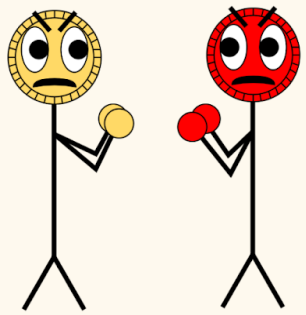
–CONNECT 4–

–REGIONAL CHAMPIONSHIP–



Are you ready for the 2022 Computer Connect-4 Regional Championship Tournament? Will you be the one to write the algorithm to win the title? Enter the competition today!

See the full list of rules and regulations below

Rules and Regulations

The Connect 4 Regional Championship (C4RC) is a team-based computer connect-4 tournament where participants will design an algorithm to efficiently and competitively play the game of connect-4. The following regulations are in place:

- Competitors are free to use any algorithm and evaluation function they like, so long as the computation is done on the local machine (e.g. no online API calls)
- The exception to the above rule is that exactly one move may be hardcoded. We recommend this to be the first move.
- The allowed time per move is limited to 0.5 seconds CPU time - that is, the time the CPU dedicates to running that process. This will be enforced by the driver code and you do not need to time your own algorithm. You should, however, ensure that you set the depth you will explore in accordance with this rule.
- The code for this competition will be written in Python 3.x. Participants are generally free to import any publicly available libraries so long as they do not make the task trivial. If you have any questions about this rule, please reach out to C4RC staff.
- Teams are to submit a report detailing their method and results against provided benchmark agents. This report will be presented to the design committee.

Best of luck to all of our contestants!

Report Guidelines

Along with the code, contestants will submit a report to the design committee to ensure fairness and integrity in the competition. The report will consist of five parts, some of which will be written and others coded. This report will be submitted as a PDF of any format (eg slides, document), and the code submitted should be the players.py file with alphaBetaAI implemented. Teams will present their report to the design committee.

Part I: Evaluation Function

Please explicitly state your evaluation function for terminal nodes. Your report must cover the following:

- An explicit mathematical function to find the evaluation function from a given board
- A brief motivation for the evaluation function (a couple of sentences to one paragraph)
- A worked example of a midgame board showing the evaluation function score

Part II: Coding the Agent

Using minimax with alpha-beta pruning, code your algorithm to play the game. This must inherit from the ‘Connect4Player’ class.

Part III: Evaluation Function

Evaluate your agent against the benchmark agents.

- <YourAgent> vs StupidAI
- <YourAgent> vs RandomAI 5 times

- <YourAgent> vs MonteCarloAI 10 times

As player 1 and as player 2. Report your results in a table. For RandomAI and MonteCarloAI, set the seeds 1-5 for reproducibility. Note that while RandomAI and StupidAI can be seen as “sanity checks” that your algorithm is working, MCAI is a much more competitive player, but certainly beatable. Winners of previous contests typically beat MCAI all 20 times.

Interview

You will present your report to the design committee on the date & time of your choosing, in accordance with their availability. You will have 10 minutes in total, 8 of which will be reserved for your presentation and 2 for questions and answers. Your team will be expected to speak and answer questions in an egalitarian manner, hopefully splitting up the time evenly, with no single individual taking the majority of the time alotted. In your interview, cover the following:

- Summarize your answer to Part I. Use your written answer as a guide, but do not read directly from it.
- Describe how testing and finetuning influenced the final form of your evaluation function.
- Explain briefly (less than 1 minute) why your algorithm makes sense for this task
- Explain briefly (less than 1 minute) the steps you took to increase the efficiency of this task (eg ordering of nodes)
- Once as player 1 and once as player 2, demonstrate your minimax implementation’s performance against:
    - StupidAI
    - RandomAI

- Demonstrate your implementation's performance against MonteCarloAI four times, twice as player 1 and twice as player 2.

Where appropriate, you may discuss one topic while demonstrating another. For instance, you may discuss how testing influenced your evaluation function while your agent is playing against the benchmarks.

Important Functions & Calls

Commandline Interface

Command	Description	Datatype	Example	Default
-p1	Agent who will be acting as player 1. Name of agent eg minimaxAI	Stirng	-p1 minimaxAI -p1 monteCarloAI	human
-p2	Agent who will be acting as player 2. Name of agent eg minimaxAI	String	-p1 minimaxAI -p1 monteCarloAI	human
-seed	Seed for AI's with stochastic elements	int	-seed 0	0
-w	Rows of gamebaord	int	-w 6	6
-l	Columns of gameboard	int	-l 7	7
-visualize	Bool to use or not use GUI	bool	-visualize True -visualize False	True
-verbose	Sends move-by-move game history to shell	bool	-verbose True -verbose False	False
-limit_players	Which agents should have time limits. Useful if you want to play an AI but don't want to have the same time	Stirng	-limit_players 1,2 -limit_players -1,2 (player 1 is not limited) -limit_players 1,-1(player 2 is not limited)	1,2

	limit. In the format “x,y” where x and y are players. Values that are not 1 or 2 can be used in place of 1 or 2 if the player should not be limited			
-time_limit	Time limit for each player. No effect if a player is not limited. In the format “x,y” where x and y are floating point numbers.	Stirng	-time_limit 0.5,0.5	0.5,0.5