CSCI-561 - Spring 2019 - Foundations of Artificial Intelligence Homework 1

Due on February 4th, 2019 11:59:59PM PT

Introduction

In this homework, you will write a program to play an adversarial board game called the *Laser Checkmate*. Your program will generate one move at a time, taking turns against an AI opponent. Your goal is to win the game.

Rules of the Game

Laser Checkmate is a board game in which two players take turns placing laser emitters on a square $N \times N$ board. Each emitter shoots laser beams in eight directions: up, down, left, right, and all four diagonal directions. The beams travel up to 3 squares in each direction, but are unable to travel through walls (around the border of the grid) or blocks (occupying squares within the grid). The goal of the game is to place emitters so that your beams cover more squares than your opponent's lasers do.

Figure 1 shows a 5×5 board with one laser emitter placed in the middle square. The black squares are blocks. The blue squares are covered by the emitter's laser beams. In this example, the emitter covers 12 squares.

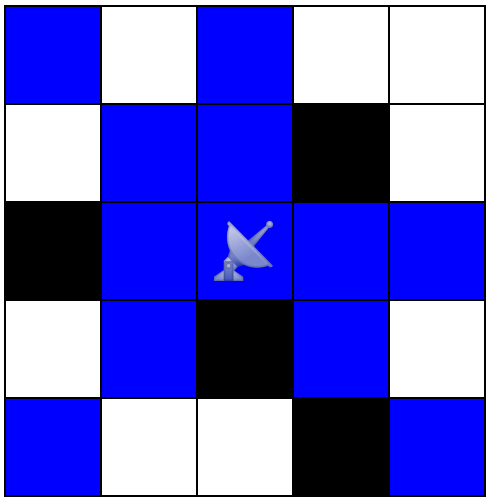

Figure 1.

The other player can now place an emitter in only the white squares. However, once the players places a new laser emitter in a white square, its beams can travel through blue squares as well. In other words, coverage of a square is not mutually exclusive, and a square can be covered by both players at the same time.

The game terminates when there are no valid moves. The final score for each player is the number of squares that their laser beams cover, including the squares where their laser emitters are placed. The game winner is the one who has the higher score in the end.

Valid Moves

The players take turns placing laser emitters on the grid. On each player's turn, a player places a new laser emitter on a square that contains no laser emitter, wall, or brick and that is not covered by any laser (from either player).

A move is specified by two numbers, a row number followed by a column number, each in the range 0 to *N*-1, as illustrated in the 7×7 board below:

(0, 0)	(0, 1)	(0, 2)	(0, 3)	(0, 4)	(0, 5)	(0, 6)
(1, 0)	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(1, 6)
(2, 0)	(2, 1)	(2, 2)	(2, 3)	(2, 4)	(2, 5)	(2, 6)
(3, 0)	(3, 1)	(3, 2)	(3, 3)	(3, 4)	(3, 5)	(3, 6)
(4, 0)	(4, 1)	(4, 2)	(4, 3)	(4, 4)	(4, 5)	(4, 6)
(5, 0)	(5, 1)	(5, 2)	(5, 3)	(5, 4)	(5, 5)	(5, 6)
(6, 0)	(6, 1)	(6, 2)	(6, 3)	(6, 4)	(6, 5)	(6, 6)

Figure 2.

Figure 3 shows a board (N=7) after one move. There are blocks at (0,4), (2,5), and (6,2). The player has chosen (3,5) as the first move. The squares covered by this new laser emitter are colored blue. The laser beam traveling up is blocked at (2,5), so the emitter does not cover (0,5) or (1,5). Because the beams all have a range of 3 squares, the beam traveling left does not reach (3,0) or (3,1). Overall, there are 15 squares covered by this emitter, so the player is currently winning 15-0.

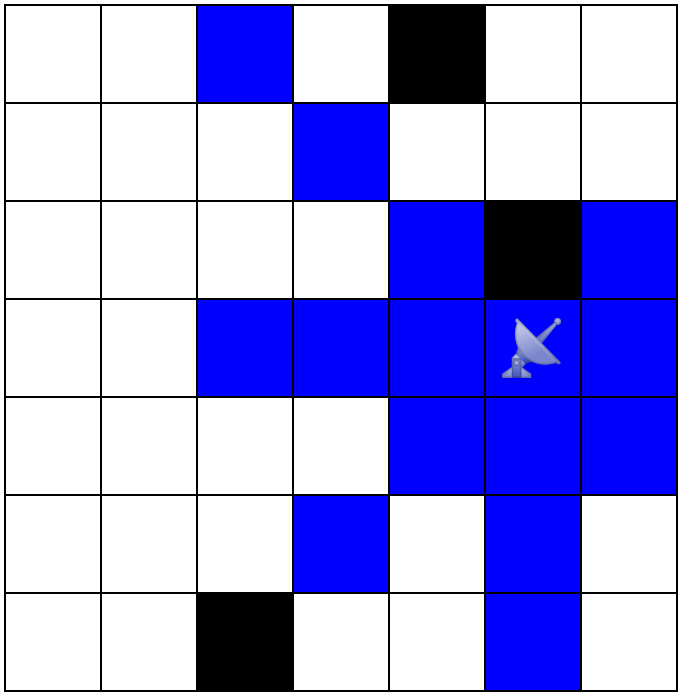

Figure 3.

It is now the other player's turn. Any squares not colored blue or black in Figure 3 are possible moves, so there are 31 moves to choose from. The player selects (1,2), and Figure 4 shows the resulting board. Squares covered by only this new emitter are colored red, while squares that are covered by both emitters are colored purple. The second player's emitter covers 17 squares, so now the first player is losing 15-17.

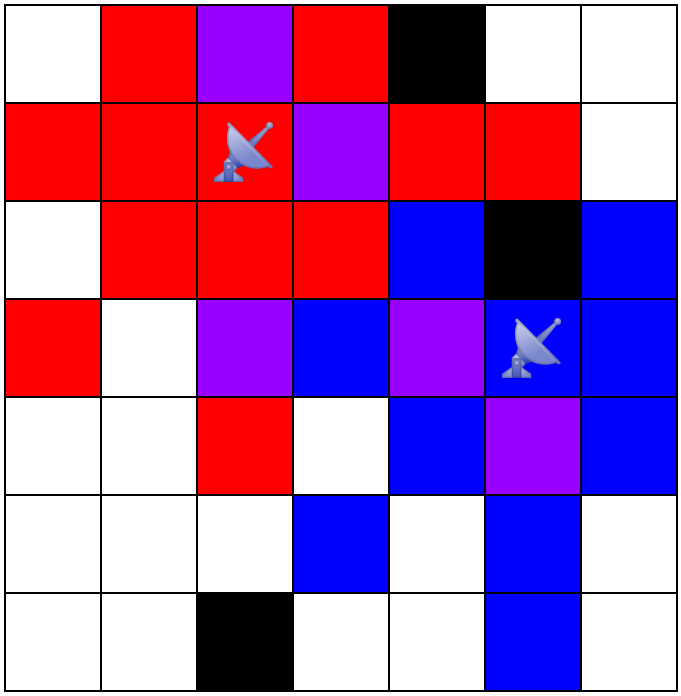

Figure 4.

It is now the first player's move again. The only valid moves are the 19 white squares in Figure 4. The game continues until there are no valid moves left (i.e., no white squares).

Your Task

Your task is to write a program that can play the Laser Checkmate Board Game. Given a board in a game, your program needs to make an optimal move; in other words, your program must specify a square for your next laser emitter that will maximize your chances of winning.

Input and Output Format

Your program must not require any command-line arguments. It should read the input file named "input.txt" in the same directory that contains a problem definition. The output should be written to a file named "output.txt" with your chosen move to the same current directory. Format for the "input.txt" and "output.txt" files is specified below.

Input:

The file "input.txt" in the current directory of your program will be formatted as follows:

First line: strictly positive integer ***N***, the width and height of the ***N*** x ***N*** grid.

*Next **N** lines:* a series of ***N*** digits representing each grid space. **0** represents an empty space, **1** represents one of your laser emitters, **2** represents one of your opponent's laser emitters, and **3** represents a block.

The input file for the grid shown in Figure 2 would be:

```
7
0000300
0000000
0000030
0000000
0000000
0000000
0030000
```

Output:

Two integers separated by a space, indicating the coordinates (row followed by column) of the square where you choose to put your next laser emitter.

The output file for the move that produced the board in Figure 3 would be:

```
3 5
```

Sample Input

You are provided sample input files. The goal of the samples input is for you to check if your program can correctly and repeatedly parse the boards and generate moves to successfully complete a whole game of Laser Checkmate. The samples are representative of the grading test cases, but it should not be assumed that if your program works on the samples it will work on all grading test cases. It is your task to make sure that your program will work correctly on any valid input. Please note that the output of your program should match the specified format *exactly*. Failure to do so will most certainly lose points.

Grading Criterion

Your program will play against an AI in a game that starts from the board given in the test case input file (with your program always getting the first move). If your program wins the game, you get full credit for the test case; if it loses or ties, you get no point for this test case. Your grade for this homework will be the sum of points your program earns over all the test cases.

For each test case, the grading script will place an "input.txt" file in your work folder, run your program (which reads "input.txt"), and extract your next move from the "output.txt" file

generated by your code. The grading script will then play your move, play the AI's move, and generate a new "input.txt" file for your program's next move.

Notes:

1. **You don't need to consider unsolvable inputs and inputs with no valid moves.** You will not be graded using irregular inputs that don't conform to the format described in this document.
2. **Some inputs may have more than one valid solution.** We will accept all solutions that win the game.

Homework Rules

1. Use only Python 2.7 to implement your homework assignment. You are allowed to use only standard libraries. You have to implement any other functions or methods by yourself.
2. When you submit your homework on Vocareum, you will receive feedback on if your program is making errors, if any. You can submit your homework as many times as you like up until the deadline. Only your last submission will be graded.
3. Your program should be named "hw1cs561s2019.py". When you submit the homework on Vocareum, the following commands will be executed:
   python hw1cs561s2019.py
4. Homework must be submitted through Vocareum. Homework submitted through any other channel will NOT be accepted. Please only upload your code to the "/work" directory. **Don't create any subfolders or upload any other files**. Please refer to http://help.vocareum.com/article/30-getting-started-students on how to get started with Vocareum.
5. **For your final submission, please do not print any logs to the console other than the required output.**
6. Your program must handle all test cases within a maximum runtime of **1 minute** per move on Vocareum.
7. Homework submitted after the deadline will not be graded. It is recommended that you submit versions of your homework early to avoid any submission issues on Vocareum.