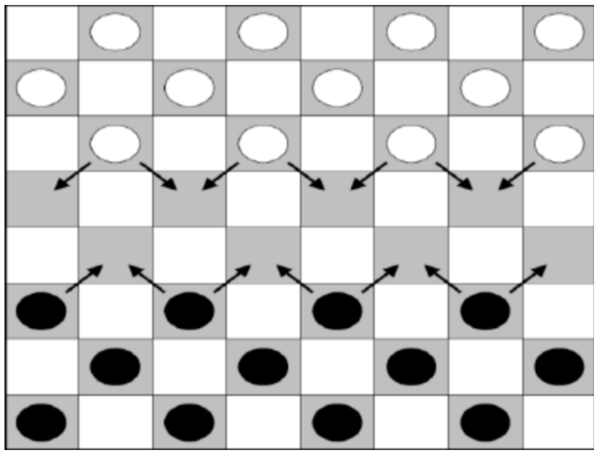

Checkers

Student Manual

for Introduction to Artificial Intelligence

Table Of Contents

1. Introduction

In this programming assignment, you are tasked with implementing a Checkers AI agent, which should be able to solve a Checkers game. You will have the choice of programming in Python, Java, or C++. A code base will be made available to you, and you are asked to edit one or more files from whichever shell you choose to use. Your agent should be able to read percepts and act rationally; your grade will be determined by your agent's performance measure. At the end of the quarter, your agent will compete against your classmates' agents in a class tournament. **All code must be able to run on Openlab.**

2. Team Formation

Students are allowed to work with one other student (teams of two) to complete the coding project. Everyone must create a team formation to submit on Canvas, even those who are working alone; see the Team Formation Instructions pdf on Gitlab for submission details related to Canvas.

3. Checkers World Game Mechanics

Environment

This shell follows the American/British ruleset for Checkers. Found on https://en.wikipedia.org/wiki/English_draughts. The rules are similar except the Checkers board in this shell can have a variable size governed by the following rules:

Board Parameters:

M = number of rows

N = number of columns

P = number of rows occupied by pieces in the initial state

Q = number of unoccupied rows that separate the two sides in the initial state

Parameter Constraints:

$Q > 0$

$M = 2P + Q$

$N*P$ is even

Actuators

Your AI will ONLY know what move your opponent just made. If your AI moves first, you will receive a Move object with col = -1, row = -1

Sensors

Your AI should return a Move object to tell the shell which step you are going to make. For more information, please read the shell manual.

4. Tasks to Complete

Setup Your Environment

In this section, you will find help setting up your coding environment. This project will use UCI's openlab; any other coding environment is not supported.

Install Required Applications

To connect to openlab, you will need to use SSH. SSH stands for Secure SHell. It is a program designed to allow users to log into another computer over a network, to execute commands on that computer and to move files to and from that computer. A Mac user can use the terminal application whereas a Windows user will either need to install PuTTY or use Git. You can download PuTTY from here. Download the MSI installer for Windows, and run the installer for PuTTY.

Connect to Openlab

Connecting to openlab is as easy as SSHing into the openlab server. If you are on Windows and using PuTTY, type "openlab.ics.uci.edu" into the Host Name box; make sure the port is 22 and the SSH flag is ticked. Click open, and login using your ICS account info. If you are using a Mac or using Git, open the Mac terminal found under Applications -> Utilities or open Git Bash and enter 'ssh ICSUSERNAME@openlab.ics.uci.edu' to login using your ICS account info.

Extra information about openlab:

- https://www.ics.uci.edu/~lab/students/#unix

- https://www.ics.uci.edu/computing/linux/hosts.php

Program Your AI

Once you have your environment setup, you can start to program your agent. In the 'src' folder of your shell you will find the source code of the project. You are only allowed to make changes to the **StudentAI** class.

Compile Your AI

Compiling your program is easy as executing the command **make** from the shell's root directory (the directory with the makefile in it) for the Java and C++ shells. For the Python shell, simply save the changes you made to your StudentAI file (Shift +zz if you're editing using vim).

Test Your AI

To run and test your AI, see page 8. If you are using the Python Shell make sure you are using Python 3.5.2. On openlab, run the command **module load python/3.5.2** to load Python 3.5.2.

Submit Your Project

At this point you should have your most up-to-date source code in the 'src' folder. Navigate out to the "Tools" folder, where you should find a script called submission.py. Please use the command ***python3 submission.py*** to make your zip file, and submit the zip file in Canvas.

Write Your Project Report (end of quarter)

After you've completed your Final AI at the end of the quarter, write a report according to the template file provided on Gitlab. Template is also shown on the last page of this student manual. The report is due one day after the Final AI. Make sure to turn in your report in **pdf** format.

5. Understanding the Tournament

After you have submitted your Final AI, you will be entered into a tournament with your classmates. (Late submissions are not valid for the tournament.) Every agent is run on the same board to ensure fairness; your agent will be timed-out if it hangs for longer than 8 minutes. Tournament results will determine how many extra credit points you get for writing a better AI than your classmates. Results/rankings will be posted to the class.

5. Grading Policy

1st Deadline for Minimal AI

40% for successful submission (submitted and could run without errors), 60% comes from 60% wins against Random AI.

For example, if your AI win 50% of the games against Random AI, your score will be 40%+50%*60%/60%=90%. If your AI win more than 60% of the games, your score will be 100%.

2nd Deadline for Draft AI

40% for successful submission (submitted and could run without errors), 60% comes from 60% wins against Poor AI.

3rd Deadline for Final AI

28% for successful submission (submitted and could run without errors), 72% comes from 60% wins against Average AI.

For example, if your AI win 50% of the games against Average AI, your score will be 28%+50%*72%/60%=88%. If your AI win more than 60% of the games, your score will be 100%.

6. Shell Manual

Board Class:

Summary

This part describes the structure of Board class under Board.cpp/Board.java/BoardClasses.py

You can import/include this class into your AI file to use this class.

If you want to write your own board class, please make sure to include all code into your StudentAI.py/.cpp/.java. Multiple files for your AI IS NOT ALLOWED.(For C++ coders, you can have .cpp file and .h file both.)

Variables (in the init function)

col: The number of columns.

row: The number of rows.

p: Number of rows filled with Checker pieces at the start.

Functions

show_board: Prints out the current board to the console

get_all_possible_moves: Returns all moves possible in the current state of the board

is_win: Check if there is a winner. Return which player wins.

Move Class:

Summary

This part describes the structure of Move class under Move.cpp/Move.java/Move.py

This class already imported to your StudentAI.

Your AI must return a 'Move' object to describe the move

Do not change this file

Variables (in the init function)

l: list describing the move

Functions

from_str: Makes a move object from a str describing the move.

Checker Class:

Summary

This part describes the structure of Checker class under Checker.cpp/Checker.java/Checker.py

You can import/include this class into your AI file to use this class.

Variables (in the init function)

color: color of the checker piece.

location: a tuple describing the location of the checker piece

Functions

get_color: Returns the color of this piece.

get_location: Returns the location of this piece.

get_possible_moves: Returns all moves possible for this checker piece in the current state of the board

Running your AI:

Manual Mode

After you compile your AI, use the following commands to run your AI in manual mode:

Python:

```
python3 main.py {col} {row} {p} m {start_player (0 or 1)}
```

C++:

```
./main {col} {row} {p} m {start_player (0 or 1)}
```

Java:

```
java -jar Main.jar {col} {row} {p} m {start_player (0 or 1)}
```

Play Against Other AI's

The shell also supports playing against other local AI shells written in different programming languages or against other AI's over the open network. To play across the network, connect through the school VPN (make sure you're using school wifi). Navigate to **Tools/AI_Runner.py** shell and run the following commands:

Local:

```
python3 AI_Runner.py {col} {row} {p} l {AI_1_path} {AI_2_path}
```

Network:

```
python3 AI_Runner.py n {AI_path}
```

Note: {AI_path} indicates:

1. main.py for Python
2. main (executable) for c++
3. Main.jar for java

In other words, provide the path to your “main” file as the {AI_path)

7. Note

All the various CS-171 AI project shells were written by former CS-171 students who became interested in AI and signed up for CS-199 in order to pursue their interest and write more interesting AI project shells. Please let Prof. Lathrop know if this is of interest to you (CS-171 grade of A- or better required).

8. Recommendations

It is recommended to use the JetBrains suite (PyCharm, CLion, IntelliJ IDEA) to code your project. These are powerful and user friendly IDEs and they are available for free to all students. As debugging on openlab is a pain, these IDEs might make your task a bit easier.

Make sure that the compiler you use for your project matches the compilers offered on openlab. **Any submission that doesn’t compile on openlab will not be graded.**