Artificial Intelligence

Dr. Qaiser Abbas
Department of Computer Science & IT,
University of Sargodha, Sargodha, 40100, Pakistan
qaiser.abbas@uos.edu.pk

ADVERSARIAL SEARCH

- *In which we examine the problems that arise when we try to plan ahead in a world where other agents are planning against us.*

- In this lecture we will cover **competitive** environments, in which the agents' goals are in conflict, giving rise to **adversarial search** problems—often known as **games**.

- We will cover the followings:
  - GAMES
  - OPTIMAL DECISIONS IN GAMES
  - MiniMax Algorithm
  - ALPHA–BETA PRUNING

Games

- A game can be defined as a search problem with the following elements:
  - S_0: **Initial state** which specifies how the game is set up at the start.
  - **PLAYER(s)**: Defines which player has the move in a state s.
  - **ACTIONS(s)**: Returns set of legal moves in a state s.

Games

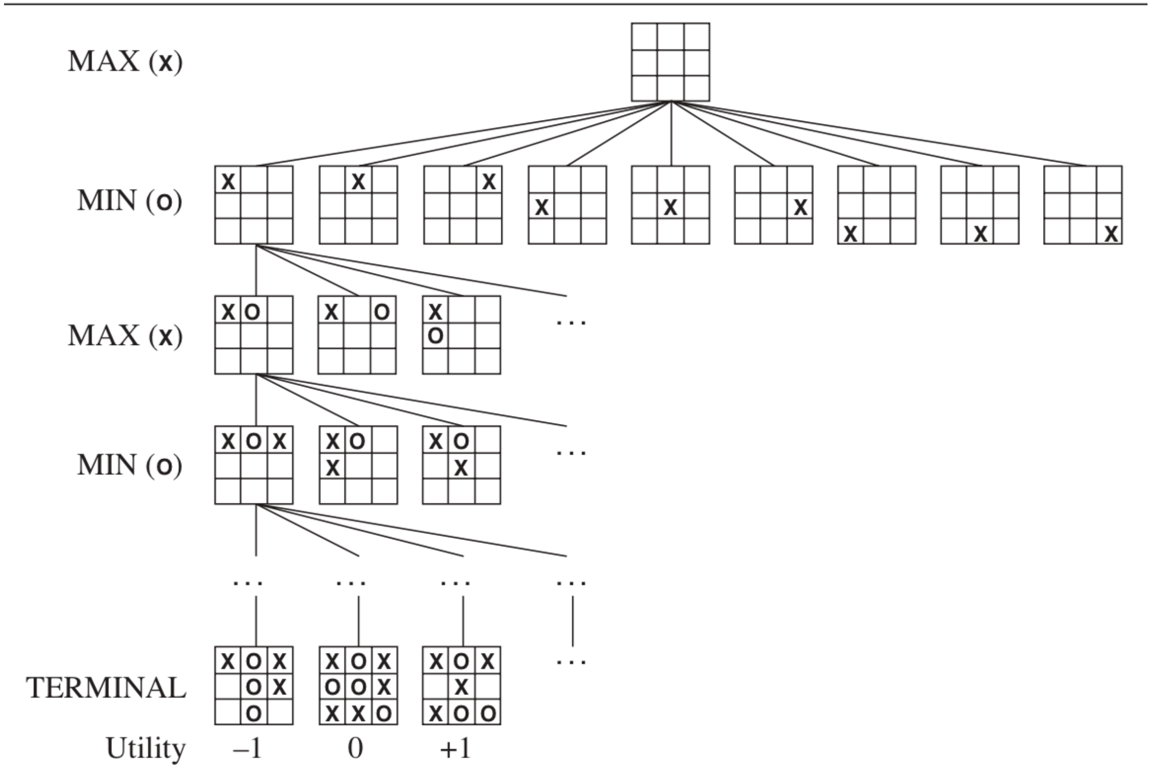
- **RESULT(s,a)**: The **transition model** defines the result of a move from state s with action a.

- **TERMINAL-TEST(s)**: A **terminal test** at a state s, which is true when the game is over and false otherwise.

- **U T I L I T Y (s, p)**: A **utility function** also called an objective function or payoff function, defines the final numeric value for a game that ends in terminal state s for a player p.

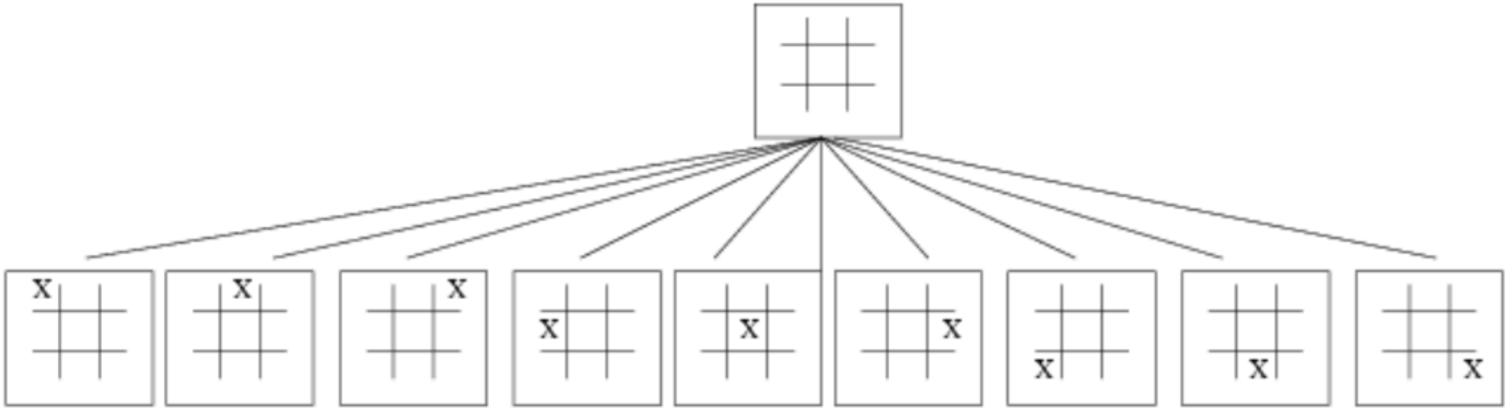
Games

- The initial state, ACTIONS function, and RESULT function define the **game tree**—a tree where the nodes are game states and the edges are moves. Figure shows part of the game tree for Tic-Tac-Toe.



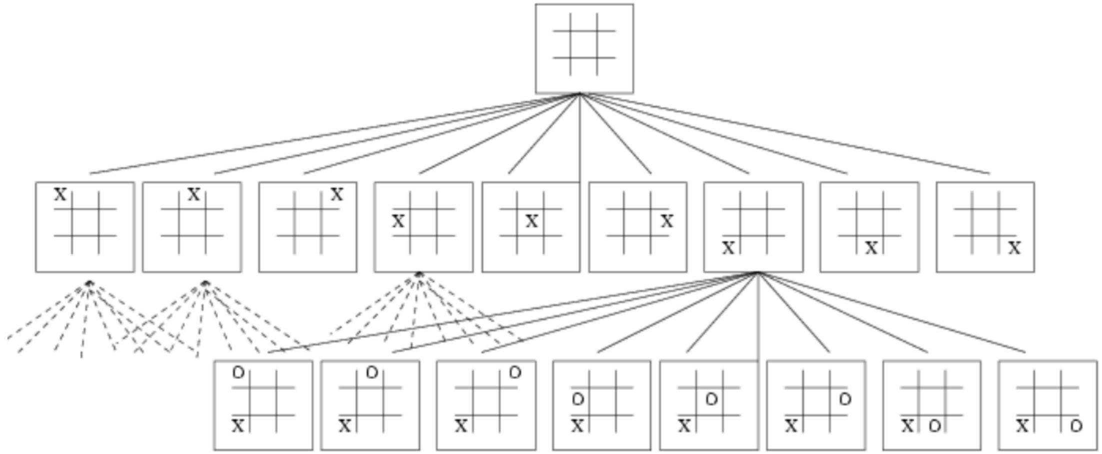
Simple Approach in Tic-Tac-Toe

- In simple algorithm, calculate all the possible moves from the current position. For a game of Noughts and Crosses the result might look like:



- Expand each of these new possible moves for the other player.
- Continue this expansion until a winning position for the player is found.

Simple Approach in Tic-Tac-Toe



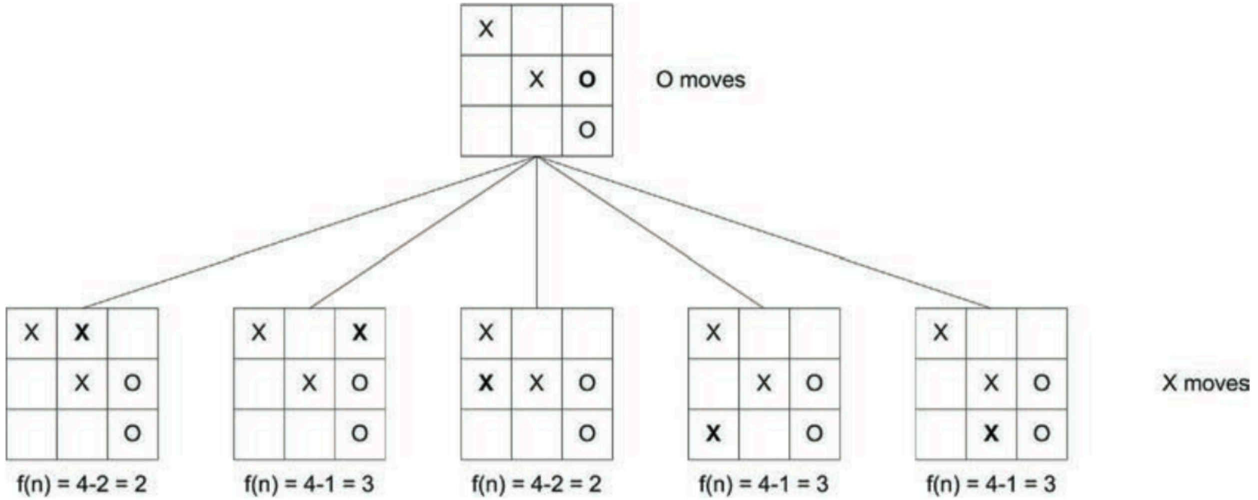
- This algorithm will work and locate a series of winning moves at the cost of enormous calculation e.g. one board at first, 9 at the next level, 9*8 next and so on. In total:

$$\sum_{0}^{n=8} \frac{9!}{(n+1)!} = 986410$$

- This is not so big that we cannot calculate it, but it is alarming since Noughts and Crosses is such a simple game.

OPTIMAL DECISIONS IN GAMES

- An optimal strategy leads to outcomes at least as good as any other strategy when one is playing an infallible opponent.
- **Two Player Games:**
  - Consider a Zero-Sum game in which gain of one player is balanced exactly by the loss of other player.
  - Static Evaluation Function *f(n)* = Complete R/C/D open positions for X – Complete R/C/D open positions for O



OPTIMAL DECISIONS IN GAMES

- Higher the result of f(n), the closer the move towards a win. Three moves result in 3 but only one move results a win for X in Figure.

- This f(n) is useful but another heuristics is necessary to pick the move with highest f(n) while protecting against a loss in the next move.

- For this purpose, a Minimax algorithm given next, in which the algorithm's opponent will be trying to minimize whatever value the algorithm is trying to maximize (hence, ”Minimax").

- Thus, the computer should make the move which leaves its opponent capable of doing the least damage.

Minimax algorithm

- Minimax uses one of the two basic strategies:
  - Entire game tree is searched to the leaf nodes
  - Tree is searched to a predefined depth and then evaluated.
- Can **pursue** the tree by making guesses as to how the opponent will play.
- **Cost function** can be used to evaluate how the opponent is likely to play.

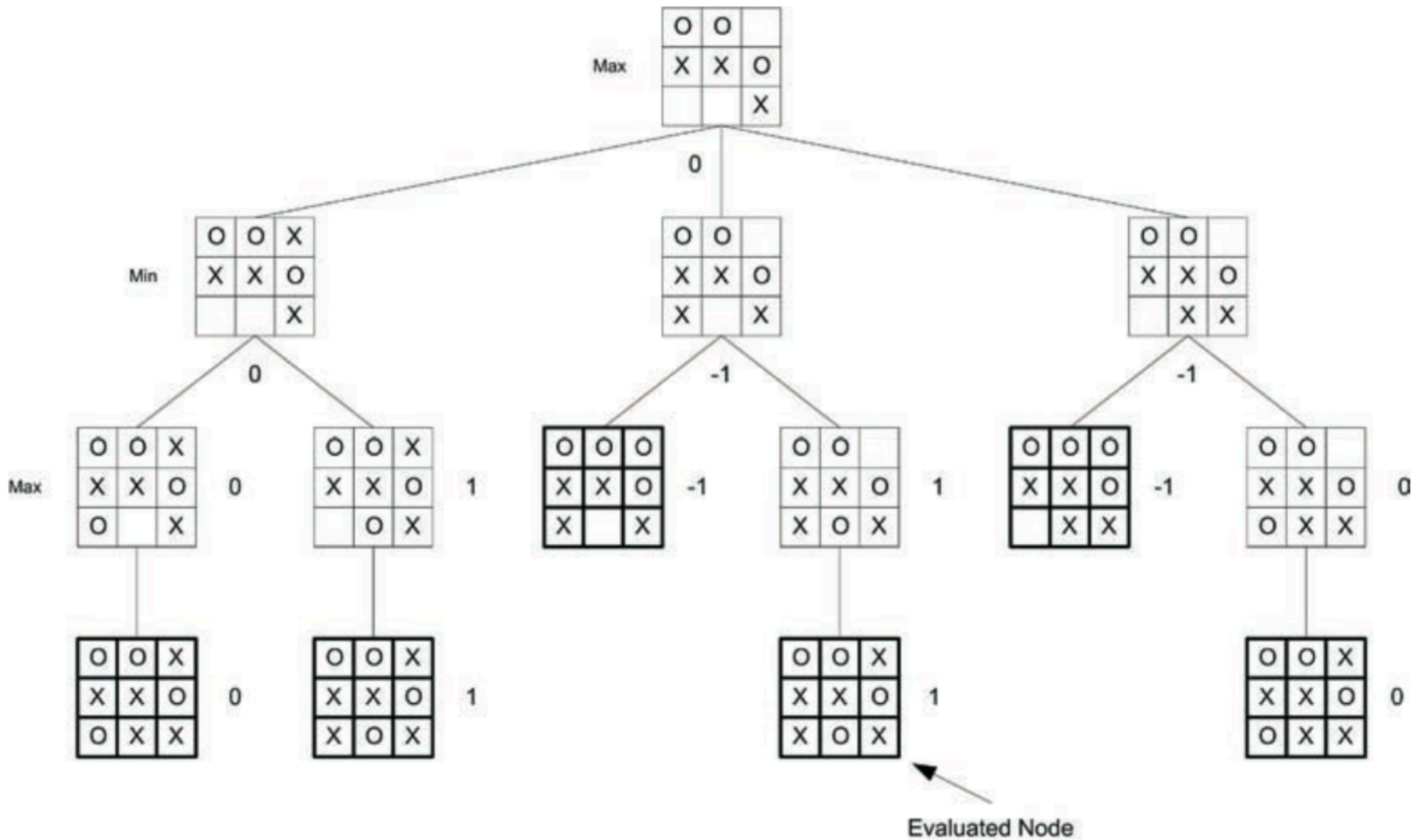
Minimax algorithm

- After evaluating some number of moves ahead we examine the total value of the cost to each player.
- The goal is to find a move which will maximize the value of our move and will minimize the value of the opponents moves.
- The algorithm used is the Minimax search procedure presented next.

The Minimax algorithm

```
function MINIMAX(N) is
begin
if N is deep enough then
        return the estimated score of this leaf
else
        Let N1, N2, .., Nm be the successors of N;
        if N is a Min node then
                return min{MINIMAX(N1), .., MINIMAX(Nm)}
        else
                return max{MINIMAX(N1), .., MINIMAX(Nm)}
end MINIMAX;
```

Partial Example Tree For Minimax Algorithm



ALPHA–BETA PRUNING

- Minimax search is exponential like DFS and couldn't be eliminated but can be effectively cut in half.

- Idea of **Pruning** to eliminate large parts of the tree can be used and the particular technique we examine is called **Alpha–Beta Pruning**.

- When applied to a standard Minimax tree, it returns the same move as Minimax would, but prunes away branches that cannot possibly influence the final decision.

ALPHA–BETA PRUNING

- Consider, the two-ply game tree from Figure given next.
- Let's go through the calculation of the optimal decision once more, this time paying careful attention to what we know at each point in the process.

ALPHA–BETA PRUNING

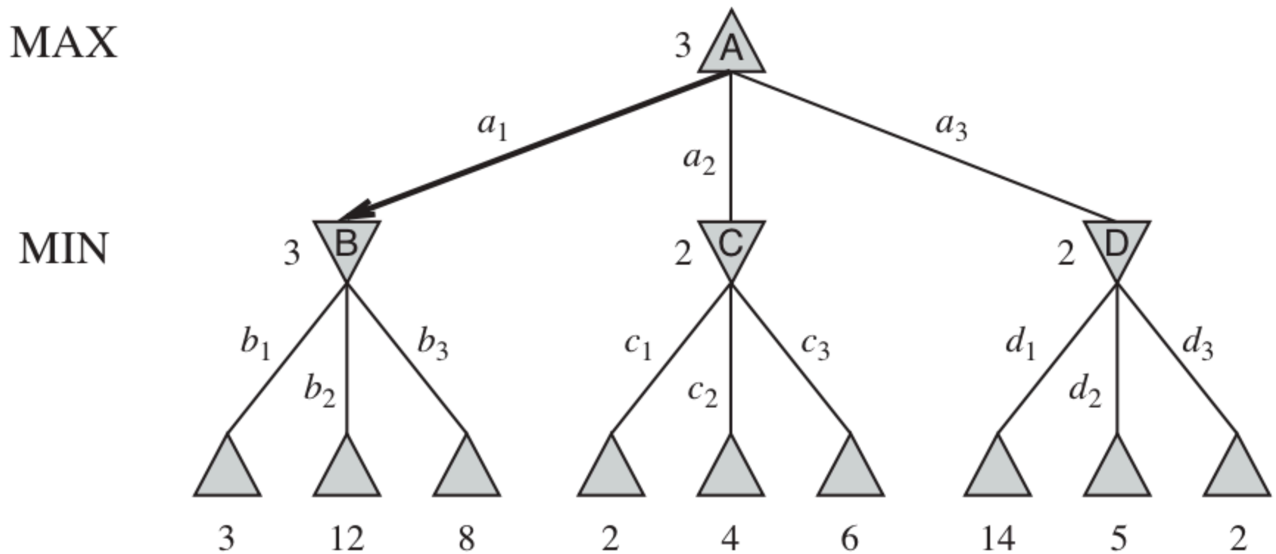


Figure 5.2 A two-ply game tree. The △ nodes are "MAX nodes," in which it is MAX's turn to move, and the ▽ nodes are "MIN nodes." The terminal nodes show the utility values for MAX; the other nodes are labeled with their minimax values. MAX's best move at the root is a_1, because it leads to the state with the highest minimax value, and MIN's best reply is b_1, because it leads to the state with the lowest minimax value.

ALPHA–BETA PRUNING

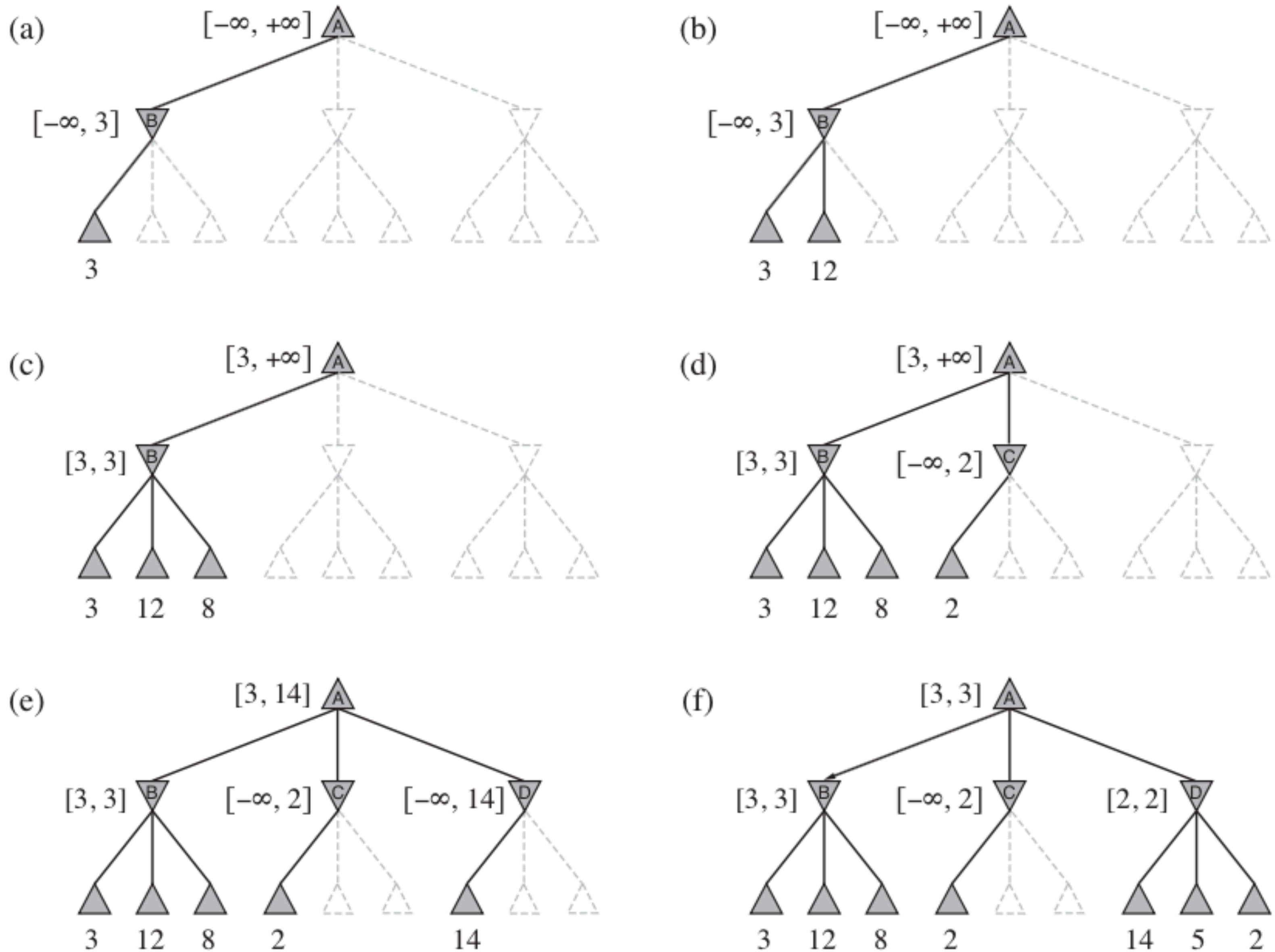
- Idea of pruning after applying it on the previous figure is described below . No need to evaluate two successors of node C.

- Suppose they have values x and y. Then the value of the root is:
  - MINIMAX(root) $= \text{Max}(\text{Min}(3,12,8),\text{Min}(2,x,y),\text{Min}(14,5,2))$
    $= \text{Max}(3,\text{Min}(2,x,y),2)$
    $= \text{Max}(3,z,2)$ where $z = \text{Min}(2,x,y) \leq 2$
    $= 3.$

- Alpha–beta pruning can be applied to trees of any depth, and it is often possible to prune entire subtrees rather than just leaves.
  - α = the value of the best (i.e., highest-value) choice we have found so far at any choice point along the path for MAX.
  - β = the value of the best (i.e., lowest-value) choice we have found so far at any choice point along the path for MIN.

- The detailed steps are explained in Figure next.

ALPHA–BETA PRUNING



ALPHA–BETA PRUNING

- Alpha–beta search updates the values of α and β as it goes along and prunes the remaining branches at a node (i.e., terminates the recursive call) as soon as the value of the current node is known to be worse than the current α or β value for MAX or MIN, respectively.

- The complete algorithm is given next. It will be good to trace its behavior when applied to the tree in Figure.

ALPHA–BETA PRUNING

function ALPHA-BETA-SEARCH($state$) **returns** an action
  $v \leftarrow$ MAX-VALUE($state, -\infty, +\infty$)
  return the $action$ in ACTIONS($state$) with value v

---

function MAX-VALUE($state, \alpha, \beta$) **returns** *a utility value*
  if TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow -\infty$
  for each a **in** ACTIONS($state$) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(RESULT(s,a), α, β))
    if $v \geq \beta$ **then return** v
    $\alpha \leftarrow$ MAX(α, v)
  return v

---

function MIN-VALUE($state, \alpha, \beta$) **returns** *a utility value*
  if TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow +\infty$
  for each a **in** ACTIONS($state$) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(RESULT(s,a) , α, β))
    if $v \leq \alpha$ **then return** v
    $\beta \leftarrow$ MIN(β, v)
  return v

Lab Project #7

- **Implement** the Tic-Tac-Toe game using any adversarial searching algorithm
  - Visit the following link: https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-3-tic-tac-toe-ai-finding-optimal-move/
  - Perform the following operations over it.
  - Download it.
  - Configure and execute it.
  - Submit the final report.