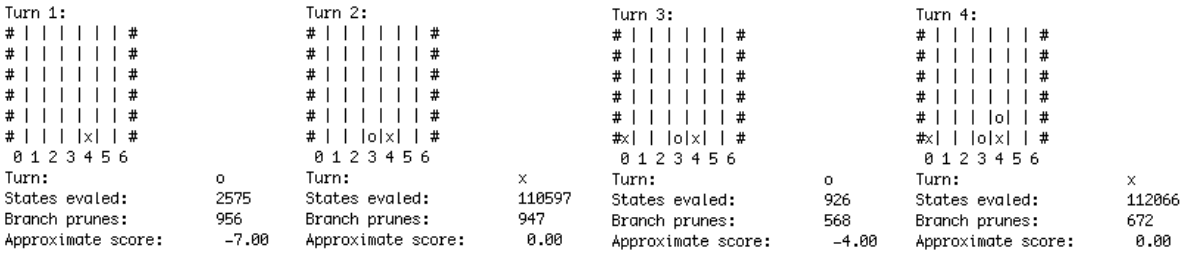
CS188: Artificial Intelligence, Fall 2009
Connect-4 and Evaluation Functions

Connect-4, originally called "The Captain's Mistress", is a two-player game where pieces are dropped onto a grid from above; the goal is to make 4 in a row, vertically, horizontally, or diagonally. With the standard 6x7 board, the player to move first can force a win (the first move is in the center column), a result proved by Victor Allis as part of his Master's Thesis in 1988.



The first few moves of a Connect-4 game. What can you infer about the search methods employed by the two agents?

(1) Consider a minimax search agent playing standard Connect-4.

(a) What is the branching factor?

7

(b) What is the size of the state space?:

Rough upper bound: 3^{42} since each square is 'x', 'o', or empty.

(c) At 3-ply search (a ply includes one move for each player), will alpha-beta prune any branches when choosing the game's first move?

Only if there's a non-zero evaluation function to score non-terminal game states. Otherwise, the game tree will not include any wins or losses (at least 4 moves are required), so all states will have equal value.

(d) In the game shown above, why does 'x' move on the far left? Does he think that is his best option?

It looks like the 'x' agent is not using much of an evaluation function. Most likely, all of his moves look equally good after searching to his maximum depth, so he chooses randomly.

(2) Consider 3 possible evaluation functions. Describe how to implement each one based on the hint.

(a) The code for a simple function is shown here. What does it do? What is the state.filled business?

```
def simpleEvalFunction(state, symbol):
  if state.winner == None: return 0
  if state.winner == 'TIE': return 0
  elif state.winner == symbol:
    return 1 + 1.0/state.filled
  elif state.winner != symbol:
    return -1 + -1.0/state.filled
```

Every state that's not a win or a loss, it returns 0. If a state is winning it returns 1, losing returns -1. Adds a constant that depends on the number of filled squares on the board so that when it sees an eventual loss, it doesn't commit suicide instantly; likewise, takes the fastest possible win.

(b) Central moves are generally better than edge moves.
See middleEvalFunction in connect_eval_functions.py

(c) It's good to have 3-in-a-rows. 2-in-a-rows are also good.
See betterEvalFunction in connect_eval_functions.py