AI Theory

What's next?

every person should run one existing algorithm. +

Quiz 3 : November 23, 2023

Search models: Adversesial

Games :

Diff with MDP : MDP is uncertain depending upon nature

Book	6.1, 6.2, 6.3, 6.5, 6.7

Games on the another hand, feature two intelligent agents.

Agents:

Proponents of minimum expected value. (in a trivial case, this may be considered an average case).

→ Expectimax

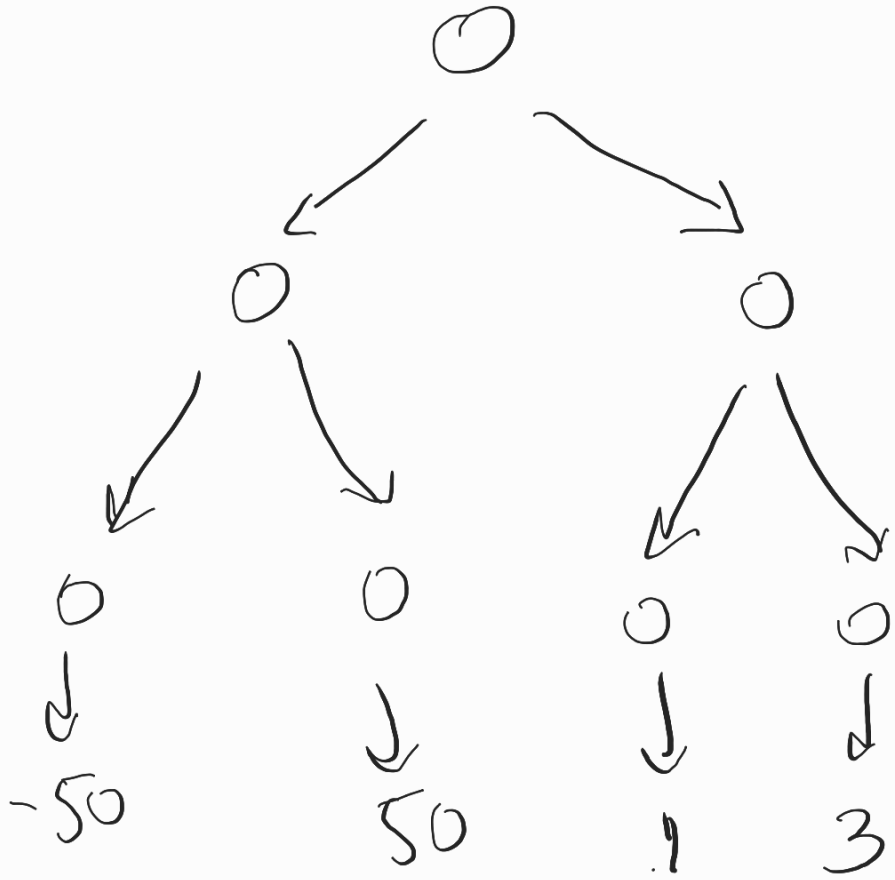
→ Minimax, Expectiminimax

→ Evaluation function

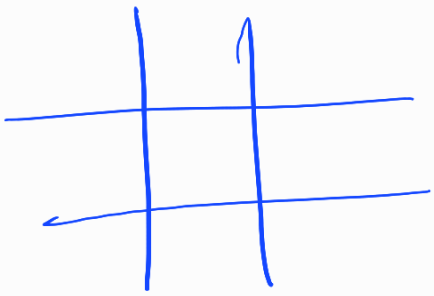
→ Alpha-beta Pruning

Idea: Game Tree:

Each node: decision tree node



"Representation that is a semantic tree in which nodes board configurations and branches denotes moves".



Total number of states: 9!

Total Player Zero Sum Game:

Players = {agent, opp}

functions: in slides.

Players: —

State s: —

Actions (s):

isEnd (s): —

Utility (s): expected value at a move.

Characteristics of game:

→ All utility is at end state.

→ Game decisions lead to outcome.

Halving Game Modeling

```
class HalvingGame.
    Start = N

    actions = ['-', '/']

    succ (state, action)
        player, no = state
    if action = '-':
```

```
        return -player, no-1
    elif action = '/':
        return -player, no//2

def utility(self, state):
    assert self.isEnd(state)
        p, n = state
        return p * +∞
```

Deterministic Policy

$$\pi_p^{(s)} \in \text{Actions}(s)$$

Stochastic Policy

$\pi_p(s,a) \in [0,1]$:

playing with probabilities

class CardGame:

S_{start} = No cards

player = [1 , -1]

Actions = [keep, discard]

<u>successor(s):</u>

player , Cards = s

card_see = choose a card to deck

action = player choice

return (-player , cards remain deck)

```
utility(s):
    player, cards = s

    if cards in order:
        and isEnd:
        return player* ∞

    return

isEnd(s):

    return 3 cards
    in sorted order
    == True
```

Game Evaluation Recurrence:

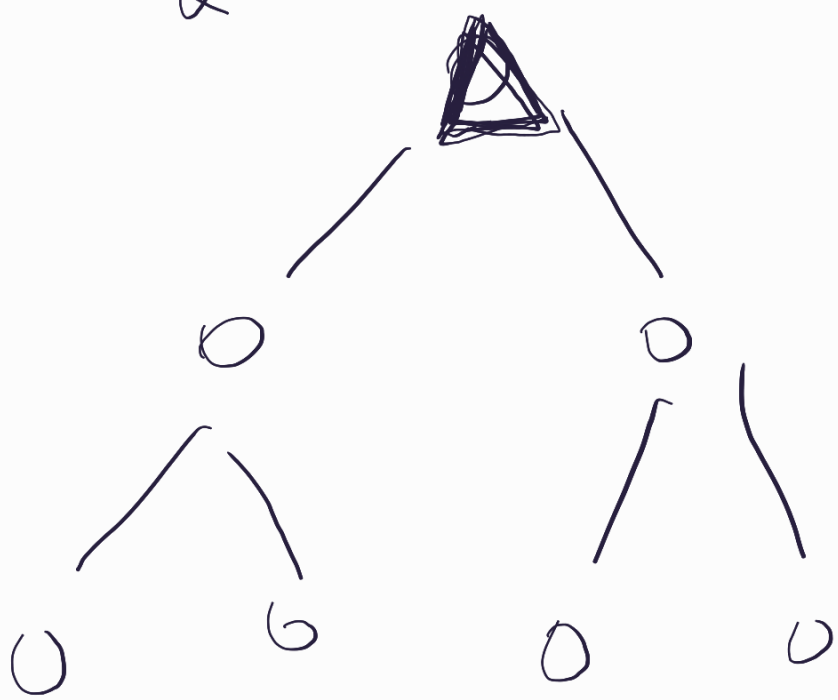
$$V_{eval}(s) = \begin{cases} \sum_{a \in actions_{agent}(s)} \pi(s,a)\, V(s') \\ \sum_{a \in actions(s)} \quad \text{same} \end{cases}$$

one for player, one for enemy

$\Rightarrow$ these equations are in book btw.

Expectimax Algorithm:

$\pi_{opp}(s,a) = \frac{1}{2}$ for $a \in actions(s)$



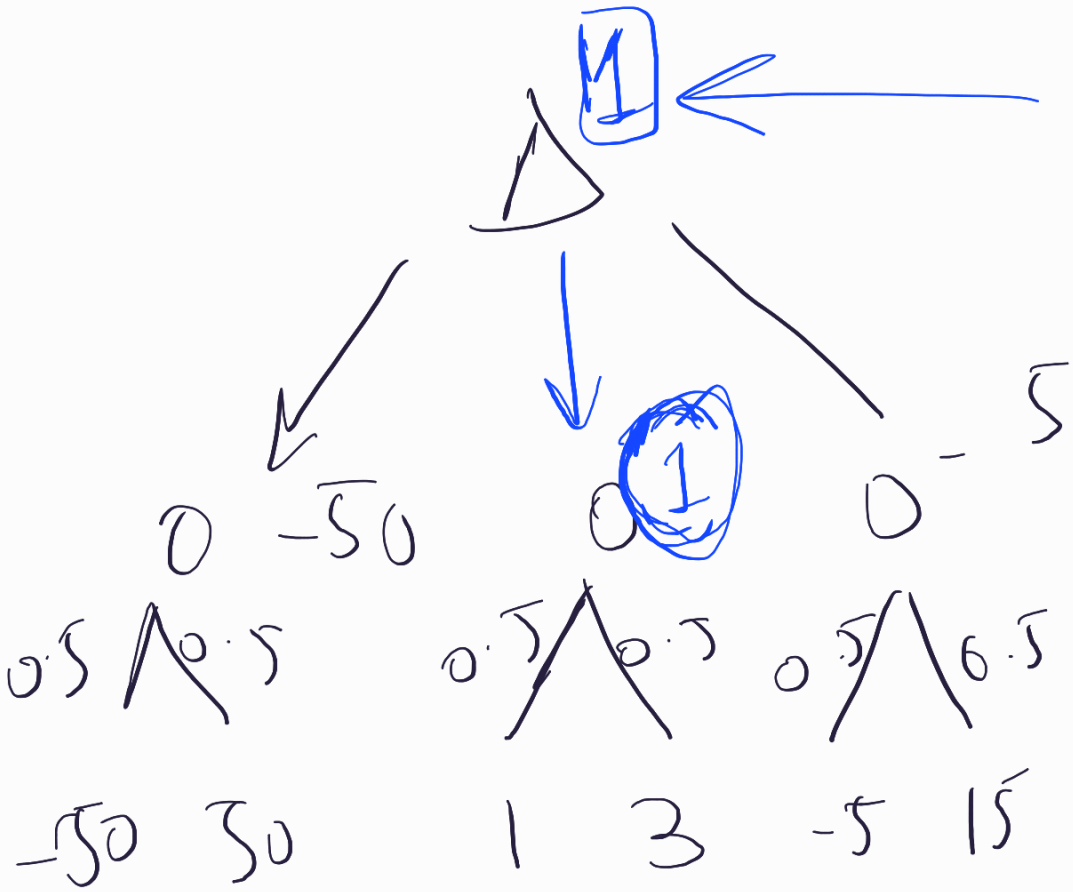
Agent will choose that move where expectation will be max. (Agent moves randomly)

$$V_{expectimax}(s,a) = \begin{cases} \sum_{a \in actions} V_{expectimax}(succ(s,a)) & \text{... Player} \\ \sum_{\cdots} \pi_{opp}(s,a)\, V_{expectimax}(succ(s,a)) & \text{... opponent} \end{cases}$$

"agent is random, opponent plays normally"

Minimax:

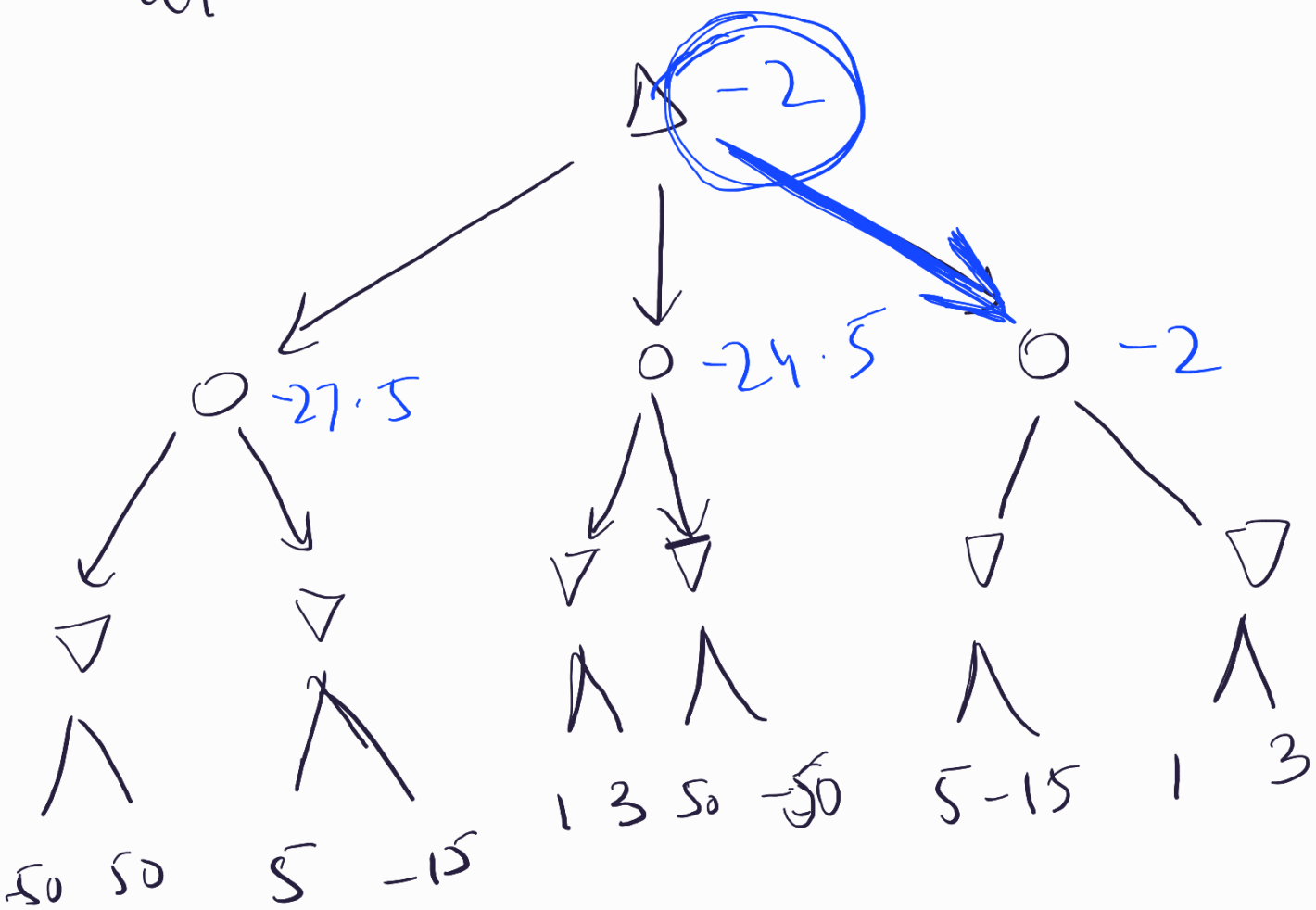
$$V_{minimax}(s) = \begin{cases} \text{Utility}(s) & \\ \sum V_{minimax}(succ(s,a)) & \text{player = agent} \\ \sum \quad '' & \text{player = opp} \end{cases}$$



Expectiminimax:

"uncertainty" acts as a third player.

$\pi_{coin}(s,a) = \frac{1}{2}$ for $a \in [0,1]$



Agent = max

Coin = Random

Opp = min

$$V(s) = \begin{cases} \text{utility}(s) \\ \max V(s') & \text{agent} \\ \min V(s') \\ \text{Coin } [0,1] \end{cases}$$

Face off in slides:

$V_{expMax} \Rightarrow V_{expMax}(7), \pi(7)$ random policy

$V_{minmax} \Rightarrow \pi(\max), \pi_{min}$

Play policies against one another:

	π_{min}	π_7
π_{max}	$V(\pi_{max}, \pi_{min})$	$V(\pi_{max}, \pi_7)$
$\pi_{expt\,min\,max}(7)$	$V(\pi_{expt\,min\,max}(7), \pi_{min})$	$V(\pi_{expt\,min\,max}(7), \pi_7)$

What is the relationship b/w these values?

Proposition: best against minmax opponent

$$V(\pi_{max}, \pi_{min}) \leq V(\pi_{max}, \pi_{min}) \text{ for all } \pi_{min}$$

Proposition: lower bound against any opponent.

$\Rightarrow$ opponent agr thori si bhi randomness leaye to expected best hai.

Proposition: not optimal if opponent is not known.

Lecture # 2

Evaluation Function:

A very weak estimate of the value $V_{minimax}(S)$.

Analogy: FutureCost(s) in search problems.

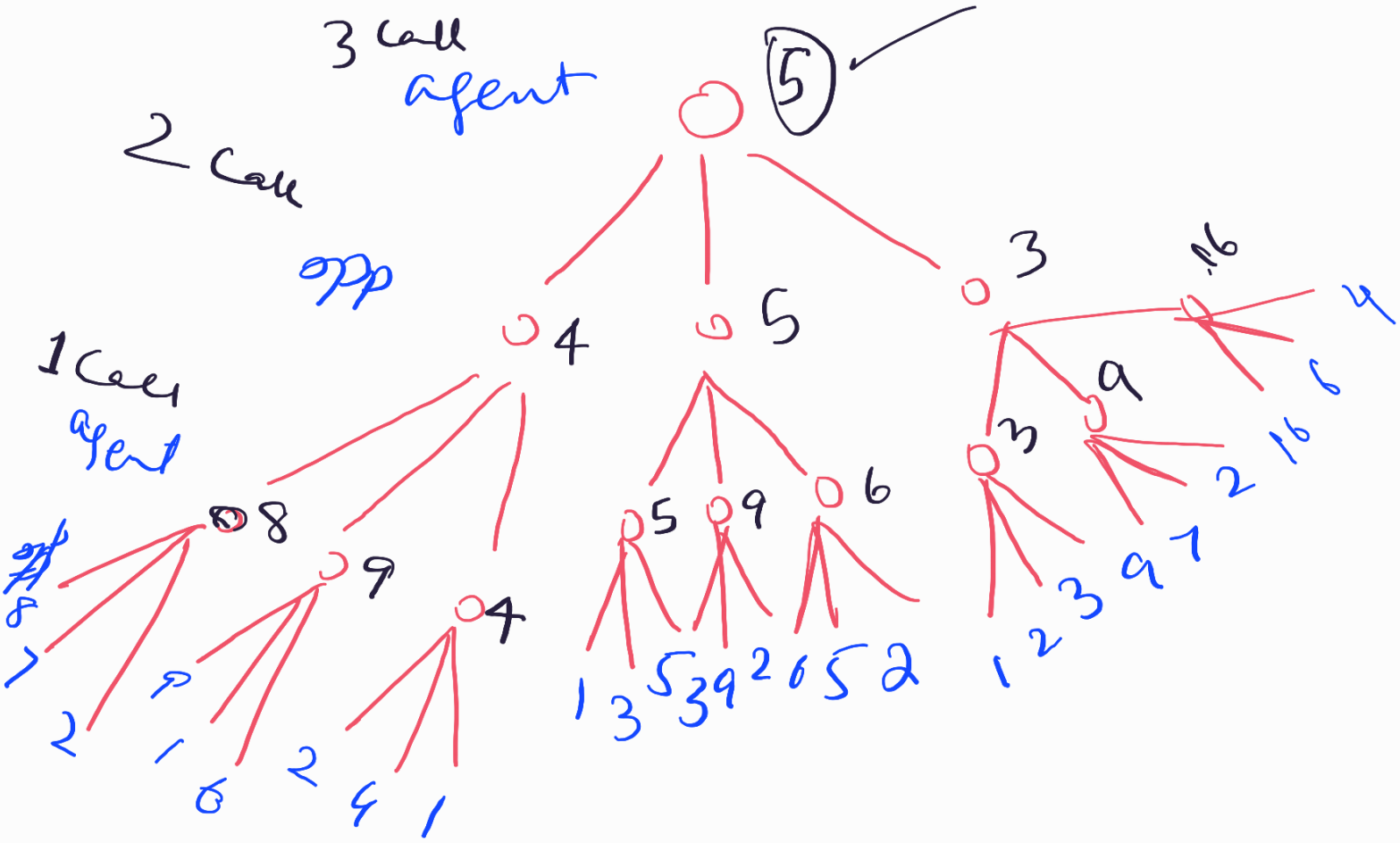
quite similar to heuristics.

Note:

Eval(s) is an estimated value, it need not be optimal.

Alpha-beta Pruning:

Idea: Eliminate un-necessary paths from the tree (literal meaning of the word "Prune")



opp. strategy = min

agent - strategy = max

Alpha-beta pruning:

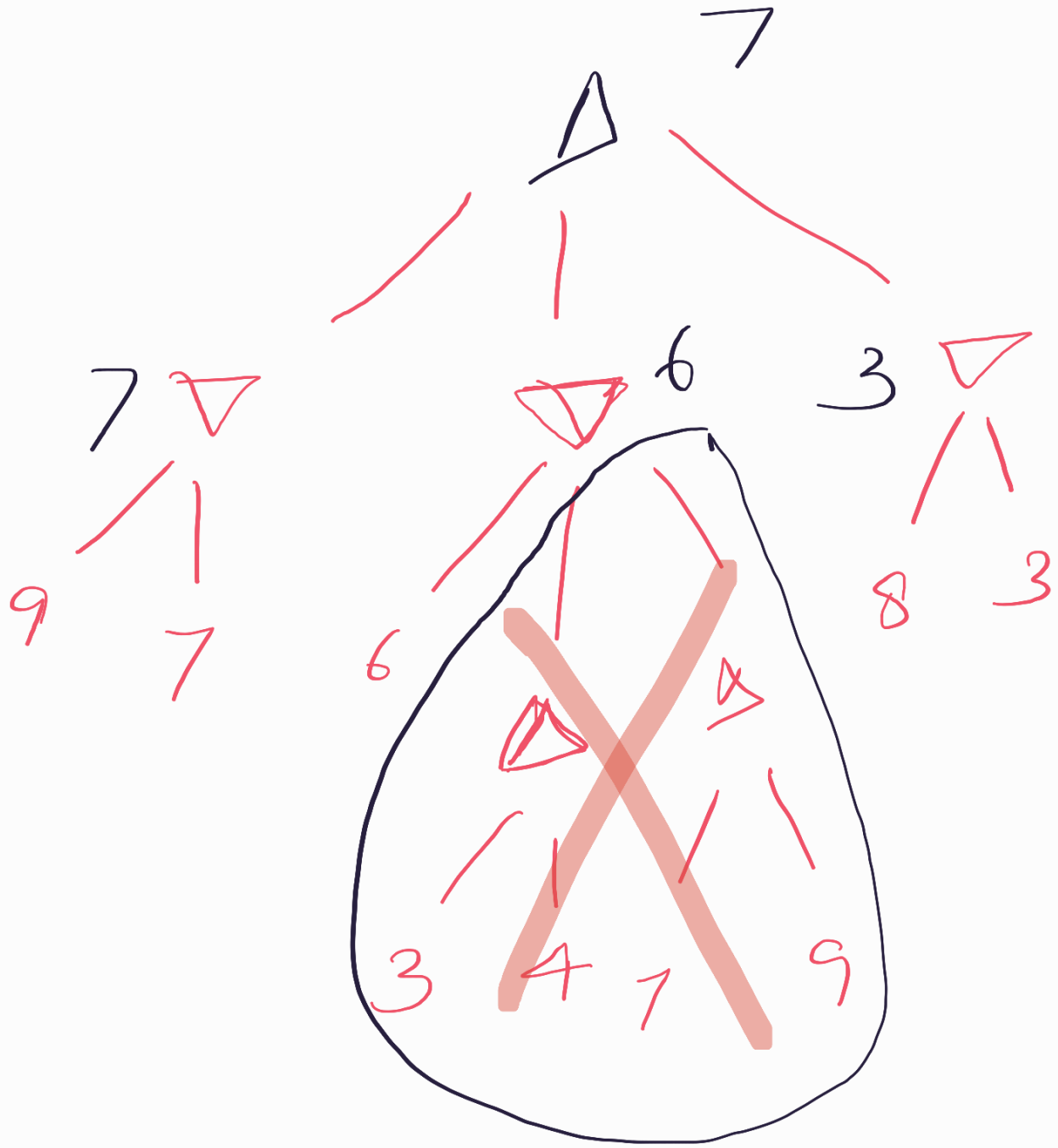
Agent turn → max

(any min-value is automatically discarded)

Opponent turn → min

(any max-value is automatically discarded)

Key idea → Pruning ⊛



Analogy:

$6 \leq 7$, any value less than 6 already should

have been selected.

If not, than that sibling tree need not be selected.