COMP 4106 - Artificial Intelligence
Winter 2020

Assignment #2

Due date: March 9, 2020

Game Playing with MiniMax

Introduction

In this assignment you will be implementing a variation of the game Mancala.

Mancala

The original version of Mancala is played on the board shown in Figure 1, where the respective players' sides are distinguished by red or blue. Each players has a "Mancala" on the board, where he will store all the pieces that he has captured during the game. For an overview of how this game is played, please take a look at the following video: "How To Play Mancala".

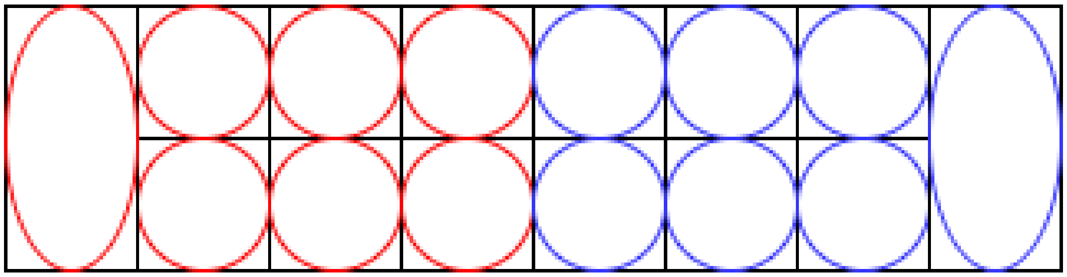

Figure 1: Original Mancala Board

For this assignment, you will be playing a Variation of Mancala using the board shown in Figure 2, where some circles are numbered for explanation purposes (see below). The game is played exactly as in the video, except that in this variation, the players move horizontally and vertically, as described by the arrows.

Rules of the Game

Each colored hole holds N_s stones, and the Mancalas are empty initially. The number of stones, N_s, should be an input parameter, with an upper limit of 6.

The player who goes first is decided randomly. During a turn, the player grabs all of the stones in a hole on his side, and drops them, one by one, in succeeding holes, in either a clockwise or a counter-clockwise direction until there are no more stones in the player's hand. It is then his opponents turn. The player can place a stone in his own Mancala. When the player reaches one of his opponent's Mancalas, the player has the choice between:

- Skip over the opponent's Mancala.
- Place a stone in his opponent's Mancala, but then the player can take at most 2 stones from the opponent's Mancala and place them directly in *his* Mancalas.

When dropping stones in holes, if a player drops a stone into his own Mancala, and it was the last stone in his hand, he gets to play again. Furthermore, if he drops a stone into a hole that was

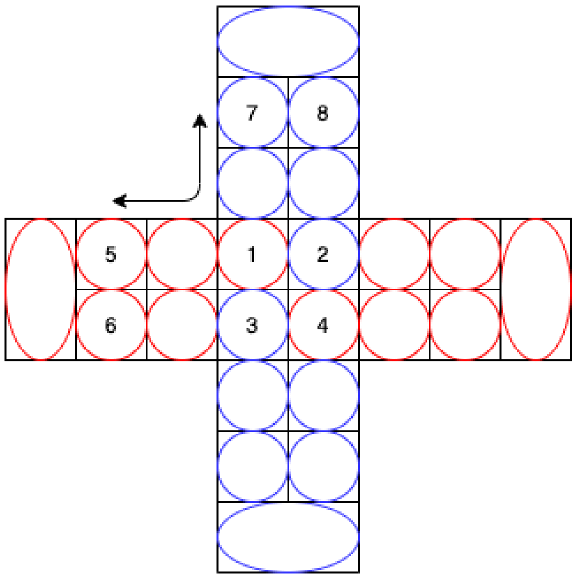

Figure 2: Board for the Variation of Mancala

previously empty and is on his side, *and* that was the last stone in his hand, the player gets to take all of the stones in the hole immediately adjacent horizontal and vertical holes. Thus, for example, if it is hole #1 shown in Figure 2, the player can choose to take all stones in hole #2 or hole #3, and place them into *his* Mancala.

The game is over when either player has no more stones on his side. The opponent then takes all of the stones remaining on his own side and places them in his own Mancala. The winner is the person with the most stones in his Mancalas.

Important: It doesn't matter if you don't understand the rules, as stated above, so exactly. Program the game with the rules as you understand them and it will be acceptable.

Assignment Objectives

- Implement MiniMax search with Alpha-Beta pruning for the Variation of Mancala.
  - Implement two different heuristics for the Variation of Mancala
  - Enable player *vs* computer play of the game, where the computer player uses one of your heuristics. **You are not being marked on the usability of your player interface, so long as it is easily readable.**
  - Enable computer *vs* computer play of the game, where each computer player uses a different heuristic. If the game play between two heuristics is identical, break the symmetry using random move ordering.
- Provide a way to bound the depth of the search.

- Code your assignment in such a way so as to be able to show every move being made in both of the games.
- Provide a way to measure and record the **Node Count** of the computer player's search. The Node Count is the number of nodes visited by the Mini-Max algorithm, excluding those pruned by alpha-beta pruning.
- Write a short report (no more than one (1) page) about the state space of the game, and about the choice of your heuristics, and the **Node Count** you had for the different options. *Please bring a hard copy of this report with your name and student number to your demo.*

Questions

During the demo you should be prepared to discuss the following questions:

- Explain the heuristics you used for each of the games.
- In each of the games, does one player always win?
- How does the Node Count change from Mini-Max without alpha-beta, to Mini-Max with alpha-beta enabled?

Tips

Don't spend too much time on the graphics. A command line representation is fine, so long as it is understandable.

Bonus

Implement Iterative Deepening, using the Principal Variation to achieve move ordering. You may also choose to implement a different move ordering algorithm such as Killer Moves or the History heuristic, but they are more complex. Be prepared to demonstrate how this impacts the Node Count of your search. (10% Bonus)