Game Playing AI

Dr. Baldassano
chrisb@princeton.edu
Yu's Elite Education

Last 2 weeks recap: Graphs

- Graphs represent pairwise relationships
- Directed/undirected, weighted/unweights
- Common algorithms:
  - Shortest path
  - Importance/centrality (pagerank)
  - Strongly connected components
  - Spanning tree

Homework: Superbull

- http://usaco.org/index.php?page=viewproblem2&cpid=531

Planning with an adversary

- We’ve talked about using graphs for planning
  - Find best plan to goal state using shortest path
- But often we aren’t the only ones trying to accomplish a goal!
  - Playing games
  - Sharing resources, e.g. internet congestion
- Game playing is a type of *adversarial search*

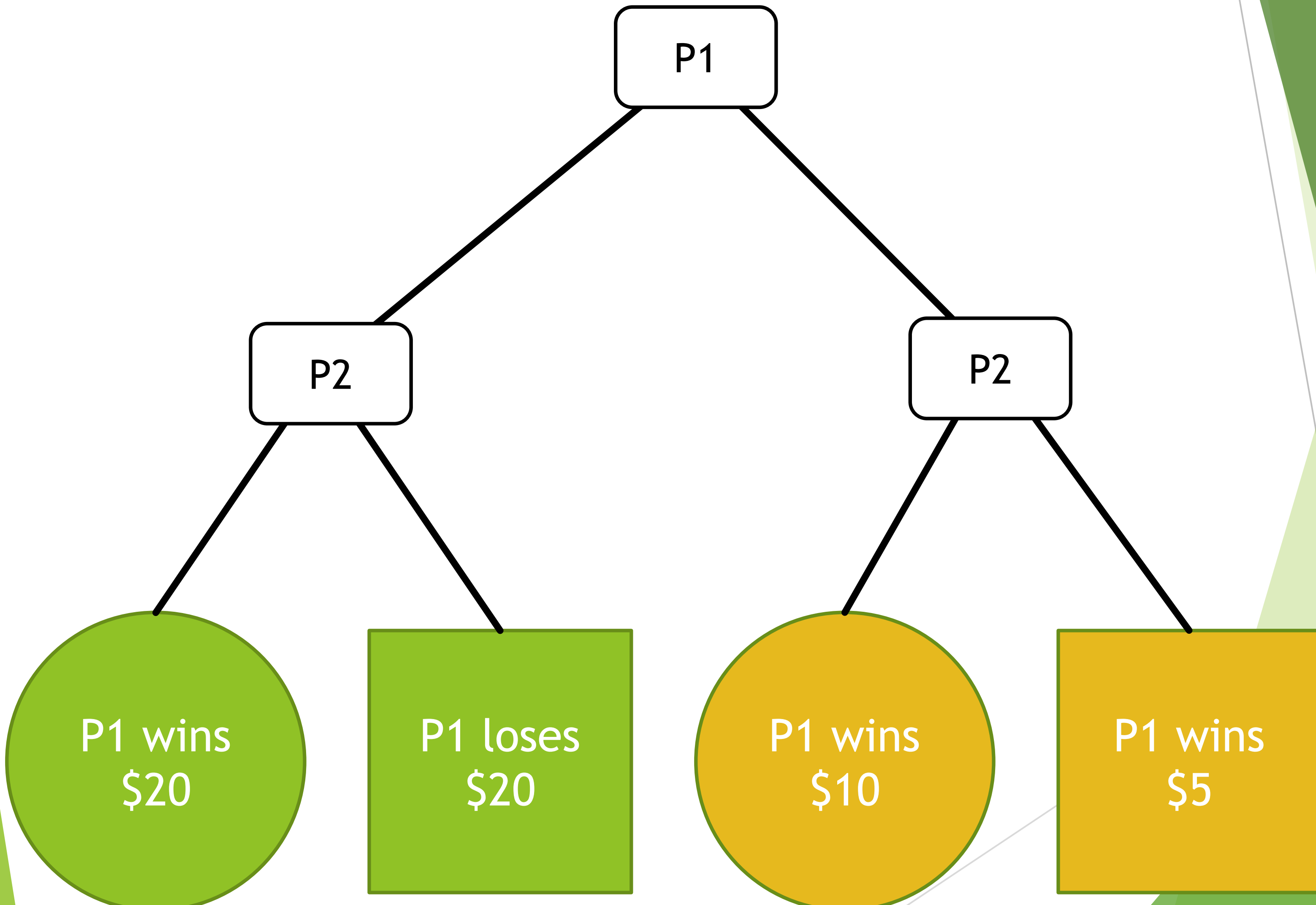
A simple game

- Player 1 picks the color
- Player 2 picks the shape

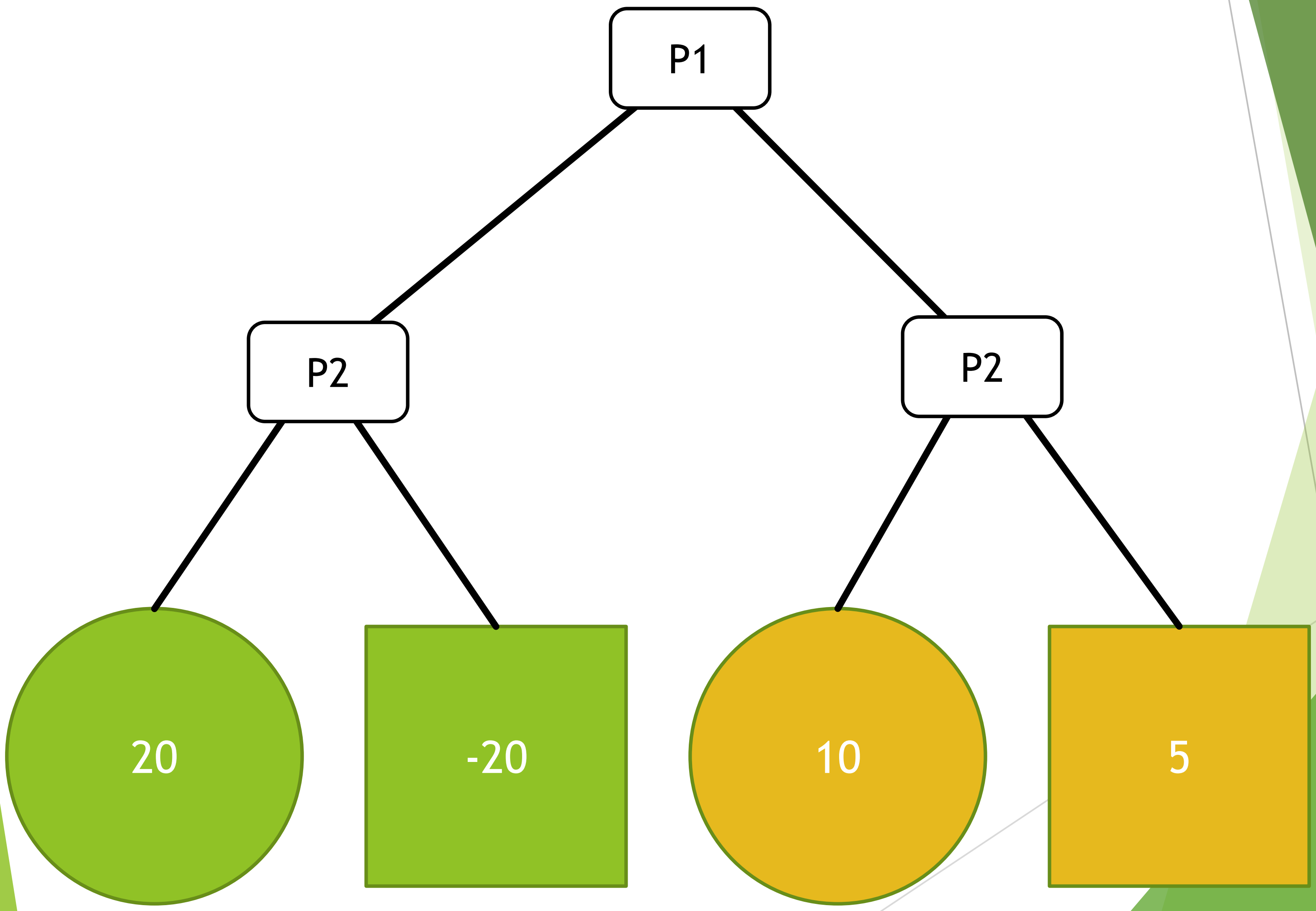
Adversarial search

- We might not be able to get to the best outcome anymore
- Assume that other player will act optimally, and make the more that will allow them to cause the least damage to us

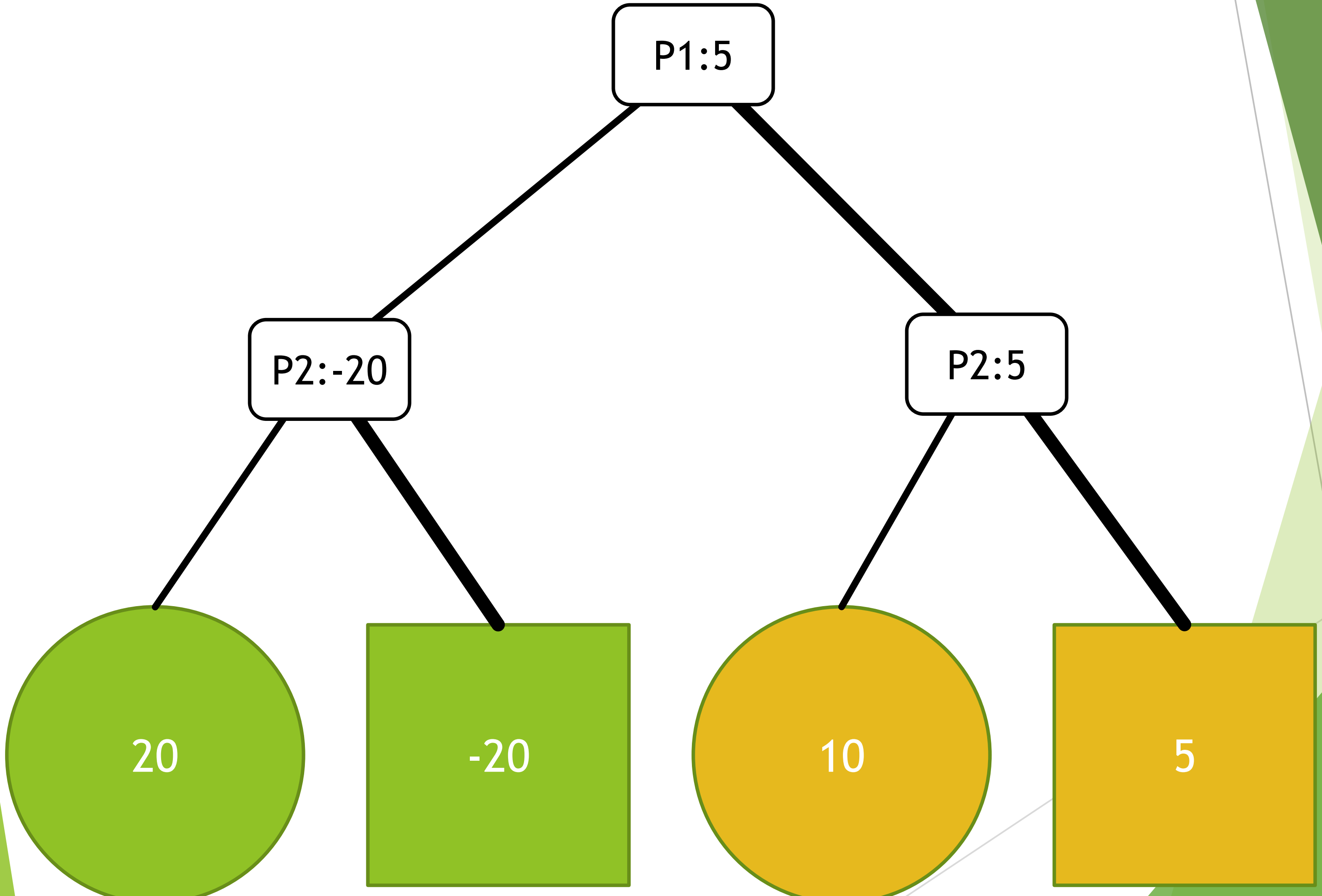
Our game as a tree



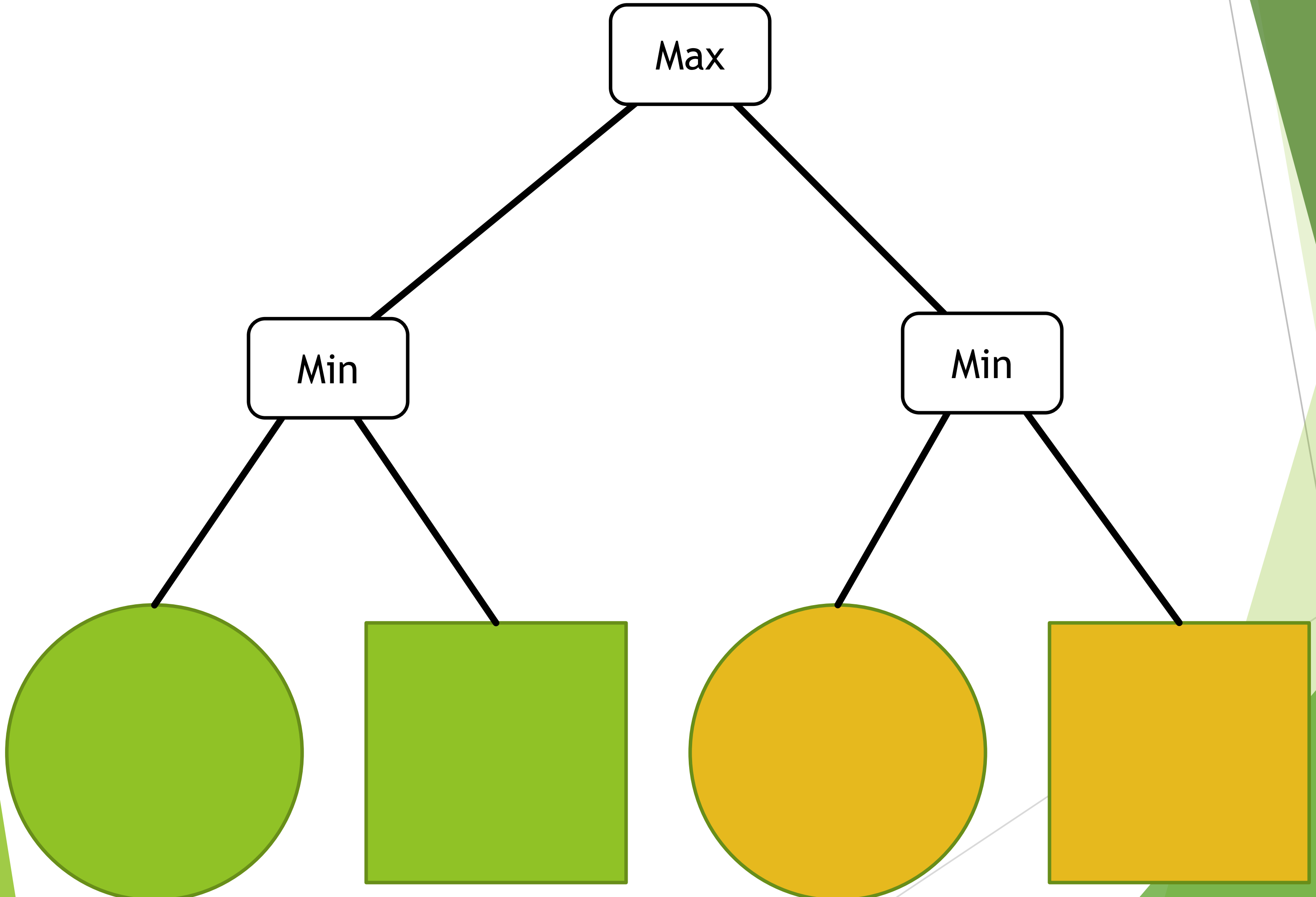
Our game as a tree



Our game as a tree



Our game as a tree



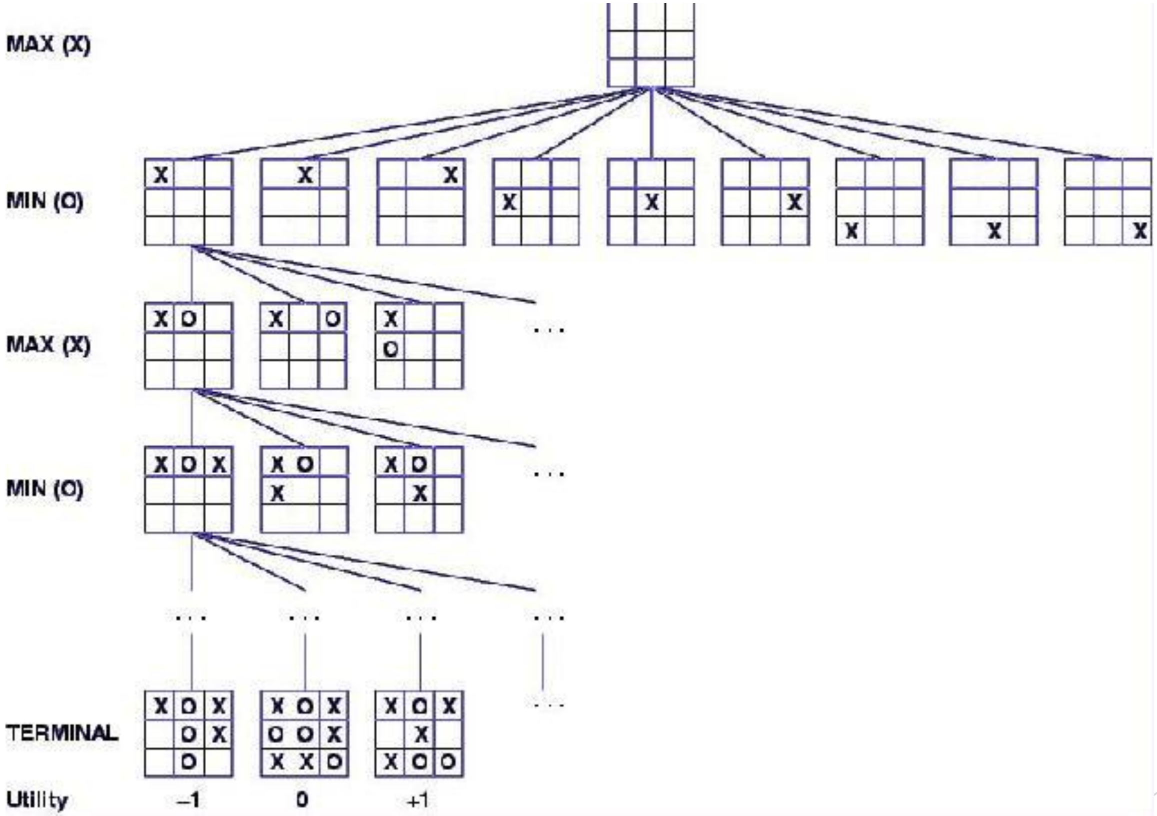
Minimax

- https://www.youtube.com/watch?v=zDskcx8FStA

Minimax search

```
NodeValue(node, depth)
    if node is a leaf
        return node.value
    if depth % 2 == 0
        return max(NodeValue(child1),NodeValue(child2))
    else
        return min(NodeValue(child1),NodeValue(child2))
```

Minimax for Tic-Tac-Toe



Problems with pure minimax

- Minimax guarantees that we'll choose the best move
- Assuming other player acts optimally, we can't lose any (fair) game!
- BUT number of possible states may explode
  - Chess has ~35 moves, ~40 move game – 10^{62} states
- How to cut down on the number of states we need to explore?

Pruning

Alpha-beta pruning

- Can keep track at each node of a lower (alpha) and upper (beta) limit on when this node will be useful
- If beta > alpha, skip the rest of this node
- http://homepage.ufp.pt/jtorres/ensino/ia/alfabeta.html

Equivalent states

- There may be more than one way to get to a particular game state
- Also, many games have symmetric states
  - Example: board rotations in Tic-Tac-Toe
- How to detect if we’ve already evaluated a state?
- Use a hash function, check for collisions

Approximate methods

- Even with alpha-beta pruning and equivalent states, state space is still way too big for DFS for anything more complicated than checkers
- Now we'll talk about approximate methods – no longer guarantee right answer

Evaluation functions

- Instead of DFSing all the way to goal states, use a stopping depth
- If we've searched D levels and haven't hit an end state, use a heuristic (evaluation function) to guess state value
- This is how humans play chess – we just plan a few moves ahead, to states that seem good
  - Experts have a deeper stopping depth and a better evaluation function than beginners

Picking a heuristic

- Option 1: Use your knowledge about the game
  - Chess: pieces remaining, square control, mobility, pawn structure…
- Option 2: Use supervised machine learning
  - Use a database of previous games to see which positions tended to lead to victory/defeat
  - Have game play itself and learn over time

Approximate pruning

- Use another heuristic function to pick which moves to try
- Again, can be hand-coded or learned

Time limits

- Often there is a time limit for us to make a move (in the game, or based on the human's patience)
- *Iterative deepening*: Set stopping depth to 1, then 2, then 3, until we run out of time

Current state of the art: Connect Four

- “Strongly solved” in 1995 by John Tromp
  - “Strongly solved” – unbeatable regardless of opponent’s actions

Current state of the art: Checkers

- Chinook became world champion in 1994
- "Weakly solved" in 2007 after 18 years of computation
  - "Weakly solved" – assumes perfect opponent, may draw rather than win if opponent makes a risky move

Current state of the art: Chess

- "Deep Blue" defeated Kasparov in 1997
- "Deep Fritz" defeated Kramnik (World Chess Champion) in 2006 (2 wins, 4 draws), last major matchup
- Now even chess programs on mobile phones play at grandmaster level

Current state of the art: Othello

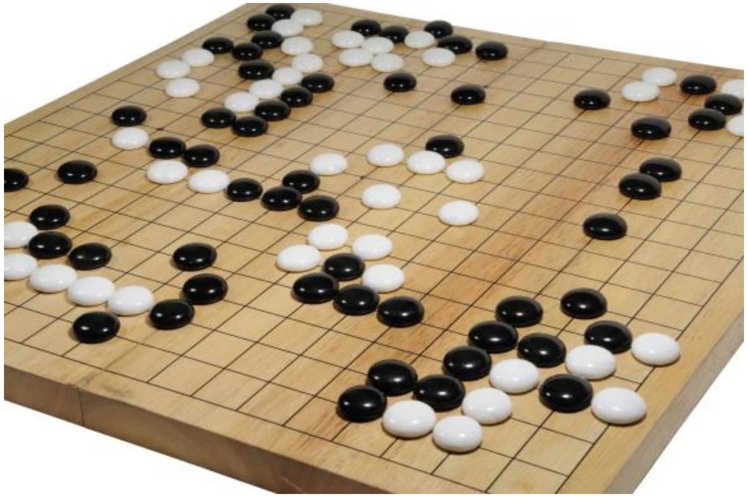
- Programs generally much better than humans
- Relatively small search space, hard for humans to evaluate positions

Current state of the art: Backgammon

- Current machine learning models rank equal to humans
- Requires incorporating chance elements and large search space

Current state of the art: Go

- Humans far better than computers!
- ~360 possible moves per position
- Computers ok in last 10 years, better than amateurs but not competitive with experts



Homework: 2-move TTT

- Modify Tic-Tac-Toe program such that each player takes two turns at a time
- How do we change the minimax procedure? Does the game still end in a draw?