A5 Option 1: Baroque Chess Agents

Team member names: Ziyuan Wang, Ye Jin

Intended personality of the agent: Ruthless and Talk Less

Status of agent: MIniMax and AlphaBeta Complete

Design retrospective:

First step: we read the starter code and the rules of the Baroque Chess. The rules are complex and hard to understand, so we found many documents about it and discussed together to figure out how each piece moves. Second step: we began to write the code for each piece moves. The most difficult piece is the Imitator, because it can imitate other pieces to kill them. The lines of the function of the Imitator moves are more than the lines of the move function of all other pieces. Third step: Ziyuan Wang was responsible for the MiniMax algorithm and Ye Jin worked on the AlphaBeta algorithm. We all wrote the code based on the Pseudocode code on Wikipedia. Fourth step: after we finished the code, we began to debug it by changing the initial condition of the check board. By this way, we could test whether our moves are illegal or not. This was not enough, and we decided to have a competition with our own. An unexpected problem occurred in this process. Some steps were time out. Therefore, we added the time function on our MiniMax and AlphaBeta algorithm to make sure it would not overtime. During this debug process, we also fixed some illegal moves. Finally, our code worked. Our agent began to have a competition with other agents and won most of time.

Partnership retrospective:

Ziyuan Wang:

In this team, I was responsible for parts of the coding work including the Zobrist hashing algorithm and part of the algorithm for the Baroque chess moves. In addition, I tested our code to find whether the illegal moves exist and debug them. During this experience, I found coding together is not simple, because it is hard to read and understand other's code. What we did was expressing our thought while coding, so it is easier to understand what the code means and follow the progress of developing the program.

Ye Jin:

In this team, I created the py file. I found it is very hard to make the logic of imitator. And there are always a lot of bug during running. Finally, I just let my own agent beat my own agent for thousands of turns. And try to fix all the mistakes.