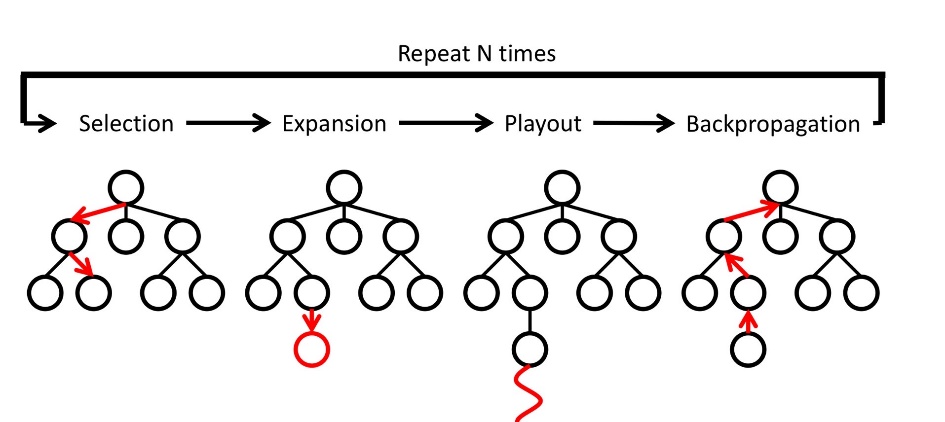
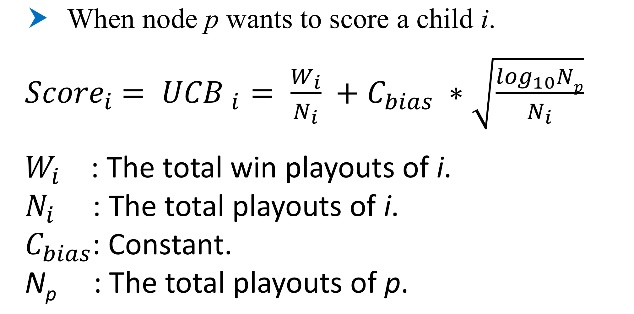
TCG-Project3 report 109550031 李旻融

**Part I: Implementation of MCTS with UCB policy**

Construct a class for nodes, with its current board state, whose turns, last action, # of wins, # of totals, UCB value, a vector of child node, and parent node as node attributes. Next, implement the MCTS in agent.h. It consists of four parts: selection, expansion, simulation, and backpropagation.



1. Selection: use UCB formula as follow and select the child which has max score



1. Expansion: expand all the valid moves
2. Simulation: randomly gets one of valid moves and return the winner
3. Backpropagation: update the nodes’ # of wins and # of totals

Finally, according to nodes’ # of totals, return the best action.

**Part II: Improvement – time management**

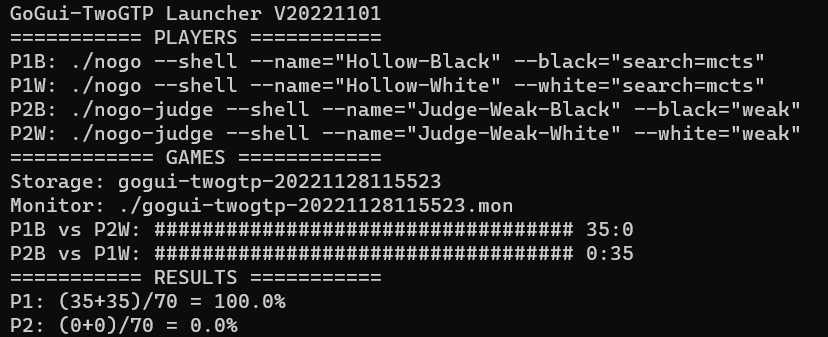
Use a table to manage how much time every step can take. In the early game, it is inefficient to simulate many times; however, in the middle game, the step is more important, so it takes longer time. Moreover, in the endgame, it still takes less time because the number of available actions is few.

**Part III: Performance**

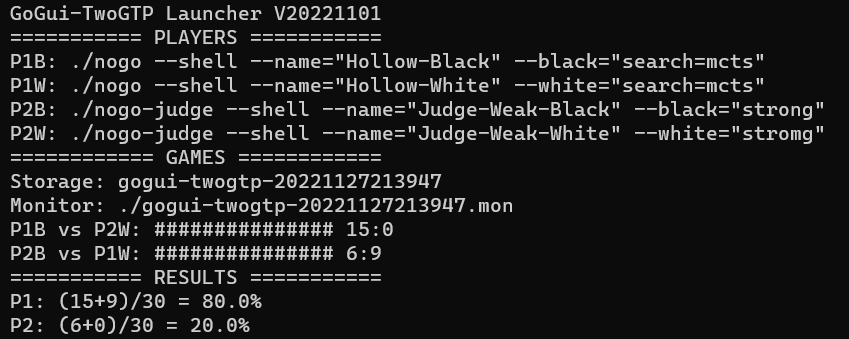
The table is the win rate of basic MCTS and MCTS with time management run 10 games against three levels AI, respectively.

|  |  |  |  |
| --- | --- | --- | --- |
|  | weak | medium | Strong |
| basic MCTS | 90 % | 40 % | 13 % |
| MCTS with time management | 100 % | 80 % | 70 % |

run 70 games against weak AI



run 30 games against strong AI



**Part IV: Difficulty**

In this project, I also implement parallel MCTS; however, performance of parallel MCTS is worse than basic MCTS. I think maybe I make a mistake in function – getbestaction( ), after checking this part, I still can’t improve the performance.