ASSIGNMENT 2: QUORIDOR GAMES (Phase 1)

Goal: The goal of this assignment is to learn the adversarial search algorithms (minimax and alpha beta pruning), which arise in sequential, deterministic, adversarial situations.

The Game of Quoridor: In the game of Quoridor two players compete to reach the other side of the board, by either moving (based on the rules of the game) or placing walls to block the other player's moves. Typically the board is 9x9 and there are a maximum of 10 walls that each player can use. To learn the rules of the game visit the Wikipedia page for Quoridor – we will play the 2 player version.

The Game of Quoridor(n,m,k): We will instead play a generalization of the game. In Quoridor(n,m,k) our board will be nxm (length n and width m) and each player will have k walls at their disposal. The players alternate as usual. Nothing else changes in the game.

Algorithm: Implement this game as an instance of minimax search with cutoff and alpha-beta pruning. Learn a rudimentary evaluation function (define features and set weights of features either by hand or by a learning procedure pitting your player against yourself). You are encouraged to gain insight in your player by pitting it against other teams.

What is being provided:

Your assignment packet contains code for the game server in python and starter code for your player in C++ that: (1) interacts with the game server, (2) allows you to manually send moves to the server. At the server end, you are provided a GUI which allows you to visualize the proceedings of the game.

Interaction with Game Server:

The server compilation that you are given would run via the command: python TkBoard.py.
The server requires pythonTk package to be installed and which can be done in Ubuntu as:
sudo apt-get install python-tk

To run the client:
./client <server-ip> <port-no>
eg] ./client 127.0.0.1 12345
Open from two different terminals.

Initialisation:
The server sends initialization information to the clients in the format: player n m k totalTime.
The client that connects first is player 1.

Sending moves to the server:
The format for sending a move is: moveType row column
moveType: 0 for moving the pawn
1 for horizontal wall
2 for vertical wall
NOTE: when you send the row and column for wall, you are specifying the center of the wall. So values for row and column will be in the range [2,9]

A pawn can move only one step at a time and cannot move diagonally. If the opponent's pawn is on the way, your pawn can jump over the opponent's pawn. In such a case, you need to specify the values for the final position. If you specify the position where the opponent's pawn is present, it will be considered as an invalid move.

e.g. If A is at (2,5) and B is at (3,5) and I need to make a move that jumps over B, then I need to send (4,5) to the server and not (3,5). Sending (3, 5) will be considered invalid.

Basically, you need to specify the final position of your pawn.

You are also expected to keep track of opponent's moves and game state on your own.

There is a total time assigned to you that gets decremented when your turn is going on (else you will lose the game). You might not want to exhaust the same.

Code: Your code must compile and run on **machine named 'todi' or any machine with similar configuration present in GCL.** Please supply a compile.sh script for compilation. Also supply a shell script run.sh. Executing the command ./run.sh server port should start your player and start interacting with game server.

What to submit?

1. Submit your code for your game player. **The code should be contained in zip file named in the format <EntryNo>.zip.** If there are two members in your team it should be called <EntryNo1>_<EntryNo2>.zip
2. Submit at-most 1 page writeup (10 pt font) describing your choices and rationale for your player. This is not graded but failure to submit a satisfactory writeup will incur negative penalty of 20% of total score. Your writeup will help us identify any common misconceptions and particularly good ideas for discussion in the class.

Evaluation Criteria

1. In Phase 1 we will test your code against a simple baseline player on Quoridor(9,9,10). Your final score will be scored as "your position in board – opponents position in board" at the time of victory/defeat. Example, if you reached the goal (9) and the opponent could only move forward 2 steps then your total margin of victory is 7. If the situations are reversed it is -7.
2. Extra credit may be awarded to standout performers.
3. The Phase 1 and Phase 2 of the project jointly carry weight of two programming assignments. Phase 1 carries only 25% of this weight and Phase 2 carries 75%.

What is allowed? What is not?

1. You may work in teams of two or by yourself. If you work in a team of two then make sure you mention the team details in the write-up. Our recommendation – this will lead to the much more open ended final project (phase 2); hence best to work with a partner with whom you communicate well. You will be allowed to use the same partner for the final project. Also, you cannot use the partner from previous assignments.

2. It is recommended that you work in C++. You may choose to not use the sample code. As an experiment we also allow the use of python. Using a different programming language makes it difficult to do a fair comparison, so we strongly recommend C++. If you choose to use python we will not be responsible for inefficiencies of the compiler.

3. Your code must be your own. You are not to take guidance from any general purpose AI code or problem specific code meant to solve this or related problem.

4. It is preferable to develop your algorithm using your own efforts. However, we will not stop you from google searching.

5. You must not discuss this assignment with anyone outside the class. **Make sure you mention the names in your write-up in case you discuss with anyone from within the class outside your team.** Please read academic integrity guidelines on the course home page and follow them carefully.

6. You get a zero if your player does not match with the interaction guidelines in this document.

7. We will run plagiarism detection software. Any team found guilty will be awarded a suitable penalty as per IIT rules.