GATE

CRASH COURSE

DS & AI

Artificial Intelligence

Lecture 04

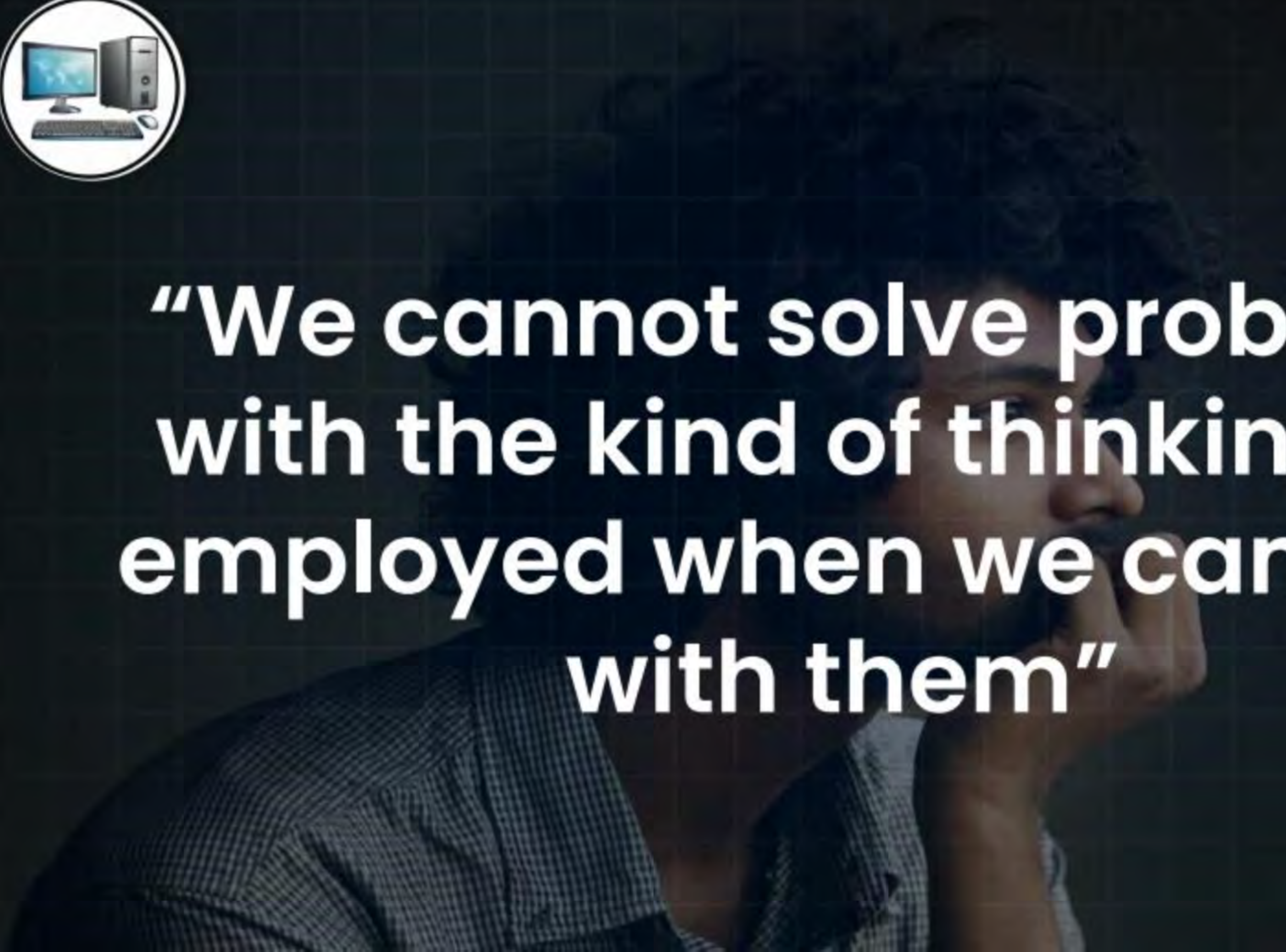
Adversarial Search

By - Siddharth Sabharwal Sir

Topics to be Covered

1. Adversarial Search

    ↳ α-β pruning.



“We cannot solve problems with the kind of thinking we employed when we came up with them”

Albert Einstein

Topic : Artificial Intelligence

Adversarial Search

- Adversarial Search: A search in scenarios where multiple agents (players) have conflicting objectives, and each agent's success is often measured against the other’s failure. Unlike traditional search problems (like pathfinding), adversarial search considers the presence of an opponent actively working against the agent's goal.

Topic : Artificial Intelligence

Minimax Algorithm:

- The Minimax algorithm is a recursive, decision-making algorithm used in two-player games, particularly zero-sum games where one player's gain is another player's loss. It systematically explores the game tree to determine the optimal move for a player, assuming that the opponent also plays optimally.
- It is a specialized Search Algo that returns optimal Sequence move for a player in a Zero-Sum Game.
- Recursive/ Back tracking algo which is used in decision making and Game theory and uses recursion to Search through Game Tree.
- Algo Computes minimax decision for current state.
- We have 2 players : max and min player. Max player select the maximum value and min player select the minimum value
- Depth-First Search Algo is used for exploration of Complete Game Tree. Used in Chess, Tic Tac Toe and other 2 player Games

Topic : Artificial Intelligence

Minimax Algorithm:

- ✓ Initial values for Max and Min player: + infinity and - Infinity This is the backtracking algorithm
- Complete and $\boxed{\text{optimal solution}}$
- Time and Space complexity
- In Minimax algorithm we use DFS

Topic : Artificial Intelligence

Minimax Algorithm:

- Conceptual Overview Two Players:
  - Often referred to as the "Maximizer" and the "Minimizer.
  - Maximizer: Tries to maximize their score.
  - Minimizer: Tries to minimize the Maximizer's score (which is equivalent to maximizing their own score if you flip the sign).
- Game Tree: A tree structure representing all possible moves (edges) from a given game state (nodes).
- Root Node: Represents the current state of the game.
- Leaf Nodes: Represent terminal states where the game ends (e.g., win, loss, or draw).

Minimax Algorithm:

- Minimax Principle
- The algorithm assumes that both players play optimally.
- It simulates all possible moves from the current state down to terminal states, then back-propagates the values up the tree to make the best decision at the root.

Minimax Algorithm:

- Minimax Algorithm Steps
- Generate the Game Tree: Start from the root node (current game state) and recursively expand all possible moves down to a certain depth or until terminal nodes (end of the game) are reached.
- Evaluate Terminal Nodes:
    - At the terminal nodes (leaf nodes), assign values based on the outcome:
    - Positive value (e.g., +1): A win for the Maximizer.
    - Negative value (e g., -1): A win for the Minimizer
    - Zero (0): A draw.

Topic : Artificial Intelligence

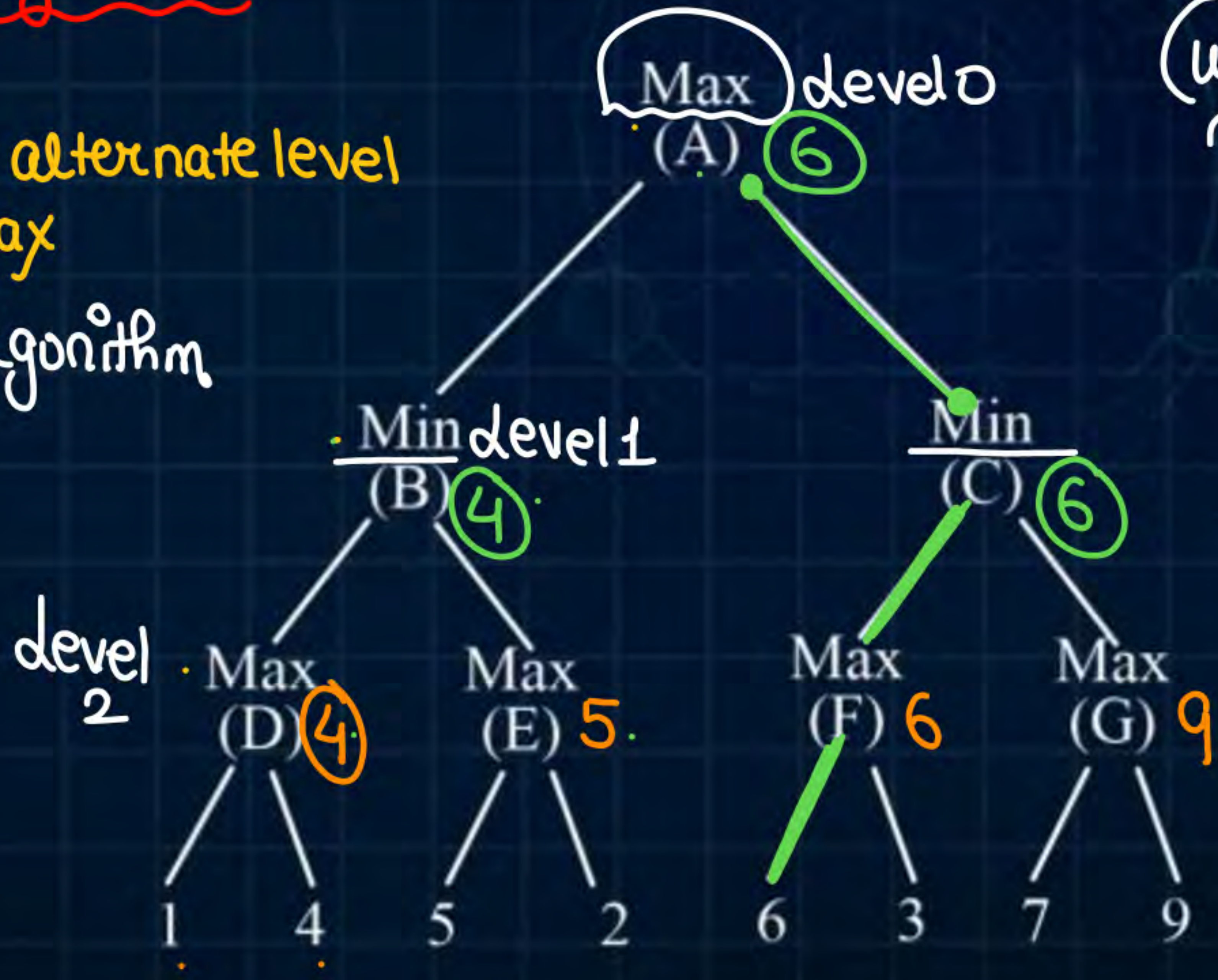
Minimax Algorithm:

- Minimax Algorithm Steps
- Backpropagate the Values:
  - Backpropagate the values from the terminal nodes up to the root.
  - At Maximizer nodes (Max nodes), select the maximum value from the child nodes.
  - At Minimizer nodes (Min nodes), select the minimum value from the child nodes.
  - Select the Best Move: At the root node (current game state), the Maximizer chooses the move corresponding to the child node with the highest value.

Topic : Artificial Intelligence

Min Max algorithm

- Here at alternate level min-max
- Max-algorithm



(we have to find best Path result maxplayer)

Max Player ⇒ 6

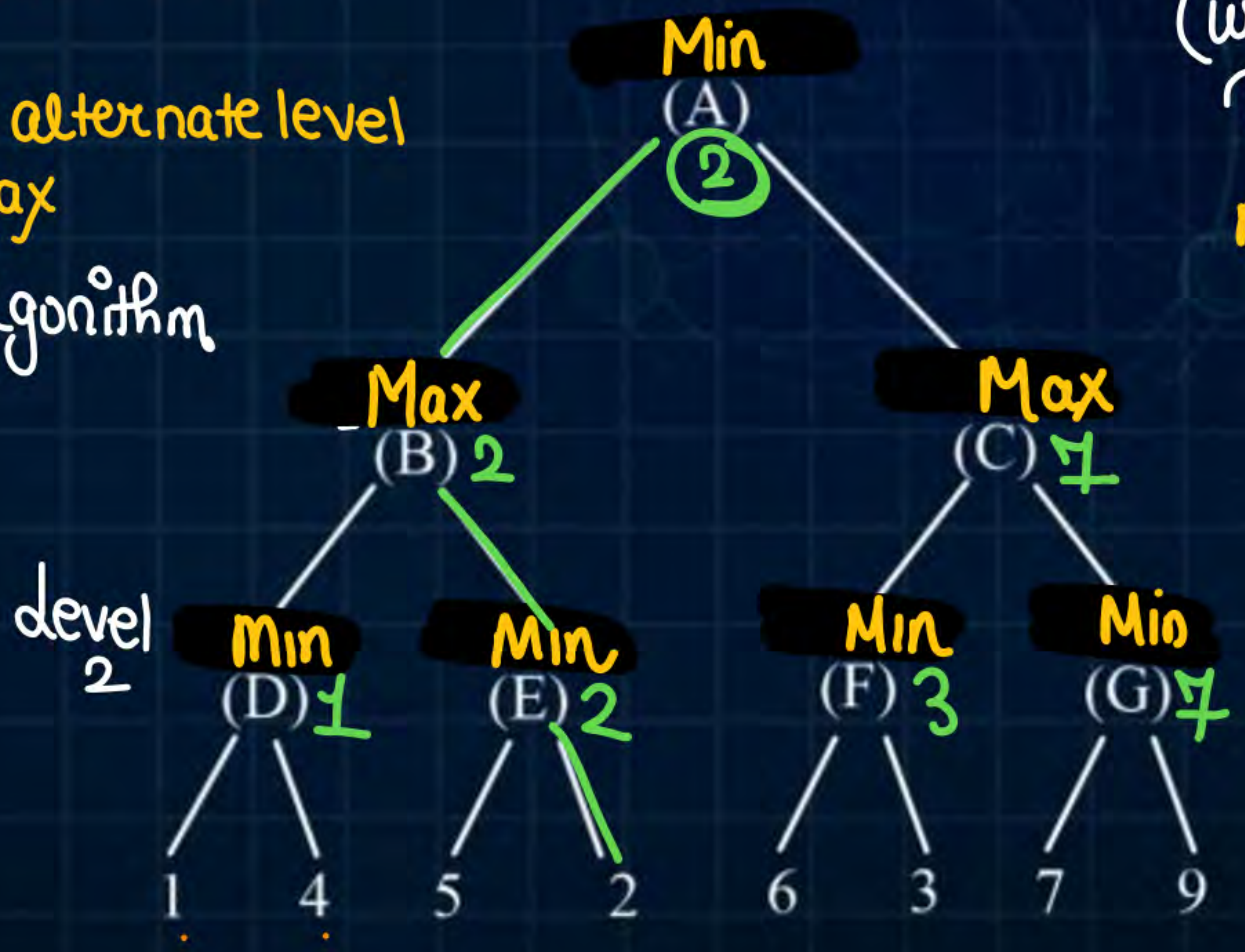
Topic : Artificial Intelligence

Min Max algorithm

- Here at alternate level min-max
- Min -algorithm

(we have to find best Path result min player)

min Player :- 2 ✓



$$\boxed{\text{Time Complexity } O(b^d)}$$

b: branching factor.

d: depth of tree

Space Complexity $\Rightarrow O(b \times d)$

Topic : Artificial Intelligence

Alpha-Beta Pruning

"efficient"

- Alpha-beta pruning is an optimization technique for the minimax algorithm, commonly used in decision-making for two-player games like chess, tic-tac-toe, and other adversarial games. The main purpose of alpha-beta pruning is to reduce the number of nodes evaluated by the minimax algorithm, thereby speeding up the decision-making process without affecting the final decision

Topic : Artificial Intelligence

Alpha-Beta Pruning

- This is advanced version of minimax algorithm
- Because minimax algorithm has to explore all the nodes thus time complexity was $O(b^d)$
- But in this new algorithm we explore less number of nodes Alpha : is for max player
- Beta : is for min player

Topic : Artificial Intelligence

Procedure for α,β pruning

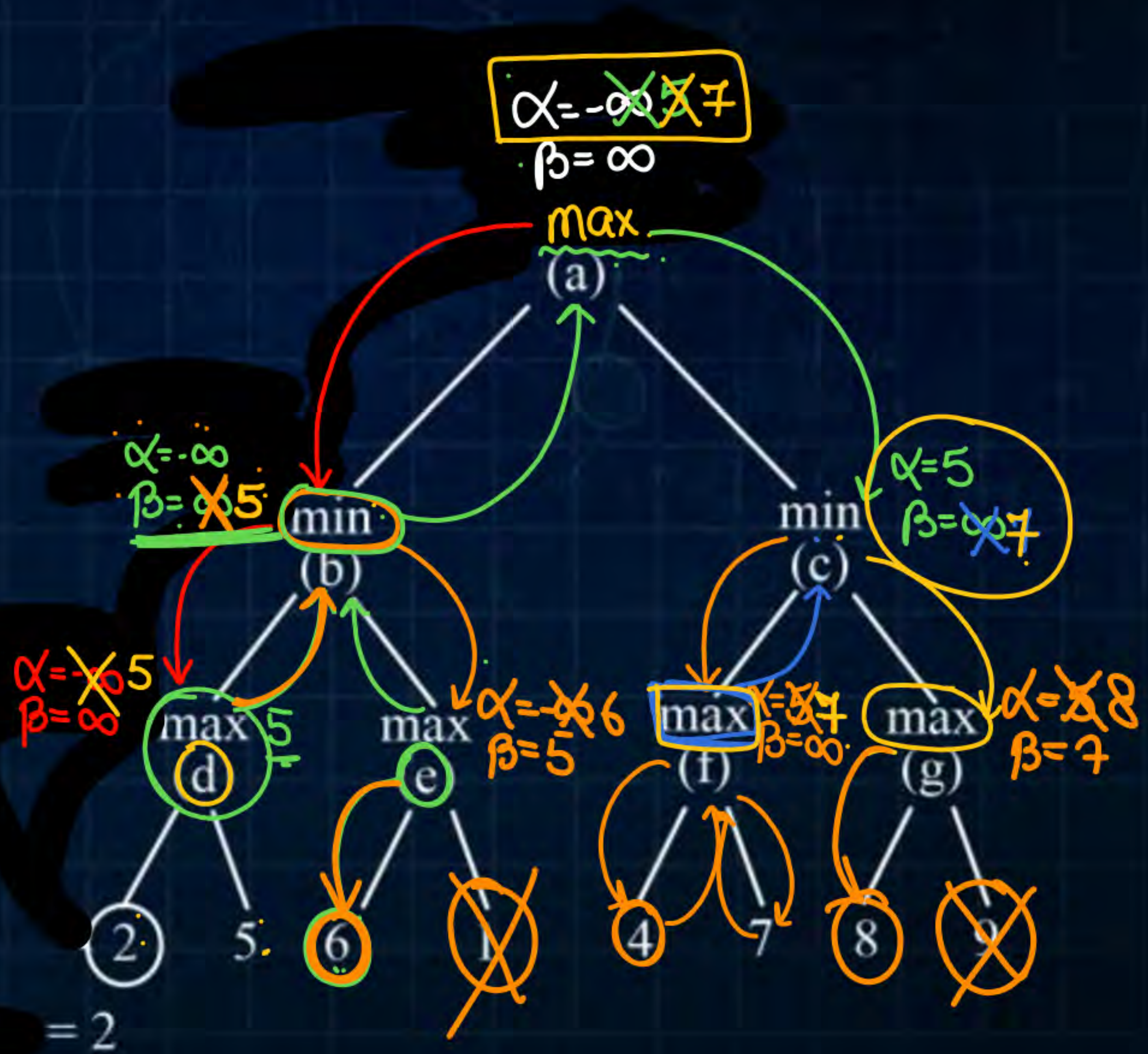
- Algorithm : DFS
- α : max player changes it new α = max (present α, child nodes min player do not change it).
- β min player change it, New β = min (present β, child nodes)
- Initial value of $\alpha = -\infty$, $\beta = \infty$.
- Pruning: if value of $\alpha \geq \beta$ at any node.

Topic : Artificial Intelligence

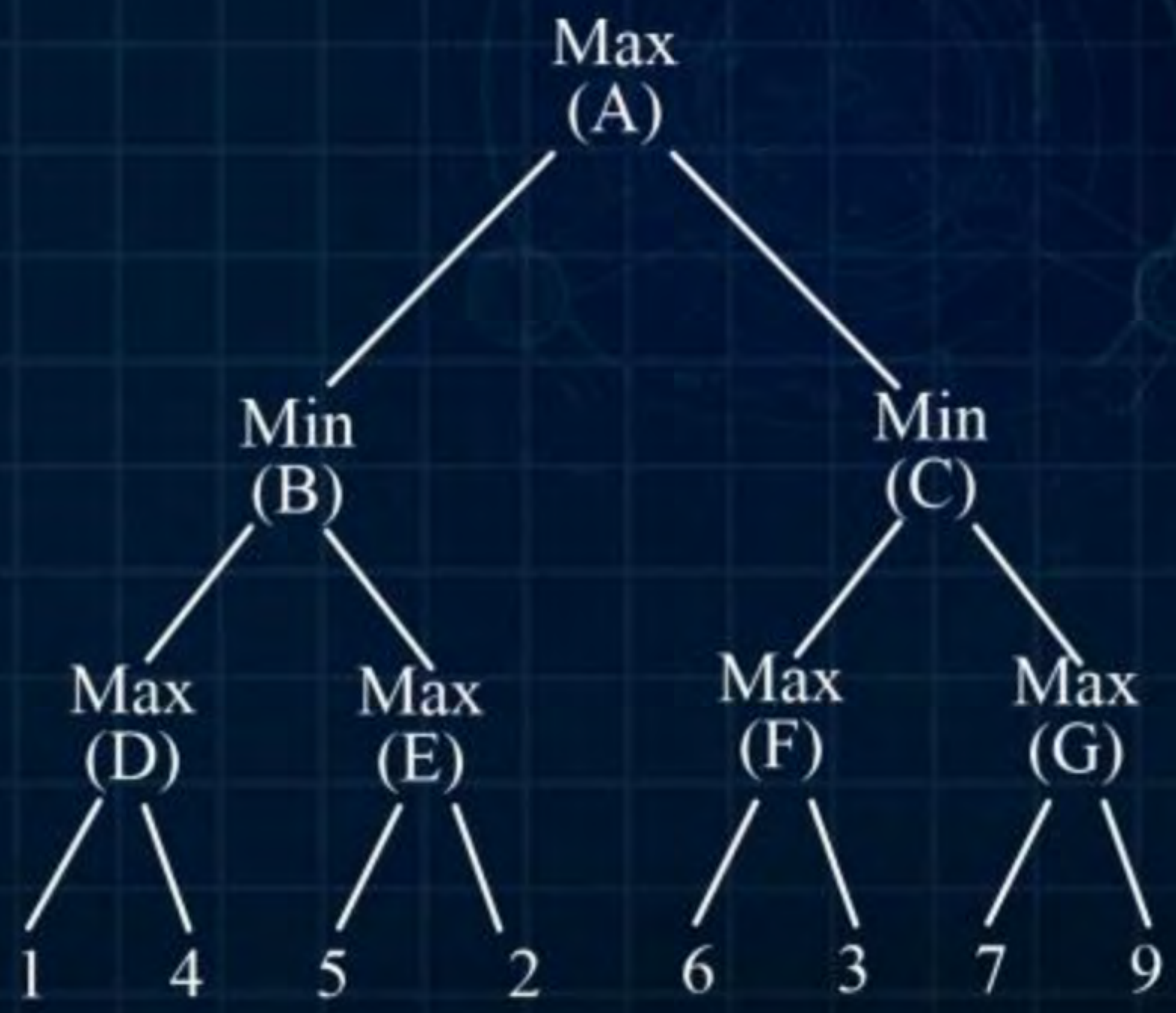
Values

α, β

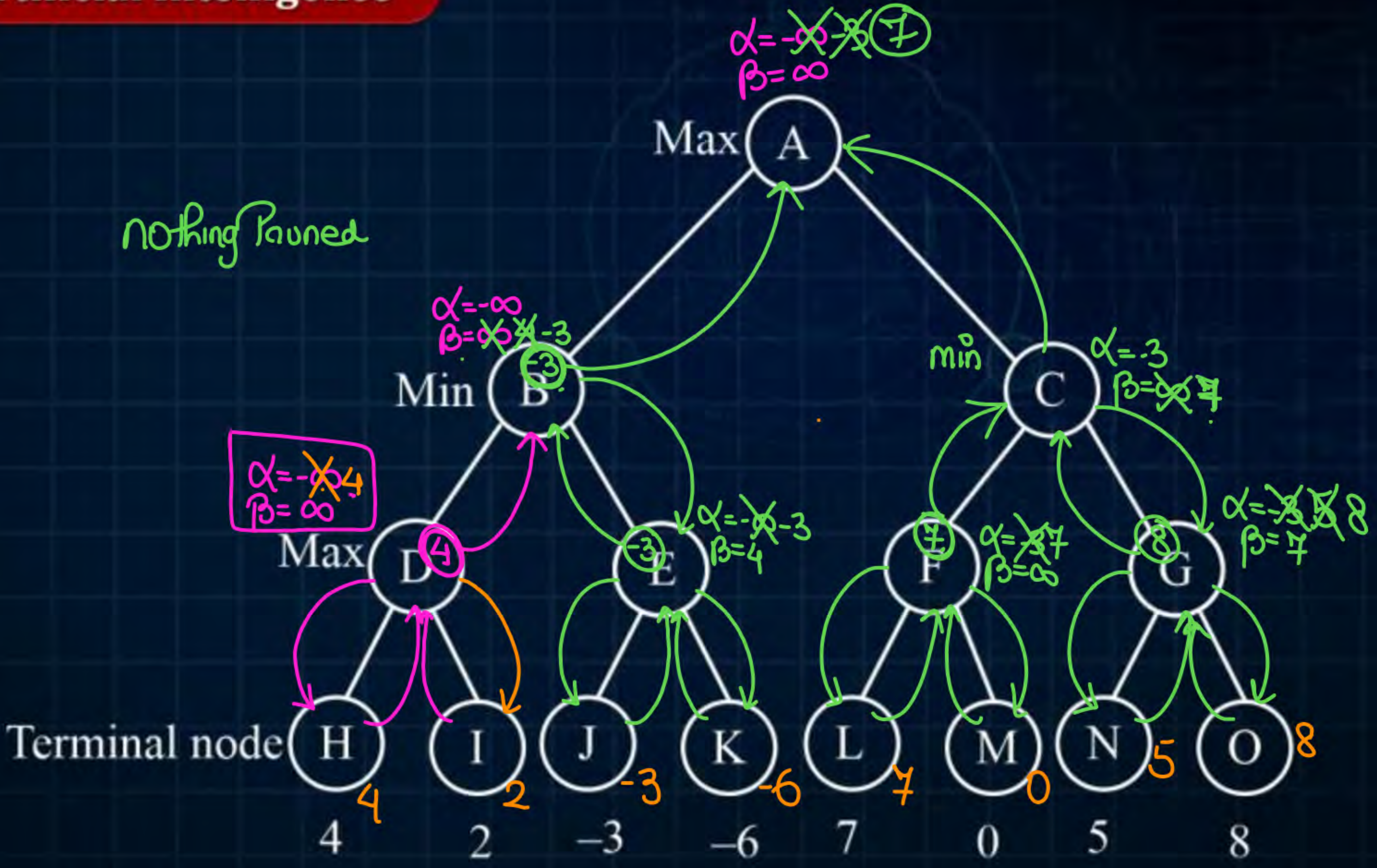
- Start @ Root node
  $\alpha = -\infty$
  $\beta = \infty$
- $\alpha \rightarrow$ updated by max node only
- $\beta \rightarrow$ " " min node only.
- DFS, @ any node $\alpha \geq \beta$ that node is pruned.



Topic : Artificial Intelligence



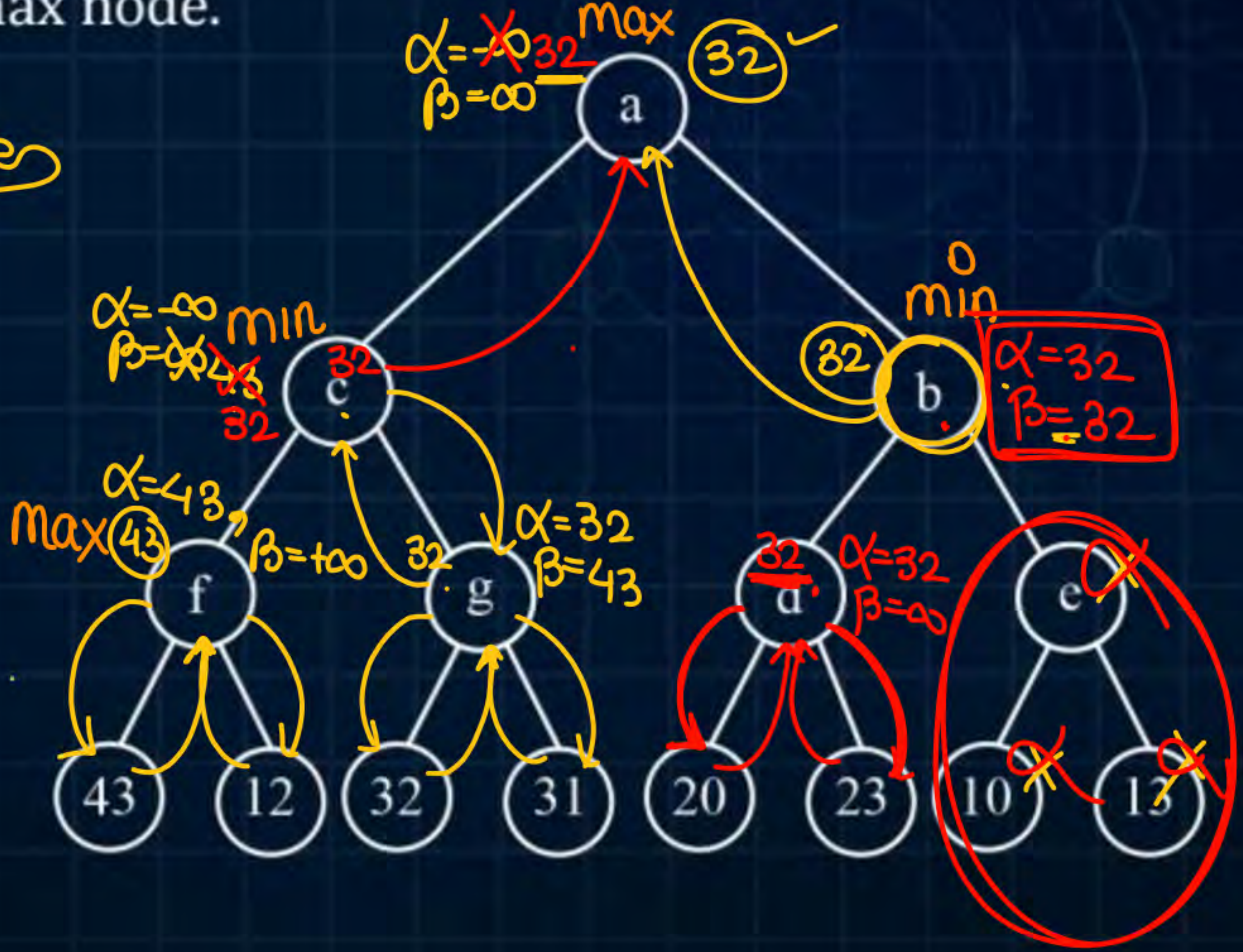
Topic : Artificial Intelligence



Topic : Artificial Intelligence

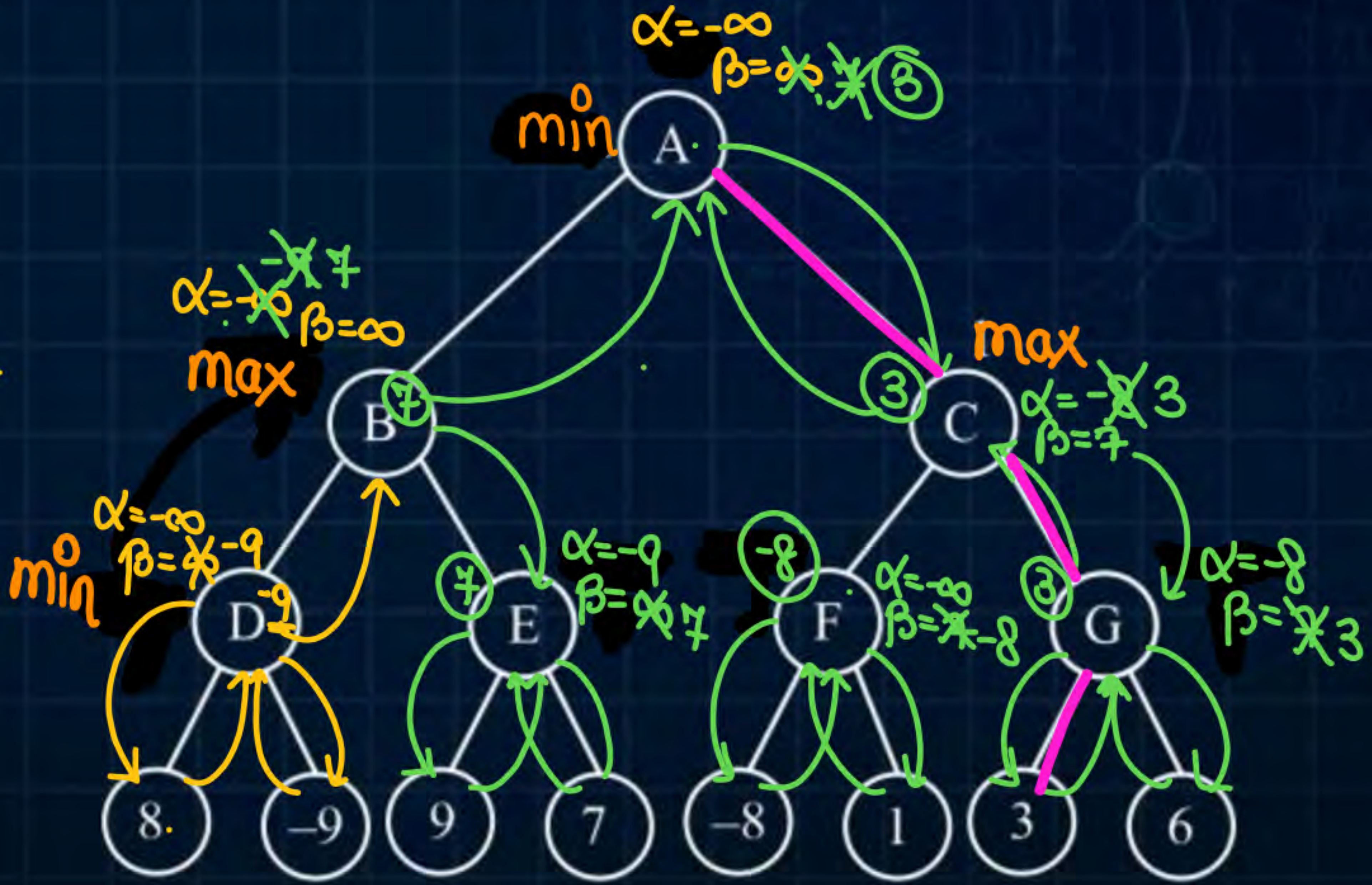
What is the value of root node after MinMax search over the given tree? The root node (labelled 'a') is a max node.

3 node Prune



Topic : Artificial Intelligence

Player 1 (max node) chooses the left action for which of the following values (s) of x. Assume optimal play and Player 1's turn at Node A.

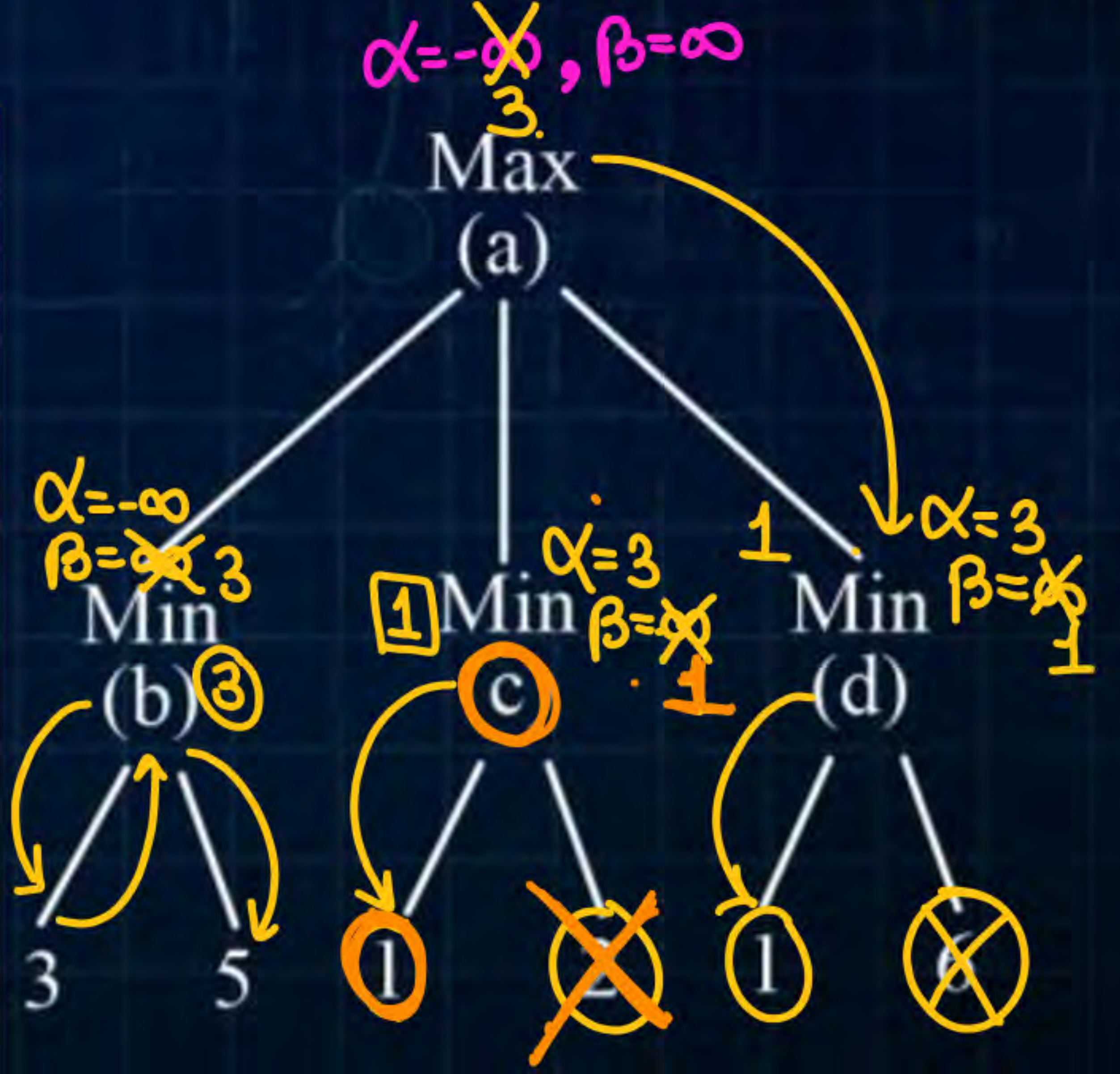


Question

#Q. Consider the following game tree. The value below each node is the output of the utility function. The subtrees rooted at which of these nodes will be pruned because of alpha-beta pruning?

2 node are prune,

Subtree under C, D are Pruned.



$\alpha = -\cancel{\infty}$ 3, $\beta = \infty$
Max
(a)
$\alpha = -\infty$
$\beta = \cancel{\infty}$ 3
Min
(b) 3
3
5
1 Min
(c)
$\alpha = 3$
$\beta = \cancel{\infty}$ 1
1
2
1
Min
(d)
$\alpha = 3$
$\beta = \cancel{\infty}$ 1
1
6

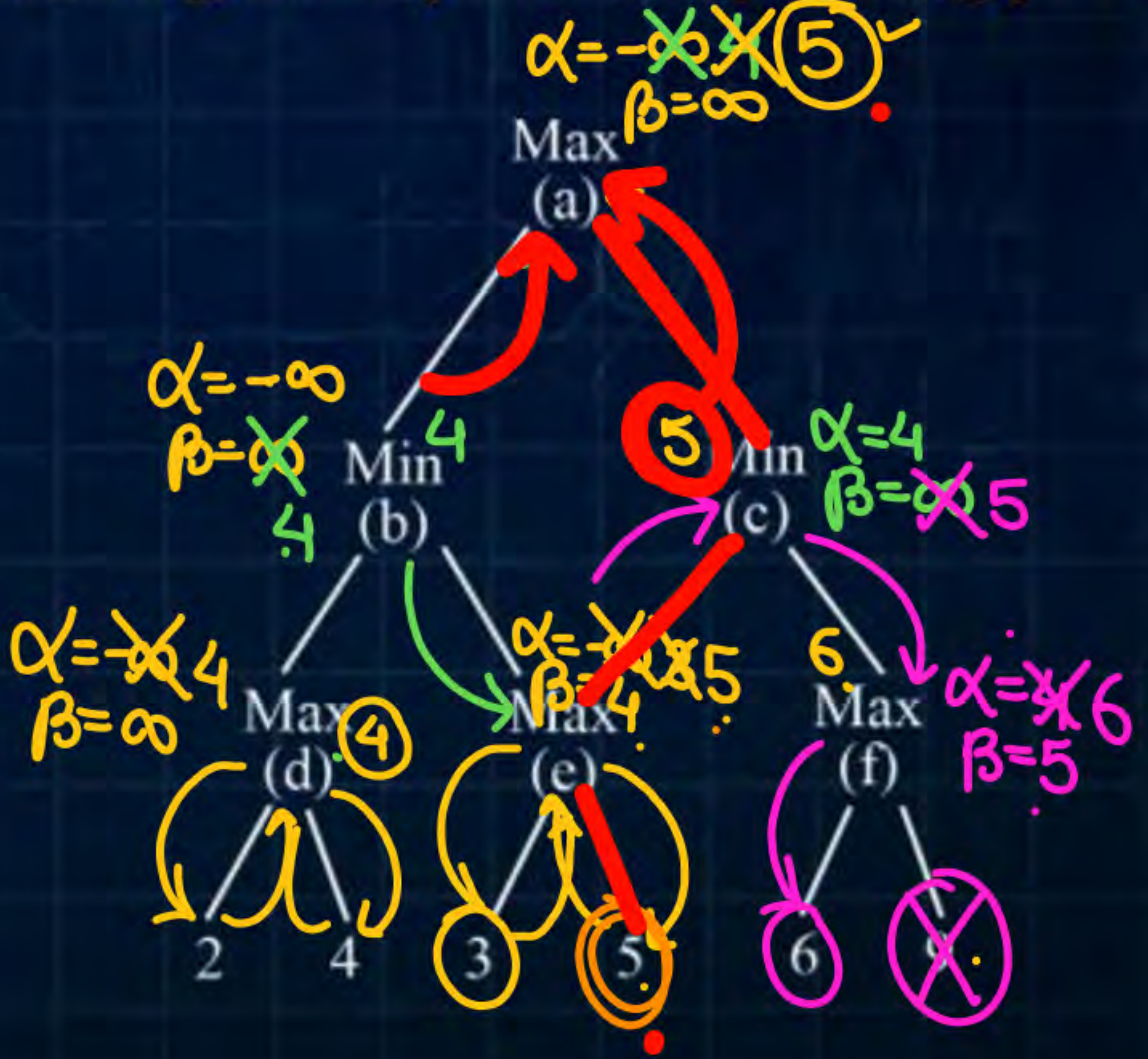
Topic : Artificial Intelligence

Consider (his game tree. The value below each node is the output of the utility function. Which nodes will be pruned during the alpha-beta pruning process?

Question

#Q. Consider this game tree. The value below each node is the output of the utility function. Which nodes will be pruned during the alpha-beta pruning process?

- 1 node Pruned under F,
- Ans = 5.



Question

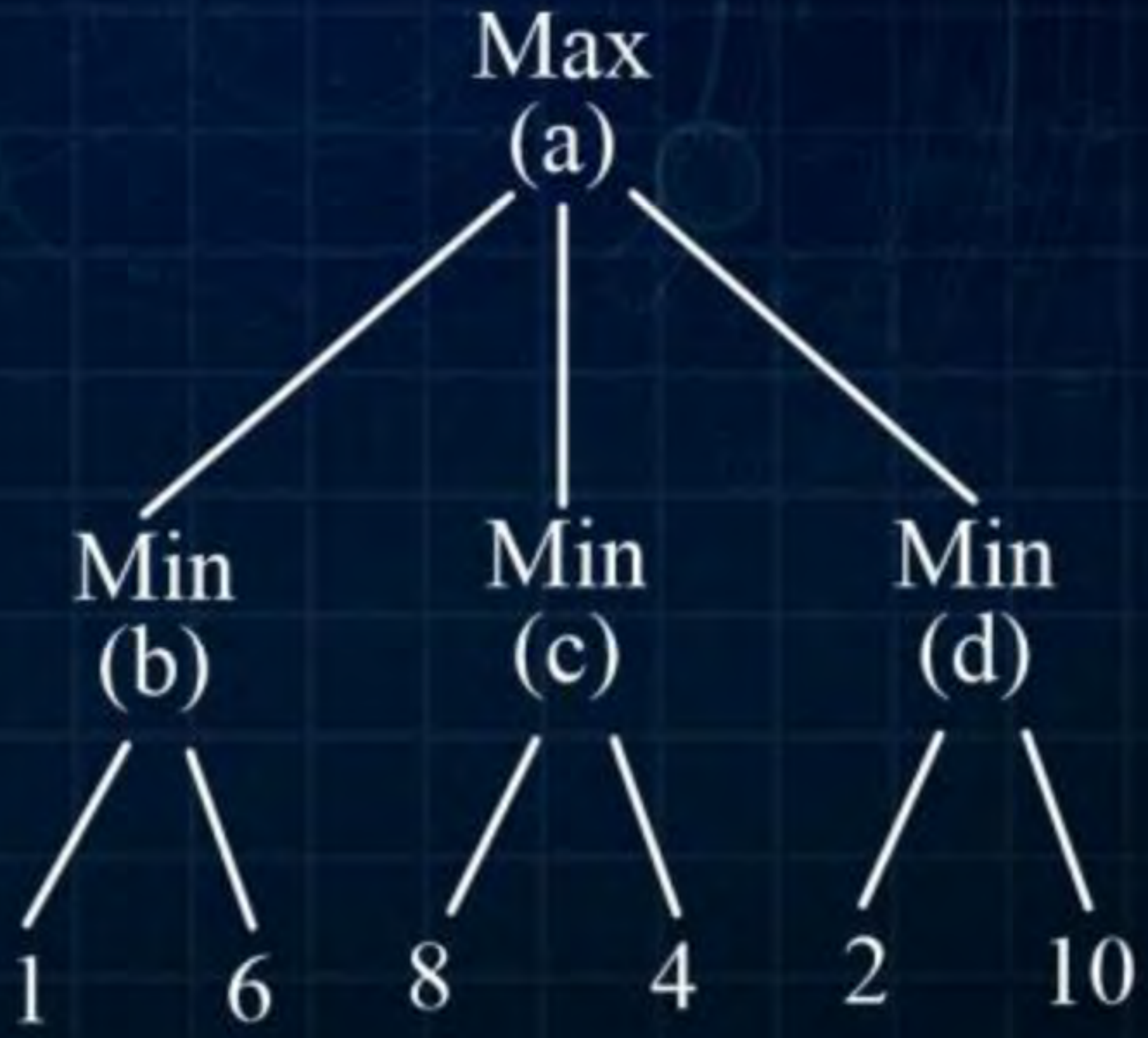
#Q. For the game tree below, which of the subtrees will be pruned due to alpha-beta pruning?

A Nodes under (b)

B Nodes under (c) and (d)

C Nodes under (d) only

D No nodes are pruned



Topic : Artificial Intelligence

Alpha-Beta Pruning - The complete procedure

- The procedure for solving any general alpha-beta pruning problem can be broken down into a set of steps that are followed to systematically explore the game tree while minimizing the number of evaluations. Below is the step-by-step procedure for solving any alpha-beta pruning problem:

1. Initialization:
   - Start at the root of the game tree.
   - Set initial values for Alpha to $-\infty$ (lowest possible) and Beta to ∞ (highest possible).
   - Alpha: Keeps track of the maximum lower bound that the maximizing player (MAX) is assured of.
   - Beta: Keeps track of the minimum upper bound that the minimizing player (MIN) is assured of.

Topic : Artificial Intelligence

Alpha-Beta Pruning - The complete procedure

2. Traverse the Tree Depth-First: Use a depth-first search (DFS) traversal to explore the tree. Start from the root node and proceed down one branch, only moving to another branch if necessary.
3. Update Alpha and Beta:
   - a At each node, based on whether it is a MAX or MIN node:
   - MAX node:
   - a Maximizing player (goal is to maximize the utility value).
   - Start with the current value of Alpha.
   - At each child update Alpha with the maximum value of its children.
   - Update: Alpha = max(Alpha, child value).
   - MIN node:
   - a Minimizing player (goal is to minimize the utility value).
   - Start with the current value of Beta.
   - a At each child update Beta with the minimum value of its children.
   - Update: Beta = min(Beta, child value).

Alpha-Beta Pruning - The complete procedure

4. Prune When Alpha ≥ Beta:
    - While traversing the tree, check if Alpha ≥ Beta at any point: Alpha ≥ Beta means that the current branch will not affect the outcome because the maximizing player can already guarantee a better result or the minimizing player can avoid a worse outcome.
    - When this condition holds, you can prune the remaining branches below that node, as further exploration is unnecessary.
5. Backtrack and Propagate Values:
    - Once a leaf node is evaluated, propagate the computed value up to its parent node:
    - At MAX nodes, propagate the maximum value from the children.
    - At MIN nodes, propagate the minimum value from the children.

Alpha-Beta Pruning - The complete procedure

6. Repeat Until All Nodes Are Visited or Pruned:
   - Continue traversing the tree until: You have visited all necessary nodes, or Pruning has eliminated further exploration.

Topic : Artificial Intelligence

Alpha-Beta Pruning - The complete procedure

- The procedure for solving any general alpha-beta pruning problem can be broken down into a set of steps that are followed to systematically explore the game tree while minimizing the number of evaluations. Below is the step-by- step procedure for solving any alpha-beta pruning problem:

1. Initialization:
    - Start at the root of the game tree.
    - Set initial values for Alpha to -∞ (lowest possible) and Beta to ∞ (highest possible).
    - Alpha: Keeps track of the maximum lower bound that the maximizing player (MAX) is assured of.
    - Beta: Keeps track of the minimum upper bound that the minimizing player (MIN) is assured of.

Topic : Artificial Intelligence

Alpha-Beta Pruning - The complete procedure

2. Traverse the Tree Depth-First: Use a depth-first search (DFS) traversal to explore the tree. Start from the root node and proceed down one branch, only moving to another branch if necessary.
3. Update Alpha and Beta:
   - a At each node, based on whether it is a MAX or MIN node:
   - MAX node:
   - a Maximizing player (goal is to maximize the utility value).
   - Start with the current value of Alpha.
   - At each child update Alpha with the maximum value of its children.
   - Update: Alpha = max(Alpha, child value).
   - MIN node:
   - a Minimizing player (goal is to minimize the utility value).
   - Start with the current value of Beta.
   - a At each child update Beta with the minimum value of its children.
   - Update: Beta = min(Beta, child value).

Alpha-Beta Pruning - The complete procedure

4. Prune When Alpha ≥ Beta:
   - While traversing the tree, check if Alpha ≥ Beta at any point: Alpha ≥ Beta means that the current branch will not affect the outcome because the maximizing player can already guarantee a better result or the minimizing player can avoid a worse outcome.
   - When this condition holds, you can prune the remaining branches below that node, as further exploration is unnecessary.
5. Backtrack and Propagate Values:
   - Once a leaf node is evaluated, propagate the computed value up to its parent node:
   - At MAX nodes, propagate the maximum value from the children.
   - At MIN nodes, propagate the minimum value from the children.

Topic : Artificial Intelligence

Alpha-Beta Pruning - The complete procedure

6. Repeat Until All Nodes Are Visited or Pruned:
    - Continue traversing the tree until: You have visited all necessary nodes, or Pruning has eliminated further exploration.

Worst Case T.C $\Rightarrow O(b^d)$ DFS

" " S.C $\Rightarrow O(b \times d)$

Alpha-Beta Pruning - The complete procedure

- Space Complexity:
- Like Minimax, the space complexity in Alpha-Beta pruning is determined by the depth of the recursion stack.
- Space Complexity: O( ),

Question

#Q. Which of the following statements is true for the Minimax algorithm in adversarial search?

(A) It only works for one-player games.

(B) It minimizes the outcome of a game for both players.

[✓ (C) $\boxed{\text{It assumes that both players play optimally}}$ ✓]

(D) It evaluates only a subset of the game tree.

Question

#Q. In a Minimax algorithm, how is the value of a node computed in a game tree?

A The sum of values of its children nodes.

B The average of values of its children nodes

[✓ C The maximum or minimum value from its children, depending on whether it's a Maximizer or Minimizer node.]

D The product of values of its children nodes.

Question

#Q. What is the time complexity of the Minimax algorithm in a game tree with branching factor b and depth d?

A $O(b \, . \, d)$

B $O(b + d)$

[✓ C $\boxed{O(b^d)}$]

D $O(b^{d/2})$

Question

#Q. In Alpha-Beta pruning, pruning occurs when:

$\boxed{\alpha \geq \beta \text{ for both max/min both}}$ ✓

A. A node 's value is greater than or equal to α for a Minimizer.

B. A node 's value is less than or equal to β for a Maximizer.

C. A node's value exceeds β for a Maximmizer or is lower than α for a Minimizer

D. A node's value is between α and β.

Question

#Q. Which of the following are benefits of Alpha-Beta Pruning over the Minimax algorithm?

[✓ A] It reduces the time complexity in the best-case scenario.

[✓ B] It guarantees optimal solutions even if some branches are pruned.

C ~~It guarantees faster execution than Minimax in all cases.~~

D ~~It reduces space complexity compared to Minimax.~~

Question

#Q. Which of the following statements are true for the Alpha-Beta pruning algorithm?

~~A~~ It evaluates every node in the game tree.

[✓ B It requires the tree to be searched depth–first] ✓

[✓ C It reduces the time complexity of Minimax in the best case to $O(b^{d/2})$.] ✓✓

[✓ D Pruning depends on the order of node exploration.]

Thank

THANK

Keep Hustling!