Artificial & Computational Intelligence

AIML CLZG557

M2 : ACO & M3 : Game Playing

V Indumathi

Guest Faculty,

BITS - WILP

BITS Pilani

Pilani Campus

ACO Pseudocode and notations

General pseudo-code

Procedure *ACO*

Schedule Activities

Initialization

Construction

Update Pheromone

Daemon Actions {optional}

// local search, elitism

End schedule activities

End *ACO*

Parameters used in ACO

Parameter	Description
N	Total No of ants ; N>1
τ_o	Initial pheromone amount
τ_{ij}	Amount of pheromone deposited while traversing from i to j
η_{ij}	Cost of link (i,j)
α	Importance coefficient of pheromone intensity
β	Importance coefficient of route cost
ρ	evaporation co-efficient; $0<\rho<1$
$visit_k$	Visited nodes table of k^{th} ant
Q	Importance - Constant value pertaining to pheromone trail
f_k	Route cost obtained by ant k

69

ACO Steps

1 **Initialization**

Place predefined number of ants on starting point

Set values for parameters α, β, ρ.

Set τ_0 to 0. τ

Construction

Compute the next node transition probability

Path

$(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}$

τ

4-1
4-2
4-3
4-5

$$NTP_{ij} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{h \notin visit_k}(\tau_{ih})^{\alpha}(\eta_{ih})^{\beta}}$$

NTP

loc 2

$P_1 \rightarrow$ Prob 1

loc 1

$P_2 \rightarrow$ Prob 2

loc 3 $P_3 \rightarrow$ Prob 3

loc 4 $P_4 \rightarrow$ Prob 4

max prob

loc 1 $\rightarrow$ loc 2

1 2 $\rightarrow$

1 3 $\rightarrow$

1 4

ACO Steps

Pheromone updation:

Pheromone reinforcement & pheromone evaporation

Direct impact on the exploitation (enhancing found food path) & exploration (discovering new path) of ant algorithms

$$\tau_{ij}^{new} = (\rho)\tau_{ij}^{old} + \Delta\tau_{ij}^{k}$$

0

Amount of pheromone deposited on (i,j) by kth at that timestamp is given by

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{\boldsymbol{f_k}} & if\ k^{th} ant\ passes\ i\ and\ j \\ 0 & otherwise \end{cases}$$

$$\boxed{\frac{Q}{f_k}}$$

Stopping criteria: reaching predetermined number of iterations

Problem: Reaching pre-determined number of iterations before reaching destination leading to ant drop

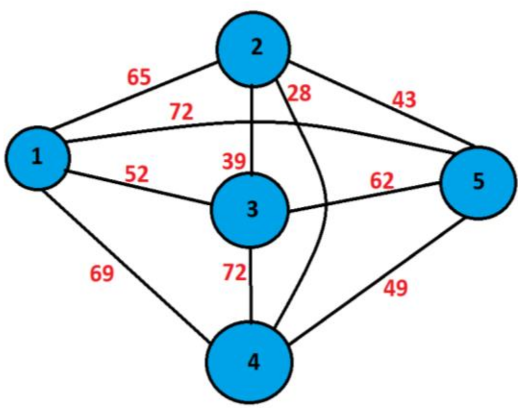
Travelling Salesman Problem

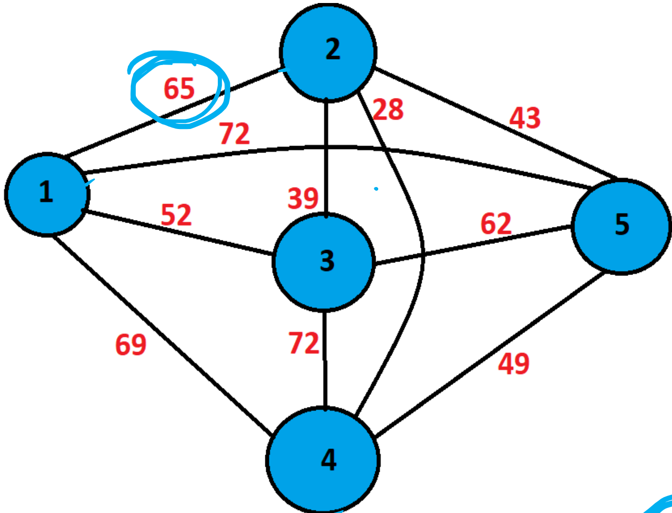
Problem: **Given n cities, the goal is to find shortest path going through all cities and visiting each exactly once**

- Consider a complete graph

- d_{ij} is the route cost over (i,j) $\{f_k\}$

- Each ant builds its own tour from starting city
- Each ant chooses a town to go to with a probability
- Keep tabs on visit list of each ant
- When tour completed, lay pheromone on each edge visited
- Next city *j* after city *i* chosen according to probability rule





Initially No. of ants = No. of cities. 5

start at 4 $\alpha = 0.5$ $\beta = 0.75$ (0–1)

$Q = 100$ $\rho = 0.1$

PM

$\tau = 0$

$\tau_{12} = \tau_{21} = 0.54$

$\tau_{13} = \tau_{31} = 0.53$

$\tau_{14} = \tau_{41} = 0.35$

$\tau_{15} = \tau_{51} = 0.24$

$\tau_{23} = \tau_{32} = 0.53$

$\tau_{24} = \tau_{42} = 0.39$

$\tau_{25} = \tau_{52} = 0.18$

$\tau_{34} = \tau_{43} = 0.32$

$\tau_{35} = \tau_{53} = 0.90$

$\tau_{45} = \tau_{54} = 0.68$

Path matrix

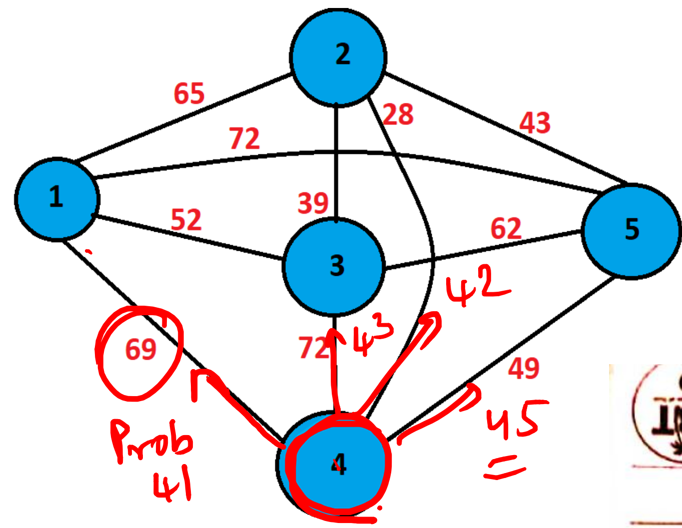
	1	2	3	4	5
1	0	65	52	69	72
2	65	0	39	28	43
3	52	39	0	72	62
4	69	28	72	0	49
5	72	43	62	49	0

12 = 21 ⇒ Same value

$\tau_{12} \Rightarrow \tau_{21}$

$\boxed{2*}$.

$\tau_{12}, \tau_{21} \Rightarrow$ Same



Initially No. of ants = No. of cities.

Start at 4. $\alpha = 0.5$ $\beta = 0.75$ (0–1)

$Q = 100$ $\rho = 0.1$ 5

$\tau = 0$

$\tau = 0$

$\tau_{12} = \tau_{21} = 0.54$

$\tau_{13} = \tau_{31} = 0.53$

$\tau_{14} = \tau_{41} = 0.35$

$\tau_{15} = \tau_{51} = 0.24$

$\tau_{23} = \tau_{32} = 0.53$

$\tau_{24} = \tau_{42} = 0.39$

$\tau_{25} = \tau_{52} = 0.18$

$\tau_{34} = \tau_{43} = 0.32$

$\tau_{35} = \tau_{53} = 0.90$

$\tau_{45} = \tau_{54} = 0.68$

Next Transition Probability.

$$P_{41} = \frac{(\tau_{41})^{\alpha} (\eta_{41})^{\beta}}{(\tau_{41})^{\alpha}(\eta_{41})^{\beta} + (\tau_{42})^{\alpha}(\eta_{42})^{\beta} + (\eta_{43})^{\beta}(\tau_{43})^{\alpha} + (\eta_{45})^{\beta}(\tau_{45})^{\alpha}}$$

$\alpha = 0.5$, $\beta = 0.75$ → 69 → 1/69 ⇒

$1/\eta_{ij}$

$\eta_{41} = 0.014$

$\eta_{45} = 0.02$

$\eta_{43} = 0.014$

$\eta_{42} = 0.036$

41 → 69 ⇓ 1/69

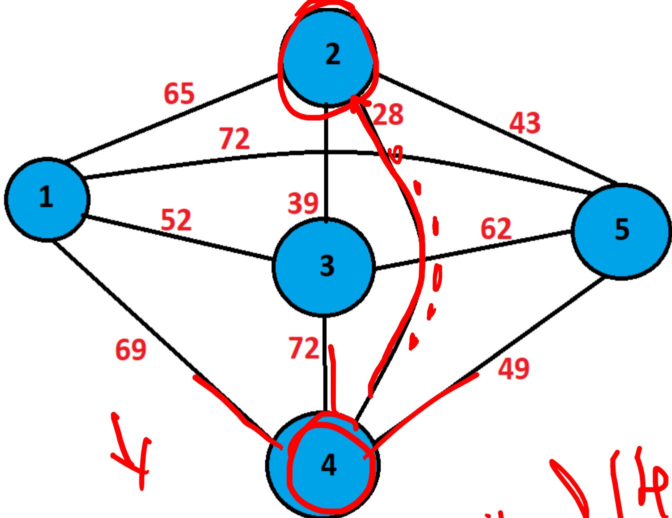
path cost

$$= \frac{(0.35)^{0.5} (0.014)^{0.75}}{(0.35)^{0.5}(0.014)^{0.75} + (0.379)^{0.5}(0.036)^{0.75} + (0.326)^{0.5}(0.014)^{0.75} + (0.683)^{0.5}(0.02)^{0.75}}$$

41 42 43 45

$$= \frac{0.024}{0.143} = \boxed{0.168}$$

$P_{41} \Rightarrow 0.168$



Initially No. of ants = No. of cities.

start at 4. $\alpha = 0.5$ $\beta = 0.75$ (0–1)

$Q = 100$ $\rho = 0.1$

visited (4, 2)

T = 0

$\tau_{12} = \tau_{21} = 0.54$

$\tau_{13} = \tau_{31} = 0.53$

$\tau_{14} = \tau_{41} = 0.35$

$\tau_{15} = \tau_{51} = 0.24$

$\tau_{23} = \tau_{32} = 0.53$

$\tau_{24} = \tau_{42} = 0.39$

$\tau_{25} = \tau_{52} = 0.18$

$\tau_{34} = \tau_{43} = 0.32$

$\tau_{35} = \tau_{53} = 0.90$

$\tau_{45} = \tau_{54} = 0.68$

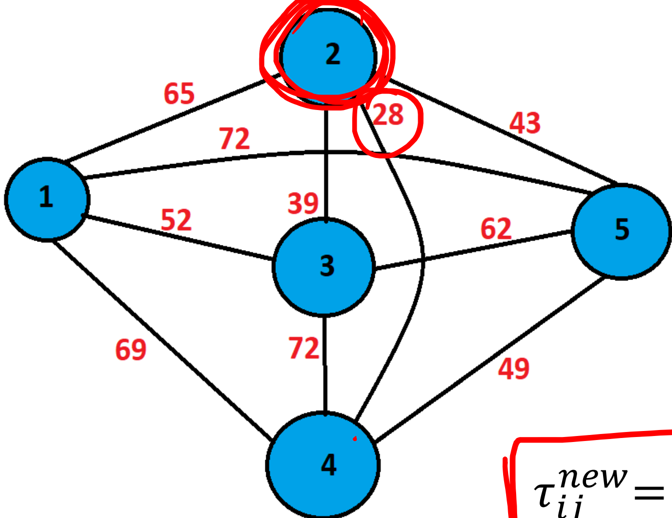
$P_{41} = \frac{0.024}{0.143} = 0.168$

$P_{42} = \frac{0.053}{0.142} = 0.364$ max

$P_{43} = \frac{0.023}{0.143} = 0.160$

$P_{45} = 0.308$

Since P_{42} is max, move from 4 to 2.



Initially No. of ants = No. of cities.
start at 4. $\alpha = 0.5$ $\beta = 0.75$ (0–1)
$Q = 100$ $\rho = 0.1$

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{f_k} & \text{if } k^{th} \text{ ant passes } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

$$\boxed{\tau_{ij}^{new} = (\rho)\tau_{ij}^{old} + \Delta\tau_{ij}^{k}}$$

$T = 0$

$\tau_{12} = \tau_{21} = 0.54$

$\tau_{13} = \tau_{31} = 0.53$

$\tau_{14} = \tau_{41} = 0.35$

$\tau_{15} = \tau_{51} = 0.24$

$\tau_{23} = \tau_{32} = 0.53$

$\tau_{24} = \tau_{42} = 0.39$

$\tau_{25} = \tau_{52} = 0.18$

$\tau_{34} = \tau_{43} = 0.32$

$\tau_{35} = \tau_{53} = 0.90$

$\tau_{45} = \tau_{54} = 0.68$

4-1

$\boxed{4-2}$ 1 0

4-3

4-5

2 $\Delta\tau_{ij}^{k}$ Computed only for the selected path 4-2

Current visit list (4,2)

Pheromone updation ($t = 1$)

$\tau_{12} = \rho(\tau_{12}) + \Delta\tau_{12} = 0.054$

0

$0.1 * 0.54 \Rightarrow 0.054$

$\tau_{13} = 0.053$

$\tau_{14} = 0.035$

$\tau_{15} = 0.024$

$\tau_{23} = 0.05$

$+100/28$

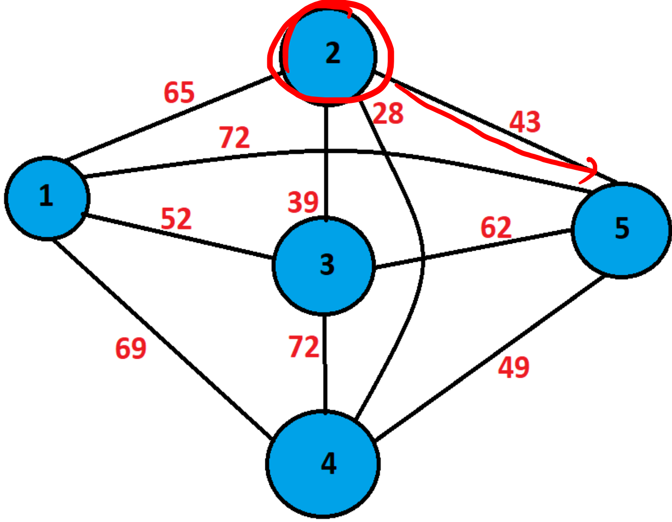
$\boxed{\tau_{24} = 0.039 + 100/28 = 3.610}$

$\tau_{25} = 0.018$

$\tau_{34} = 0.032$

$\tau_{35} = 0.09$

$\tau_{45} = 0.068$



Initially No. of ants = No. of cities.
start at 4. $\alpha = 0.5$ $\beta = 0.75$ (0–1)
$Q = 100$ $\rho = 0.1$

Now ant is at 2.

$\left\{\begin{matrix} 2-1 \\ 2-5 \\ 2-3 \end{matrix}\right\}$ $\cancel{2-4}$

P_{21} = [illegible]
P_{23} = [illegible]
P_{25} = [illegible]

2-3 — max

Current visit list (4,2).

Pheromone updation (t = 1)

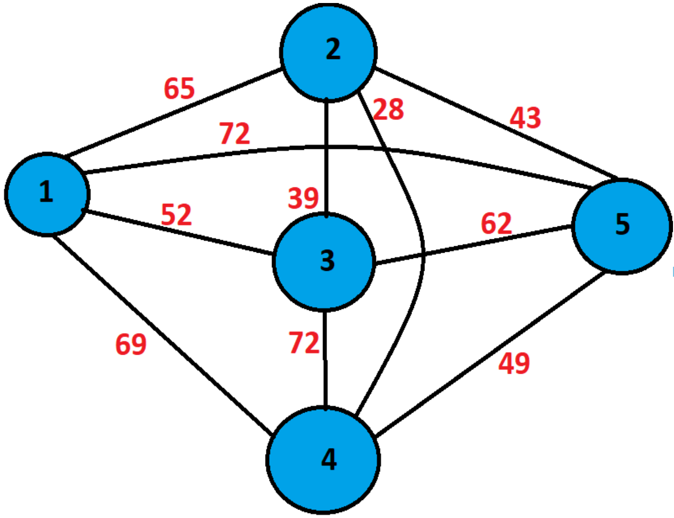
$\tau_{12} = \rho(\tau_{12}) + \Delta\tau_{12} = 0.054$
$\tau_{13} = 0.053$
$\tau_{14} = 0.035$
$\tau_{15} = 0.024$
$\boxed{\tau_{23} = 0.05}$
$\tau_{24} = 0.039 + 100/28 = 3.610$
$\boxed{\tau_{25} = 0.018}$
$\tau_{34} = 0.032$
$\tau_{35} = 0.09$
$\tau_{45} = 0.068$

$$P_{21} \Rightarrow \frac{(\tau_{21})^{\alpha}(\eta_{21})^{\beta}}{(\tau_{21})^{\alpha}(\eta_{21})^{\beta} + (\tau_{23})^{\alpha}(\eta_{23})^{\beta} + (\tau_{25})^{\alpha}(\eta_{25})^{\beta}}$$

1/65 =)
1/39

$$= \frac{(0.054)^{0.5} * (0.15)^{0.75}}{(\quad) + (0.05)^{0.5} * (0.025)^{0.75} + (0.018)^{0.5} * (1/43)^{0.75}}$$

$\Rightarrow 0.320$



Initially No. of ants = No. of cities.

start at 4. $\alpha = 0.5$ $\beta = 0.75$ (0-1)

$Q = 100$ $\rho = 0.1$

Now ant moves to 3.

Pheromone updation ($t = 2$) $\Delta \to 0$

$\tau_{12} = 0.005$

$\tau_{13} = 0.005$

$\tau_{14} = 0.003$

$\tau_{15} = 0.002$

$\boxed{\tau_{23} = \rho(0.05) + 100/39 = \cancel{0.01}\ 2.569}$

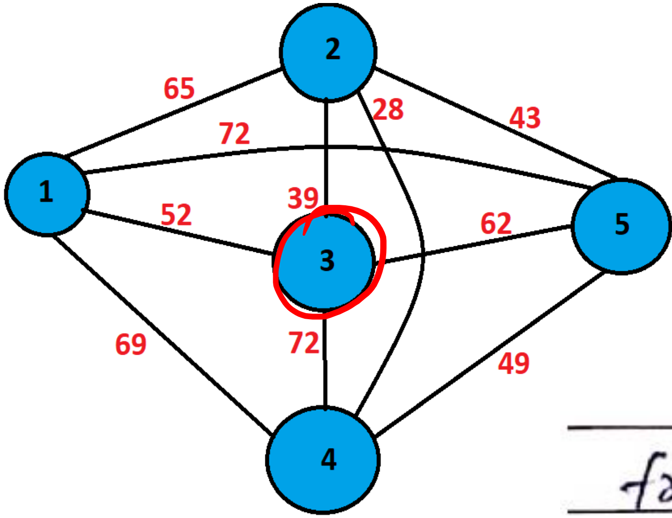
$\tau_{24} = (0.1)(3.616) = 0.361$

$\tau_{25} = 0.001$

$\tau_{34} = 0.003$

$\tau_{35} = 0.009$

$\tau_{45} = 0.006$



Initially No. of ants = No. of cities.
start at 4. $\alpha = 0.5$ $\beta = 0.75$ $(0-1)$
$Q = 100$ $\rho = 0.1$

from 3, find P_{31}, P_{35}

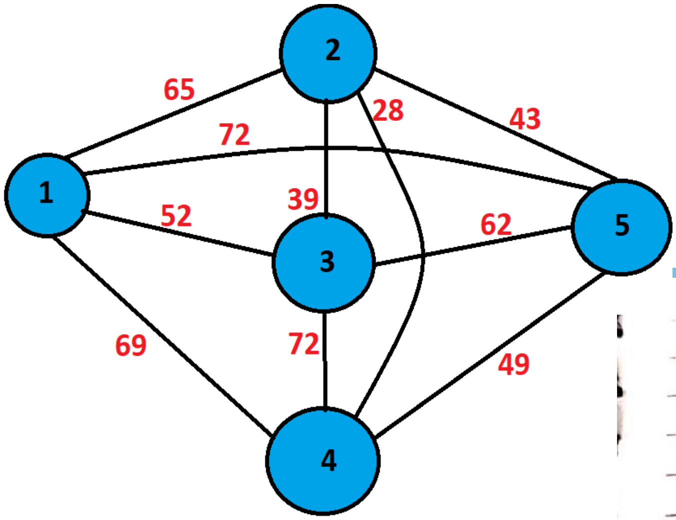
$P_{31} = 0.554$ — max.

$P_{35} = 0.446$.

P_{31} | max
P_{35} |

$3 \rightarrow 1$
$3 - 5$
~~3 - 2~~
~~3 - 4~~

$\{4, 2\}$



5 cities

t

Initially No. of ants = No. of cities.

start at 4. $\alpha = 0.5$ $\beta = 0.75$ (0–1)

$Q = 100$ $\rho = 0.1$

From 3, ant moves to 1.

Tabu list [4, 2, 3, 1.]

Pheromone updation $t = 3$.

4231

$\tau_{12} = 0.0005$

$\boxed{\tau_{13} = (0.1)(0.005) + 100/52 = 1.9235}$

$\tau_{14} = 0.0003$

$\tau_{15} = 0.002$

$\tau_{23} = 0.25$

$\tau_{24} = 0.03$

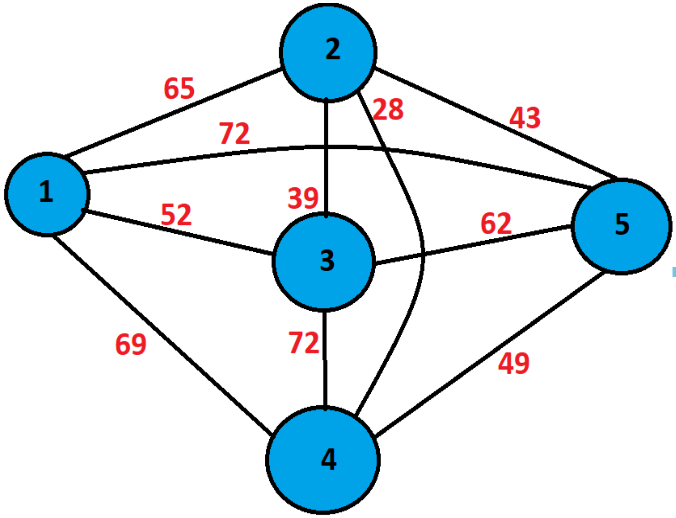
$\tau_{25} = 0.0001$

$\tau_{34} = 0.0003$

$\tau_{35} = 0.0009$

$\tau_{45} = 0.0006$.

From 1, Now move to 5. Since it is the only non visited city.



N

Initially No. of ants = No. of cities.
start at 4. $\alpha = 0.5$ $\beta = 0.75$ (0–1)
$Q = 100$ $\rho = 0.1$

Pheremone Updation. $t = 4$.

$\tau_{12} = 0.00005$

$\tau_{13} = 0.019$

$\tau_{14} = 0.00003$

$\tau_{15} = 0.00002 + 100/72 = 1.388$.

$\tau_{23} = 0.025$

$\tau_{24} = 0.003$

$\tau_{25} = 0.00001$

$\tau_{34} = 0.00003$

$\tau_{35} = 0.00009$

$\tau_{45} = 0.00006$.

P_{15}

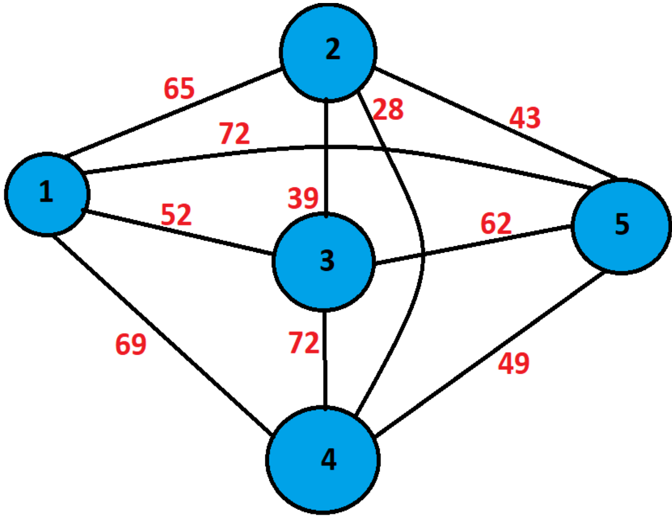
$\{4, 2, 3, 1\}$

5

Now back to origin since all states visited

update $\tau_{45} = 0.000006 * 100/49$

$\tau_{45} = 2.0408$.



Initially No. of ants = No. of cities.
start at 4. $\alpha = 0.5$ $\beta = 0.75$ (0–1)
$Q = 100$ $\rho = 0.1$

Pheromone Matrix

	t=0	t=1	t=2	t=3	t=4	t=5
τ_{12}	0.54	0.054	0.0054	0.0005	0.00005	0.000005
τ_{13}	0.53	0.053	0.0053	$\boxed{1.9235}$	0.192	0.019
τ_{14}	0.35	0.035	0.0035	0.0003	0.00003	0.000003
τ_{15}	0.24	0.024	0.0024	0.0002	$\boxed{1.388}$ ~~0.00002~~	0.138
τ_{23}	0.53	0.053	$\boxed{2.569}$	0.25	0.025	0.0025
τ_{24}	0.39	$\boxed{3.610}$	0.361	0.036	0.003	0.0003
τ_{25}	0.18	0.018	0.001	0.0001	0.00001	0.000001
τ_{34}	0.32	0.032	0.003	0.0003	0.00003	0.000003
τ_{35}	0.90	0.090	0.009	0.0009	0.00009	0.000009
τ_{45}	0.68	0.068	0.006.	0.0006	0.00006	$\boxed{2.0408}$

Path

$4 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 5$

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

h(n) → NA → uninformed Search

h(n) → Presence → informed "

Local Search

8t

Module 3 : Searching to play games

A. Minimax Algorithm

B. Alpha-Beta Pruning

C. Making imperfect real time decisions

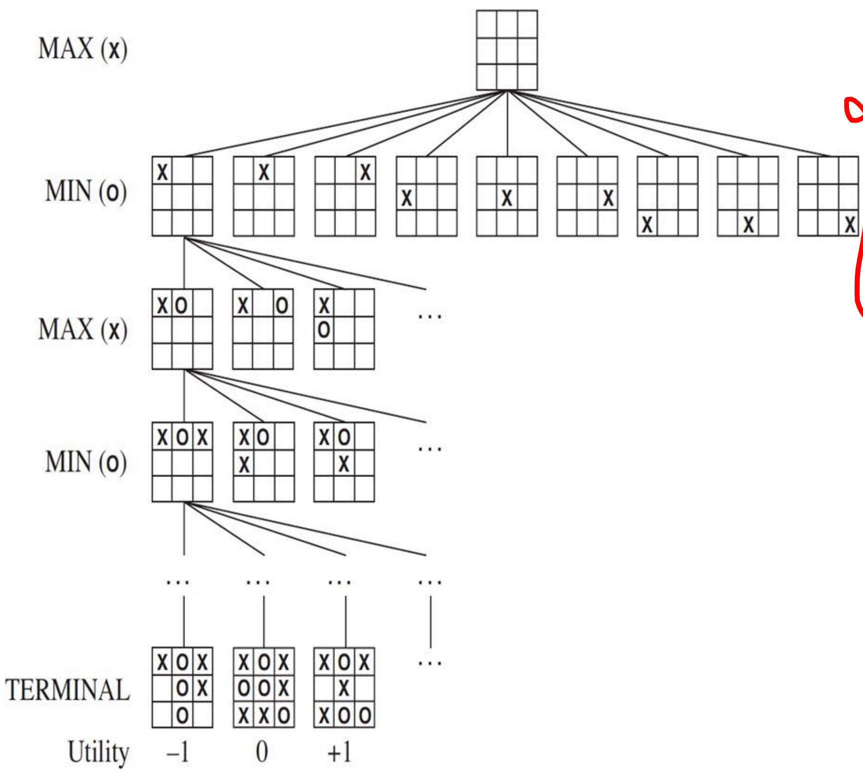
Learning Objective

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 – f/P
- Discrete ✓
- Deterministic ✓
- Number of Agents : Single / multiple agents

Adversarial Search

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 → 2 | 1

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

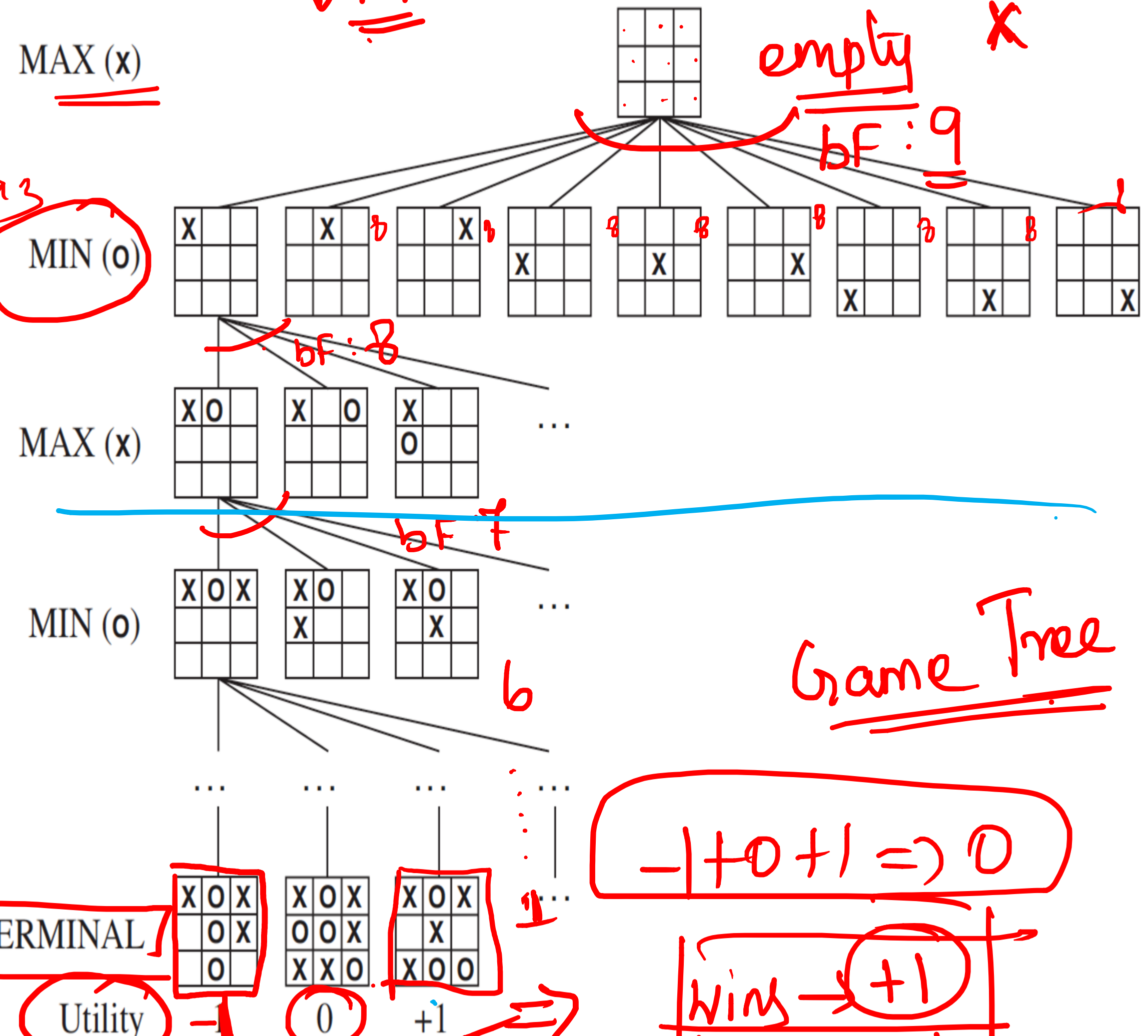
Player X starts the game → Max

Player O ↓ MIN

PSA : Representation of Game:

- INITIAL STATE: S0
- PLAYER(s)
- ACTIONS(s) – $a_1\ a_2\ a_3$
- RESULT(s, a)
- TERMINAL-TEST(s)
- UTILITY(s, p)

Eg., Tic Tac Toe



Game Tree

Zero Sum Problem

+1
−1
0

Max (X)

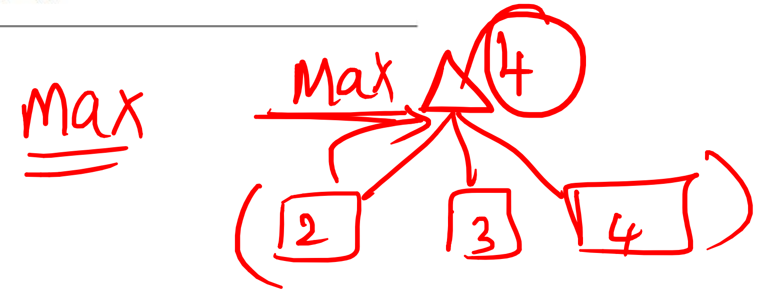
w.r.t only one Player

−1+0+1 => 0

Wins → +1
loses → −1
Draw → 0

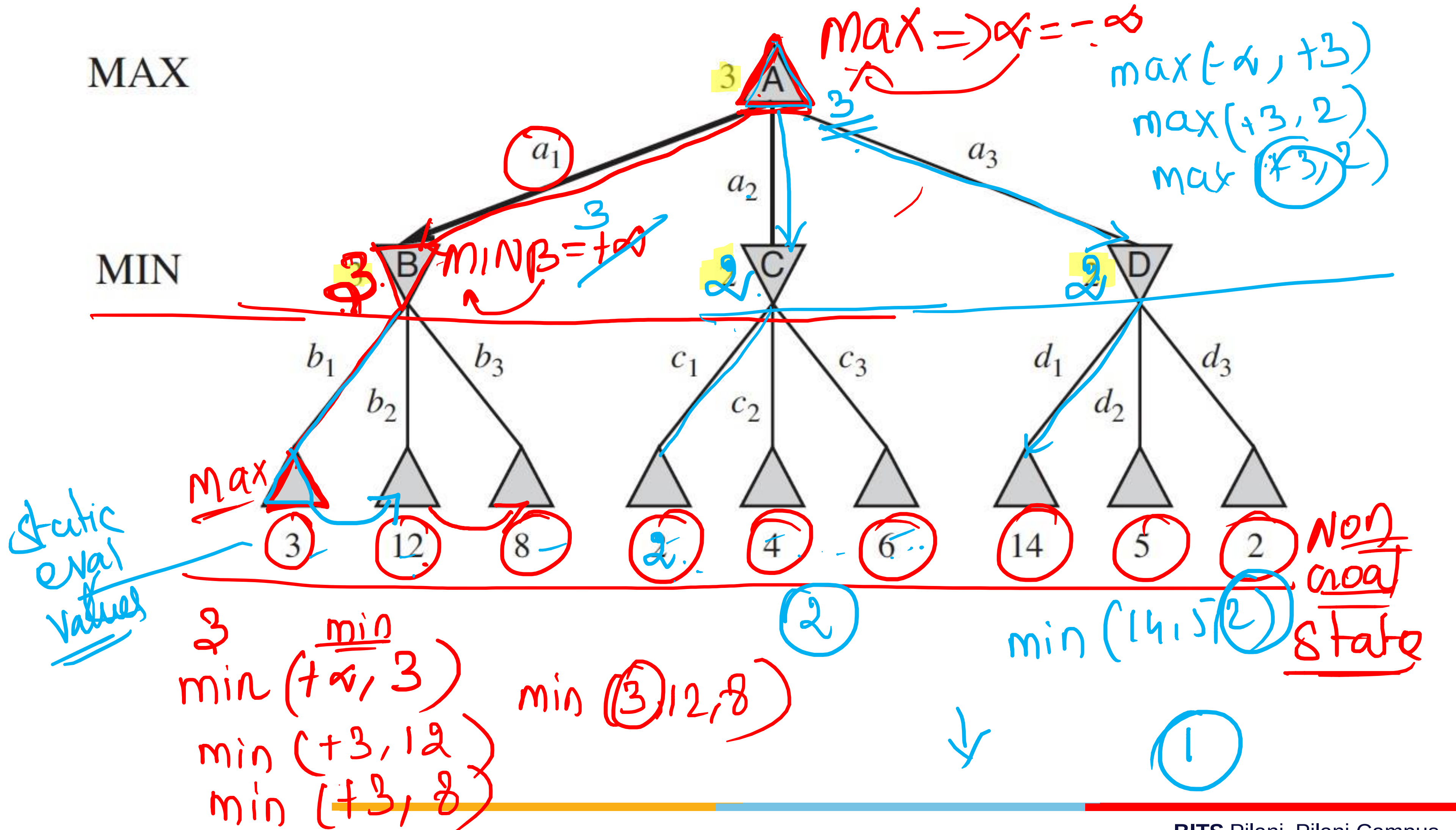
Min-Max Algorithm

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

function MAX-VALUE(state) 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)))
   return v

function MIN-VALUE(state) 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)))
   return v
```



DFS

Book Example



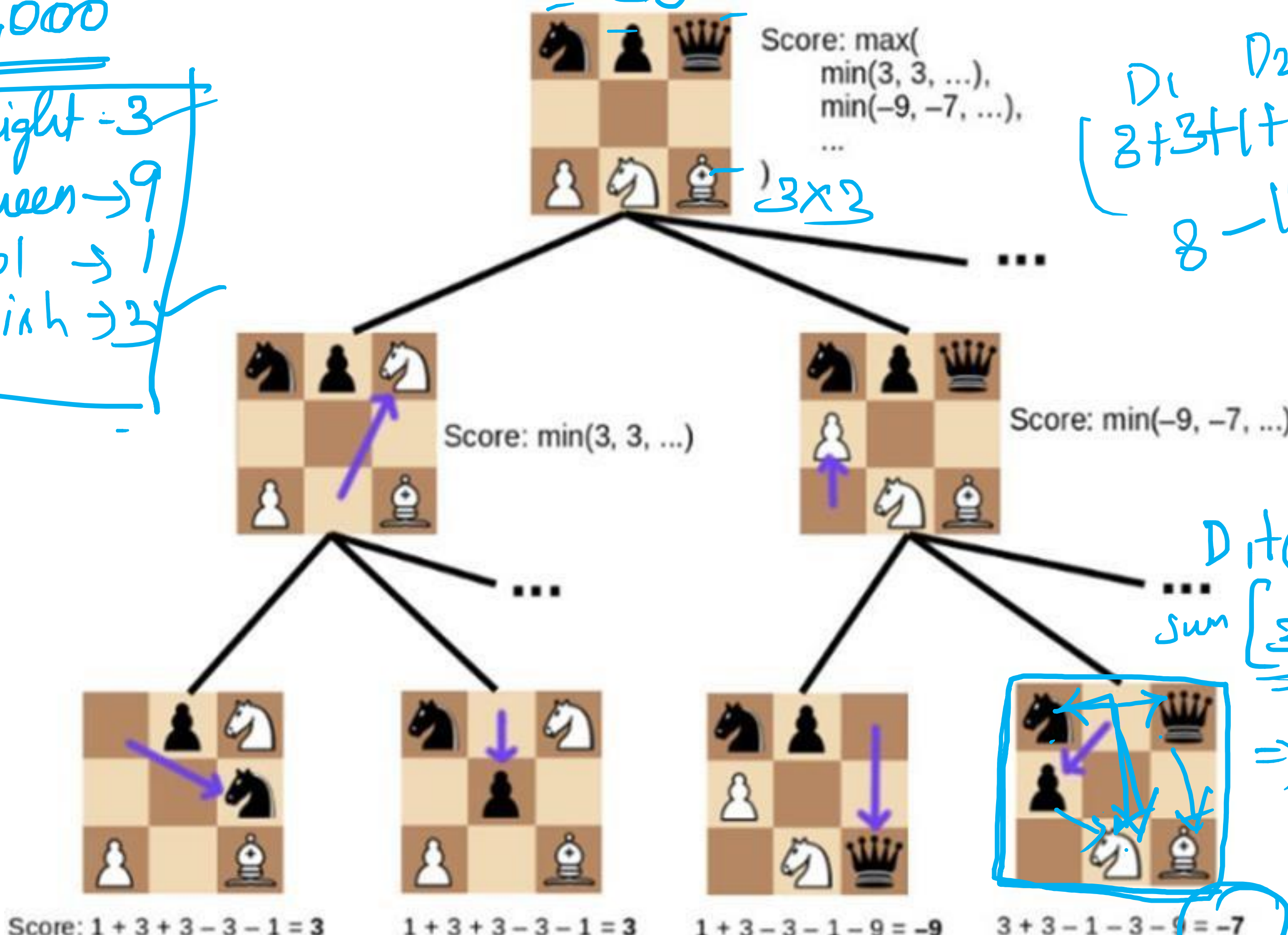
Design of Static Evaluation Values

2 Player → Black
→ White

18,000

Knight - 3
Queen → 9
Sol → 1
Bish → 3

So



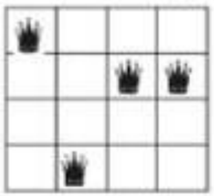
3×3

D1 D2
[3+3+1+1] — [3+1+9+1+1+1]
8 − 16 ⇒ −8

white black
U(max) − U(min)
D1 + D2
sum [3 + 3] − [3 + 9 ...]
⇒ 6 − 13
⇒ −7

Design of Static Evaluation Values

N-Queens



1	4	2	2	4

Tic-Tac-Toe

0	0	x
x	0	0
0	0	x

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

max - min

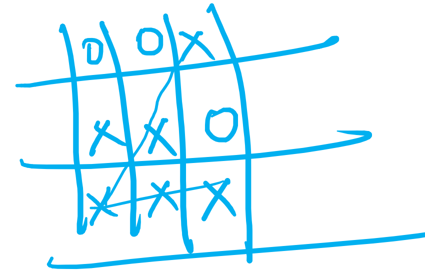
2 - 1 => 1 D1

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

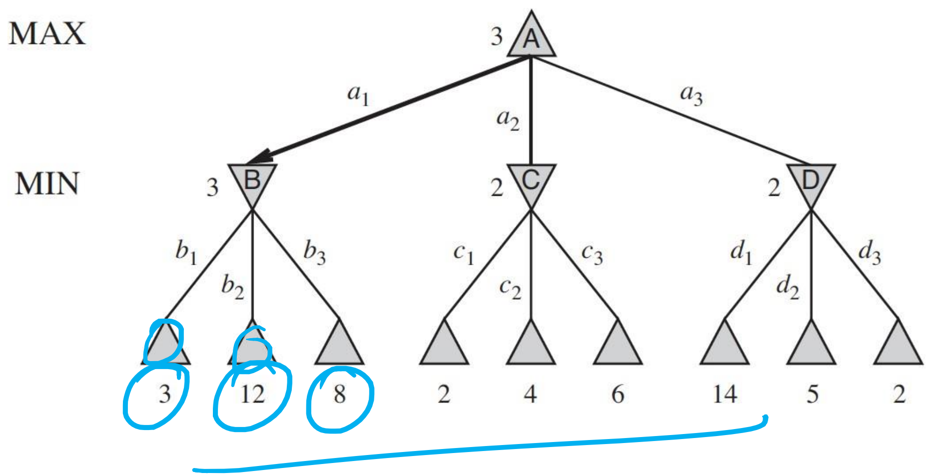


0.5 0.3 0.2 => 1

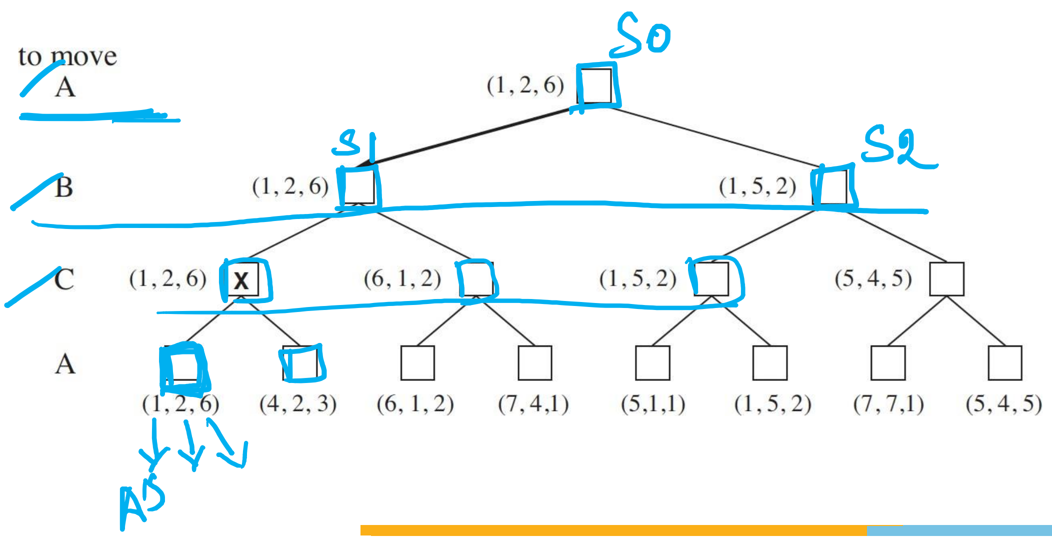
D1 D2 D3

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

= 0.6 (MaxChance - MinChance) + 0.4 (MaxPairs - MinPairs)



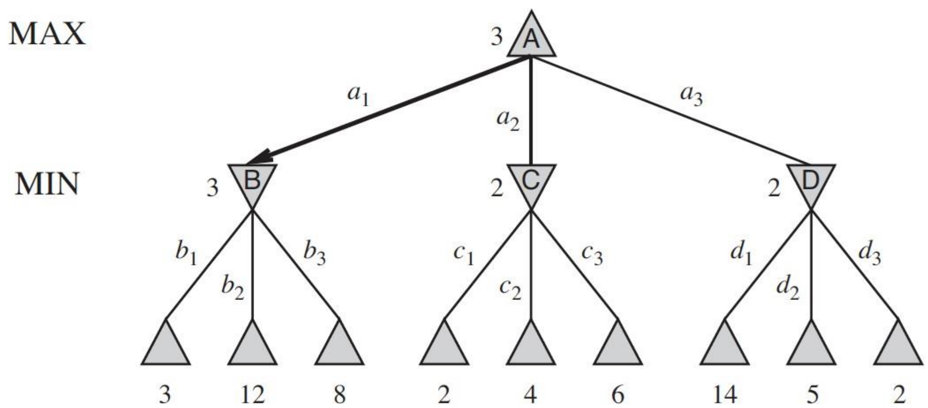
Two Player Game : 1-Ply Game



Multiplayer Game

3 different values

1 Max

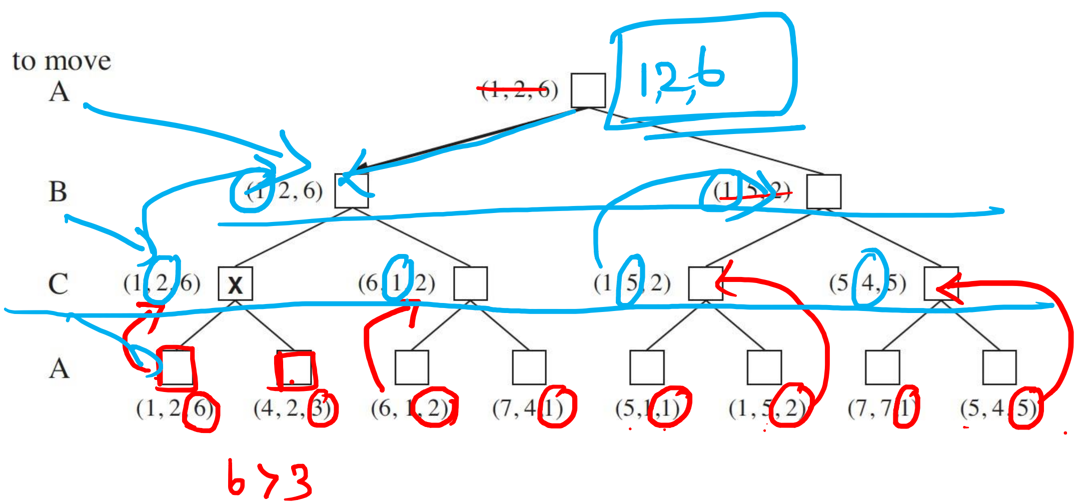


Two Player Game : 1-Ply Game

B

A

C



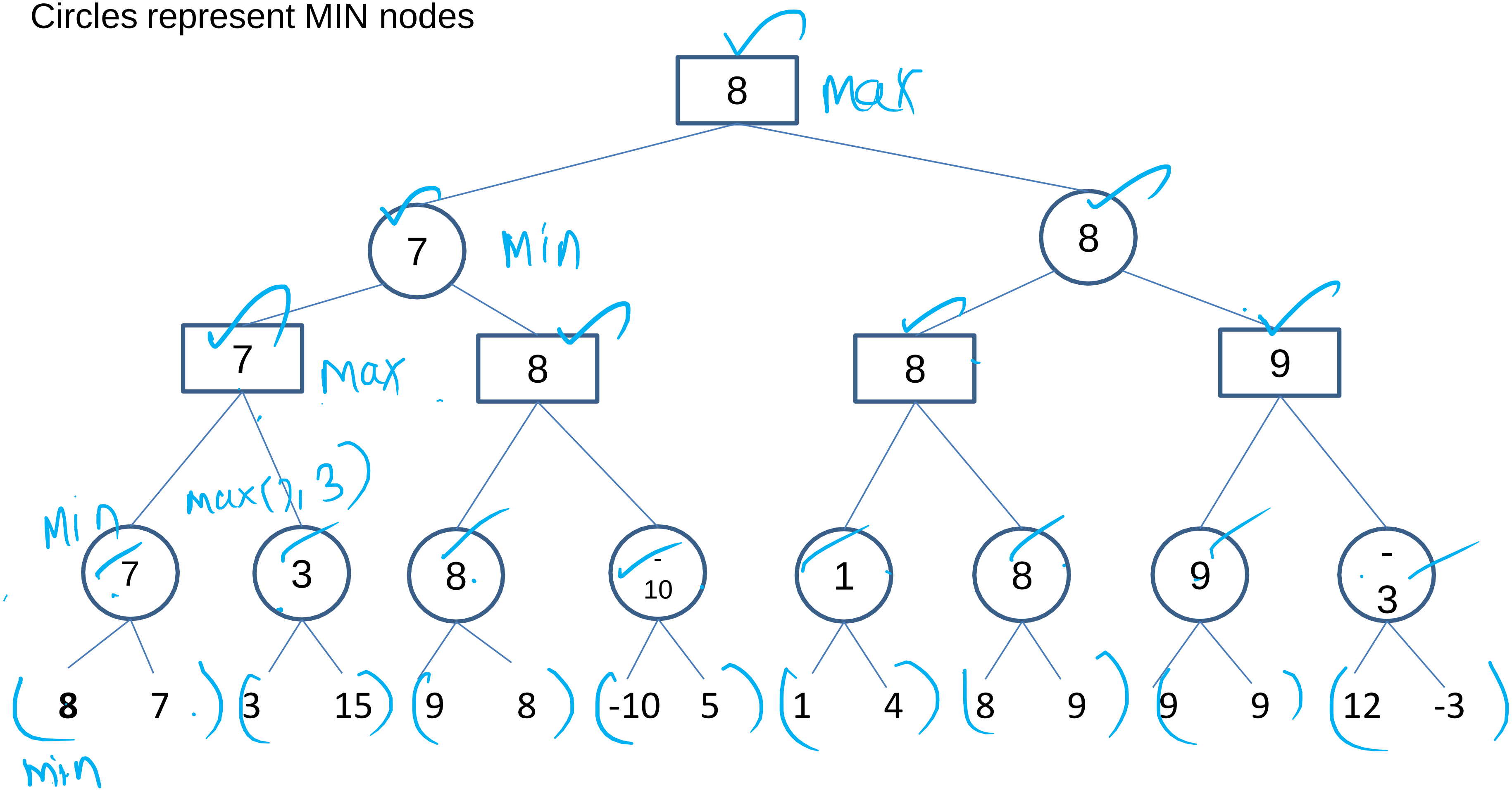
Multiplayer Game

max

Min-Max Algorithm – Example -1

Squares represent MAX nodes

Circles represent MIN nodes



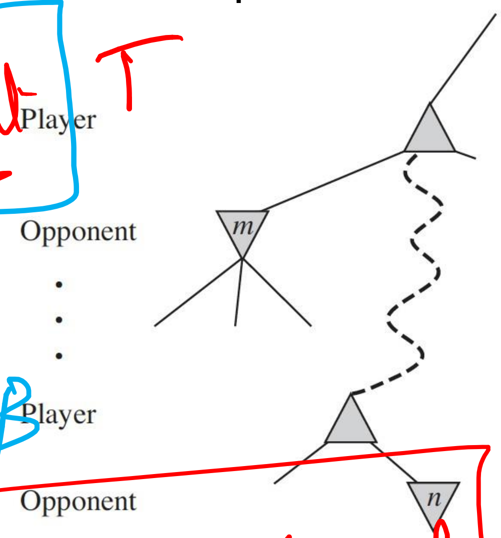
Alpha – beta Pruning → removing

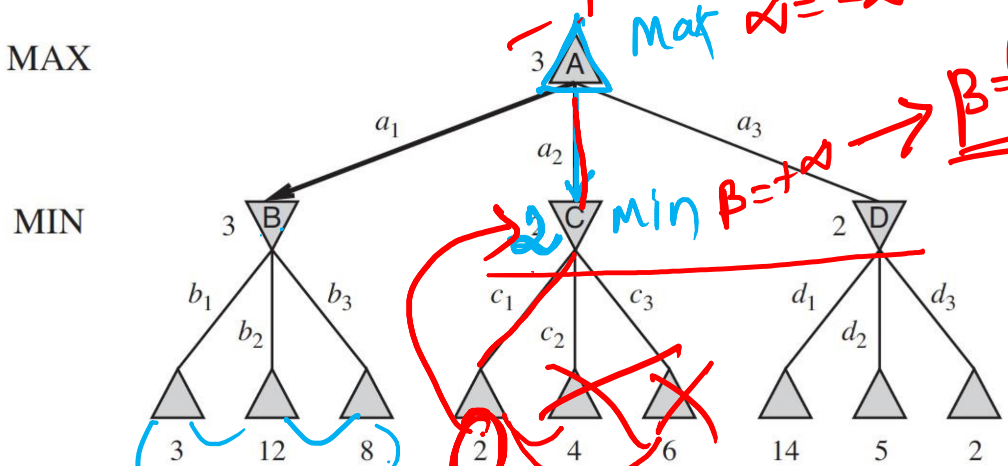
General Principle:

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

$\beta_{parent} \le \alpha_{current}$

min max





$\beta_{current} \le \alpha_{parent}$

$+2 \le 3$ True

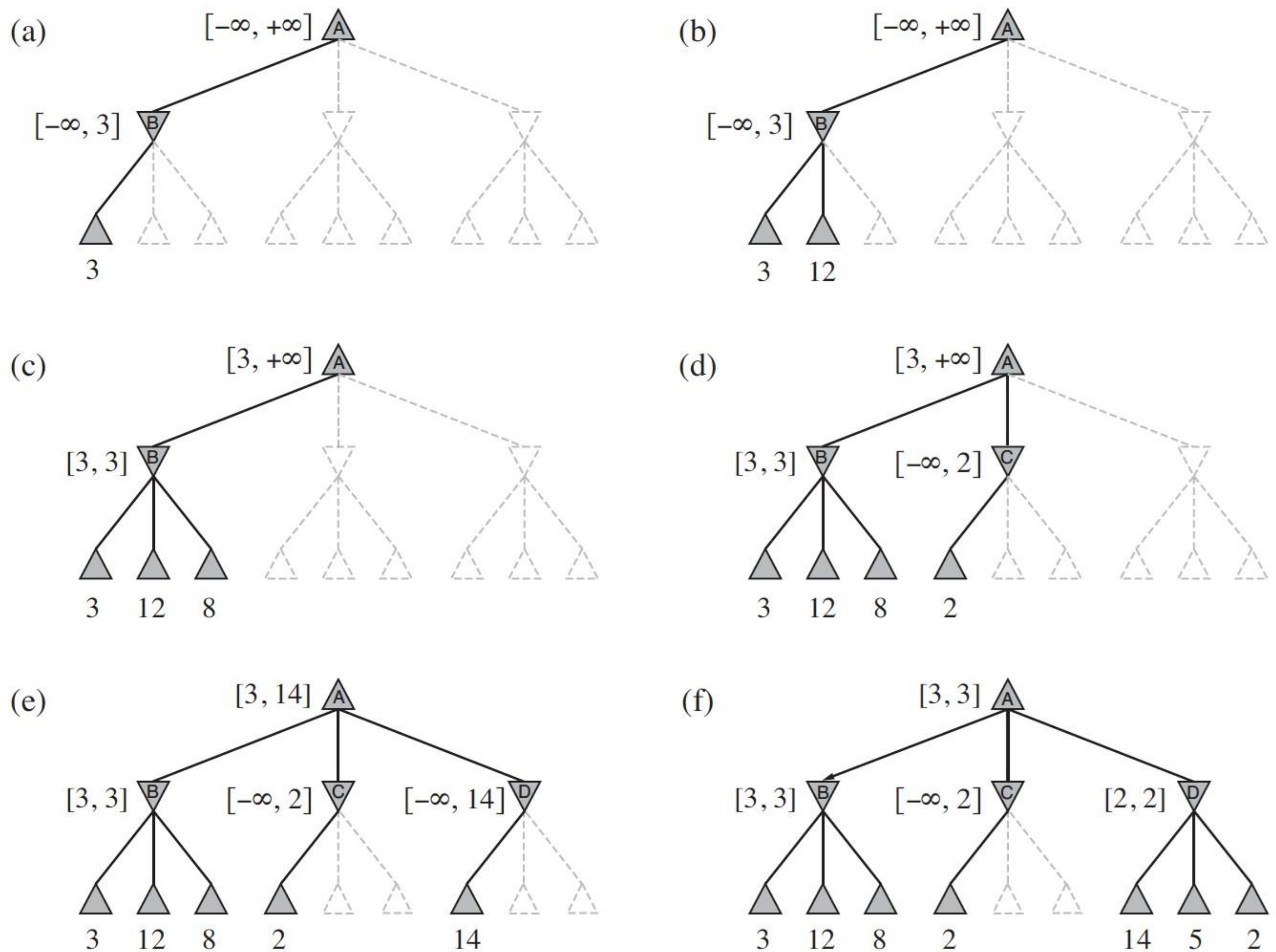
$$\begin{aligned}
\text{MINIMAX}(root) &= \max(\min(3,12,8), \min(2,x,y), \min(14,5,2)) \\
&= \max(3, \min(2,x,y), 2) \\
&= \max(3, z, 2) \qquad \text{where } z = \min(2,x,y) \le 2 \\
&= 3.
\end{aligned}$$

2, 100

1

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?

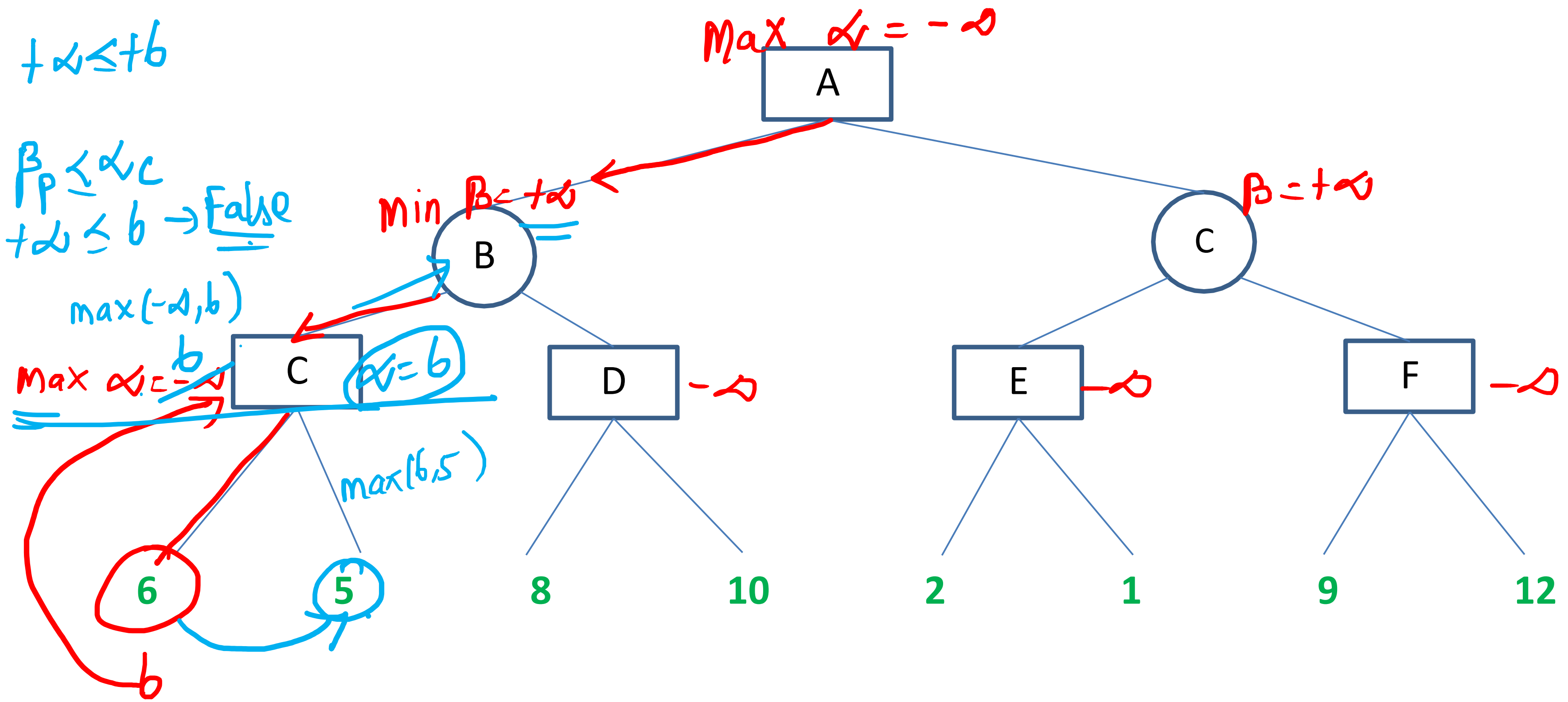
Alpha – beta Pruning

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

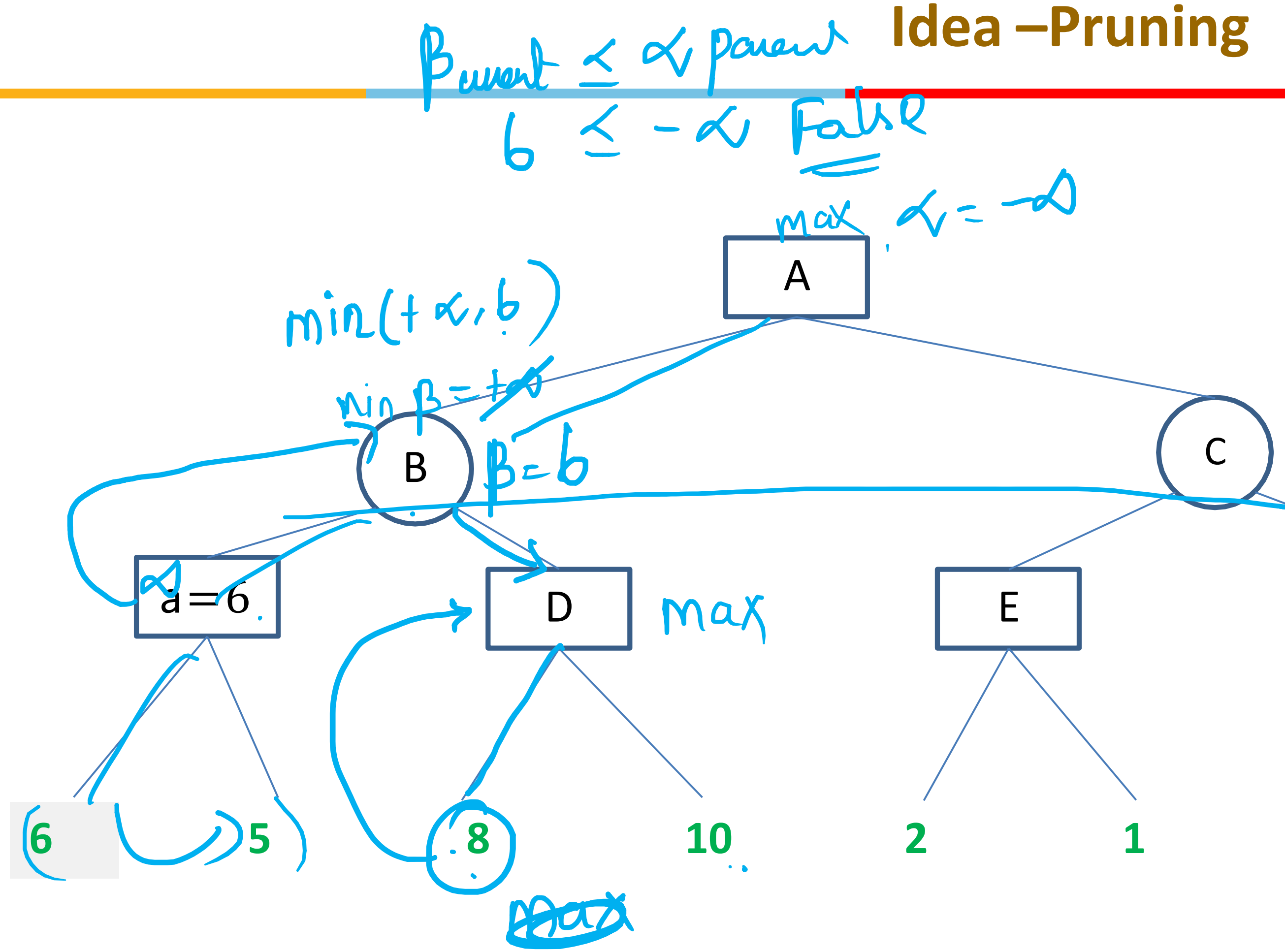
Alpha Beta Pruning - Another Example

Idea –Pruning



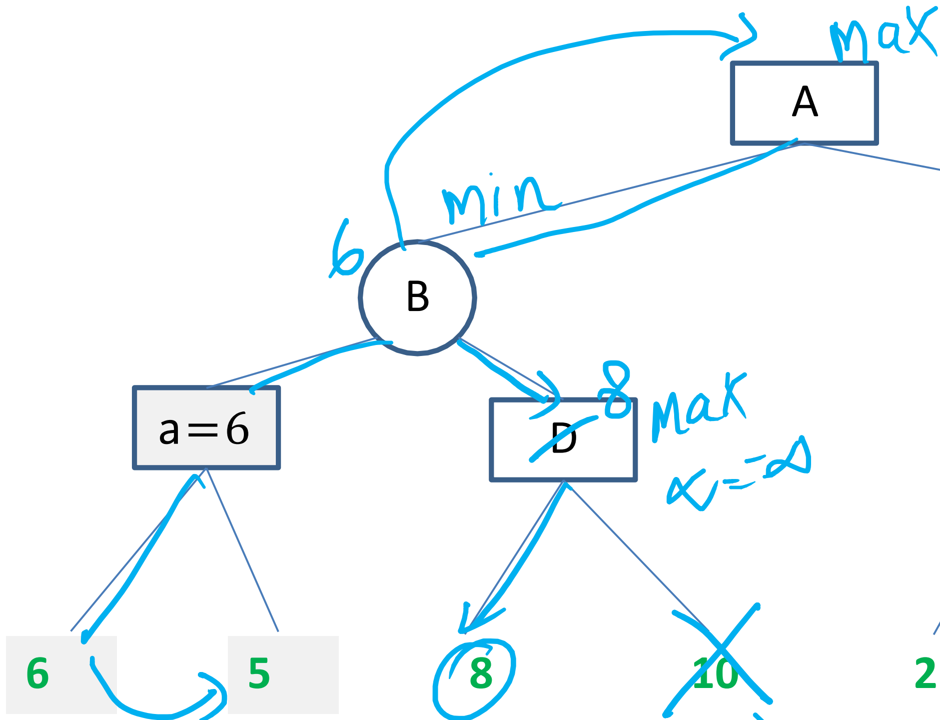
Alpha Beta Pruning

Idea –Pruning



Alpha Beta Pruning

Idea –Pruning



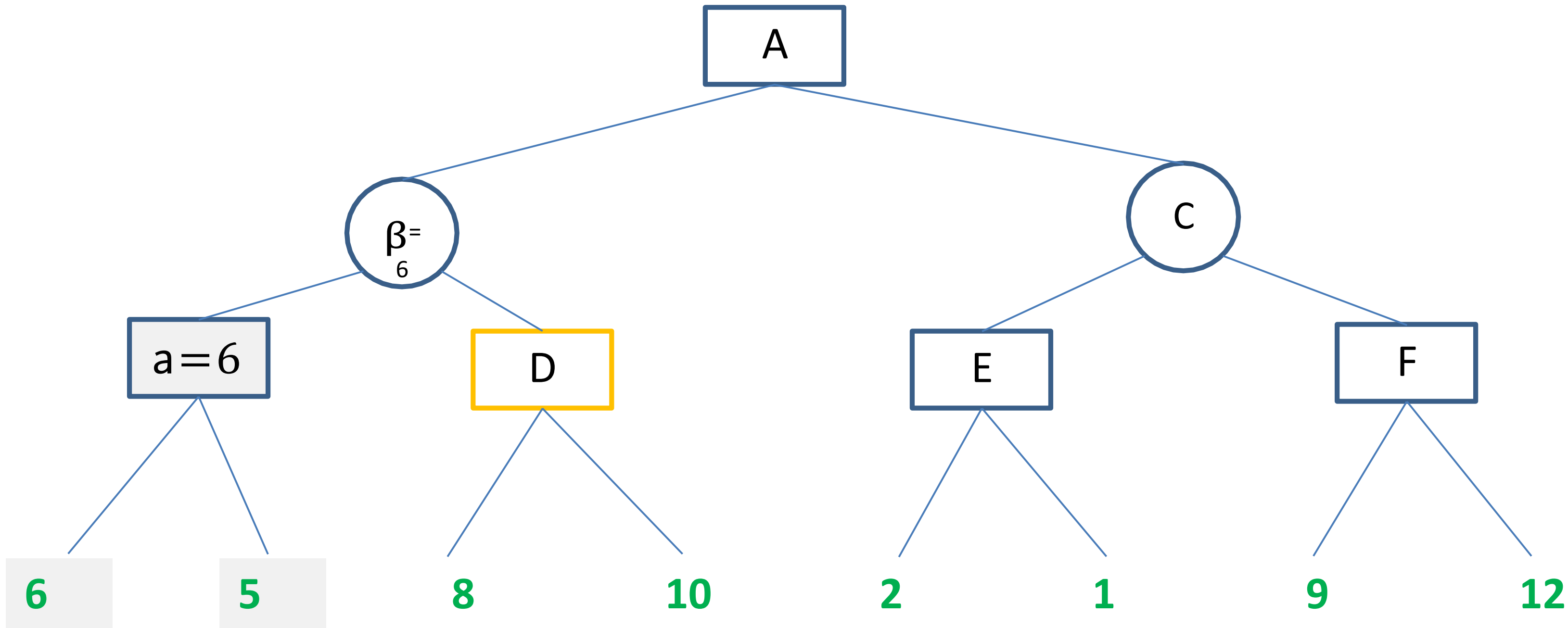
max $(8, -\infty)$

$\beta_{parent} \leq \alpha_{parent}$

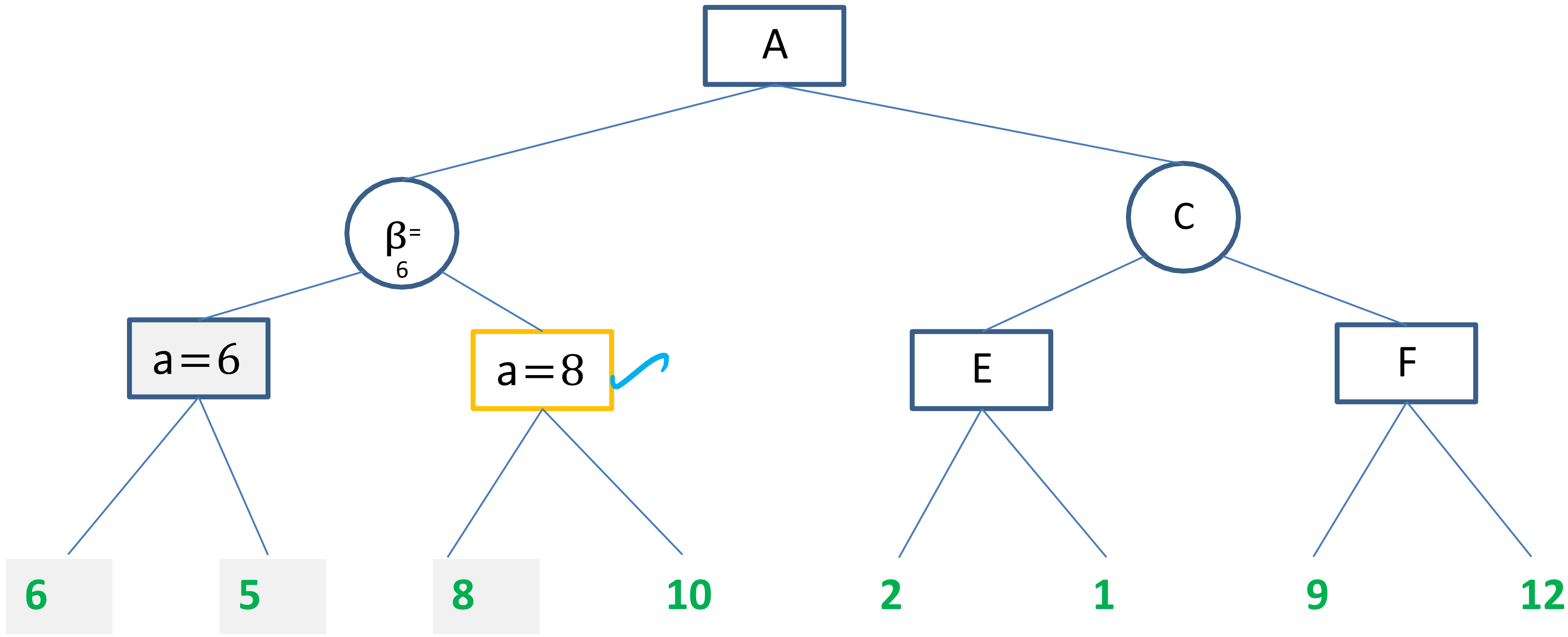
$6 \leq 8$ True

Alpha Beta Pruning

Idea –Pruning

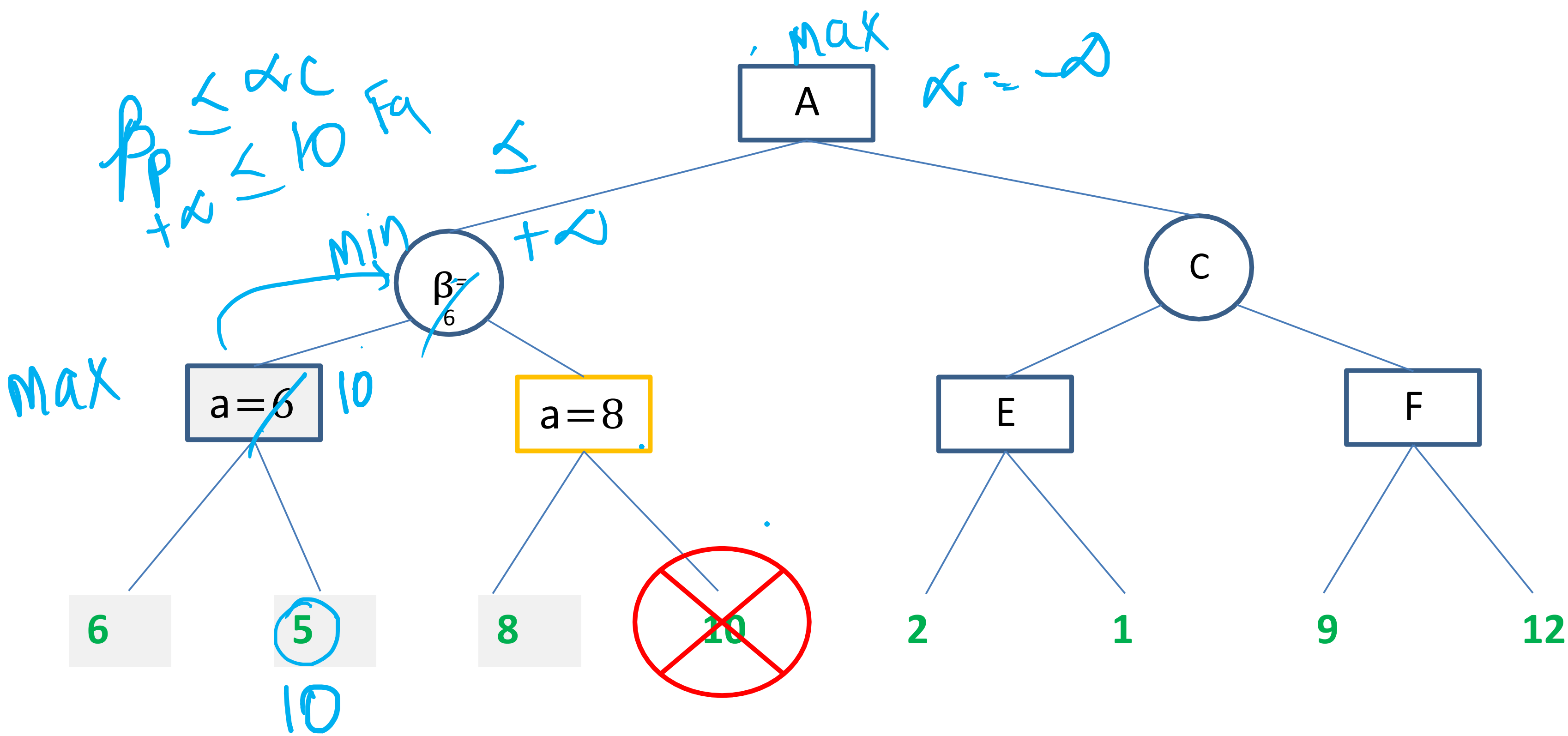


Idea – Alpha Pruning



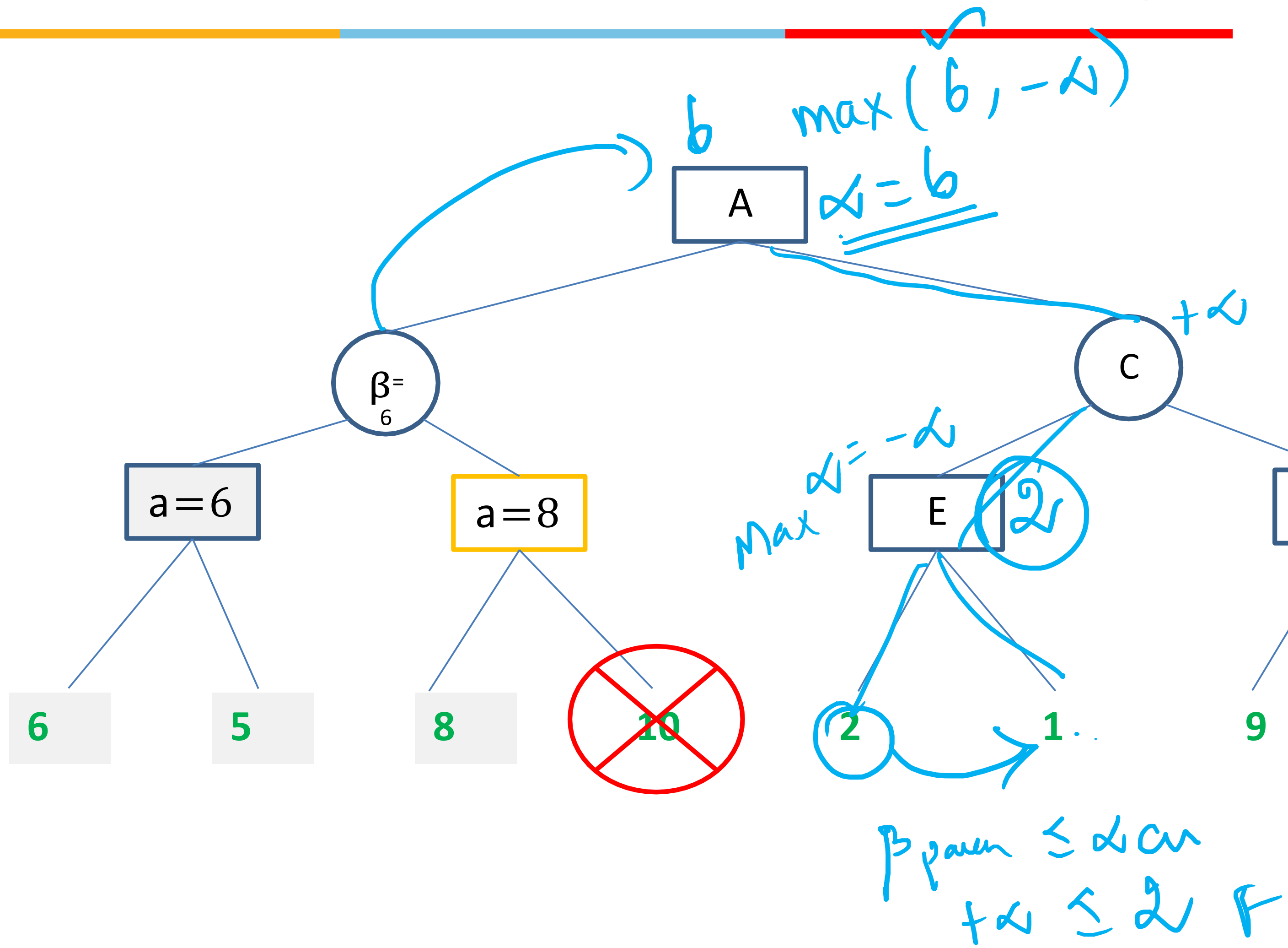
Alpha Beta Pruning

Idea – Beta Pruning

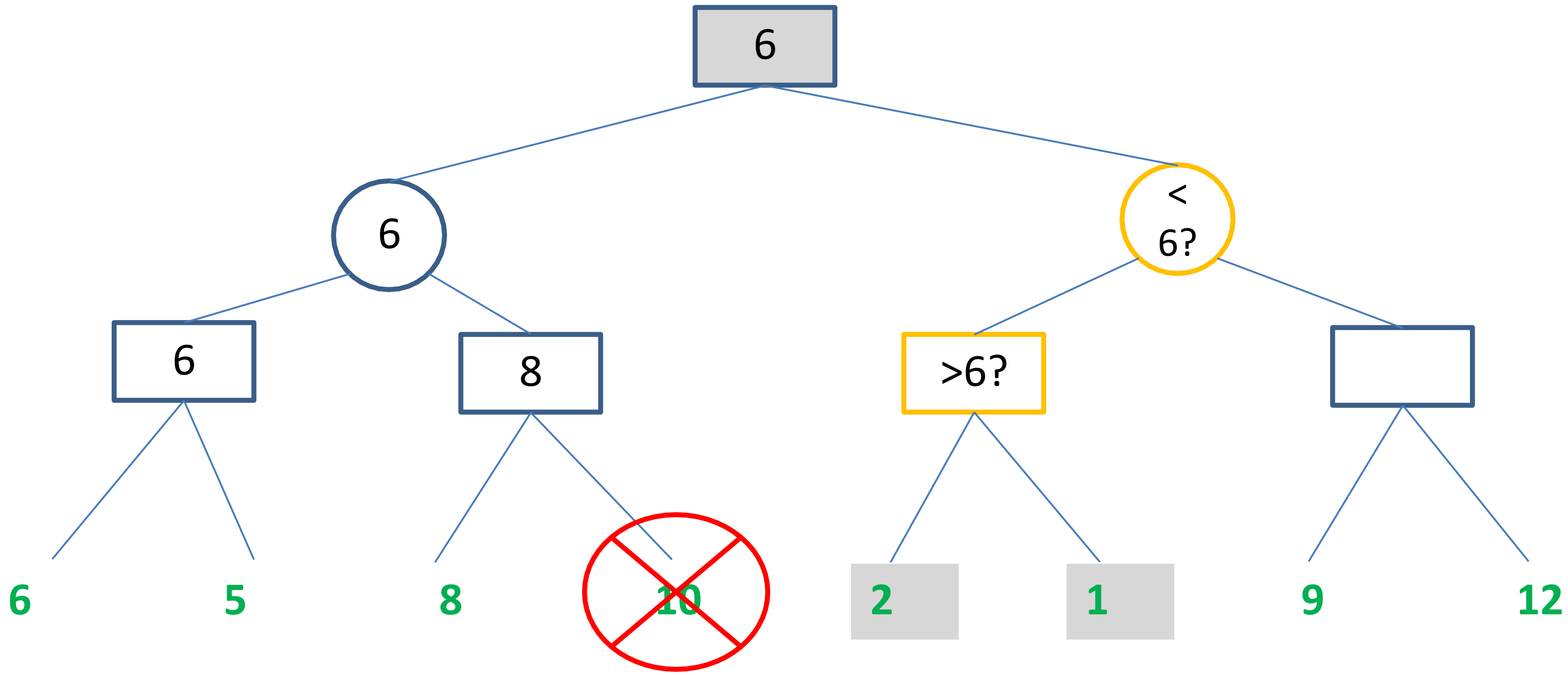


Alpha Beta Pruning

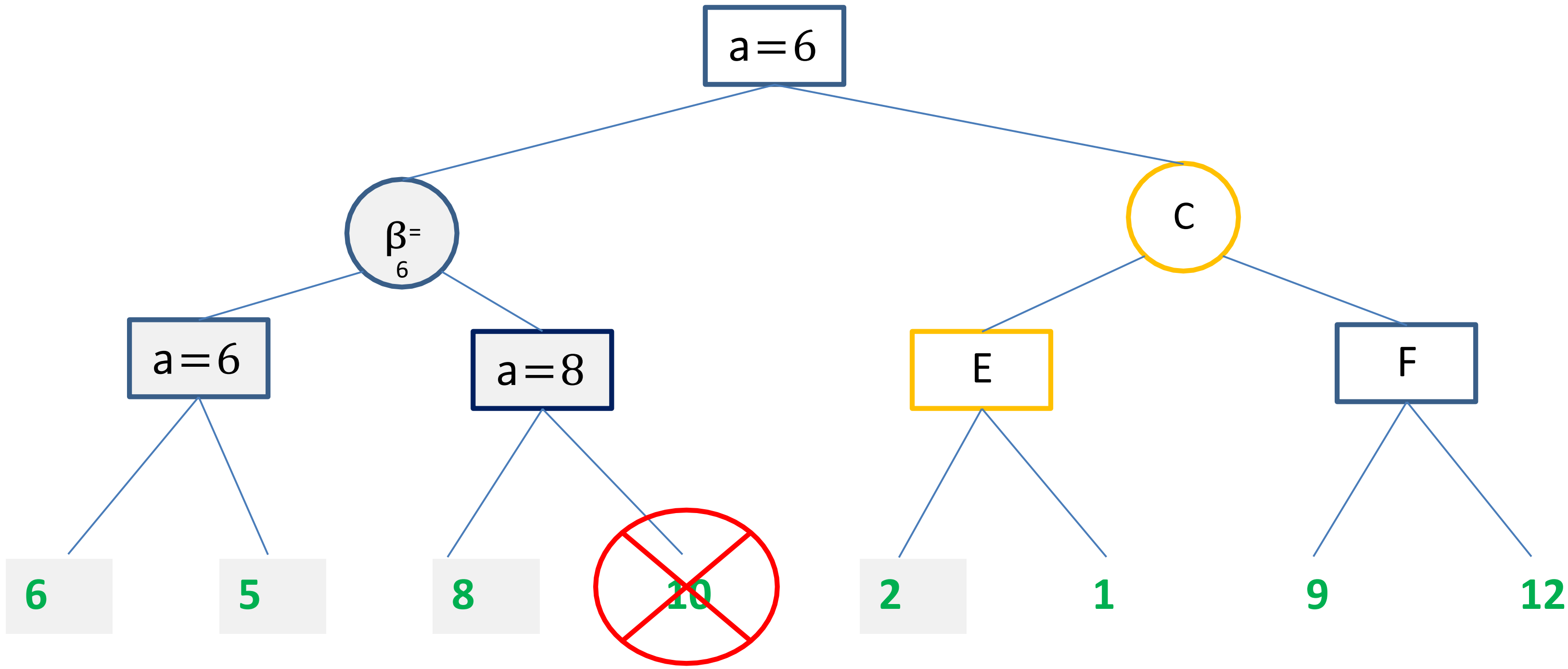
Idea –Pruning



Idea –Pruning

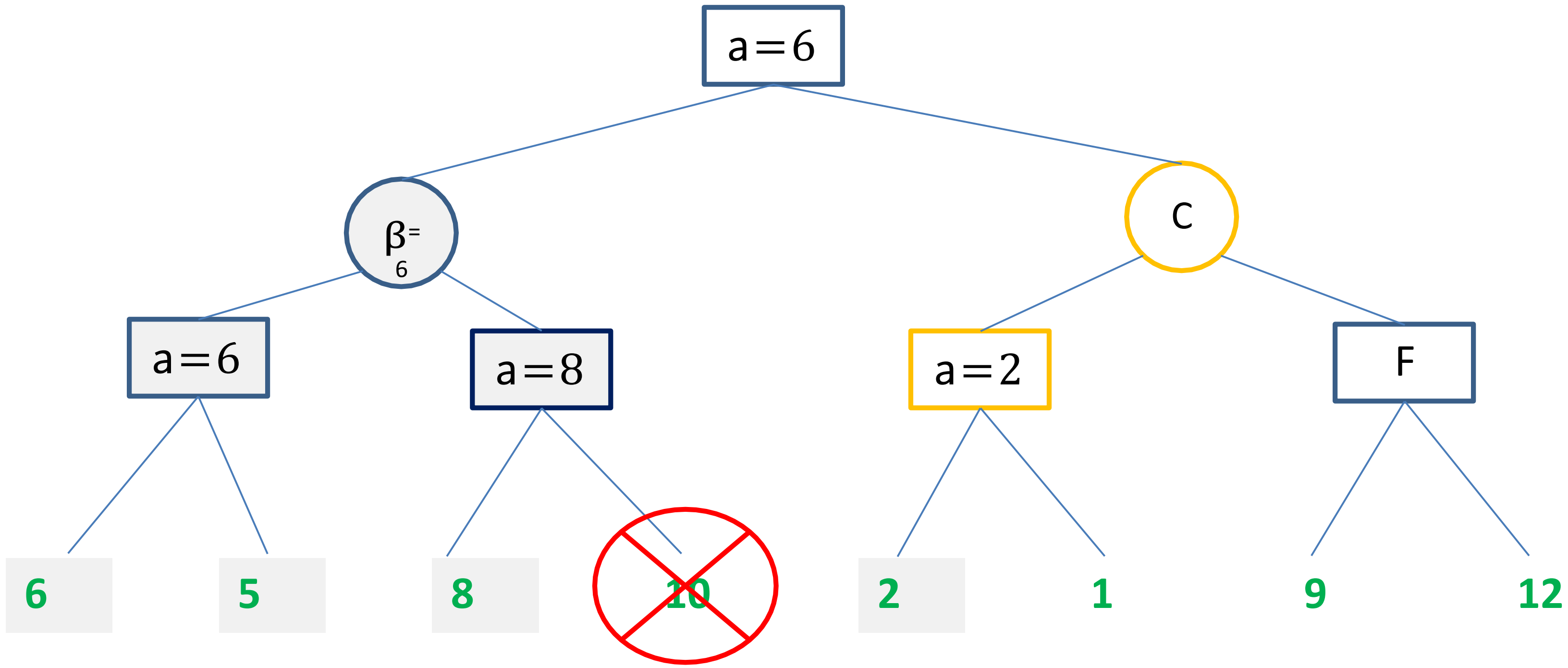


Idea –Pruning

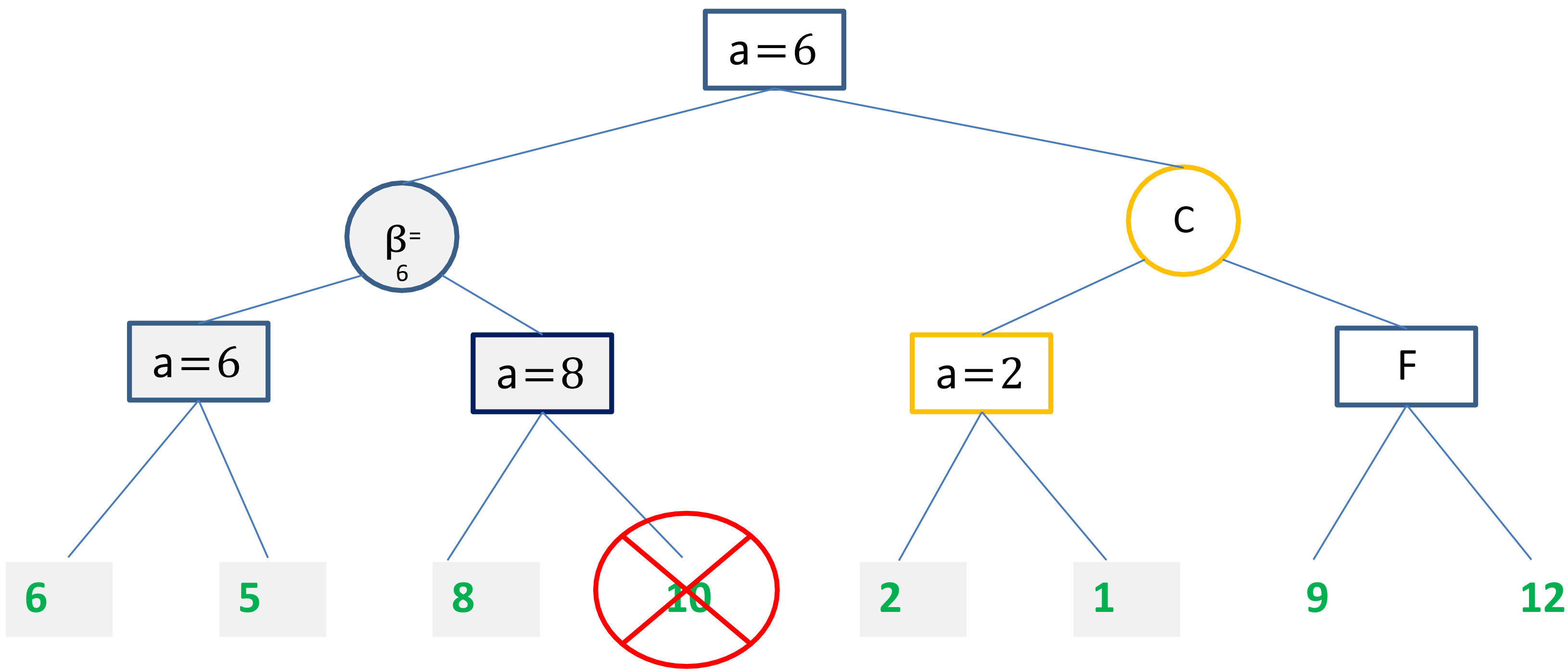


Alpha Beta Pruning

Idea –Pruning

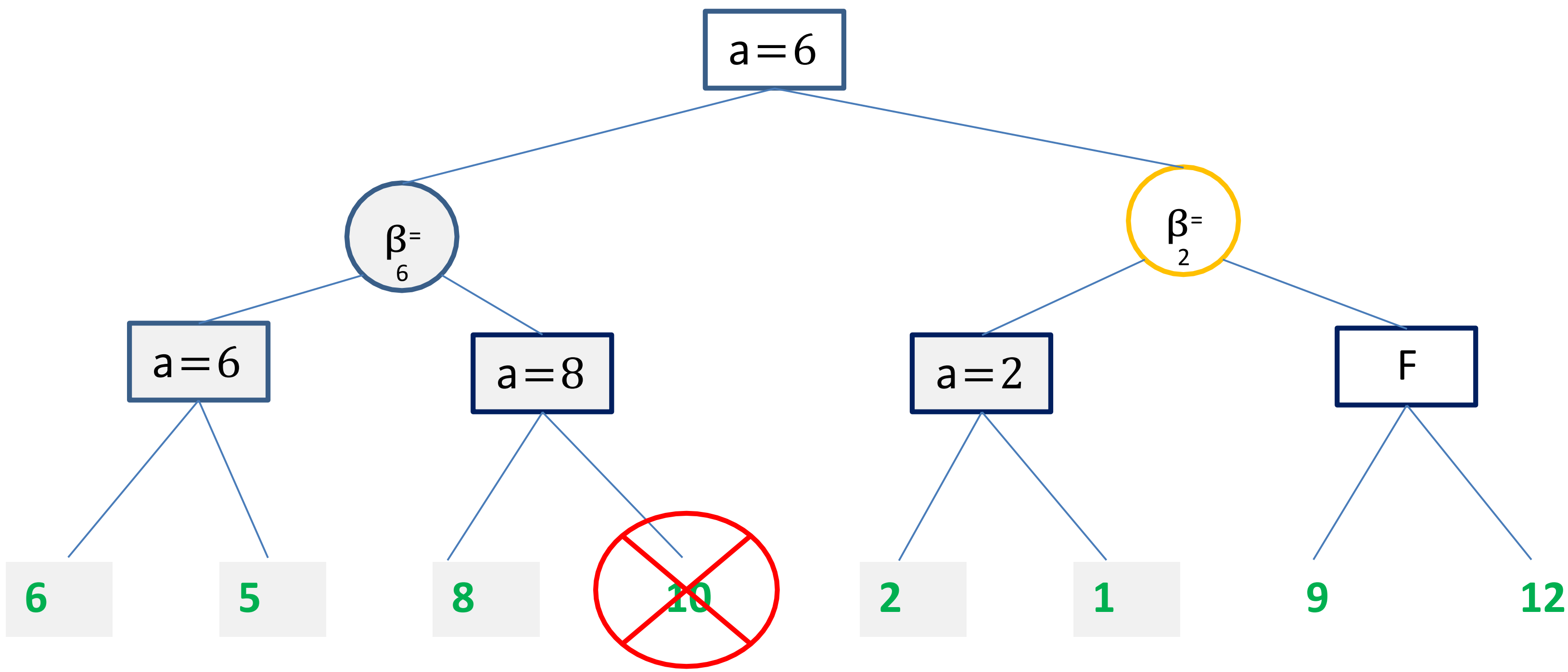


Idea –Pruning



Alpha Beta Pruning

Idea –Pruning

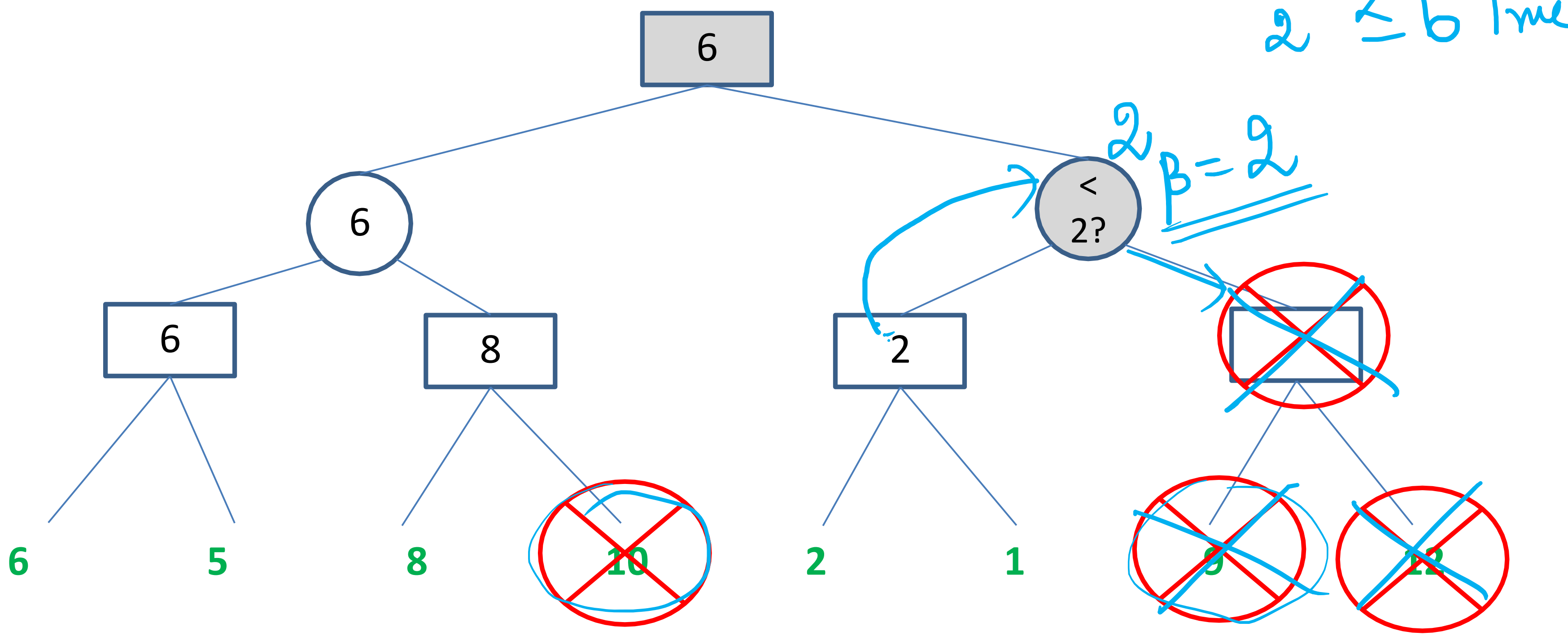


Alpha Beta Pruning

Idea – Alpha Pruning

$\beta_{curr} \leq \alpha_{parent}$

$2 \leq 6$ True



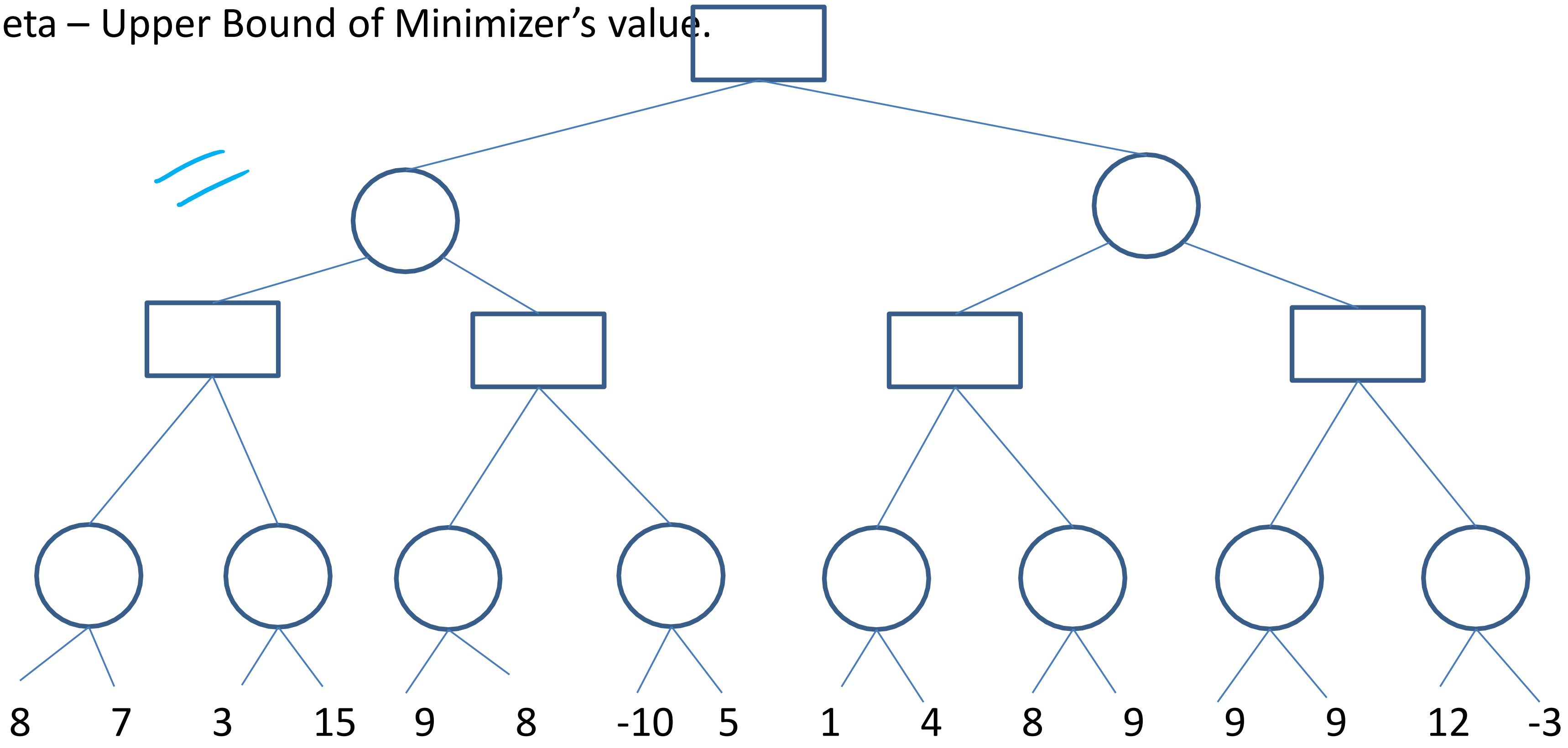
Alpha – Lower bound of Maximizer's value. Perceived value that Maximizer hopes to against a competitive Minimizer

Alpha – beta Pruning – Example -4

Do for practice.

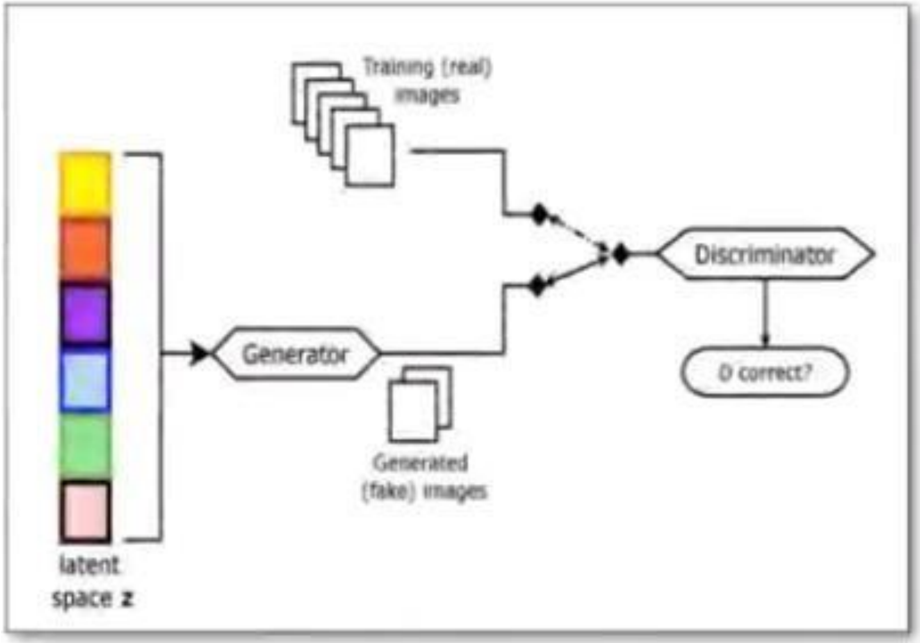
Alpha – Lower bound of Maximizer's value. Perceived value that Maximizer hopes to get with a competitive Minimizer

Beta – Upper Bound of Minimizer's value.



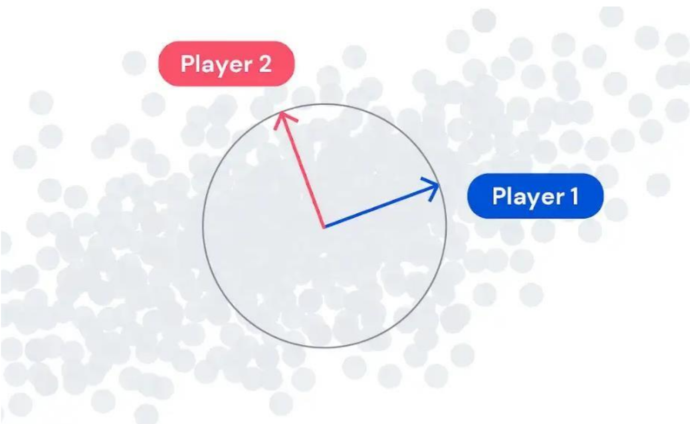
Game Playing
(Interesting Case Studies)

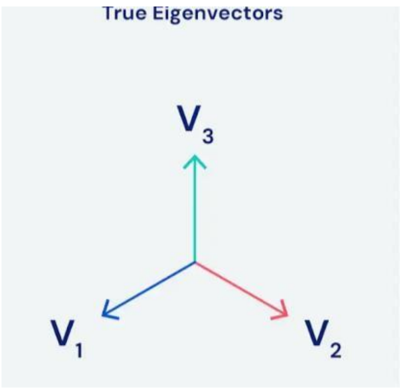
Games in Image Processing

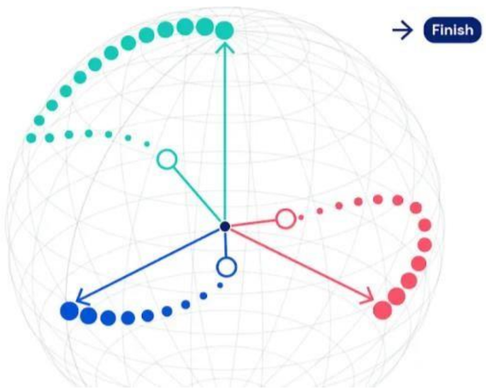


Source Credit:
[2019 - Analyzing and Improving the Image Quality of StyleGAN](#)
[Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, Timo Aila](#)
https://thispersondoesnotexist.com/

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]()

So

Gaming – Sample Question

Two Robots A and B competes to leave the maze through either of exits: E1 and E2, as 40 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.

			■	E1		E2
			■			
x		■	■	■		
A		x	B			

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

9 – 5

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

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

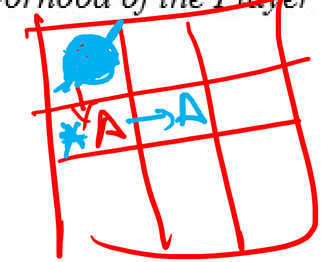
Penalty = Number of unsafe cells (blockage+ poisonous squares) with 4 degree of freedom(Up, Down, Right, Left) in the immediate neighborhood of the Player "Z's" position.

A → E_1 → $dist_1$

E_2 → $dist_2$

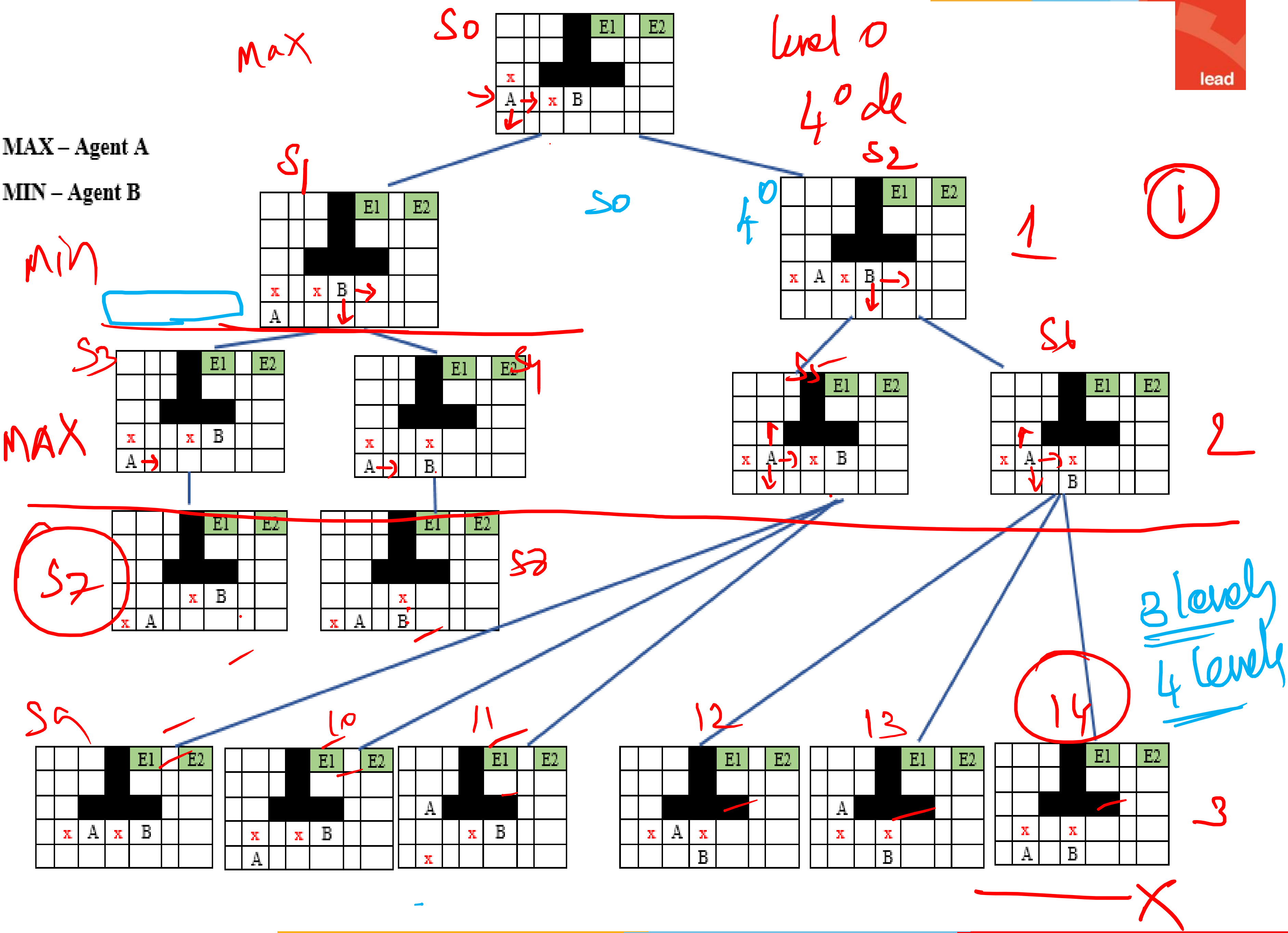
+

Penalty



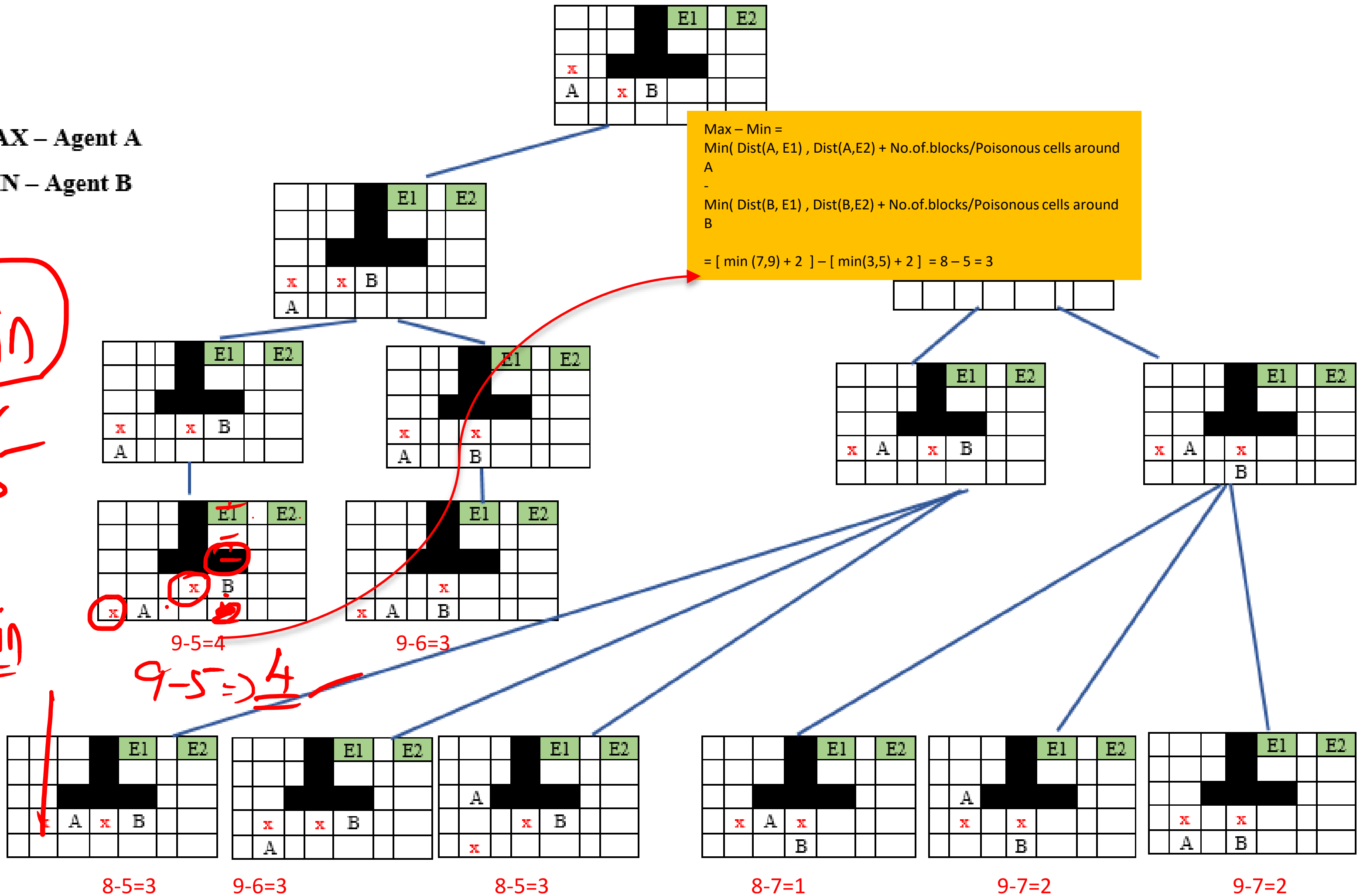
MAX – Agent A

MIN – Agent B



MAX – Agent A

MIN – Agent B



max - Min

9 5

A, E1 ⇒ 7 } min
A, E2 → 9

9-5 ⇒ 4

7+1+1
⇒ 9

B, E1 ⇒ 3
B, E2 → 5

3+1+1+1 ⇒ 5

MAX – Agent A

MIN – Agent B

Max

mi

$A \to E_1 \to 7$ } min

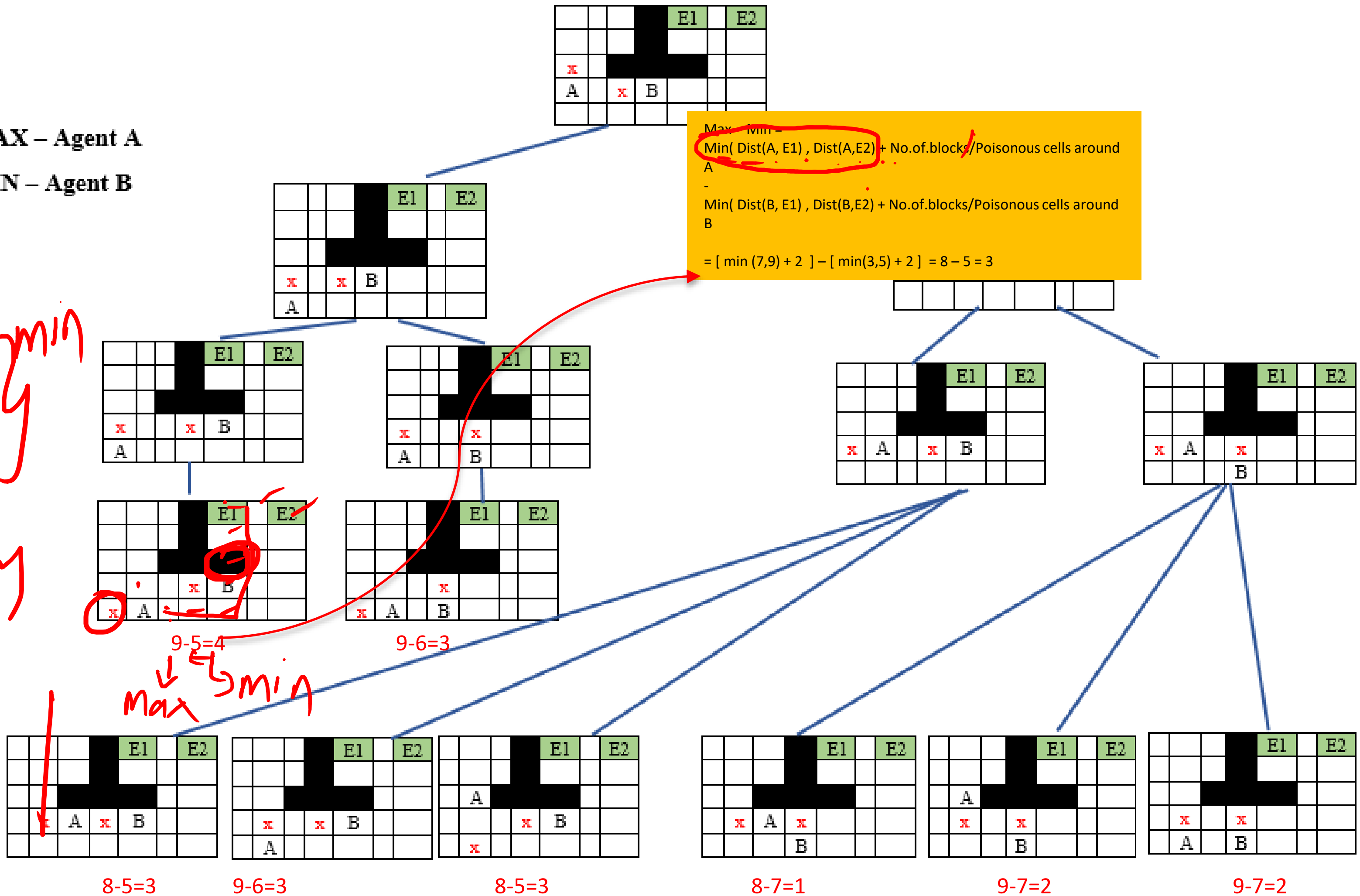
$A - E_2 \to 9$

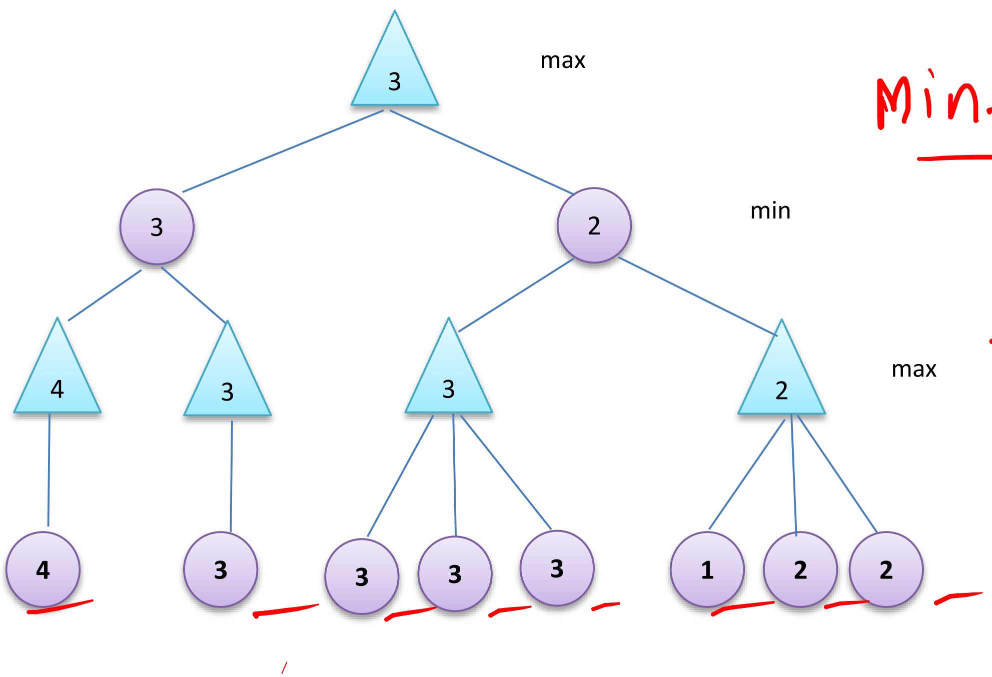
7 + Penalty

1+1

7+1+1

=> 9





Min-Max

α-β

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