**AI Term Project Spring 2019**

Due Date: 11:50PM May 31, 2019

Project:

Design an AI program to play a two-player n×n board game.

Language:

Any commonly used programming language, e.g. C/C++, Java, Python, etc.

What to turn in:

1. A report of your search strategy (be precise),
2. A copy of your source code (be correct, i.e. it runs),
3. A document explaining how to run your code (be clear),
4. A complete program output that shows the sequence of moves for a pilot test (posted on e3 later). Note. A fancy output is not required. Just clearly show the alternating moves made by you and your AI program (be neat, not necessarily fancy).

Grading policy:

1. You will get reasonable points for your project if you turn in what you are supposed to even though your AI program is not among the top-ranking programs in class.
2. The higher ranking, the more points, of course.
3. And yes, to be fair, an incomplete project will incur a low grade.

Project description:

1. Introduction

You are asked to implement an AI program to play a two-player n×n board game, e.g. 4×4.

|  |  |  |  |
| --- | --- | --- | --- |
| R | B | B | R |
| R | B | R | B |
| B | R | B | R |
| B | B | R | R |

Fig. 1. A random initial board configuration.

Each tile is colored in Red (denoted by R) or Blue (denoted by B), and the numbers of the color tiles are even, e.g. 8 Red and 8 Blue on a 4×4 board, as shown above.

1. Objective

You can choose R or B first, and then you can choose to make the 1st move or the next.

Each single move involves the removal of a colored tile of the choice. For example, given the initial board configuration above, assuming your color is R, and you’ll make the 1st move, one possible legal move will be removing the R on the upper left corner. The new board configuration is shown below after you remove that R.

|  |  |  |  |
| --- | --- | --- | --- |
|  | B | B | R |
| R | B | R | B |
| B | R | B | R |
| B | B | R | R |

Fig. 2. New board configuration after removing the R on the upper left corner.

This is a two-player game that takes alternating turns. After your move, it is the AI program’s turn to make its move, e.g. removing the B on the low left corner. Now the new board configuration becomes

|  |  |  |  |
| --- | --- | --- | --- |
|  | B | B | R |
| R | B | R | B |
| B | R | B | R |
|  | B | R | R |

Fig. 3. New board configuration after AI’s move.

The one that removes all the colored tiles of the choice first is the winner. In this example, if you remove all the R’s before the AI removes all the B’s, you win.

At this point, the game may look naïve because whoever makes the 1st move is guaranteed to win. Nevertheless, there are other rules of the game, as detailed below, that make this game a bit more complicated.

1. Rules

* The removal of a tile will cause side effects that apply ONLY to its “direct” neighbors regardless of their colors.

E.g. the removal of the R will affect its direct neighbors right above, right below, to its left, and to its right (as indicated by the four question marks).

|  |  |  |  |
| --- | --- | --- | --- |
|  | ? |  |  |
| ? | R | ? |  |
|  | ? |  |  |
|  |  |  |  |

Fig. 4. Case 1. side effects.

E.g. the removal of the R will affect its direct neighbors right below and to its right (as indicated by the two question marks).

|  |  |  |  |
| --- | --- | --- | --- |
| R | ? |  |  |
| ? |  |  |  |
|  |  |  |  |
|  |  |  |  |

Fig. 5. Case 2 side effects.

E.g. the removal of the R will affect its direct neighbors right below, to its left, and to its right (as indicated by the three question marks).

|  |  |  |  |
| --- | --- | --- | --- |
| ? | R | ? |  |
|  | ? |  |  |
|  |  |  |  |
|  |  |  |  |

Fig. 6. Case 13. side effects

* The side effects can cause the automatic removals of these direct neighbors under certain conditions.

Each tile (as a square) has 4 sides. After a tile is removed, you need to check its direct neighbors. Depending on its location on the board, it can have at most 4 direct neighbors (note. the colors of neighbors do not matter). For each of these neighbors, unless it has three or four sides connected to other tiles or to the wall (side) of the board, it will be removed automatically.

E.g.

Current board configuration:

|  |  |  |  |
| --- | --- | --- | --- |
|  | R | B |  |
| B | R | B | R |
| R | R | B | B |
|  |  |  |  |

After removing R (4 direct neighbors in green):

|  |  |  |  |
| --- | --- | --- | --- |
|  | R | B |  |
| B |  | B | R |
| R | R | B | B |
|  |  |  |  |

Evaluate side effects on direct neighbors (in green):

|  |  |  |  |
| --- | --- | --- | --- |
|  | R | B |  |
| B |  | B | R |
| R | R | B | B |
|  |  |  |  |

After automatic removal:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | B |  |
|  |  | B | R |
| R |  | B | B |
|  |  |  |  |

**NOTE. The tiles (2 B’s and 1 R in orange) will NOT be removed (not even the orange R) because chain reaction (or ripple effect) is NOT considered in this game. The side effects ONLY apply to direct neighbors (in green).**

1. Two video clips are posted on Google for your reference.

<https://drive.google.com/open?id=1TWYRQe_kPSdtLMWfB2e1hJEh_DNxM9Xv>

<https://drive.google.com/open?id=1JgbZkChmhZYcH2N7yW7FnOX6m18fXKF6>