Artificial & Computational Intelligence

AIML CLZG557

M3 : Game Playing

Dr. Sudheer Reddy

BITS Pilani
Pilani Campus

Course Plan

M1 Introduction to AI

M2 Problem Solving Agent using Search

M3 Game Playing

M4 Knowledge Representation using Logics

M5 Probabilistic Representation and Reasoning

M6 Reasoning over time

M7 Ethics in AI

Module 3 : Searching to play games

A. Minimax Algorithm

B. Alpha-Beta Pruning

C. Making imperfect real time decisions

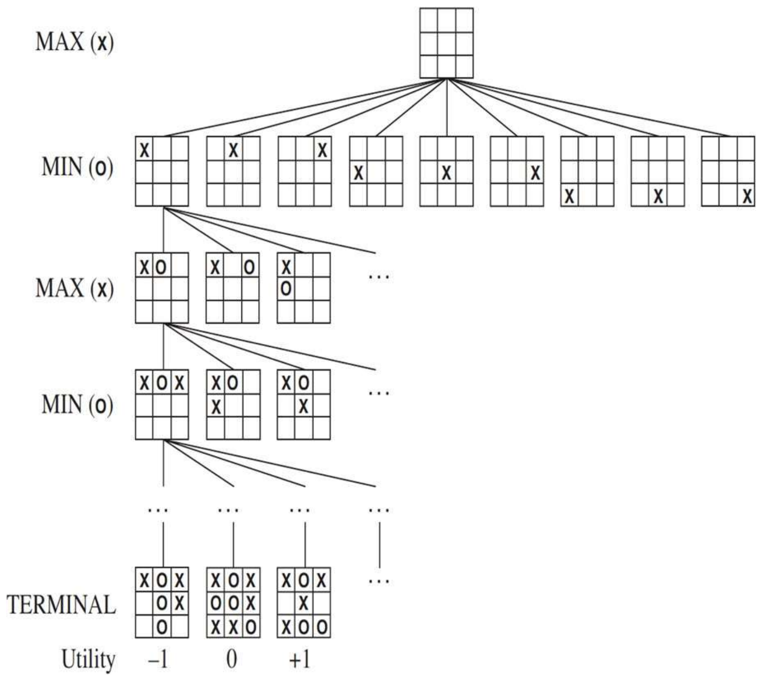
Learning Objective

At the end of this class , students should be able to:

1. Convert a given problem into adversarial search problem
2. Formulate the problem solving agent components
3. Design static evaluation function value for a problem
4. Construct a Game tree
5. Apply Min-Max
6. Apply and list nodes pruned by alpha pruning and nodes pruned by beta pruning

Task Environment

Phases of Solution Search by PSA



Assumptions – Environment :

Static

Observable

Discrete

Deterministic

Number of Agents

Game Problem

Study & design of games enables the computers to model ways in which humans think & act hence simulating human intelligence.

AI for Gaming:

- Interesting & Challenging Problem
- Larger Search Space Vs Smaller Solutions
- Explore to better the Human Computer Interaction

Characteristics of Games:

- Observability
- Stochasticity
- Time granularity
- Number of players

Adversarial Games:

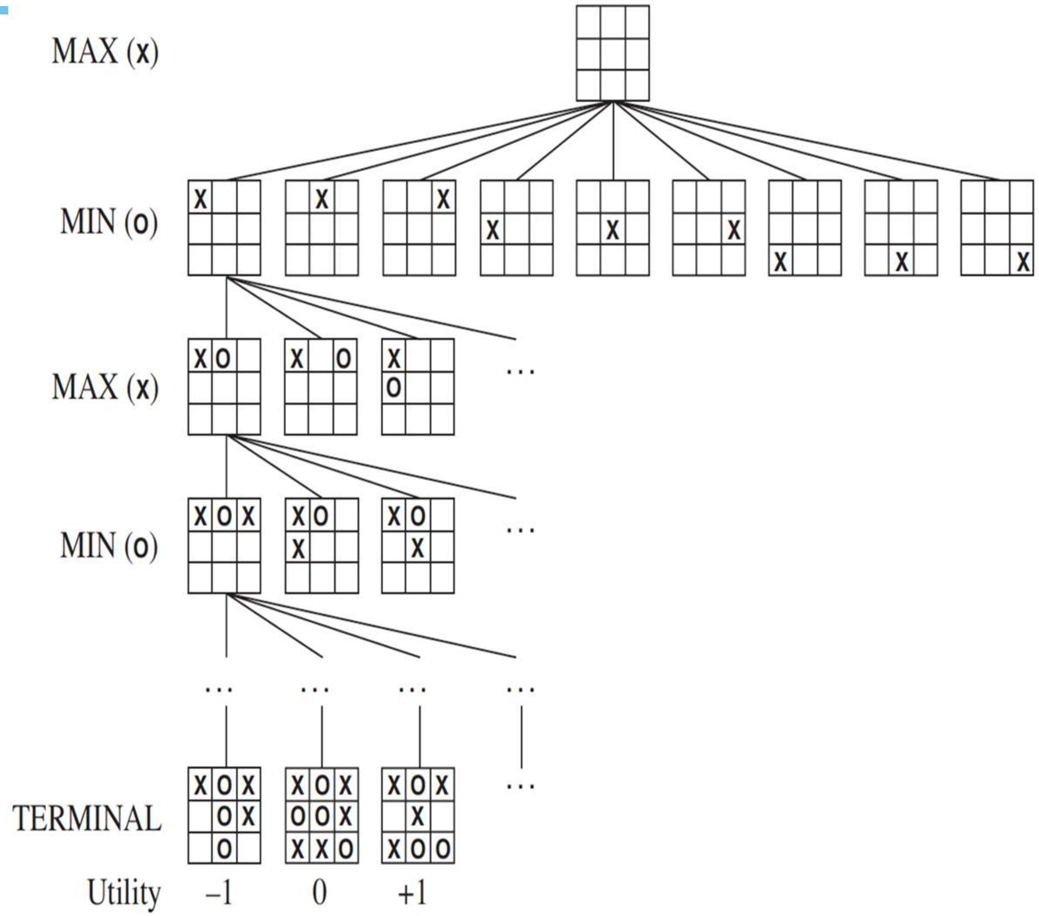
Goals of agents are in conflict where one’s optimized step would reduce the utility value of the other.

Games as Search Problem

PSA : Representation of Game:

- INITIAL STATE: S0
- PLAYER(s)
- ACTIONS(s)
- RESULT(s, a)
- TERMINAL-TEST(s)
- UTILITY(s, p)

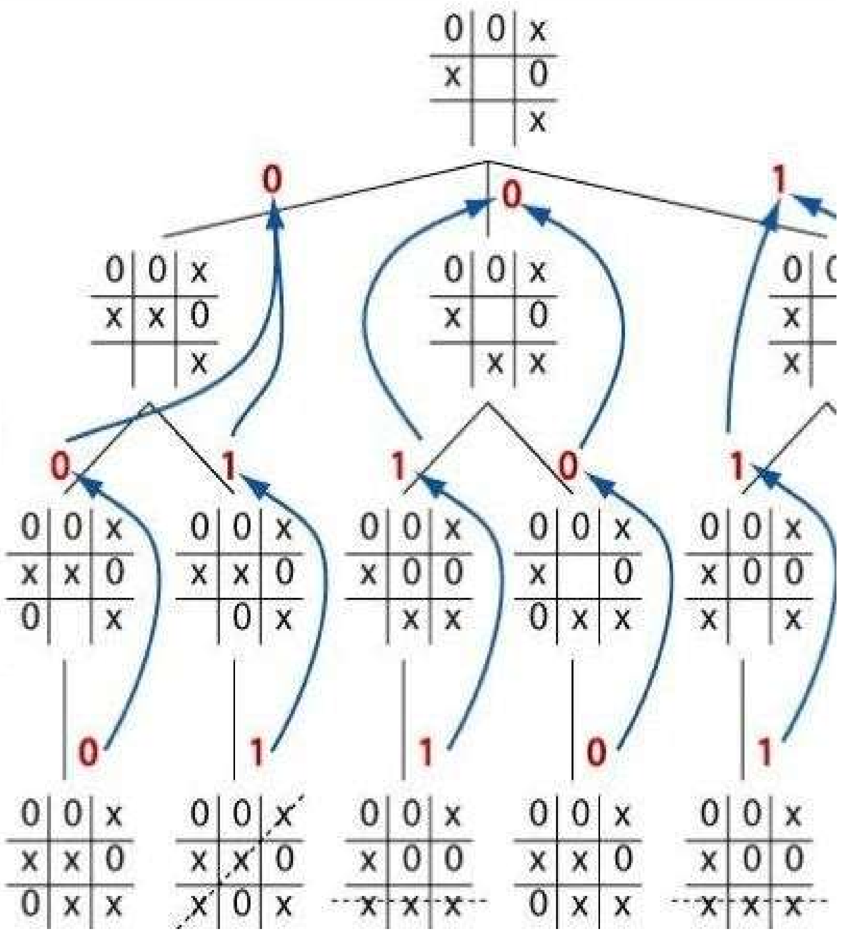
Eg., Tic Tac Toe



Min-Max Algorithm

Idea: Uses Depth – First search exploration to decide the move

Let
start Player = MAX
Depth m =3



Min-Max Algorithm

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

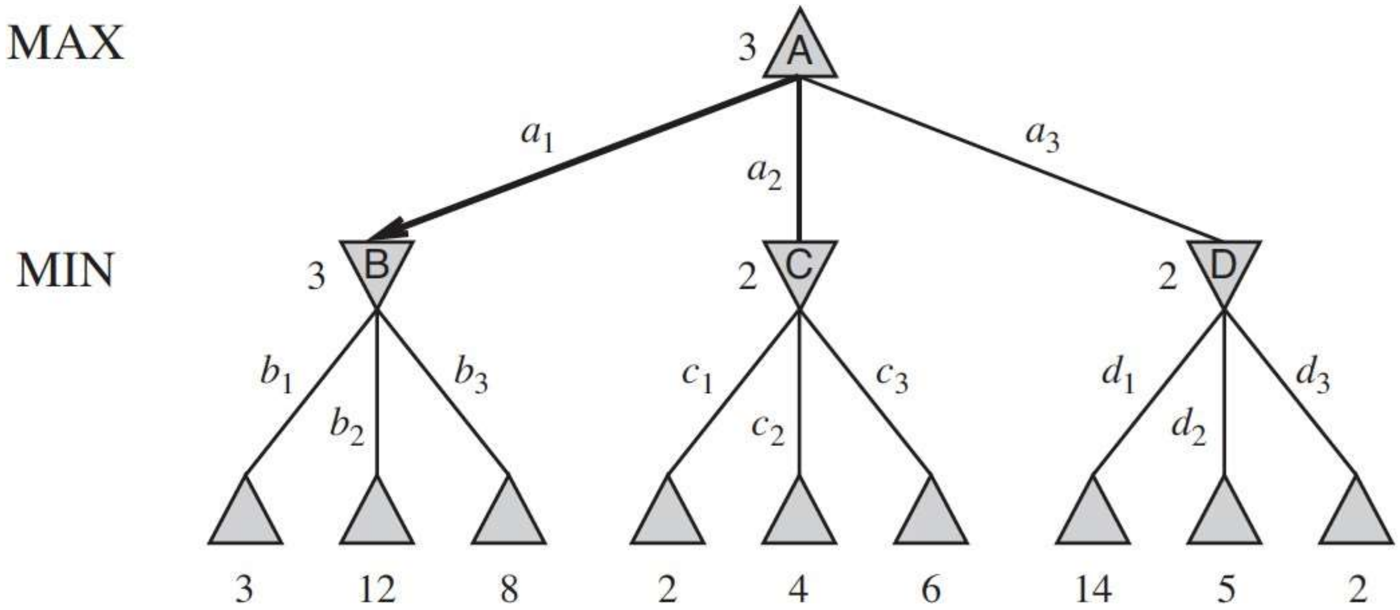
---

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

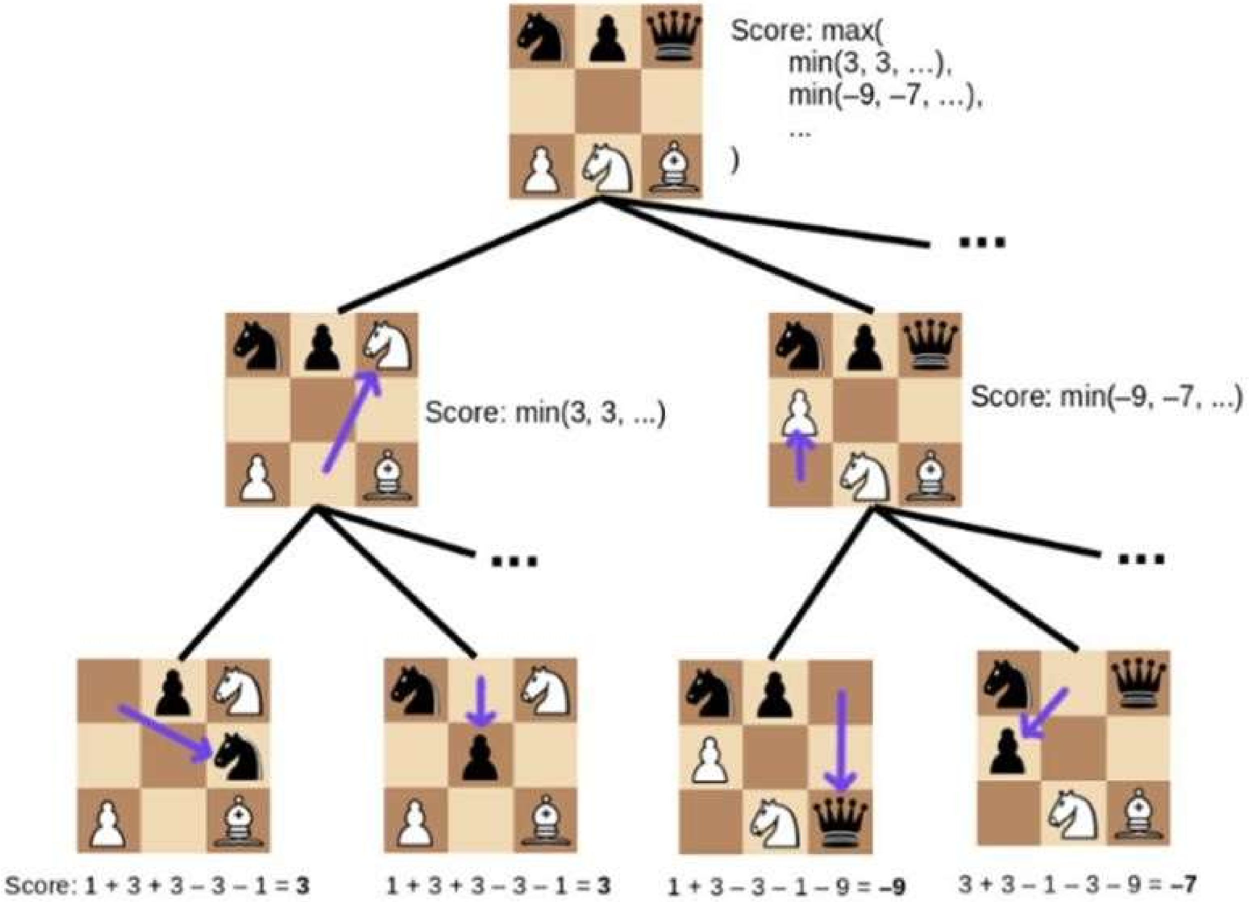
---

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

Book Example

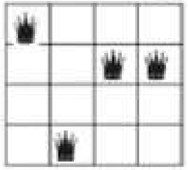


Design of Static Evaluation Values



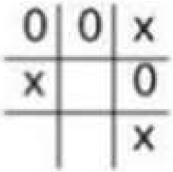
Design of Static Evaluation Values

N-Queens



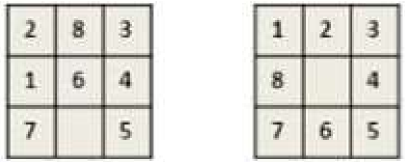
1	4	2	2	4

Tic-Tac-Toe



Max's Share	2
Min's Share	1
Board Value	1

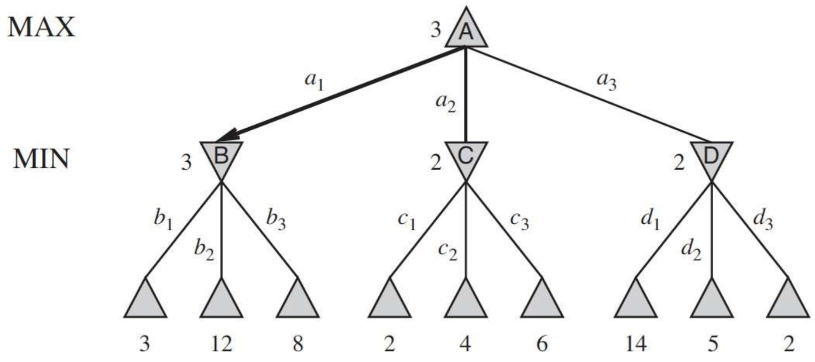
N-Tile



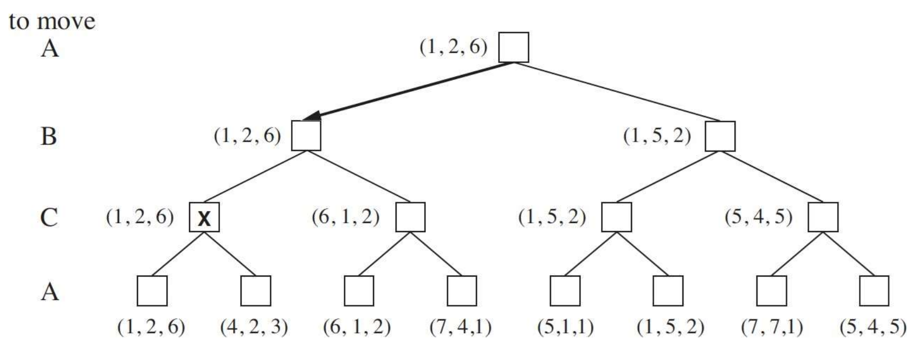
No.of.Tiles Out of Place	5

$$Eval(S) = w_1 f_1(s) + w_2 f_2(s) + ... + w_n f_n(s)$$

$$= 0.6\,(MaxChance - MinChance) + 0.4\,(MaxPairs - MinPairs)$$



Two Player Game : 1-Ply Game

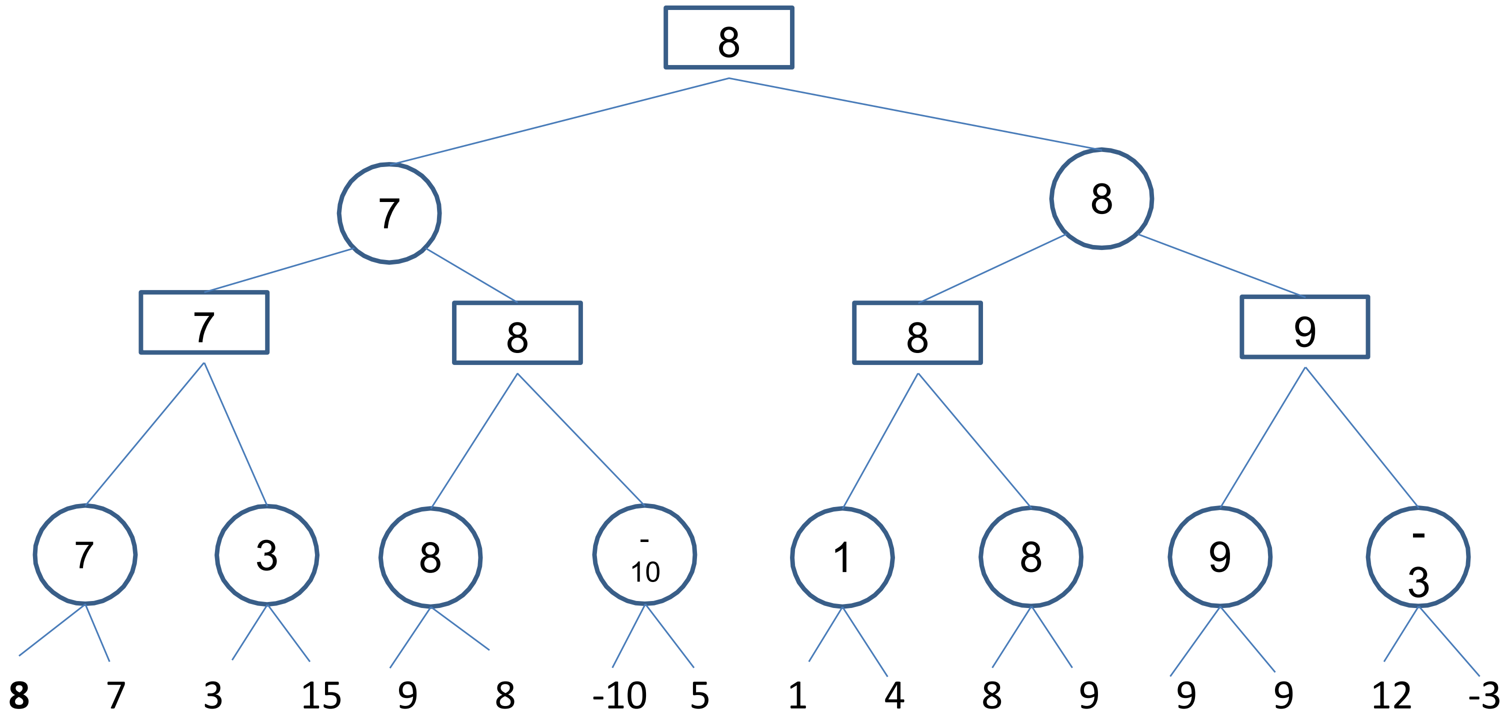


Multiplayer Game

Min-Max Algorithm – Example -1

Squares represent MAX nodes

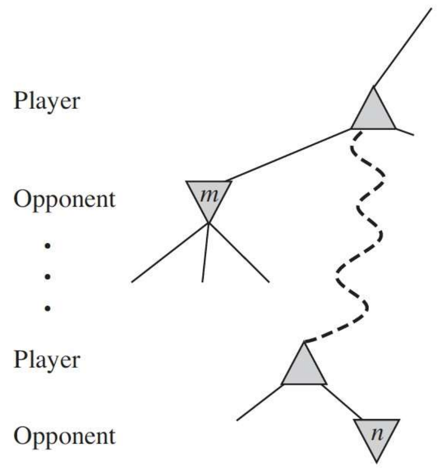
Circles represent MIN nodes

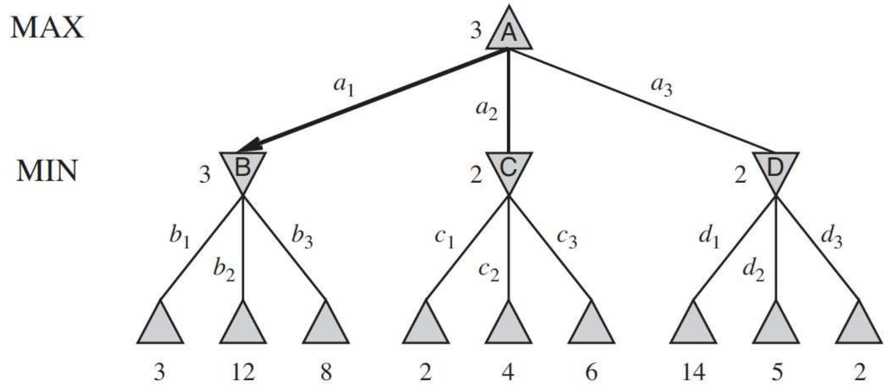


Alpha – beta Pruning

General Principle:

At a node ***n*** if a player has better option at the parent of ***n*** or further up, then ***n*** node will never be reached .Hence the entire subtree pruned

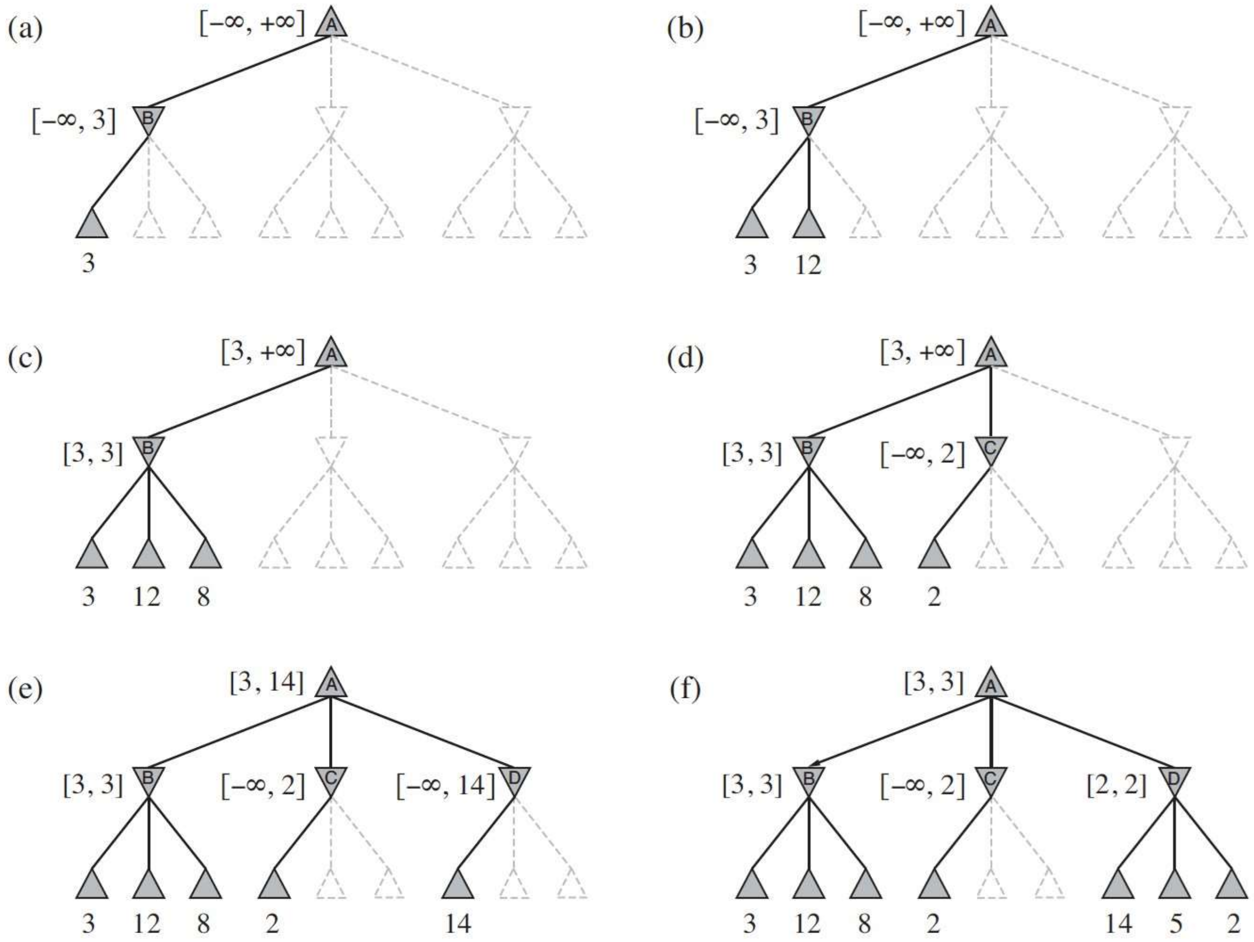




$$\begin{aligned}\text{MINIMAX}(root) &= \max(\min(3,12,8), \min(2,x,y), \min(14,5,2)) \\ &= \max(3, \min(2,x,y), 2) \\ &= \max(3, z, 2) \qquad \text{where } z = \min(2,x,y) \le 2 \\ &= 3.\end{aligned}$$

Alpha Beta Pruning

Book Example



Min-Max Algorithm

Alpha beta Modifications

```
function ALPHA-BETA-SEARCH(state) returns an action
   v ← MAX-VALUE(state, −∞, +∞)
   return the action in ACTIONS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value
   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← −∞
   for each a in ACTIONS(state) do
      v ← MAX(v, MIN-VALUE(RESULT(s,a), α, β))
      if v ≥ β then return v
      α ← MAX(α, v)
   return v

function MIN-VALUE(state, α, β) returns a utility value
   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← +∞
   for each a in ACTIONS(state) do
      v ← MIN(v, MAX-VALUE(RESULT(s,a) , α, β))
      if v ≤ α then return v
      β ← MIN(β, v)
   return v
```

Is it possible to compute the minimax decision for a node without looking at every successor node?

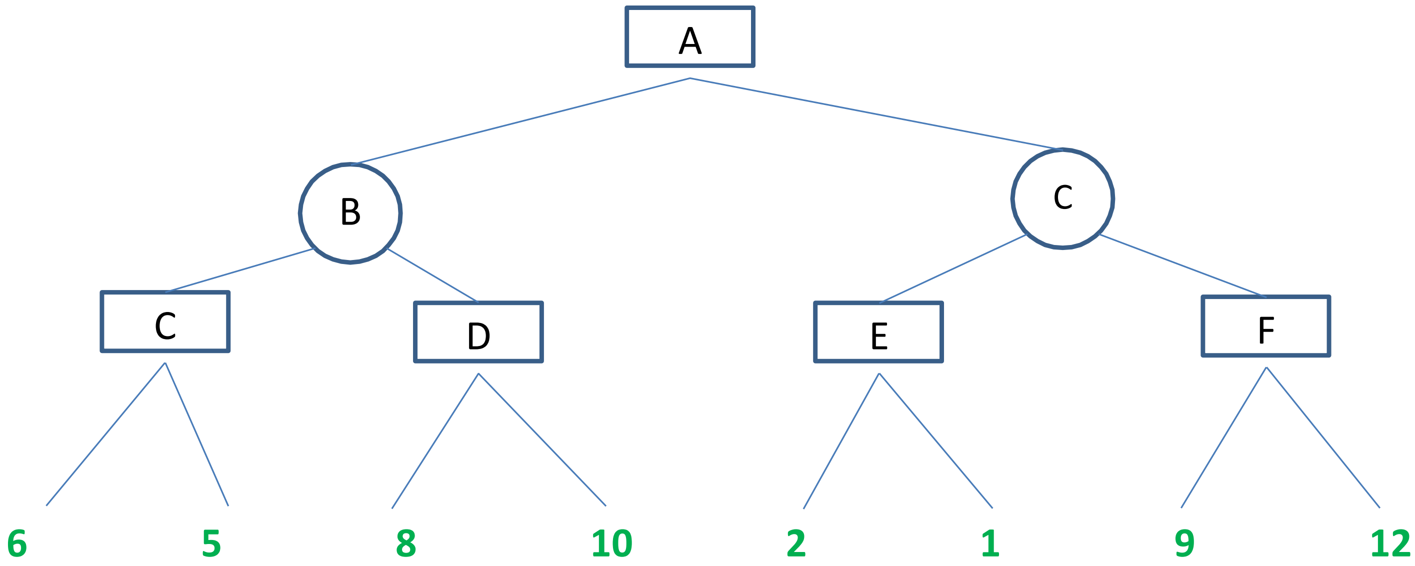
Alpha – beta Pruning

Steps in Alpha – Beta Pruning:

1. At root initialize alpha = -∞ and beta = + ∞. This is to set the worst case boundary to start the algorithm which aims to increase alpha and decrease beta as much as optimally possible
2. Navigate till the depth / limit specified and get the static evaluated numeric value.
3. For every value VAL being analyzed : Loop till all the leaf/terminal/specified state level nodes are analyzed & accounted for OR until **beta <= alpha.**
    1. If the player is MAX :
        1. If VAL > alpha
        2. then reset alpha = VAL
        3. also check **if** beta <=alpha **then** tag the path as unpromising (TO BE AVOIDED) **and** prune the branch from game tree. Rest of their siblings are not considered for analysis
    2. Else if the player is MIN:
        1. If VAL < beta
        2. then reset beta = VAL
        3. also check **if** beta <=alpha **then** tag the path as unpromising (TO BE AVOIDED) **and** prune the branch from game tree. Rest of their siblings are not considered for analysis

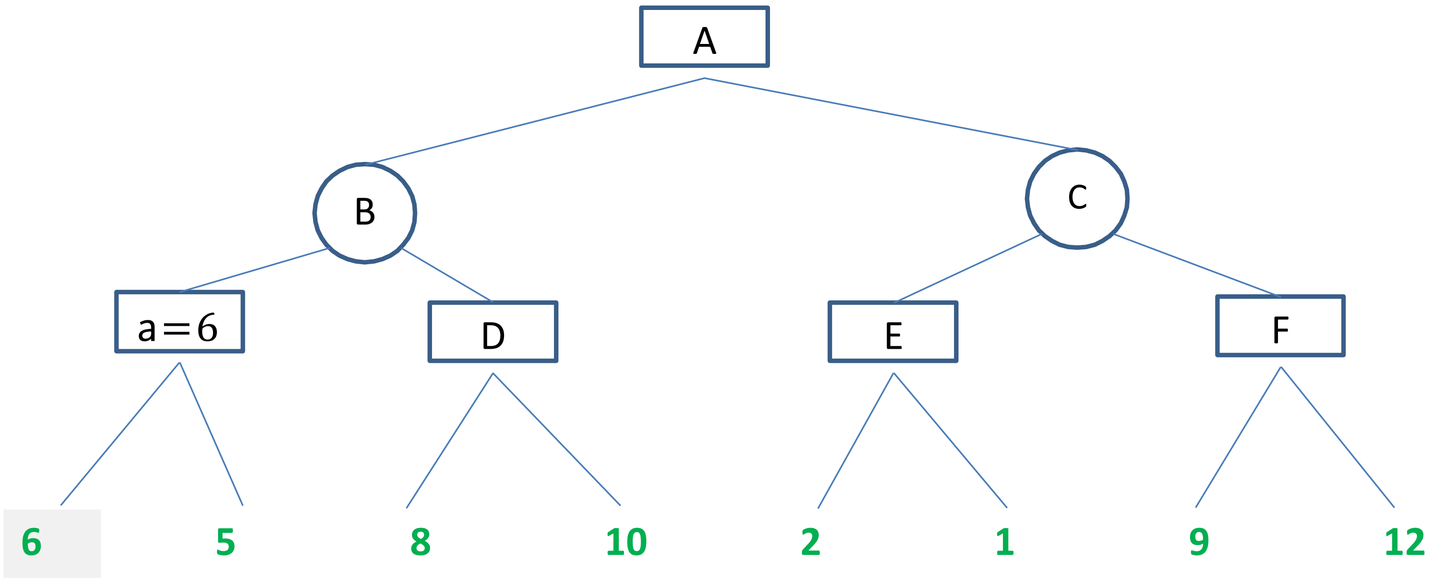
Alpha Beta Pruning - Another Example

Idea –Pruning

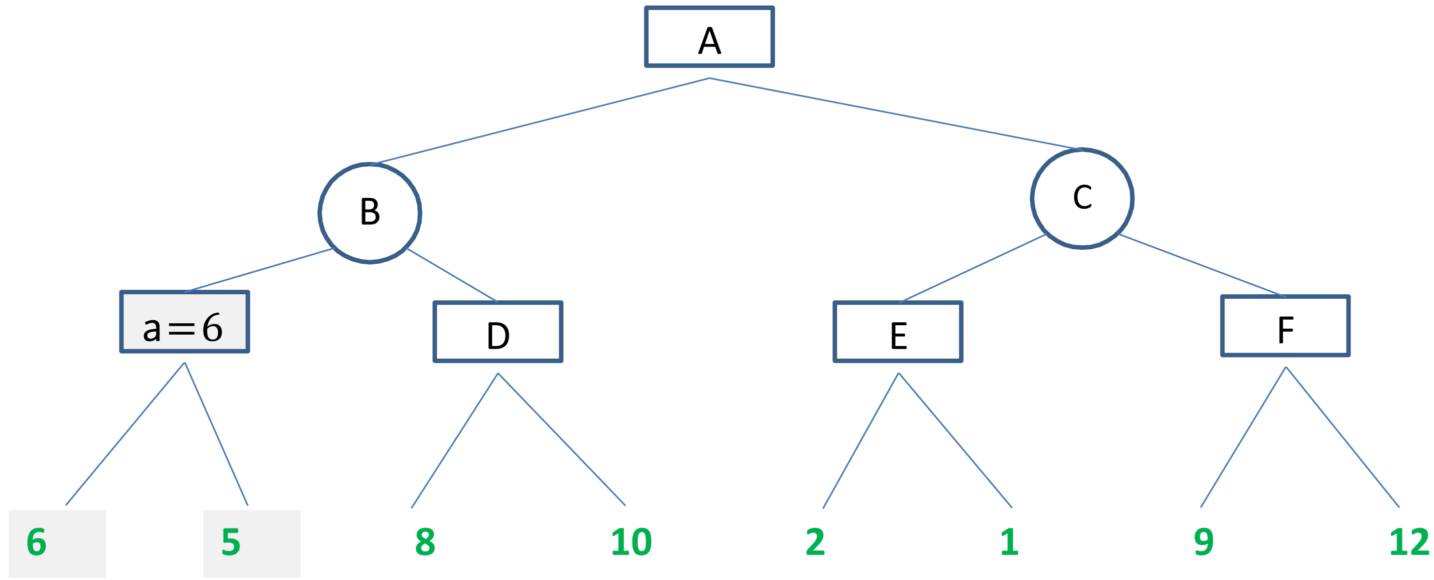


Alpha Beta Pruning

Idea –Pruning

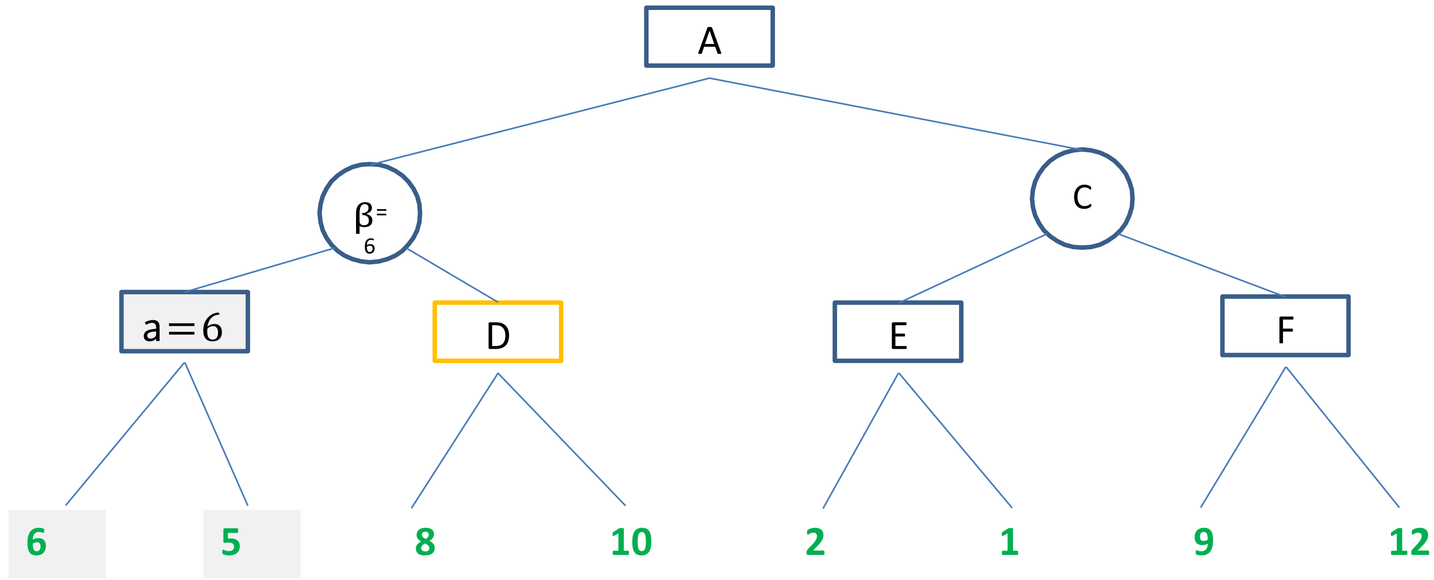


Idea –Pruning

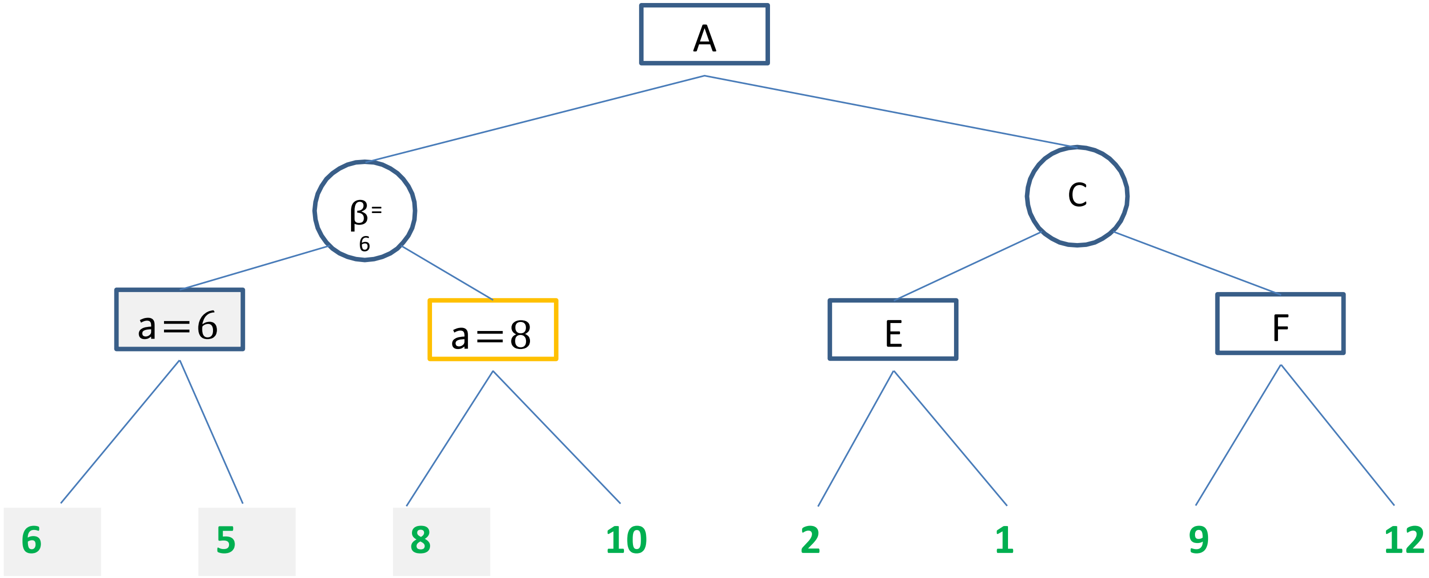


Alpha Beta Pruning

Idea –Pruning

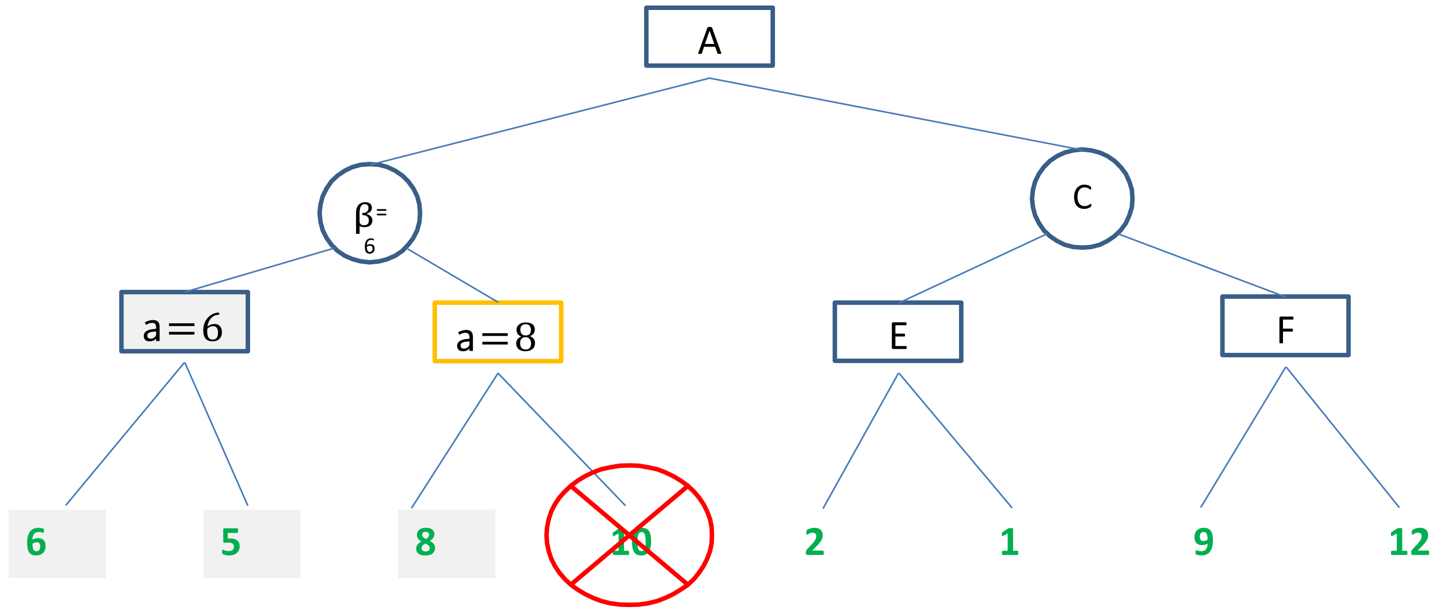


Idea – Alpha Pruning



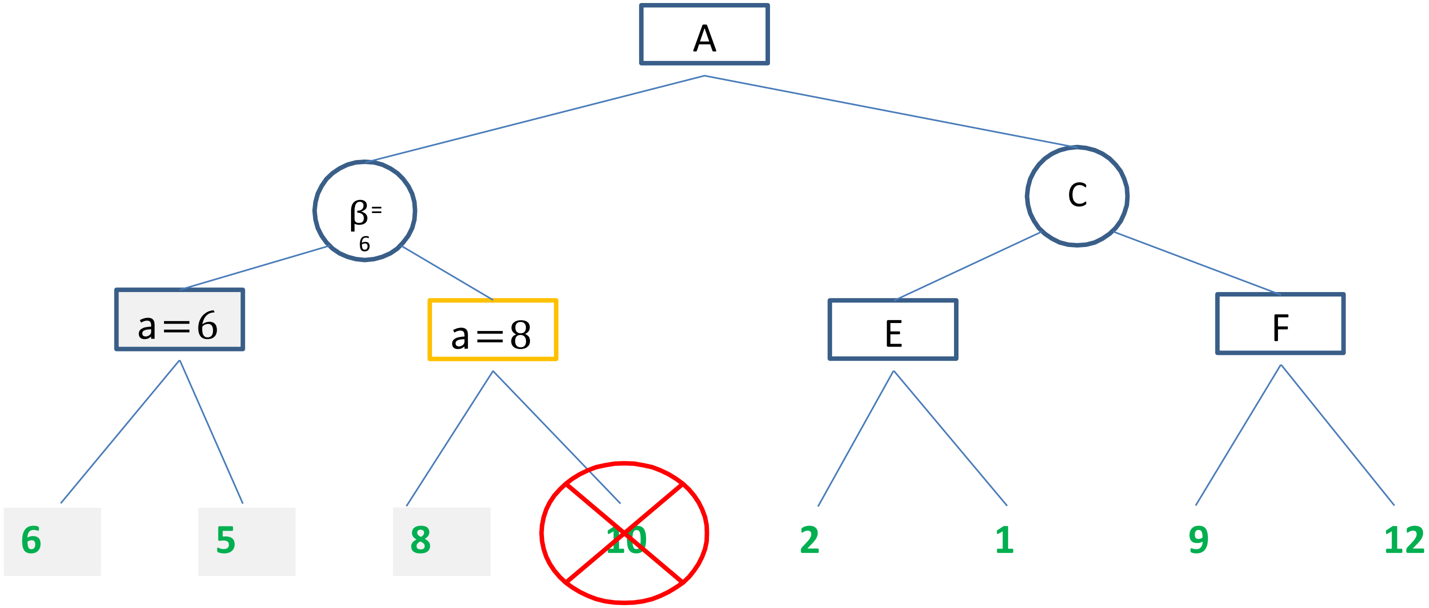
Alpha Beta Pruning

Idea – Beta Pruning

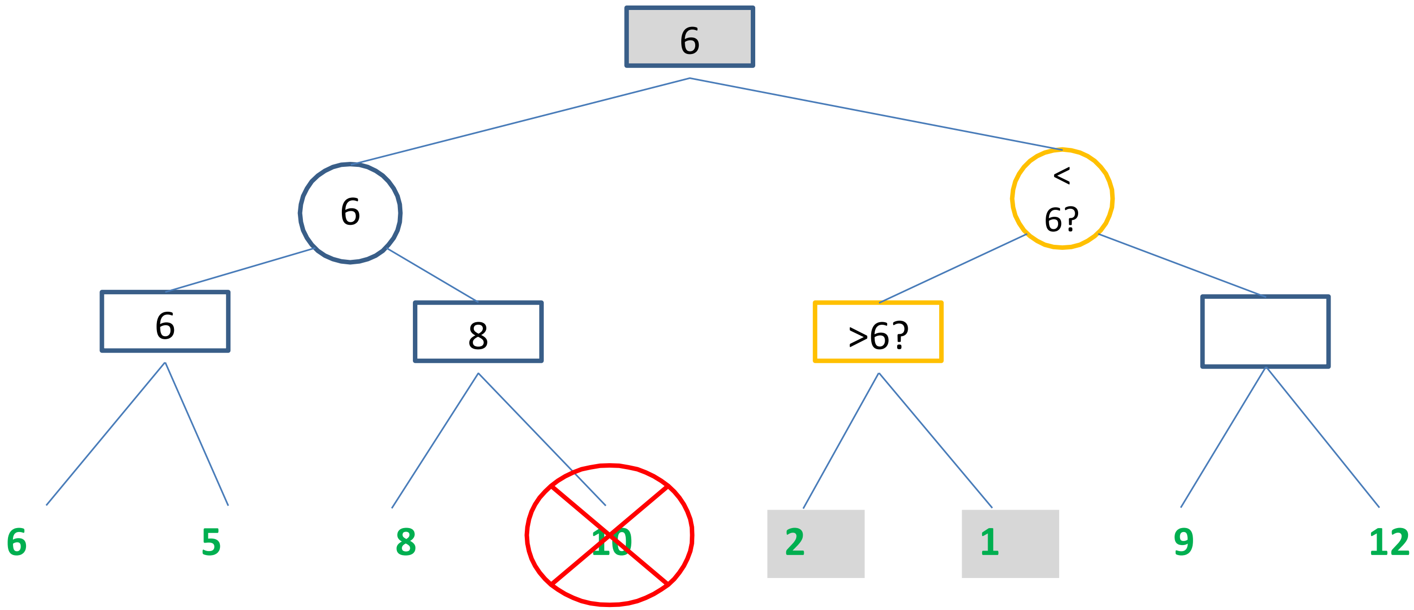


Alpha Beta Pruning

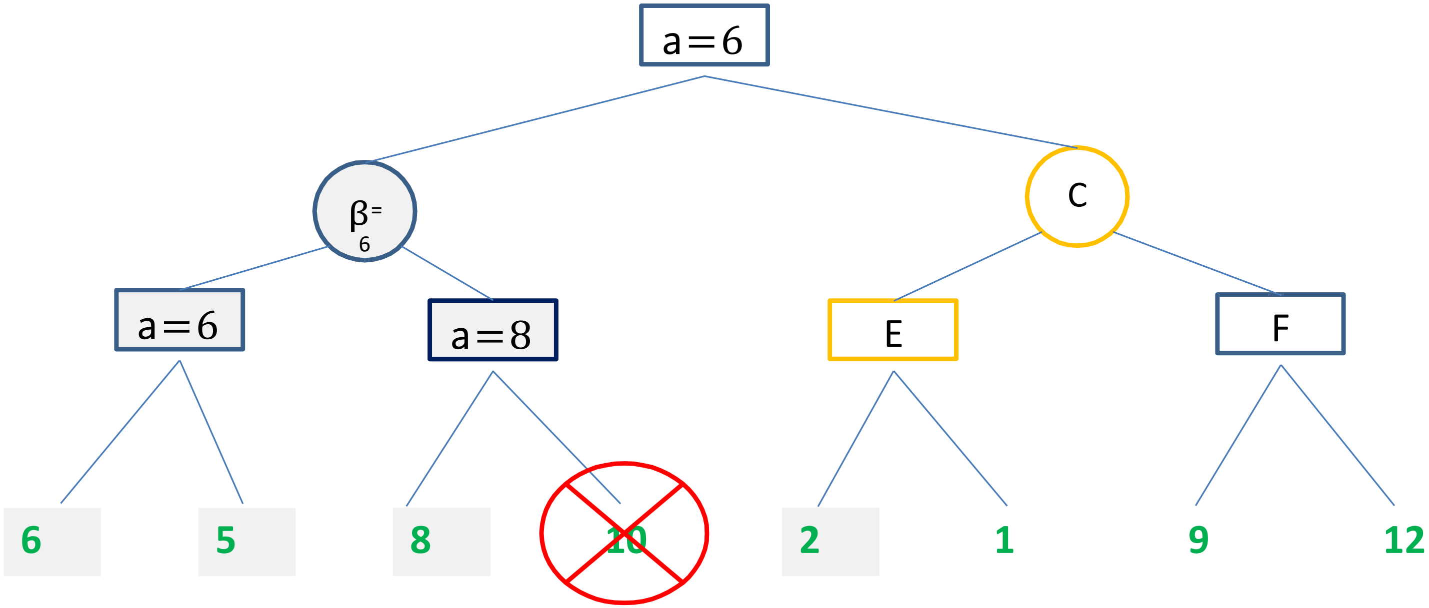
Idea –Pruning



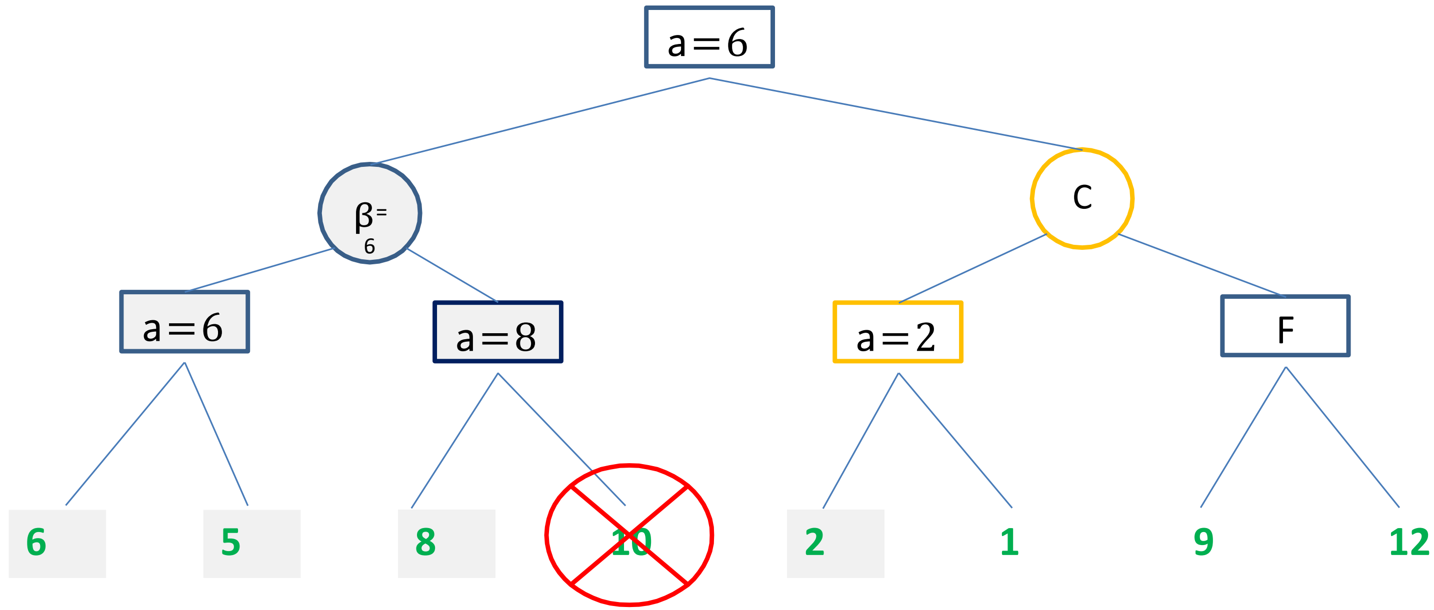
Idea –Pruning



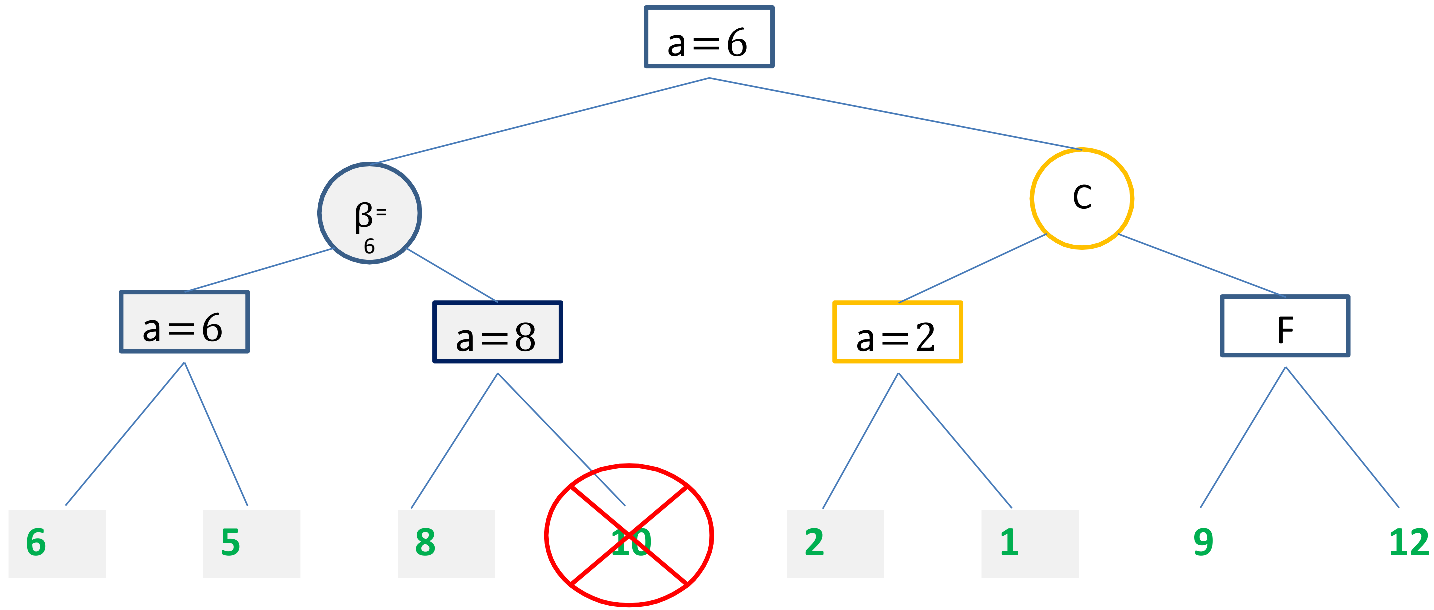
Idea –Pruning



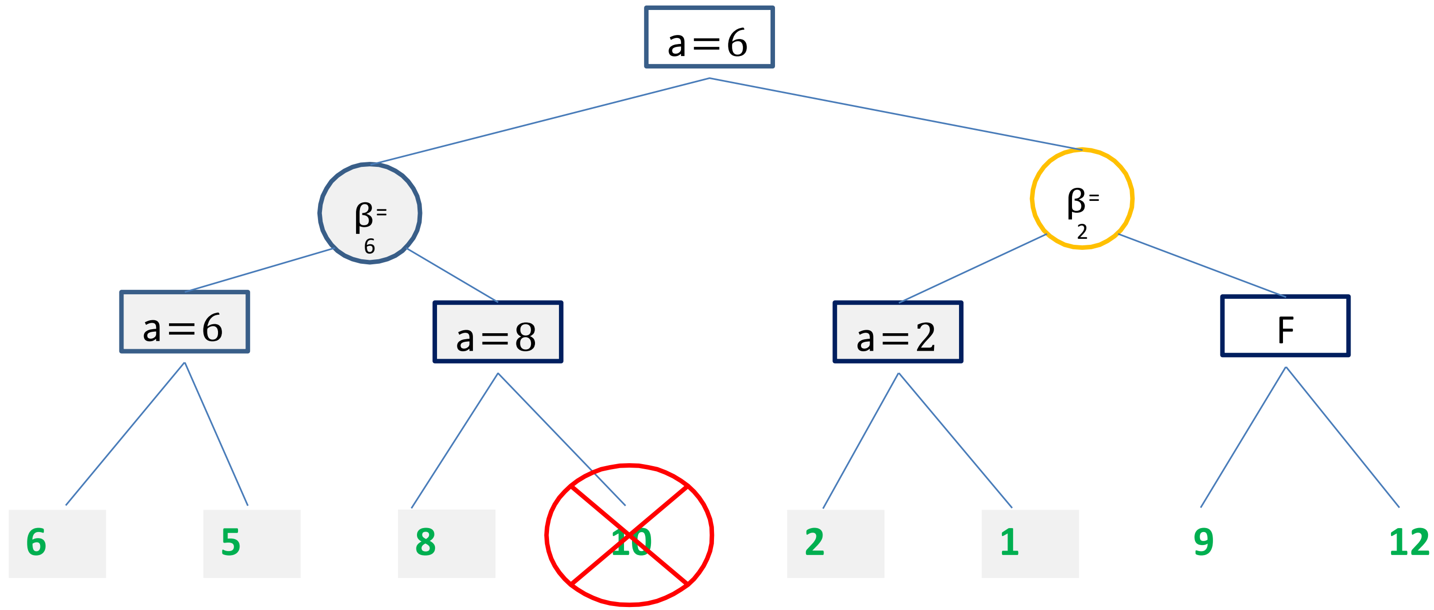
Idea –Pruning



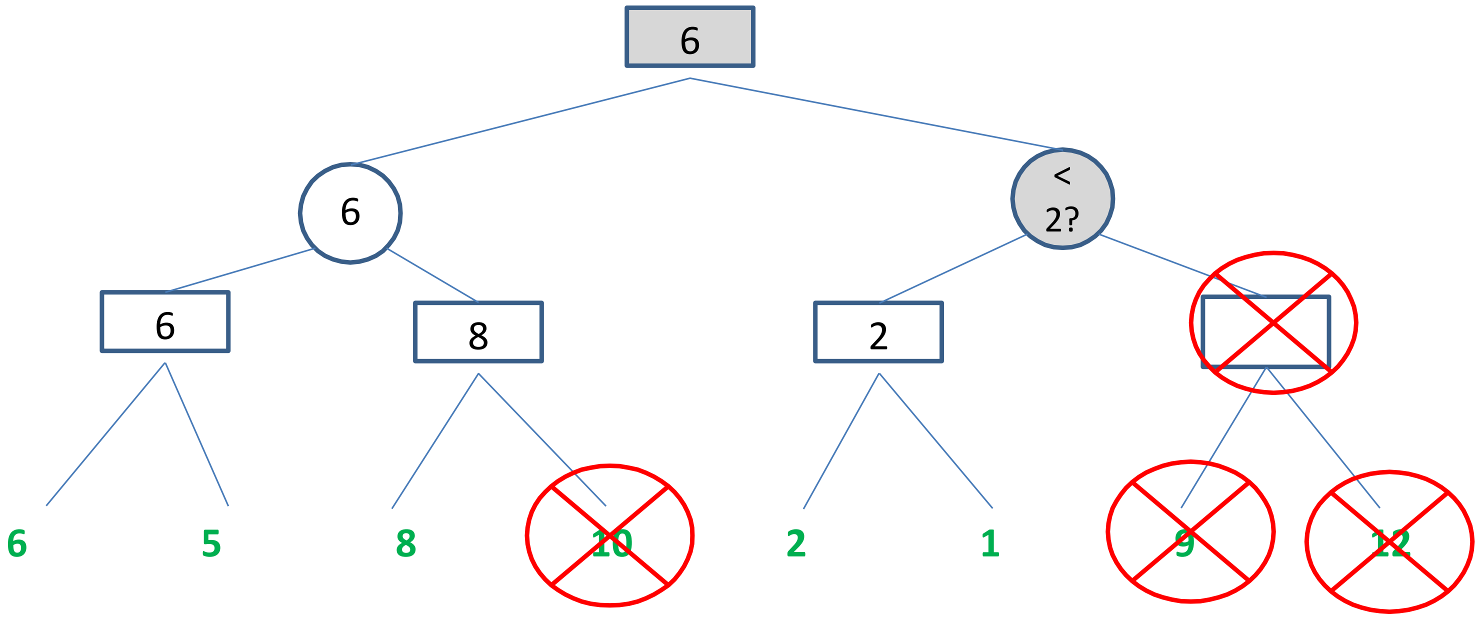
Idea –Pruning



Idea –Pruning



Idea – Alpha Pruning



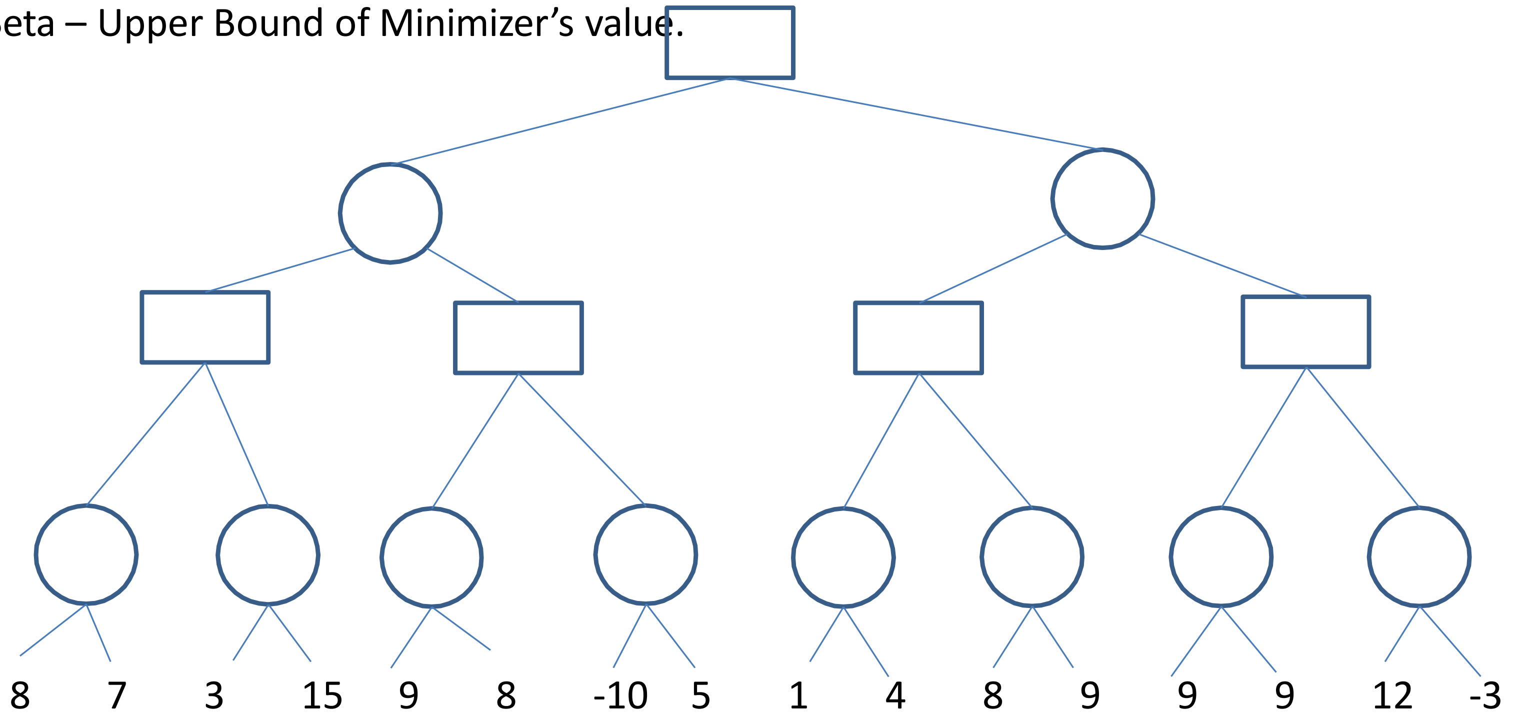
Alpha – Lower bound of Maximizer's value. Perceived value that Maximizer hopes to against a competitive Minimizer

Alpha – beta Pruning – Example -4

Do for practice.

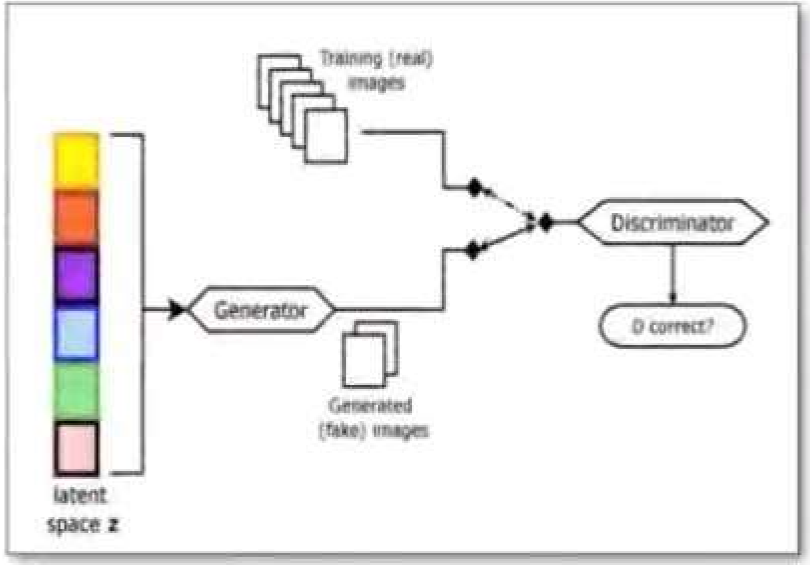
Alpha – Lower bound of Maximizer's value. Perceived value that Maximizer hopes to get with a competitive Minimizer

Beta – Upper Bound of Minimizer's value.



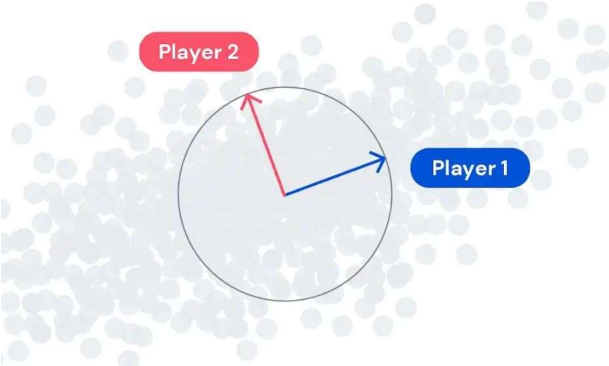
Game Playing
(Interesting Case Studies)

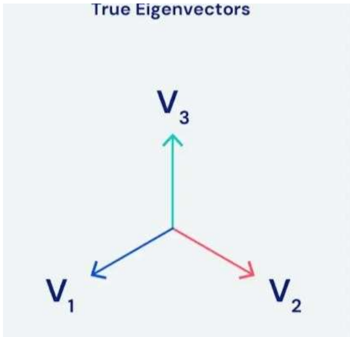
Games in Image Processing

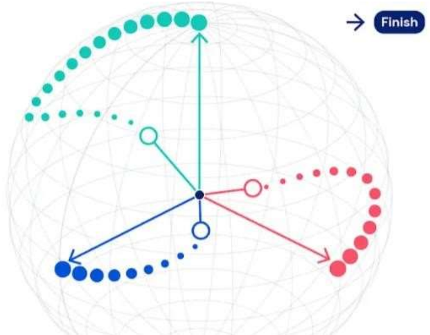


Source Credit:
2019 - Analyzing and Improving the Image Quality of StyleGAN
Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, Timo Aila
https://thispersondoesnotexist.com/

Games in Feature Engineering







Source Credit:
https://deepmind.com/blog/article/EigenGame
2021 - EigenGame: PCA as a Nash Equilibrium , Ian Gemp, Brian McWilliams, Claire Vernade, Thore Graepel

Games in Feature Engineering

$$\texttt{Utility}(v_i \mid v_{j<i}) = \boxed{\texttt{Var}(v_i)} - \sum_{j<i} \boxed{\texttt{Align}}(v_i, v_j)$$

Source Credit:
https://deepmind.com/blog/article/EigenGame
2021 - EigenGame: PCA as a Nash Equilibrium , Ian Gemp, Brian McWilliams, Claire Vernade, Thore Graepel

Gaming – Sample Question

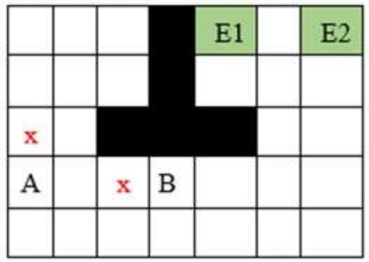
Two Robots A and B competes to leave the maze through either of exits: E1 and E2, as shown in the diagram. At each time step, each Robot move to an adjacent free square. Robots are not allowed to enter squares that other robots are moving into. The same exit cannot be used by both robots at once, but either robot may use either exit. A poisonous gas is left behind when a robot moves. No robot may enter the poisonous square for the duration of the gas's 1-time-step presence (ie., If the square is left free for one game round, the poison evaporates and is no longer dangerous). The poisonous squares are represented as ×'s in the diagram. For utility calculation consider the below assumptions.

H(n) = (Max player's ability to win in the board position) – (Min player's ability to win in the board position)

Note: *Player "Z's" ability to win is given by below:*

Utility = Minimum (manhanttanDistance(Player "Z", Exit E1), manhanttanDistance(Player "Z", Exit E2)) + Penalty

Penalty = Number of unsafe cells (blockage+ poisonous squares) with 4 degree of freedom(Up, Down, Right, Left) in the immediate neighborhood of the Player "Z's" position.

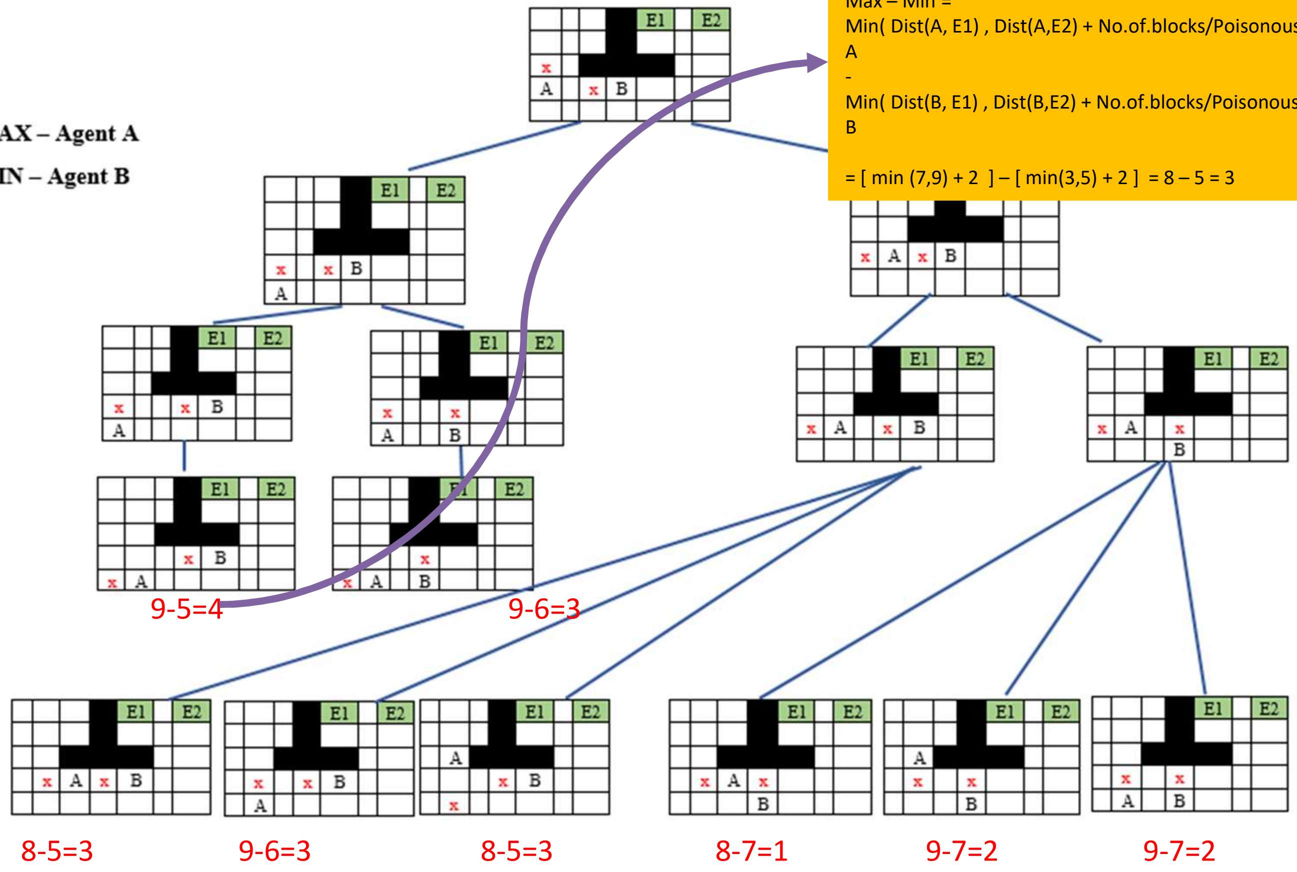


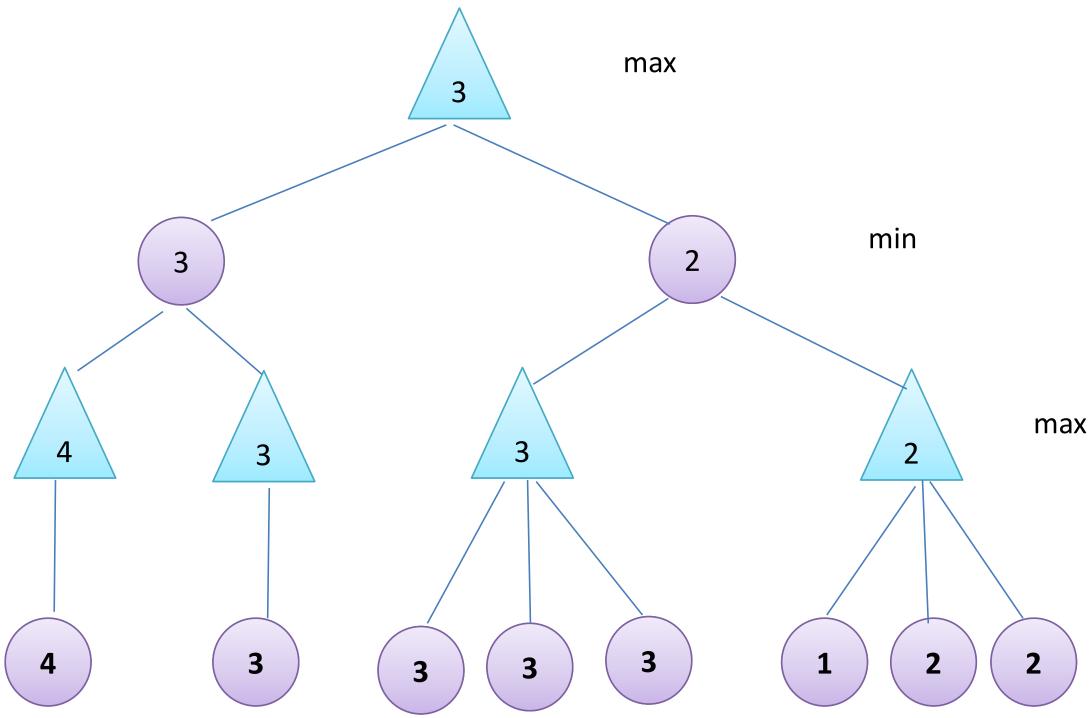
MAX – Agent A

MIN – Agent B

Max – Min =
Min(Dist(A, E1) , Dist(A,E2) + No.of.blocks/Poisonous cells around A
-
Min(Dist(B, E1) , Dist(B,E2) + No.of.blocks/Poisonous cells around B

= [min (7,9) + 2] – [min(3,5) + 2] = 8 – 5 = 3





max
3
min
3
2
max
4
3
3
2
4
3
3
3
3
1
2
2

Required Reading: AIMA - Chapter # 5.1, 5.2, 5.3

Thank You for all your Attention

Note : Some of the slides are adopted from AIMA TB materials