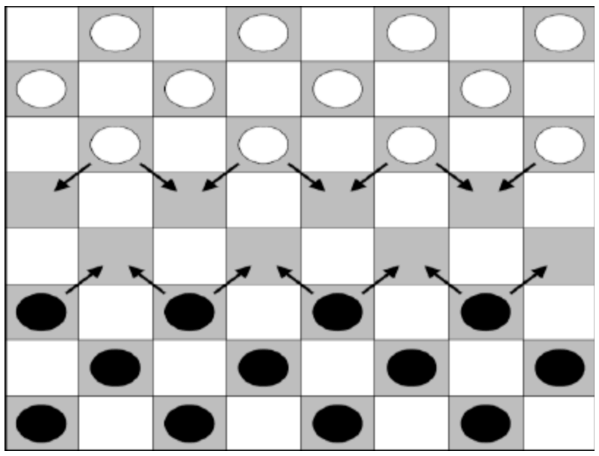

Checkers

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 tournament. **Remember, you should run all codes in openlab.**

2. Team Formation

Please form projects teams (individual or pairs) and submit your team information.

Submission

- Please use only letters, digits in your team name.
- Please do not use space, underscore or any other special characters in your team name.
- Team names may not contain any personally identifiable information, especially not the students' name or ID nor any part thereof. Doing so would result in a violation of legally protected student private confidential information, because we will release the results of the Tournament publicly by team name.
- Team names may not contain any personally identifiable information, especially not the students' name or ID nor any part thereof. Doing so would result in a violation of legally protected student private confidential information,

because we will release the results of the Tournament publicly by team name.

- Please follow the **exact** format shown below or points will be deducted.

```
TeamName: <enter_team_name>

Member1: <enter member 1 name only,format: LastName, FirstName>

Member2: <enter member 2 name only,format: LastName, FirstName>
```

Example:

```
TeamName: SmartAI

Member1: John Doe

Member2: Jane Doe
```

If there is no partner, please delete third line.

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

You 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.Task to Complete

Setup Your Environment

In this section, you will find help setting up your coding environment. This project will take advantage of 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 need to install PuTTY. 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, open the terminal found under Applications -> Utilities. Enter 'sshICSUSERNAME@openlab.ics.uci.edu' and 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 MyAI 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).

Test Your AI

To run your program after you have compiled it, navigate to the bin folder. You should find the compiled program inside. Refer to the Shell Manual Appendix for help running it. To generate large amounts of worlds to use with the folder option, refer to the World Generator. If you are using the Python Shell make sure you are using Python 3.5.2. On openlabs, run the command **module load python/3.5.2** to load Python 3.5.2.

Write Your Project Report

Write a report according to Professor's instructions. Make sure your report is in **pdf** format. Template will be provided. **The template is in last page**.

Submit Your Project

At this point you should have your most up-to-date source code in the 'src' folder and also have submission.py script under tools. Please use the command ***python3 submission.py*** to make your zip file, and submit the zip file in Canvas.

The Final report is due one day after Final AI. You should submit a pdf version in Canvas.

5. Understanding the Tournament

After you submit your project and the deadline passes, you will be entered into a tournament with your classmates. The tournament checks to make sure you followed all the instructions correctly, then runs your agent against all other classmate's agent. Every agent is run on the same board to ensure fairness. Your agent will be timed-out if it hangs for longer than 2 hours. After the scoreboard is constructed and checked for any illegal submissions.

6. Grading Policy

Manual AI (Given to students for testing)

1st Deadline for Draft AI

Your AI should beat Random AI (6 times) 40% for successful submission, 10% per test

2nd Deadline for Minimal AI

Your AI should beat Poor AI (6 times) 40% for successful submission, 10% per test

3rd Deadline for Final AI

Your AI should beat Average AI (6 times)28% for successful submission, 12% per test

7.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 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.

k: No 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:

Summary

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

Python:

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

C++:

/main.cpp {col} {row} {k} m {start_player (0 or 1)}

Java:

java -jar main.java {col} {row} {k} l {AI_1_path} {AI_2_path}

To help revise your AI, we will provide sample AI for your testing later.

To play against other AIs use the following command:

python:

```
python3 main.py {col} {row} {k} l {AI_1_path} {AI_2_path}
```

C++:

```
./main.cpp {col} {row} {k} l {AI_1_path} {AI_2_path}
```

Java:

```
java -jar main.java {col} {row} {k} l {AI_1_path} {AI_2_path}
```

To play against other AIs over the openly network. Connect through the school VPN and use the following commands:

python:

```
python3 main.py {col} {row} {k} n {AI_path}
```

C++:

```
./main.cpp {col} {row} {k} n {AI_path}
```

Java:

```
java -jar main.java {col} {row} {k} n {AI_path}
```

8. 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).

9. 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.**