UNIVERSITY OF SASKATCHEWAN

Department of Computer Science
176 Thorvaldson Building
110 Science Place, Saskatoon, SK, S7N 5C9, Canada
Telephine: (306) 966-4886, Facimile: (306) 966-4884

CMPT 317
Winter 2023
Introduction to Artificial Intelligence

Assignment 6

Game Tree Search

Date Due: Thursday March 9, 11:59pm **Total Marks: 15**

General Instructions

- **This assignment is individual work.** You may discuss questions and problems with anyone, but the work you hand in for this assignment must be your own work.
- If you intend to use resources not supplied by the instructor, please request permission prior to starting. **You should not use any resource that substantially evades the learning objectives for this assignment.** You must provide an attribution for any external resource (library, API, etc) you use. If there's any doubt, ask! No harm can come from asking, even if the answer is no.
- Each question indicates what the learning objective is, what to hand in, and how you'll be evaluated.
- **Do not submit folders, or zip files, even if you think it will help.**
- Assignments must be submitted to Canvas.

Mega Man Battle Arena

The following is the description of a novel game that you will be exploring as part of this assignment.

Consider a 2-person perfect information game in the style of an **auto-battler**. In this game, both players will select a champion to enter a battle arena and the two champions will fight it out without further interference from the players. The two players are asymmetric:

1. Dr. Light (player one) controls 9 robot champions: Spark Man, Snake Man, Needle Man, Hard Man, Top Man, Gemini Man, Magnet Man, Shadow Man and of course his beloved robot son, Mega Man!

2. Dr. Wily (player two) controls 9 robot champions: Bubble Man, Air Man, Quick Man, Heat Man, Wood Man, Metal Man, Flash Man, Crash Man, and the stolen ultimate weapon peace-keeping robot, Gamma!

The objective of both players is to eliminate all of the champions on the opposing team.

How the Game Proceeds

The two players take turns, starting with Dr. Light. On a player's turn, that player must choose one of their remaining champions to enter the **battle arena**.

Only one champion per player can be in the arena at a time. If, at the start of a player's turn, they still have a champion in the arena (due to that champion having survived their previous battle), then the player can choose to send a new champion (with the damaged former champion withdrawing to the player's roster for now), or else 'send' the existing champion again (effectively just keeping them in place).

If, after sending a champion, both players now have a champion in the arena, then a battle will occur. The battle continues automatically without input from the players until one of the two champions in the arena is defeated!

How the Robots Battle

At the start of the game, all robots have **30 lifepoints**, except for the super-robot **Gamma**, who has **99 lifepoints**. In a battle, the two robots in the arena will trade blows, doing a set amount of damage per blow, until one or the other is defeated. The robot who is just newly entering the arena strikes first. The amount of damage each robot inflicts depends on which robot they are facing. Some robots are super effective against others!

Light Robot Damage

The 8 main robots belonging to Dr. Light do damage according to the following table. Mega Man is a bit special, and his damage will be described slightly later.

Attacker/Target	Metal	Quick	Air	Crash	Flash	Bubble	Wood	Heat	Gamma
Spark Man	2	1	4	1	1	4	1	1	1
Snake Man	1	4	2	1	1	1	4	1	2
Needle Man	1	1	1	2	4	2	4	2	1
Hard Man	4	1	2	7	1	2	2	2	4
Top Man	1	1	1	4	1	1	2	7	12
Gemini Man	1	4	2	1	4	1	1	1	1
Magnet Man	4	2	4	1	1	1	1	1	1
Shadow Man	2	2	1	1	2	4	2	4	2

Wily Robot Damage

The robots belonging to Dr. Wily are simpler and inflict damage as follows.

- All 8 robots **except** for Gamma deal 1 damage against Mega Man
- Gamma does 2 damage against Mega Man
- All of Dr. Wily's robots (including Gamma) do 2 damage to all other Light robots

Special Rules: Mega Man

As Dr. Light's greatest creation, Mega Man is a bit special. Whenever Mega Man defeats an opposing robot...

- Mega Man heals 5 lifepoints (this can take him above his starting value)
- Mega Man **gains the weapon** of the defeated robot to use in future battles

The damage inflicted by Mega Man's possible weapons against each target is in the table below. Mega Man will always choose to use the weapon that does the best damage in each battle.

Weapon/Target	Metal	Quick	Air	Crash	Flash	Bubble	Wood	Heat	Gamma
Mega (default)	1	2	2	1	2	1	1	2	1
Metal	14	0	0	0	4	4	1	1	1
Quick	4	0	2	1	0	2	0	2	2
Air	0	2	0	10	0	0	4	2	1
Crash	0	4	0	0	3	2	2	0	6
Flash	0	14	0	0	0	0	0	0	0
Bubble	0	0	0	1	2	0	0	6	10
Wood	0	0	8	0	0	0	0	0	0
Heat	2	6	4	2	3	0	30	0	0

Special Rules: Gamma

Dr. Wily's ultimate wepaon, Gamma, is very powerful and starts the game with 99 lifepoints. However Dr. Wily can only send Gamma into the arena once all of his other champions have been defeated.

Sample Game Play

The following is an example of how the game might unfold over the first few turns.

- Turn 1. The Arena is empty. Dr. Light sends Shadow Man into the arena.
- Turn 2. Dr. Wily sends Bubble Man into the arena to fight Shadow Man. A battle occurs! The robots will trade blows, with Bubble Man dealing 2 damage per blow and Shadow Man dealing 4 damage per blow. Shadow Man wins this battle with 14 lifepoints remaining.
- Turn 3. Dr. Light sends Needle Man to the arena (Shadow Man, with his 14 lifepoints, returns to Dr. Light's roster)
- Turn 4. Dr. Wily sends Metal Man to the arena. A battle occurs! The robots will trade blows, with Metal Man dealing 2 damage and Needleman dealing 1 damage. Metal Man wins this battle with 16 lifepoints remaining.
- Turn 5. Dr. Light sends Mega Man to the arena. A battle occurs! The robots will trade blows, with Mega Man dealing 1 damage and Metal Man dealing 1 damage. Mega Man wins the battle with 15 lifepoints remaining, but then heals by 5 up to 20 lifepoints, and also acquires Metal Man's weapon for future use.

The game then continues until one side or the other loses all of their champions.

Provided Code

A collection of Python scripts is provided to you. The files contain substantial internal documentation. The following is a light overview.

- `Minimax.py`: Depth-Limited Minimax Search (AIMA Chapter 5.2, 5.4)
- `AlphaBeta_Full.py`: Depth-Limited Minimax Search with Alpha-Beta pruning and Transposition Table (AIMA Chapter 5.3, 5.4)
- `Players.py`: Interface class to make it easier to play full games
- `compute_firstmove.py`: Top-level script to compute the minimax value of the game's initial state
- `play_full_games.py`: Top-level script to play full games between computer players and report a result
- `human_vs_machine.py`: Top-level script to allow a human to play vs a computer player at the console
- `Megaman_Arena.py`: Skeleton code for the game class for Mega Man Battle Arena. You will have to implement much of this.

Here is some advice for dealing with this codebase.

- You probably do not need to even open `Players.py` and `human_vs_machine.py` at all
- You are encouraged to lightly browse `Minimax.py` and `AlphaBeta_Full.py` and compare their differences, but you will not need to modify either one.
- You will need to make very light modifications to `compute_firstmove.py` and `play_full_games.py` to generate data. Mostly, this will involve changing the files that are being imported and modifying the lists of depth limits to try
- You will need to implement all of the missing publically-required game class methods in `Megaman_Arena.py`, as well as any helper functions that you think you need. These are marked with #TODO in the code.

Question 1 (2 points):

Purpose: To make sure you understand the game

Find one other person in the class and **play one game** of Mega Man Battle Arena. This will make sure that you understand the rules.

You will need a pen and paper to keep track of the robot healths, and you will need to have the rules handy for the damage charts.

Submit a report of the outcome of your game. Indicate who you played against, which side you played, and the game outcome. For the winning side, report which robots were still alive at the end and how many remaining lifepoints they had.

What to Hand In

- A document named `a6q1` with an appropriate extension (e.g., txt, pdf, docx, etc) containing your game report as described above

Evaluation

- 2 marks: Your game report includes all relevant information (players, sides, details of outcome)

Question 2 (7 points):

Purpose: To implement the rules of a new game

You have been provided with skeleton code in `Megama_Arena.py`. The `GameState` class is already completed with a suggested state representation. The `Game` class has headers for all the public methods that it needs to interact with the provided game search classes, some of which are missing a method body. You will need to:

- In class **Game**:
  - Implement actions()
  - Implement result()
  - Implement utility()
  - Design an evaluation function and implement eval()
  - Implement congratulate()

Each of these is marked with a `#TODO` in the provided code. You may, of course, also implement as many helper functions as you wish.

Evaluation Function

Most of the methods above simply need to be a correct implementation of the game rules. However, you will need to **design** your own evaluation function, which requires some creativity. Recall that the evaluation function maps a state to a number. The number represents a guess as to "how good" the state is for the max player. For example, in the game of chess, a typical evaluation function might involve counting how many white pieces are left on the board compared to the number of black pieces.

The more accurate your evaluation function, the better your computer player will perform. Make sure to clearly indicate in your code exactly what it is that your evaluation function is looking at. Also make sure that the numbers generated by your evaluation function are **always inside** the bounds of the numbers used in your `utility()` method.

What to Hand In

- Your modified version of `Megaman_Arena.py` implementing the game rules

Be sure to include your name, NSID, student number, and course number at the top of all documents.

Evaluation

- 5 marks: The Game methods are well implemented
- 2 marks: Your evaluation function is non-trivial and well-implemented

Question 3 (3 points):

Purpose: To quantify the effect of alpha-beta pruning and transposition tables and choose a depth-limit for a computer player

AIMA Chapter(s): 5.3-5.4

When you are playing a game against a computer, how long are you willing to wait for it to make a move? We'll explore that question here.

You have been provided with two game-tree search algorithms: basic minimax search, and minimax search with its main standard enhancements: alpha-beta pruning and a transposition table. Both of these algorithms are implemented using a **depth limit**. A higher depth-limit almost always means the algorithm will find higher-quality moves. In general, game-playing programs want to use a depth-limit that is as high as possible while still making a move within a reasonable amount of time.

Collect move-time data

First, use the provided script `compute_firstmove.py` to build the following table of data for Mega Man Arena for all depths between 1 and 9.

Algorithm	Depth limit	Time (sec.)
Minimax	1	
Minimax	2	
Minimax	...	
Minimax	9	
Alphabeta	1	
AlphaBeta	2	
AlphaBeta	...	
AlphaBeta	9	

You should expect plain minimax to be MUCH slower. Note that plain minimax and AlphaBeta will find the same move. AlphaBeta is simply a universally-used performance enhancement to minimax search. One of the goals here is just to see for yourself how much of a difference it makes.

Play Mega Man Battle Arena

Next, use the provided script `human_vs_machine.py` to **play two games** of Mega Man Battle Arena against your program. Play one game as Dr. Light and the other as Dr. Wily. For both games, use the highest depth limit from your table for which AlphaBeta search was able to make moves in what you consider to be a reasonable amount of wait-time. Report the results of your two games in your document, indicating the depth limit that you used.

What to Hand In

- Your document called `a6q3` with an appropriate extension (e.g., txt, pdf, docx, etc) that includes:
  - Your data table from running with different depth limits
  - Your report on the outcome of the games you played against your program

Evaluation

- 2 marks: Data tables are complete and readable and data is plausible
- 1 marks: You included your game output report

Question 4 (3 points):

Purpose: To consider and evaluate the concept of game balance

Highly asymmetric games that are fun and challenging to play can be difficult to design and often have problems with **balance**. Generally speaking, for a game to have good balance, the following properties are desirable:

- If both players are of roughly the same skill, either might win (or else a draw is likely, if draws are allowed)
- If one player is more skilled, that player should usually be able to win playing either side

You may, at this point, have developed an opinion as to whether you think Mega Man Battle Arena is currently balanced or not.

Your task is to think about small changes to the rules of the game that might improve the balance of the game (and hopefully make it even more fun to play at the same time). You can think about changing starting lifepoints, changing weapon damages, or other simple rules changes. Come up with a few ideas and then start evaluating them.

To evaluate your proposal, you'll need to generate data indicating that the balance goals stated above have been roughly achieved. To do this, it will be useful for you to be able to pit computer players of different skill against each other. The provided script `play_full_games.py` helps you do this. In the context of computer players, using a higher depth limit can nearly always be used to mean "more skilled". Use the script to play entire games using a variety of skill levels (i.e. depth limits) for both players. It may take a few tries for you to find a change that you believe is balanced, so stay organized with your data collection.

Once you have settled on a result, you must submit both your proposed rules change and the data that you believe backs it up in an organized and readable format. Briefly explain how your data shows that you have met the balance objectives shown above.

What to Hand In

- A document named `a6q4` with an appropriate extension (e.g., txt, pdf, docx, etc) containing:
  - Your proposal for your rules change
  - Data (likely in table format) that supports your conclusion that the game is now reasonably balanced
  - Your explanation for how your data supports your proposed value

Be sure to include your name, NSID, student number, and course number at the top of all documents.

Evaluation

- 1 marks: The rules change you propose is reasonable
- 2 mark: Your data is well-organized and you convincingly show it supports your proposal