Game Trees

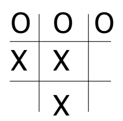
Introduction to
Artificial Intelligence
COS302
Michael L. Littman
Fall 2001

Administration

- Questions?
- Anything on Rush hour?
- HW reminder: Opportunity for feedback. Not seeking perfection!

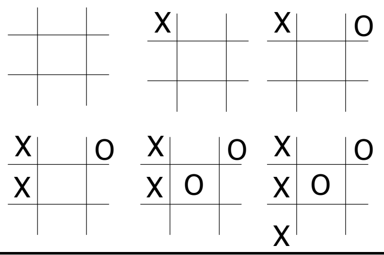
Search in Games

- Consider tic-tac-toe as a search problem.
- States, neighbors, goal?



Problem

- Path to goal isn't quite right.



Game Model

- X: states for player 1 (max) to go
- Y: states for player 2 (min) to go
- N(s): Set of states neighboring s
- G(s): 0, game continues; 1, game ends
- V(s): score for ending in state s
- (Assume alternation: simpler.)

Nim: Example Game

- n piles of sticks, c_i sticks in pile i
- X: sizes of piles on player 1's turn
- Y: sizes of piles on player 2's turn
- N(s): all reductions of a pile by one or more, swapping turns
- G(s): all sticks gone
- V(s): +1 if s in X, else -1 (lose if take last stick)

II-Nim

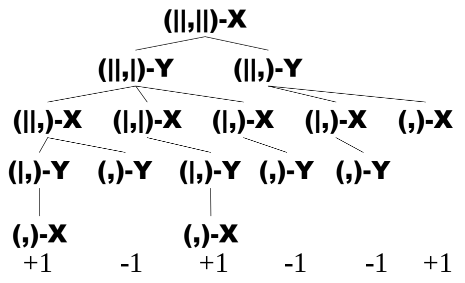
Specific game, starts with two piles, two sticks each.

(||,||)-X, (|,||)-X, (,||)-X, (||,|)-X, (|,|)-X, (,|)-X, (|,)-X, (||,)-X, (,)-X

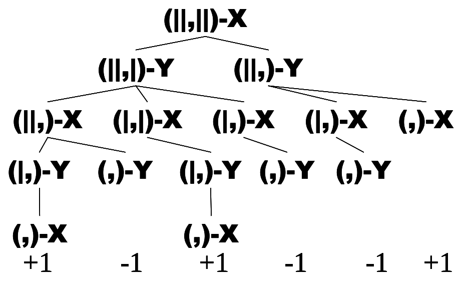
(||,||)-Y, (|,||)-Y, (,||)-Y, (||,|)-Y, (|,|)-Y, (,|)-Y, (|,)-Y, (||,)-Y, (,)-Y

Simplification to reduce states?

Game Tree for II-Nim



Minimax Values



DFS Version

```
Minimax-val(s) = {
If (G(s)) return V(s)
Else if s in X
  return $\max_{s' \text{ in } N(s)}$ minimax-val(s')
Else
  return $\min_{s' \text{ in } N(s)}$ minimax-val(s')
}
```

Questions

- Does BFS minimax make sense?
- If there are loops in the game
  - Will minimax-val always succeed?
  - Will minimax-val always fail?
  - Is minimax even well defined?
  - Can we fix things?

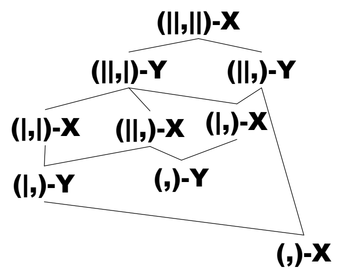
Dynamic Programming

Depth l, branching factor b, minimax-val takes $\Theta(b^l)$ always.

Might be far fewer states: chess $b^l = 10^{120}$, only 10^{40} states

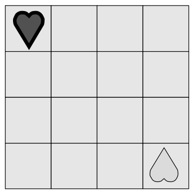
What do you do if far fewer states than tree nodes?

Record Visited States



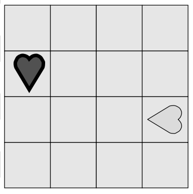
Sharks

Each turn move forward, rotate.
Lose if trapped or “face off”ed.

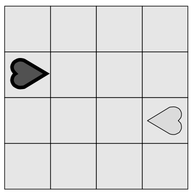


Label States w/ Winner

Win for red



Win for green



Loopy Algorithm

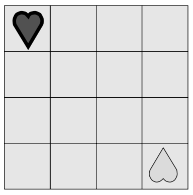
Init: Label all states as “?”. L(s) = “?”
For all x in X:
- If some y in N(x), L(y) = +1, L(x) = +1
- If all y in N(x), L(y) = -1, L(x) = -1

For all y in Y:
- If some x in N(y), L(x) = -1, L(y) = -1
- If all x in N(y), L(x) = +1, L(y) = +1

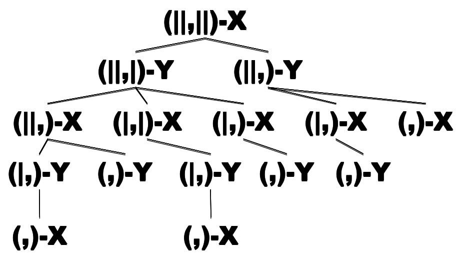
Repeat until no change.

Loopy Analysis

If L(s) = +1, forced win for X
If L(s) = -1, forced win for Y
What if L(s) = “?” at the end?
Neither player can force a win: stalemate.



Pruning the Search

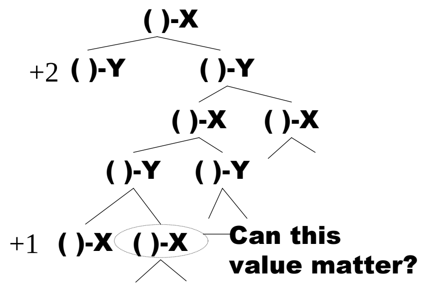


Unknown Terminal Vals

This example made heavy use of the fact that –1 and +1 were the only legal values of terminal states.

What sort of pruning can we do without this knowledge?

Use Ancestors to Prune



Alpha-beta

Max-val(s,alpha,beta)

s: current state (max player to go)

alpha: best score (highest) for max along path to s

beta: best score (lowest) for min along path to s

Output: min(beta, best score for max available from s)

Max-val

```
Max-val(s, alpha, beta) =
• If (s in G(s)), return V(s).
• Else for each s' in N(s)
    – alpha = max(alpha, Min-val(s',
                       alpha,beta))
    – If alpha >= beta, return beta
• Return alpha
```

Min-val

```
Min-val(s, alpha, beta) =
• If (s in G(s)), return V(s).
• Else for each s' in N(s)
    – beta = min(beta, Max-val(s',
                       alpha,beta))
    – If alpha >= beta, return alpha
• Return beta
```

Scaling Up

Best case, alpha-beta cuts branching factor in half.

Doesn't scale to full games like chess without some approximation.

Heuristic Evaluation

Familiar idea: Compute h(s), a value meant to correlate with the game-theoretic value of the game.

After searching as deeply as possible, plug in h(s) instead of searching to leaves.

Building an Evaluator

Compute numeric features of the state: f1,…,fn.

Take a weighted sum according to a set of predefined weights w1,…,wn.

Features chosen by hand. Weights, too, although they can be tuned automatically.

Some Issues

Use h(s) only if s “quiescent”.

Horizon problem: Delay a bad outcome past search depth.

Chess and checkers, pieces leave the board permanently. How can we exploit this?

Openings can be handled also.

Some Games

Chess: Deep Blue beat Kasparov

Checkers: Chinook beat Tinsley (opening book, end game DB)

Othello, Scrabble: near perfect

Go: Branching factor thwarts computers

What to Learn

Game definition (X, Y, N, G, V).

Minimax value and the DFS algorithm for computing it.

Advantages of dynamic programming.

How alpha-beta works.

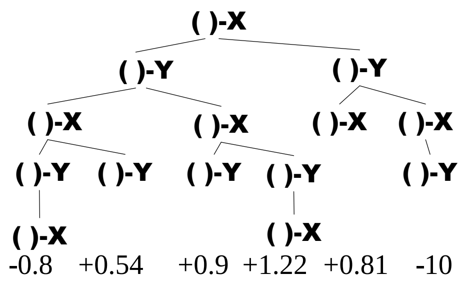
Homework 4 (due 10/17)

1. In graph partitioning, call a *balanced* state one in which both sets contain the same number of nodes. (a) How can we choose an initial state at random that is balanced? (b) How can we define the neighbor set N so that every neighbor of a balanced state is balanced? (c) Show that standard GA crossover (uniform) between two balanced states need not be balanced. (d) Suggest an alternative crossover operator that preserves balance.

HW (continued)

2. Label the nodes of the game tree on the next page with the values assigned by alpha-beta. Assume children are visited left to right.

Problem 2


( )-X
( )-Y
( )-Y
( )-X
( )-X
( )-X
( )-X
( )-Y
( )-Y
( )-Y
( )-Y
( )-Y
( )-X
( )-X
-0.8
+0.54
+0.9
+1.22
+0.81
-10