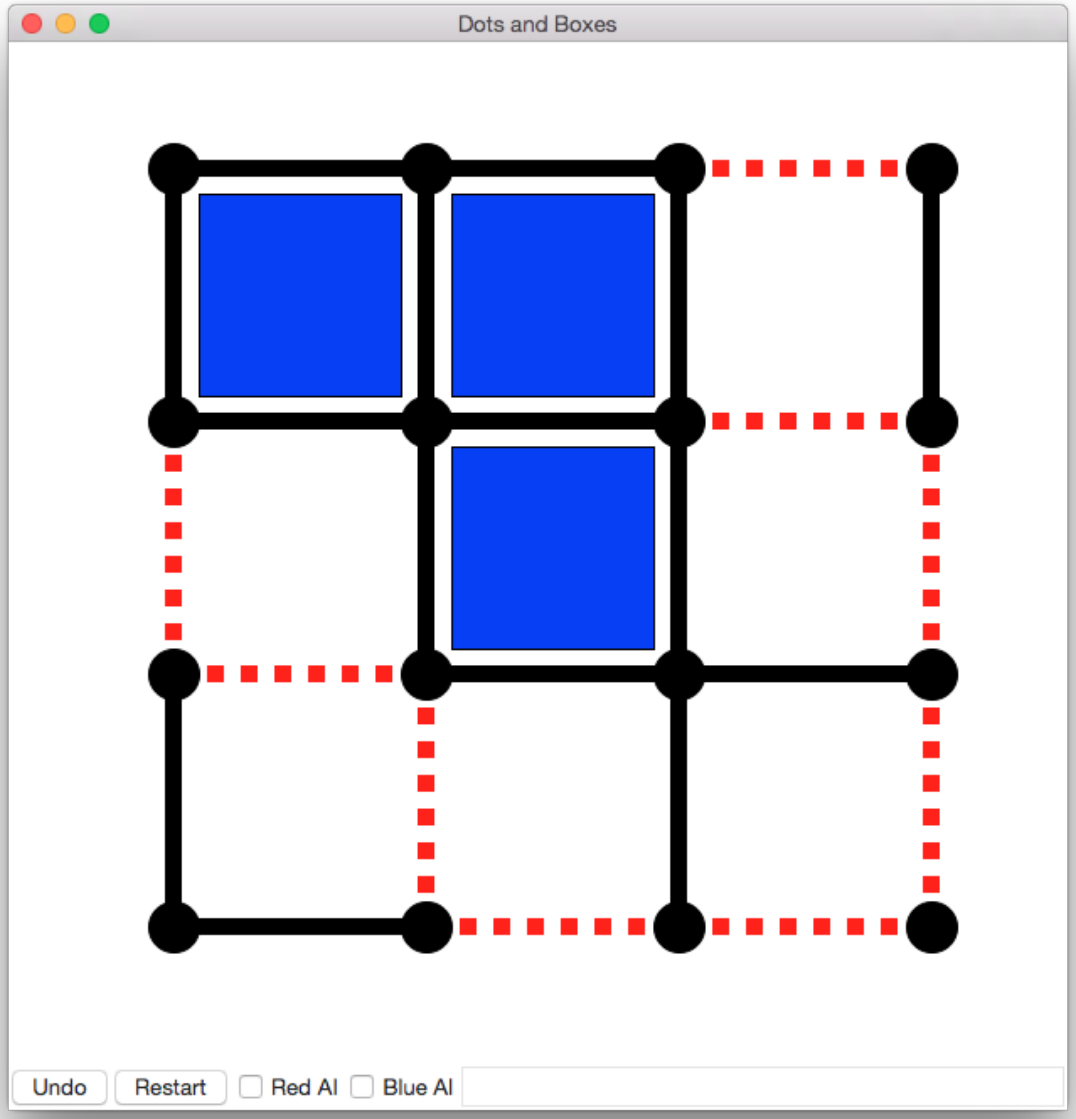
P2: Dots and Boxes Bots



Objectives

- Understand how fixed programs can play arbitrary games by exploring a game tree.
- Understand how to explore a game tree incrementally.
- Understand the UCT algorithm.
- Familiarize yourself with running graphical Python programs.

Requirements

- Implement uniform_bot.py that chooses amongst legal moves uniformly (use the choice function in Python's random library)
- Implement greedy_bot.py that chooses a legal move that maximizes the immediate score gain (it looks 1 move into the future and chooses the one that increases its score the most)
- Implement uct_bot.py that uses MCTS with full rollout. The reward function should be the current player's score minus the other player's score.
- Implement fast_bot.py that uses MCTS to a rollout depth of at most 5.
- Use python's time module to make sure that your bots do not think about their move for more than a second.
- For uct_bot.py and fast_bot.py print out the number of rollouts per second at the end of each move.

Grading Criteria

(equal weight for each question)

- Is uniform_bot.py implemented correctly?
- Is greedy_bot.py implemented correctly?
- Is uct_bot.py implemented correctly?
- Is fast_bot.py implemented correctly?
- Are rollout rates for the UCT bots visible somewhere?

Resources

- Use the Python code for UCT available on http://mcts.ai/ as a reference, but don't just copy-paste what you find. It will take some adaptation to work with our state space and simulator.
- Browse Cameron Browne's MCTS lecture slides: http://ccg.doc.gold.ac.uk/teaching/ludic_computing/ludic16.pdf

Base Code

- https://drive.google.com/a/ucsc.edu/folderview?id=0B-PPiU3Ga8Z7fjlwNVJIb0pTYnp3WEF1YlhiT0Q0MnFwZXZfX0F3LVBUZUpSa29wUWxMcnc&usp=sharing
- Commands:
    - $ python p2_gui.py
        - An interactive, graphical version of the game. Bots are disabled by default, but you can turn them on with checkboxes and then restart the game.
    - $ python p2_sim.py
        - A high-speed command-line simulator useful for running repeated rounds between pairs of bots.
- To change which bots are active in either the graphical or tournament versions of the game, edit the lines that look like this:
    - import first_bot as red_bot
- Here's how you'd have the greedy bot play the blue player

    - import greedy_bot as blue bot
- In your bot modules, implement a function called “think” that takes two arguments:
    - state:
        - The current state of the game (an instance of State from p2_game.py). Your code should not depend on the internals of the state object at all. Here is the allowed interface:
            - state.get_whos_turn() → ‘red’ or ‘blue’
            - state.get_score() → dict of scores for each player
            - state.get_moves() → a list of legal moves
            - state.apply_move(move) → updates state to apply a move, assuming it was legal
            - state.copy() → produces a copy of this state that can be mutated (with apply_move) without changing the original
    - quip(string):
        - A function you can call with a string to show text in the GUI or print it in the command line for the high-speed simulator