Tak Game Bot

Project for CS 4180 - Fall 2021

By: Pablo Kvitca

Problem

- 2 player board game
- Full-knowledge (state and transitions are fully observable)
- Reward is the score at the end of the game
- Types of pieces: Flat stones, Capstone, Standing Stones
- Goal:
  - Connect a path of stones (of your color) from one edge to its opposite side
    - Either left<->right or top<->bottom
    - Stones connect to orthogonal directions (UP, RIGHT, DOWN, LEFT)
    - Only “flat stones” and “capstones” count for paths
- Other end conditions:
  - No empty spaces
  - No stones to place
- Starting condition: empty board (different sizes: 3x3, 4x4, 5x5, ... 8x8)
- 1st move is placing the first (flat) stone of the opponent’s color
- Following moves can:
  - Place a stone on an empty position
  - Pick up a stack (pieces, max=*board size*) and move (dropping of at least one) in one direction
    - Possibly flatten a standing stone

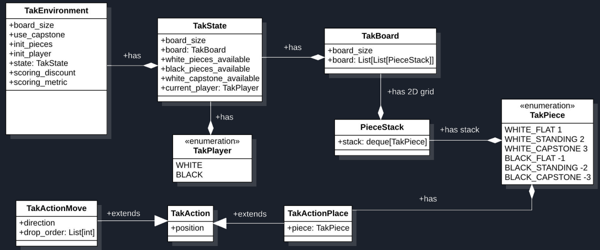
Upper bound of # actions for a player:

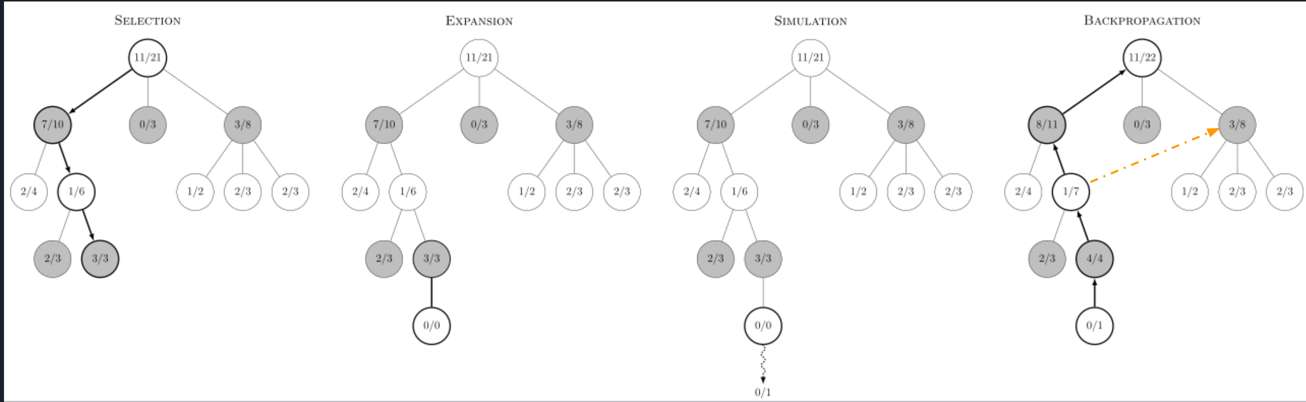
$$3b^2 + 4b^2 \sum_{i=1}^{5} count(ordpartitions(i)) = 3b^2 + 4b^2 \sum_{i=1}^{5} 2^{i-1}$$

Learn more at: ustak.org
Created by: James Ernest and Patrick Rothfuss

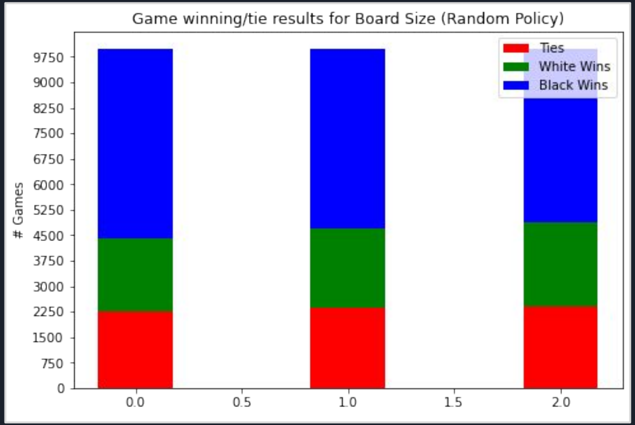
Approach - Monte-Carlo Tree Search

1. Implemented the environment, following OpenAI Gym's interface
2. Implemented Random Policy (selects valid actions randomly)
3. Implemented Random Policy with equal probability of place/move type actions
    a. At the start valid place actions are more common than moves, but less effective
4. Implemented Monte-Carlo Tree Search for this environment:
    a. Modified from the version seen in class
    b. The previous states to get to the current state don't matter, so when doing rollouts for one state, we also update the other potential (if known) paths to get to that state.
        i. Not a tree structure in general, but every "game" and expansion sub-graph is a tree
    c. Uses an e-greedy approach to expanding nodes (before rollout), some probability *p* to choose a random action, with *1-p* to choose the best action
    d. This is done both ways for the current player and the "adversarial" player, maximizing and minimizing, respectively





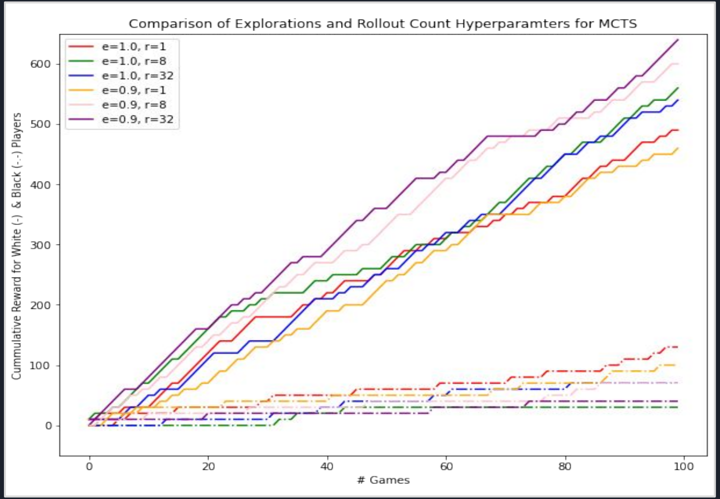
Results - So Far



Playing randomly results mostly in ties

MCTS Player vs Random Player

- Exploration on MCTS "expansión" improves performance (e-greedy, *e*)
- # Rollouts on MCTS: more rollout runs improve performance (*r*)



However, MCTS with random policy rollout ends in ties most of the time (~65%). Seems to learn to get **higher game score**, but **not to win** more often

Next Steps:

- Create a better policy to test on its own,
- and as the MCTS' rollout policy

Questions?