CMSC 474, Introduction to Game Theory

14. Sub-game Perfect Equilibrium

Mohammad T. Hajiaghayi
University of Maryland

Definition of Subgame-Perfect Equilibrium

- Given a perfect-information extensive-form game *G*, the **subgame** of *G* rooted at node *h* is the restriction of *G* to the descendants of *h*
- Now we can define a refinement of the Nash equilibrium that eliminates noncredible threats
- A **subgame-perfect equilibrium** (SPE) is a strategy profile **s** such that for every subgame G' of *G*, the restriction of **s** to G' is a Nash equilibrium of G'
  - Since *G* itself is is a subgame of *G*, every SPE is also a Nash equilibrium
- Every perfect-information extensive-form game has at least 1 SPE
  - Can prove this by induction on the height of the game tree

Example

- Recall that we have three Nash equilibria:

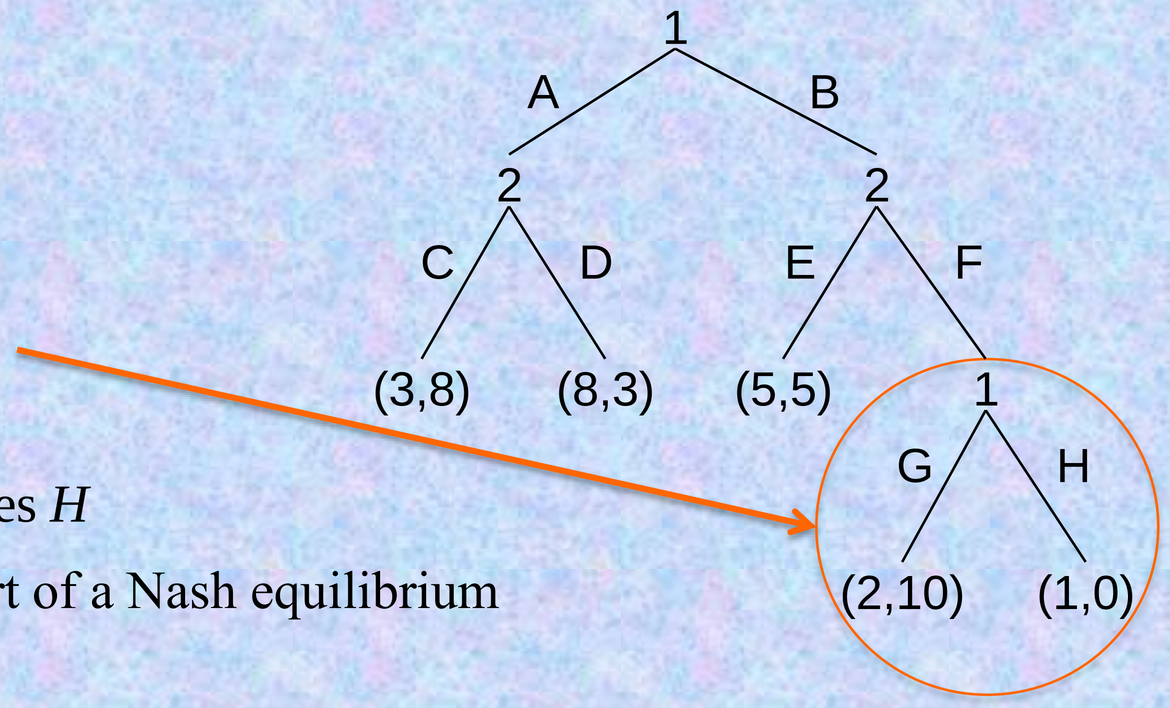
  {(*A*, *G*), (*C*, *F*)}

  {(*A*, *H*), (*C*, *F*)}

  {(*B*, *H*), (*C*, *E*)}

- Consider this subgame:
  - For agent 1,

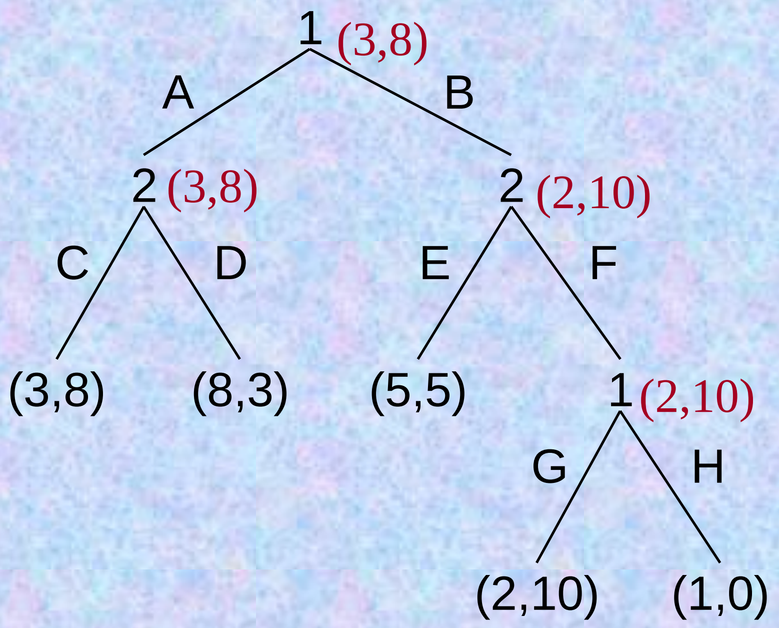
    G strictly dominates *H*
  - Thus *H* can't be part of a Nash equilibrium



- This excludes {(*A*, *H*), (*C*, *F*)} and {(*B*, *H*), (*C*, *E*)}
- Just one subgame-perfect equilibrium
  - {(*A*, *G*), (*C*, *F*)}

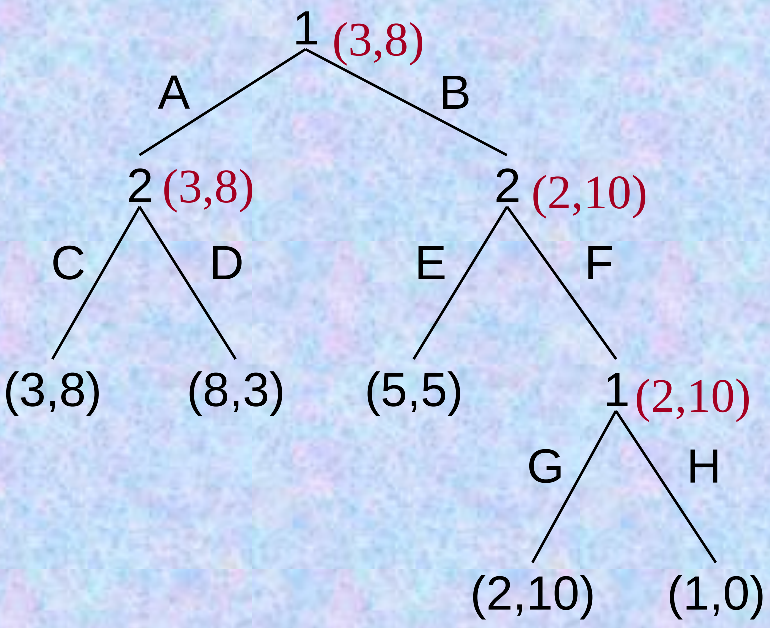
Backward Induction

- To find subgame-perfect equilibria, we can use **backward induction**
- Identify the Nash equilibria in the bottom-most nodes
  - Assume they'll be played if the game ever reaches these nodes
- For each node *h*, recursively compute a vector $v_h = (v_{h1}, \ldots, v_{hn})$ that gives every agent's equilibrium utility
  - At each node *h*,
    - If *i* is the agent to move, then *i*'s equilibrium action is to move to a child *h'* of *h* for which *i*'s equilibrium utility $v_{h'i}$ is highest



Backward Induction

- To find subgame-perfect equilibria, we can use **backward induction**
- Identify the Nash equilibria in the bottom-most nodes
  - Assume they'll be played if the game ever reaches these nodes



procedure backward-induction(h)

if $h \in Z$ **then return u**(h)

bestv = $(-\infty, \ldots, -\infty)$

forall $a \in \chi(h)$ **do**

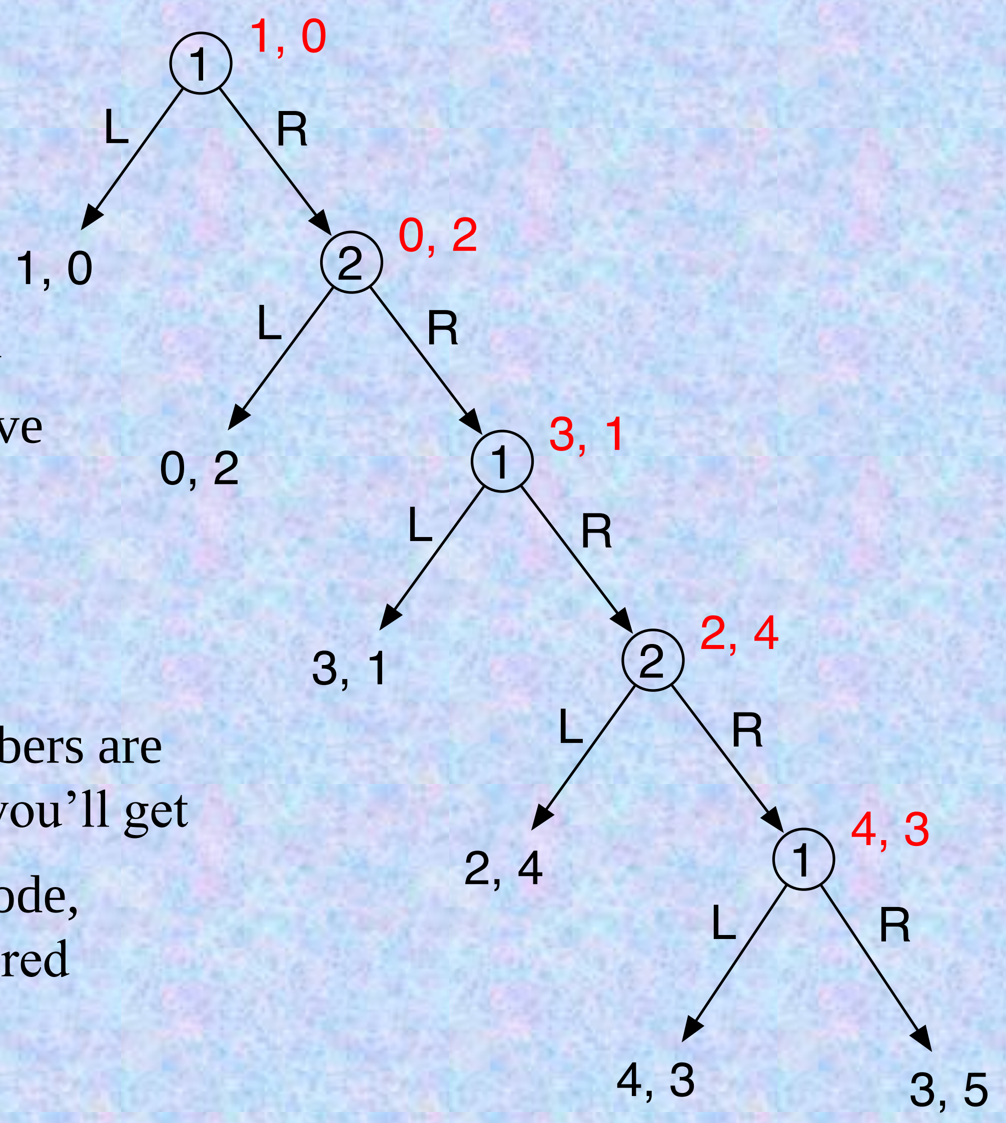
v = backward-induction($\sigma(h,a)$)

if v[$\rho(h)$] > **bestv**[$\rho(h)$] **then bestv** = **v**

return bestv

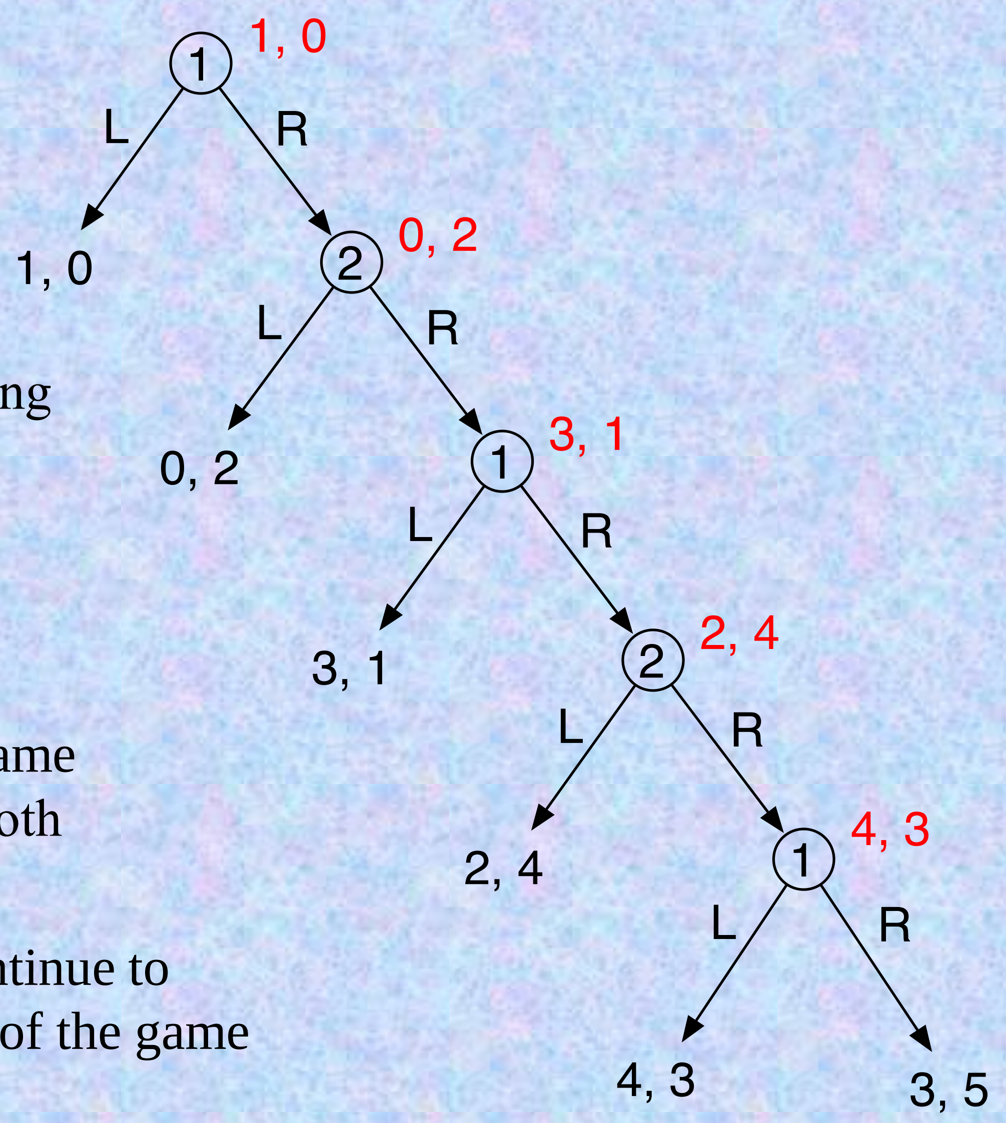
The Centipede Game

- The players move in alternation
  - Player 1 makes the first move
  - Each player can go either *Left* or *Right*
- At each terminal node, the numbers are how many pieces of chocolate you'll get
  - Next to each nonterminal node, I've put the SPE payoffs in red



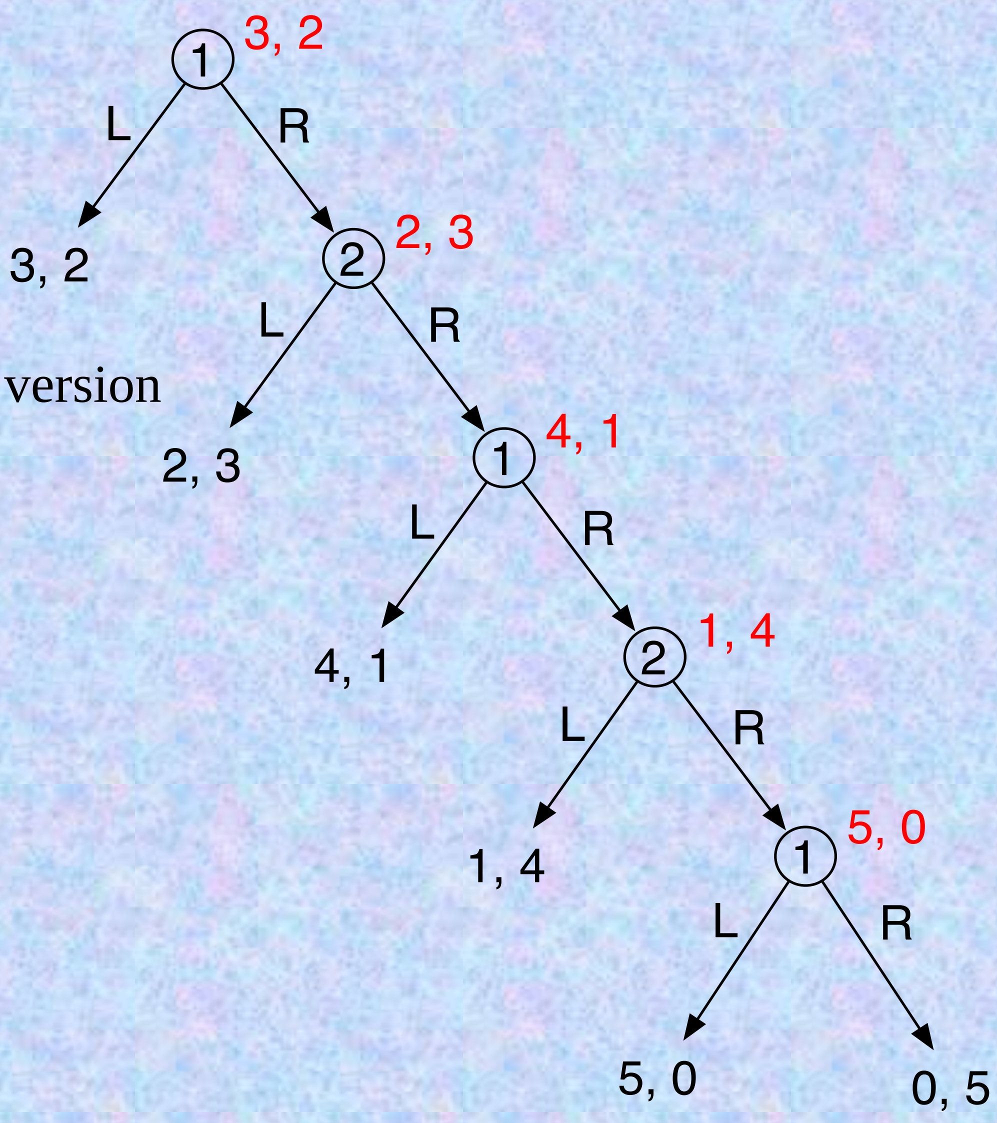
A Problem with Backward Induction

- Can extend the centipede game to any length
- The only SPE is for each agent always to move *Left*
- But this isn't intuitively appealing
- Seems unlikely that one would want to choose *Left* near the start of the game
  - If the agents continue the game for several moves, they'll both get higher payoffs
- In lab experiments, subjects continue to choose *Right* until near the end of the game



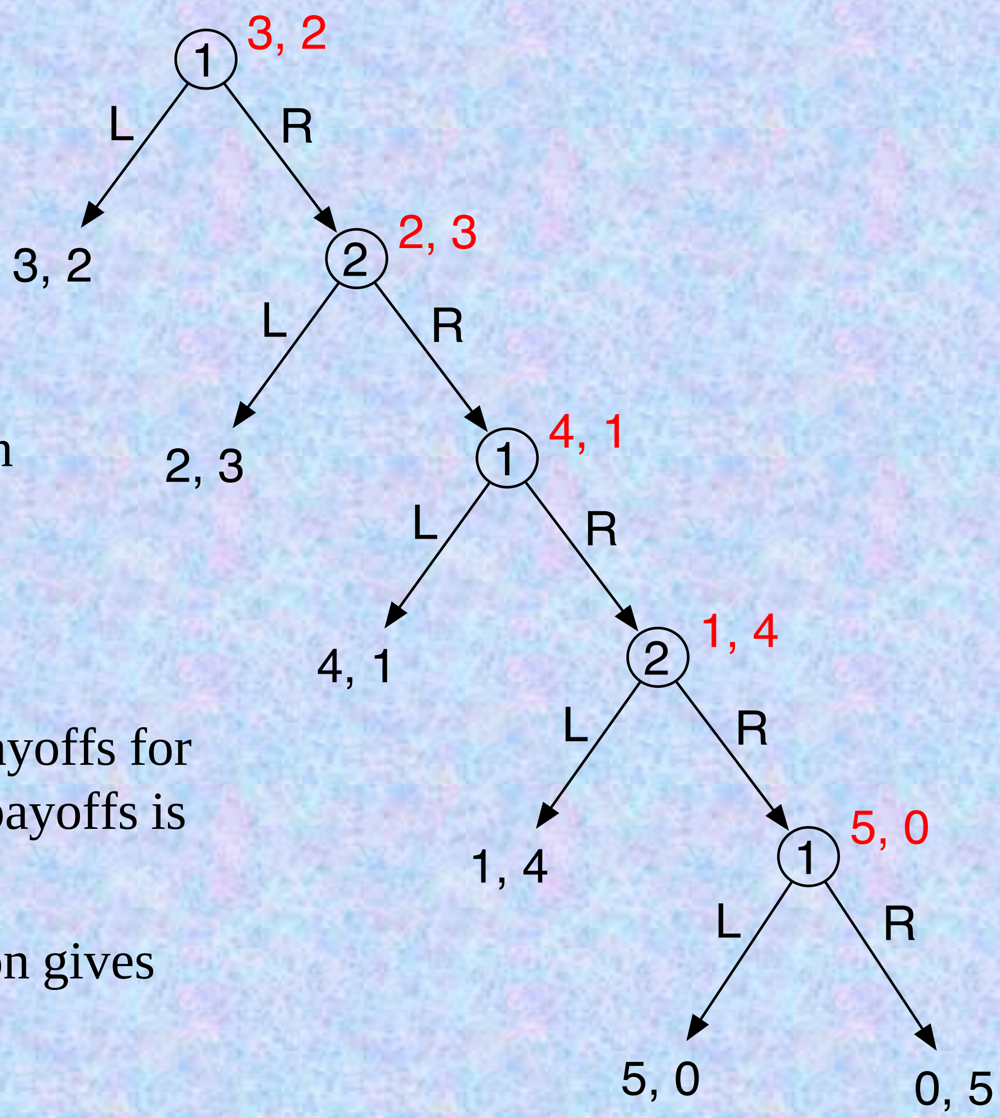
Constant-Sum Centipede Game

- Now consider a **constant-sum** version of the centipede game
- At every node, $u_2 = 5 - u_1$



Constant-Sum Centipede Game

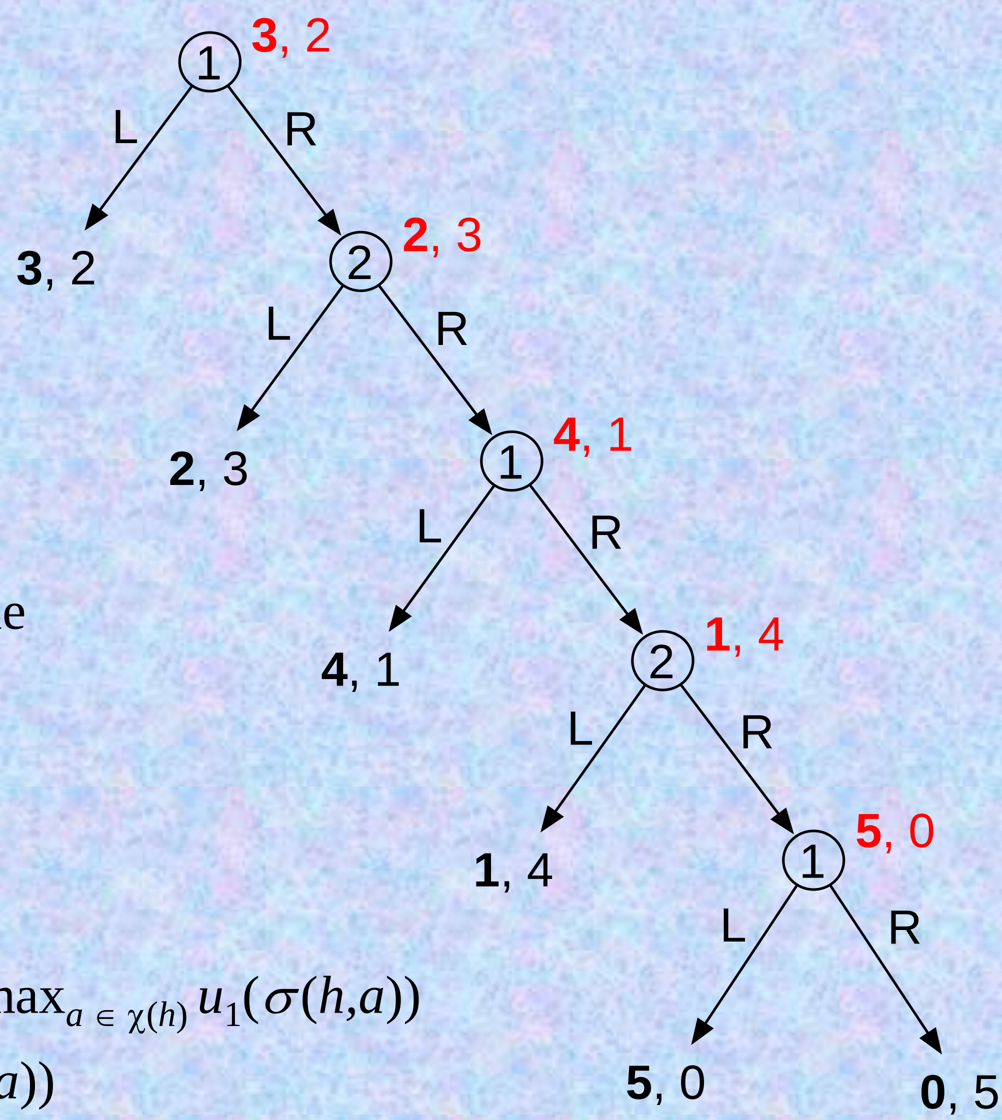
- I need two more volunteers to play a **constant-sum** version of the centipede game
- At every node, $u_2 = 5 - u_1$
- Instead of having increasing payoffs for both players, the sum of their payoffs is always the same
- In this case, backward induction gives much more accurate results



The Minimax Algorithm

- In constant-sum games, only need to compute agent 1's SPE utility, u_1
  - $u_2 = c - u_1$
- From the Minimax Theorem,
  - at each node,

    agent 1's minmax value
    = agent 1's maxmin value
    = agent 1's SPE utility



procedure minimax(h)

  if $h \in Z$ **then return** $u_1(h)$

  else if $\rho(h) = 1$ **then return** $\max_{a \in \chi(h)} u_1(\sigma(h,a))$

  else return $\min_{a \in \chi(h)} u_1(\sigma(h,a))$

Summary

- Extensive-form games
  - relation to normal-form games
  - Nash equilibria
  - subgame-perfect equilibria
  - backward induction
    - The Centipede Game
  - backward induction in constant-sum games
    - minimax and negamax algorithms

- In extensive-form games, the game tree is often too big to search completely
  - E.g., game tree for chess: about 10^{150} nodes