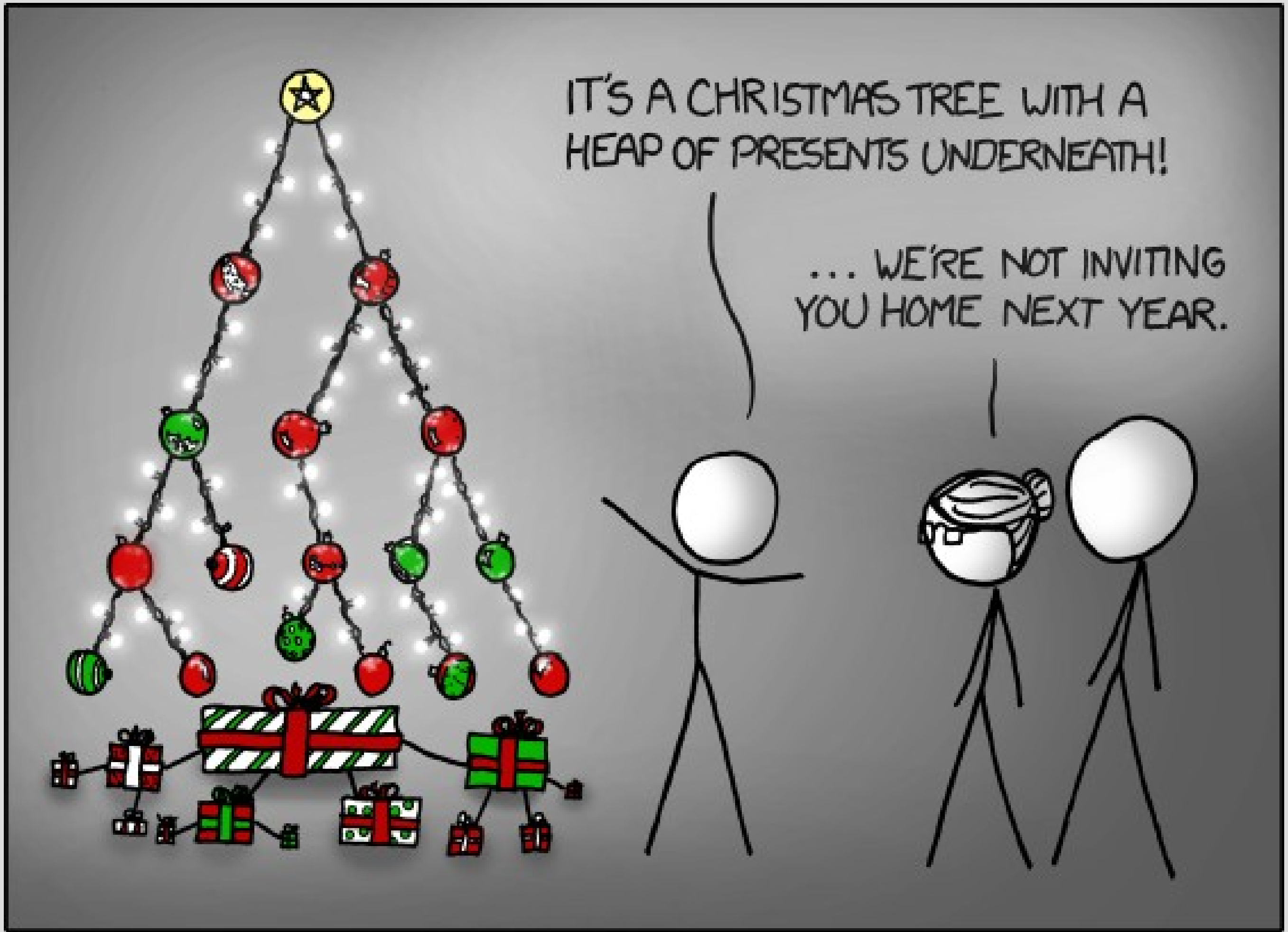
More on games (Ch. 5.4-5.7)



Announcements

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

Writing 2 posted

Writing 1 regrades – until 10/25

Random games

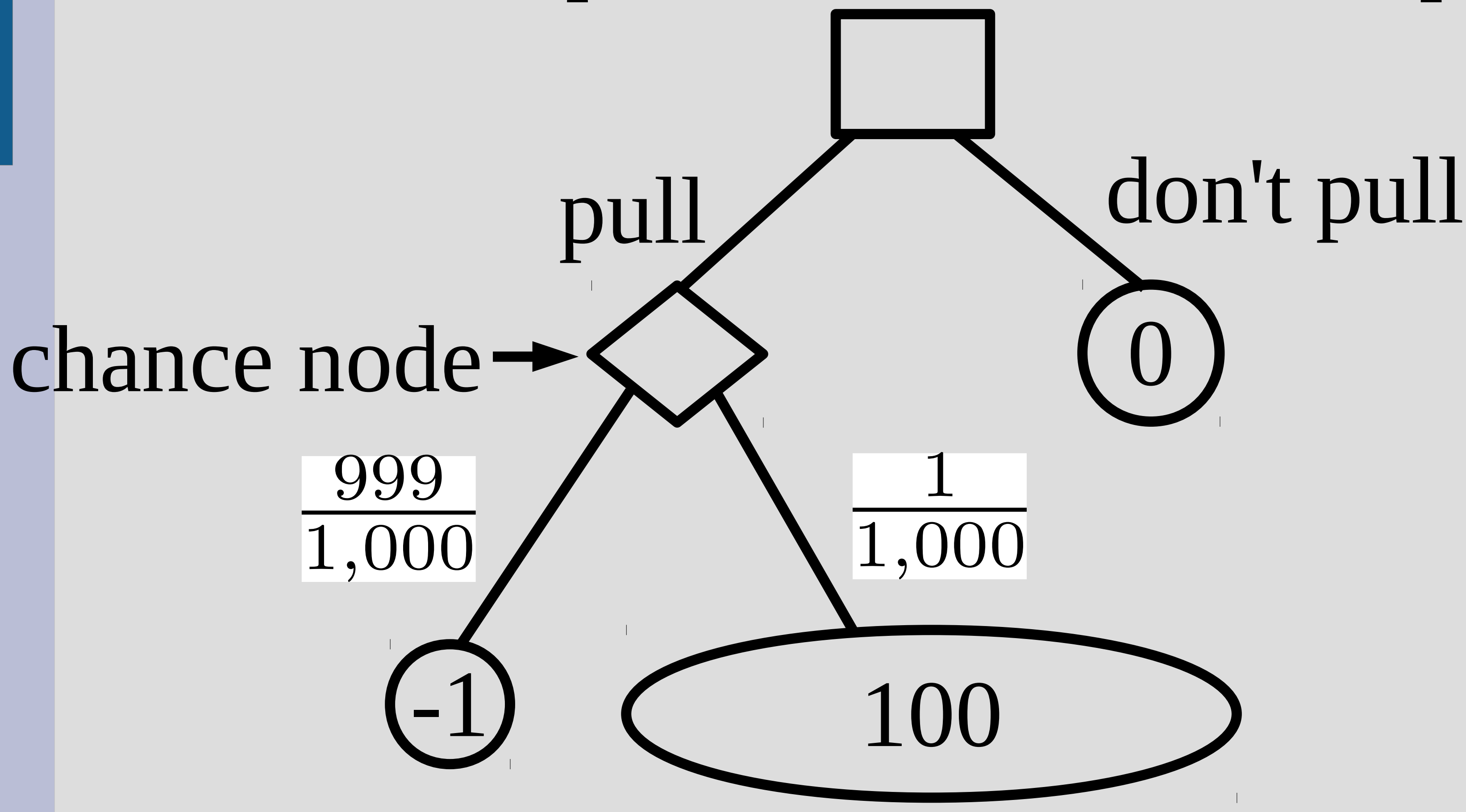
If we are playing a “game of chance”, we can add <u>chance nodes</u> to the search tree

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:

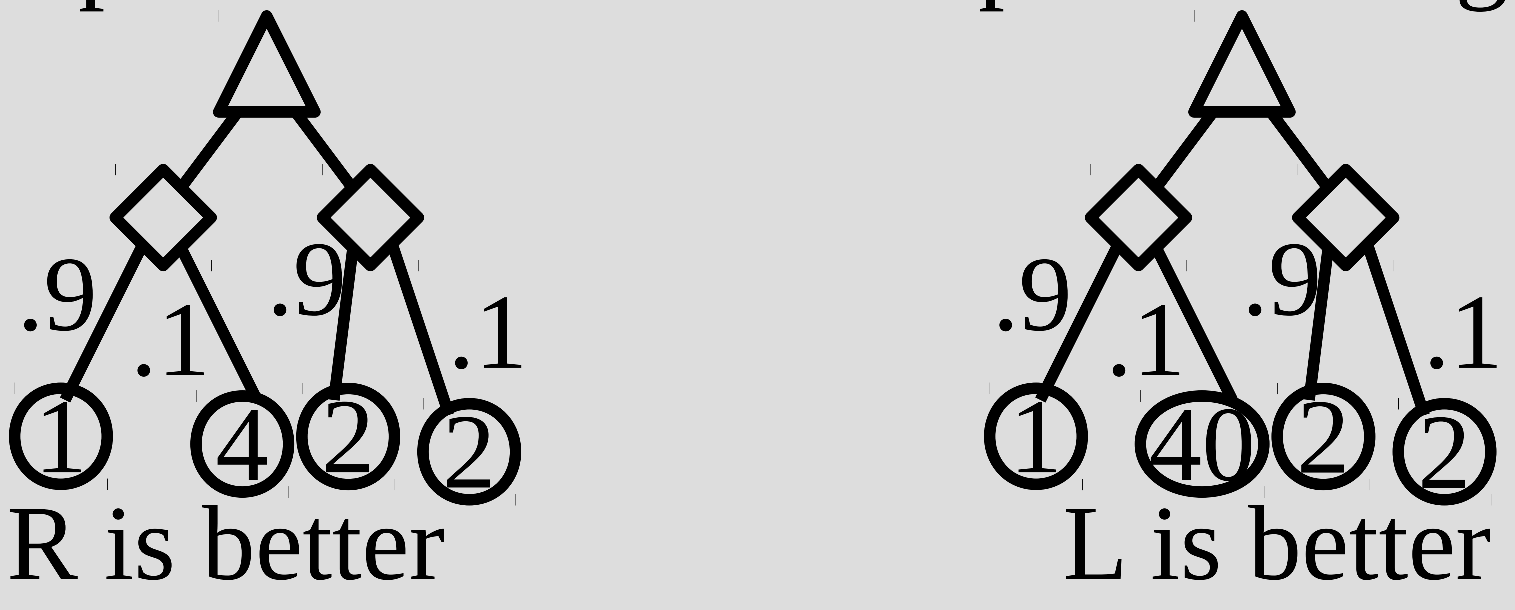


V(chance) = $-1 \cdot \frac{999}{1,000} + 100 \cdot \frac{1}{1,000} = -0.899$

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:



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 <u>sample</u>

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”?

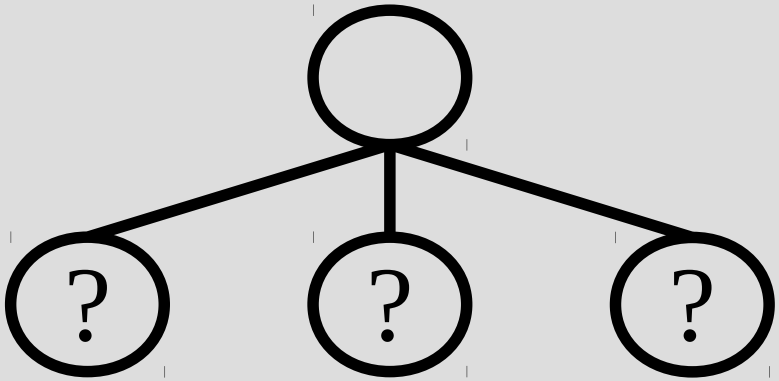
The “Upper Confidence Bound applied to Trees” UCT is commonly used:

$$\max\left(\frac{win(n)}{times(n)} + \sqrt{\frac{2\ln TotalTimes}{times(n)}}\right)$$

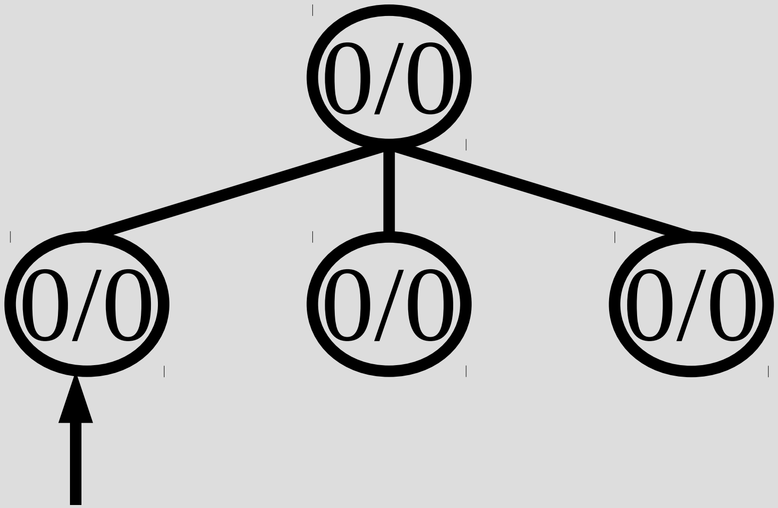
This ensures a trade off between checking branches you haven't explored much and exploring hopeful branches

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

MCTS

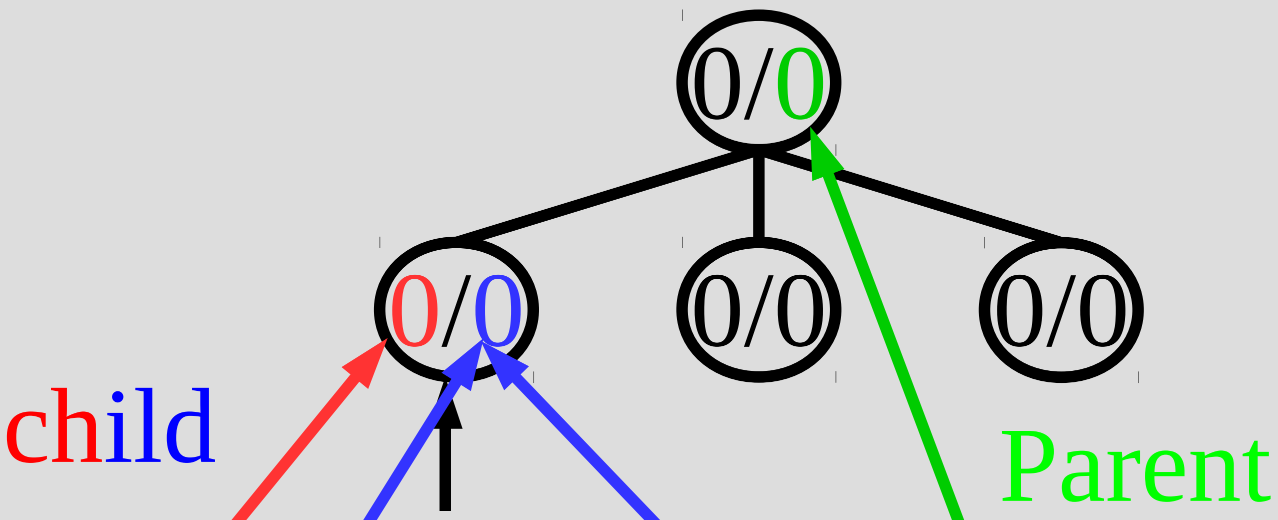


MCTS



$$\frac{win(n)}{times(n)} + \sqrt{\frac{2\ln TotalTimes}{times(n)}}$$
$$= \frac{0}{0} + \sqrt{\frac{2ln0}{0}}$$
$$= \infty$$

MCTS

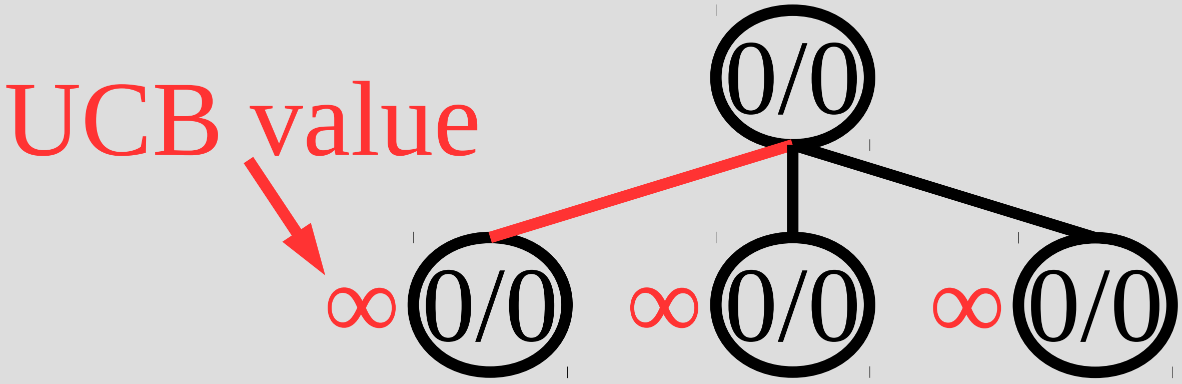


$$\frac{win(n)}{times(n)} + \sqrt{\frac{2\ln TotalTimes}{times(n)}}$$

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

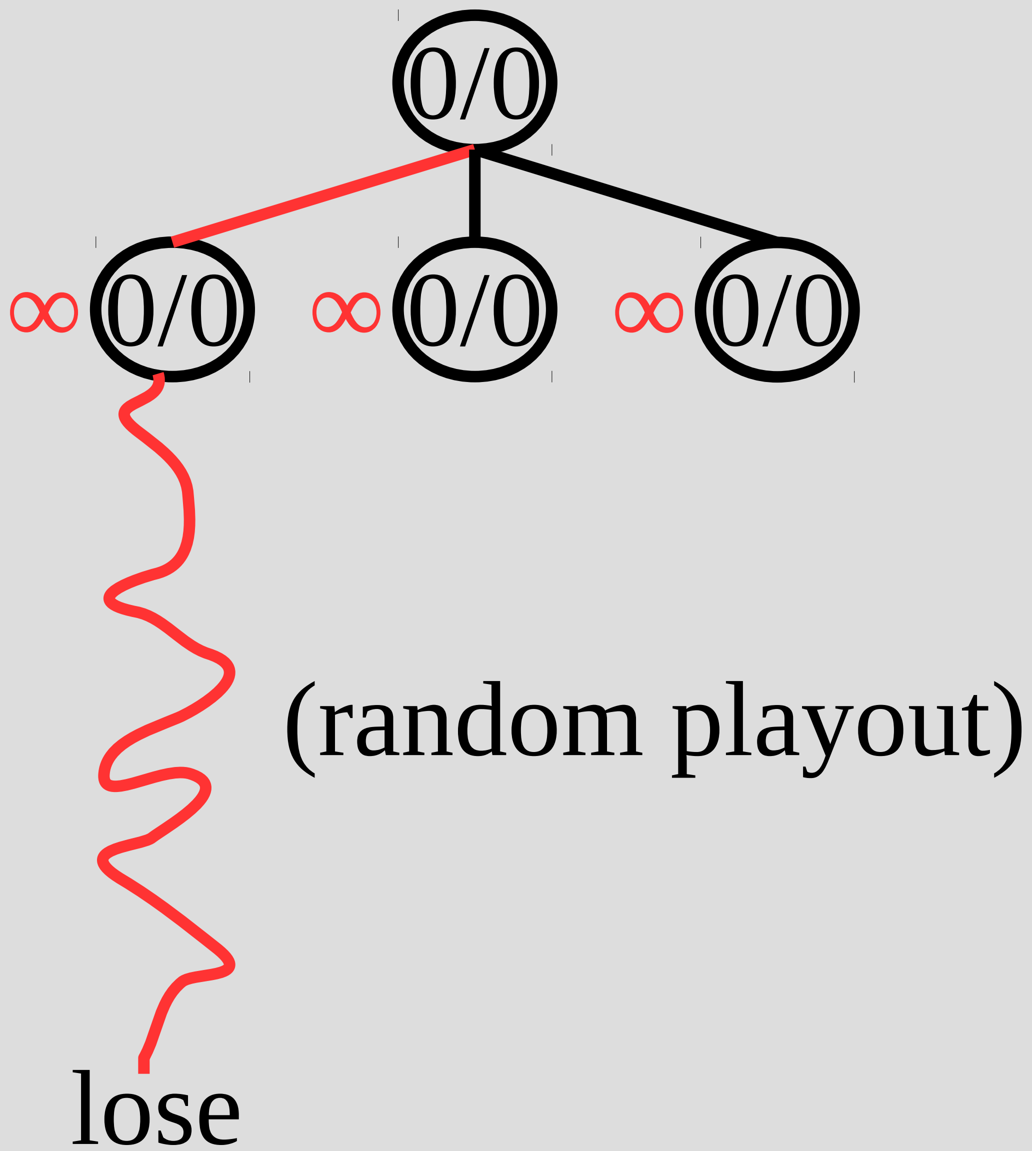
$$= \infty$$

MCTS

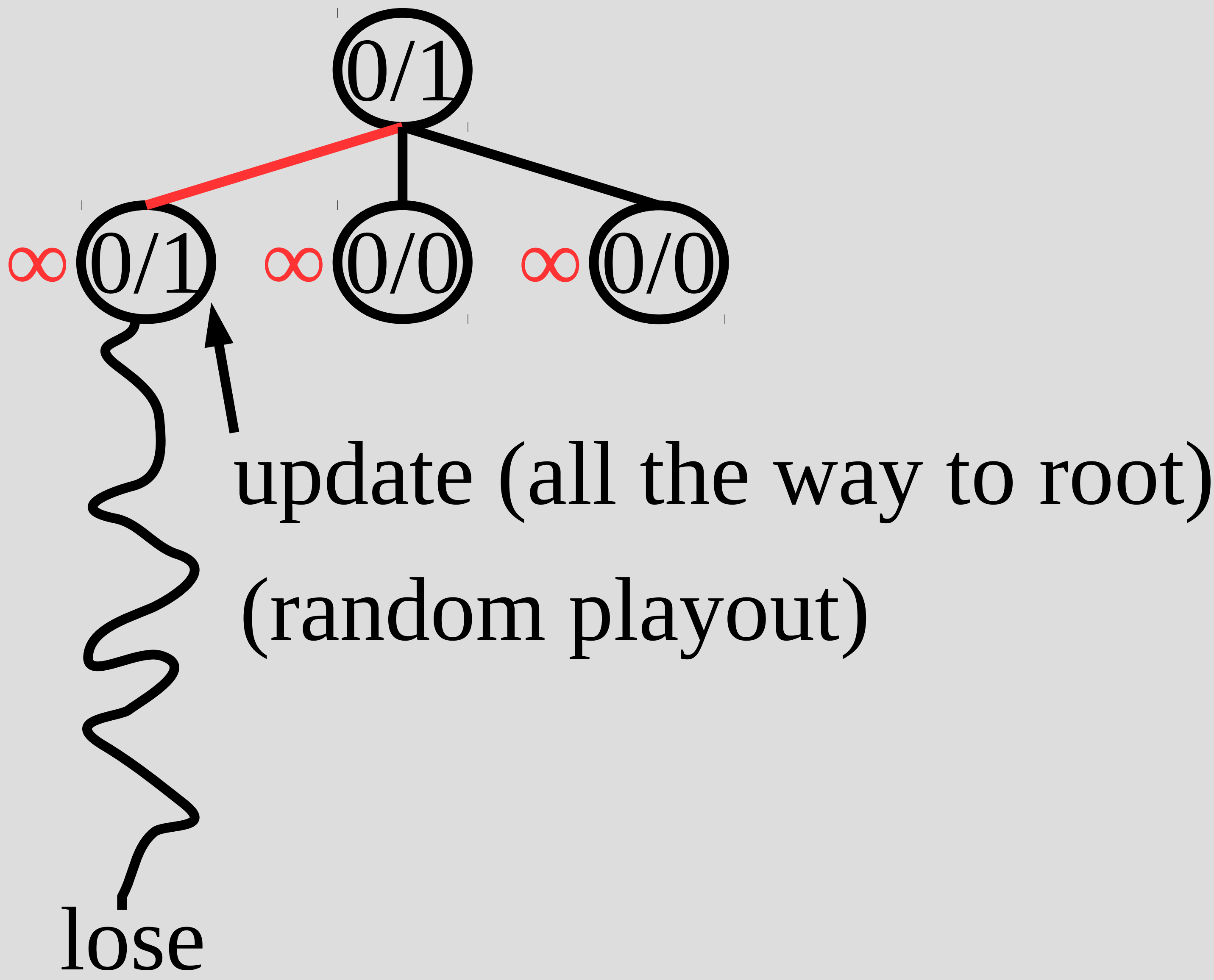


Pick max (I'll pick left-most)

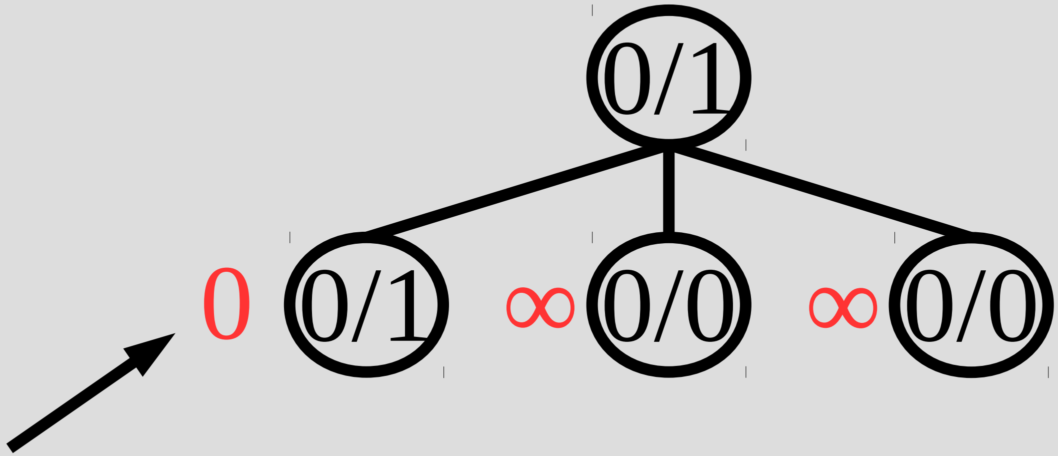
MCTS



MCTS



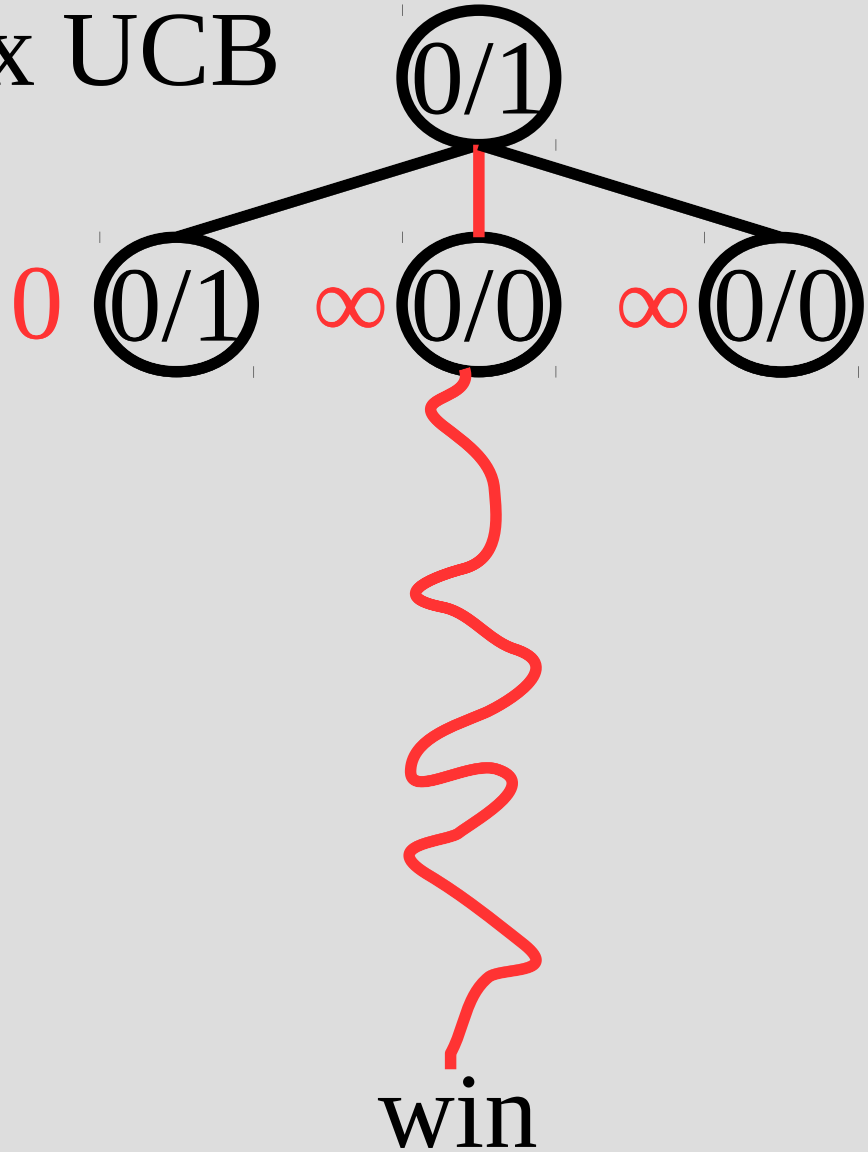
MCTS



update UCB values (all nodes)

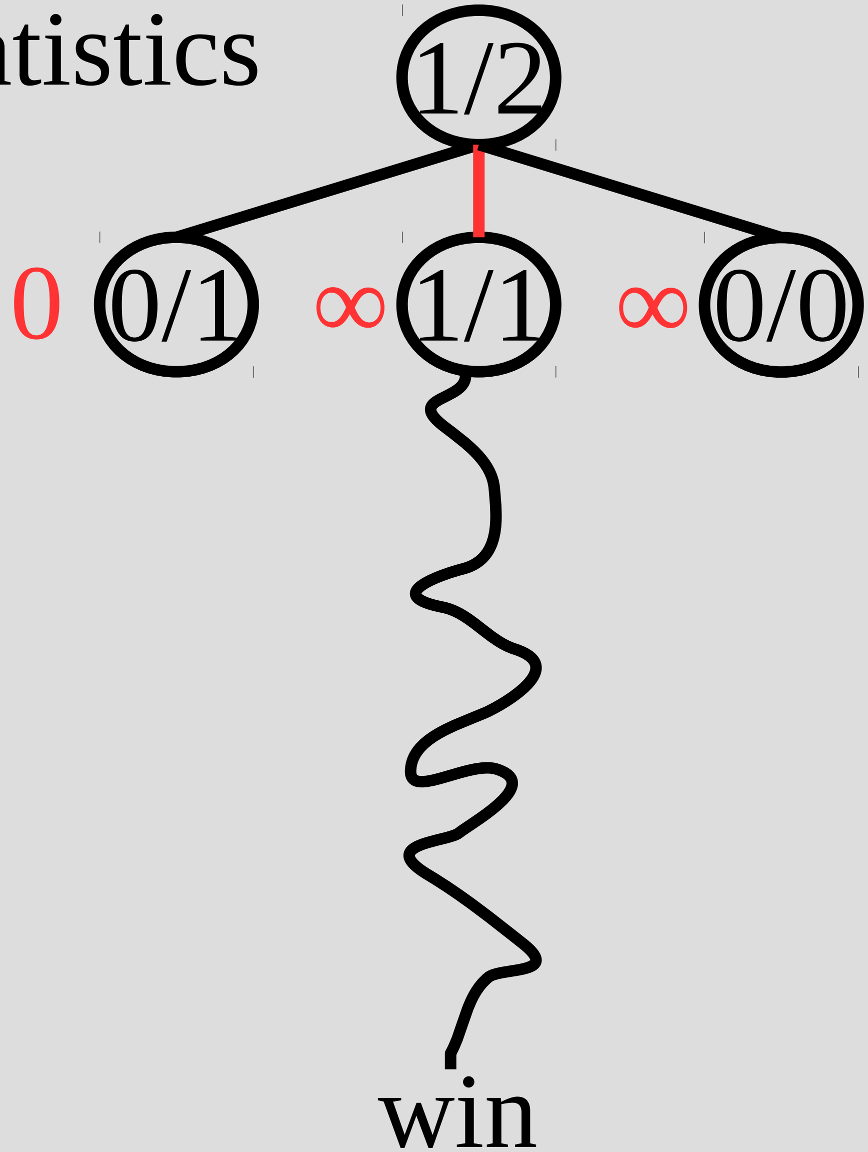
MCTS

select max UCB
& rollout



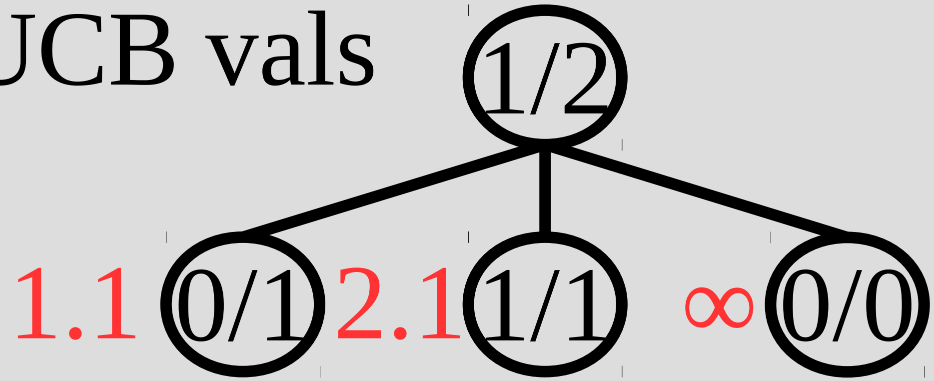
MCTS

update statistics



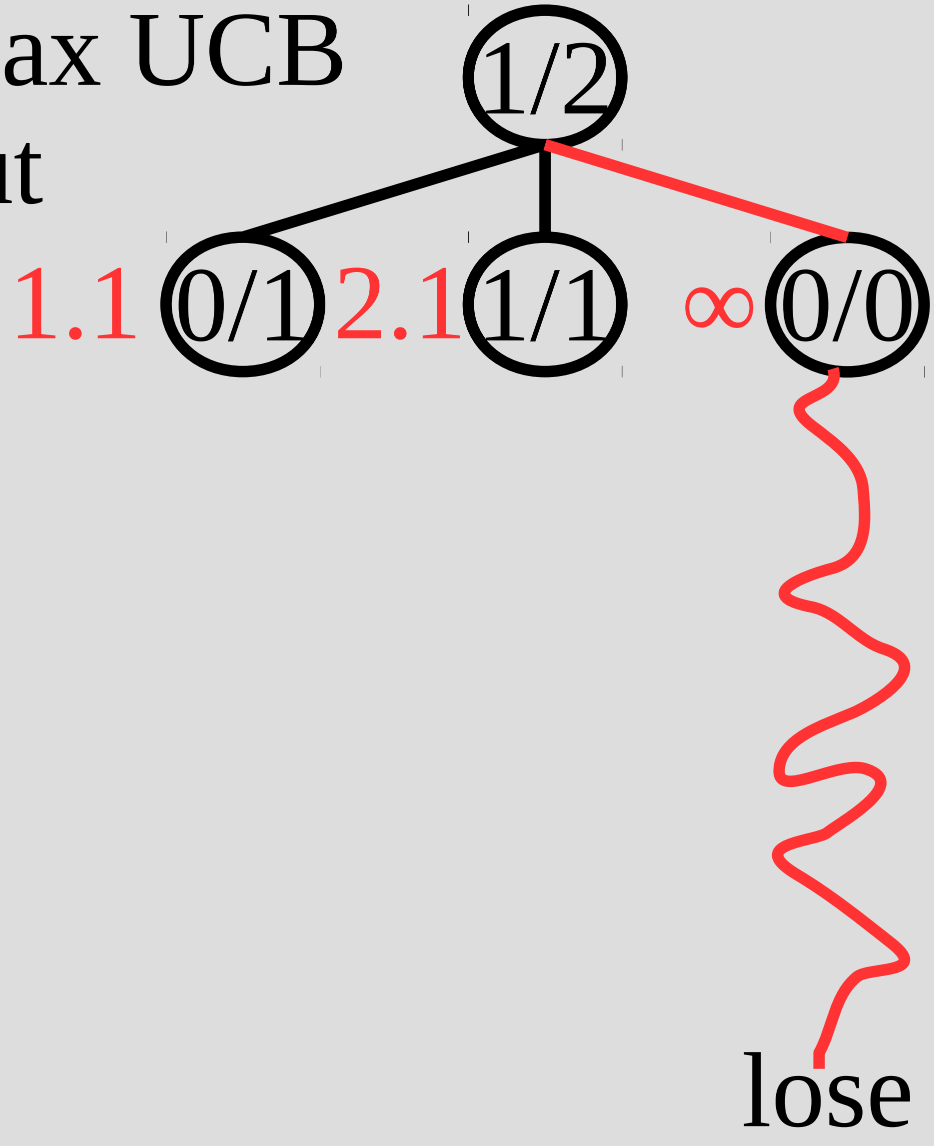
MCTS

update UCB vals



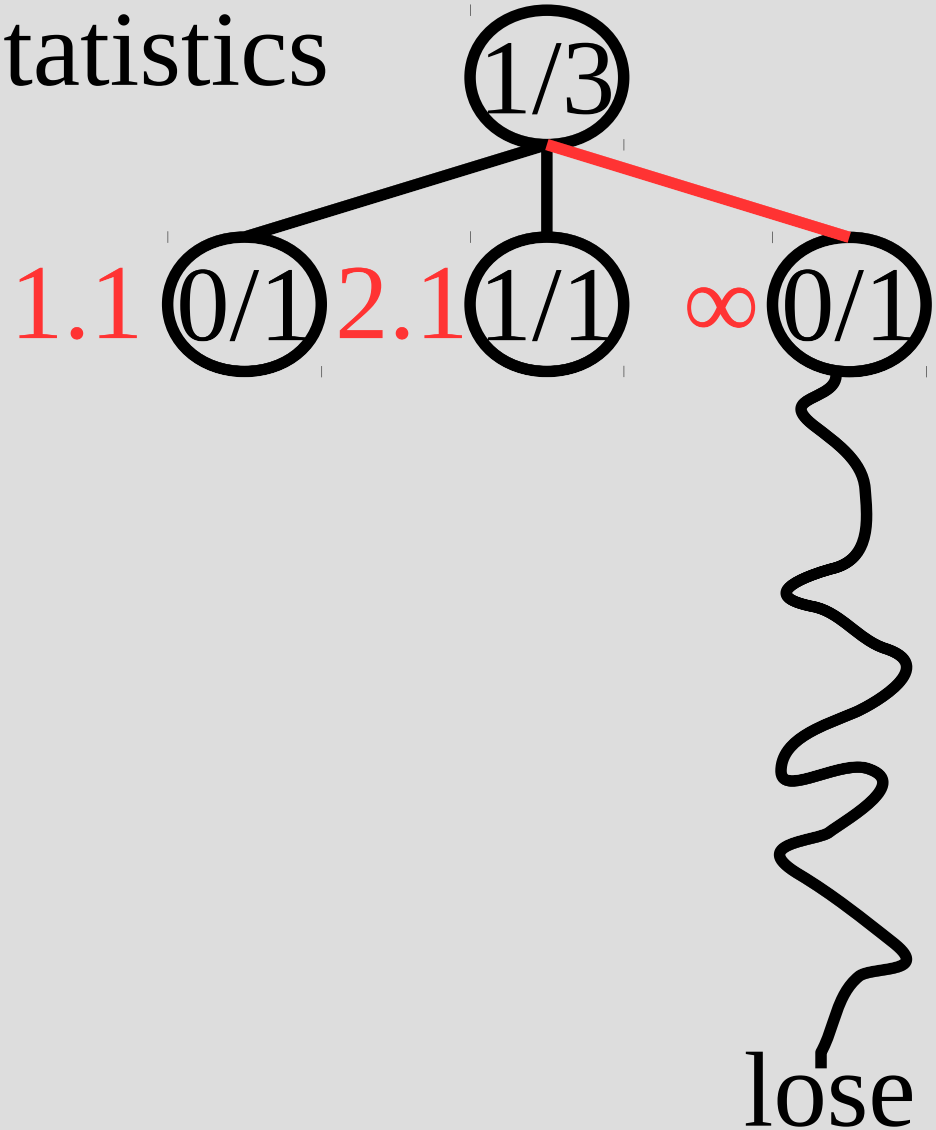
MCTS

select max UCB
& rollout



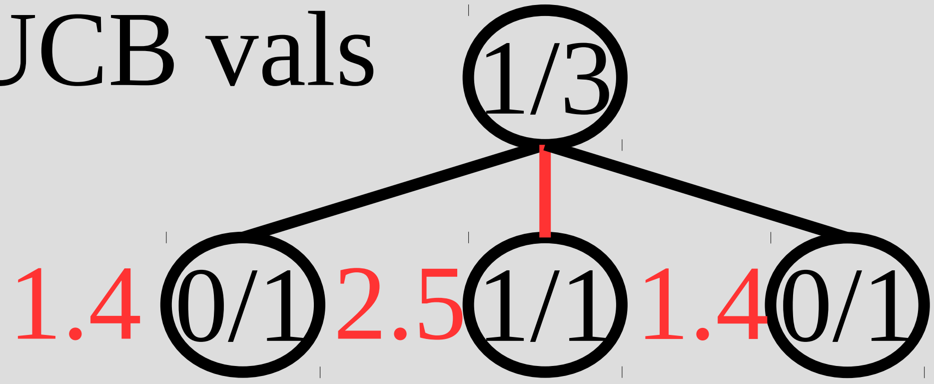
MCTS

update statistics



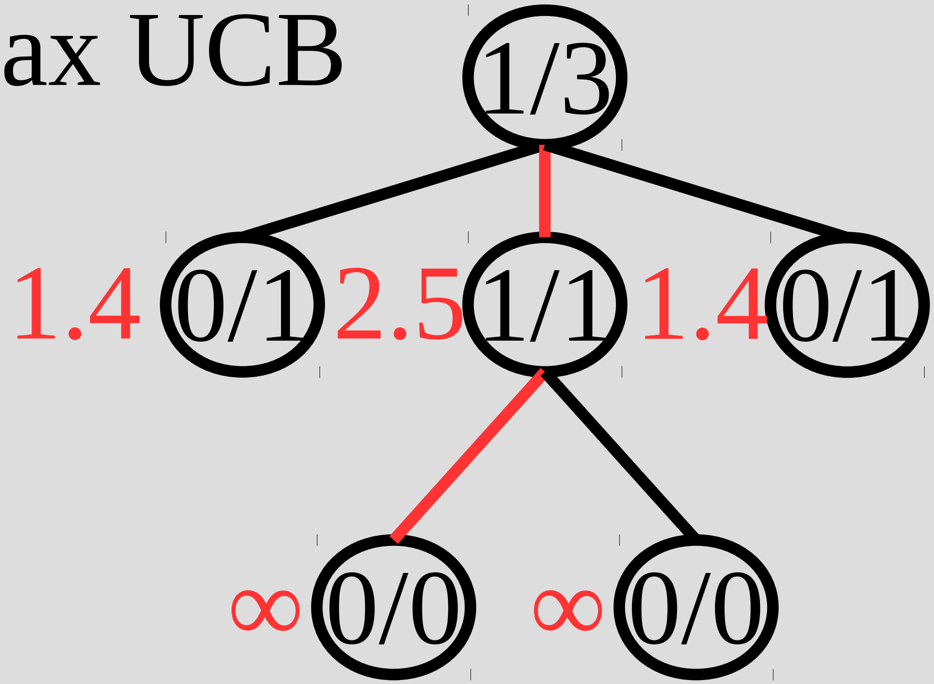
MCTS

update UCB vals



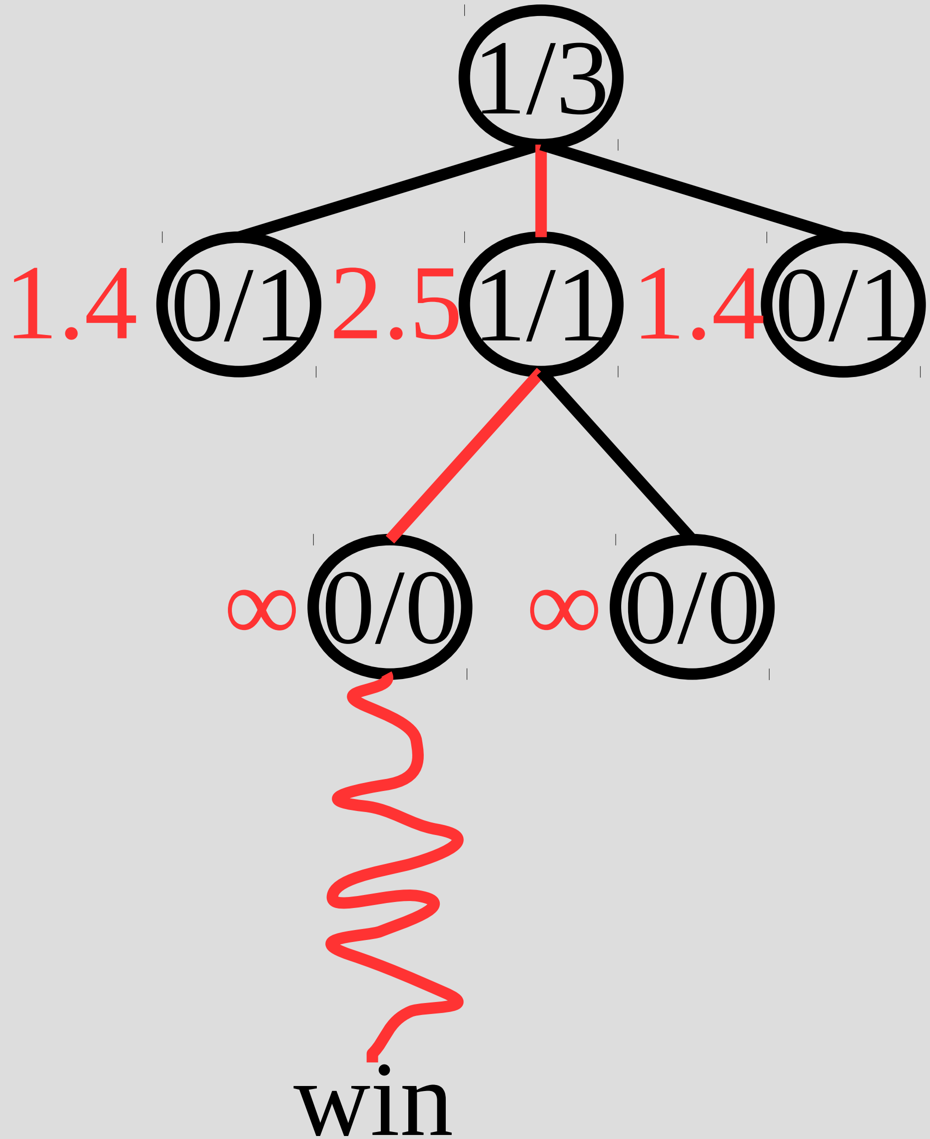
MCTS

select max UCB



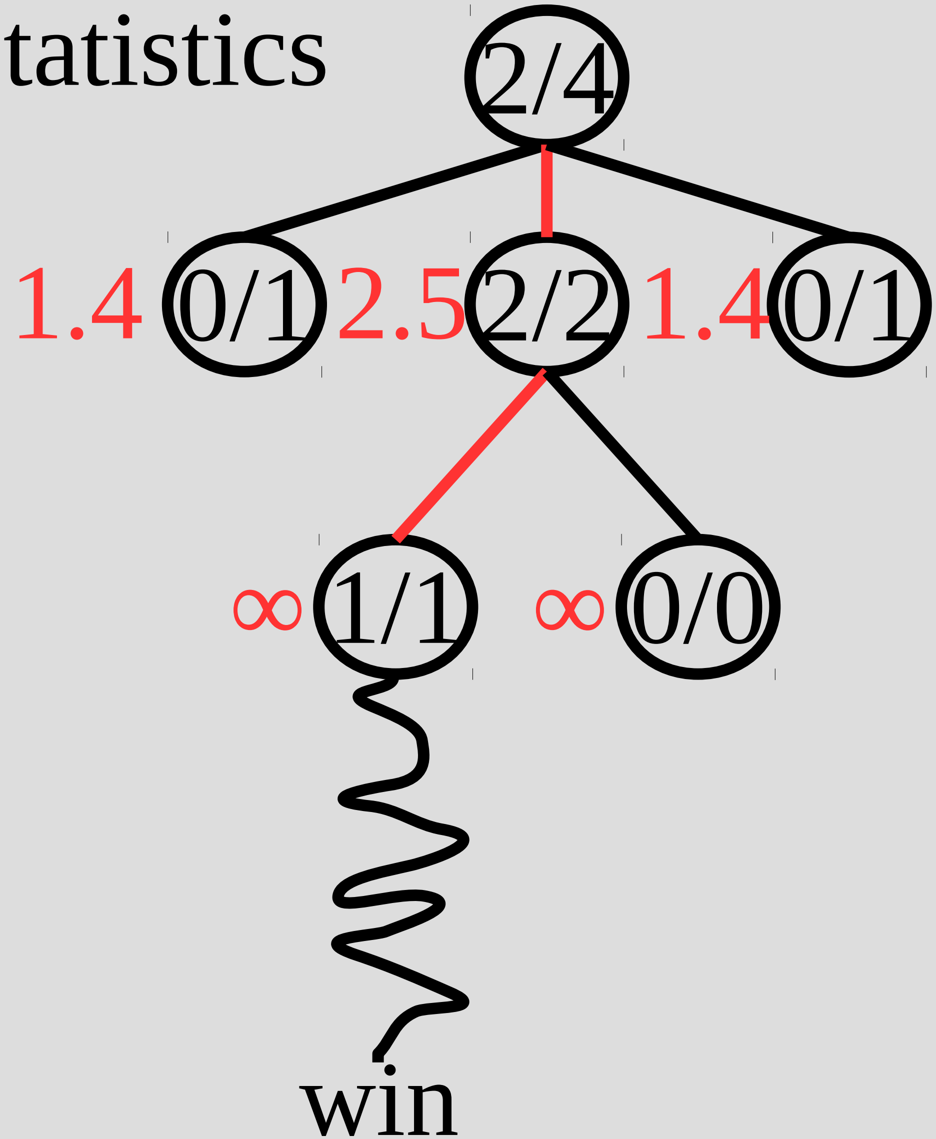
MCTS

rollout



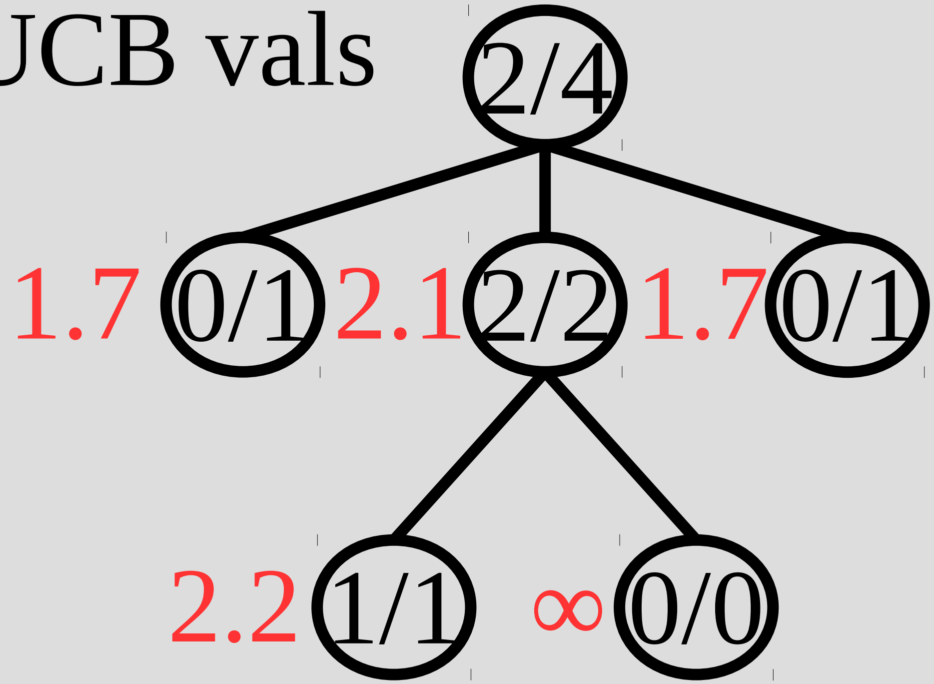
MCTS

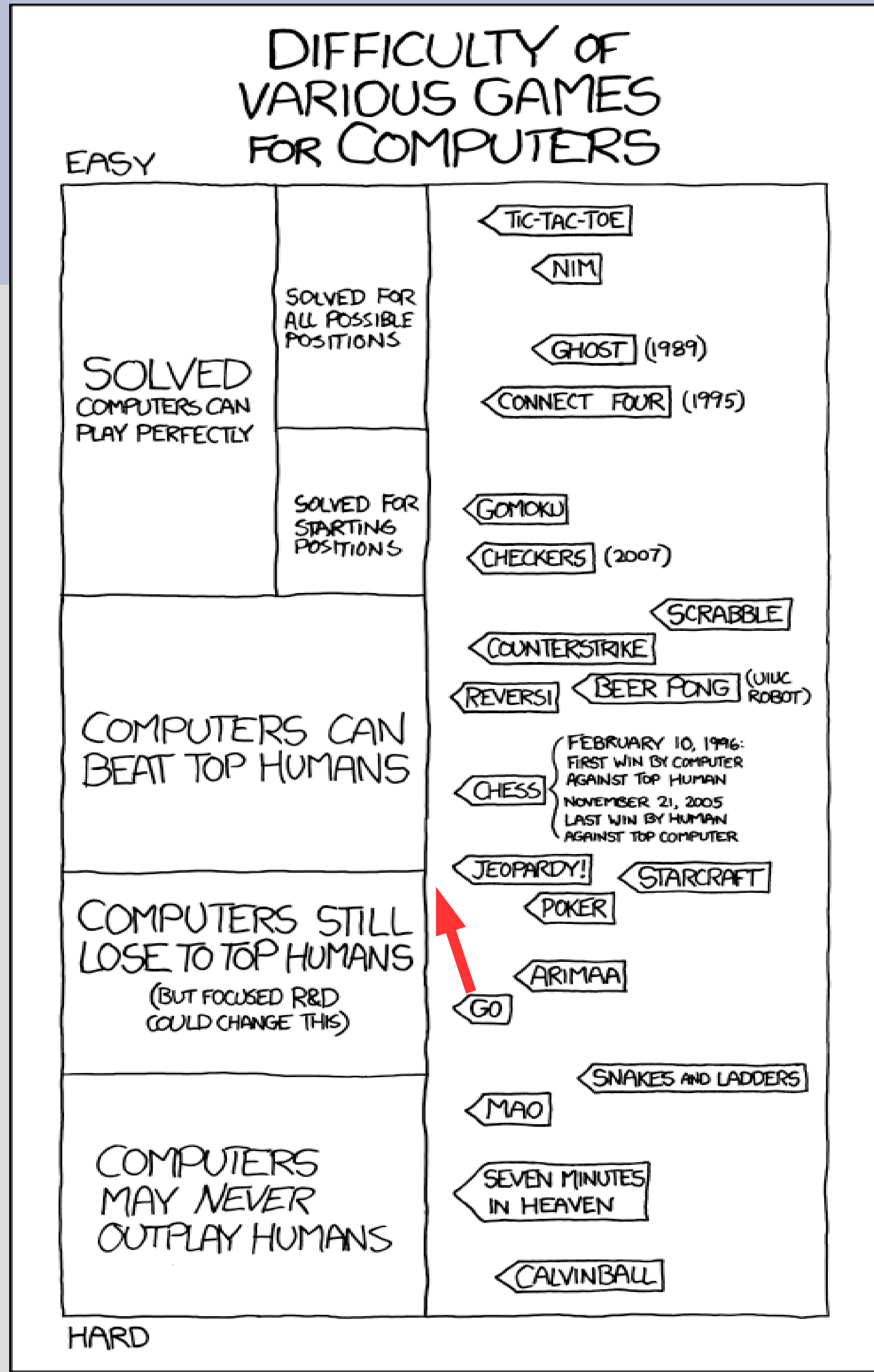
update statistics



MCTS

update UCB vals





DIFFICULTY OF VARIOUS GAMES FOR COMPUTERS
EASY
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
HARD
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