

# Dynamic Games

- 2 Player zero sum turn taking games
- Markov Games
- Extensive Form Games

# Turn-taking Games

# Turn-taking Games



Terminology

# Turn-taking Games



## Terminology

- Max and Min players

# Turn-taking Games



## Terminology

- Max and Min players
- deterministic

# Turn-taking Games



## Terminology

- Max and Min players
- deterministic
- two player

# Turn-taking Games



## Terminology

- Max and Min players
- deterministic
- two player
- zero-sum

# Turn-taking Games



## Terminology

- Max and Min players
- deterministic
- two player
- zero-sum
- perfect information

# Minimax Trees

# Minimax Trees

MDP Expectimax Tree

# Minimax Trees

MDP Expectimax Tree

Minimax Tree

# Minimax Trees

MDP Expectimax Tree

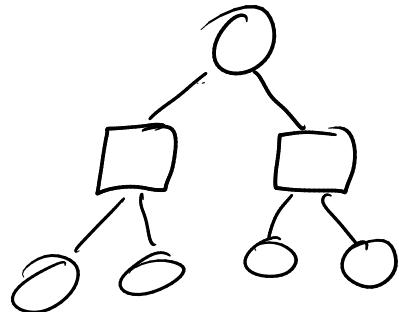
$$V(s) = \max_{a \in \mathcal{A}} (R(s, a) + \mathbb{E}[V(s')])$$

Minimax Tree

# Minimax Trees

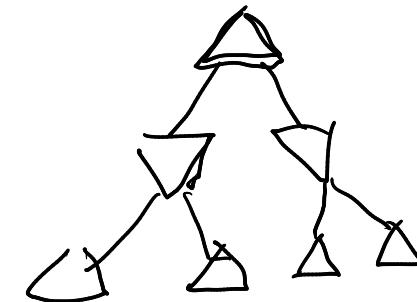
MDP Expectimax Tree

$$V(s) = \max_{a \in \mathcal{A}} (R(s, a) + \gamma \mathbb{E}[V(s')])$$

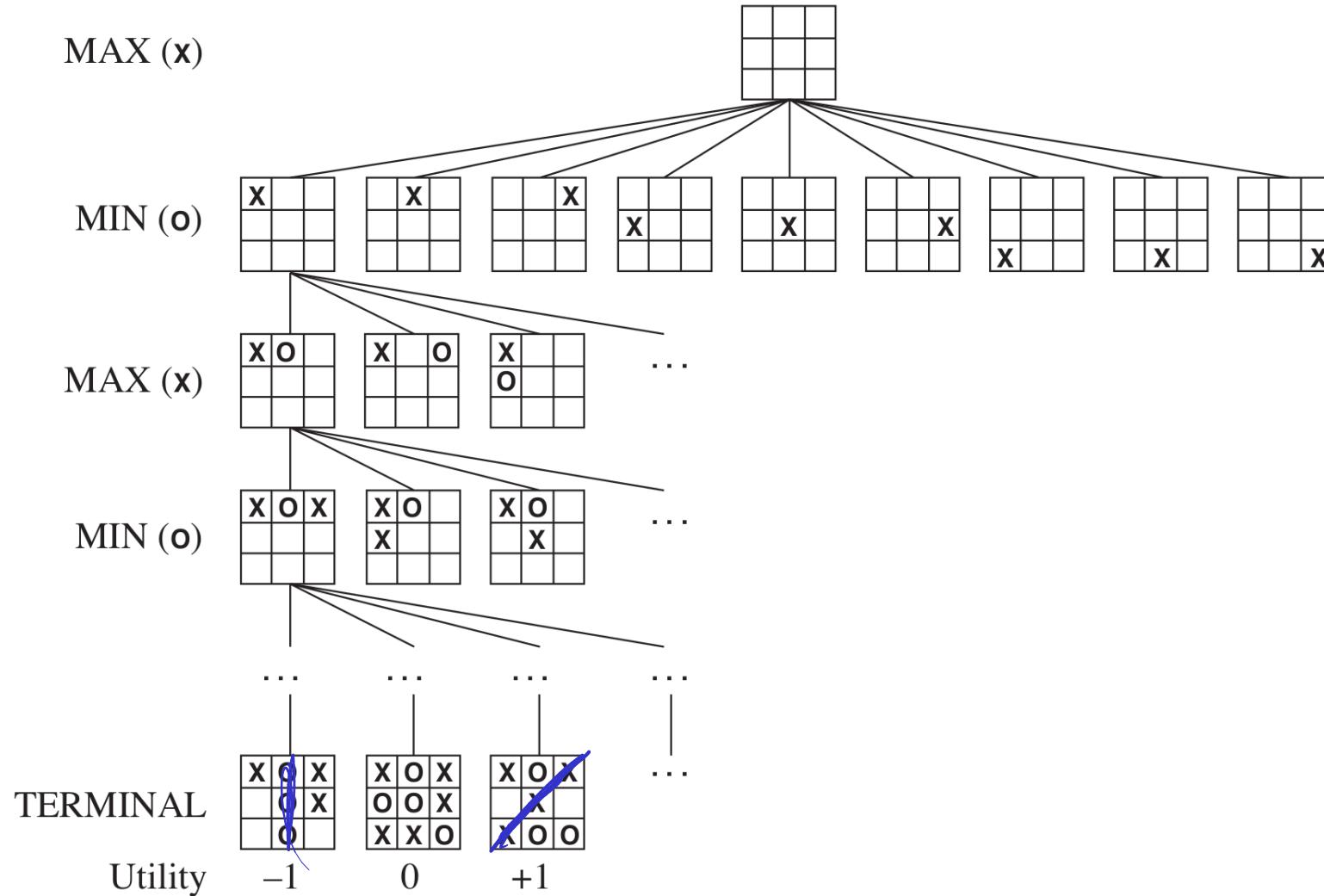
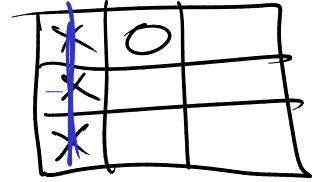


Minimax Tree

$$V(s) = \underbrace{\max_{a \in \mathcal{A}_1} (R(s, a) + \min_{a' \in \mathcal{A}_2} (R(s', a') + V(s')))}_{\text{Max}}$$



# Tic-Tac-Toe Example



# Tree Backup Example

# Tree Backup Example

---

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

---

```
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
         $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a)))$ 
    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
         $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a)))$ 
    return v
```

# Tree Backup Example

Z turns  
a bc  
~~left or right sum~~

---

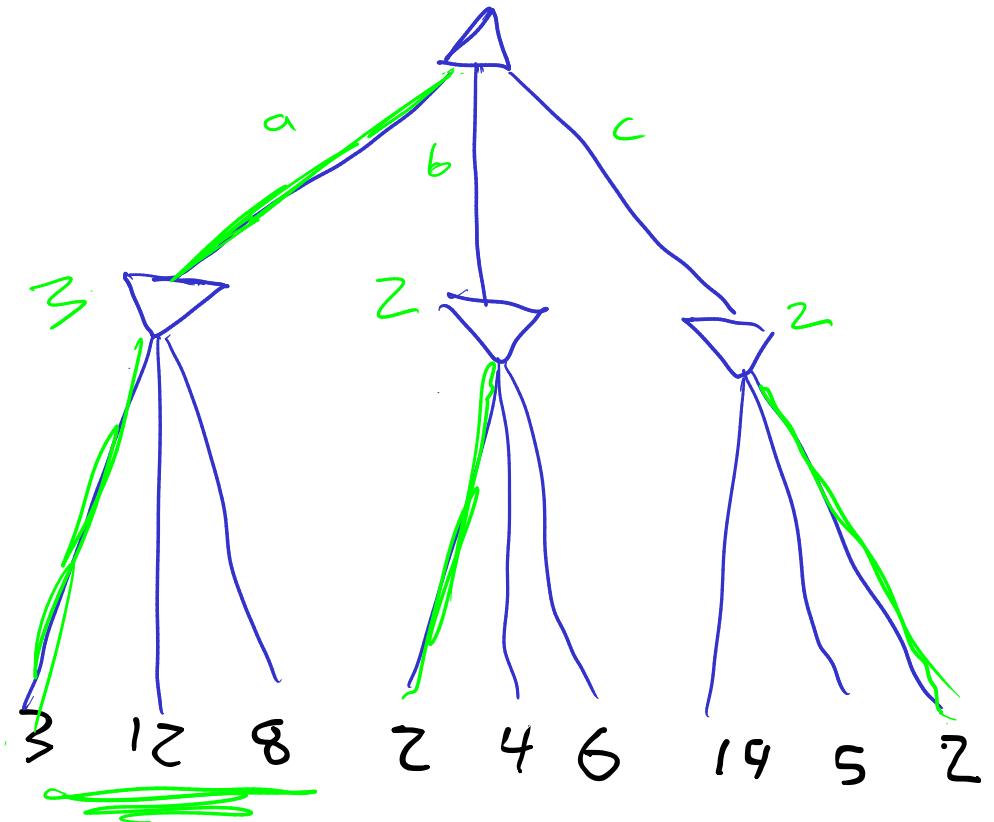
**function** MINIMAX-DECISION(*state*) **returns** *an action*  
    **return**  $\arg \max_{a \in \text{ACTIONS}(s)} \text{MIN-VALUE}(\text{RESULT}(s, a))$

---

**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*  $\leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a)))$   
    **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*  $\leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a)))$   
    **return** *v*



# Tree Backup Example

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

---

```
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
         $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a)))$ 
    return v
```

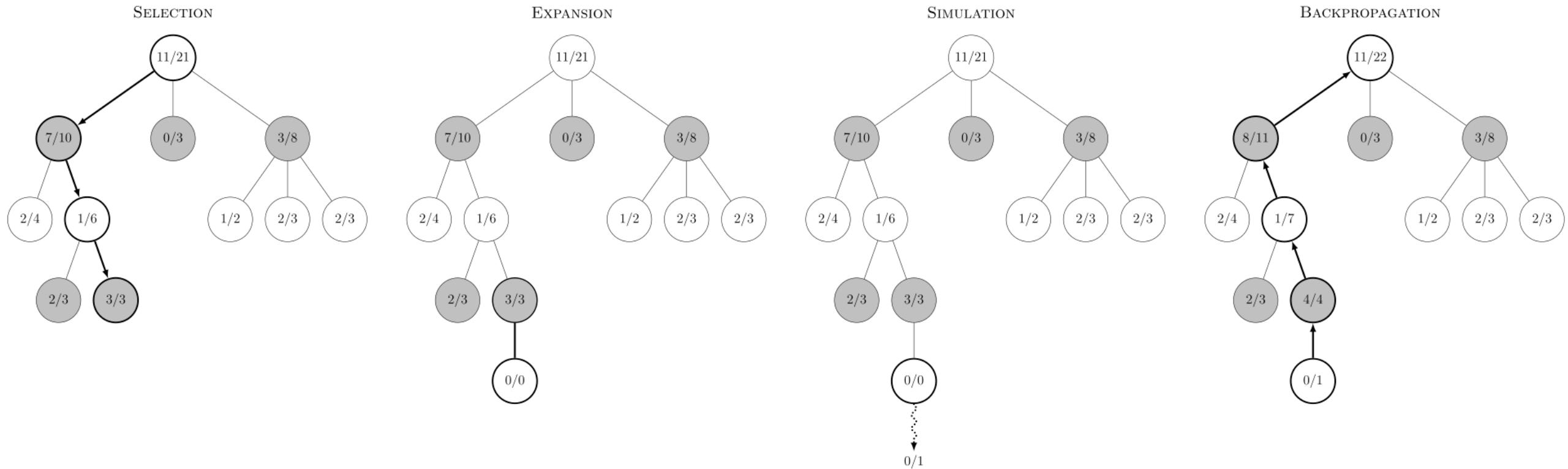
---

```
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
         $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a)))$ 
    return v
```

3 12 8 2 4 6 19 5 2

Why is this harder than an MDP? (think  
back to sparse sampling)

# MCTS for Games



# Markov Games

# Nash equilibrium at every state

$$\underset{\pi, U}{\text{minimize}} \quad \sum_{i \in \mathcal{I}} \sum_s \left( U^i(s) - Q^i(s, \pi(s)) \right)$$

subject to  $U^i(s) \geq Q^i(s, a^i, \pi^{-i}(s))$  for all  $i, s, a^i$

$$\sum_{a^i} \pi^i(a^i \mid s) = 1 \text{ for all } i, s$$

$$\pi^i(a^i \mid s) \geq 0 \text{ for all } i, s, a^i$$

.

where

$$Q^i(s, \pi(s)) = R^i(s, \pi(s)) + \gamma \sum_{s'} T(s' \mid s, \pi(s)) U^i(s')$$

# Incomplete Information: Extensive-Form Games



# Incomplete Information

Extensive-form game definition ( $h$  is a sequence of actions called a "history"):

# Incomplete Information

Extensive-form game definition ( $h$  is a sequence of actions called a "history"):

- Finite set of  $n$  players, plus the "chance" player

# Incomplete Information

Extensive-form game definition ( $h$  is a sequence of actions called a "history"):

- Finite set of  $n$  players, plus the "chance" player
- $P(h)$  (player at each history)

# Incomplete Information

Extensive-form game definition ( $h$  is a sequence of actions called a "history"):

- Finite set of  $n$  players, plus the "chance" player
- $P(h)$  (player at each history)
- $A(h)$  (set of actions at each history)

# Incomplete Information

Extensive-form game definition ( $h$  is a sequence of actions called a "history"):

- Finite set of  $n$  players, plus the "chance" player
- $P(h)$  (player at each history)
- $A(h)$  (set of actions at each history)
- $I(h)$  (information set that each history maps to)

# Incomplete Information

Extensive-form game definition ( $h$  is a sequence of actions called a "history"):

- Finite set of  $n$  players, plus the "chance" player
- $P(h)$  (player at each history)
- $A(h)$  (set of actions at each history)
- $I(h)$  (information set that each history maps to)
- $U(h)$  (payoff for each leaf node in the game tree)

# King-Ace Poker Example

# King-Ace Poker Example

- 4 Cards: 2 Aces, 2 Kings

# King-Ace Poker Example

- 4 Cards: 2 Aces, 2 Kings
- Each player is dealt a card

# King-Ace Poker Example

- 4 Cards: 2 Aces, 2 Kings
- Each player is dealt a card
- P1 can either *raise* ( $r$ ) the payoff  
to 2 points or *check* ( $k$ ) the  
payoff at 1 point

# King-Ace Poker Example

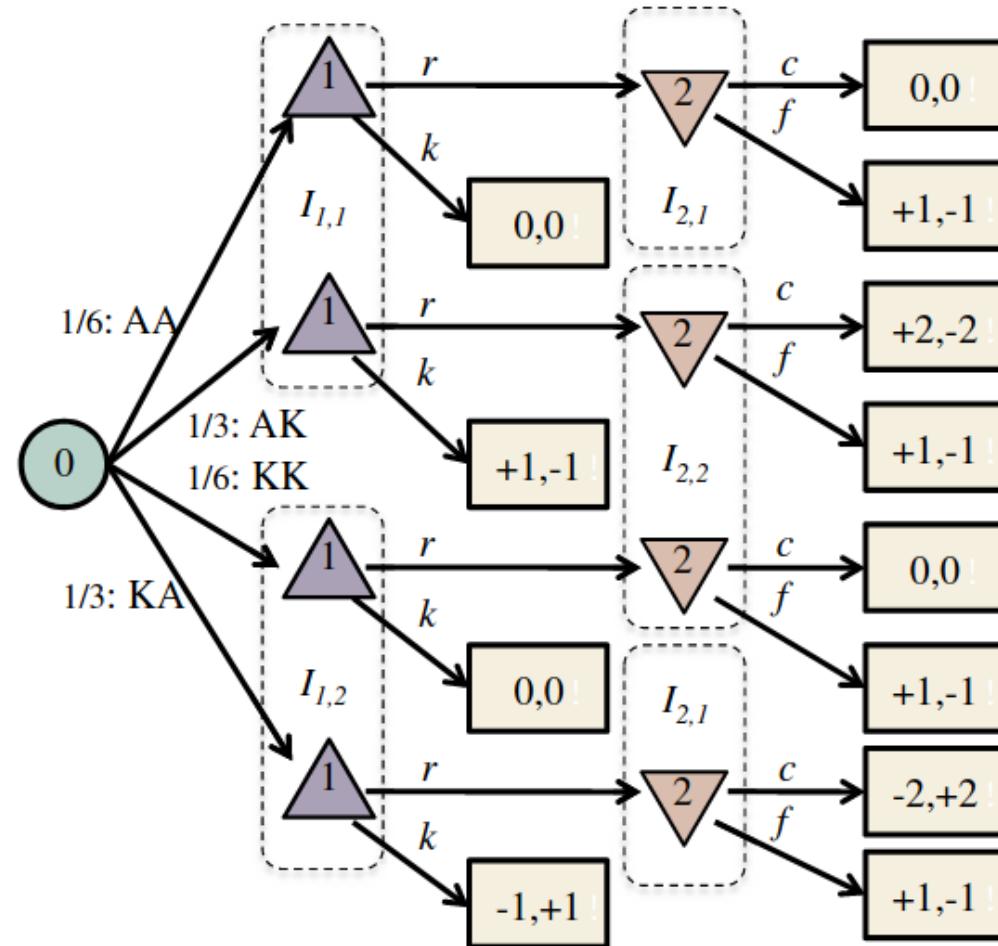
- 4 Cards: 2 Aces, 2 Kings
- Each player is dealt a card
- P1 can either *raise* ( $r$ ) the payoff to 2 points or *check* ( $k$ ) the payoff at 1 point
- If P1 raises, P2 can either *call* ( $c$ ) Player 1's bet, or *fold* ( $f$ ) the payoff back to 1 point

# King-Ace Poker Example

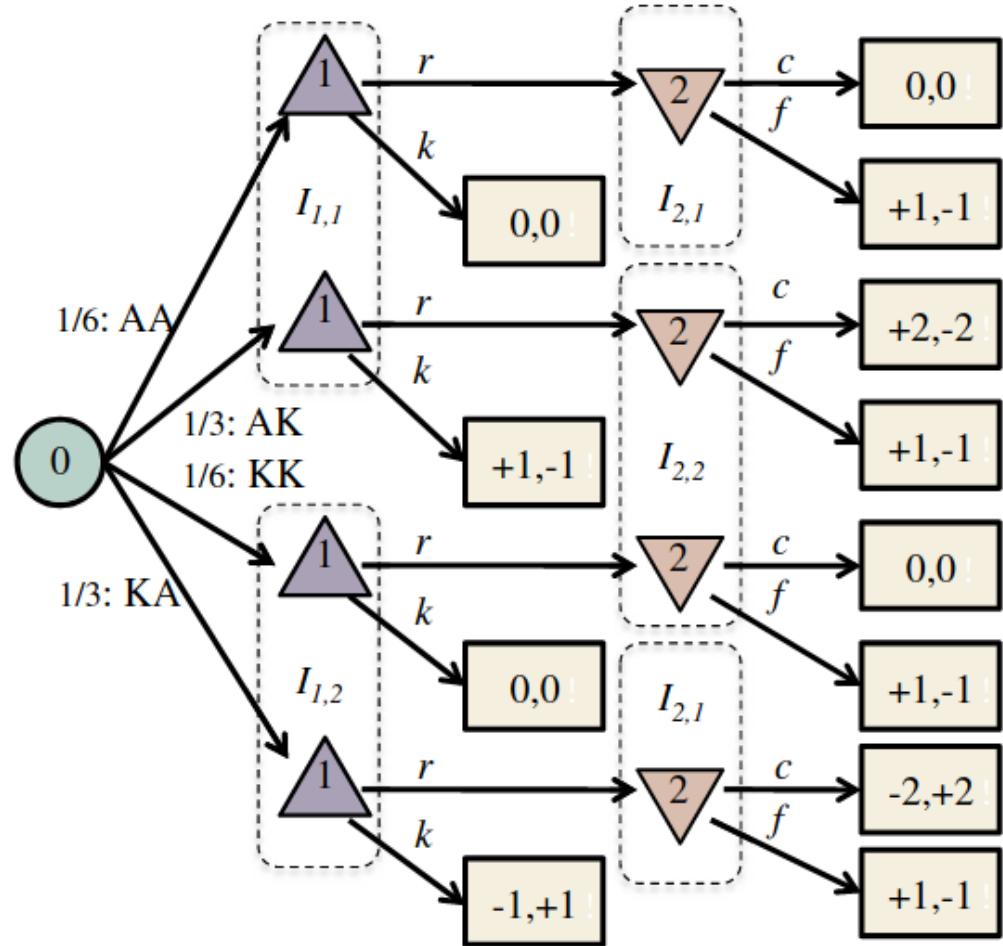
- 4 Cards: 2 Aces, 2 Kings
- Each player is dealt a card
- P1 can either *raise* ( $r$ ) the payoff to 2 points or *check* ( $k$ ) the payoff at 1 point
- If P1 raises, P2 can either *call* ( $c$ ) Player 1's bet, or *fold* ( $f$ ) the payoff back to 1 point
- The highest card wins

# King-Ace Poker Example

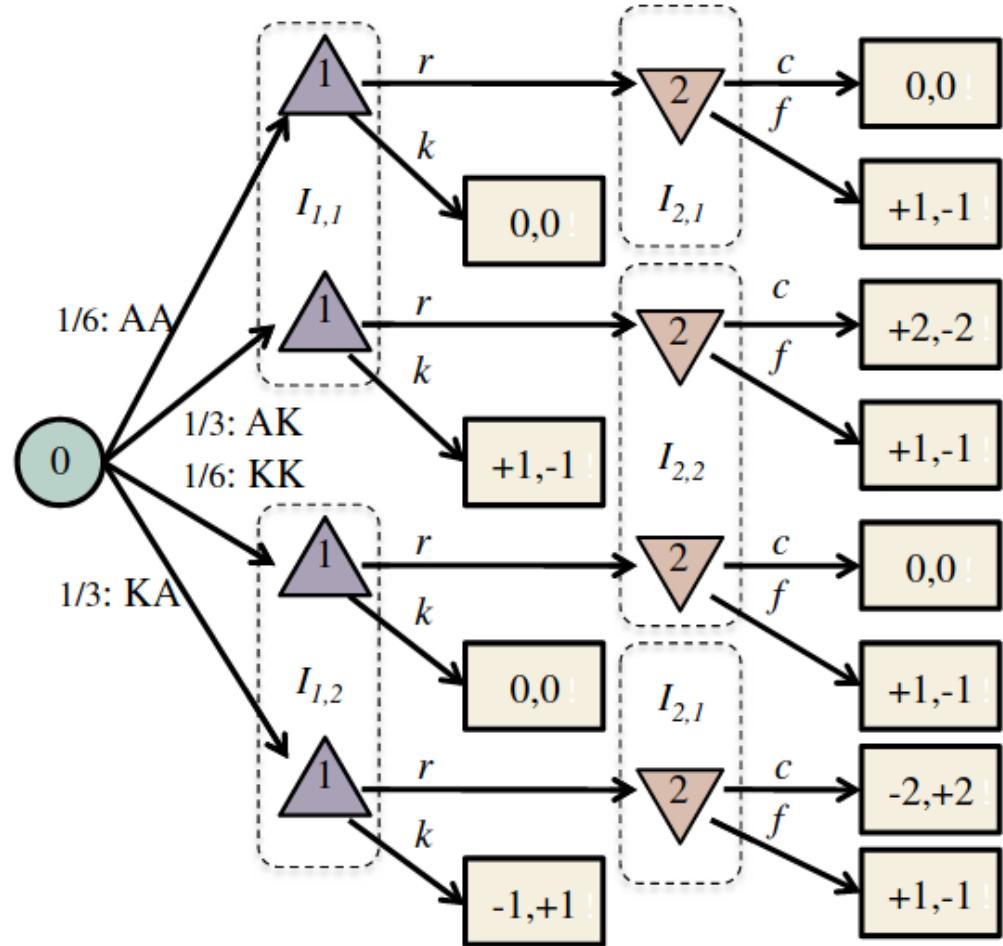
- 4 Cards: 2 Aces, 2 Kings
- Each player is dealt a card
- P1 can either *raise* ( $r$ ) the payoff to 2 points or *check* ( $k$ ) the payoff at 1 point
- If P1 raises, P2 can either *call* ( $c$ ) Player 1's bet, or *fold* ( $f$ ) the payoff back to 1 point
- The highest card wins



# Extensive to Matrix Form

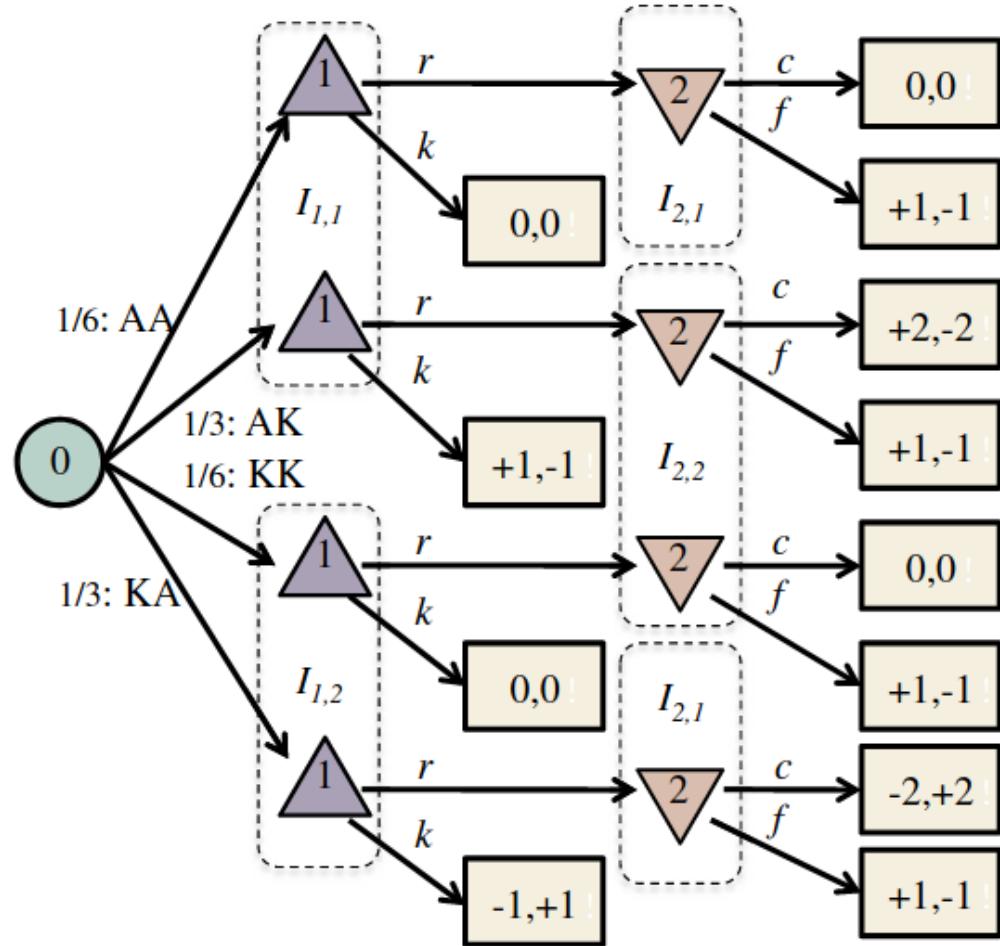


# Extensive to Matrix Form



	$2:cc$	$2:cf$	$2:ff$	$2:fc$
$1:rr$	0	$-1/6$	1	$7/6$
$1:kr$	$-1/3$	$-1/6$	$5/6$	$2/3$
$1:rk$	$1/3$	0	$1/6$	$1/2$
$1:kk$	0	0	0	0

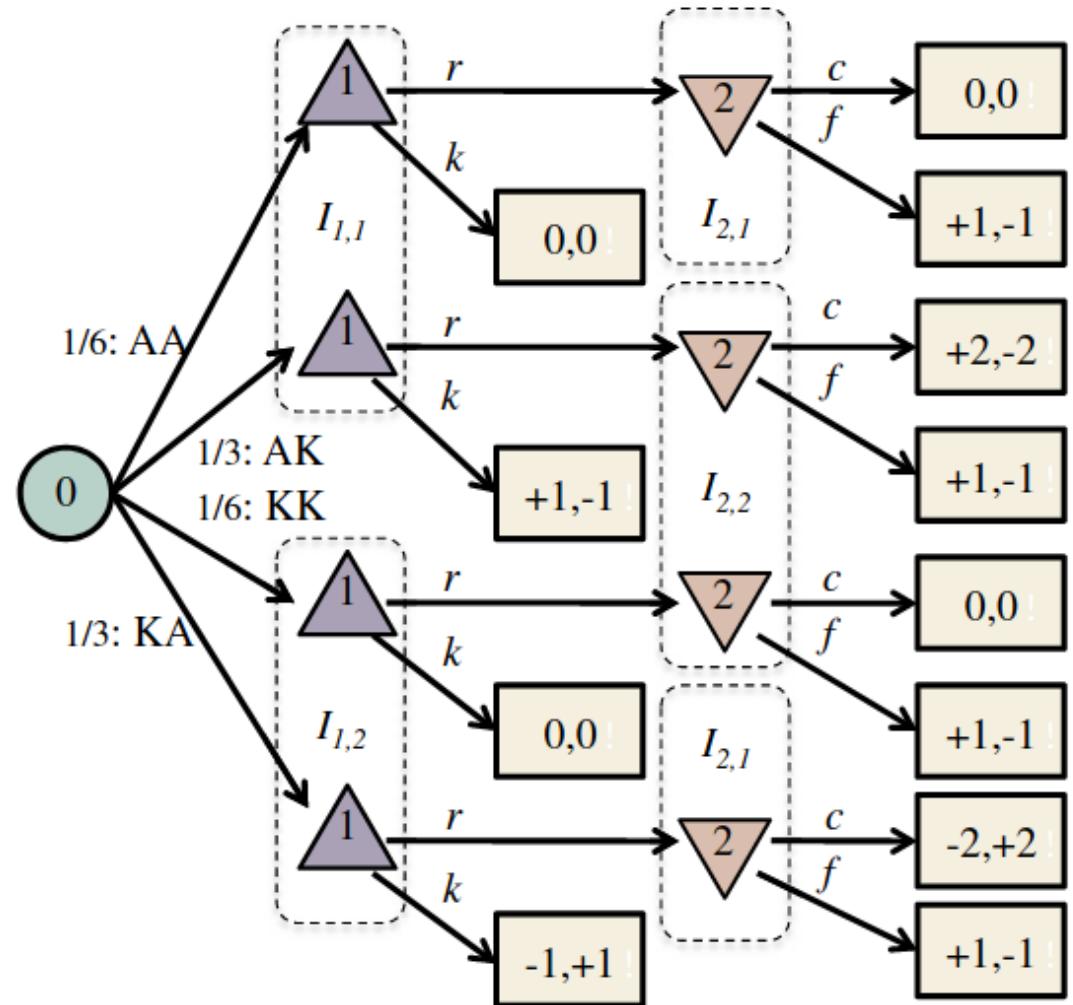
# Extensive to Matrix Form



	$2:cc$	$2:cf$	$2:ff$	$2:fc$
$1:rr$	0	$-1/6$	1	$7/6$
$1:kr$	$-1/3$	$-1/6$	$5/6$	$2/3$
$1:rk$	$1/3$	0	$1/6$	$1/2$
$1:kk$	0	0	0	0

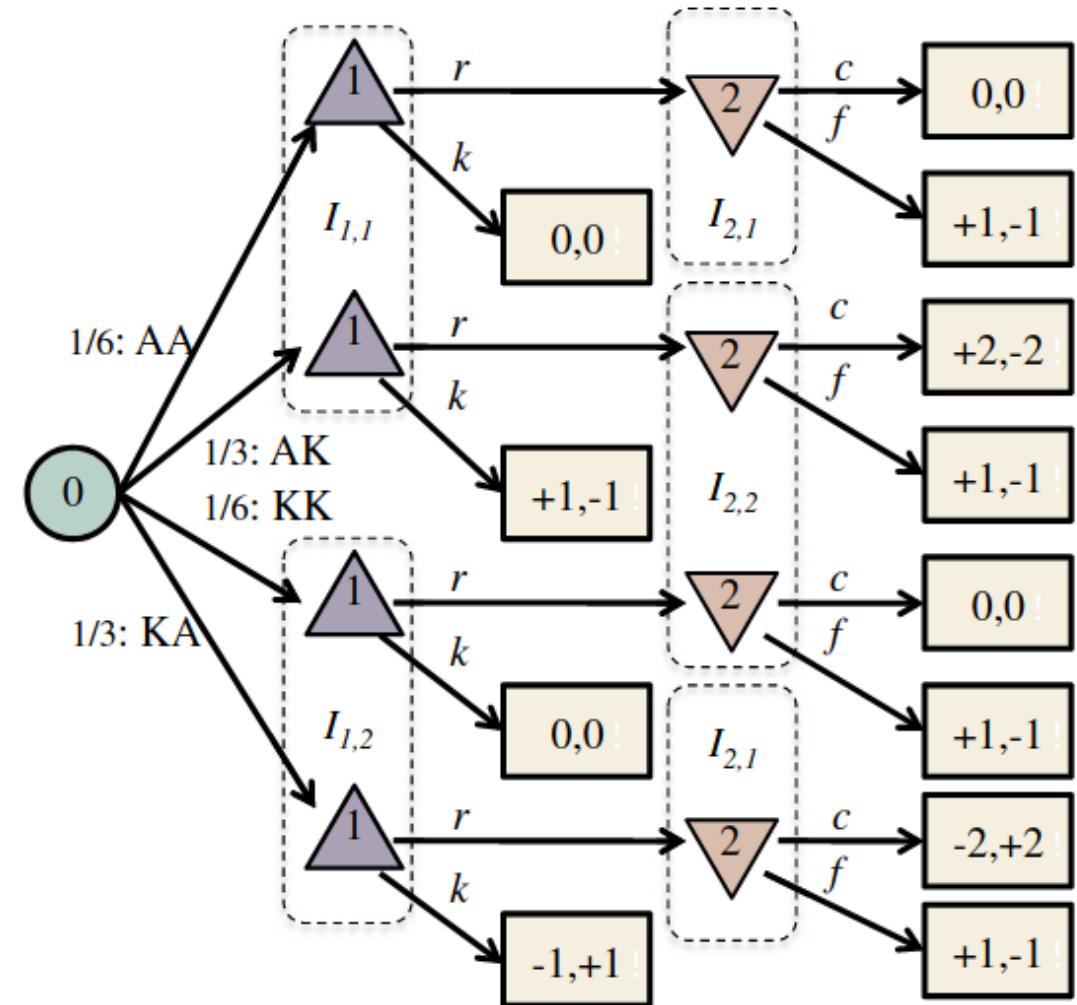
Exponential in number of info states!

# Counterfactual Regret Minimization



# Counterfactual Regret Minimization

- 1: Initialize cumulative regret tables:  $\forall I, r_I[a] \leftarrow 0$ .
- 2: Initialize cumulative strategy tables:  $\forall I, s_I[a] \leftarrow 0$ .
- 3: Initialize initial profile:  $\sigma^1(I, a) \leftarrow 1/|A(I)|$



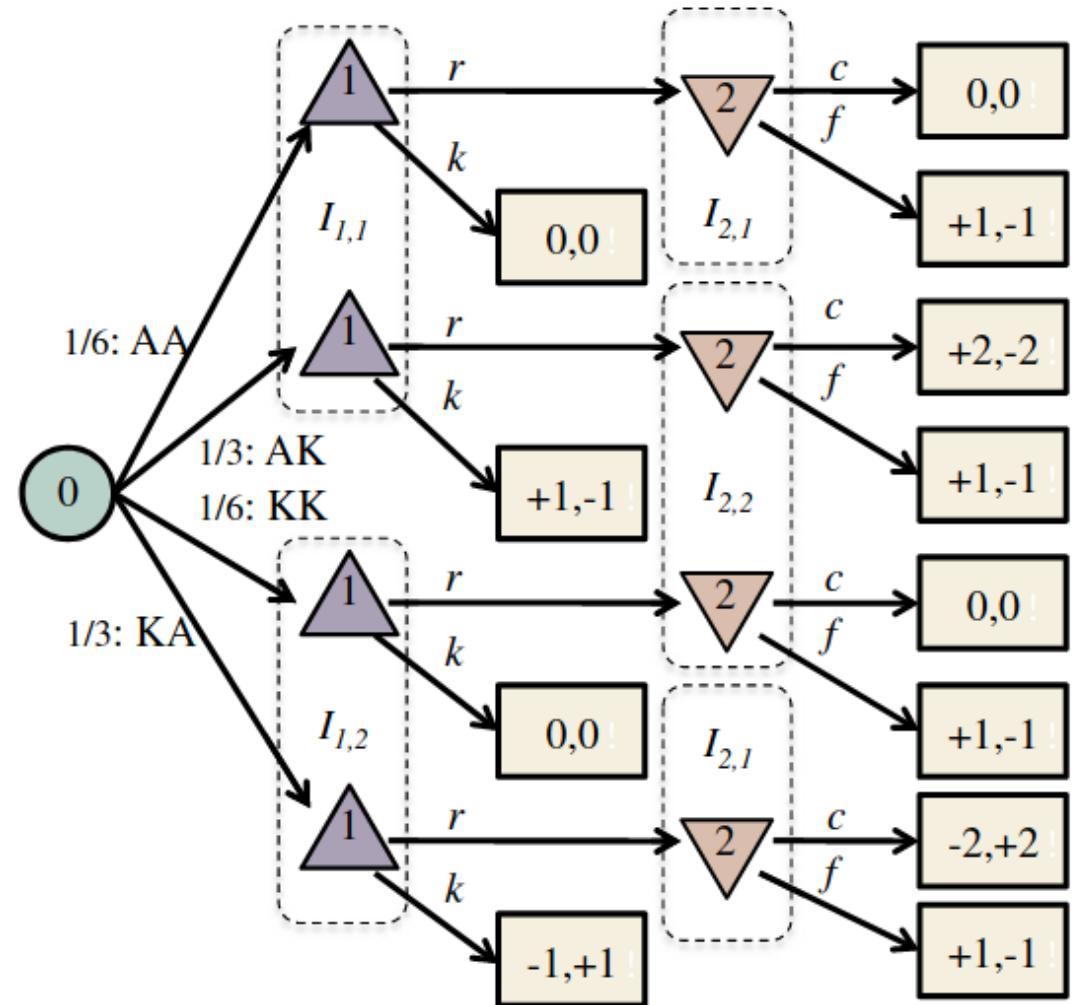
# Counterfactual Regret Minimization

- 1: Initialize cumulative regret tables:  $\forall I, r_I[a] \leftarrow 0$ .
- 2: Initialize cumulative strategy tables:  $\forall I, s_I[a] \leftarrow 0$ .
- 3: Initialize initial profile:  $\sigma^1(I, a) \leftarrow 1/|A(I)|$

```

32: function Solve():
33:   for  $t = \{1, 2, 3, \dots, T\}$  do
34:     for  $i \in \{1, 2\}$  do
35:       CFR( $\emptyset, i, t, 1, 1$ )
36:     end for
37:   end for

```



# Counterfactual Regret Minimization

```

1: Initialize cumulative regret tables:  $\forall I, r_I[a] \leftarrow 0$ .
2: Initialize cumulative strategy tables:  $\forall I, s_I[a] \leftarrow 0$ .
3: Initialize initial profile:  $\sigma^1(I, a) \leftarrow 1/|A(I)|$ 

5: function CFR( $h, i, t, \pi_1, \pi_2$ ):
6:   if  $h$  is terminal then
7:     return  $u_i(h)$ 
8:   else if  $h$  is a chance node then
9:     Sample a single outcome  $a \sim \sigma_c(h, a)$ 
10:    return CFR( $ha, i, t, \pi_1, \pi_2$ )
11:   end if
12:   Let  $I$  be the information set containing  $h$ .
13:    $v_\sigma \leftarrow 0$ 
14:    $v_{\sigma_{I \rightarrow a}}[a] \leftarrow 0$  for all  $a \in A(I)$ 
15:   for  $a \in A(I)$  do
16:     if  $P(h) = 1$  then
17:        $v_{\sigma_{I \rightarrow a}}[a] \leftarrow \text{CFR}(ha, i, t, \sigma^t(I, a) \cdot \pi_1, \pi_2)$ 
18:     else if  $P(h) = 2$  then
19:        $v_{\sigma_{I \rightarrow a}}[a] \leftarrow \text{CFR}(ha, i, t, \pi_1, \sigma^t(I, a) \cdot \pi_2)$ 
20:     end if
21:      $v_\sigma \leftarrow v_\sigma + \sigma^t(I, a) \cdot v_{\sigma_{I \rightarrow a}}[a]$ 
22:   end for
23:   if  $P(h) = i$  then
24:     for  $a \in A(I)$  do
25:        $r_I[a] \leftarrow r_I[a] + \pi_{-i} \cdot (v_{\sigma_{I \rightarrow a}}[a] - v_\sigma)$ 
26:        $s_I[a] \leftarrow s_I[a] + \pi_i \cdot \sigma^t(I, a)$ 
27:     end for
28:      $\sigma^{t+1}(I) \leftarrow \text{regret-matching values}$ 
29:   end if
30:   return  $v_\sigma$ 

```

