)eggies, (e)scargot
- B: (w)ater, (s)oda, (m)ilk
- C: (f)ish, (b)eef, (p)asta
- D: (a)pple pie, (i)ce cream, (ch)eese

Because all of your guests get the same menu, it must obey the following dietary constraints:

- (i) Vegetarian options: The appetizer must be veggies or the main course must be pasta or fish (or both).
- (ii) Total budget: If you serve the escargot, you cannot afford any beverage other than water.
- (iii) Calcium requirement: You must serve at least one of milk, ice cream, or cheese.

(a) Draw the constraint graph over the variables

(b) Imagine we assign $A = e$. Cross out the eliminated values to show the domains of the variable after forward checking

A		e	
B	w	s	m
C	f	b	p
D	a	i	ch

(c) Imagine again that $A = e$. Cross out the eliminated values after enforcing arc consistency.

A		e	
B	w	s	m
C	f	b	p
D	a	i	ch

(d) Give a solution for this CSP or show that none exists.

(e) Define in your own words the terms constraint, backtracking search, arc consistency, backjumping, min-conflicts, and cycle cutset.

4 MDP and RL

(a) Suppose that we define the utility of a state sequence to be the maximum reward obtained in any state in the sequence. Show that this utility function does not result in stationary preferences between state sequences. Is it still possible to define a utility function on states such that MEU decision making gives optimal behavior?

(b) Can all MDPs be solved using expectimax search? Justify your answer

(c) Let's consider a two-player MDP that correspond to a zero-sum, turn-taking game. Let the players be A and B, and let $R(s)$ be the reward for player A in state s. (The reward for B is always equal and opposite.). Let $V_A^*(s)$ be the utility of state s when it is A's turn to move in s, and let $V_B^*(s)$ be the utility of state s when it is B's turn to move in s. All rewards and utilities are calculated from A's point of view (just as in a minimax game tree). Write down the definitions of $V_A^*(s)$ and $V_B^*(s)$ in terms of expected future utility.

(d) Explain how to do two-player value iteration with these equation. Additionally, state how to check whether the algorithm has converged.

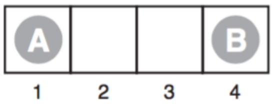


Figure 1: The starting position for a simple game. The two players take turns moving, and each player must move his token to an open adjacent space in either direction. If the opponent occupies an adjacent space, then a player may jump over the opponent to the next open space if any. (For example, if A is on 3 and B is on 2, then A may move back to 1.) The game ends when one player reaches the opposite end of the board. If player A reaches space 4 first, then the value of the game to A is $+1$; if player B reaches space 1 first, then the value of the game to A is -1.

(e) Consider the game described in the figure above. Draw the state space (rather than the game tree), showing the moves by A as solid lines and moves by B as dashed lines. Mark each state with $R(s)$. You will find it helpful to arrange the states (s_A, s_B) on a two-dimensional grid, using s_A and s_B as "coordinates."

(f) Now apply two-player value iteration to solve this game, and derive the optimal policy. Use a γ of 1 for the derivation.

(g) When using features to represent the Q-function is it guaranteed that the feature-based Q-learning finds the same optimal Q^* as would be found when using a tabular representation for the Q-function?

(h) Why is temporal difference (TD) learning of Q-values (Q-learning) superior to TD learning of values?