CS 188 Fall 2023 | Introduction to Artificial Intelligence | Exam Prep 4 Solutions

Q1. Games

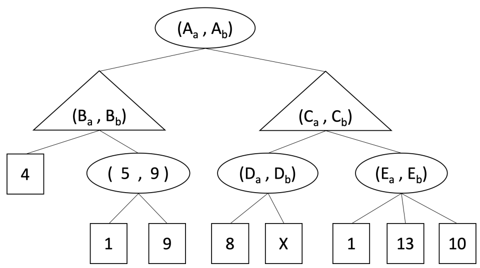
Alice is playing a two-player game with Bob, in which they move alternately. Alice is a maximizer. Although Bob is also a maximizer, Alice believes Bob is a minimizer with probability 0.5, and a maximizer with probability 0.5. Bob is aware of Alice's assumption.

In the game tree below, square nodes are the outcomes, triangular nodes are Alice's moves, and round nodes are Bob's moves. Each node for Alice/Bob contains a tuple, the left value being Alice's expectation of the outcome, and the right value being Bob's expectation of the outcome.

Tie-breaking: choose the left branch.

The left values are Alice's expectations, and are the only thing Alice can refer to when making decisions.

The right values are Bob's expectations, and they also accurately track the expected outcome of the game according to each choice of branching (regardless of it is Alice's or Bob's decision, since Bob has all the information). Hence the right values are accurate information about the game, and would be what Bob looks at when making his decisions. However, when it is Alice's turn to make decisions, Bob will think about how Alice would maximize the outcome w.r.t to what she believes, and he will update his expectations accordingly.



(a) In the blanks below, fill in the tuple values for tuples (B_a, B_b) and (E_a, E_b) from the above game tree.

$(B_a, B_b) = ($ 5 , 9 $)$

$(E_a, E_b) = ($ 7 , 13 $)$

For a square node, its value v means the same to Alice and Bob, i.e., we can think of it as a tuple (v,v).

The left value of Alice's nodes is the maximum of the left values of it's children nodes, since Alice believes that the values of the nodes are given by left values, and it's her turn of action, so she will choose the largest value.

The right value of Alice's nodes is the right value from the child node that attains the maximum left value since Bob's expectation is consistent with how Alice will act.

So for a triangular node, its tuple is the same as its child that has the maximum left value.

The left value of Bob's nodes is the average of the maximum and minimum of the left values of it's children nodes since Alice believes Bob is 50% possible to be adversarial and 50% possible to be friendly.

The right value of Bob's nodes is the maximum of the right values of the immediate children nodes since Bob would choose the branch that gives the maximum outcome during his turn.

So for a round node, left = 0.5(max(children.left) + min(children.left)), and right = max(children.right).

(b) In this part, we will determine the values for tuple (D_a, D_b).

(i) $D_a =$ ○ 8 ○ X ○ 8+X ● 4+0.5X ○ min(8,X) ○ max(8,X)

(ii) $D_b =$ ○ 8 ○ X ○ 8+X ○ 4+0.5X ○ min(8,X) ● max(8,X)

It's a round node, so left = 0.5(max(children.left) + min(children.left)), and right = max(children.right).
Its children: (8,8) and (X,X). So left = 0.5(8+X) = 4+0.5X, and right = max(8, X).

(c) Fill in the values for tuple (C_a, C_b) below. For the bounds of X, you may write scalars, ∞ or $-\infty$.
If your answer contains a fraction, please write down the corresponding **simplified decimal value** in its place. (i.e., 4 instead of $\frac{8}{2}$, and 0.5 instead of $\frac{1}{2}$).

1. If $-\infty <$ X $<$ [6], $(C_a, C_b) =$ ([7], [13])
2. Else, $(C_a, C_b) =$ ([4+0.5X], max([8], [X]))

It's a triangular node, so its tuple is the same as its child that has the maximum left value.
Its children: (4+0.5X, max(8,X)) and (7, 13).
So if 4+0.5X < 7, i.e. $-\infty < X < 6$, it's the same as child node (7, 13), and otherwise it's (4+0.5X, max(8,X)).

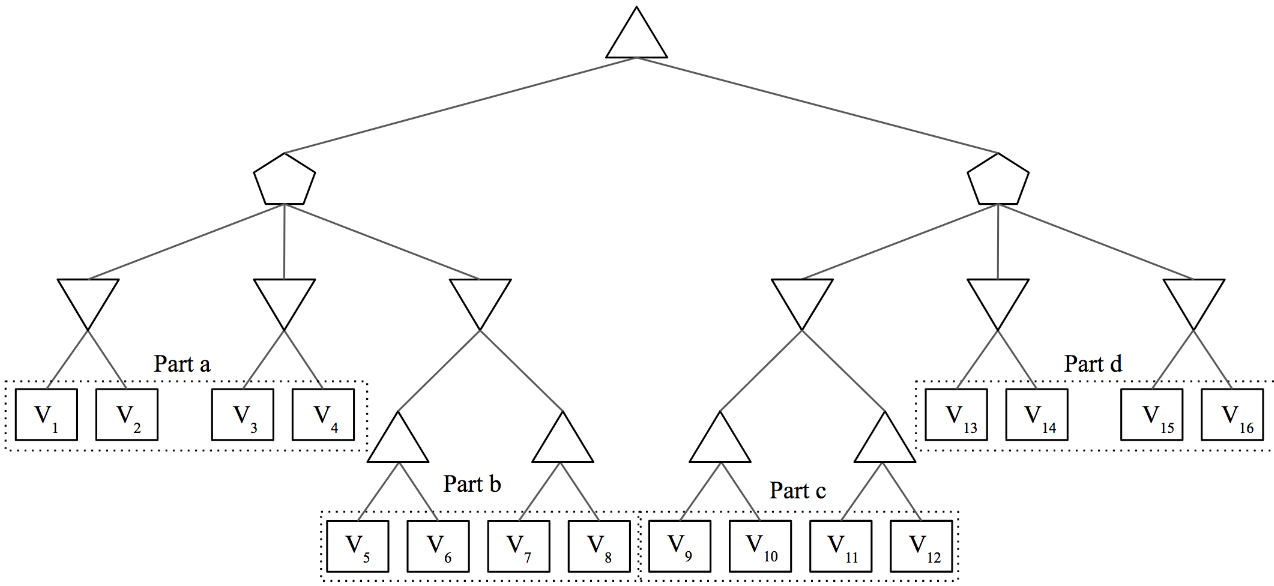
(d) Fill in the values for tuple (A_a, A_b) below. For the bounds of X, you may write scalars, ∞ or $-\infty$.
If your answer contains a fraction, please write down the corresponding **simplified decimal value** in its place. (i.e., 4 instead of $\frac{8}{2}$, and 0.5 instead of $\frac{1}{2}$).

1. If $-\infty <$ X $<$ [6], $(A_a, A_b) =$ ([6], [13])
2. Else, $(A_a, A_b) =$ ([4.5+0.25X], max([9], [X]))

It's a round node, so left = 0.5(max(children.left) + min(children.left)), and right = max(children.right).
Its children: (5,9) and node "Part (c)".
If $-\infty < X < 6$, these children are (5,9) and (7, 13).
left = 0.5(max(children.left) + min(children.left)) = 0.5(5+7) = 6
right = max(children.right) = max(9, 13) = 13.
Otherwise ($6 < X < +\infty$), these children are (5,9) and (4+0.5X, max(8,X)).
left = 0.5(max(children.left) + min(children.left)) = 0.5(5+4+0.5X) = $4.5 + 0.25X$
right = max(children.right) = max(9, max(8,X)) = max(9,X).

Q2. MedianMiniMax

You're living in utopia! Despite living in utopia, you still believe that you need to maximize your utility in life, other people want to minimize your utility, and the world is a 0 sum game. But because you live in utopia, a benevolent social planner occasionally steps in and chooses an option that is a compromise. Essentially, the social planner (represented as the pentagon) is a median node that chooses the successor with median utility. Your struggle with your fellow citizens can be modelled as follows:



There are some nodes that we are sometimes able to prune. In each part, mark all of the terminal nodes such that **there exists a possible situation** for which the node **can be pruned**. In other words, you must consider **all** possible pruning situations. Assume that evaluation order is **left to right** and all V_i's are **distinct.**

Note that as long as there exists ANY pruning situation (does not have to be the same situation for every node), you should mark the node as prunable. Also, alpha-beta pruning does not apply here, simply prune a sub-tree when you can reason that its value will not affect your final utility.

(a)
- [ ] V_1
- [ ] V_2
- [ ] V_3
- [ ] V_4
- [x] None

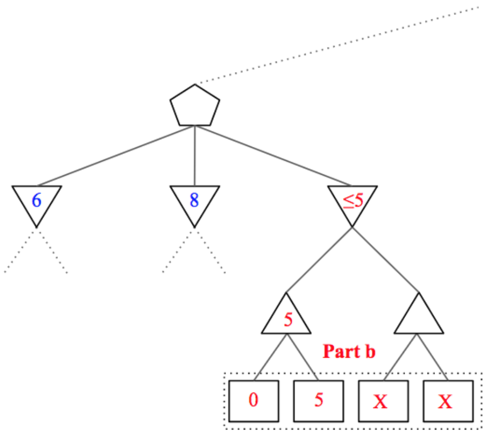
(b)
- [ ] V_5
- [x] V_6
- [x] V_7
- [x] V_8
- [ ] None

(c)
- [ ] V_9
- [ ] V_{10}
- [x] V_{11}
- [x] V_{12}
- [ ] None

(d)
- [ ] V_{13}
- [x] V_{14}
- [x] V_{15}
- [x] V_{16}
- [ ] None

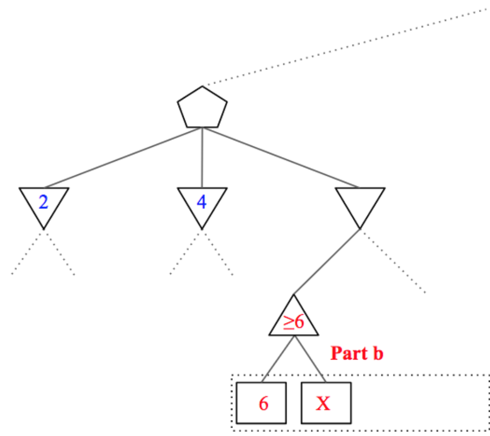
Part a:

For the left median node with three children, at least two of the childrens' values must be known since one of them will be guaranteed to be the value of the median node passed up to the final maximizer. For this reason, none of the nodes in part a can be pruned.



The value of this subtree will only get smaller.

The median node will **NOT** choose the value of this subtree. 6 is the median.



The value of this subtree will only get bigger.

*If the value of this subtree is chosen by the minimizer**, it will **NOT** be chosen by the median node.

**It is possible that the median is the value of the subtree to the right that we haven't looked at yet*

Part b (pruning V_7, V_8):

Let min_1, min_2, min_3 be the values of the three minimizer nodes in this subtree.

In this case, we may not need to know the final value min_3. The reason for this is that we may be able to put a bound on its value after exploring only partially, and determine the value of the median node as either min_1 or min_2 if $min_3 \leq \min(min_1, min_2)$ or $min_3 \geq \max(min_1, min_2)$.

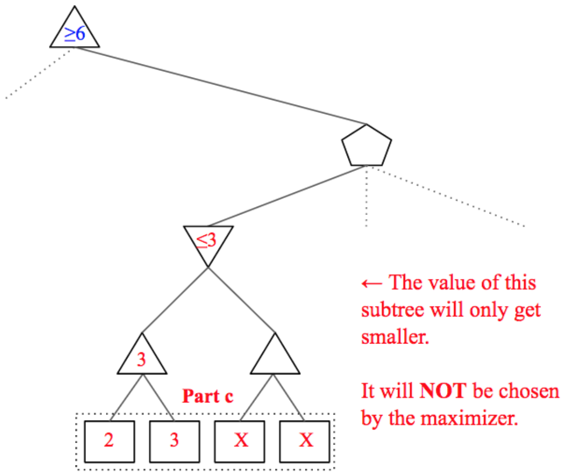
We can put an upper bound on min_3 by exploring the left subtree V_5, V_6 and if $\max(V_5, V_6)$ is lower than both min_1 and min_2, the median node's value is set as the smaller of min_1, min_2 and we don't have to explore V_7, V_8 in Figure 1.

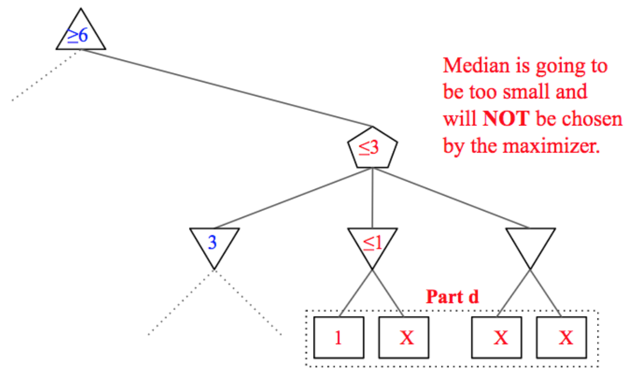
Part b (pruning V_6):

It's possible for us to put a lower bound on min_3. If V_5 is larger than both min_1 and min_2, we do not need to explore V_6.

The reason for this is subtle, but if the minimizer chooses the left subtree, we know that $min_3 \geq V_5 \geq \max(min_1, min_2)$ and we don't need V_6 to get the correct value for the median node which will be the larger of min_1, min_2.

If the minimizer chooses the value of the right subtree, the value at V_6 is unnecessary again since the minimizer never chose its subtree.





Part c (pruning V_{11}, V_{12}):
Assume the highest maximizer node has a current value $max_1 \geq Z$ set by the left subtree and the three minimizers on this right subtree have value min_1, min_2, min_3.

In this part, if $min_1 \leq \max(V_9, V_{10}) \leq Z$, we do not have to explore V_{11}, V_{12}. Once again, the reasoning is subtle, but we can now realize if either $min_2 \leq Z$ or $min_3 \leq Z$ then the value of the right median node is for sure $\leq Z$ and is useless.

Only if both $min_2, min_3 \geq Z$ will the whole right subtree have an effect on the highest maximizer, but in this case the exact value of min_1 is not needed, just the information that it is $\leq Z$. Clearly in both cases, V_{11}, V_{12} are not needed since an exact value of min_1 is not needed.

We will also take the time to note that if $V_9 \geq Z$ we do have to continue the exploring as V_{10} could be even greater and the final value of the top maximizer, so V_{10} can't really be pruned.

Part d (pruning V_{14}, V_{15}, V_{16}):
Continuing from part c, if we find that $min_1 \leq Z$ and $min_2 \leq Z$ we can stop.

We can realize this as soon we explore V_{13}. Once we figure this out, we know that our median node's value must be one of these two values, and neither will replace Z so we can stop.