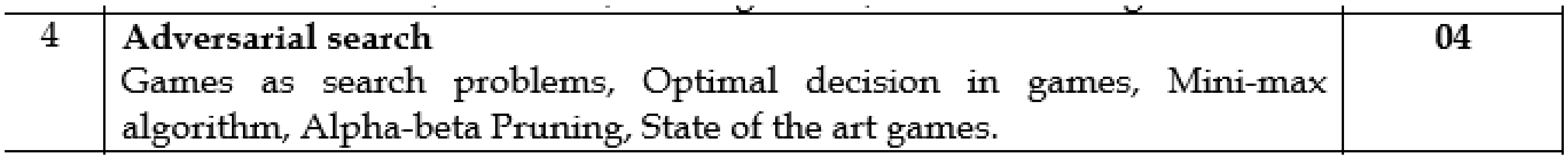
UNIT4
ADVERSARIAL SEARCH

Ami Munshi

Syllabus and reference



4	**Adversarial search** Games as search problems, Optimal decision in games, Mini-max algorithm, Alpha-beta Pruning, State of the art games.	**04**

Reference:

Stuart Russel and Peter Norvig, *Artificial Intelligence: A Modern Approach*, 4th edition, Pearson Education, 2021.
Elaine Rich, Kavin Knight, Shivshankar Nair, Artificial intelligence, 3rd edition, Tata McGraw Hill 2019

Introduction

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

- Adversarial??
- Involving or characterized by conflict or opposition
  - Oxford dictionary
- Involving two people or two sides who oppose each other
  - Merriam Webster dictionary

Stuart Russel and Peter Norvig, *Artificial Intelligence: A Modern Approach*, 4th edition, Pearson Education, 2021.

Adversarial search

- Dealing with problems where we try to plan ahead in the world where other agents(players) are planning against us
- Example- Board games
- Multiagent environment
  - Each agents need to consider action of other agents
- Competitive environment
- Agents goal will be in conflict, giving rise to adversarial problems called GAMES
- Mathematical **game theory**, a branch of economics
  - Views any multi-agent environment as a game
  - The impact of each agent on the others is “significant,” regardless of whether the agents are cooperative or competitive.
- Environment is
  - **deterministic, fully observable environment in which two agents act alternately**
  - Utility values at the end of the game are always equal and opposite
  - For example
    - player wins a game of chess, the other player necessarily loses
- It is this opposition between the agents’ utility functions that makes the situation adversarial

Search Vs Games

- Typical Search problem
  - No adversary
  - Heuristic techniques can find optimal solution
  - Evaluation function could be estimate cost from start to end
  - Examples: planning and scheduling activities
- Games
  - Adversary present
  - Optimal solution depends on opponent
  - Evaluation function depends on goodness of game postion
  - Example-Chess, Checkers etc

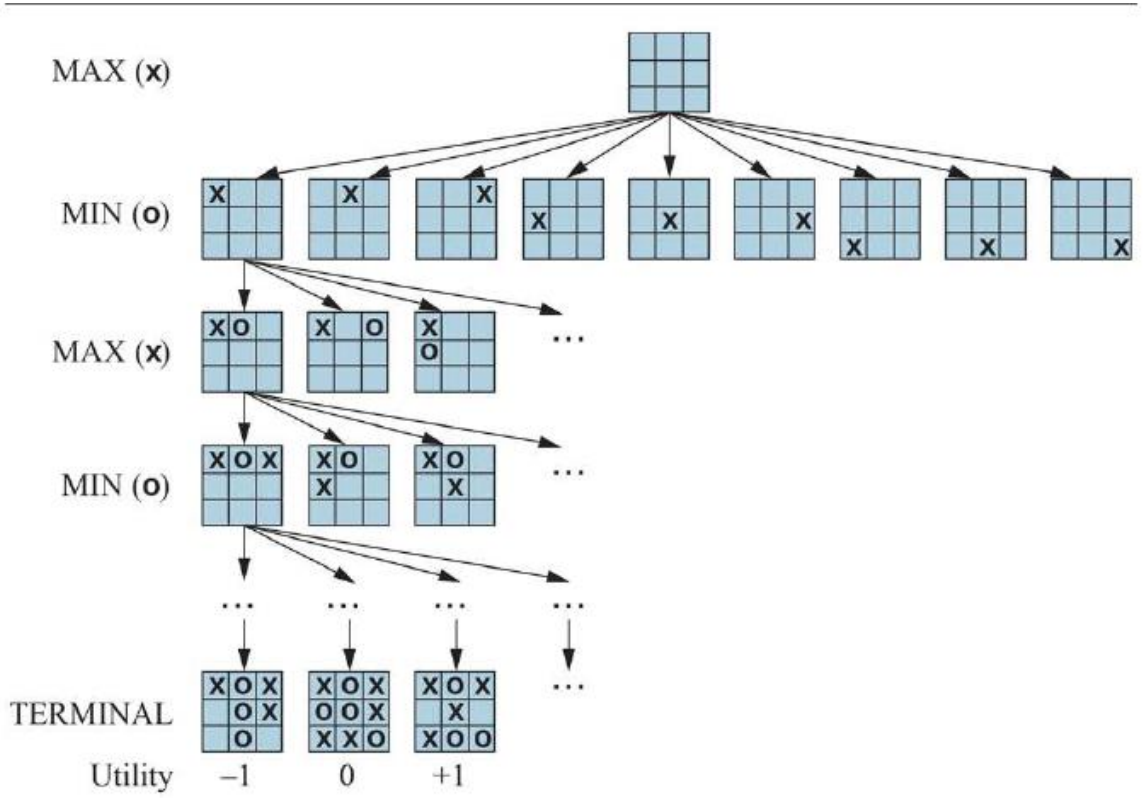
Game as a search problem

- Can be defined as search problem as follows
  - S0: The **initial state**, which specifies how the game is set up at the start – Example board configuration in chess
  - PLAYER(s): Defines which player has the move in a state
  - ACTIONS(s): Returns the set of legal moves in a state
  - RESULT(s, a): The **transition model**, which defines the result of a move
  - Successor function: list of (move, state) pairs specifying legal moves
  - TERMINAL-TEST(s): A **terminal test**, which is true when the game is over and false otherwise
  - TERMINAL STATES: States where the game has ended are called **terminal states**
  - UTILITY(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
    - In chess, the outcome is a win, loss, or draw, with values +1, 0, or 1

Two player zero sum game

- Games most commonly studied within AI (such as chess and Go) are what game theorists call deterministic, two-player, turn-taking, **perfect information, zero-sum games**
- "Perfect information" is a synonym for "fully observable"
- "Zero-sum" means that what is good for one player is just as bad for the other- there is no "win-win" outcome
- For games we often use the term **move** as a synonym for "action" and **position** as a synonym for "state."

Example Tic-Tac-Toe game



A (partial) game tree for the game of tic-tac-toe. The top node is the initial state, and MAX moves first, placing an x in an empty square. We show part of the tree, giving alternating moves by MIN (o) and MAX (x), until we eventually reach terminal states, which can be assigned utilities according to the rules of the game.

- Players are called MIN and MAX

For tic-tac-toe the game tree is relatively small—fewer than $9! = 362{,}880$ terminal nodes (with only 5,478 distinct states)

But for chess there are over nodes, so the game tree is best thought of as a theoretical construct that we cannot realize in the physical world

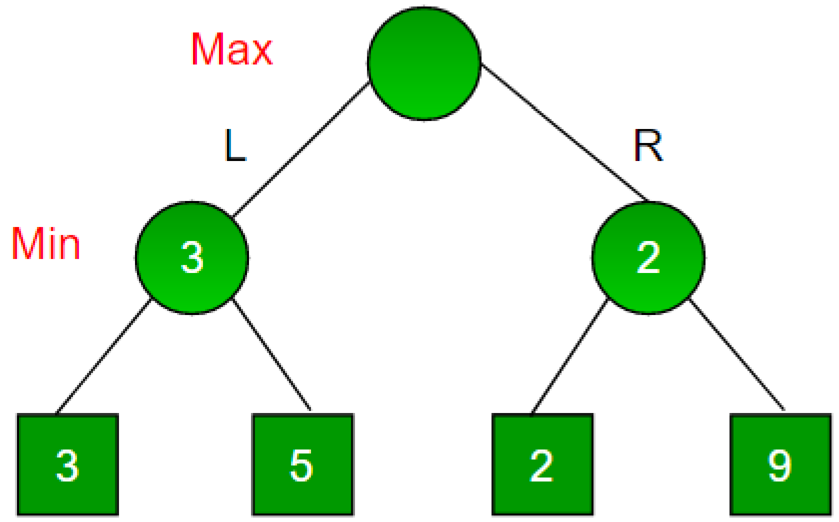
Optimal Decisions in Games

- MAX wants to find a sequence of actions leading to a win,
- However MIN has its own plans
- So MAX's strategy must be a conditional plan—a contingent strategy specifying a response to each of MIN's possible moves
- In games that have a binary outcome (win or lose), AND–OR search to generate the conditional plan could be used
- For games with multiple outcome scores, slightly more general algorithm called **minimax search** can be used

Game problem formulation

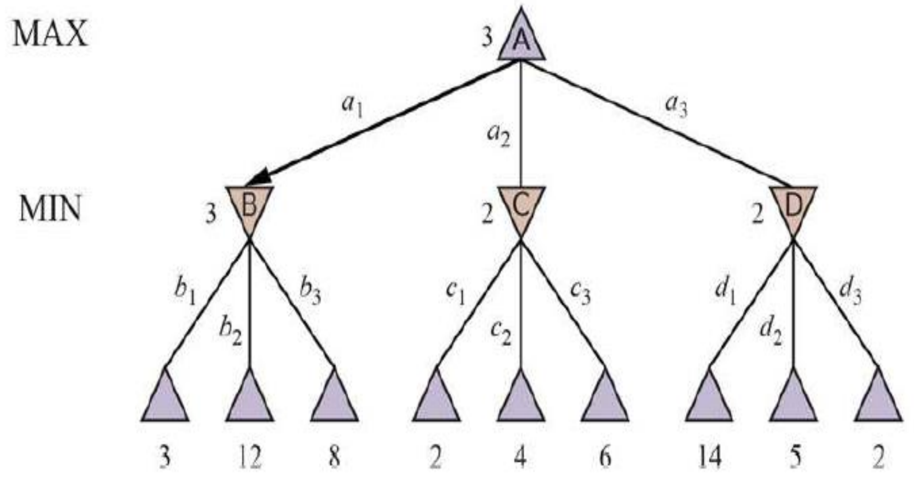
- initial state: board position
- player: player to move
- successor function: a list of legal (move, state) pairs
- goal test: whether the game is over – terminal states
- utility function: gives a numeric value for the terminal states (win, loss, draw)

Minimax Example



https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/

Minimax Example



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.

•Perfect play for deterministic games
•Max will select best move for itself
•Min will also select best move for itself
•Objective is to choose a move to position with highest minimax value= best achievable payoff against best play
E.g., 2-ply game:

Minimax search algorithm

- Search algorithm that finds the best move for MAX by
  - trying all actions and
  - choosing the actions whose resulting state has the highest MINIMAX value
- It is a recursive algorithm that proceeds all the way down to the leaves of the tree
- Then **backs up** the minimax values through the tree as the recursion unwinds
- **NOTE:** In some games, the word "move" means that both players have taken an action; therefore the word **ply** is used to unambiguously mean one move by one player, bringing us one level deeper in the game tree.

```
function MINIMAX-DECISION($state$) returns $an\ action$
  return $\arg\max_{a\ \in\ \text{ACTIONS}(s)}$ MIN-VALUE(RESULT($state$, $a$))
```

```
function MAX-VALUE($state$) 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($s$, $a$)))
  return $v$
```

```
function MIN-VALUE($state$) 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($s$, $a$)))
  return $v$
```

An algorithm for calculating minimax decisions. It returns the action corresponding to the best possible move, that is, the move that leads to the outcome with the best utility, under the assumption that the opponent plays to minimize utility. The functions MAX-VALUE and MIN-VALUE go through the whole game tree, all the way to the leaves, to determine the backed-up value of a state. The notation $\text{argmax}_{a \in S} f(a)$ computes the element a of set S that has the maximum value of $f(a)$.

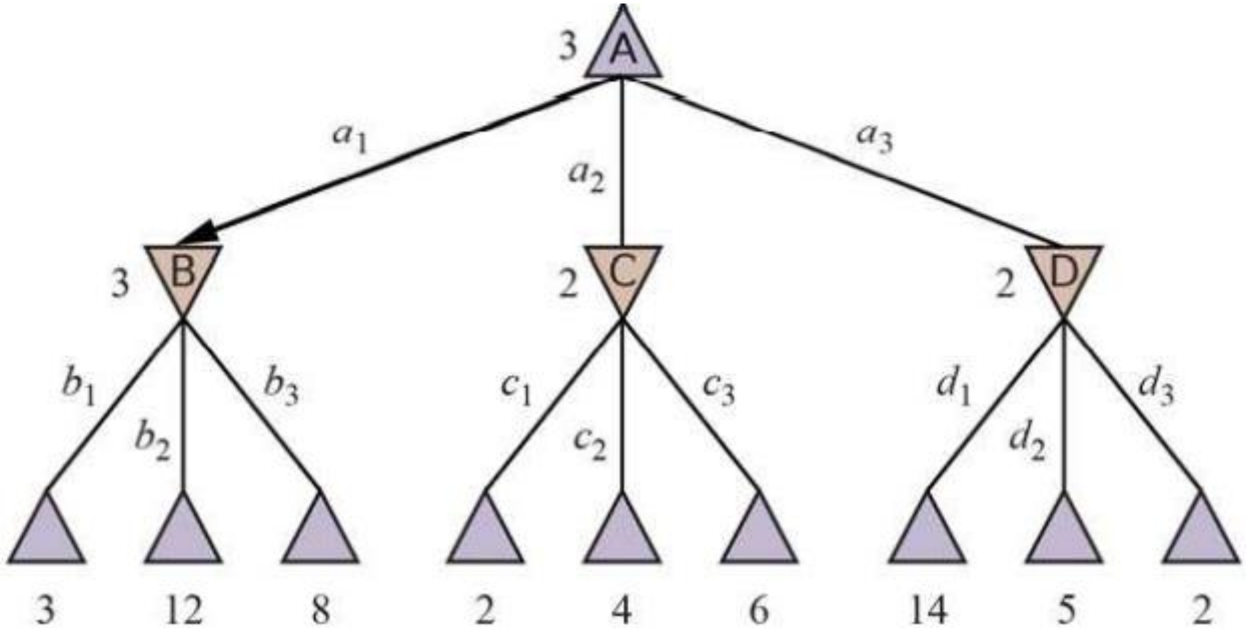
Time and space complexity

- Minimax algorithm performs a complete depth-first exploration of the game tree
- If the maximum depth of the tree is m and there are b legal moves at each point, then
  - Time complexity of the minimax algorithm is $O(b^m)$
  - Space complexity is $O(bm)$for an algorithm that generates all actions at once
- This exponential complexity makes minimax algorithm impractical for complex games
- For example
  - Chess has a branching factor of about 35 and the average game has depth of about 80 ply, and it is not feasible to search states$35^{80} = 10^{123}$ States
- However, minimax can serve as basis for deriving more practical algorithms

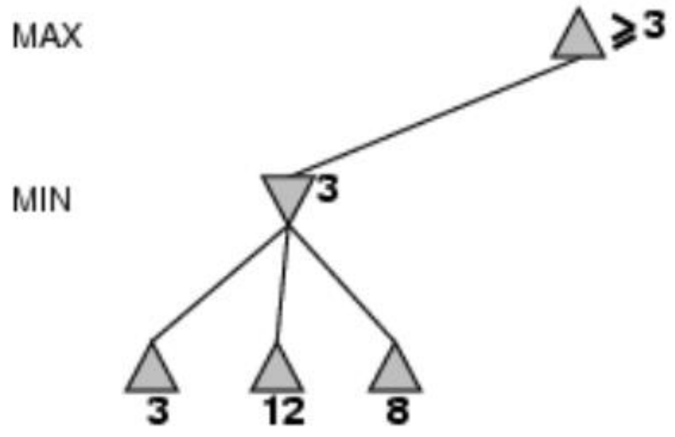
Alpha Beta Pruning

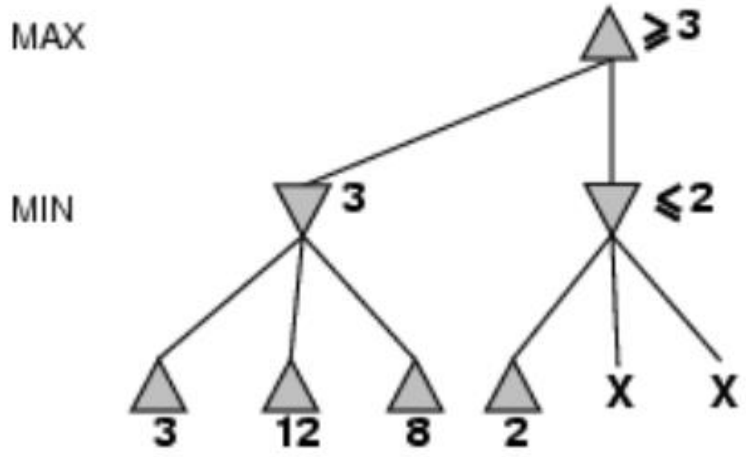
- Number of game states is exponential in the depth of the tree
- No algorithm can completely eliminate the exponent, but we can sometimes cut it in half
- Computing the correct minimax decision without examining every state by **pruning** large parts of the tree that make no difference to the outcome
- The particular technique we examine is called **alpha–beta pruning**

Alpha Beta Pruning example

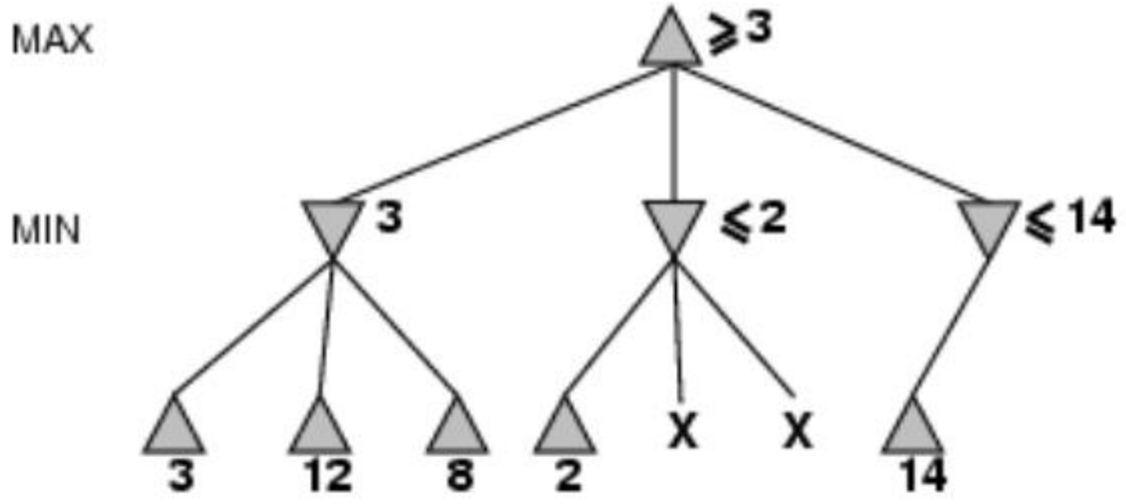


Alpha Beta Pruning example

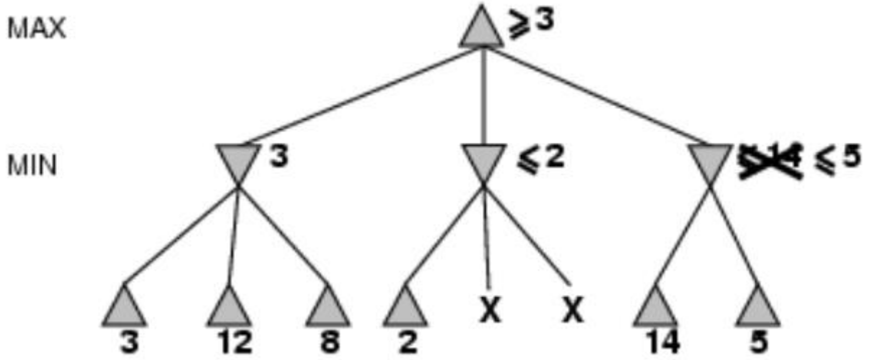




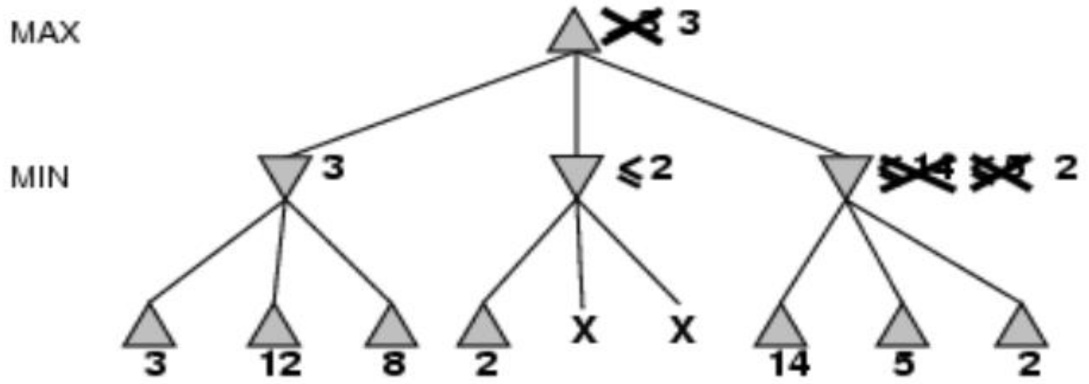
Alpha Beta Pruning example



Alpha Beta Pruning example

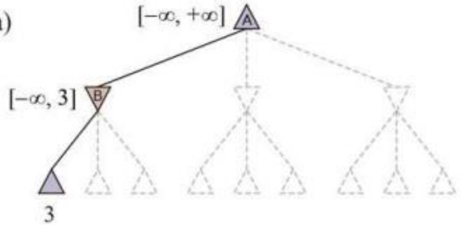


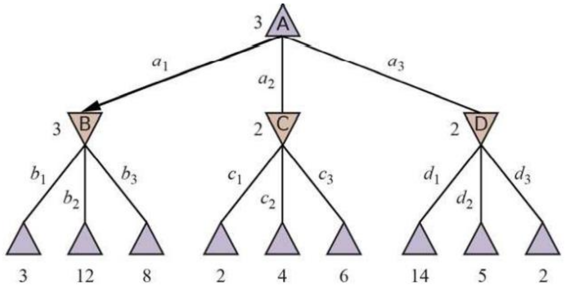
Alpha Beta Pruning example

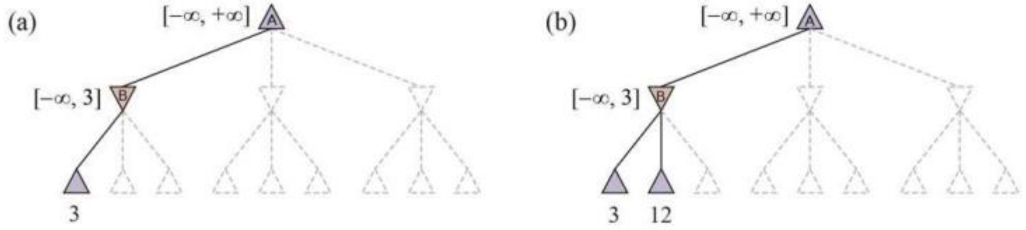


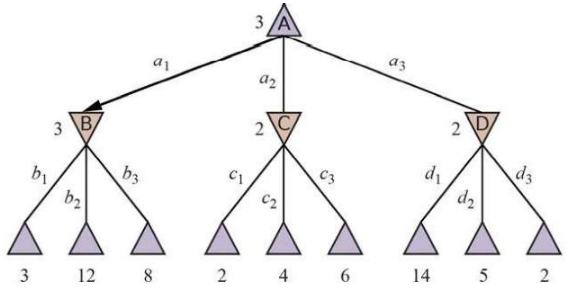
$$\begin{aligned}\textsc{Minimax}(root) &= \max(\min(3,12,8), \min(2,x,y), \min(14,5,2)) \\ &= \max(3, \min(2,x,y), 2) \\ &= \max(3, z, 2) \qquad \text{where } z = \min(2,x,y) \leq 2 \\ &= 3.\end{aligned}$$

(a)

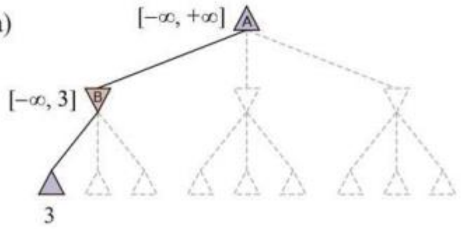





(a)
[−∞, +∞]
A
[−∞, 3]
B
3
(b)
[−∞, +∞]
A
[−∞, 3]
B
3
12

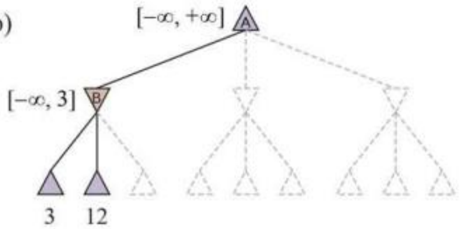

3
A
a_1
a_2
a_3
3
B
2
C
2
D
b_1
b_2
b_3
c_1
c_2
c_3
d_1
d_2
d_3
3
12
8
2
4
6
14
5
2

(a)

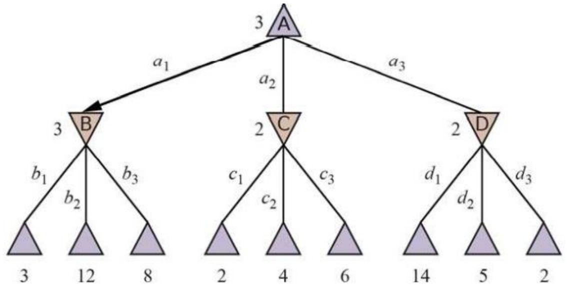


[−∞, +∞]
A
[−∞, 3]
B
3

(b)

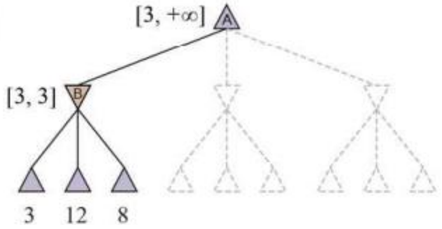


[−∞, +∞]
A
[−∞, 3]
B
3
12

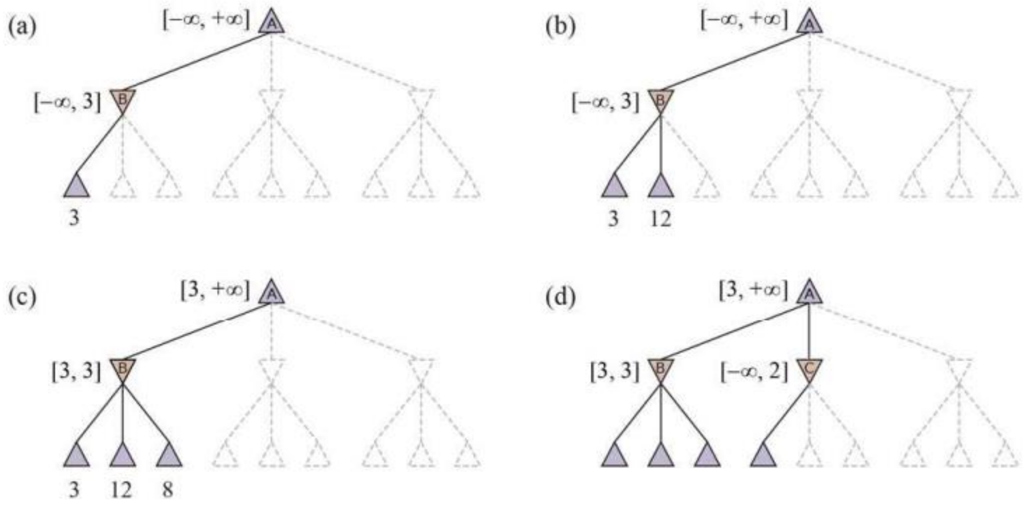


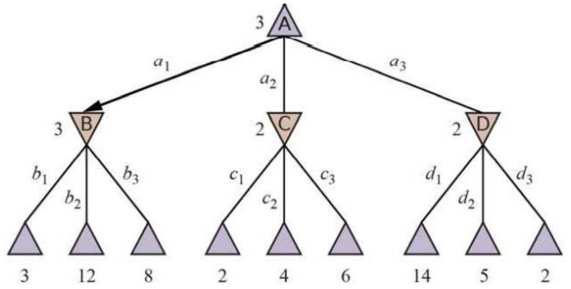
3
A
a_1
a_2
a_3
3
B
2
C
2
D
b_1
b_2
b_3
c_1
c_2
c_3
d_1
d_2
d_3
3
12
8
2
4
6
14
5
2

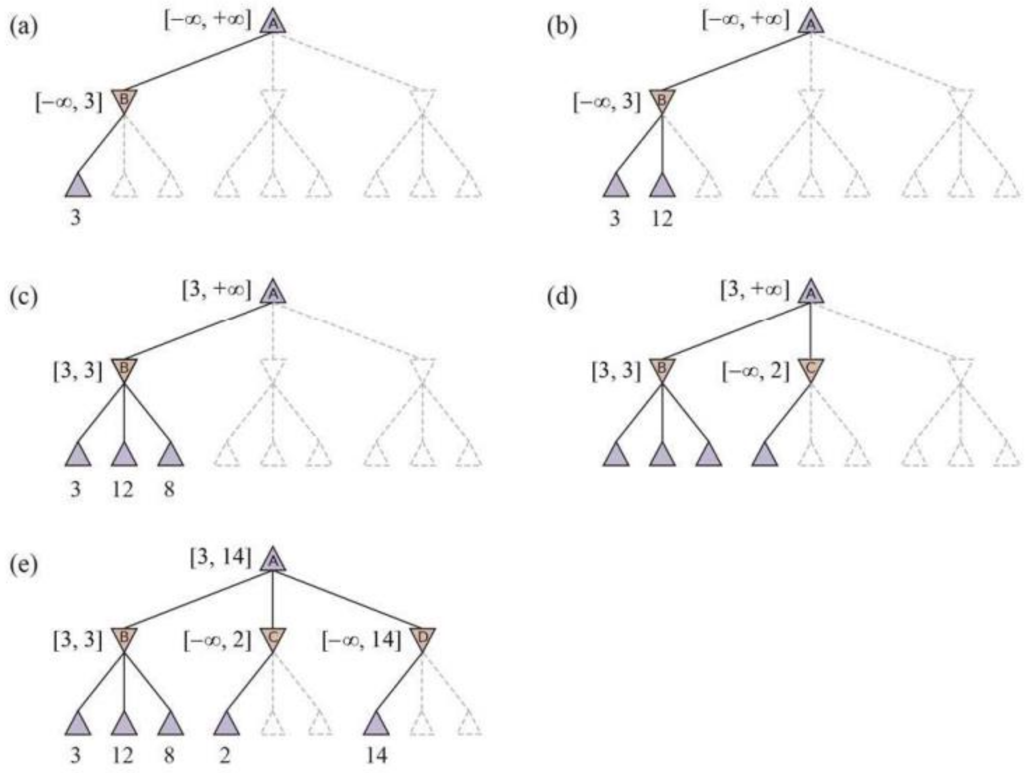
(c)

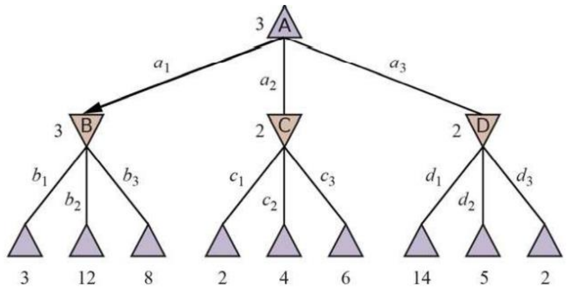


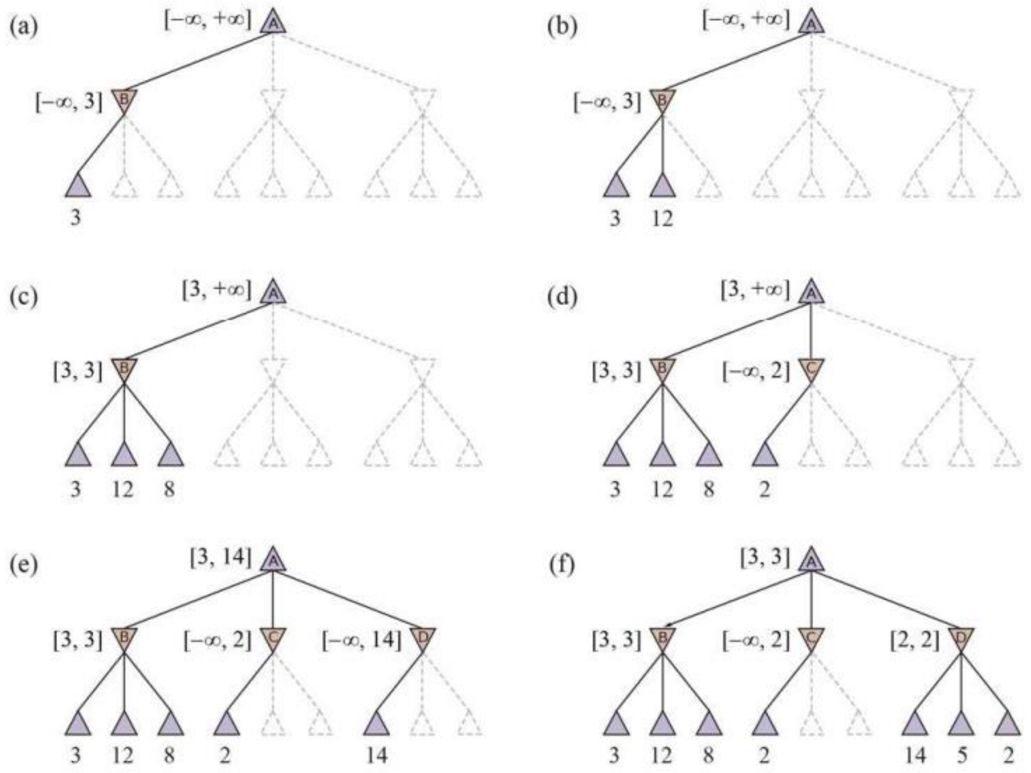
[3, +∞]
A
[3, 3]
B
3
12
8

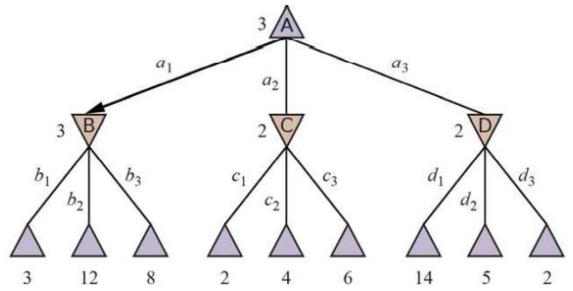

(a)
[−∞, +∞]
A
[−∞, 3]
B
3
(b)
[−∞, +∞]
A
[−∞, 3]
B
3
12
(c)
[3, +∞]
A
[3, 3]
B
3
12
8
(d)
[3, +∞]
A
[3, 3]
B
[−∞, 2]
C


3
A
a_1
a_2
a_3
3
B
2
C
2
D
b_1
b_2
b_3
c_1
c_2
c_3
d_1
d_2
d_3
3
12
8
2
4
6
14
5
2

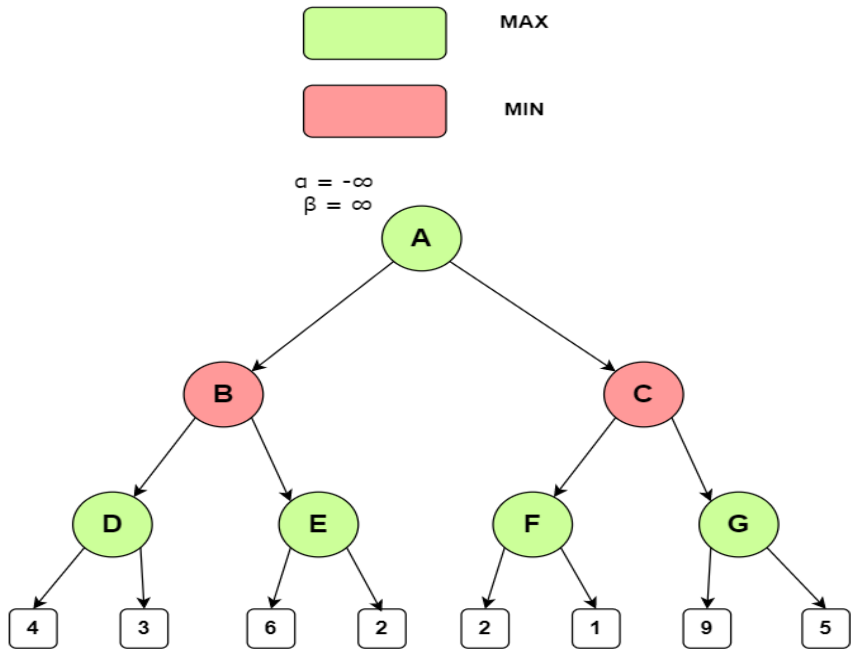

(a)
[−∞, +∞]
A
[−∞, 3]
B
3
(b)
[−∞, +∞]
A
[−∞, 3]
B
3
12
(c)
[3, +∞]
A
[3, 3]
B
3
12
8
(d)
[3, +∞]
A
[3, 3]
B
[−∞, 2]
C
(e)
[3, 14]
A
[3, 3]
B
[−∞, 2]
C
[−∞, 14]
D
3
12
8
2
14


3
A
a_1
a_2
a_3
3
B
2
C
2
D
b_1
b_2
b_3
c_1
c_2
c_3
d_1
d_2
d_3
3
12
8
2
4
6
14
5
2


(a)
[−∞, +∞] A
[−∞, 3] B
3
(b)
[−∞, +∞] A
[−∞, 3] B
3 12
(c)
[3, +∞] A
[3, 3] B
3 12 8
(d)
[3, +∞] A
[3, 3] B
[−∞, 2] C
3 12 8 2
(e)
[3, 14] A
[3, 3] B
[−∞, 2] C
[−∞, 14] D
3 12 8 2 14
(f)
[3, 3] A
[3, 3] B
[−∞, 2] C
[2, 2] D
3 12 8 2 14 5 2


3 A
a_1 a_2 a_3
3 B
2 C
2 D
b_1 b_2 b_3 c_1 c_2 c_3 d_1 d_2 d_3
3 12 8 2 4 6 14 5 2

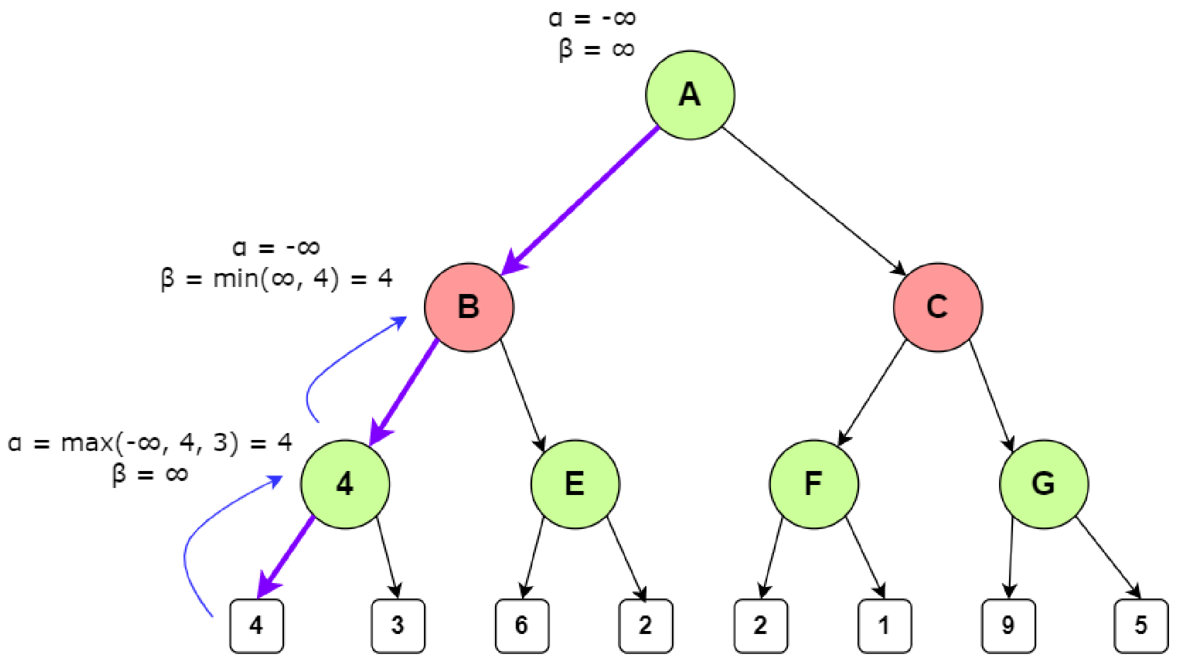
Alpha Beta Pruning Example



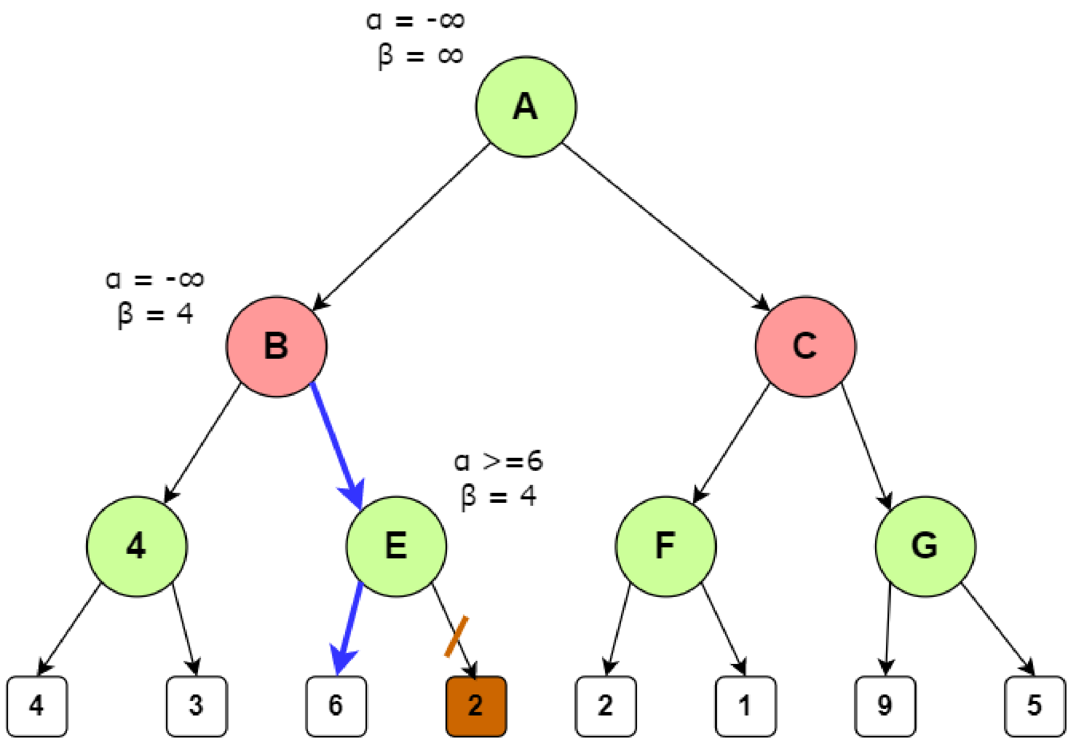
The condition used by alpha beta pruning to prune the useless branches is: alpha >= beta

- Alpha is used and updated only by the Maximizer
- It represents the maximum value found so far. The initial value is set to -∞
- The value of alpha is passed down to children node, but the actual node values are passed up while backtracking

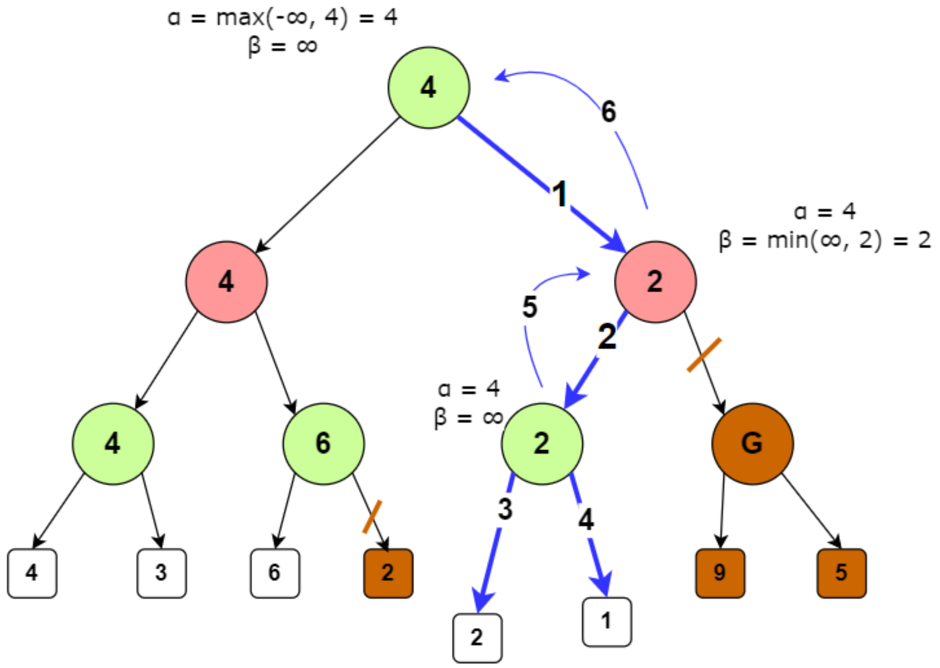
- Beta is used and updated only by the Minimizer It represents the minimum value found
- The initial value is ∞
- The value of beta is passed down to the children node, but actual node values are used to backtrack



We move down, depth first, until we reach the left most leaf node. Here, we have to explore both the leaf nodes to determine the maximum value for max's turn. The maximum value is found to be 4, which is the value that alpha is set at. Now, the node's value, i.e., 4 is used to back track. When we reach node B, the value of beta is updated to become 4 as well.



Now, moving towards the next branch, we reach the leaf node with value 6. The turn is of max, therefore, it will choose the maximum value from its children node. When we explore 6, we realize that the value of alpha will either be 6 or greater than 6. We do not need to explore the other child node since the value of alpha is now greater than beta. So 2 is pruned.



B's node value becomes 4 as it is the minimizer and min(4,6) = 4. This value is passed up and the value of alpha is changed by the maximizer to become 4 as well. Now moving towards the next leaf node, we follow the blue path to reach 2. We need to explore both the branches to find the maximum value for the maximizer, which is 2. This node value is passed up and the minimizer changes beta's value to 2. Now alpha is again greater than beta so the entire right branch is pruned.