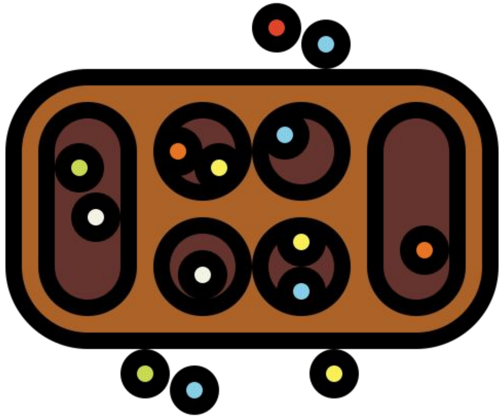
Kalah Game

GROUP 36

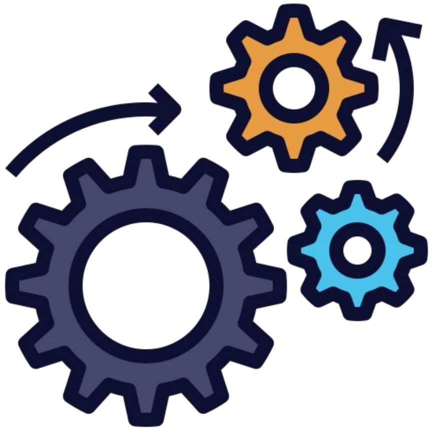
Bushui, Weilue, Yecheng, Zhaoyu

Game and parser



Mancala Game in python

Used in
- Developing strategies
- Evaluation between agents

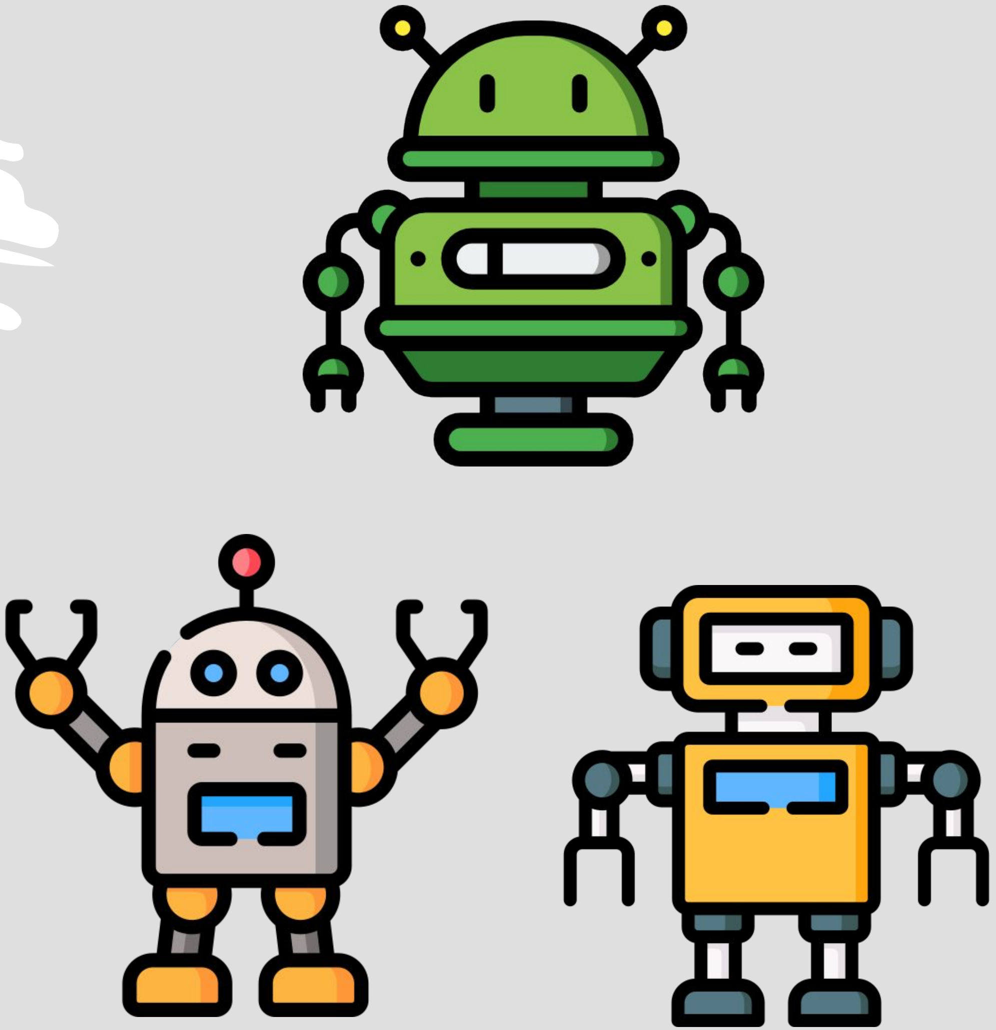


Parser

Protocol info -> information needed
Provide state/board info etc.
Basic communication with protocol

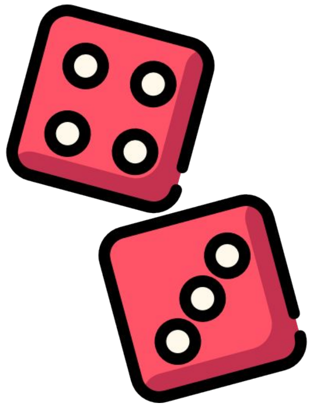
Agents

- Random Agent
- Simple Agent
- Alpha-beta pruning Agent
- MCTS Agent
- Model Agent



Random Agent

- Perform random move from available moves



Simple Agent

Implemented using simple rules:

Go through available moves, do:

- Winning move

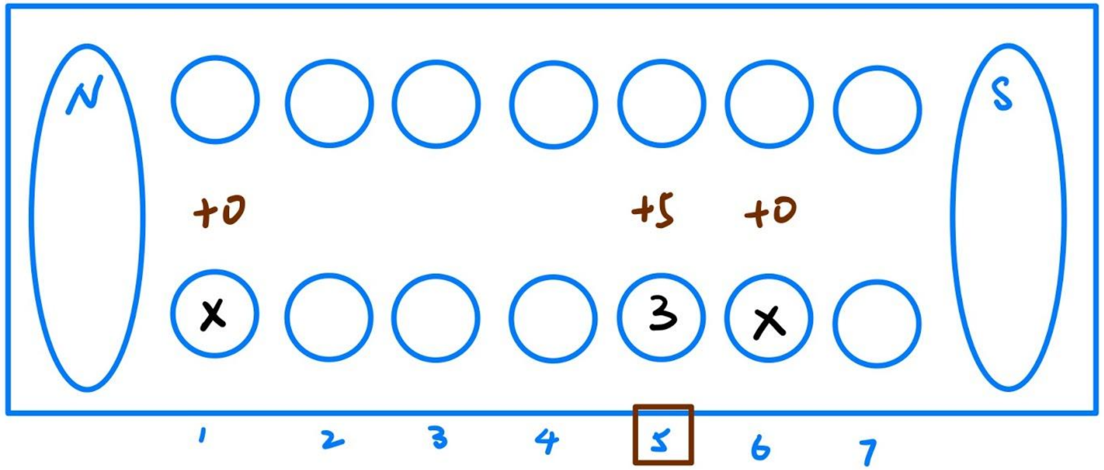
Else, consider:

- Extra move chance
- Maximise self-gain and Minimize opponent-gain

Heuristics

⇒ avaliability of extra move

⇒ score increase



step 1

All avaliable moves for me: +12 +1 +6

step 2

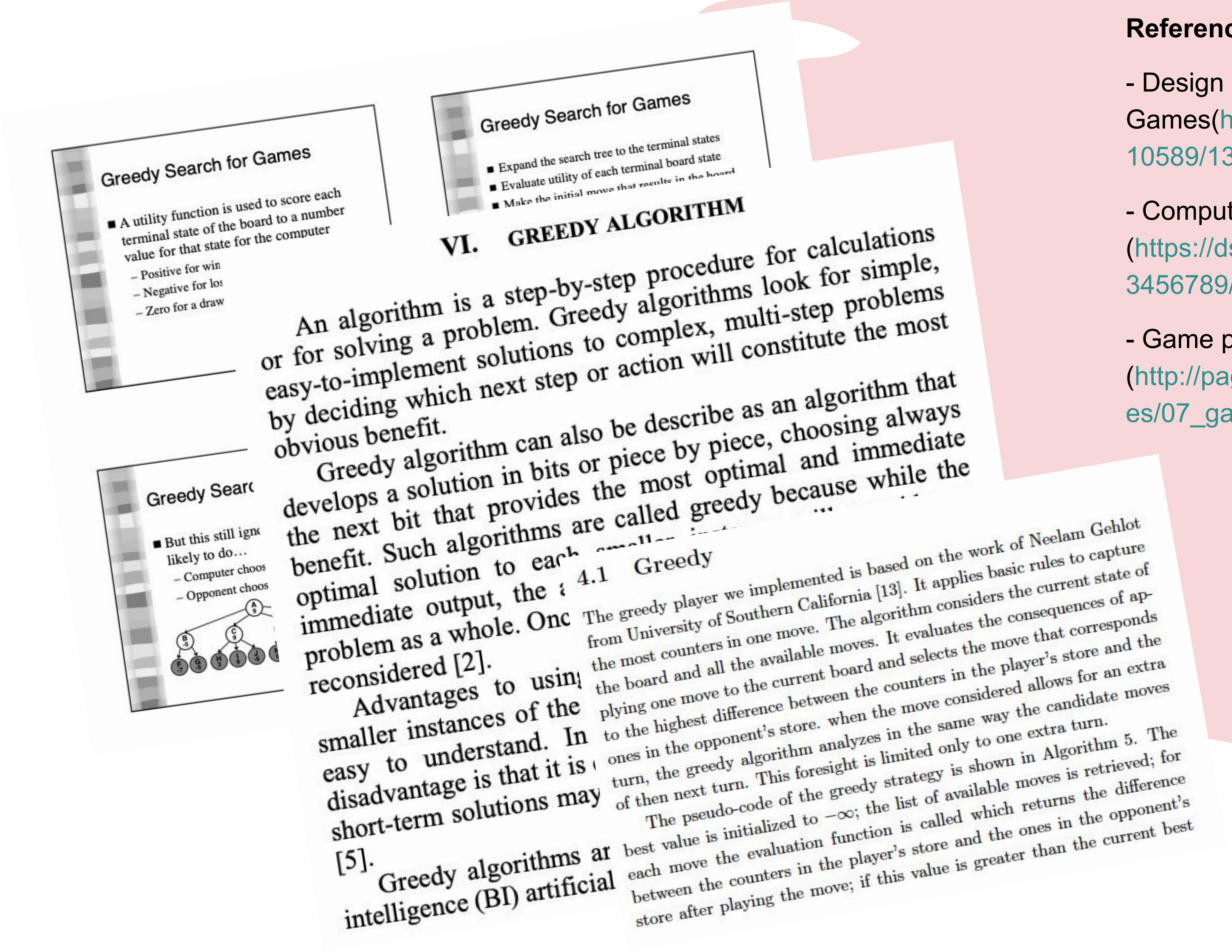
max of all avaliable moves of opponent: max oppo. +15 +5 +1

step 3

suppose no extra moves

return move of max value: 12-15=-3 5+1-5=1 6-1=5

step 4



References:

- Design of Artificial Intelligence for Mancala Games(https://www.politesi.polimi.it/bitstream/10589/134455/3/Thesis.pdf) page 31-32

- Computerized Board Game: Dara (https://dspace.unijos.edu.ng/jspui/bitstream/123456789/799/1/L_10853_11.6951.pdf) page 3

- Game playing (http://pages.cs.wisc.edu/~bsettles/cs540/lectures/07_game_playing.pdf) page 2

Alpha beta pruning agent

Heuristics:

- Scores of our side - scores of opponent side
  - Straightforward and easy to implement
- Refined heuristics
  - A lot of variants bring a lot of uncertainties
  - Time consumed.
  - If time is enough, experiments will be made to find better heuristics.

Reference:

- Design of Artificial Intelligence for Mancala Games(https://www.politesi.polimi.it/bitstream/10589/134455/3/Thesis.pdf) page 34

MCTS Agent

Using Monte Carlo Tree Search to make the move

- Using RAVE(*Rapid Action Value Estimation*), with modified UCB1 formula to calculate the score of each move

$$(1-\beta(n_i,\tilde{n}_i))\frac{w_i}{n_i}+\beta(n_i,\tilde{n}_i)\frac{\tilde{w}_i}{\tilde{n}_i}+c\sqrt{\frac{\ln t}{n_i}}$$

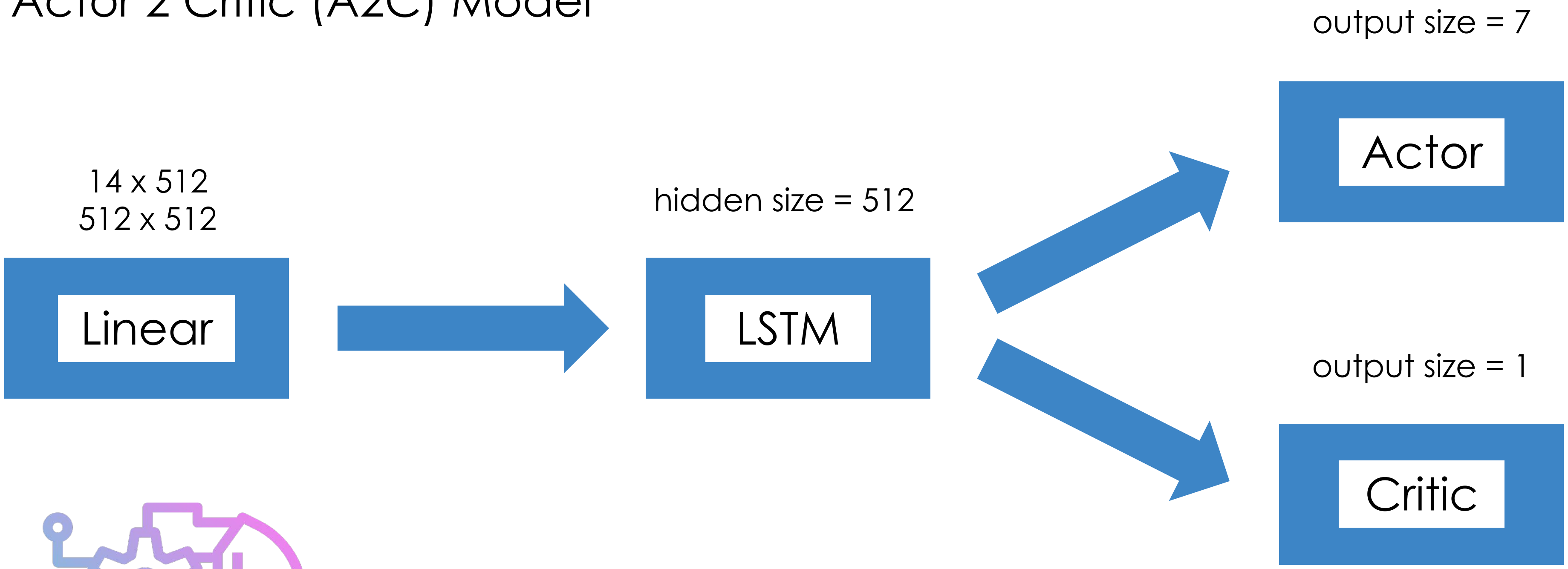
- with alpha-beta pruning agent to forward each move
- return if there exists one child node with score 1 OR the move of the child node with the most visits

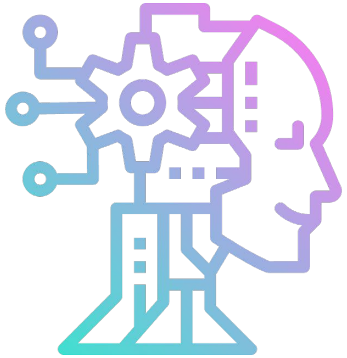
References:

- Design of Artificial Intelligence for Mancala Games(https://www.politesi.polimi.it/bitstream/10589/134455/3/Thesis.pdf) page 37 - 38
- Chang, Hyeong Soo; Fu, Michael C.; Hu, Jiaqiao; Marcus, Steven I. (2005). "An Adaptive Sampling Algorithm for Solving Markov Decision Processes" (PDF). *Operations Research*. **53**: 126–139. doi:10.1287/opre.1040.0145. hdl:1903/6264
- David Silver (2009). *Reinforcement Learning and Simulation-Based Search in Computer Go* (PDF). PhD thesis, University of Alberta.

Model Agent - Architecture

Actor 2 Critic (A2C) Model





https://arxiv.org/abs/1602.01783v2
https://arxiv.org/abs/1909.09586
https://github.com/pytorch/examples/blob/master/reinforcement_learning/actor_critic.py

Model Agent - Loss

Reward Loss

1. Calculate the reward for each move by according to discount factor.
2. Normalize the reward.
3. Multiply each move’s reward by move’s probability.

Value Loss

1. L1 smooth loss of critic output and reward

Total Loss = reward loss + 0.5 * value loss

https://pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html#torch.nn.SmoothL1Loss
https://github.com/pytorch/examples/blob/master/reinforcement_learning/reinforce.py#L62
http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-6.pdf

Evaluation - each agent against each other

```
==========================================
RandomAgent vs AlphaPruningAgent
holes=7, stones=7, games=100
------------------------------------------
RandomAgent :
        wins: 2/100, 2.00%
        avg num moves: 13.77
        avg move time: 0.00000589s
        num first move: 56
        first move wins: 2/56, 3.57%
------------------------------------------
AlphaPruningAgent:
        wins: 98/100, 98.00%
        avg num moves: 19.39
        avg move time: 0.01229027s
        num first move: 44
        first move wins: 44/44, 100.00%
------------------------------------------
draws: 0/100, 0.00%
exceed: 0/100, 0.00%
total time: 0:00:24
==========================================
```

```
==========================================
SimpleAgent vs AlphaPruningAgent
holes=7, stones=7, games=100
------------------------------------------
SimpleAgent :
        wins: 0/100, 0.00%
        avg num moves: 17.92
        avg move time: 0.00222917s
        num first move: 49
        first move wins: 0/49, 0.00%
------------------------------------------
AlphaPruningAgent:
        wins: 100/100, 100.00%
        avg num moves: 26
        avg move time: 0.06855777s
        num first move: 51
        first move wins: 51/51, 100.00%
------------------------------------------
draws: 0/100, 0.00%
exceed: 0/100, 0.00%
total time: 0:03:02
==========================================
```

```
==========================================
MCTSAgent vs AlphaPruningAgent
holes=7, stones=7, games=100
------------------------------------------
MCTSAgent :
        wins: 41/100, 41.00%
        avg num moves: 7
        avg move time: 11.0258996s
        num first move: 52
        first move wins: 41/52, 78.84%
------------------------------------------
AlphaPruningAgent:
        wins: 59/100, 100.00%
        avg num moves: 11
        avg move time: 0.18192353s
        num first move: 48
        first move wins: 48/48, 100.00%
------------------------------------------
draws: 0/100, 0.00%
exceed: 0/100, 0.00%
total time: 1:47:33
==========================================
```

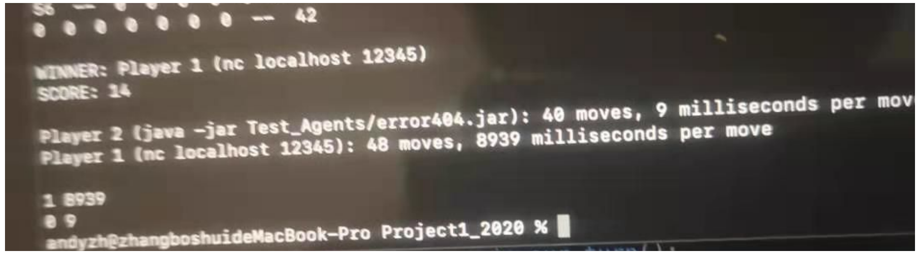
```
==========================================
ModelAgent vs AlphaPruningAgent
holes=7, stones=7, games=100
------------------------------------------
ModelAgent :
        wins: 33/100, 33.00%
        avg num moves: 31.18
        avg move time: 0.001815418s
        num first move: 54
        first move wins: 33/54, 61.11%
------------------------------------------
AlphaPruningAgent:
        wins: 67/100, 100.00%
        avg num moves: 36.1
        avg move time: 0.05134670s
        num first move: 46
        first move wins: 46/46, 100.00%
------------------------------------------
draws: 0/100, 0.00%
exceed: 0/100, 0.00%
total time: 0:03:13
==========================================
```

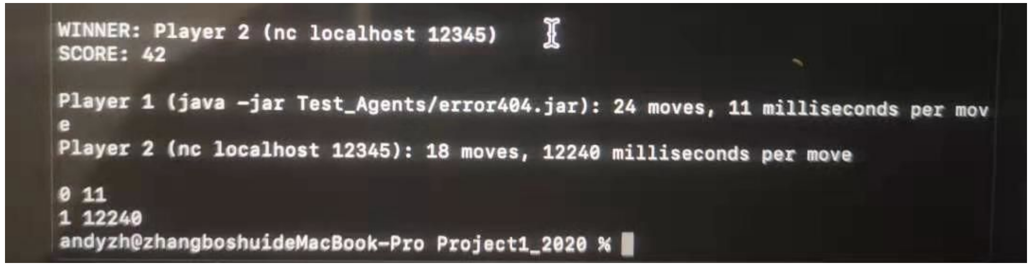
Result shows that alpha beta pruning agent comes to the best

Random < Simple < MCTS < Model < Alpha Beta Pruning

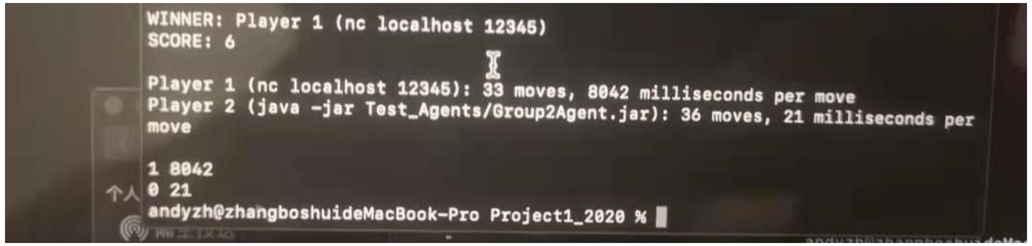
Evaluation - final agent with test agents
- with alpha-beta pruning agent with depth 7

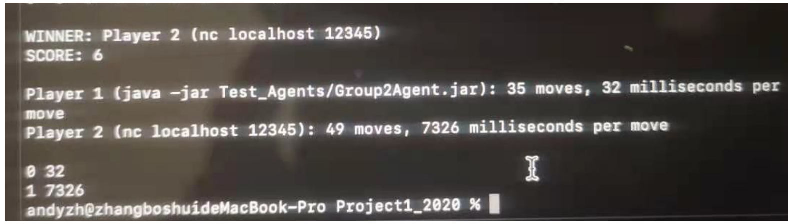
error404.jar



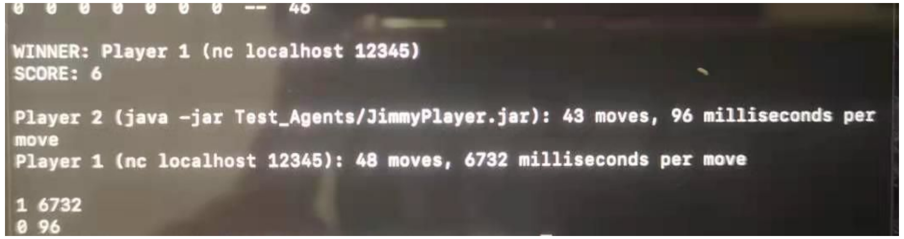


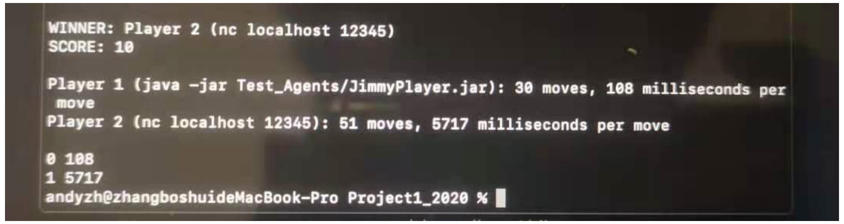
Group2Agent.jar





JimmyPlayer.jar

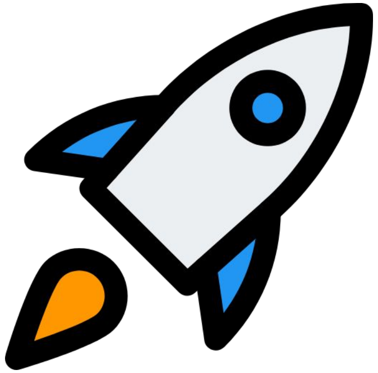




Alpha Beta Pruning Optimization

Python - depth = 8

- More than 1 mins per move



Java without Optimization depth = 10

```
[32, 27, 28, 19, 16, 16, 19, 15, 16, 11,
```

In seconds

Java with Optimization depth = 10 (multi-threading, etc...)

```
[10, 8, 7, 7, 6, 5, 5, 5, 5, 5,
```

In seconds

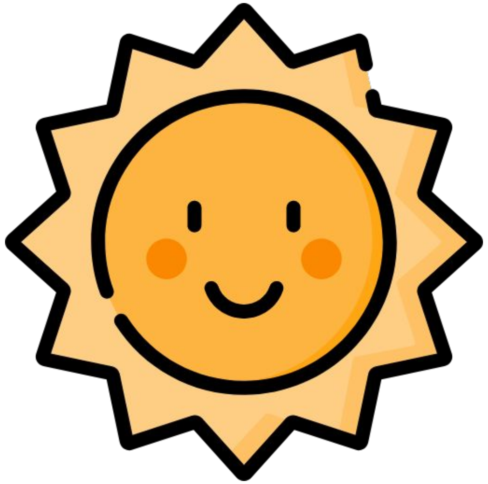
Expected accomplishment

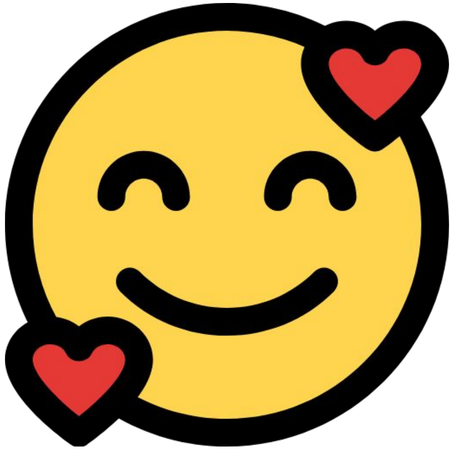
- We will submit the Java Implementation of Alpha Beta Pruning Agent.
- We will have tests to ensure correctness of our code
- Ensure runtime is within time limit
  - For the first several steps, higher searching depth
  - With time elapsing, reduce searching depth

Thank You!

Group 36

- Bushui
- Weilue
- Yecheng
- Zhaoyu





https://github.com/Redcxx/KalahPlayer