Lecture 18

CS 6380 Artificial Intelligence

Monte Carlo Tree Search

Department of Computer Science and Engineering

Indian Institute of Technology Madras (IITM), India



Recap

Beyond Minimax

- Minimax with alpha beta pruning leans on two crutches
  - A branching factor small enough to search several plies deep
  - An evaluation function that scores cutoff states sensibly
- Some games deny us both
  - Go has roughly 250 legal moves per position
  - No simple board features reliably predict the winner in Go
- A different idea
  - Estimate the value of a position by playing many random games from it
  - The fraction of wins tells us how promising the position is

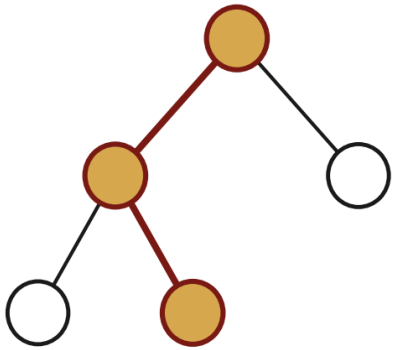
The Core Idea of MCTS

- Evaluate positions by simulation instead of a handcrafted function
  - Play out a complete game with a rollout policy and record the result
- Average the outcomes of many playouts
  - The average win rate estimates the value of a node
- Grow the search tree gradually and asymmetrically
  - Promising moves receive more simulations and deeper subtrees
  - Weak moves are still visited occasionally so nothing is ruled out too early
- Needs only the rules of the game and a source of randomness

The Four Phases

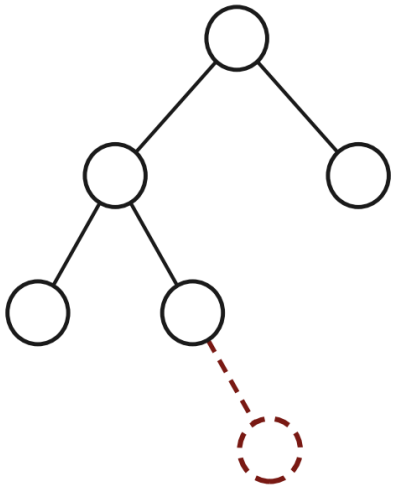
The Four Phases of MCTS

1 Selection



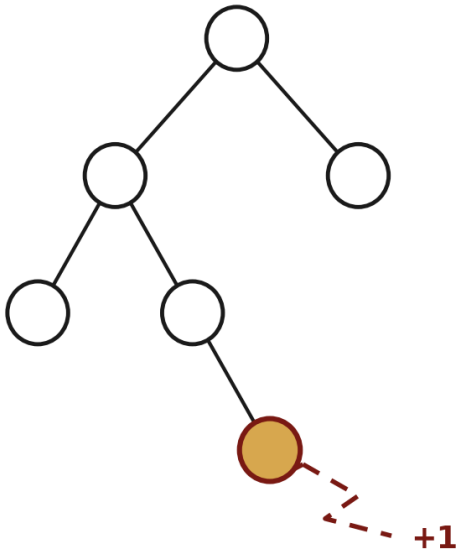
follow the tree policy
to a promising leaf

2 Expansion



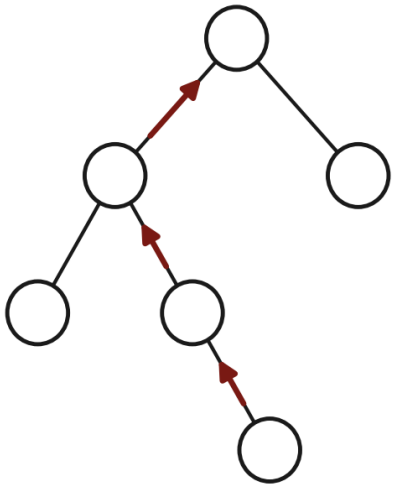
add one new child
to the tree

3 Simulation



play a random game
to the end

4 Backpropagation



send the result back
up the path

- Repeat all four phases until the time budget runs out, then play the most visited move at the root

Selection with UCB1

- Each selection step inside the tree is a small bandit problem
- Exploitation favors children with a high average win rate
- Exploration favors children that have been visited rarely
- The UCB1 rule combines the two pressures
  - $UCB1(n) = v(n) + C \sqrt{(\ln N(parent) / N(n))}$
  - v(n) is the average result and N counts visits
- The constant C controls the tradeoff and is tuned per game
- Always pick the child with the highest UCB1 score

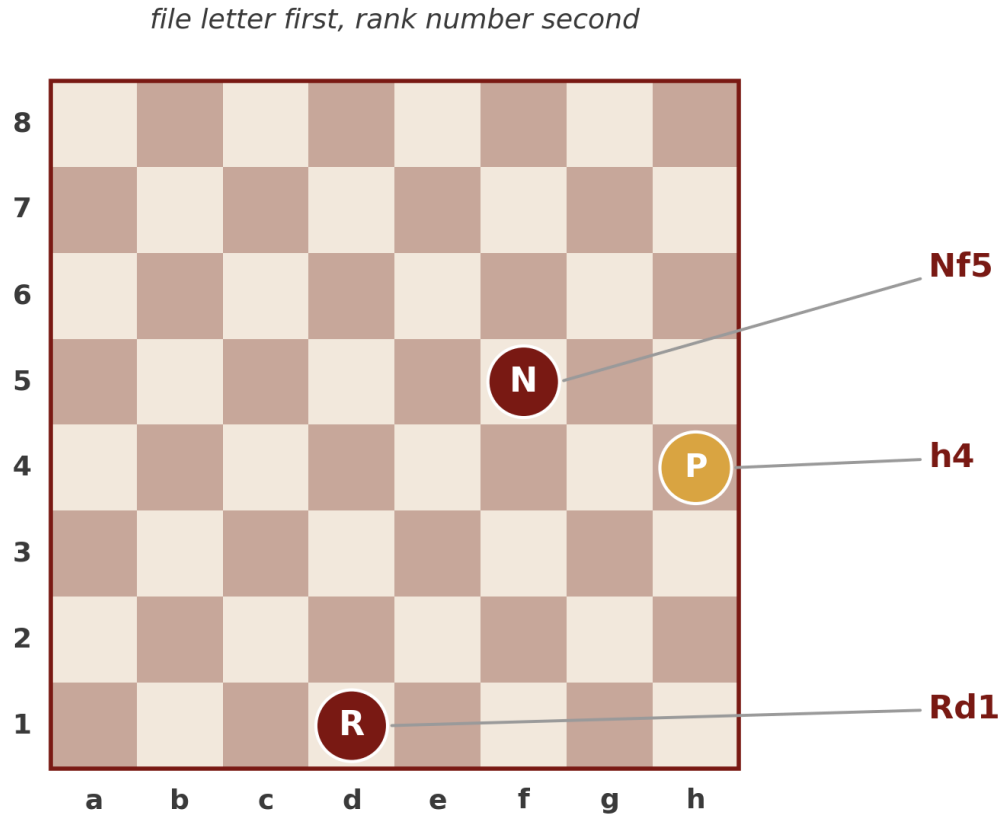
MCTS by Hand, a Worked Example

Seven iterations of a chess position, every number computed on the slide



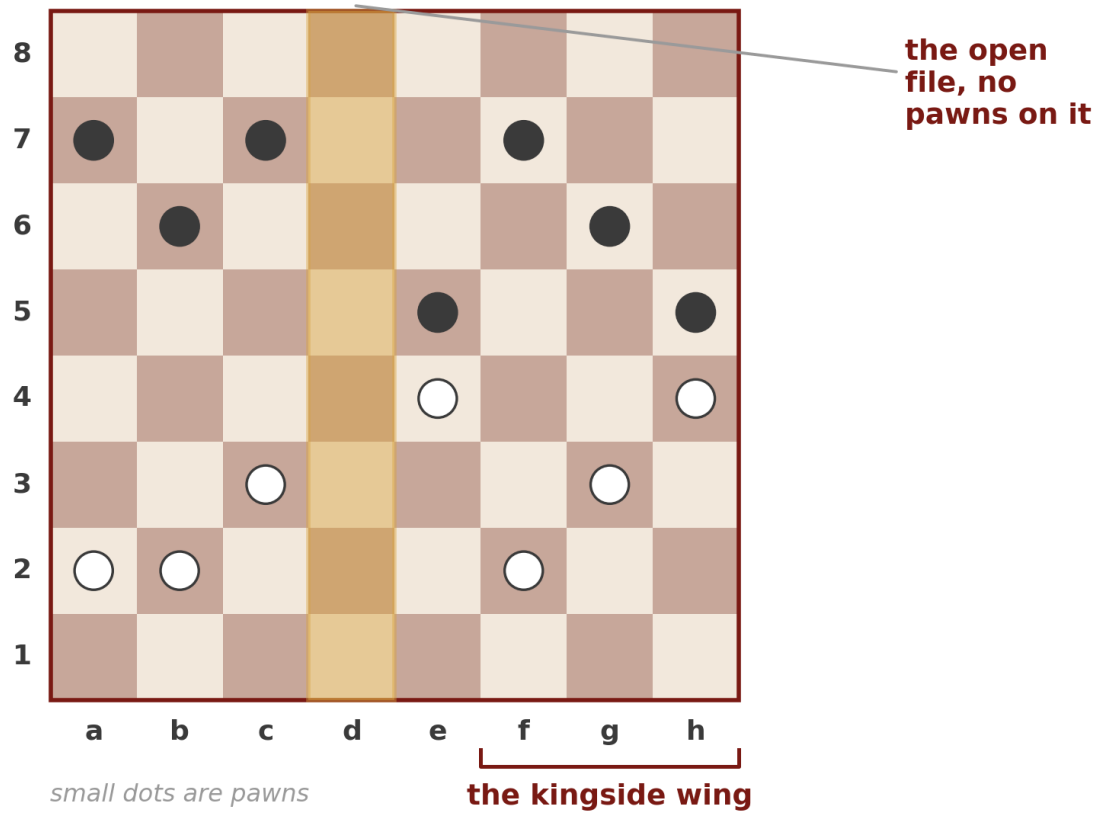
Chess Notation in One Minute

- Every square has a name, file letter first, rank number second
  - Files run a to h from left to right, ranks 1 to 8 from White's side
- Pieces are written K, Q, R, B, N for king, queen, rook, bishop, knight
- A move names the piece letter and its destination square
  - Nf5 places a knight on the square f5
  - Rd1 places a rook on the square d1
  - Pawn moves name only the square, so h4 pushes the h pawn to h4
- That is every piece of notation this lecture needs



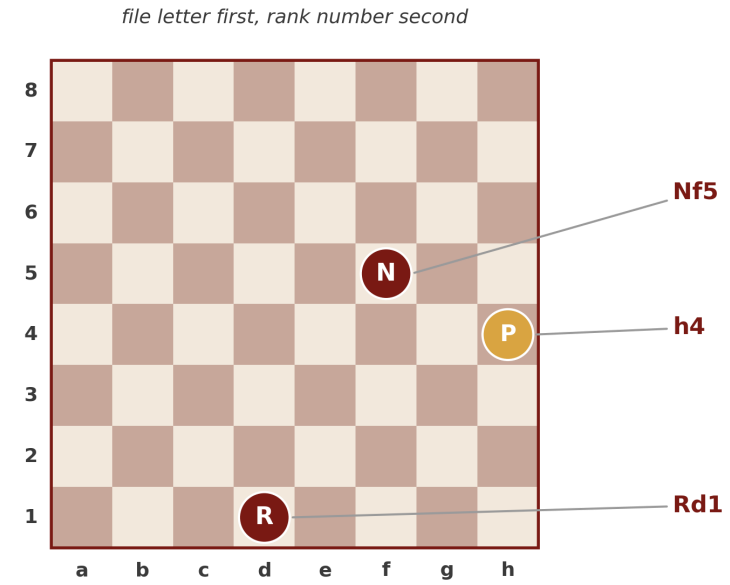
Five Chess Terms We Will Use

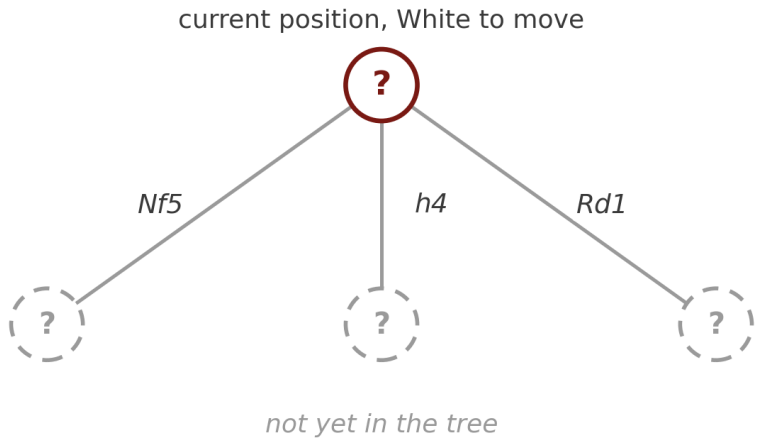
- **Middlegame** the phase after both sides have developed, where plans collide
- **Ply** one move by one player, so a full move pair is two plies
- **Candidate moves** the short list a player seriously considers in a position
- **Open file** a column with no pawns on it, the natural home for a rook
- **Wing** the flank away from the centre, the kingside or the queenside



The Setup

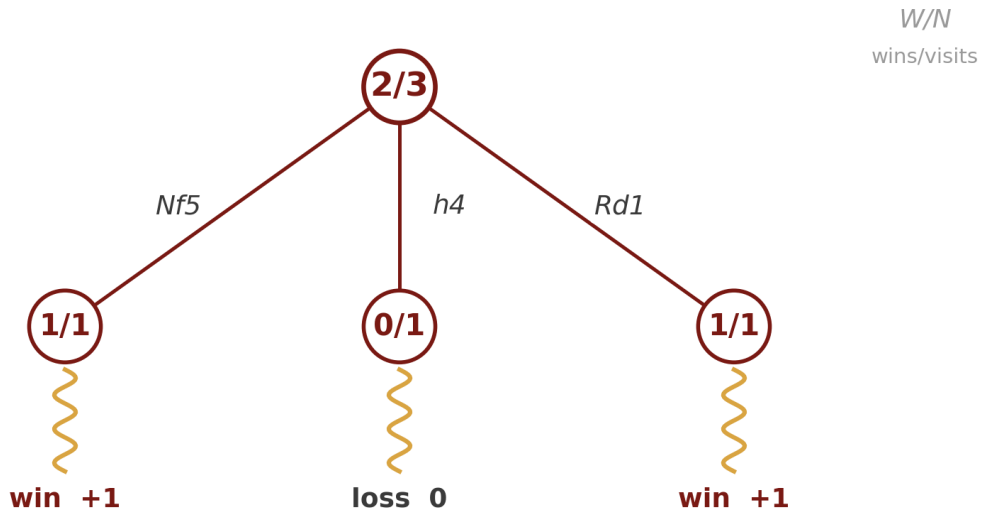
- A chess middlegame, White to move, three candidate moves
  - **Nf5** an aggressive knight jump into enemy territory
  - **h4** a pawn push on the wing
  - **Rd1** a quiet rook move to the open file
- Every tree node stores two numbers, written **W/N**
  - W is the total wins credited, N is the number of visits
  - v(n) = W/N is the average result that UCB1 uses
- Rules of the trace
  - Exploration constant C = 1.4, roughly $\sqrt{2}$
  - A playout returns win = 1 or loss = 0, recorded for White
  - A simulation budget of seven iterations, then we must move





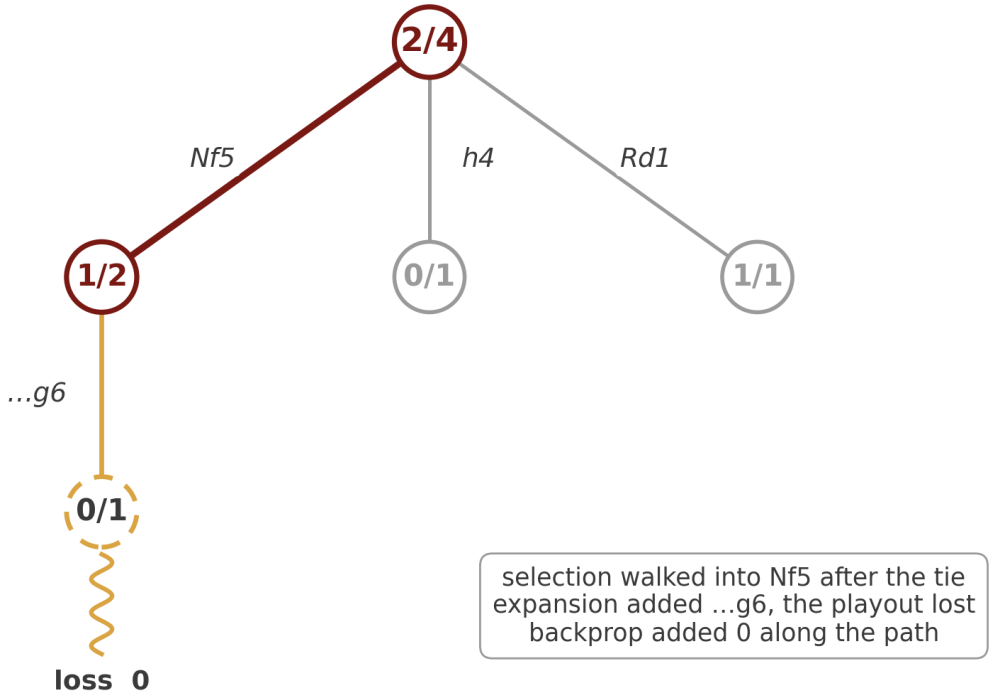
Iterations 1 to 3, Try Every Move Once

- The tree starts as only the root, so there is nothing to select yet
  - An unvisited child has N = 0, so its UCB1 value is infinite
  - Untried moves are therefore always expanded first
- **Iteration 1** expand Nf5, random playout → win, so Nf5 reads 1/1
- **Iteration 2** expand h4, random playout → loss, so h4 reads 0/1
- **Iteration 3** expand Rd1, random playout → win, so Rd1 reads 1/1
- Backpropagation updates the root after every playout
  - After three iterations the root reads 2/3, two wins in three games



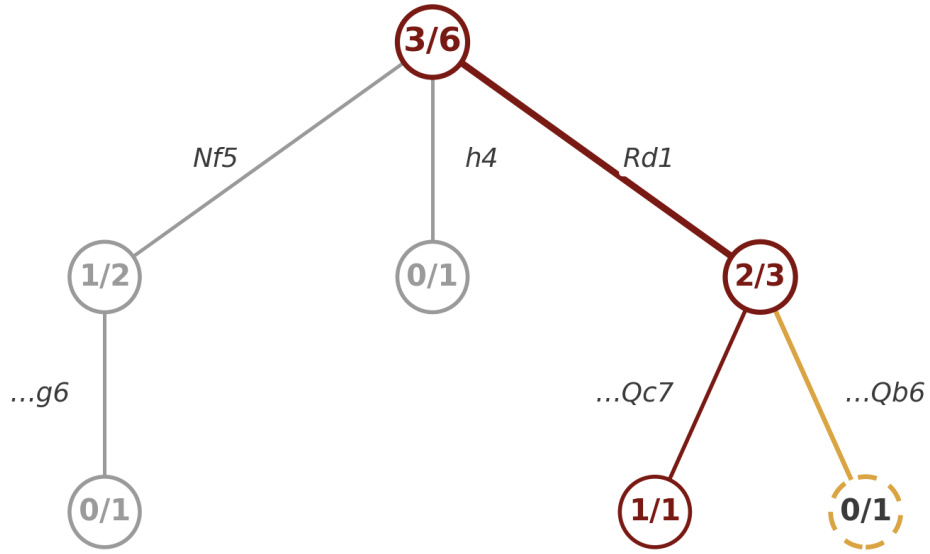
Iteration 4, UCB1 Takes Over

- All children are visited, so selection needs UCB1, with N(root) = 3
  - **Nf5** 1/1 + 1.4 √(ln 3 / 1) = 1.00 + 1.47 = 2.47
  - **h4** 0/1 + 1.4 √(ln 3 / 1) = 0.00 + 1.47 = 1.47
  - **Rd1** 1/1 + 1.4 √(ln 3 / 1) = 1.00 + 1.47 = 2.47
- Nf5 and Rd1 tie; break the tie by move order and walk into Nf5
- Nf5 is a leaf of the tree, so the walk stops there
- Expand one reply, …g6, and the playout is a loss
- Backpropagation makes …g6 0/1, drops Nf5 to 1/2, and the root reads 2/4



Iterations 5 and 6, Growth Where Wins Are

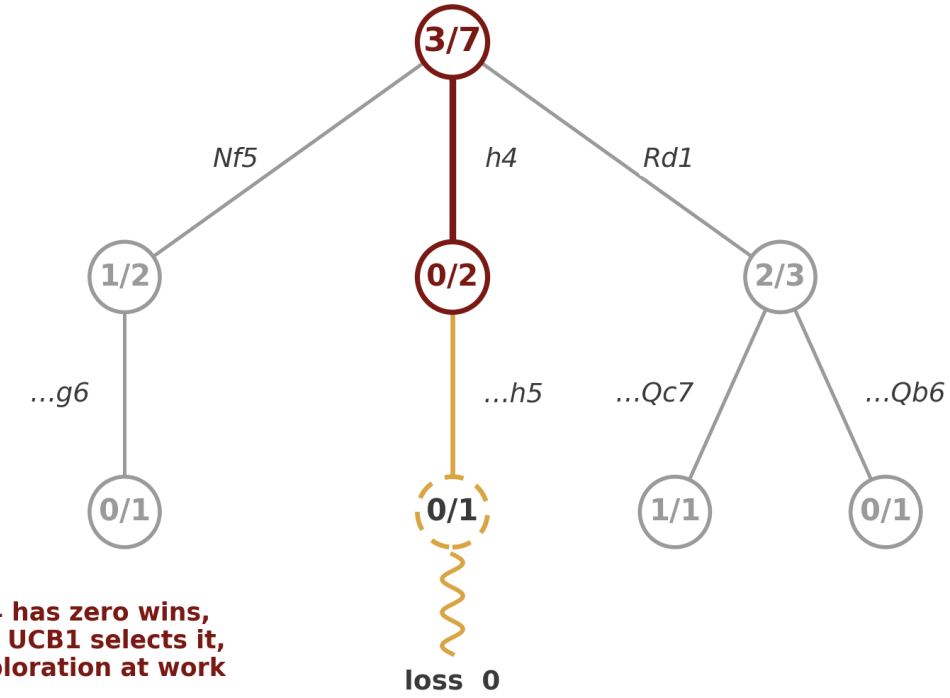
- **Iteration 5** with N(root) = 4 gives Nf5 1.67, h4 1.65, Rd1 2.65
  - Rd1 selected; expand …Qc7, the playout wins, so Rd1 reads 2/2 and root 3/5
- **Iteration 6** with N(root) = 5 gives Nf5 1.76, h4 1.78, Rd1 2.26
  - Rd1 selected again; expand …Qb6, the playout loses, so Rd1 reads 2/3 and root 3/6
- Both iterations flowed to the move that keeps winning its playouts
  - The tree is now two plies deep under Rd1 and shallow elsewhere
- Meanwhile h4 sits untouched, but its exploration bonus keeps climbing



the tree deepens under Rd1,
the move that is winning its playouts

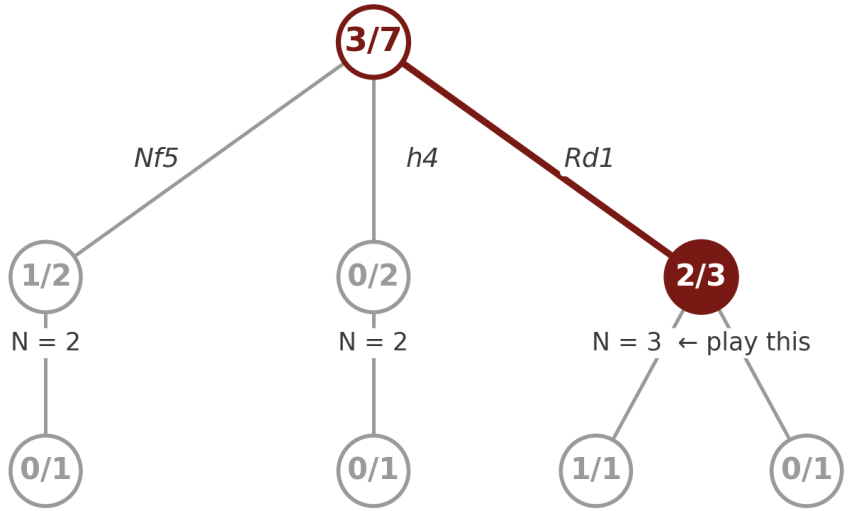
Iteration 7, Exploration Rescues the Underdog

- **With N(root) = 6** the scores are Nf5 1.83, h4 1.87, Rd1 1.75
- h4 has zero wins, yet its UCB1 score is now the highest
  - Its average is 0.00, but 1.4 √(ln 6 / 1) = 1.87 has grown large
  - Rd1 pays for its extra visits since its bonus shrank to 1.4 √(ln 6 / 3)
- Selection walks into h4; expand …h5, and the playout is a loss
- h4 ends the trace at 0/2, the root at 3/7
- The lesson is that no move is ever written off on one bad sample
  - Sooner or later every child is re-examined, however weak it looks



The Final Decision

- The budget is spent and the statistics at the root are final
  - **Nf5** 1/2, mean 0.50
  - **h4** 0/2, mean 0.00
  - **Rd1** 2/3, mean 0.67, and the most visits
- Play the most visited move, not the best average → **Rd1**
- Why visits rather than averages?
  - A lucky playout can inflate a mean built on one or two games
  - A large visit count is earned by surviving many UCB1 comparisons
  - Here the two criteria agree; with tiny budgets they often do not



budget spent, so play the most visited move at the root, Rd1

Your Turn, Continue the Trace

- Iteration 8 begins where we stopped, with N(root) = 7, ln 7 ≈ 1.95, C = 1.4
- **Q1** Compute UCB1 for Nf5 (1/2), h4 (0/2), and Rd1 (2/3)
- **Q2** Which child does selection walk into, and why does the walk stop there?
- **Q3** Its playout ends in a win, so write every W/N that changes during backpropagation
- **Q4** After iteration 8, has the move we would play changed? Justify with visit counts
- As a sanity check, your three scores in Q1 should round to 1.88, 1.38, and 1.79

MCTS in Practice

- Anytime behavior
  - Stop whenever you like and act on the current visit counts
- No evaluation function is required since playouts carry the signal
- Given enough simulations the estimates converge to minimax values
- The tree grows deep exactly where the strongest play lies
- AlphaGo combined MCTS with learned policy and value networks
  - Guided rollouts and better leaf evaluation beat top human players