Artificial & Computational Intelligence

AIMLCZG557

Contributors & Designers of document content : Cluster Course Faculty Team

M3 : Game Playing

Presented by
Faculty Name
BITS Email ID

BITS Pilani
Pilani Campus

Artificial and Computational Intelligence

Disclaimer and Acknowledgement

- Few content for these slides may have been obtained from prescribed books and various other source on the Internet
- I hereby acknowledge all the contributors for their material and inputs and gratefully acknowledge people others who made their course materials freely available online.
- .I have provided source information wherever necessary
- This is not a full fledged reading materials. Students are requested to refer to the textbook w.r.t detailed content of the presentation deck that is expected to be shared over e-learning portal - taxilla.
- I have added and modified the content to suit the requirements of the class dynamics & live session's lecture delivery flow for presentation

- **Slide Source / Preparation / Review:**
- From BITS Pilani WILP**:** Prof.Raja vadhana, Prof. Indumathi, Prof.Sangeetha
- From BITS Oncampus & External : Mr.Santosh GSK

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. Min-max Algorithm

B. Alpha-Beta Pruning

C. Making imperfect real time decisions

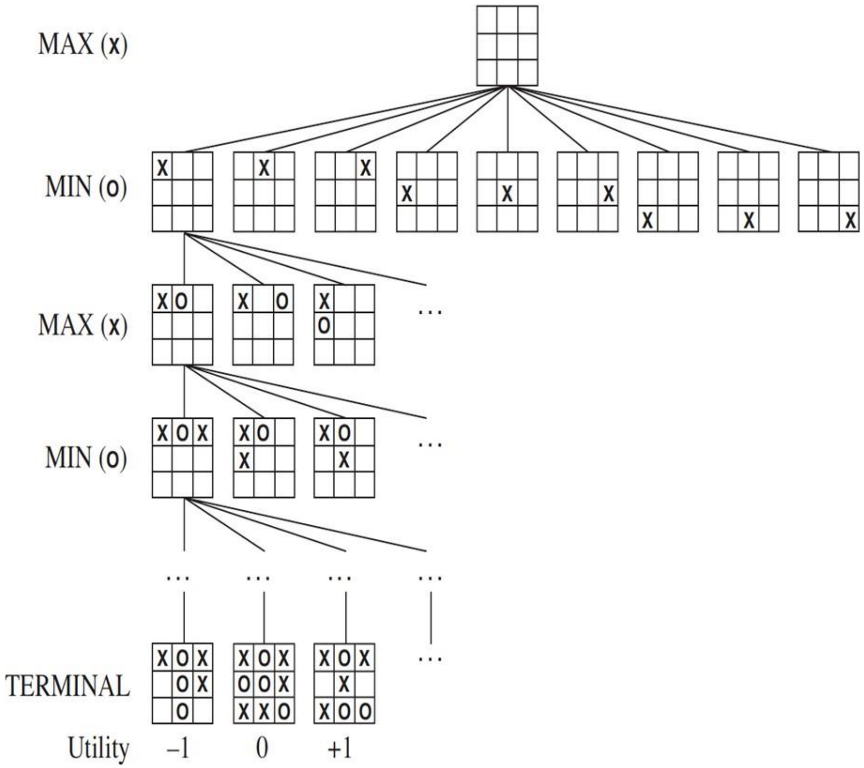
Learning Objective

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 (4.5)

Observable

Discrete (4.4)

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.

<u>AI for Gaming:</u>

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

<u>Characteristics of Games:</u>

- Observability
- Stochasticity
- Time granularity
- Number of players

<u>Adversarial Games:</u>

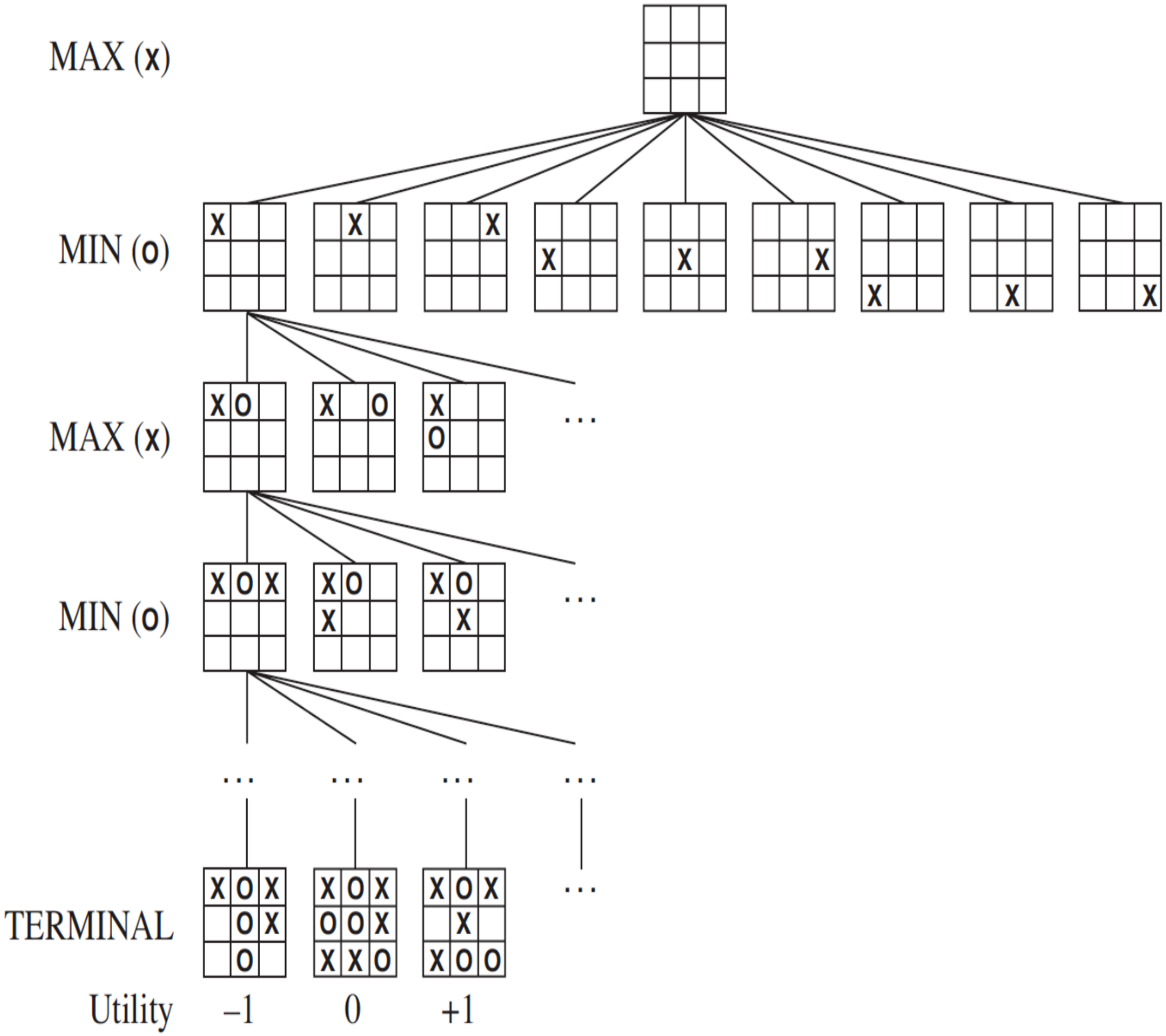
Goals of agents are in conflict where one's optimized step would reduce the utility value of the other.

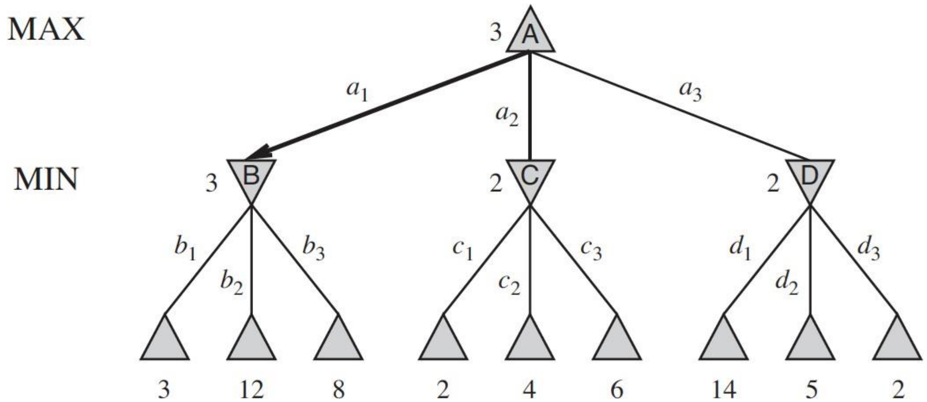
Games as Search Problem

PSA : Representation of Game:

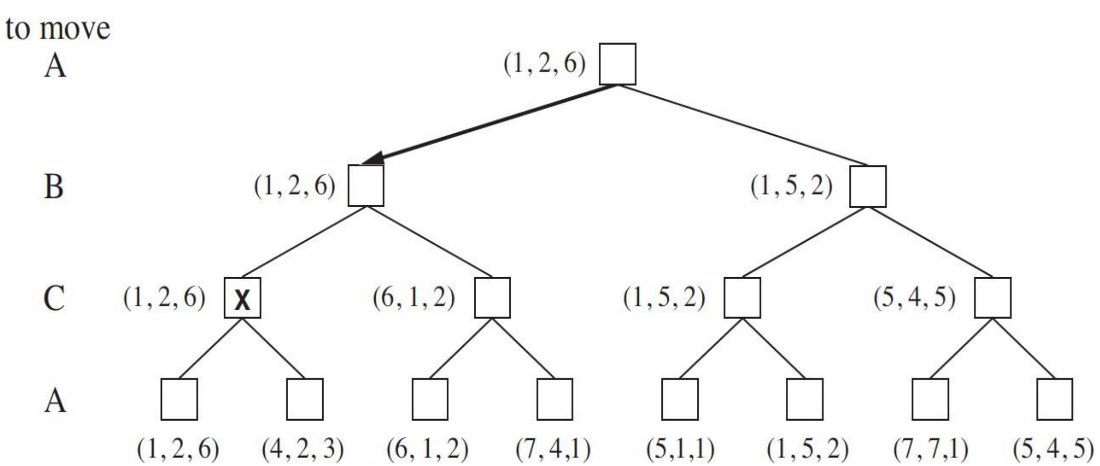
INITIAL STATE: S0
PLAYER(s)
ACTIONS(s)
RESULT(s, a)
TERMINAL-TEST(s)
UTILITY(s, p)

Eg., Tic Tac Toe





Two Player Game : 1-Ply Game



Multiplayer Game

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

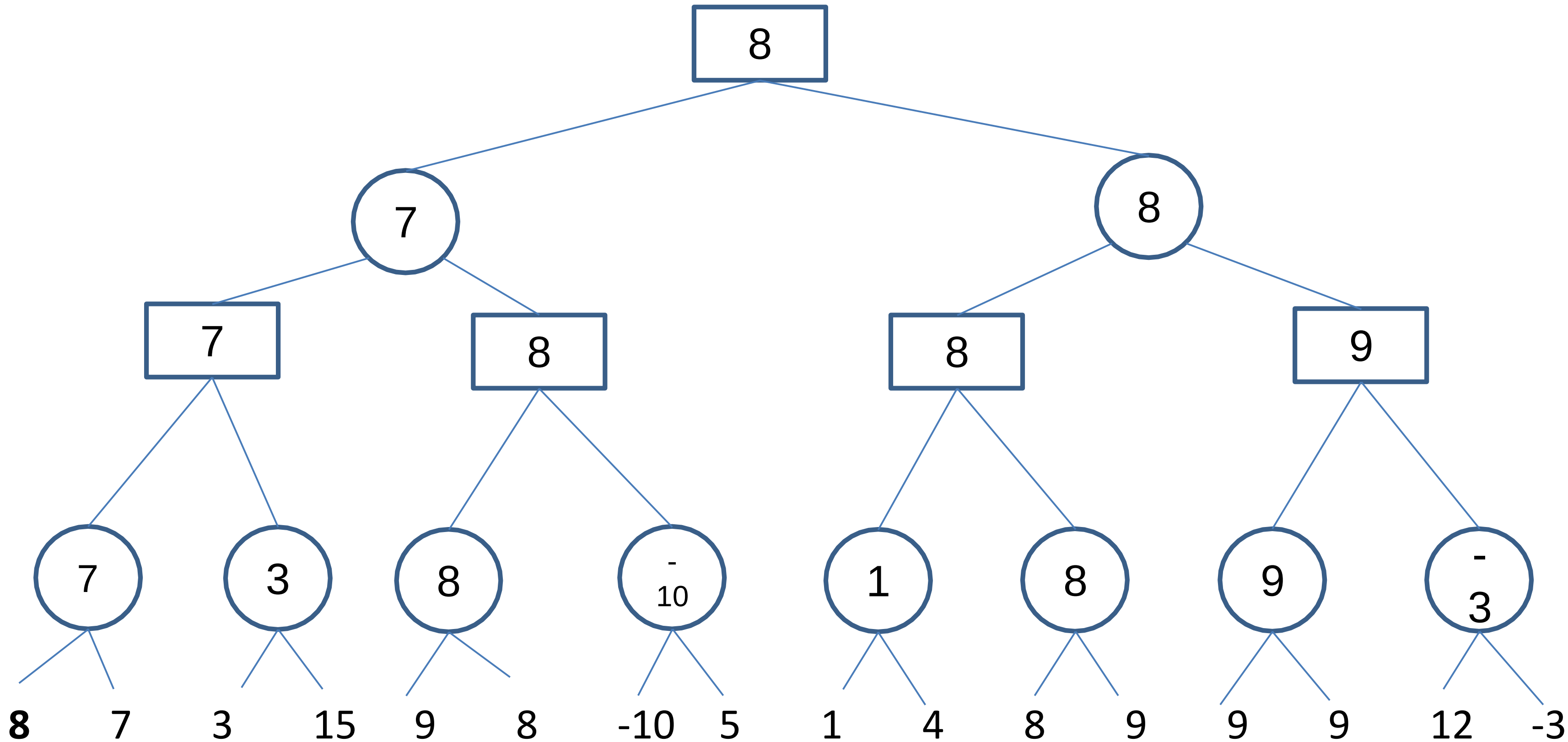
---

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

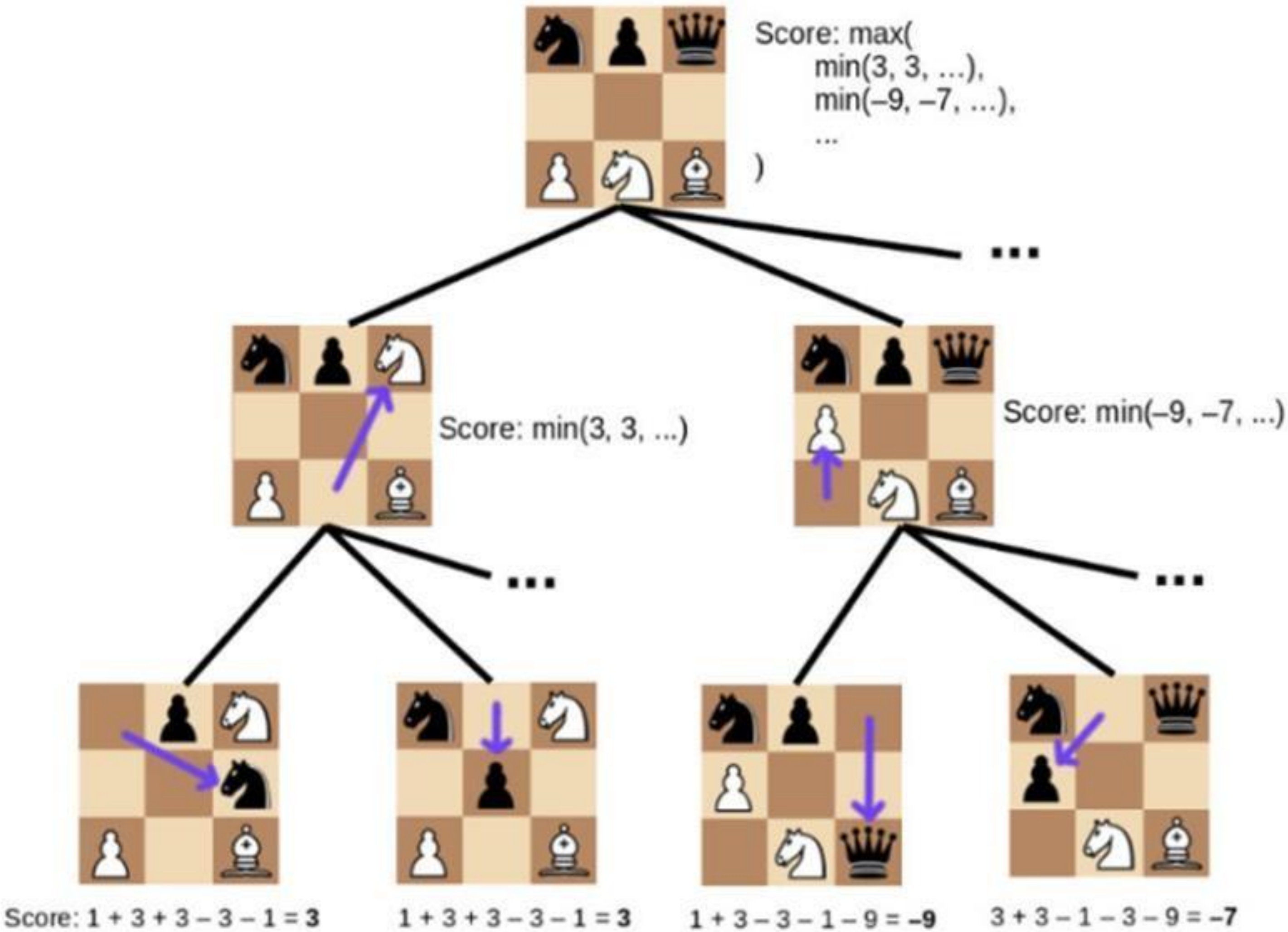
Min-Max Algorithm – Example -1

Squares represent MAX nodes

Circles represent MIN nodes

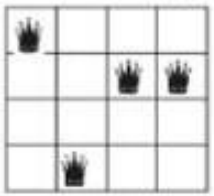


Design of Static Evaluation Values



Design of Static Evaluation Values

N-Queens



1	4	2	2	4

Tic-Tac-Toe

0	0	x
x		0
		x

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

N-Tile

2	8	3
1	6	4
7		5

1	2	3
8		4
7	6	5

No.of.Tiles Out of Place	5

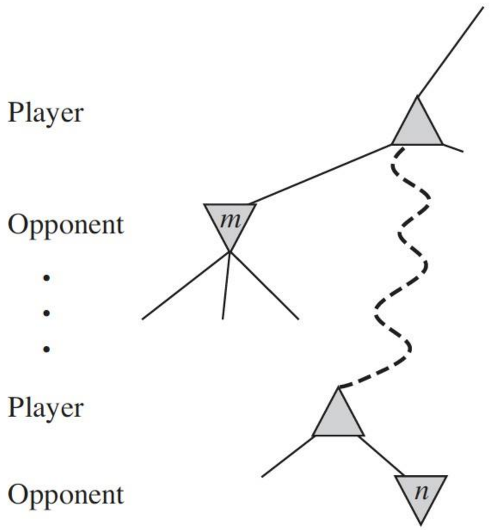
$$Eval(S) = w_1 f_1(s) + w_2 f_2(s) + ... + w_n f_n(s)$$

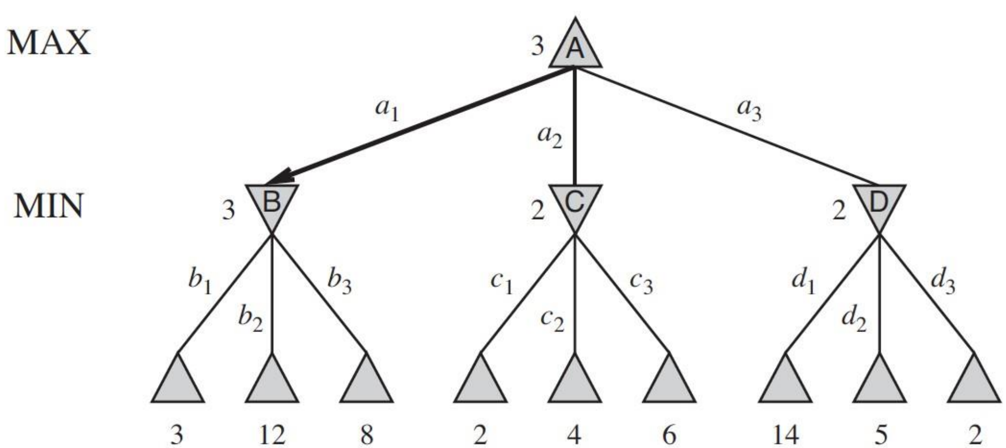
$$= 0.6\,(MaxChance - MinChance) + 0.4\,(MaxPairs - MinPairs)$$

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

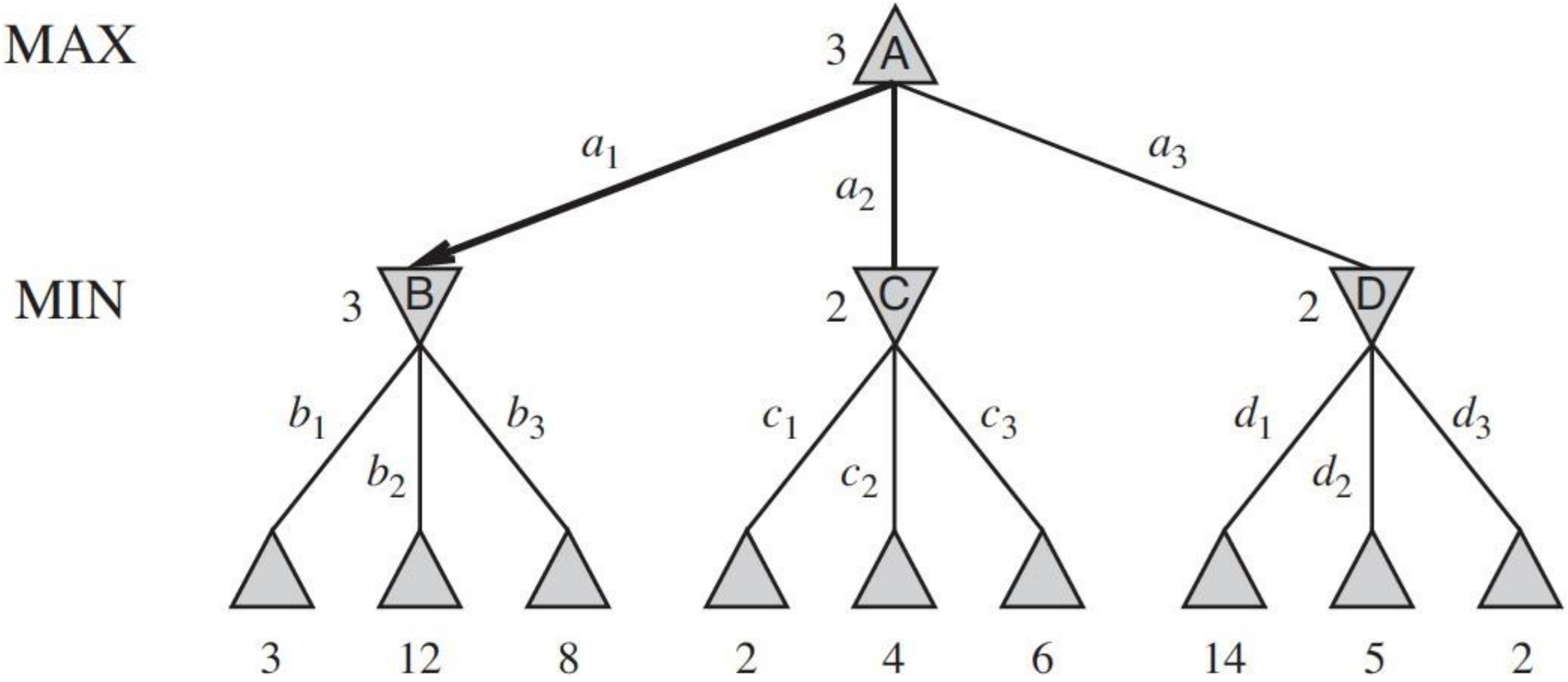




$$
\begin{aligned}
\textsc{Minimax}(root) &= \max(\min(3,12,8), \min(2,x,y), \min(14,5,2)) \\
&= \max(3, \min(2,x,y), 2) \\
&= \max(3, z, 2) \qquad \text{where } z = \min(2,x,y) \le 2 \\
&= 3.
\end{aligned}
$$

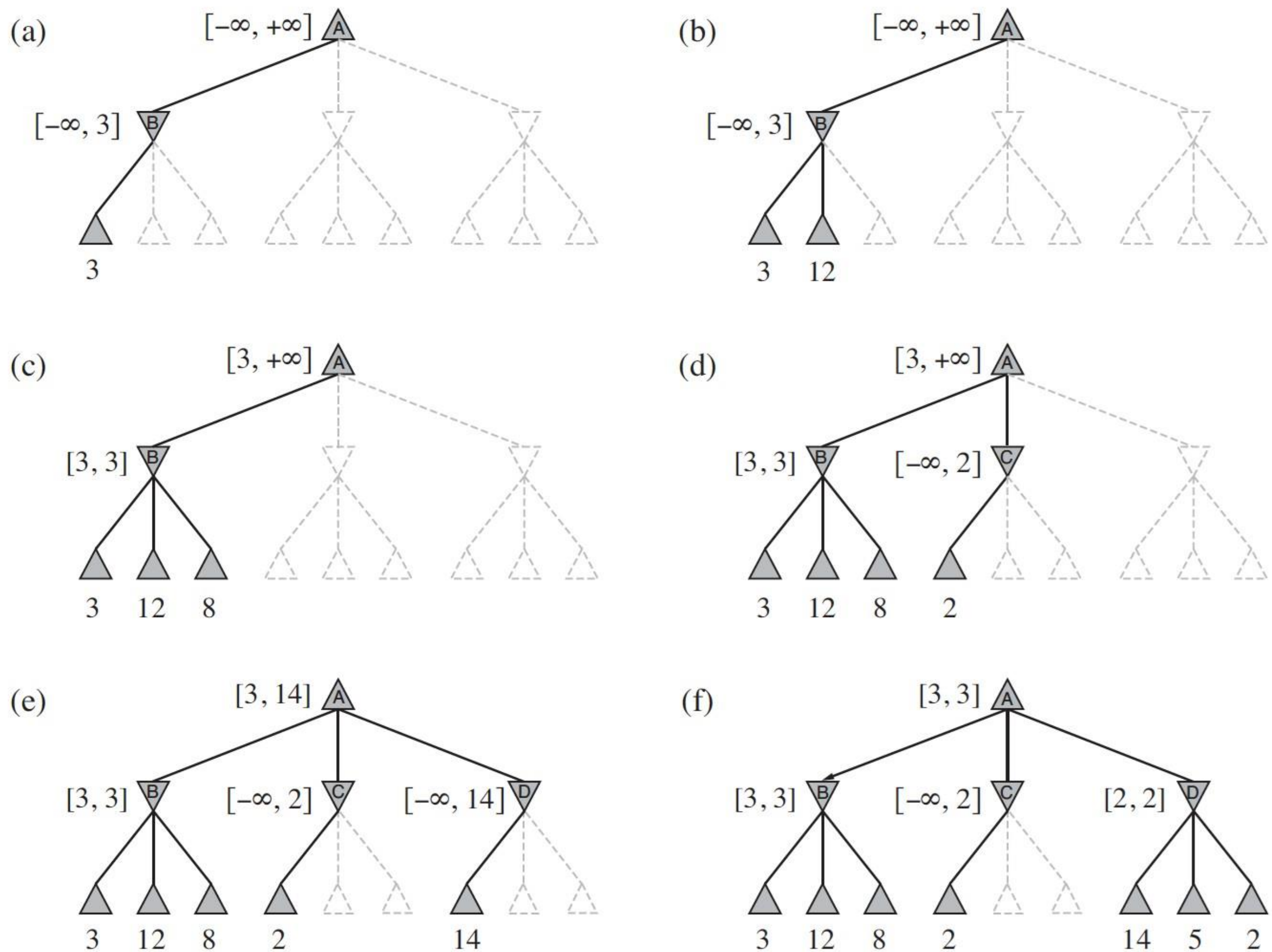
Alpha Beta Pruning

Book Example



Alpha Beta Pruning

Book Example



Min-Max Algorithm

Alpha beta Modifications

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

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

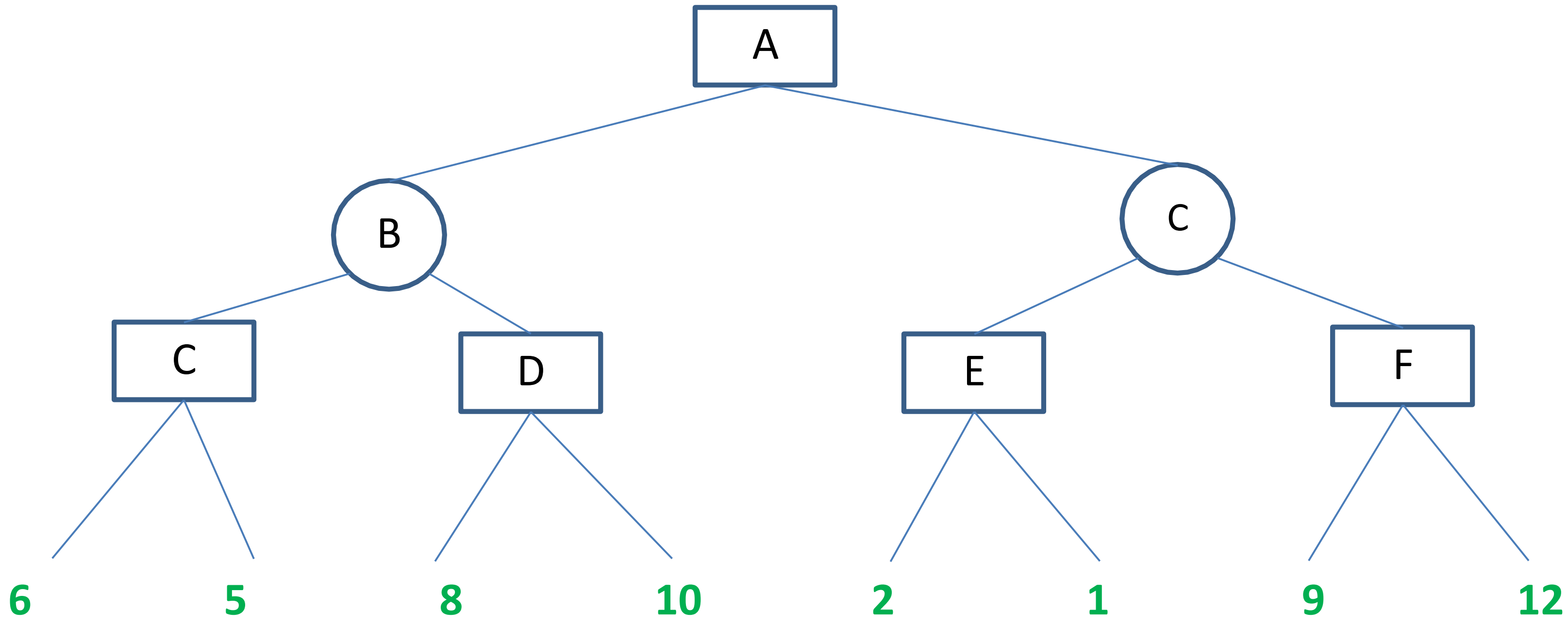
Alpha – beta Pruning

Steps in Alpha – Beta Pruning:

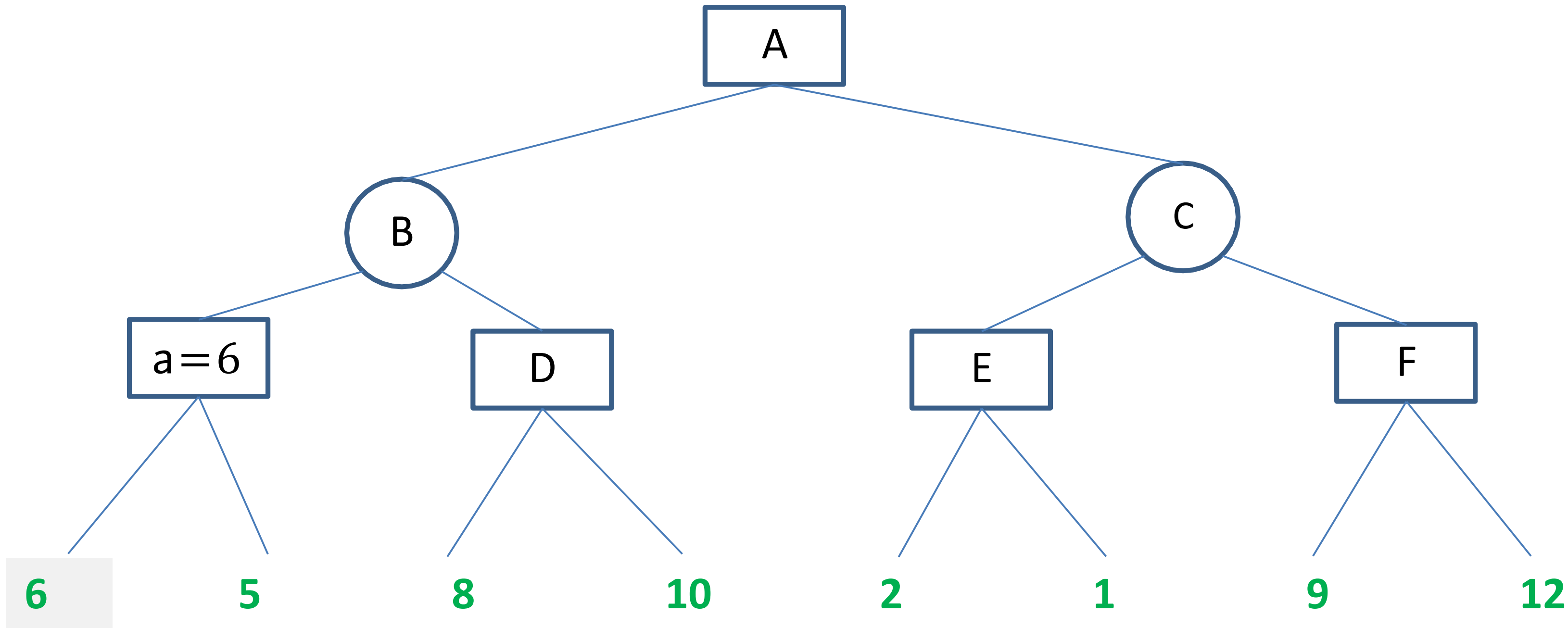
1. At root initialize alpha = -∞ and beta = + ∞. This is to set the worst case boundary to start the algorithm which aims to increase alpha and decrease beta as much as optimally possible
2. Navigate till the depth / limit specified and get the static evaluated numeric value.
3. For every value VAL being 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

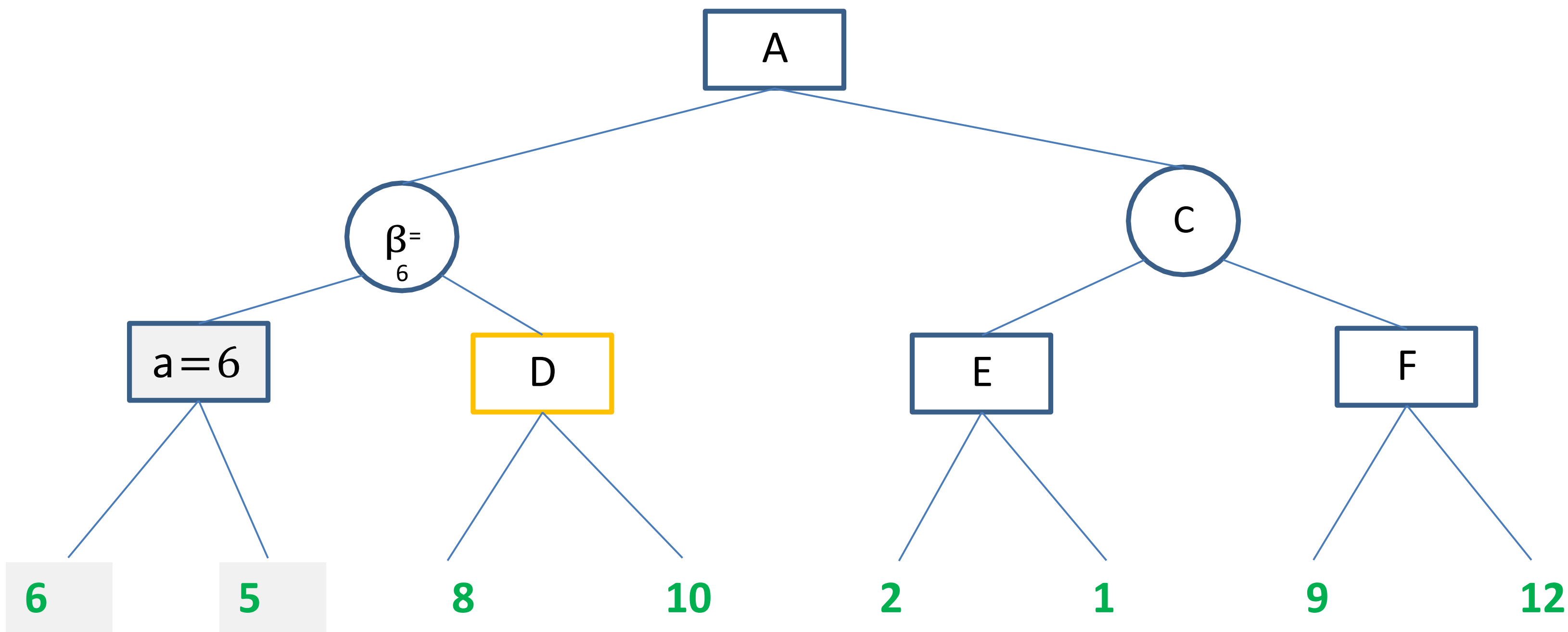


Idea –Pruning



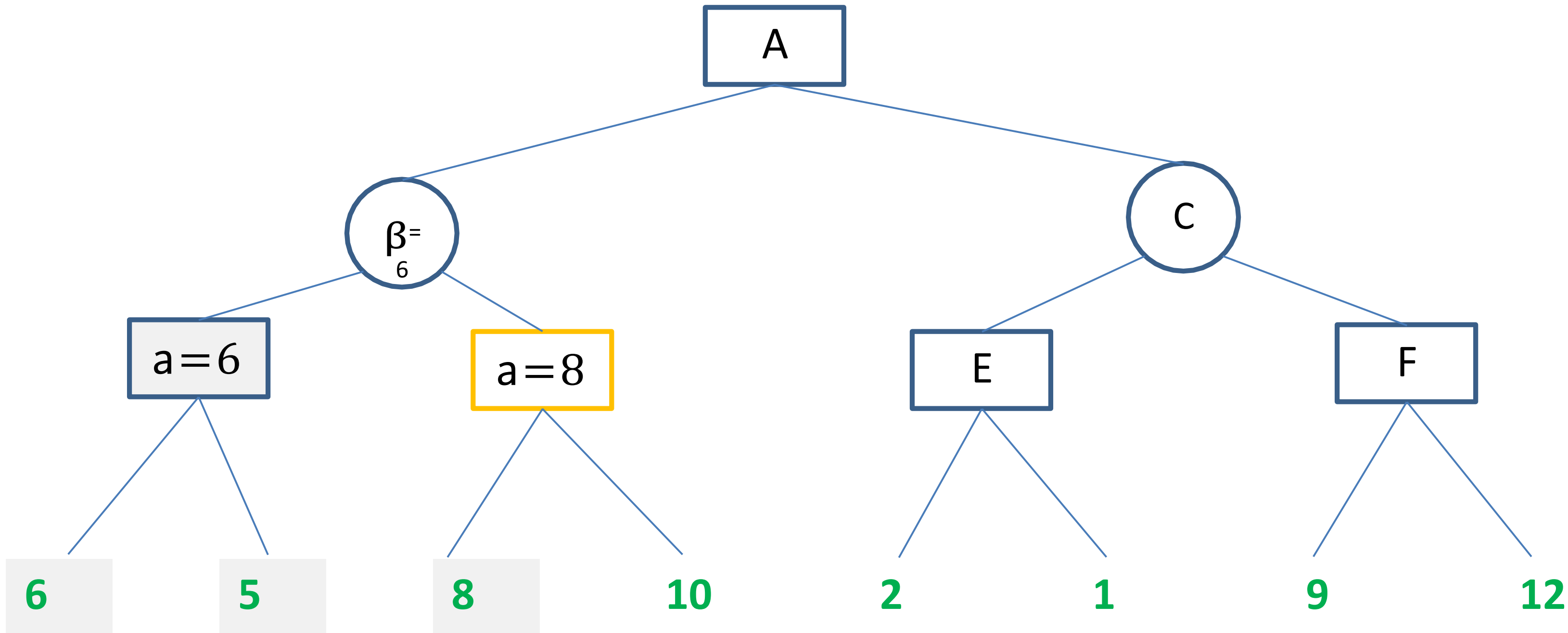
Alpha Beta Pruning

Idea –Pruning

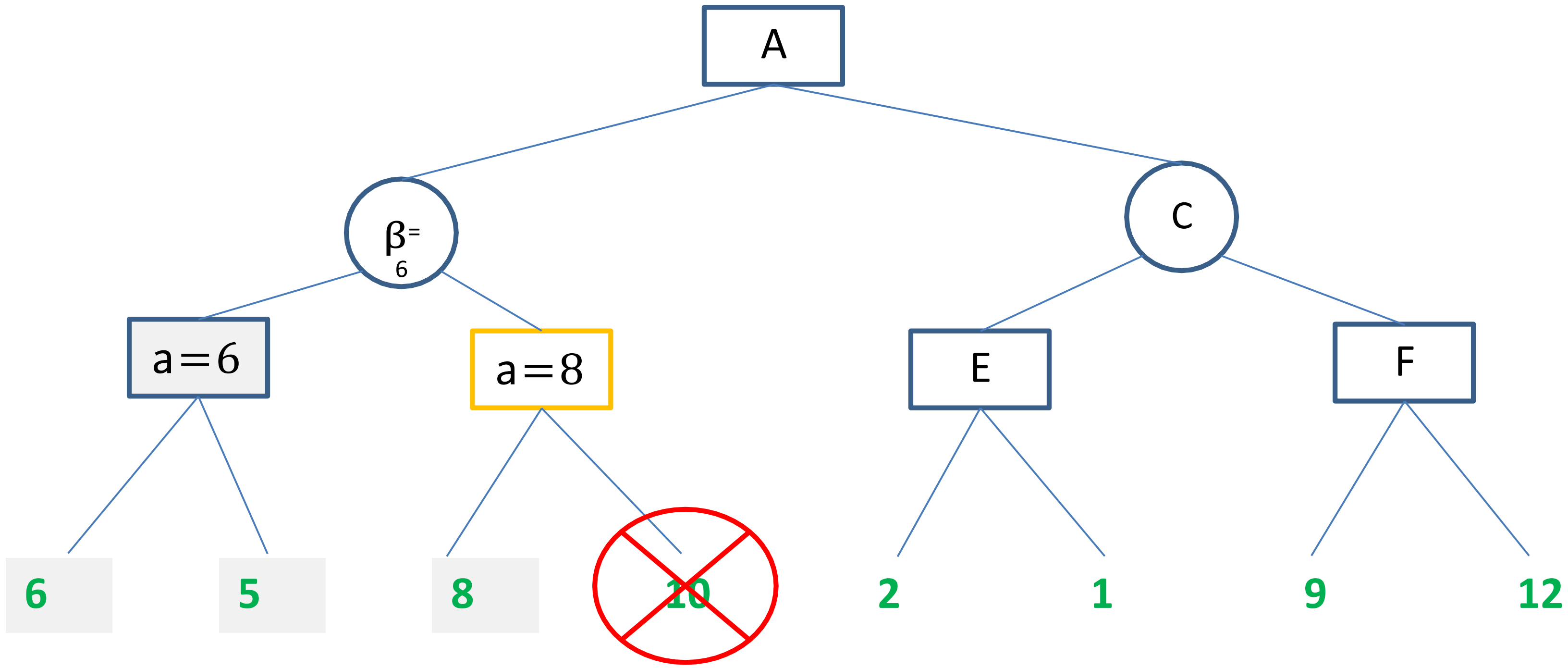


Alpha Beta Pruning

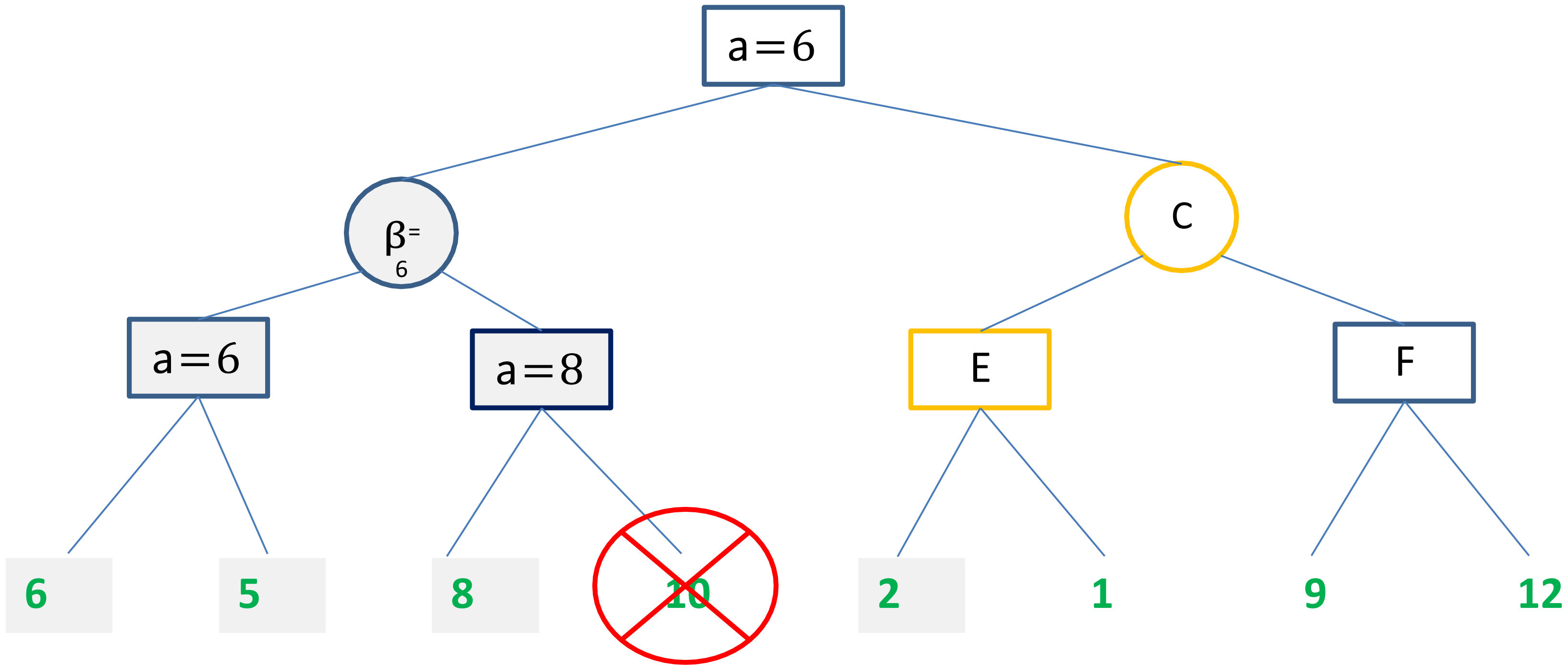
Idea – Alpha Pruning



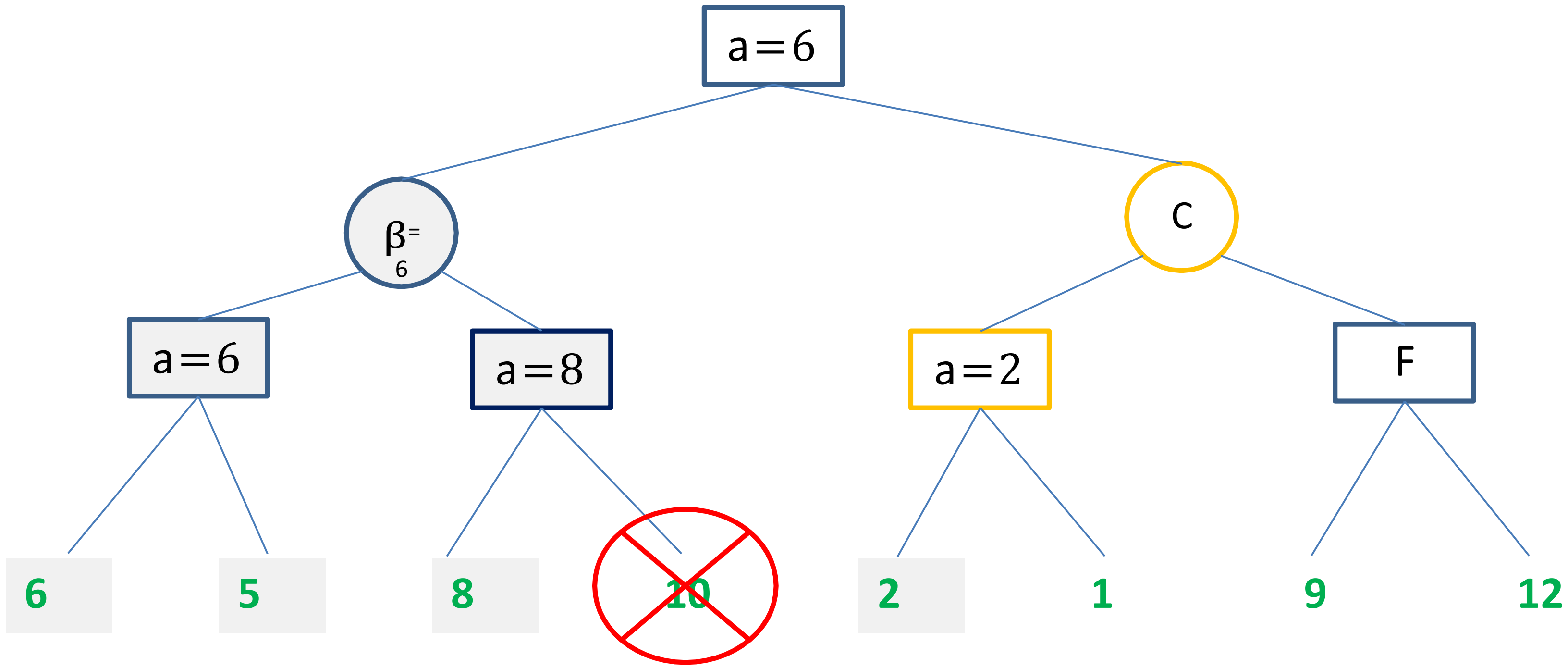
Idea – Beta Pruning



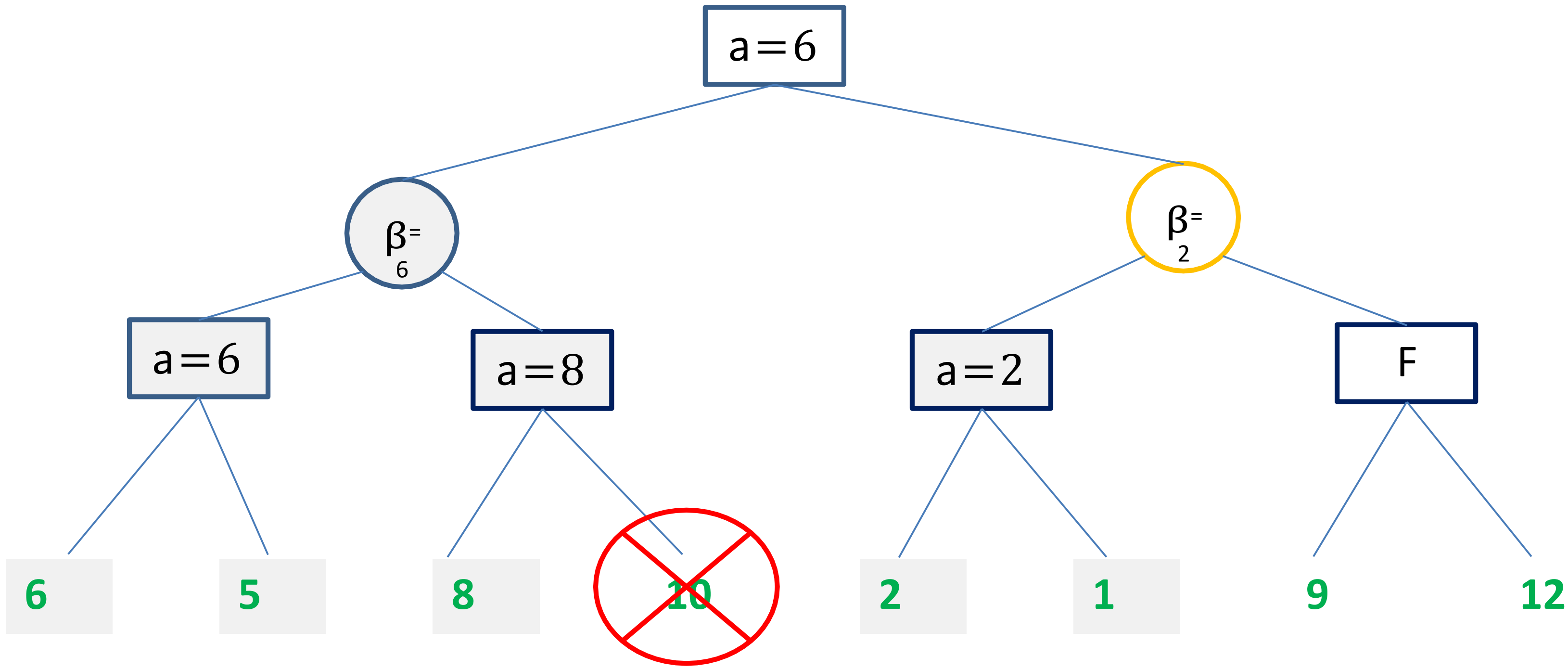
Idea –Pruning



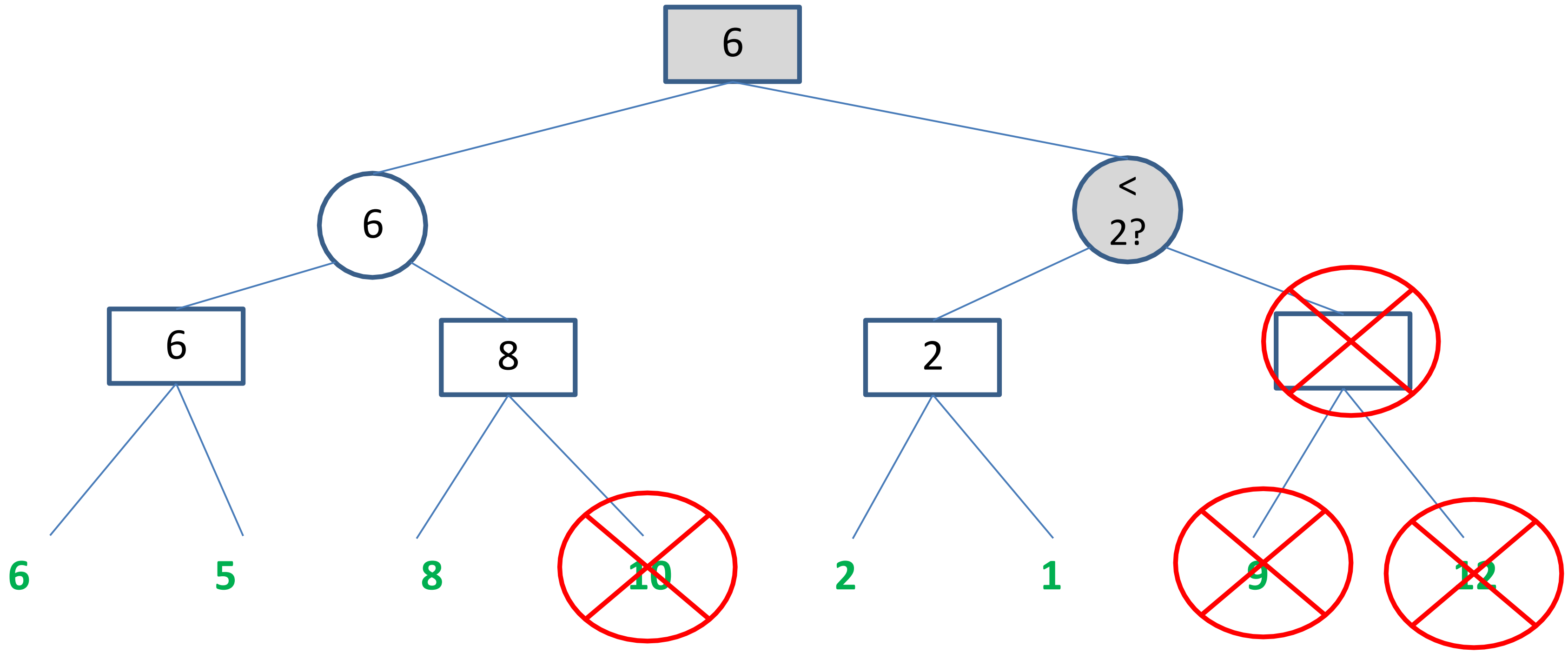
Idea –Pruning



Idea –Pruning



Idea – Alpha Pruning



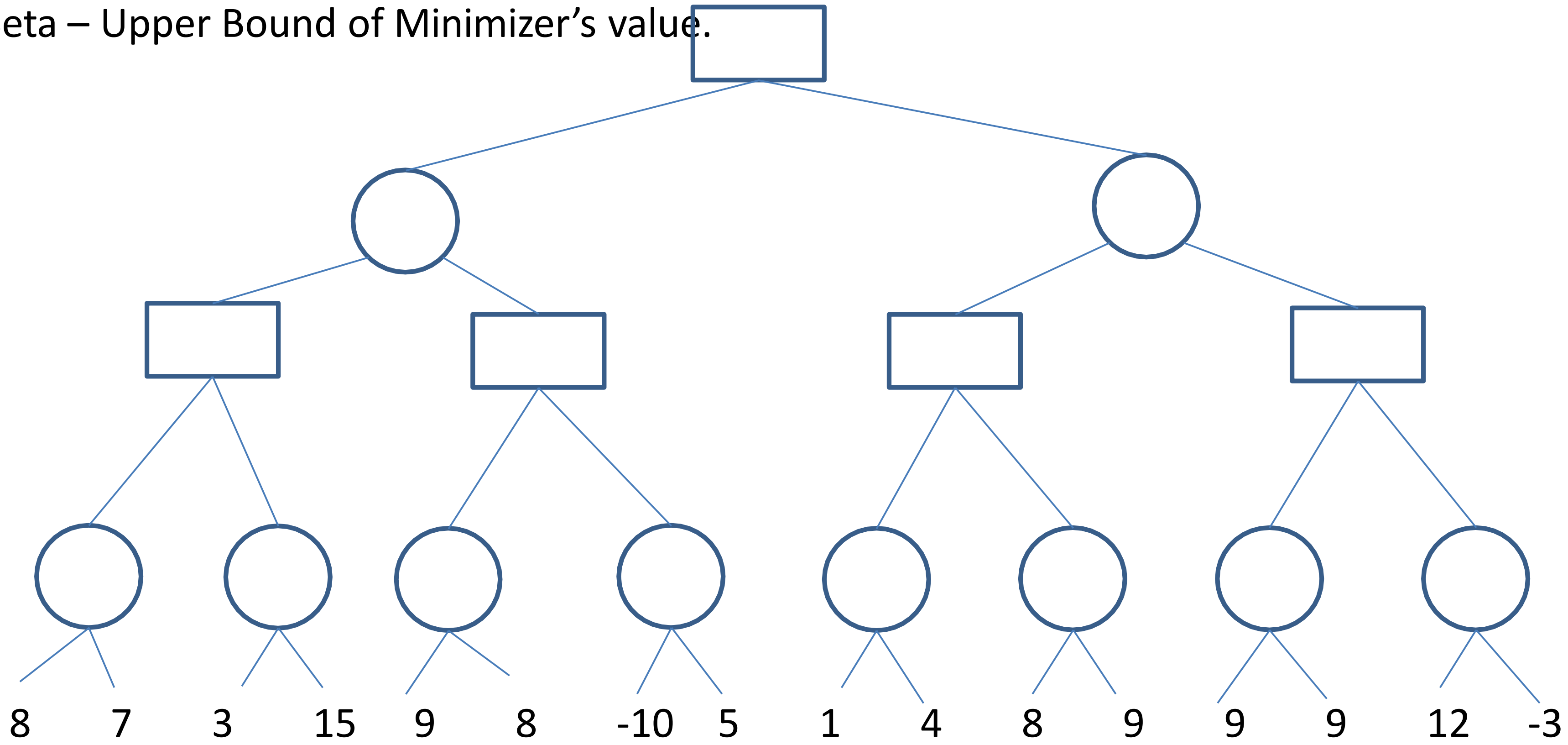
Alpha – Lower bound of Maximizer's value. Perceived value that Maximizer hopes to against a competitive Minimizer

Alpha – beta Pruning – Example -4

Do for practice.

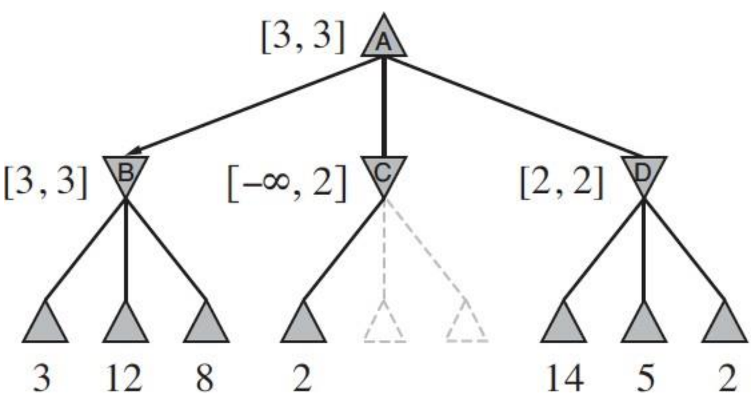
Alpha – Lower bound of Maximizer’s value. Perceived value that Maximizer hopes to get with a competitive Minimizer

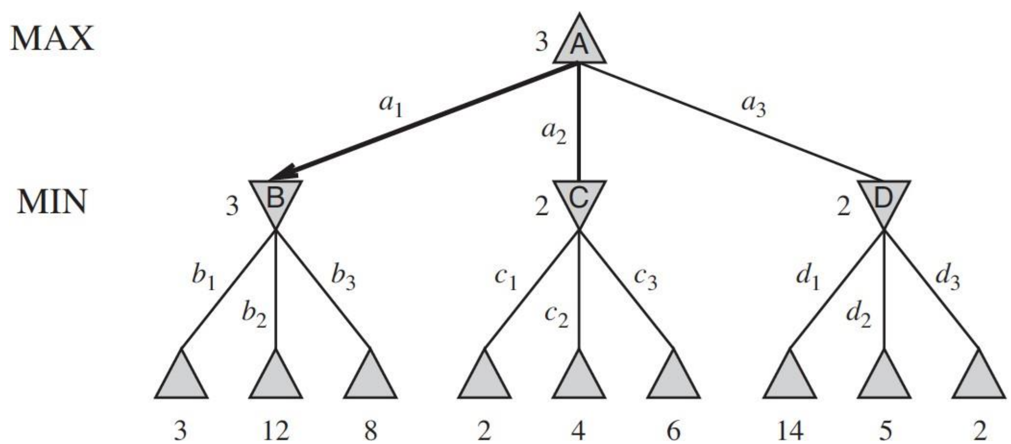
Beta – Upper Bound of Minimizer’s value.



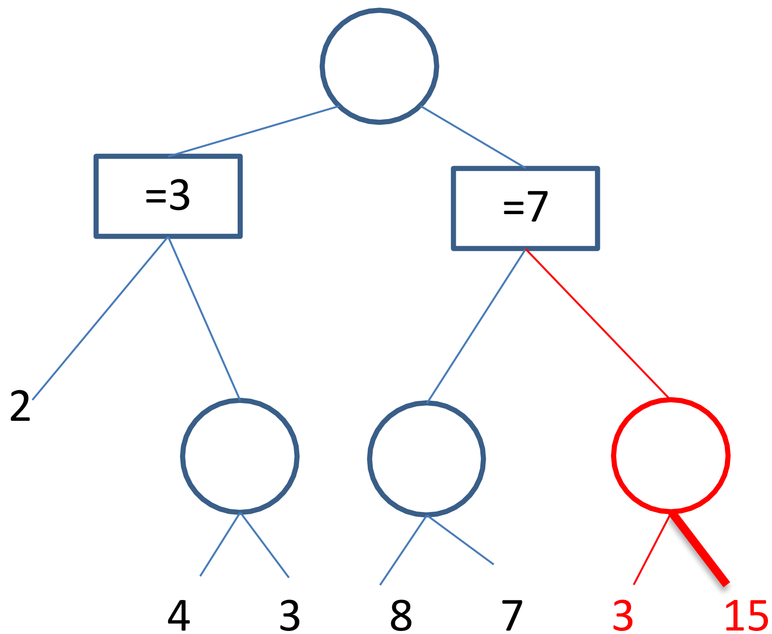
Computational Efficiency

How to reduce the move generations better along while doing Alpha-Beta Pruning?

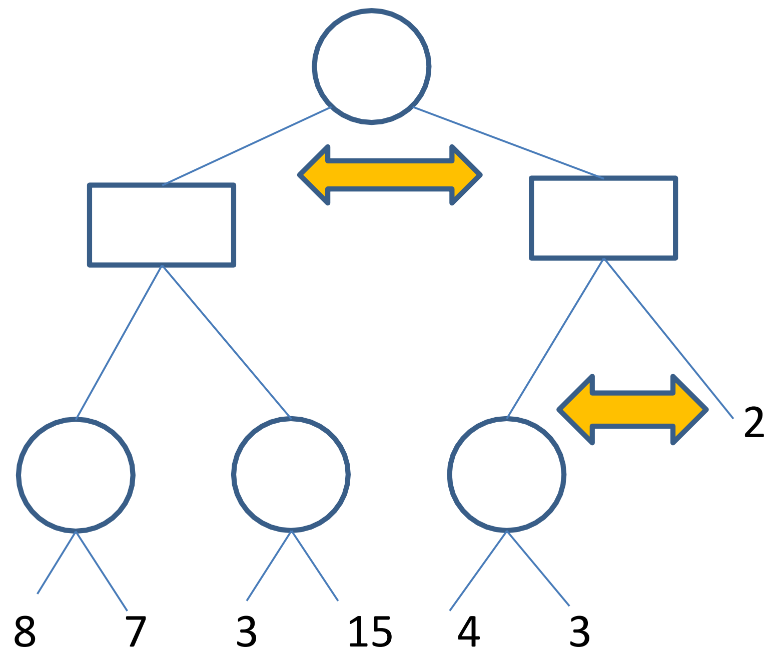




After Move Ordering



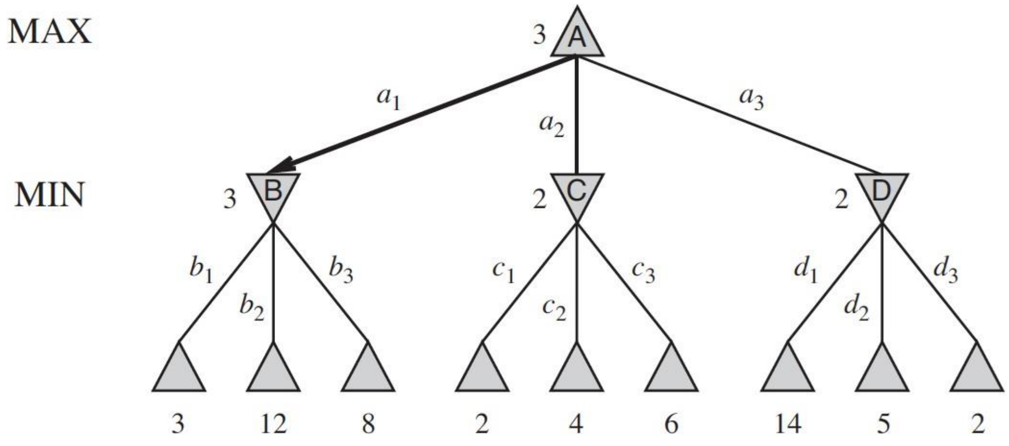
Before Move Ordering



Gaming
(Imperfect Decisions)

Computational Efficiency

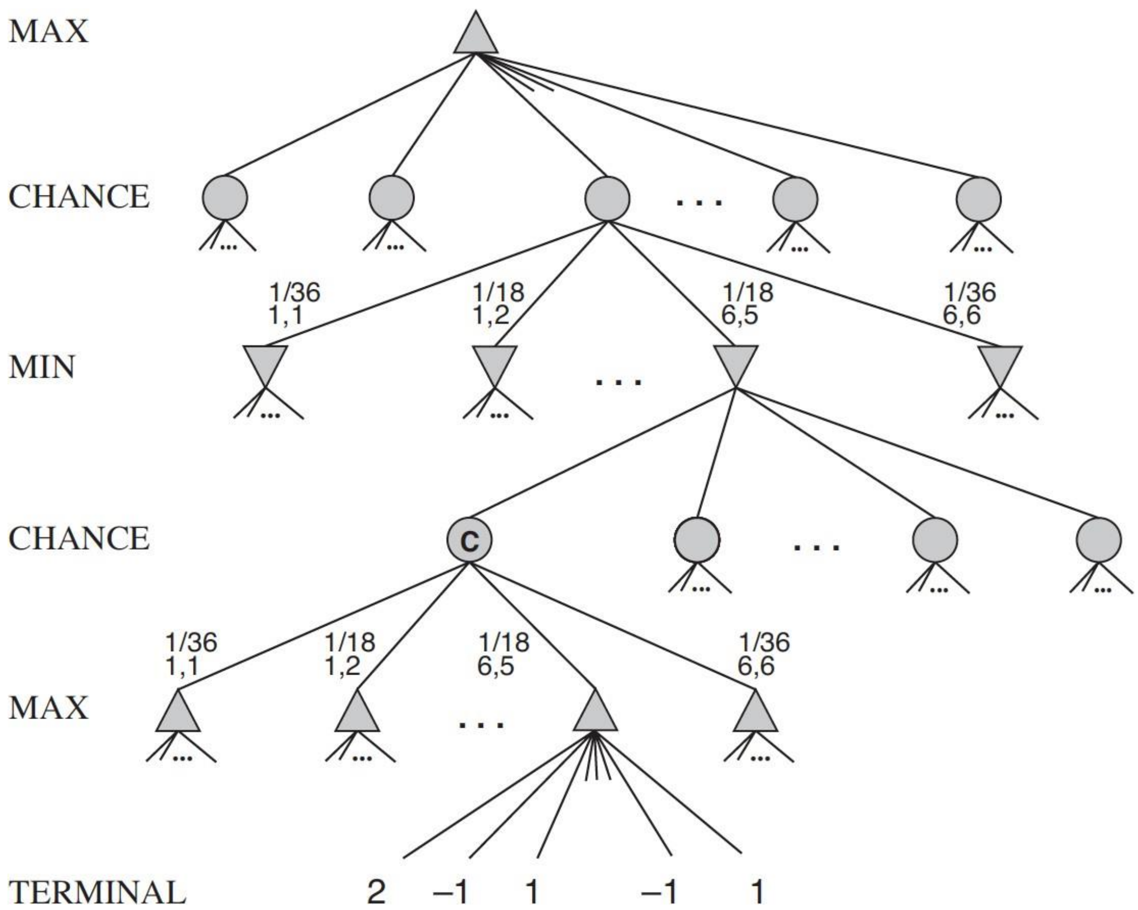
How games can be designed to handle imperfect decisions in real-time?



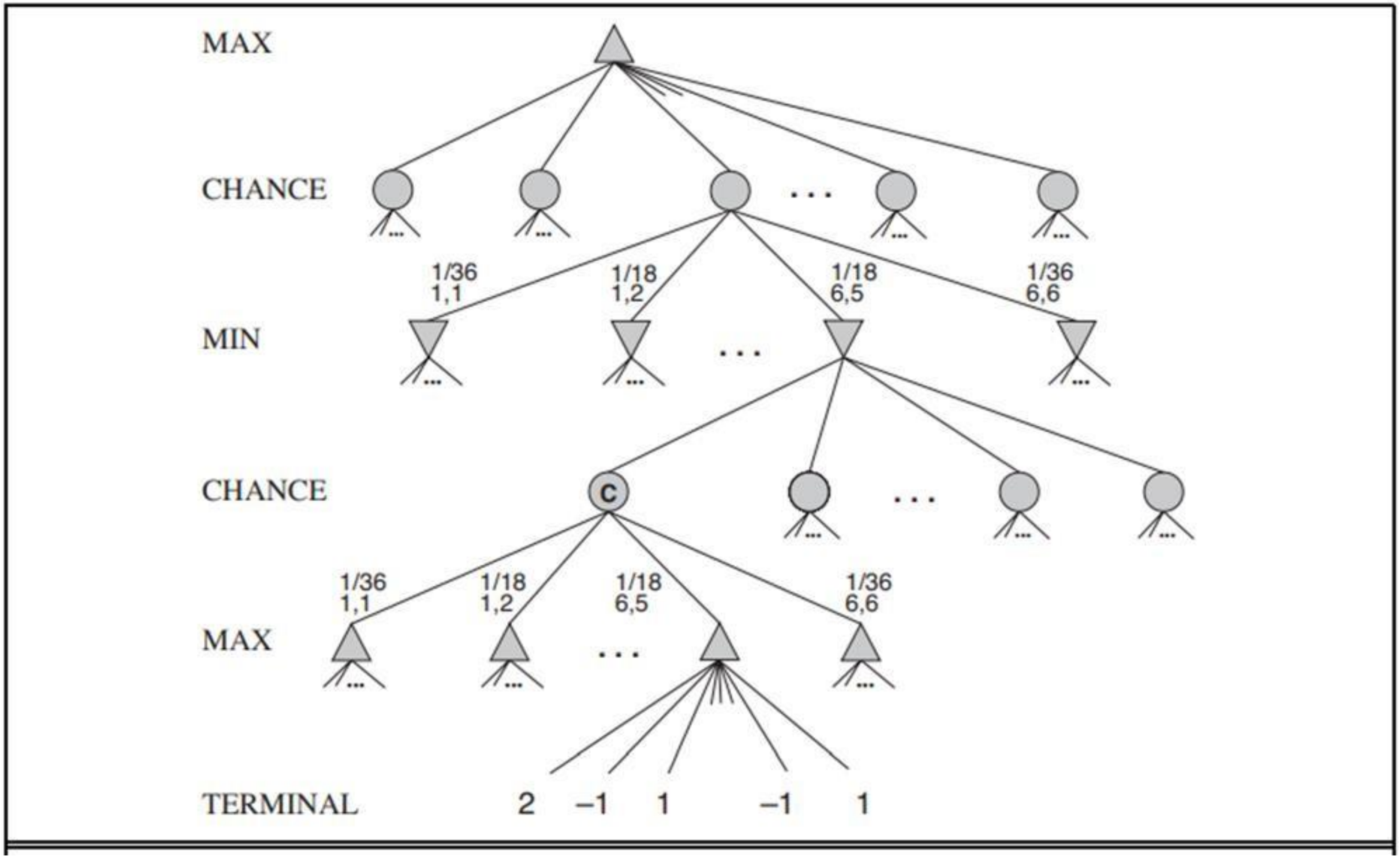
Computational Efficiency

<u>Idea : Chance Node:</u>

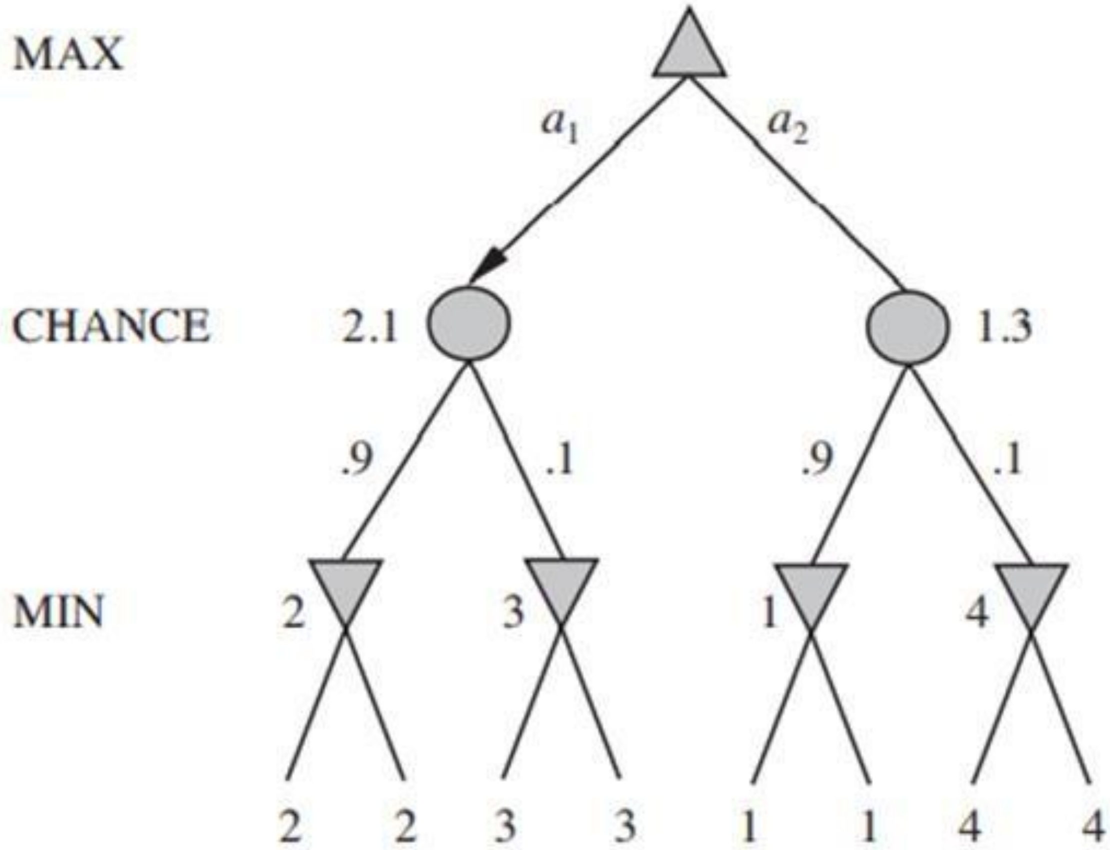
Holds the expected values that are computed as a sum of all outcomes weighted by their probability (of dice roll)

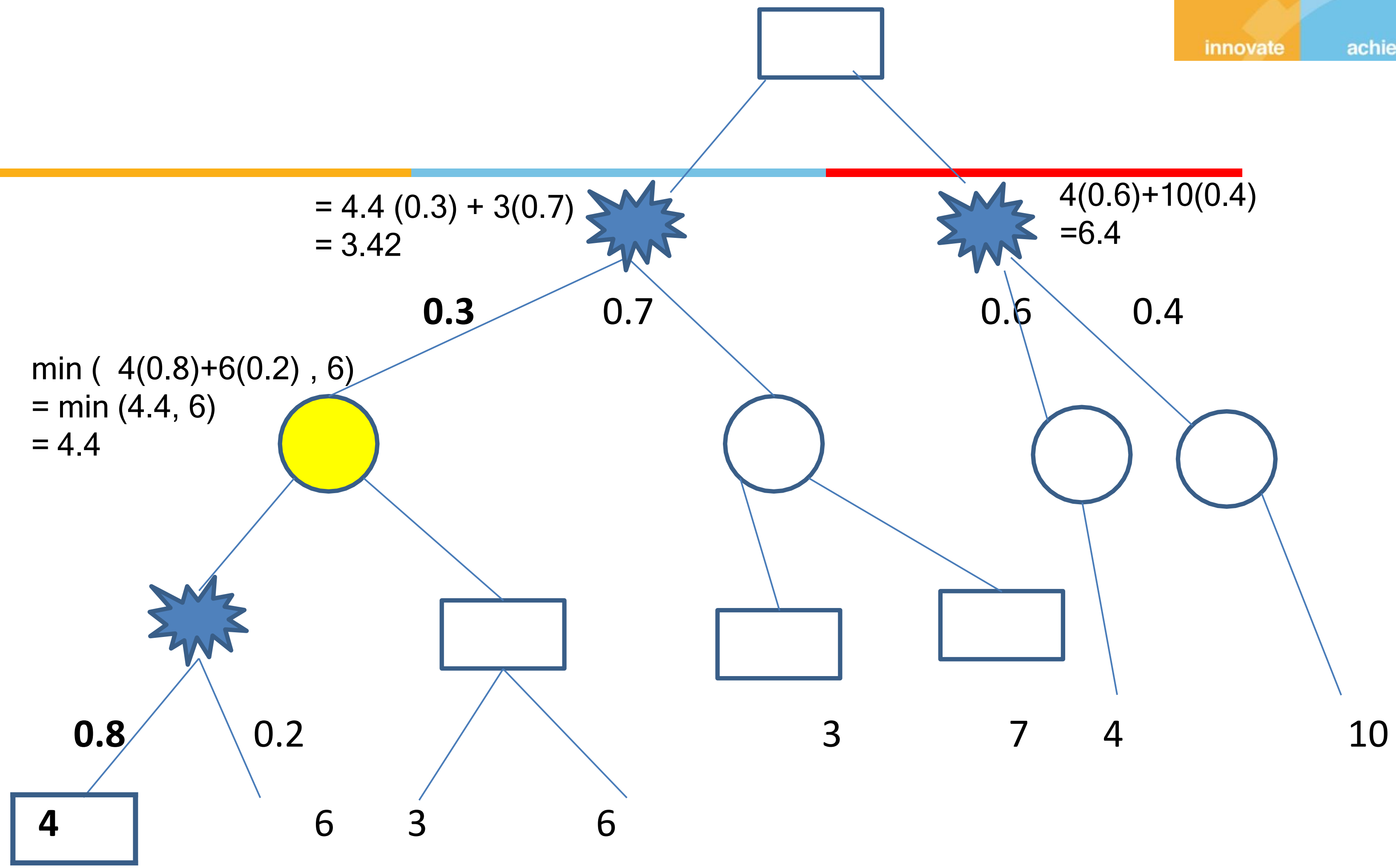


Expecti Mini Max Algorithm



Expecti Mini Max Algorithm

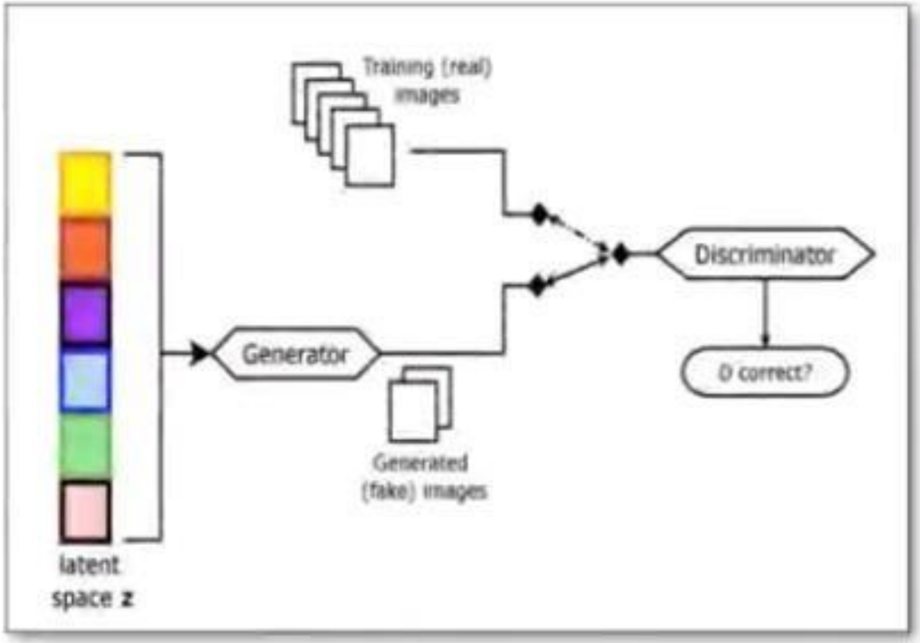




= 4.4 (0.3) + 3(0.7)
= 3.42
4(0.6)+10(0.4)
=6.4
0.3
0.7
0.6
0.4
min (4(0.8)+6(0.2) , 6)
= min (4.4, 6)
= 4.4
0.8
0.2
3
7
4
10
4
6
3
6

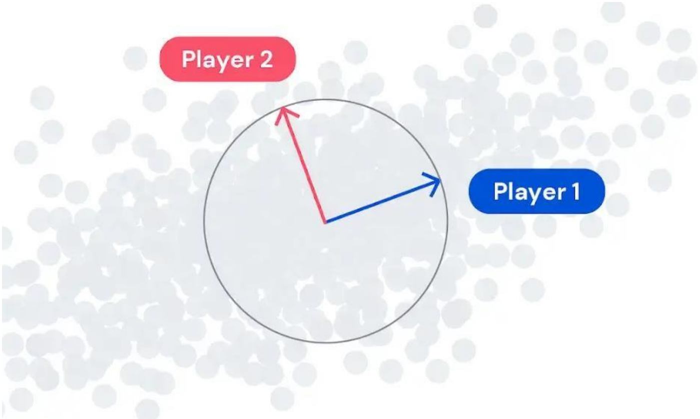
Game Playing
(Interesting Case Studies)

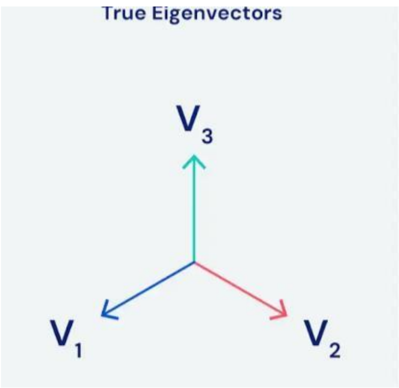
Games in Image Processing

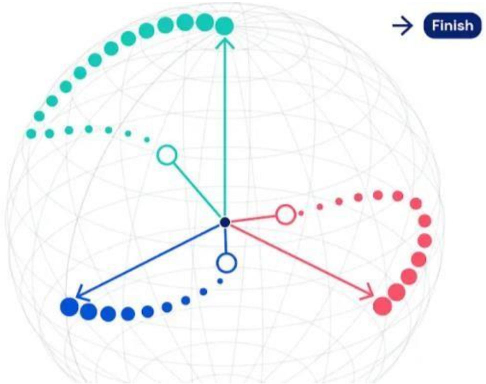


Source Credit:
[2019 - Analyzing and Improving the Image Quality of StyleGAN](#)
[Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, Timo Aila](#)
https://thispersondoesnotexist.com/

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]

Required Reading: AIMA - Chapter # 4.1, #4.2, #5.1

Thank You for all your Attention

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