CS 188 Fall 2021 | Introduction to Artificial Intelligence | Challenge Q9 HW3

- **Due:** Tuesday 9/21 at 10:59pm.
- **Policy:** Can be solved in groups (acknowledge collaborators) but must be submitted individually.
- **Make sure to show all your work and justify your answers**.
- **Note:** This is a typical exam-level question. On the exam, you would be under time pressure, and have to complete this question on your own. We strongly encourage you to first try this on your own to help you understand where you currently stand. Then feel free to have some discussion about the question with other students and/or staff, before independently writing up your solution.
- Your submission on Gradescope should be a PDF that matches this template. Each page of the PDF should align with the corresponding page of the template (page 1 has name/collaborators, question begins on page 2.). **Do not reorder, split, combine, or add extra pages**. The intention is that you print out the template, write on the page in pen/pencil, and then scan or take pictures of the pages to make your submission. You may also fill out this template digitally (e.g. using a tablet.)

First name	NAMEERA
Last name	FAISAL AKHTAR
SID	3034244256
Collaborators	SARA INAM

For staff use only:

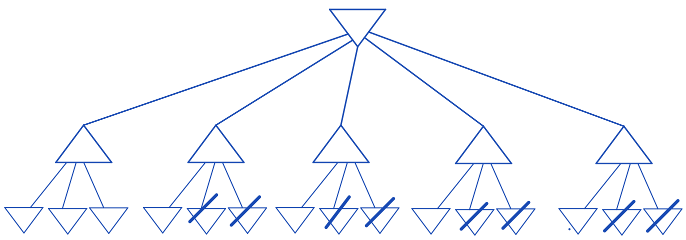
Q9.	Challenge Question (Multi-Agent Search)	/17

Q9. [17 pts] Challenge Question (Multi-Agent Search)

Let's consider a game played on a board with M rows. The maximizing agent (PacMax) has access to N different types of pieces while the minimizing agent (PacAdv) has access to only one type of piece. At each round, the PacAdv puts one piece to the right end of a row. After that PacMax chooses a piece to put on the left side of that row.

Assume we run alpha-beta pruning on the game tree described above for one turn.

9.1) (2pts) If $M = 5$ and $N = 3$, then what are the upper and lower bounds on the number of leaf nodes we can prune?

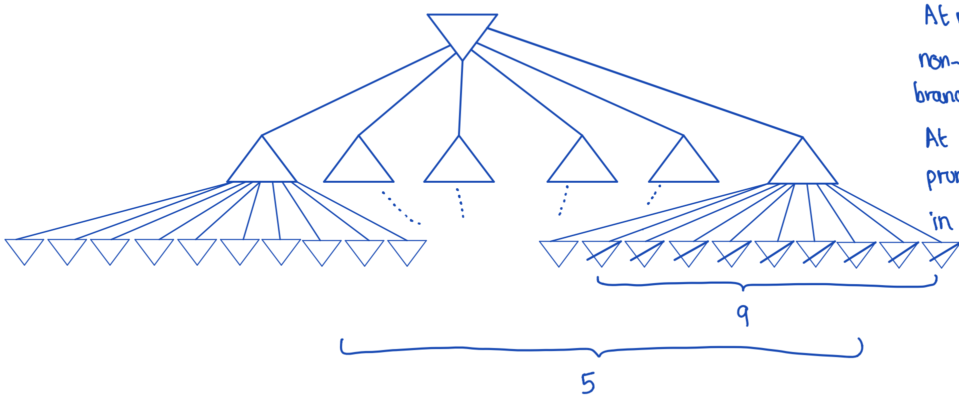


At most, you can prune all the non-leftmost leaf nodes in each branch (4 branches x 2 nodes = 8 nodes)

At the minimum, you don't have to prune any nodes at all (if nodes in descending order)

$\boxed{\text{Upper bound: } 8 \quad \text{Lower bound: } 0}$

9.2) (1pts) If we choose $M = 6$ and $N = 10$, then what is the upper bound on the number of leaf nodes pruned?



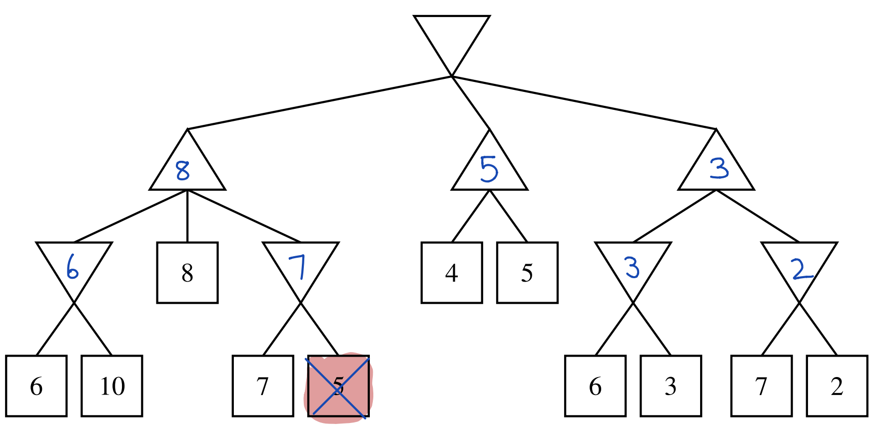
= 9 x 5 = 45 nodes

At most, you can prune all the non-leftmost leaf nodes in each branch (5 branches x 9 nodes = 45 nodes)

At the minimum, you don't have to prune any nodes at all (if nodes in descending order)

$\boxed{\text{Upper bound: } 45 \quad \text{Lower bound: } 0}$

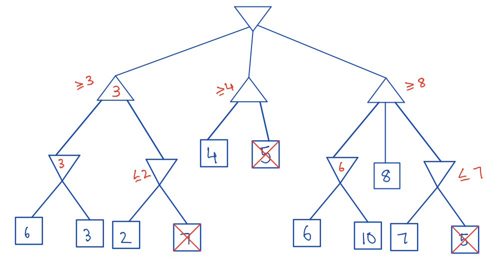
Now let's consider the following game tree:



9.3) (1pts) Assuming that we visit branches from left to right, if we perform alpha-beta pruning how many leaf nodes will be pruned?

$\boxed{\text{1 node gets pruned}}$. This is the highlighted node above i.e. 5 because it doesn't matter what 5 is – the maximizer node will choose 8 anyway since the minimizer will be less than or equal to 7.

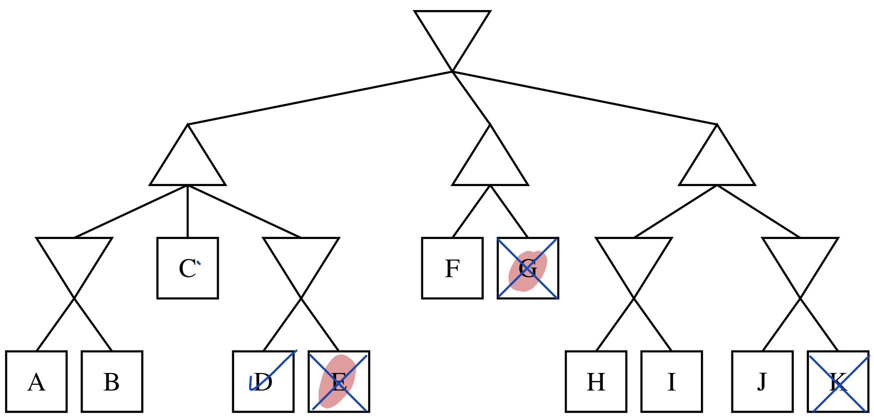
9.4) (2pts) Now let's consider a slightly more involved case. For this part, assume again that children of maximizing nodes are visited from left to right. If we re-order the branches of minimizing nodes in order to maximize the number of leaf nodes pruned, what is the number of leaf nodes we can now prune? Assume that we do not prune on equality.



In the tree on the right, we prune $\boxed{3 \text{ nodes}}$.

This works b/c in order to prune as many nodes as possible, we put 2 as early as possible since it is the min value.

Consider the tree of the previous question, in which we have shuffled the leaf node values so that we check the least number of nodes. The structure of the tree remains unchanged. The new ordering of the leaf node values is $A, B, C, \ldots$, while in the previous question the ordering was $6, 10, \ldots$. Assume again that we do not prune on equality and that we traverse from left to right.



9.5) (2pts) Consider all these possible aforementioned orderings. Which of the following are the possible root node values?

A. 2
B. 3
C. 4
D. 5
E. 6
F. 7
G. 8
H. 9
I. 10

In order for E to get pruned, we need $D < \max(X, C)$.

The root node cannot be 2 or 3 because 2 or 3 can never be the maximum of three values.

The root cannot be 9 because 9 isn't in the tree.

The root node cannot be 8 or 10 because 8 or 10 can never be the minimum of three values.

The root cannot be 7 b/c in order for pruning to occur, 7 must be the min of 7, 8, 10 and in leftmost node at level 2. However, this isn't possible due to the $D < \max(X, C)$ condition.

9.6) (3pts) If the value at the root is 4 for the new ordering then which of the following leaf nodes are going to have a value less than or equal to 5?

A. A
B. B
C. C
D. D
E. E
F. F
G. G
H. H
I. I
J. J
K. K
L. None

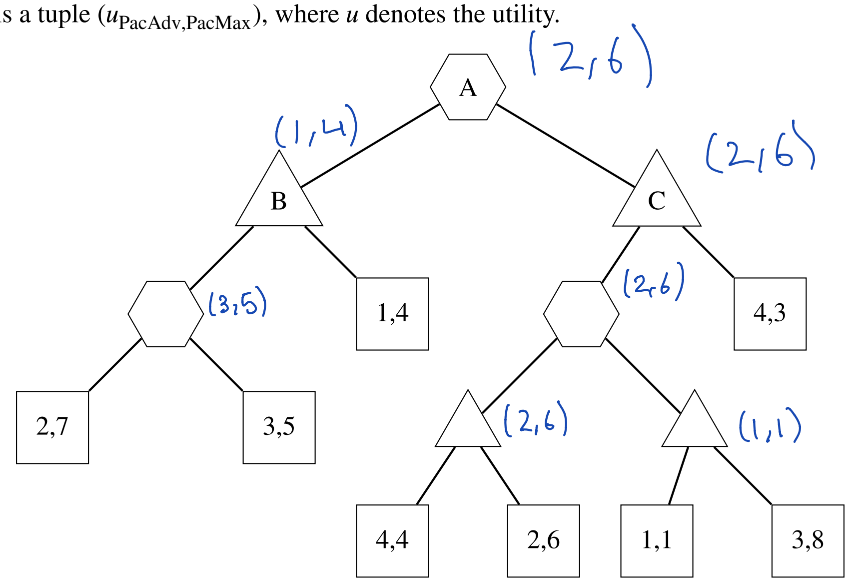
If 4 was the root and we want max pruning, we know that 4 must come from the leftmost node at depth 2 i.e Node Z.

In order for E to get pruned, $X \le 4$ → not necessarily A or B!

$C \le 4$

$D \le 4$

Assume that PacMax knows PacAdv's utility function but PacAdv does not know PacMax's utility. Now PacMax is no longer interested in matching PacAdv's move on the same row. On the contrary, Pacmax wants to put as many pieces on the board as possible. Find the PacAdv's and PacMax's utilities on the following tree. If there is a tie choose the leftmost branch. Hexagons denote when PacAdv is deciding, triangles denote Pacmax's decision nodes and squares denote the leaf nodes. The format in all nodes is a tuple ($u_{\text{PacAdv,PacMax}}$), where u denotes the utility.



9.7) (2pts) Write down the utilities at node B as a tuple ($u_{\text{PacAdv,B}}, u_{\text{PacMax,B}}$).

$\boxed{(1,4)}$. Since the hexagon maximises the utility for PacAdv, (2,7) ⬡ (3,5) becomes (3,5).

Then, since the triangle minimizes the utility for PacAdv, (3,5) Δ (1,4) becomes (1,4).

Hence, the utilities at node B are (1,4).

9.8) (2pts) Write down the utilities at node C as a tuple ($u_{\text{PacAdv,C}}, u_{\text{PacMax,C}}$).

$\boxed{(2,6)}$. Since the triangle minimizes the utility for PacAdv, (4,4) Δ (2,6) becomes (2,6).

and (1,1) Δ (3,8) becomes (1,1).

Then, since the hexagon maximises the utility for PacAdv, (2,6) Δ (1,1) becomes (2,6).

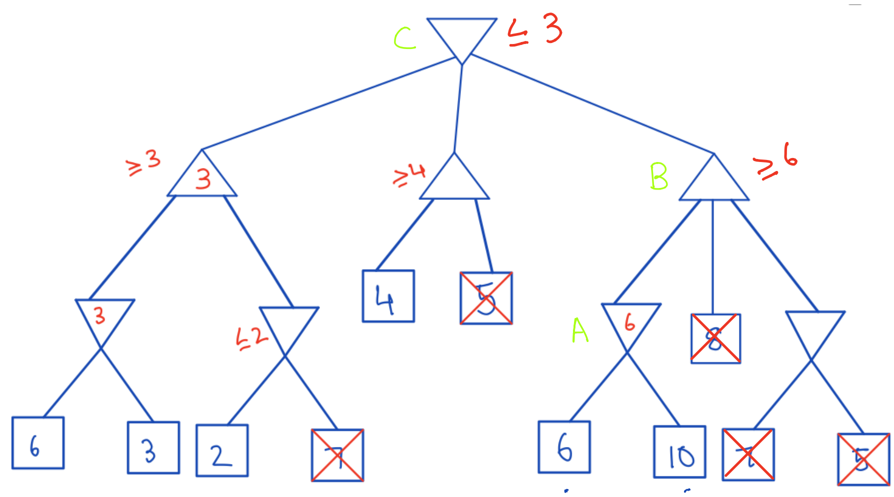
Then, (2,6) Δ (4,3) becomes (2,6). Hence the utilities at node C are (2,6).

9.9) (2pts) Write down the utilities at node A as a tuple ($u_{\text{PacAdv,A}}, u_{\text{PacMax,A}}$).

$\boxed{(2,6)}$. Since the hexagon maximises the utility for PacAdv, (1,4) ⬡ (2,6) becomes (2,6).

Hence, the utilities at node A are (2,6).

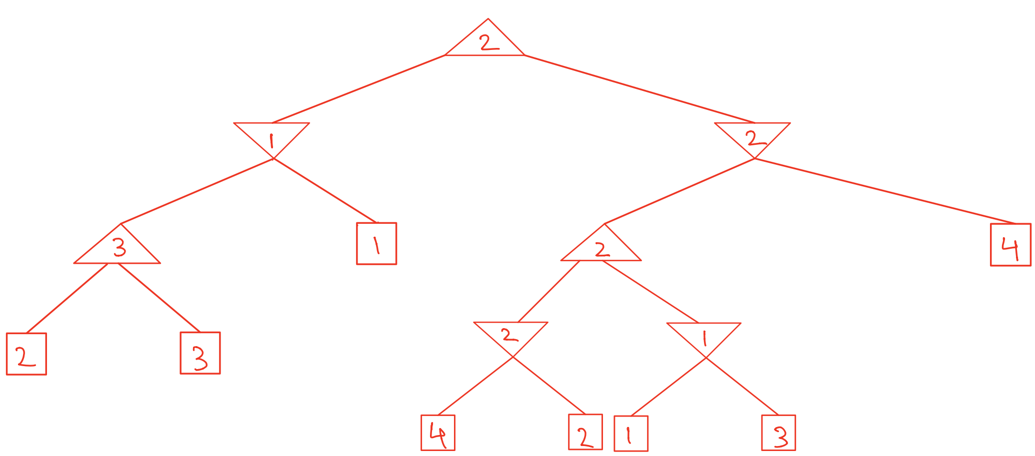
(9.4) My answer was previously 3, but I failed to account for more nodes that could be pruned. Consider the graph below from my initial submission:



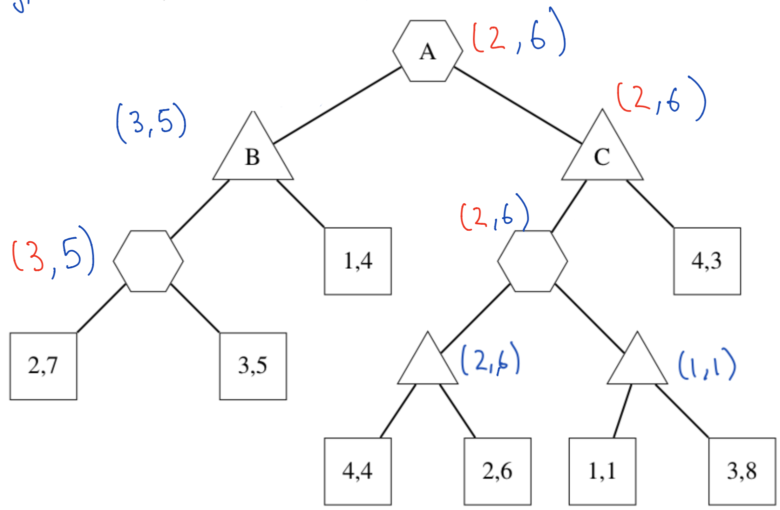
What I didn't consider was that after leaf nodes 6 and 10 were visited, we know that Node A = 6. Hence, Node B ≥ 6. This tells us that Node C will NEVER choose Node B because it already has a lower number (3). As a result, everything to the right of leaf node 6 also gets pruned. In total, leaf nodes 7, 5, 8, 7, 5 can be pruned. So $\boxed{\text{5 nodes can be pruned.}}$

(9.7) I misinterpreted how Pacmax and Pacadv choose their utilities.

From PacAdv's perspective, the game is a Minimax: (these are the values for the first element of the tuple in hexagons)



Similarly, Pacmax maximizes the 2nd number.



Hence, the $\boxed{\text{value of node B is } (3,5)}$.