Henry Jacobs

CS 380 – Artificial Intelligence

Professor Popyack

6/2/2022

Pentago.py Documentation

The Program

I implemented three main functions for this assignment: Player.hmj32_h, Player.monteCarlo, and Player.miniMax. I describe the heuristic function (Player.hmj32_h) in detail in the titled "hmj32 – heuristic documentation". I made one slight change to the heuristic from the point that it was originally submitted on Tuesday May 31st and my submission now (June 2nd). This difference is explained in depth at the end of the heuristic documentation file. **The main program file is Pentago.py. I also included the heuristic in a separate file called hmj32_h.py.**

Player.miniMax Function

The minimax function is straightforward and almost identical to the one given in the lecture slides. It recursively calls itself and returns the maximum value for the current player. It utilizes alpha-beta pruning, which significantly cuts the runtime down. I attempted using a global list to store all the maximum moves so that I could run a Monte Carlo Tree Search on all the moves that returned maximum values. To achieve this, you just must change the last if statement in miniMax to >= and append the max and move after they have been assigned. I didn't actually end up using this list because I was unable to get my Player.monteCarlo function to work properly.

Player.hmj32_h Function

This is the heuristic function used to score a given board for the current player. The heuristic primarily scores players based off how many pieces they have in a row, and awards extra points for moves with four in a row, and awards INFINITY – 1 to boards with 5 in a row. More can be read about this via the documentation file "hmj32 – heuristic documentation".

Player.monteCarlo Function

This function attempted to model the lookahead function given in the Pentago interlude on the course website. It takes a current state and a move to initially apply. Once it applies the move, it plays the game alternating players until one wins or looses. It keeps track of whether the given strategy is good or bad for the current player by keeping track of a master_sum value, which can be used to rank multiple moves for the current player in terms of their strength.

I was unable to get the function to work properly in the program before the assignment was due. I kept running into a ValueError (i.e., something was returning an object of the wrong type). I left the code so that it could be inspected, and possibly awarded partial credit.

Timestamps

I included a file “timestamps.txt” which compares the time it took for the program to complete for a given state with and without alpha-beta pruning. The program performed significantly faster with alpha-beta pruning (in some cases decreasing the time it took by a factor of almost 15!

Sample Output

I also included a file “sample_output.txt” which includes the program running through a game at a specific state listed in the output file. The output contains the heuristic value for the current state and the move made. The heuristic value represents the state that is printed directly above it. If you scroll to the end, you’ll see the winning state has a score of 9999.