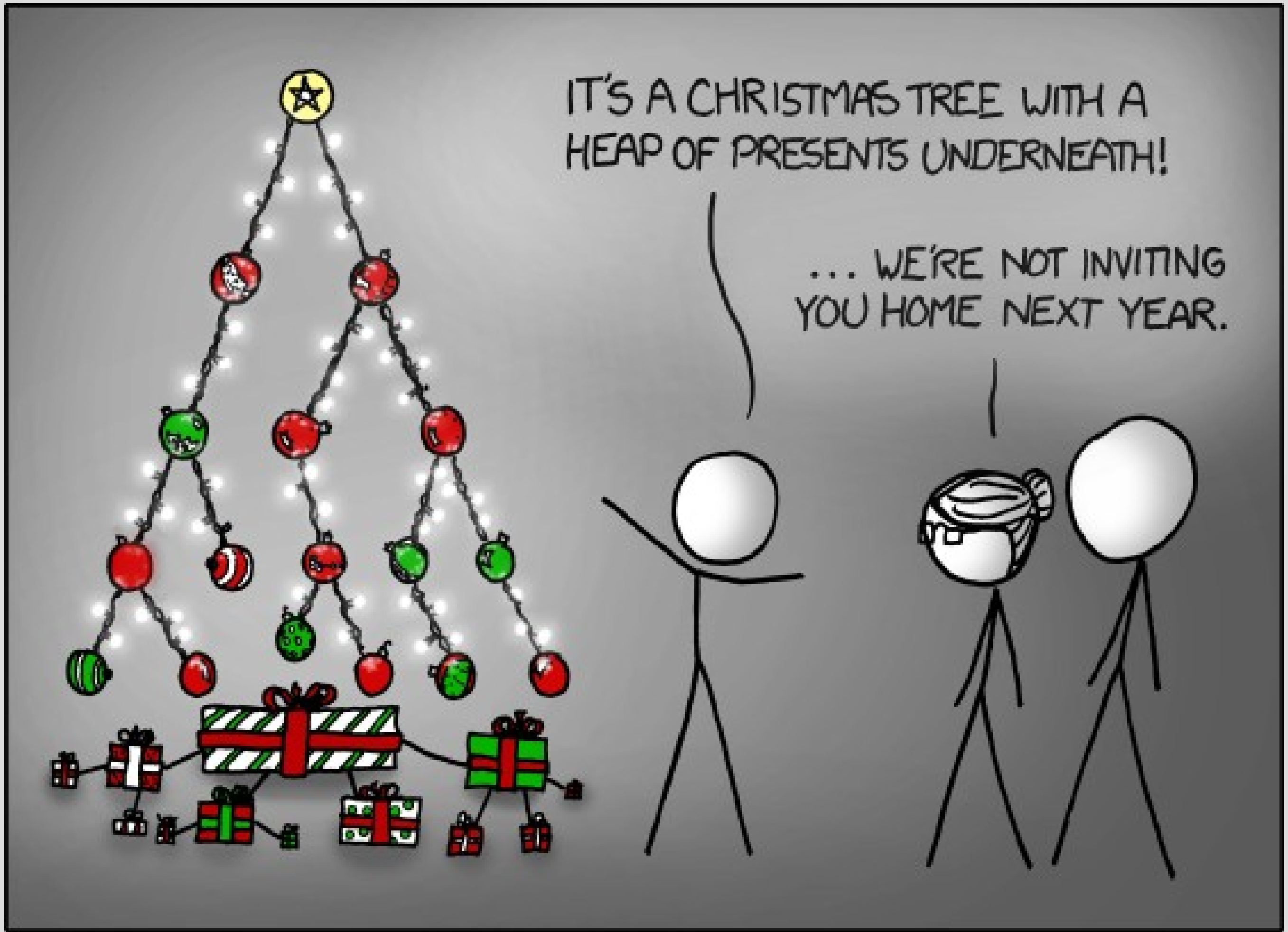
More on games (Ch. 5.4-5.7)


IT'S A CHRISTMAS TREE WITH A HEAP OF PRESENTS UNDERNEATH!
... WE'RE NOT INVITING YOU HOME NEXT YEAR.

Announcements

HW3 posted, due Wednesday after break

Midterm will be on “gradescope” (got an email from them... signup optional)

Forward pruning

You can also save time searching by using “expert knowledge” about the problem

For example, in both Go and Chess the start of the game has been very heavily analyzed over the years

There is no reason to redo this search every time at the start of the game, instead we can just look up the “best” response

Random games

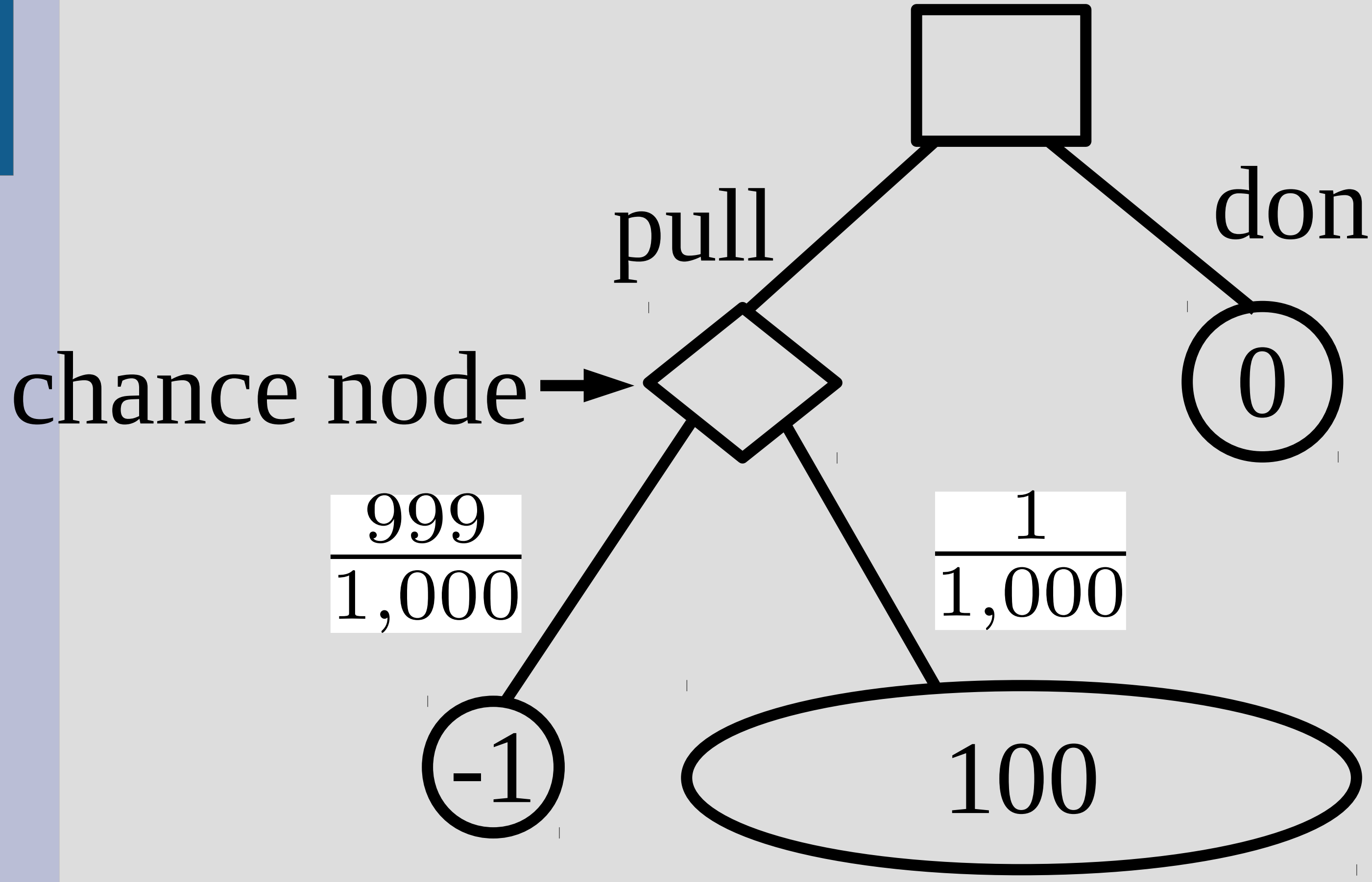
If we are playing a “game of chance”, we can add chance nodes to the search tree

Instead of either player picking max/min, it takes the expected value of its children

This expected value is then passed up to the parent node which can choose to min/max this chance (or not)

Random games

Here is a simple slot machine example:

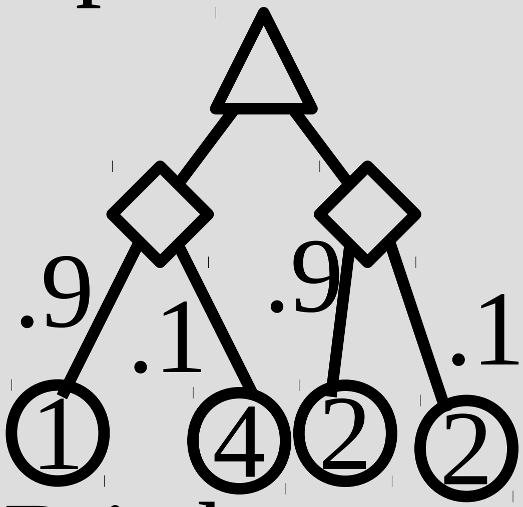


V(chance) = $-1 \cdot \frac{999}{1,000} + 100 \cdot \frac{1}{1,000} = -0.899$

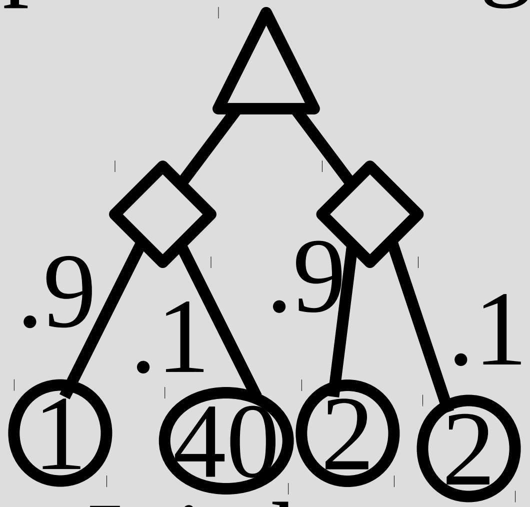
Random games

You might need to modify your mid-state evaluation if you add chance nodes

Minimax just cares about the largest/smallest, but expected value is an implicit average:



R is better



L is better

Random games

Some partially observable games (i.e. card games) can be searched with chance nodes

As there is a high degree of chance, often it is better to just assume full observability (i.e. you know the order of cards in the deck)

Then find which actions perform best over all possible chance outcomes (i.e. all possible deck orderings)

Random games

For example in blackjack, you can see what cards have been played and a few of the current cards in play

You then compute all possible decks that could lead to the cards in play (and used cards)

Then find the value of all actions (hit or stand) averaged over all decks (assumed equal chance of possible decks happening)

Random games

If there are too many possibilities for all the chance outcomes to “average them all”, you can sample

This means you can search the chance-tree and just randomly select outcomes (based on probabilities) for each chance node

If you have a large number of samples, this should converge to the average

MCTS

How to find which actions are “good”?

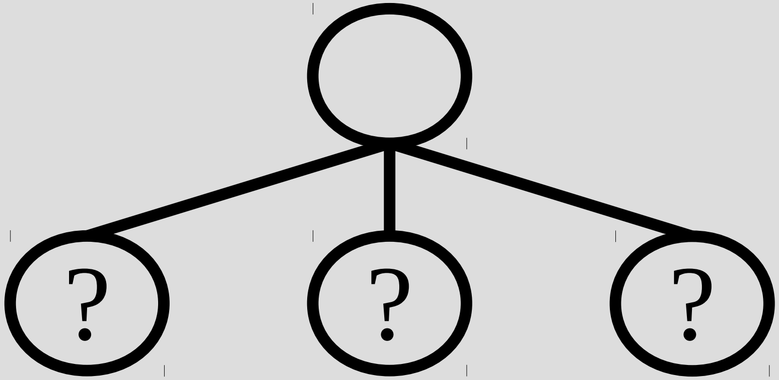
The “Upper Confidence Bound applied to Trees” UCT is commonly used:

$$\max_{n \in children} \left(\frac{win(n)}{times(n)} + \sqrt{\frac{2 \ln times(parent(n))}{times(n)}} \right)$$

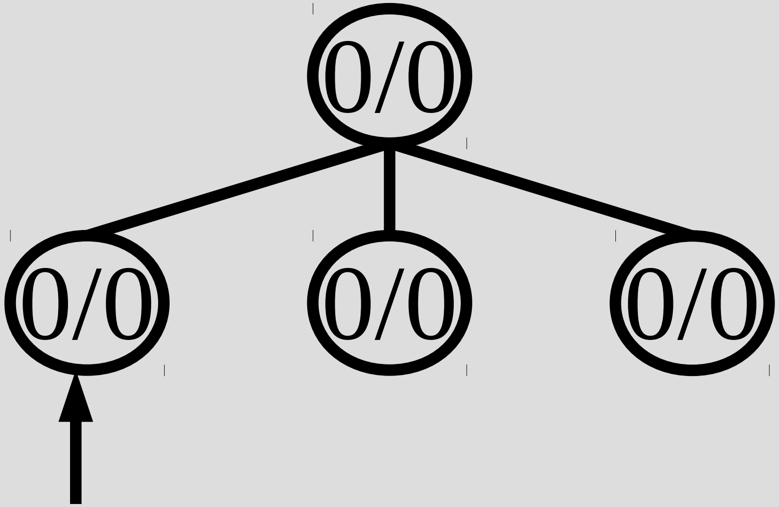
This ensures a trade off between checking branches you haven't explored much and exploring hopeful branches

(https://www.youtube.com/watch?v=Fbs4lnGLS8M)

MCTS



MCTS

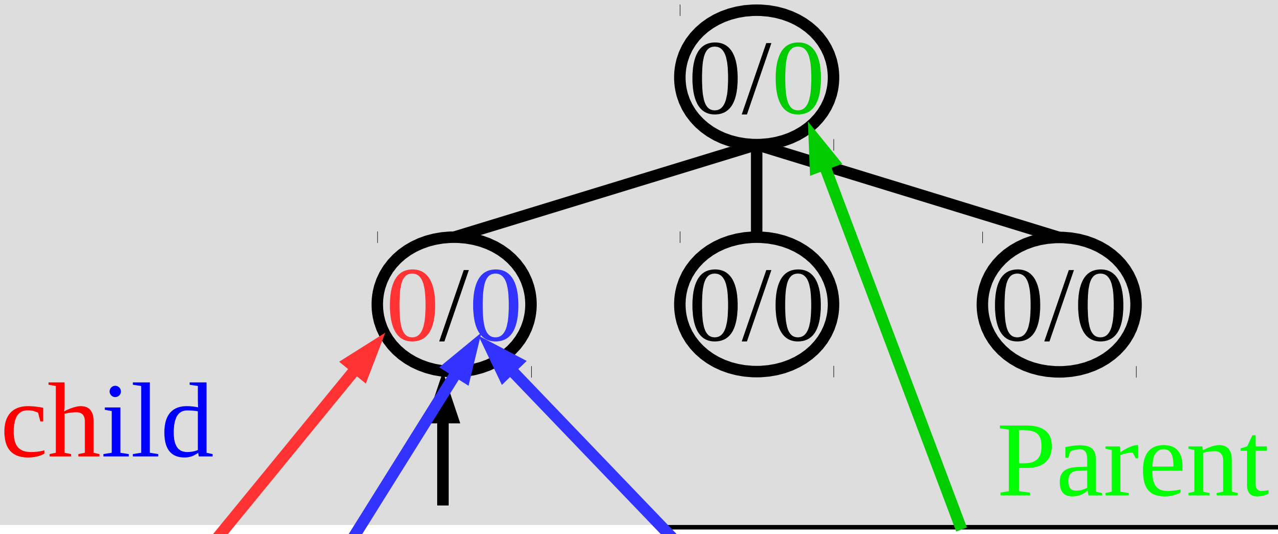


$$\frac{win(n)}{times(n)} + \sqrt{\frac{2\ln times(parent(n))}{times(n)}}$$

$$= \frac{0}{0} + \sqrt{\frac{2ln0}{0}}$$

$$= \infty$$

MCTS

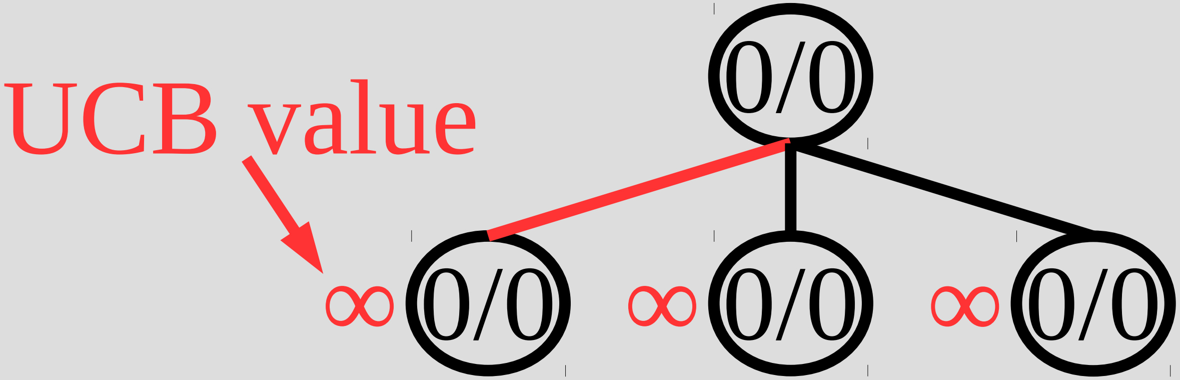


$$\frac{win(n)}{times(n)} + \sqrt{\frac{2\ln times(parent(n))}{times(n)}}$$

$$= \frac{0}{0} + \sqrt{\frac{2ln0}{0}}$$

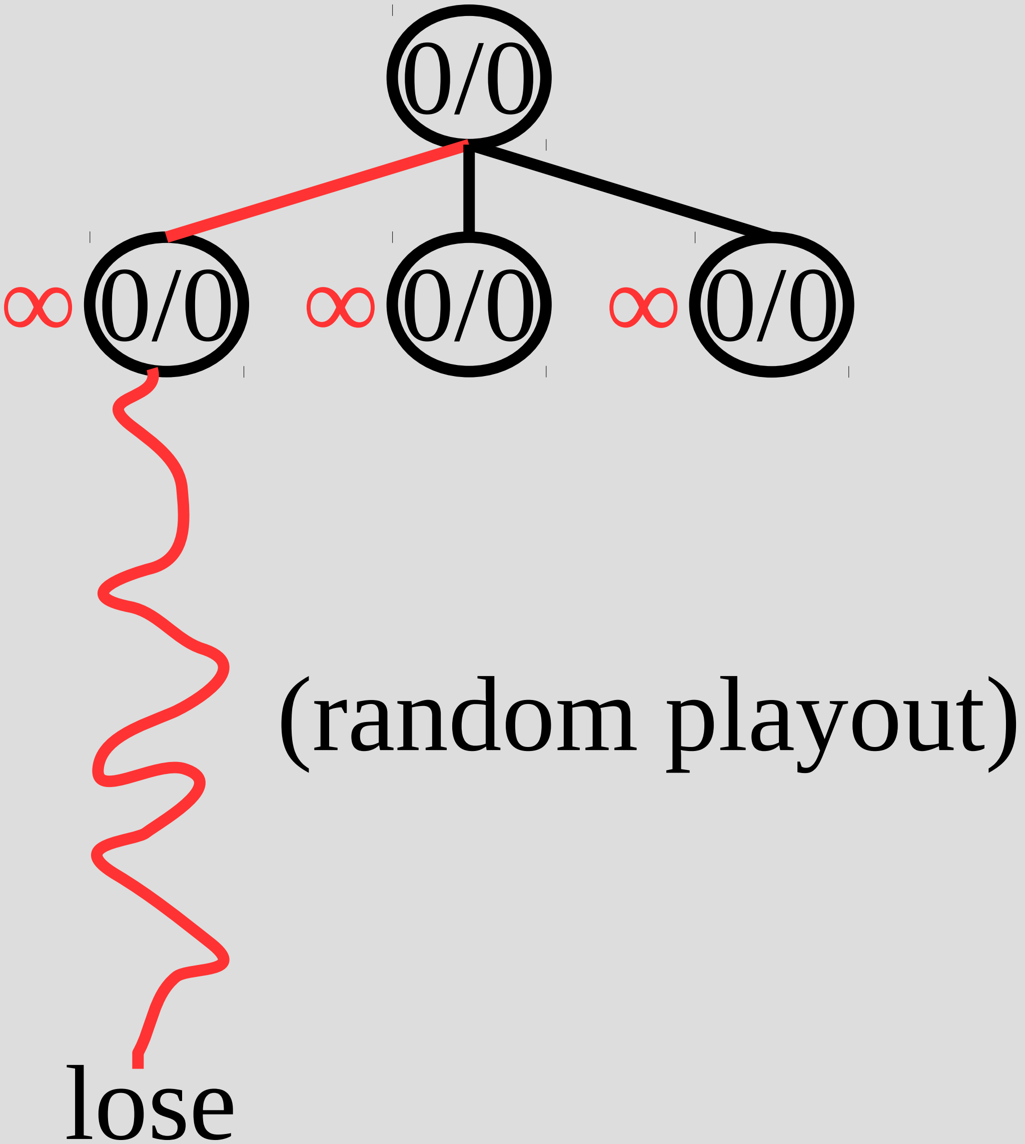
$$= \infty$$

MCTS

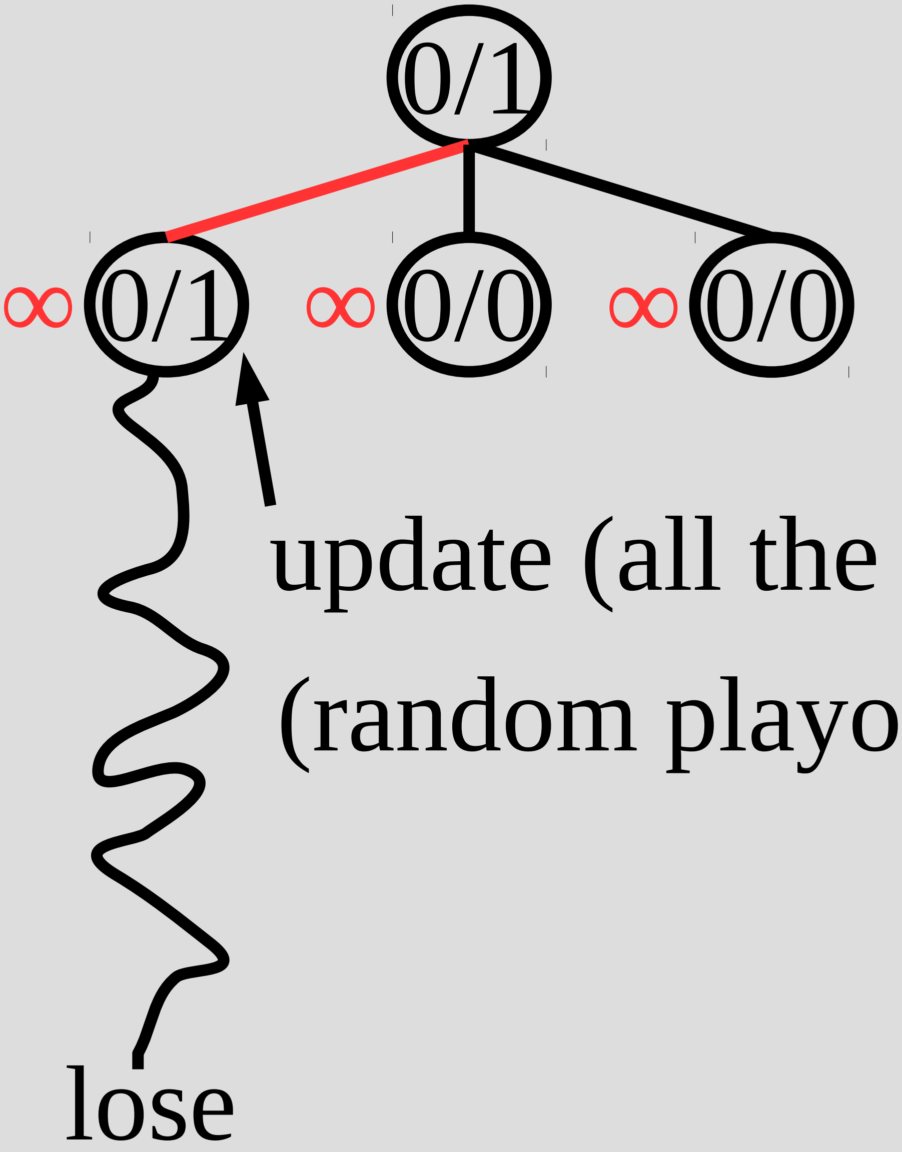


Pick max on depth 1 (I'll pick left-most)

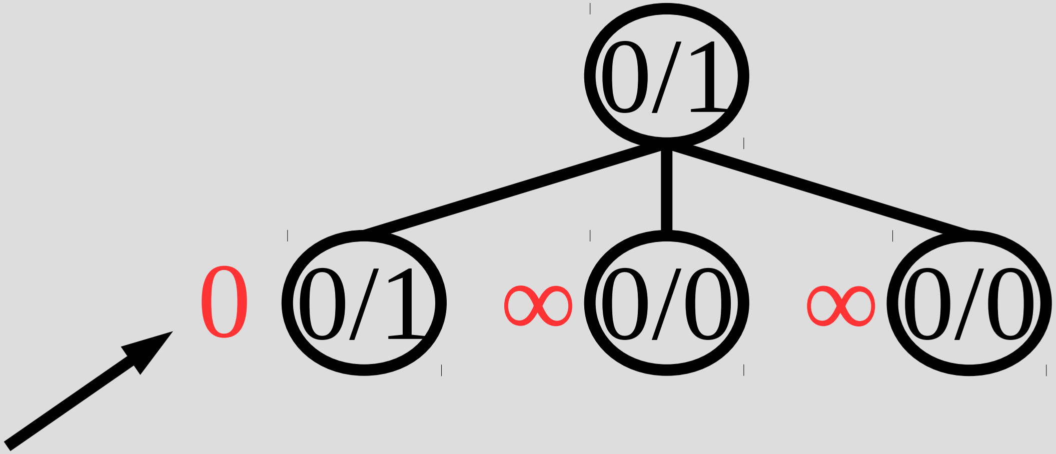
MCTS



MCTS



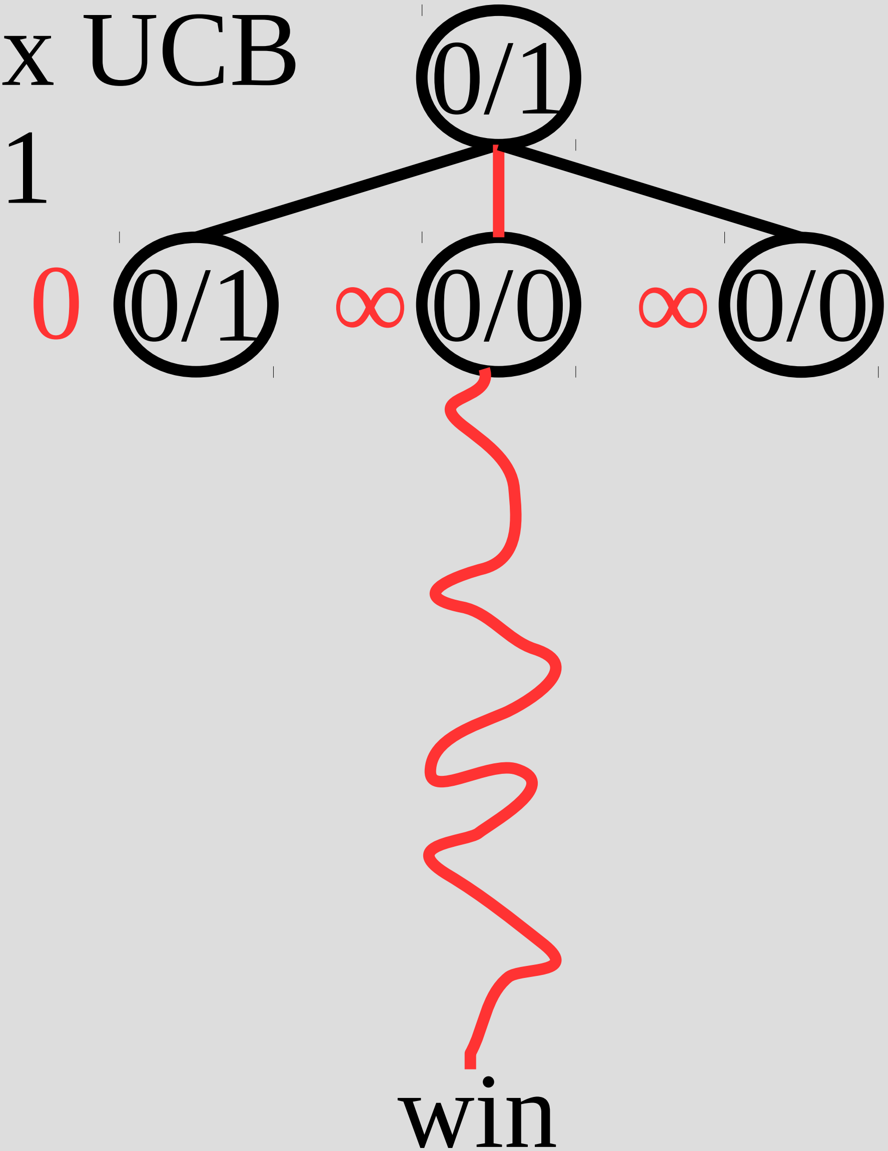
MCTS



update UCB values (all nodes)

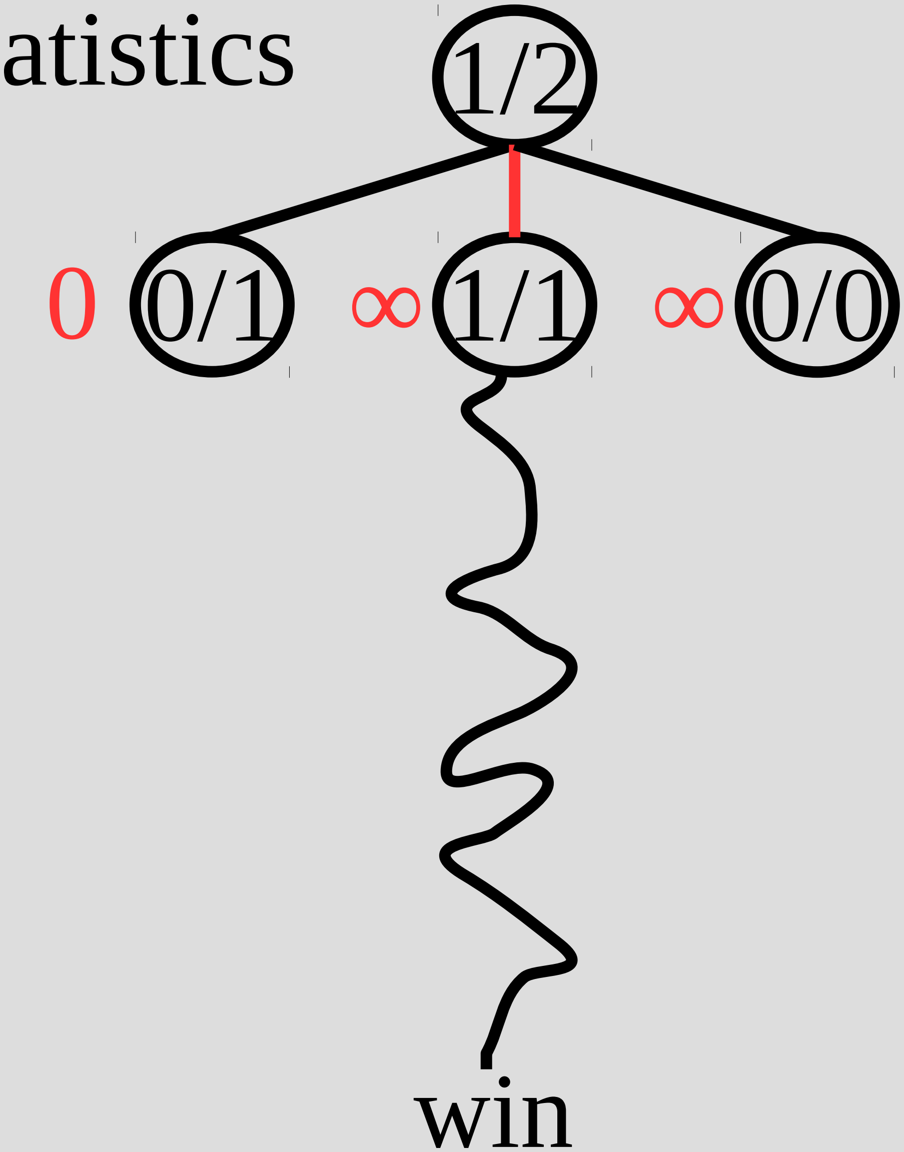
MCTS

select max UCB
on depth 1
& rollout



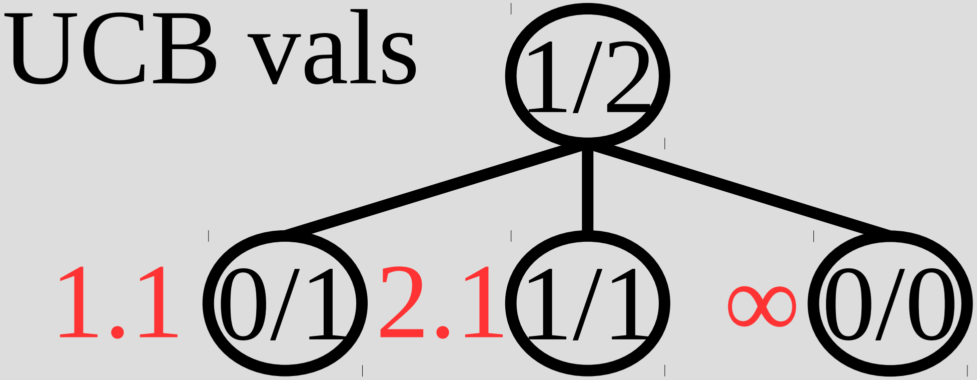
MCTS

update statistics



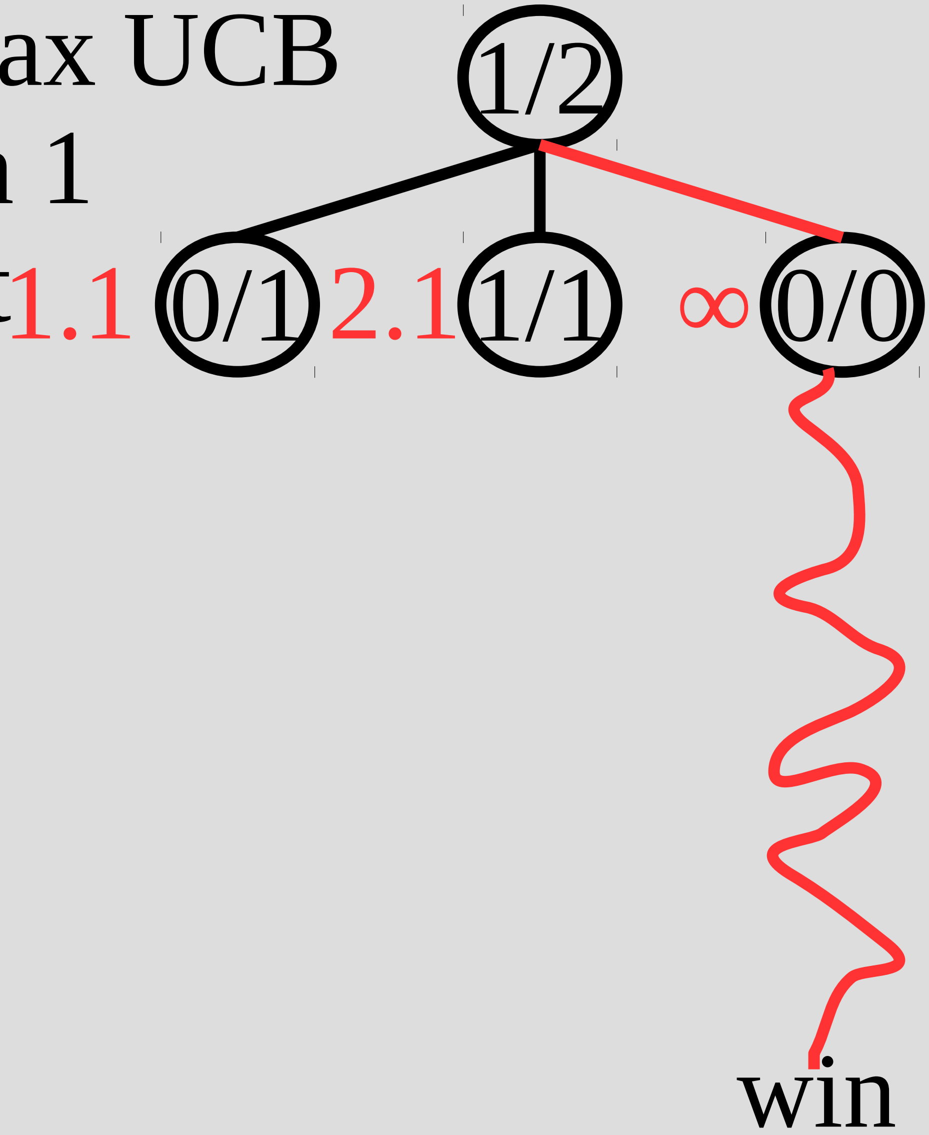
MCTS

update UCB vals



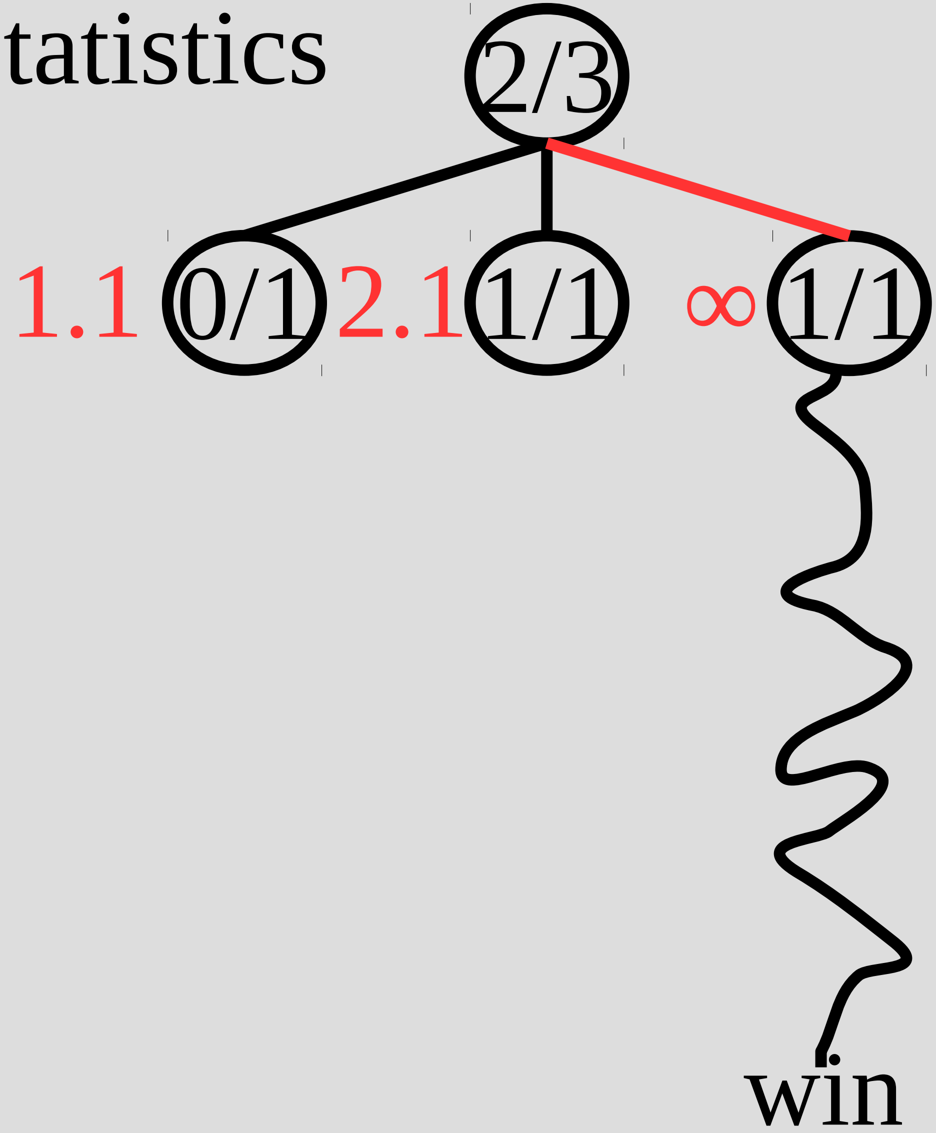
MCTS

select max UCB
on depth 1
&rollout



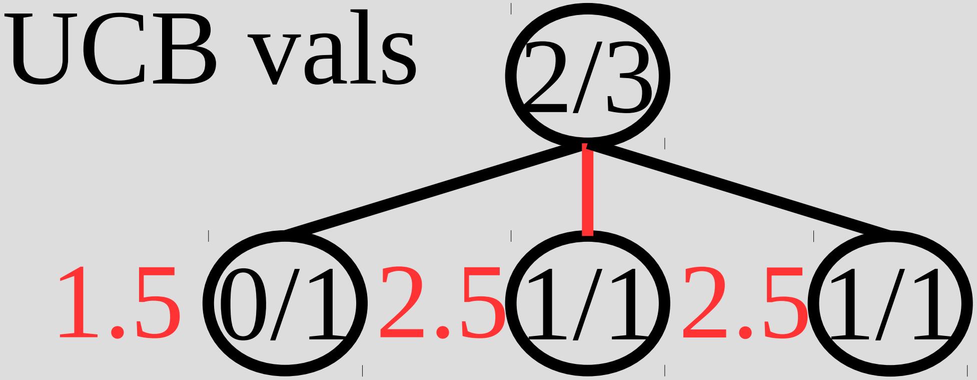
MCTS

update statistics



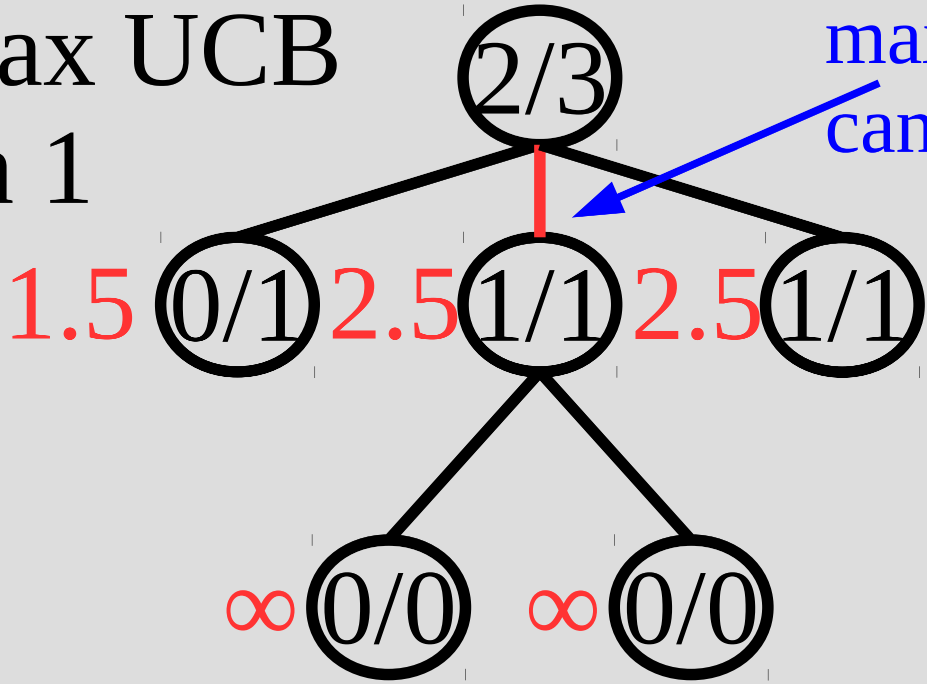
MCTS

update UCB vals



MCTS

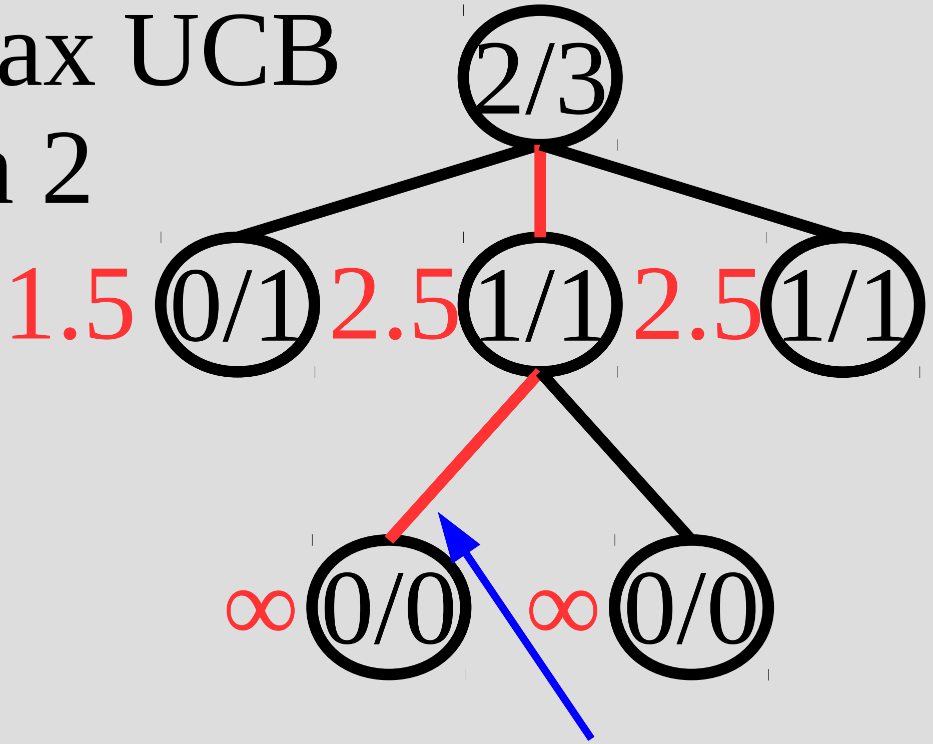
select max UCB on depth 1



max on depth 1 a tie, can pick either

MCTS

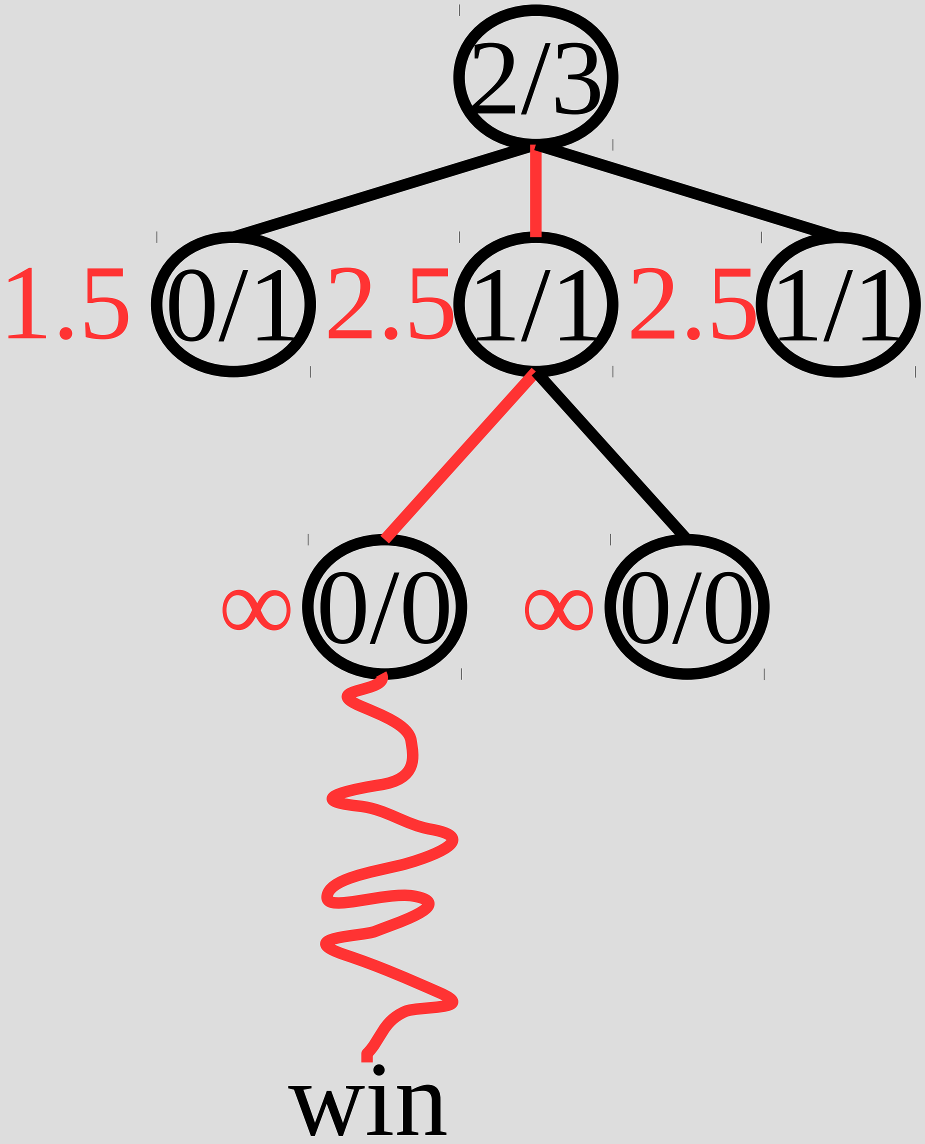
select max UCB
on depth 2



also a tie on depth 2,
can pick either (I go left)

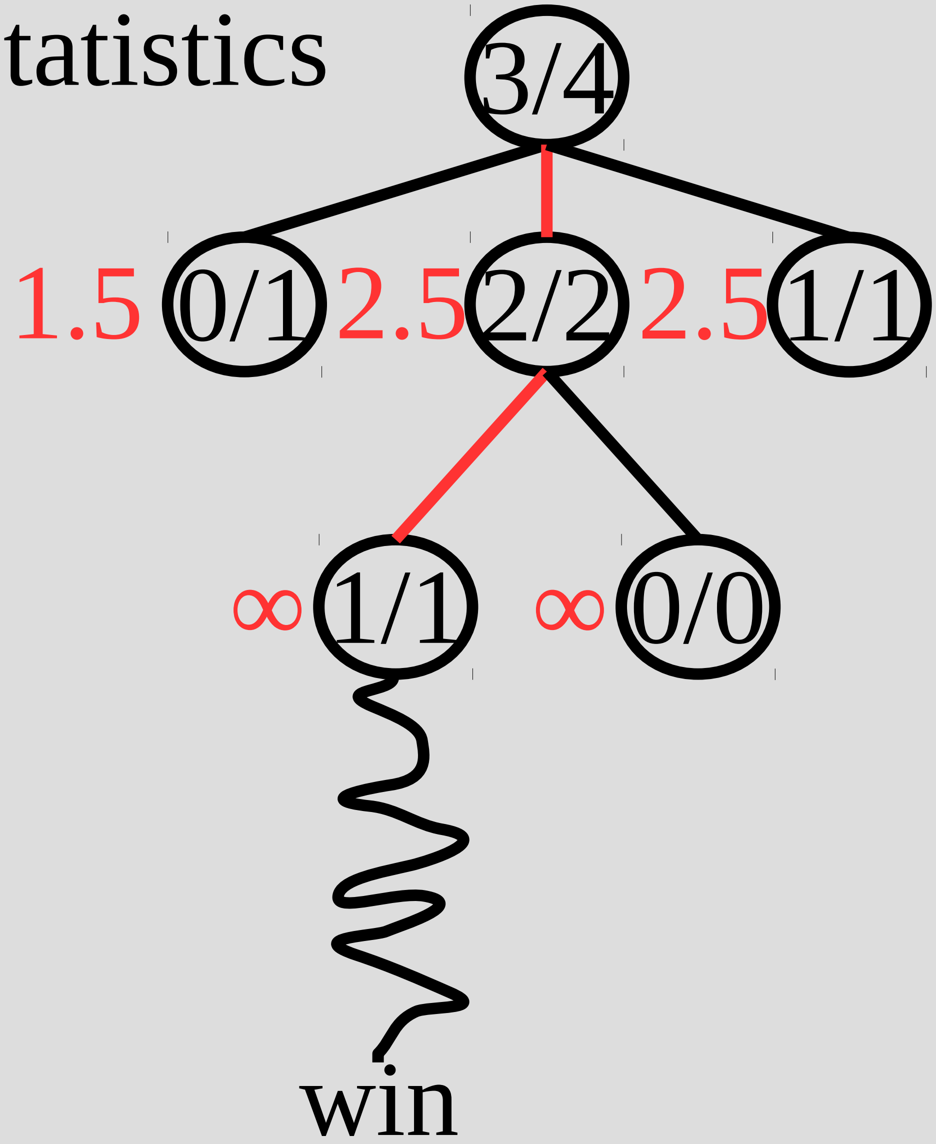
MCTS

rollout



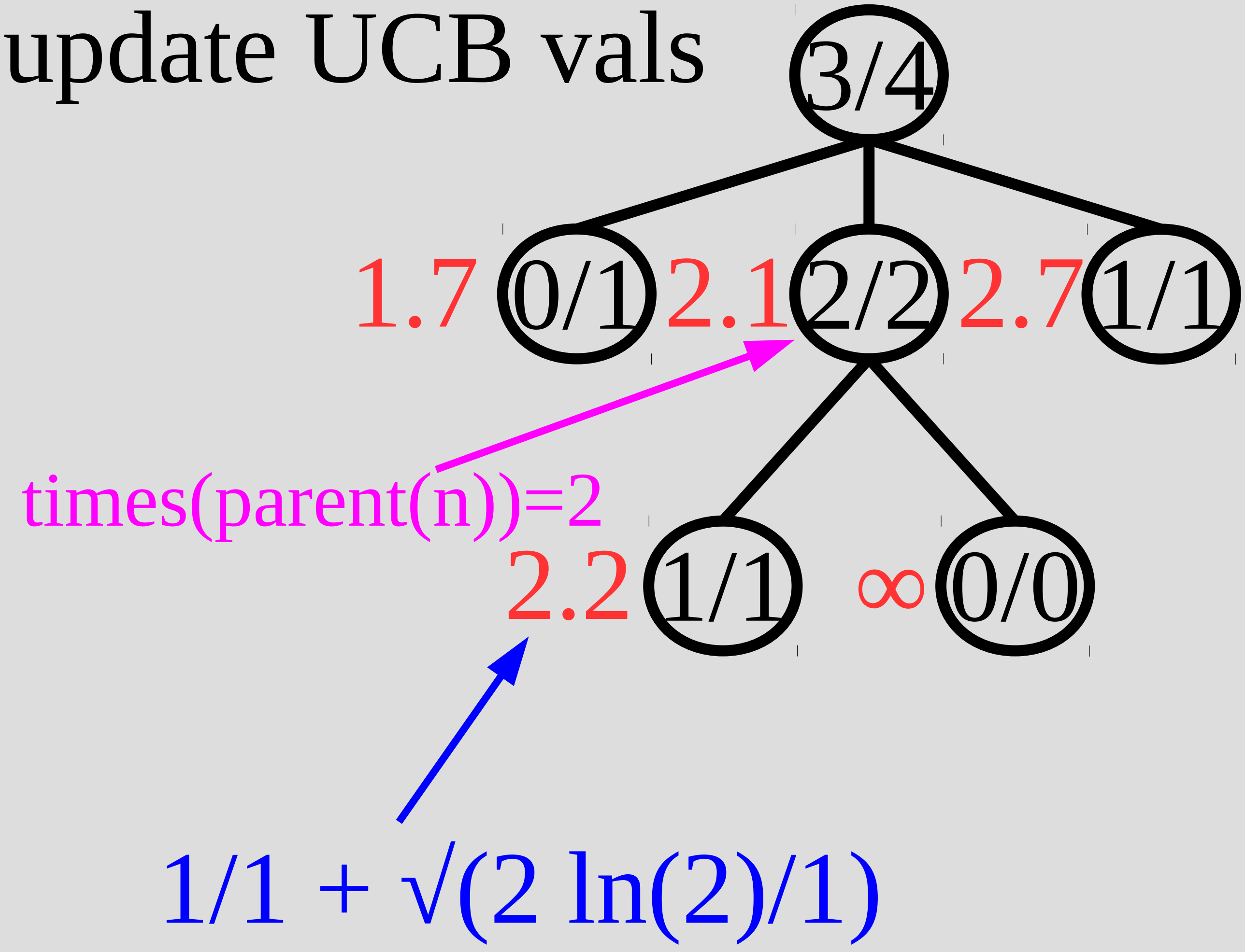
MCTS

update statistics



MCTS

update UCB vals

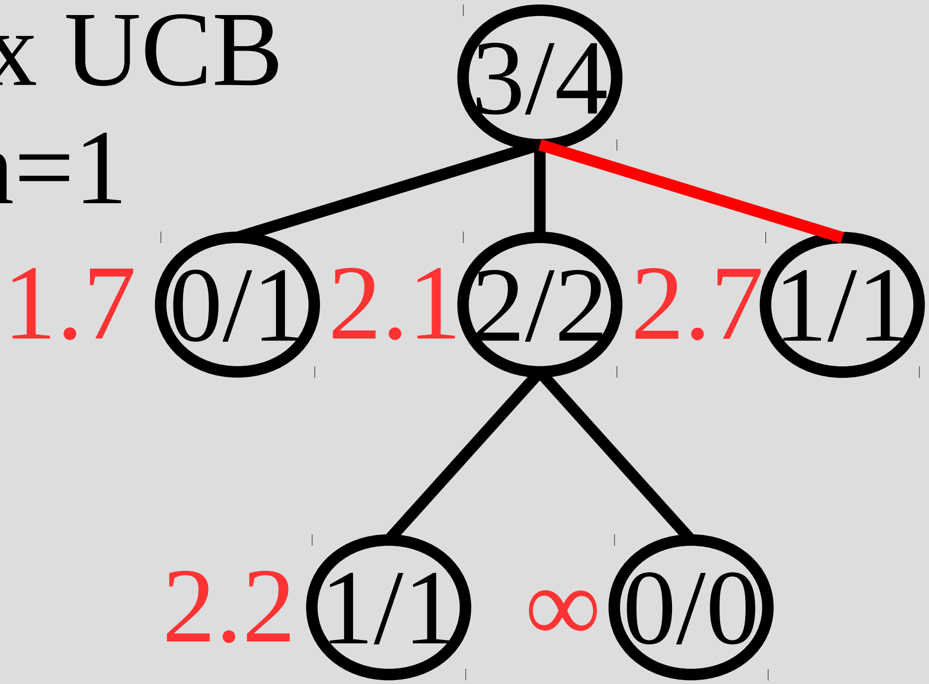


times(parent(n))=2

1/1 + √(2 ln(2)/1)

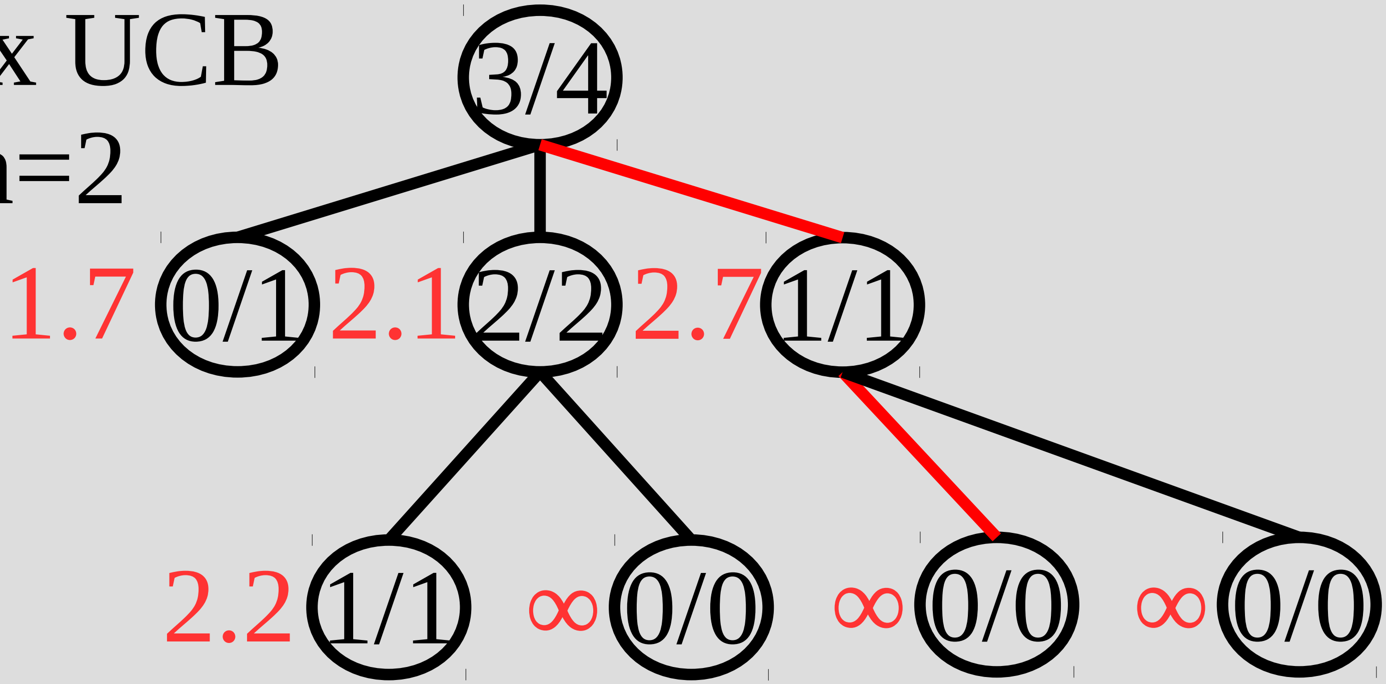
MCTS

pick max UCB
on depth=1



MCTS

pick max UCB
on depth=2



MCTS

So the algorithm’s pseudo-code is:

```
Loop:
    (1) Start at root
    (2) Pick child with best UCB value
    (3) If current node visited before,
            goto step (2)
    (4) Do a random “rollout” and record
            result up tree until root
```

MCTS

Pros:

(1) The “random playouts” are essentially generating a mid-state evaluation for you

(2) Has shown to work well on wide & deep trees, can also combine distributed comp.

Cons:

(1) Does not work well if the state does not “build up” well

(2) Often does not work on 1-player games

MCTS in games

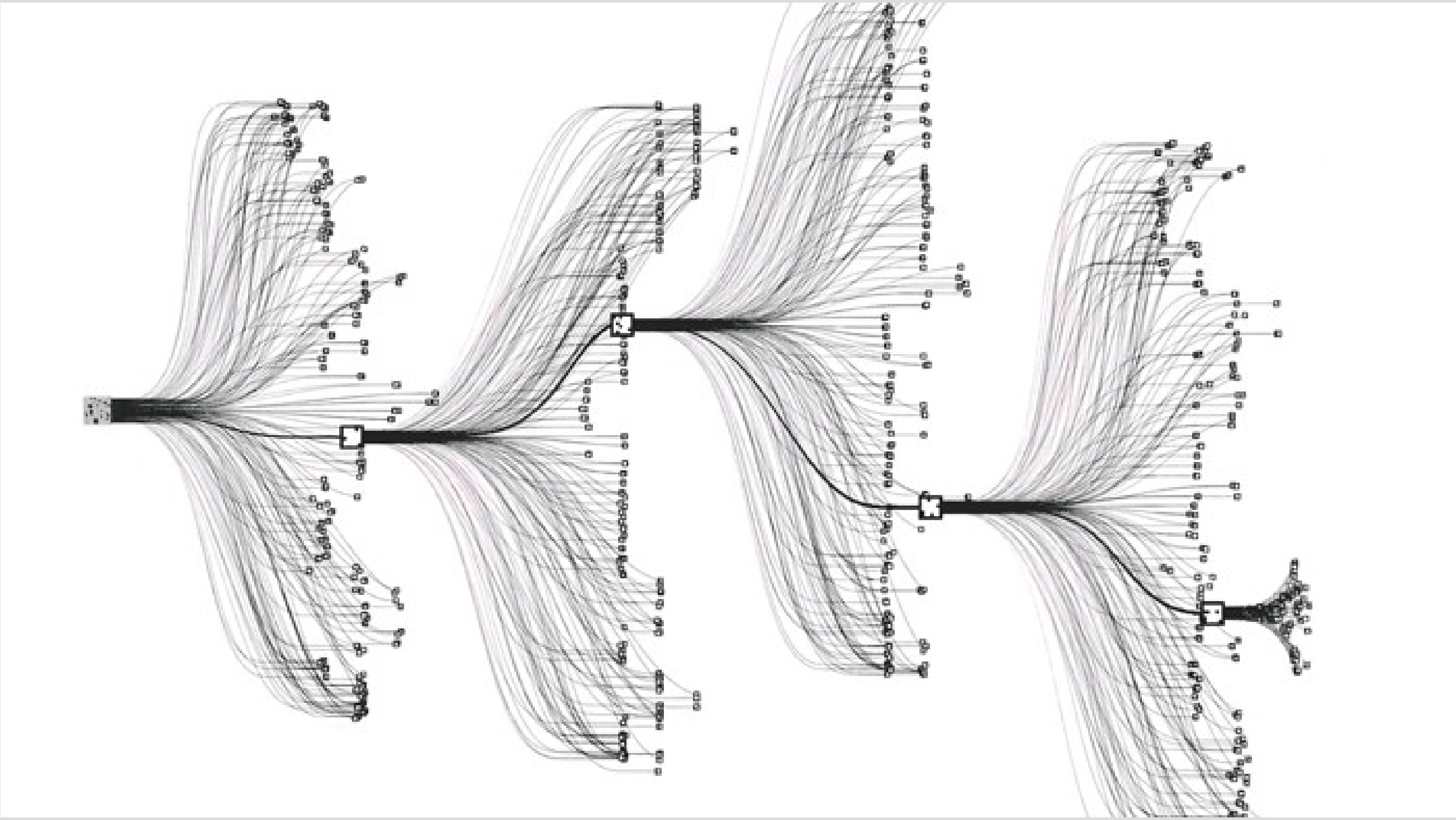
AlphaGo/Zero has been in the news recently, and is also based on neural networks

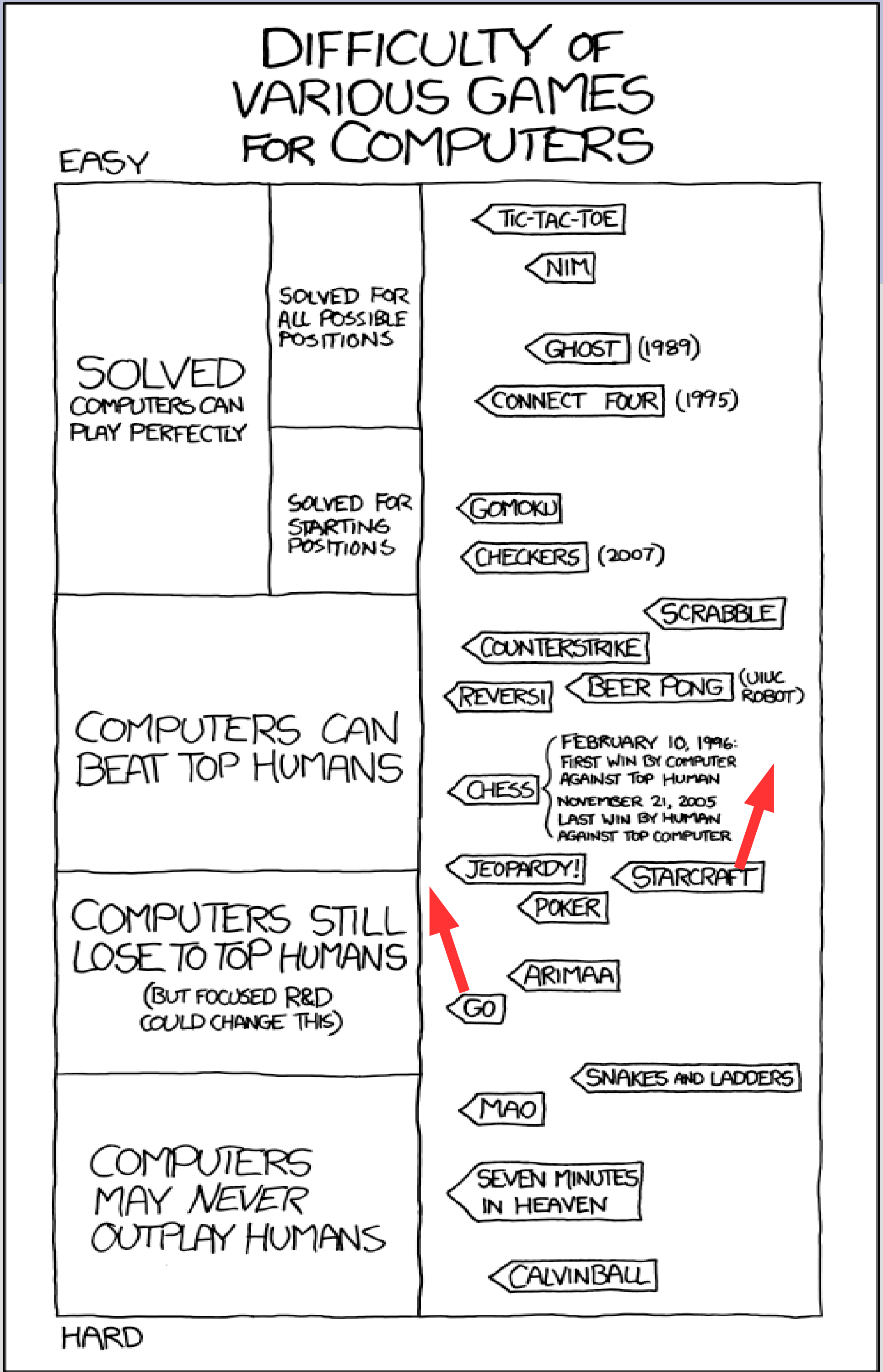
AlphaGo uses Monte-Carlo tree search guided by the neural network to prune useless parts

Often limiting Monte-Carlo in a static way reduces the effectiveness, much like mid-state evaluations can limit algorithm effectiveness

MCTS in games

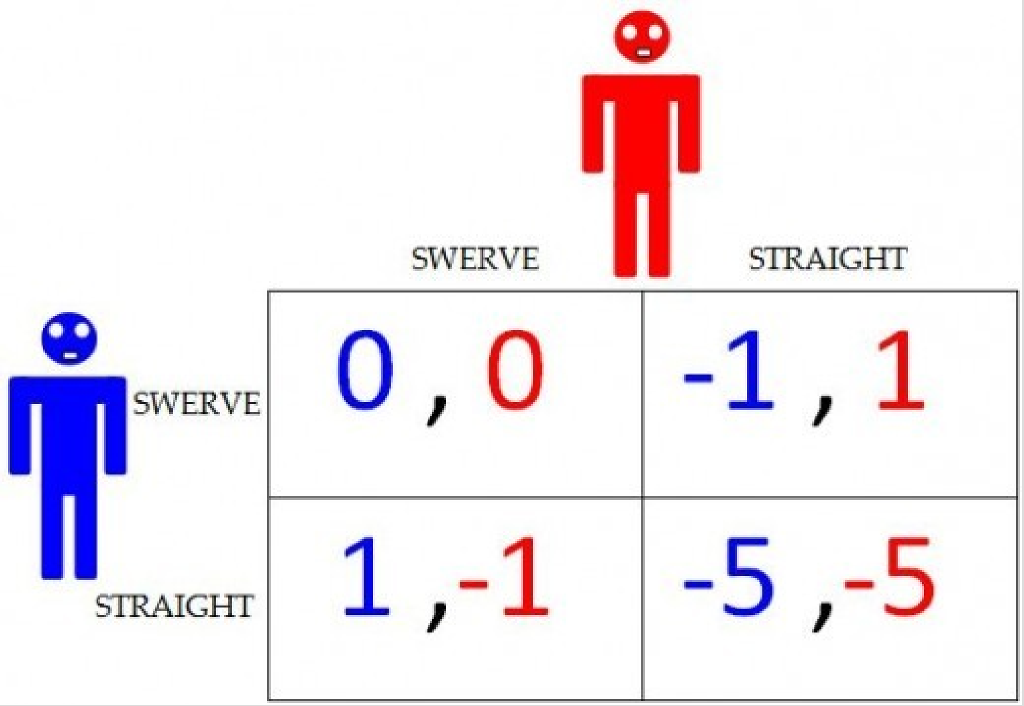
Basically, AlphaGo uses a neural network to “prune” parts for a Monte-carlo search




DIFFICULTY OF VARIOUS GAMES FOR COMPUTERS
EASY
HARD
SOLVED
COMPUTERS CAN PLAY PERFECTLY
SOLVED FOR ALL POSSIBLE POSITIONS
SOLVED FOR STARTING POSITIONS
COMPUTERS CAN BEAT TOP HUMANS
COMPUTERS STILL LOSE TO TOP HUMANS
(BUT FOCUSED R&D COULD CHANGE THIS)
COMPUTERS MAY NEVER OUTPLAY HUMANS
TIC-TAC-TOE
NIM
GHOST (1989)
CONNECT FOUR (1995)
GOMOKU
CHECKERS (2007)
SCRABBLE
COUNTERSTRIKE
REVERSI
BEER PONG (UIUC ROBOT)
CHESS
FEBRUARY 10, 1996: FIRST WIN BY COMPUTER AGAINST TOP HUMAN
NOVEMBER 21, 2005 LAST WIN BY HUMAN AGAINST TOP COMPUTER
JEOPARDY!
STARCRAFT
POKER
ARIMAA
GO
SNAKES AND LADDERS
MAO
SEVEN MINUTES IN HEAVEN
CALVINBALL

Game theory

Typically game theory uses a <u>payoff matrix</u> to represent the value of actions



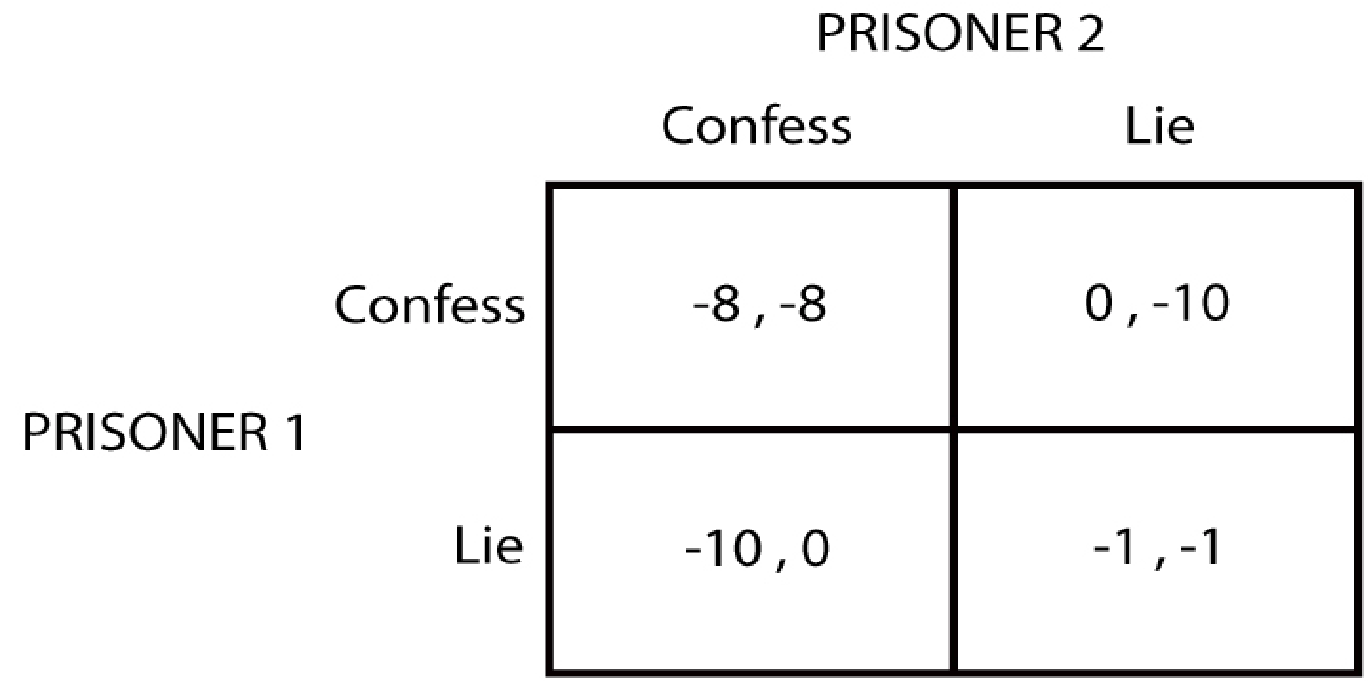
	SWERVE	STRAIGHT
SWERVE	0 , 0	-1 , 1
STRAIGHT	1 ,-1	-5 ,-5

The first value is the reward for the left player, right for top (positive is good for both)

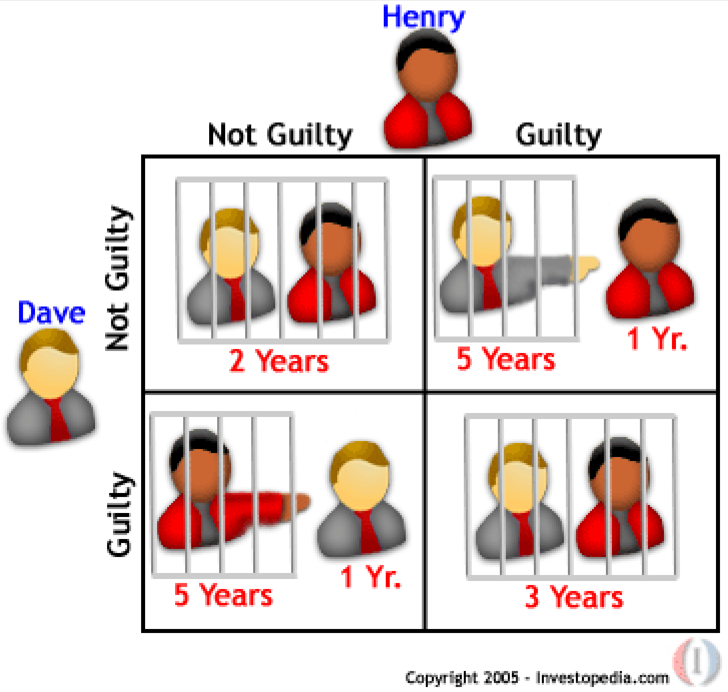
Dominance & equilibrium

Here is the famous “prisoner's dilemma”

Each player chooses one action without knowing the other's and the is only played once



		PRISONER 2	
		Confess	Lie
PRISONER 1	Confess	-8 , -8	0 , -10
	Lie	-10 , 0	-1 , -1

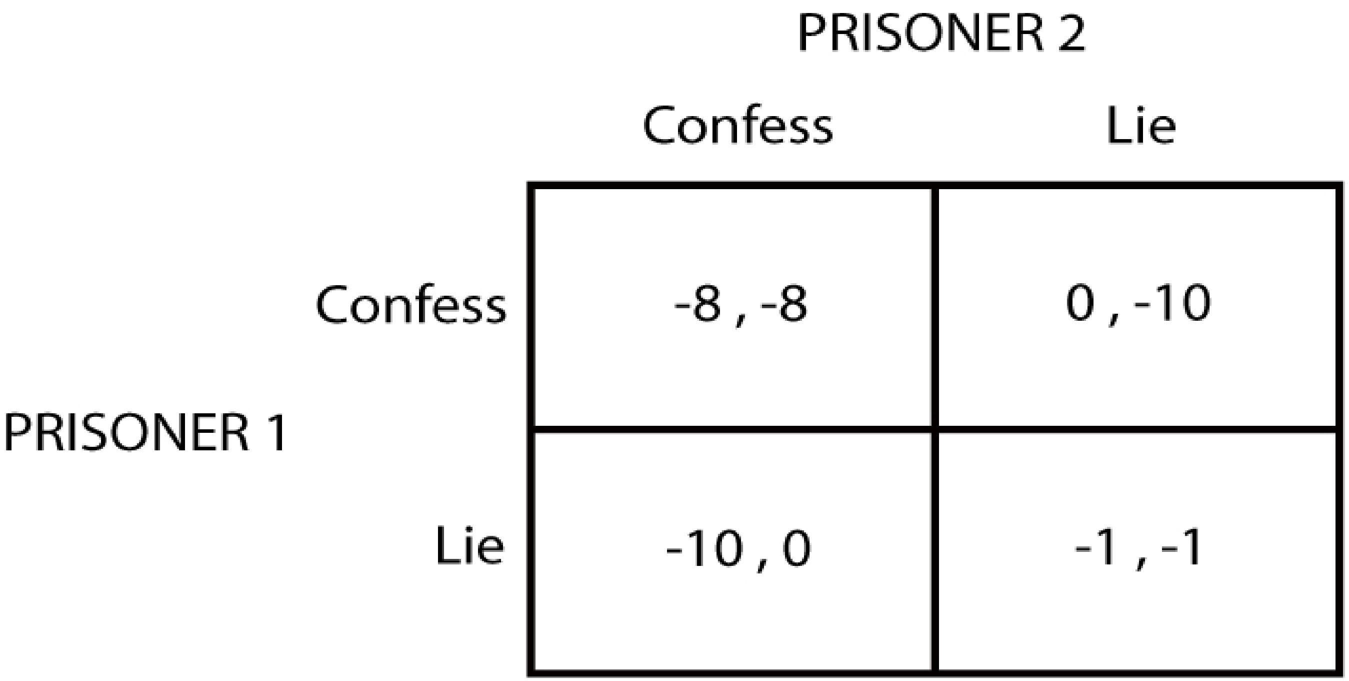


Copyright 2005 - Investopedia.com

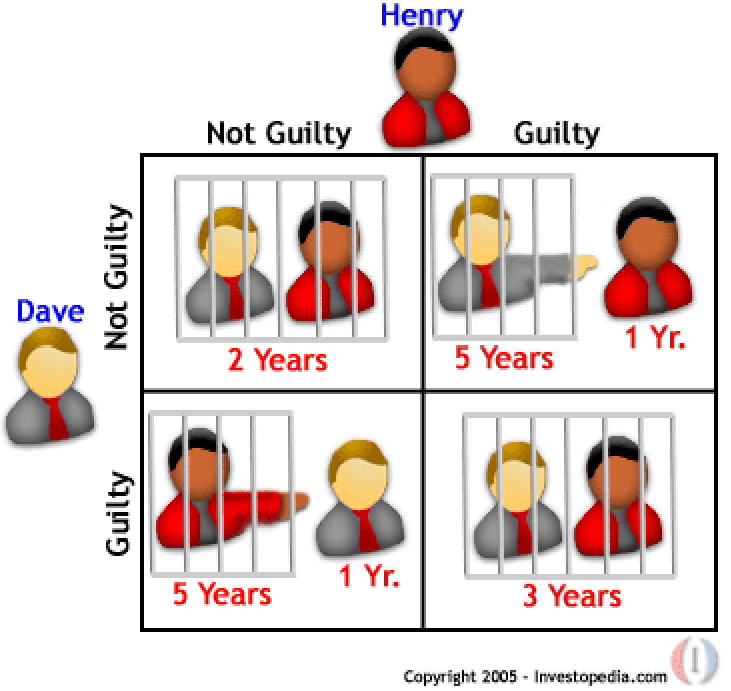
Dominance & equilibrium

What option would you pick?

Why?



		PRISONER 2	
		Confess	Lie
PRISONER 1	Confess	-8 , -8	0 , -10
	Lie	-10 , 0	-1 , -1



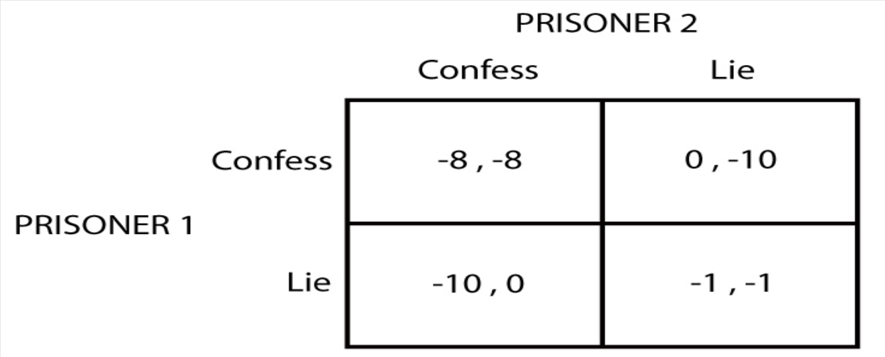
Copyright 2005 - Investopedia.com

Dominance & equilibrium

What would a rational agent pick?

If prisoner 2 confesses, we are in the first column... -8 if we confess, or -10 if we lie
--> Thus we should confess

If prisoner 2 lies, we are in the second column,
0 if we confess,
-1 if we lie
--> We should confess



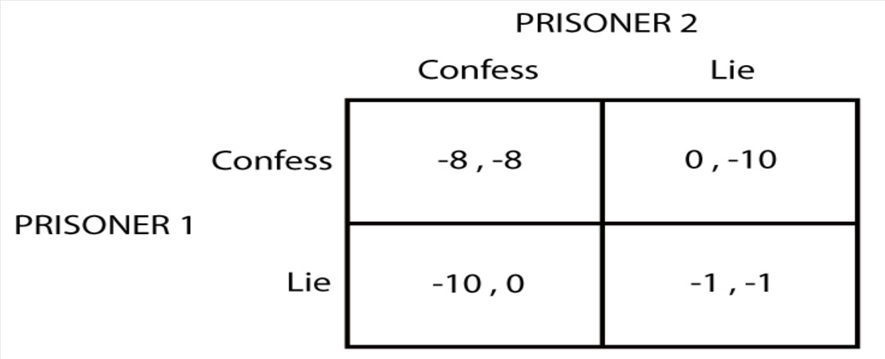
		PRISONER 2	
		Confess	Lie
PRISONER 1	Confess	-8 , -8	0 , -10
	Lie	-10 , 0	-1 , -1

Dominance & equilibrium

It turns out regardless of the other player's action, it is in our personal interest to confess

This is the Nash equilibrium, as any deviation of strategy (i.e. lying) can result in a lower score (i.e. if opponent confesses)

The Nash equilibrium looks at the worst case and is greedy

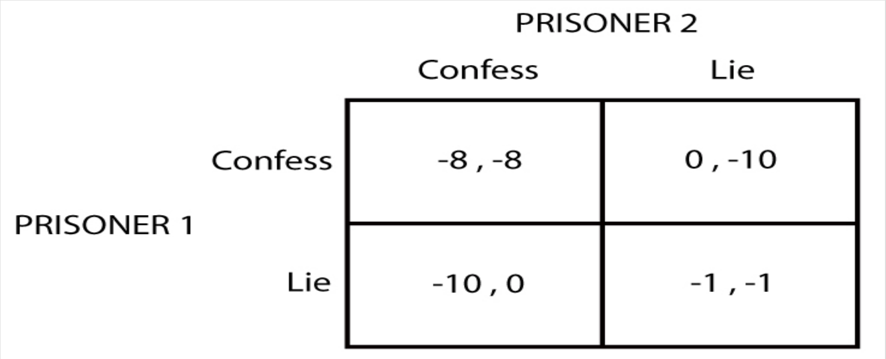


		PRISONER 2	
		Confess	Lie
PRISONER 1	Confess	-8 , -8	0 , -10
	Lie	-10 , 0	-1 , -1

Dominance & equilibrium

Formally, a Nash equilibrium is when the combined strategies of all players give no incentive for any single player to change

In other words, if any single person decides to change strategies, they cannot improve



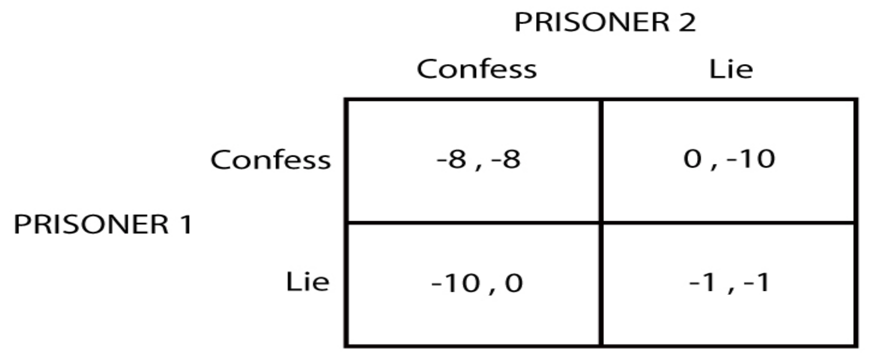
		PRISONER 2	
		Confess	Lie
PRISONER 1	Confess	-8 , -8	0 , -10
	Lie	-10 , 0	-1 , -1

Dominance & equilibrium

Alternatively, a Pareto optimum is a state where no other state can result in a gain or tie for all players (excluding all ties)

If the PD game, [-8, -8] is a Nash equilibrium, but is not a Pareto optimum (as [-1, -1] better for both players)

However [-10,0] is also a Pareto optimum...



		PRISONER 2	
		Confess	Lie
PRISONER 1	Confess	-8 , -8	0 , -10
	Lie	-10 , 0	-1 , -1

Dominance & equilibrium

Every game has at least one Nash equilibrium and Pareto optimum, however...

- Nash equilibrium might not be the best outcome for all players (like PD game, assumes no cooperation)

- A Pareto optimum might not be stable (in PD the [-10,0] is unstable as player 1 wants to switch off “lie” and to “confess” if they play again or know strategy)

Dominance & equilibrium

Find the Nash and Pareto for the following:
(about lecturing in a certain csci class)

Student

	pay attention	sleep
prepare well	5, 5	-2, 2
slack off	1, -5	0, 0

Teacher

Find best strategy

How do we formally find a Nash equilibrium?

If it is zero-sum game, can use minimax
as neither player wants to switch for Nash
(our PD example was not zero sum)

Let's play a simple number game: two players write down either 1 or 0 then show each other. If the sum is odd, player one wins. Otherwise, player 2 wins (on even sum)

Find best strategy

This gives the following payoffs:

		Player 2
Player 1	Pick 0	Pick 1
Pick 0	-1, 1	1, -1
Pick 1	1, -1	-1, 1

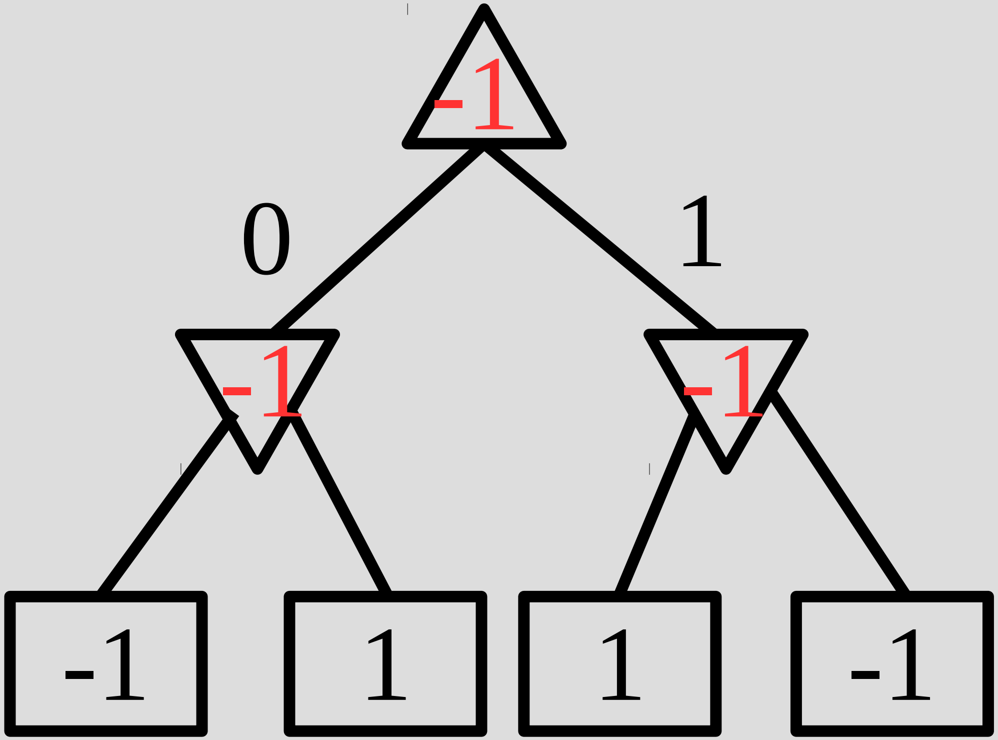
(player 1's value first, then player 2's value)

We will run minimax on this tree twice:

1. Once with player 1 knowing player 2's move (i.e. choosing after them)
2. Once with player 2 knowing player 1's move

Find best strategy

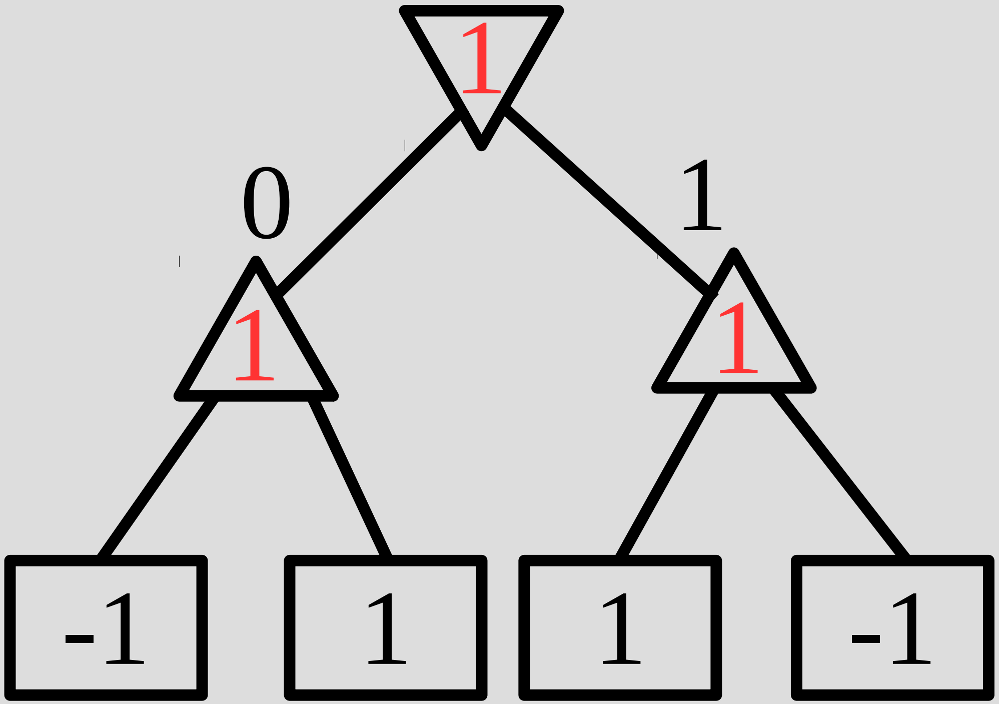
Player 1 to go first (max):



If player 1 goes first, it will always lose

Find best strategy

Player 2 to go first (min):



If player 2 goes first, it will always lose

Find best strategy

This is not useful, and only really tells us that the best strategy is between -1 and 1
(which is fairly obvious)

This minimax strategy can only find pure strategies (i.e. you should play a single move 100% of the time)

To find a “mixed strategy” (probabilistically play), we need to turn to linear programming

Find best strategy

A pure strategy is one where a player always picks the same strategy (deterministic)

A mixed strategy is when a player chooses actions probabilistically from a fixed probability distribution (i.e. the percent of time they pick an action is fixed)

If one strategy is better or equal to all others across all responses, it is a dominant strategy

Find best strategy

The definition of a Nash equilibrium is when no one has an incentive to change the combined strategy between all players

So we will only consider our opponent's rewards (and not consider our own)

This is a bit weird since we are not considering our own rewards at all, which is why the Nash equilibrium is sometimes criticized

Find best strategy

First we parameterize this and make the tree stochastic:

Player 1 will choose action “0” with probability p, and action “1” with (1-p)

If player 2 always picks 0, so the payoff for p2:

$(1)p + (-1)(1-p)$

If player 2 always picks 1, so the payoff for p2:

$(-1)p + (1)(1-p)$

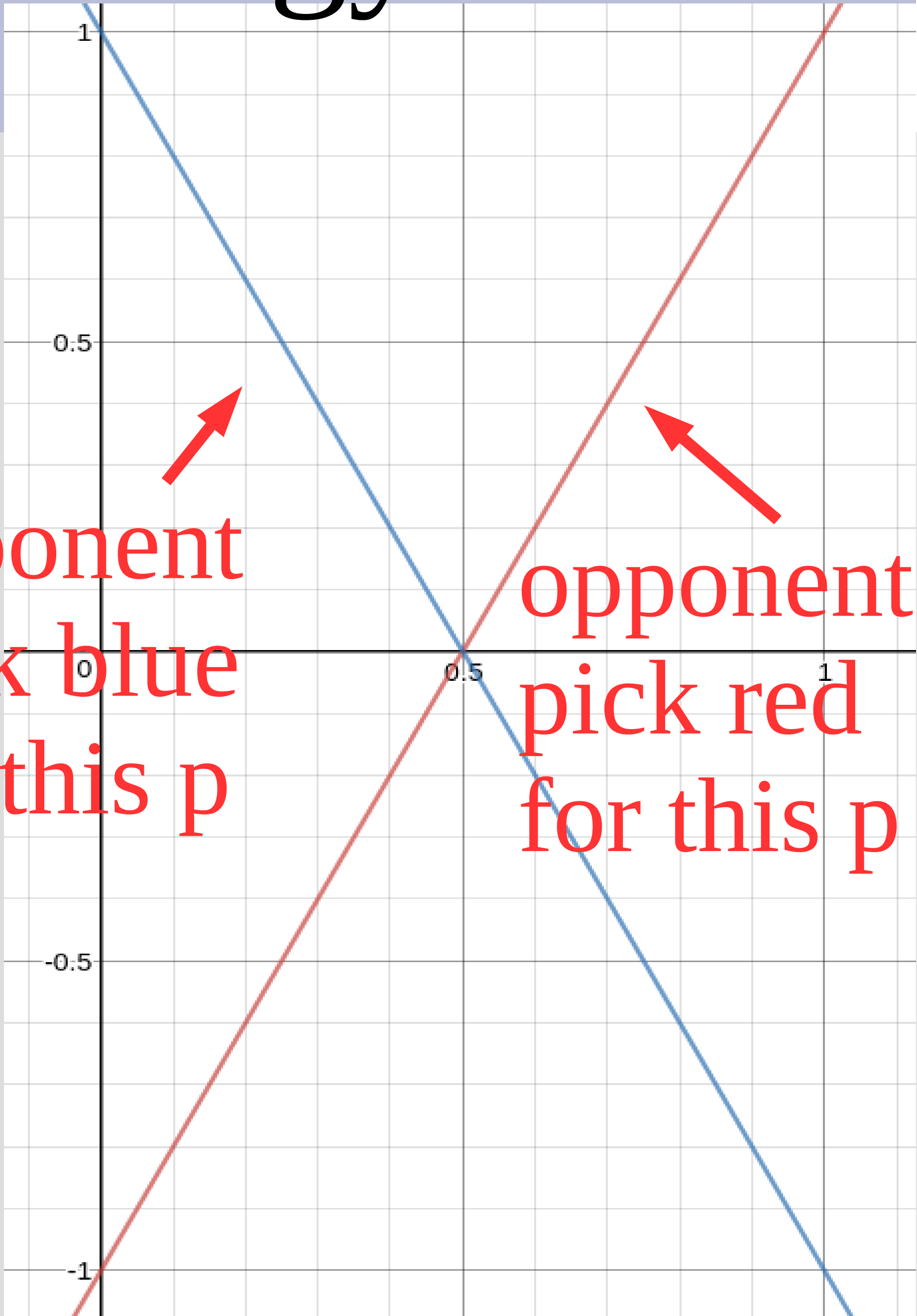
Find best strategy

Plot these two lines:

$$U = (1)p + (-1)(1-p)$$
$$U = (-1)p + (1)(1-p)$$

As we maximize, the opponent gets to pick which line to play

Thus we choose the intersection



opponent pick blue for this p

opponent pick red for this p

Find best strategy

Thus we find that our best strategy is to play 0 half the time and 1 the other half

The result is we win as much as we lose on average, and the overall game result is 0

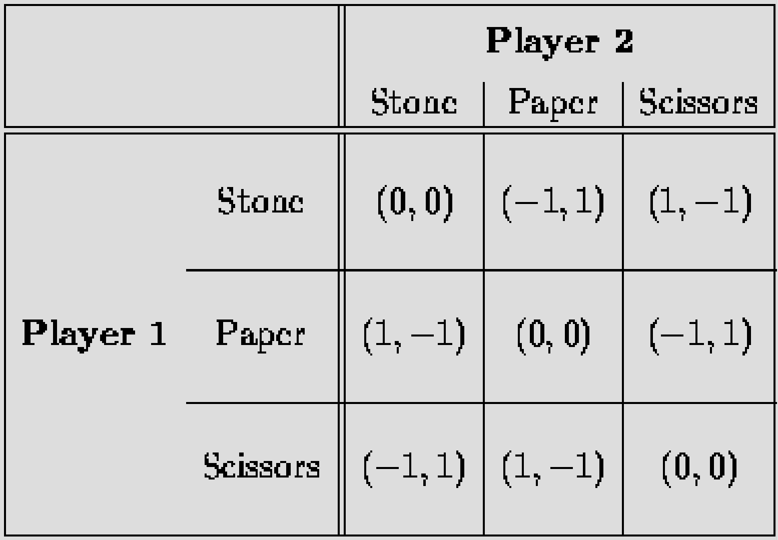
Player 2 can find their strategy in this method as well, and will get the same 50/50 strategy (this is not always the case that both players play the same for Nash)

Find best strategy

We have two actions, so one parameter (p) and thus we look for the intersections of lines

If we had 3 actions (rock-paper-scissors), we would have 2 parameters and look for the intersection of 3 planes (2D)

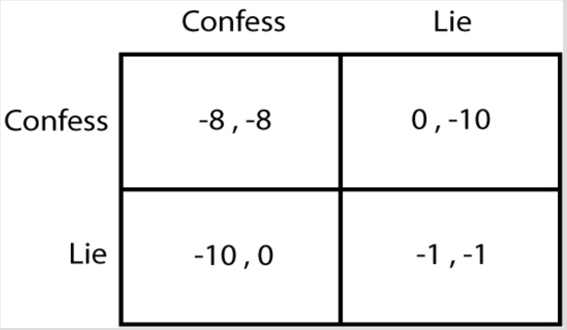
This can generalize to any number of actions (but not a lot of fun)



		Player 2		
		Stone	Paper	Scissors
Player 1	Stone	$(0, 0)$	$(-1, 1)$	$(1, -1)$
	Paper	$(1, -1)$	$(0, 0)$	$(-1, 1)$
	Scissors	$(-1, 1)$	$(1, -1)$	$(0, 0)$

Find best strategy

How does this compare on PD?



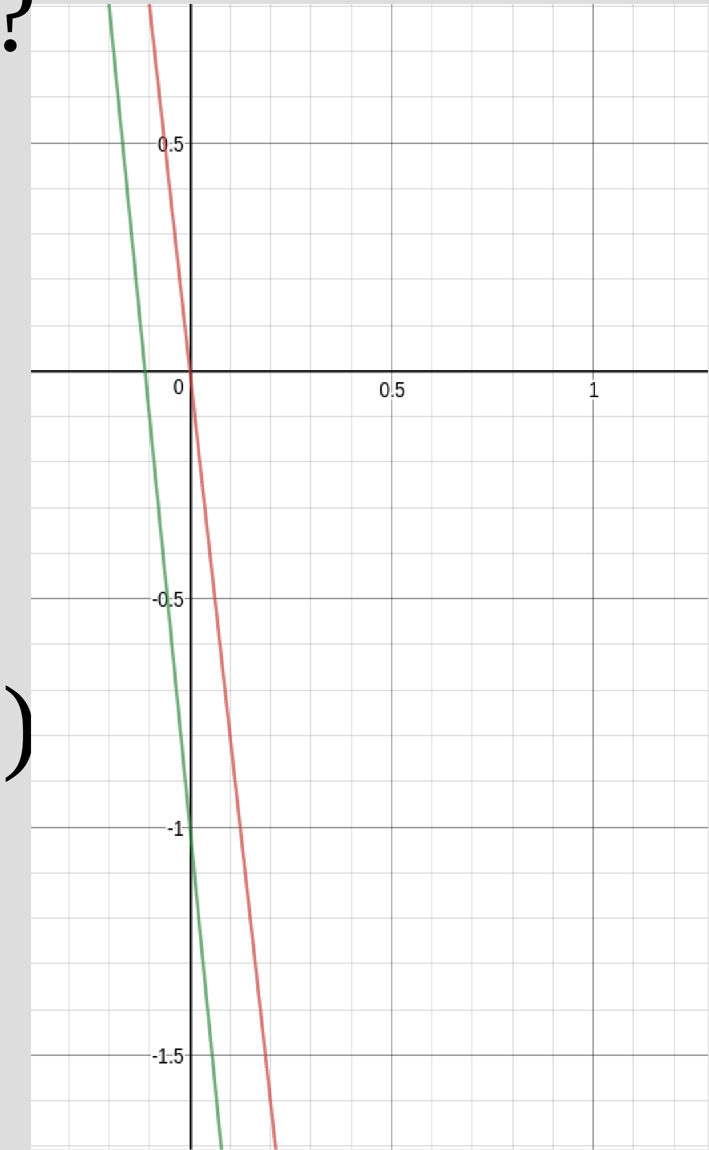
	Confess	Lie
Confess	-8 , -8	0 , -10
Lie	-10 , 0	-1 , -1

Player 1: p = prob confess...

P2 Confesses: -8*p + 0*(1-p)

P2 Lies: -10*p + (-1)*(1-p)

Cross at negative p, but red line is better (confess)


0.5
0
0.5
1
-0.5
-1
-1.5