Project #2: Minesweeper

Aditya Dhawan (Section #3), Annie Thach (Section #6)

1 Introduction

1.1 Statements

All work shown in the following report and in related code files is entirely our own. No online work nor work done by other students were used in this project. A.D., A.T.

Write-up done in LateX.

1.2 Contributions

Coding: Aditya, Annie
Write-up: Aditya, Annie

1.3 About

This program was done in Java.
To compile, go to the source directory, "src", and type into the terminal:
javac Minesweeper/Main.java
To run this program, while in the source directory, type into the terminal:
java Minesweeper.Main
The user will then be presented with a list of commands and prompted to input commands. The commands are as follows:
'g' to generate new board.
'b' to run basic agent.
'a' to run advanced agent.
'avg' to get the average scores of both agents for the current board.
'pb' to print the original board.
'pkb' to print the basic agent's resulting knowledge base.
'pka' to print the advanced agent's resulting knowledge base.
'h' to bring this list up again.
'q' to quit the program.

2 Questions

2.1 Representation

How did you represent the board in your program, and how did you represent the information / knowledge that clue cells reveal? How could you represent inferred relationships between cells?

The board itself is represented as an array of integers, where the numbers represent the state of the cells. -1 represents the mine and numbers greater than or equal to 0 are clues that tell the player how many mines are adjacent to the cell.

The basic agent's knowledge base is a parallel matrix that represents the board, but with a data structure to hold information about each cell. The Cell data structure tracks whether or not the cell has been revealed, the clue that was revealed (taken from the board when the agent decides to select the cell), the number of safe neighbors around the cell, the number mines revealed around the cell, and the number of hidden cells as well as the agent's guess as to whether or not the cell is safe or a mine.

In addition to the knowledge base the agent keeps a stack of safe cells to visit and a list of suspected mines to not visit. Cells that have been deemed safe using the knowledge base are represented with 's' and suspected mines are represented by 'm'.

The improved agent selects cells with the highest probability of being safe. It represents information and knowledge the exact same way as the basic agent, except it makes use of a ModifiedCell data structure. This structure carries with it additional instance variables that represent its probability of being a mine, which is based on both the number of clues that neighbor it and the total number of unrevealed neighboring cells around those clues.

2.2 Inference

When you collect a new clue, how do you model / process / compute the information you gain from it? In other words, how do you update your current state of knowledge based on that clue? Does your program deduce everything it can from a given clue before continuing? If so, how can you be sure of this, and if not, how could you consider improving it?

Whenever a cell is selected, it then updates the probability values of the surrounding cells that have not yet been revealed, and from those values determines whether neighboring cells are either mines or safe. To represent this information, we make use of ArrayLists that update whenever a cell can be concluded as safe or as a mine. These ArrayLists hold the Index type, which is an abstract data type that holds information on the specific position of the cell. Essentially, before a cell is selected, the program checks to see if the ArrayList for safe cells contains anything to check first, that way the agent

can step to those cells to clear more of the board and gain more clues. If in the case that there is nothing in the ArrayList for safe cells, the program will try to randomly select a position in the matrix, and will check against the ArrayList for cells that are suspected as mines in order to avoid them.

We believe that our program is able to deduce as much as possible from an uncovered clue. If we consider how the program is implemented, a cell that neighbors a previously-revealed clue will have already had its probability of being a mine set accordingly. In turn, once we uncover a new clue that neighbors that same cell, it will cause the probability of that cell to update to reflect the combined chance of being a mine from both clues. From this reasoning, we can be sure that any newly-uncovered clue will work in tandem with previously-uncovered clues to deduce probable mines and safe spots.

2.3 Decisions

Given a current state of the board, and a state of knowledge about the board, how does your program decide which cell to search next?

Given the current state of the board and knowledge about the board, the program prioritizes selecting any cell that exists within the ArrayList for safe cells. Cells would have been added to this ArrayList if they were confirmed to be safe by previous clues. If this ArrayList happens to be empty, then the program will attempt to select a cell semi-randomly, while taking probabilities into account; the lower the probability that a cell is a mine, the more likely it is to be selected.

2.4 Performance

For a reasonably-sized board and a reasonable number of mines, include a play-by-play progression to completion or loss. Are there any points where your program makes a decision that you don't agree with? Are there any points where your program made a decision that surprised you? Why was your program able to make that decision?

Original 10 x 10 board:

−1	2	0	1	−1	3	−1	3	1	1
−1	3	0	1	1	3	−1	3	−1	1
−1	3	0	0	0	1	1	2	1	1
−1	2	0	0	0	0	0	0	0	0
1	1	0	0	0	1	1	1	0	0
0	0	1	1	1	1	−1	1	0	0
0	0	2	−1	2	1	1	1	0	0
1	2	4	−1	3	2	2	2	1	1
1	−1	−1	2	2	−1	−1	2	−1	1
1	2	2	1	1	2	2	2	1	1

Basic agent's knowledge base:

m	2	0	1	-1	3	-1	3	1	1
m	3	0	1	1	3	-1	3	m	1
m	3	0	0	0	1	1	2	1	1
m	2	0	0	0	0	0	0	0	0
1	1	0	0	0	1	1	1	0	0
0	0	1	1	1	1	-1	1	0	0
0	0	2	-1	2	1	1	1	0	0
1	2	4	-1	3	2	2	2	1	1
1	m	-1	2	2	-1	m	2	-1	1
1	2	2	1	1	2	2	2	1	1

Advanced agent's knowledge base:

m	2	0	1	m	3	-1	3	1	1
m	3	0	1	1	3	-1	3	m	1
m	3	0	0	0	1	1	2	1	1
m	2	0	0	0	0	0	0	0	0
1	1	0	0	0	1	1	1	0	0
0	0	1	1	1	1	m	1	0	0
0	0	2	m	2	1	1	1	0	0
1	2	4	m	3	2	2	2	1	1
1	m	m	m	2	m	m	2	m	1
1	2	2	1	1	2	2	2	1	1

Basic Agent Play-By-Play

Selected (7, 0); revealed safe (1), 0 safe neighbors, 0 mines, 5 hidden neighbors, unknown
Selected (4, 8); revealed safe (0), 0 safe neighbors, 0 mines, 8 hidden neighbors, unknown
Safe cells: [(3, 7), (3, 8), (3, 9), (4, 7), (4, 9), (5, 7), (5, 8), (5, 9)]
Selected (5, 9); revealed safe (0), 1 safe neighbors, 0 mines, 4 hidden neighbors, probably safe
Safe cells: [(3, 7), (3, 8), (3, 9), (4, 7), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9)]
Selected (6, 9); revealed safe (0), 1 safe neighbors, 0 mines, 4 hidden neighbors, probably safe
Safe cells: [(3, 7), (3, 8), (3, 9), (4, 7), (4, 9), (5, 7), (5, 8), (6, 8), (7, 8), (7, 9)]
Selected (7, 9); revealed safe (1), 1 safe neighbors, 0 mines, 4 hidden neighbors, probably safe
Safe cells: [(3, 7), (3, 8), (3, 9), (4, 7), (4, 9), (5, 7), (5, 8), (6, 8), (7, 8)]
Selected (7, 8); revealed safe (1), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Safe cells: [(3, 7), (3, 8), (3, 9), (4, 7), (4, 9), (5, 7), (5, 8), (6, 8)]
Selected (6, 8); revealed safe (0), 4 safe neighbors, 0 mines, 4 hidden neighbors, unknown
Safe cells: [(3, 7), (3, 8), (3, 9), (4, 7), (4, 9), (5, 7), (5, 8), (6, 7), (7, 7)]
Selected (7, 7); revealed safe (2), 2 safe neighbors, 0 mines, 6 hidden neighbors, probably safe
Safe cells: [(3, 7), (3, 8), (3, 9), (4, 7), (4, 9), (5, 7), (5, 8), (6, 7)]
Selected (6, 7); revealed safe (1), 3 safe neighbors, 0 mines, 5 hidden neighbors, unknown

Safe cells: [(3, 7), (3, 8), (3, 9), (4, 7), (4, 9), (5, 7), (5, 8)]
Selected (5, 8); revealed safe (0), 5 safe neighbors, 0 mines, 3 hidden neighbors, unknown
Safe cells: [(3, 7), (3, 8), (3, 9), (4, 7), (4, 9), (5, 7)]
Selected (5, 7); revealed safe (1), 4 safe neighbors, 0 mines, 4 hidden neighbors, probably safe
Safe cells: [(3, 7), (3, 8), (3, 9), (4, 7), (4, 9)]
Selected (4, 9); revealed safe (0), 3 safe neighbors, 0 mines, 2 hidden neighbors, probably safe
Safe cells: [(3, 7), (3, 8), (3, 9), (4, 7)]
Selected (4, 7); revealed safe (1), 3 safe neighbors, 0 mines, 5 hidden neighbors, unknown
Safe cells: [(3, 7), (3, 8), (3, 9)]
Selected (3, 9); revealed safe (0), 2 safe neighbors, 0 mines, 3 hidden neighbors, probably safe
Safe cells: [(3, 7), (3, 8), (2, 8), (2, 9)]
Selected (2, 9); revealed safe (1), 1 safe neighbors, 0 mines, 4 hidden neighbors, probably safe
Safe cells: [(3, 7), (3, 8), (2, 8)]
Selected (2, 8); revealed safe (1), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Safe cells: [(3, 7), (3, 8)]
Selected (3, 8); revealed safe (0), 6 safe neighbors, 0 mines, 2 hidden neighbors, unknown
Safe cells: [(3, 7), (2, 7)]
Selected (2, 7); revealed safe (2), 2 safe neighbors, 0 mines, 6 hidden neighbors, probably safe
Safe cells: [(3, 7)]
Selected (3, 7); revealed safe (0), 5 safe neighbors, 0 mines, 3 hidden neighbors, unknown
Safe cells: [(2, 6), (3, 6), (4, 6)]
Selected (4, 6); revealed safe (1), 3 safe neighbors, 0 mines, 5 hidden neighbors, probably safe
Safe cells: [(2, 6), (3, 6)]
Selected (3, 6); revealed safe (0), 4 safe neighbors, 0 mines, 4 hidden neighbors, unknown
Safe cells: [(2, 6), (2, 5), (3, 5), (4, 5)]
Selected (4, 5); revealed safe (1), 2 safe neighbors, 0 mines, 6 hidden neighbors, probably safe
Safe cells: [(2, 6), (2, 5), (3, 5)]
Selected (3, 5); revealed safe (0), 3 safe neighbors, 0 mines, 5 hidden neighbors, unknown
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (4, 4)]
Selected (4, 4); revealed safe (0), 2 safe neighbors, 0 mines, 6 hidden neighbors, probably safe
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (4, 3), (5, 3), (5, 4), (5, 5)]
Selected (5, 5); revealed safe (1), 3 safe neighbors, 0 mines, 5 hidden neighbors, probably safe
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (4, 3), (5, 3), (5, 4)]
Selected (5, 4); revealed safe (1), 3 safe neighbors, 0 mines, 5 hidden neighbors, unknown

Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (4, 3), (5, 3)]
Selected (5, 3); revealed safe (1), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (4, 3)]
Selected (4, 3); revealed safe (0), 3 safe neighbors, 0 mines, 5 hidden neighbors, unknown
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (3, 2), (4, 2), (5, 2)]
Selected (5, 2); revealed safe (1), 2 safe neighbors, 0 mines, 6 hidden neighbors, probably safe
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (3, 2), (4, 2)]
Selected (4, 2); revealed safe (0), 3 safe neighbors, 0 mines, 5 hidden neighbors, unknown
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (3, 2), (3, 1), (4, 1), (5, 1)]
Selected (5, 1); revealed safe (0), 2 safe neighbors, 0 mines, 6 hidden neighbors, probably safe
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0), (5, 0), (6, 0), (6, 1), (6, 2)]
Selected (6, 2); revealed safe (2), 3 safe neighbors, 0 mines, 5 hidden neighbors, probably safe
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0), (5, 0), (6, 0), (6, 1)]
Selected (6, 1); revealed safe (0), 4 safe neighbors, 0 mines, 4 hidden neighbors, unknown
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0), (5, 0), (6, 0), (7, 1), (7, 2)]
Selected (7, 2); revealed safe (4), 2 safe neighbors, 0 mines, 6 hidden neighbors, probably safe
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0), (5, 0), (6, 0), (7, 1)]
Selected (7, 1); revealed safe (2), 4 safe neighbors, 0 mines, 4 hidden neighbors, unknown
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0), (5, 0), (6, 0)]
Selected (6, 0); revealed safe (0), 4 safe neighbors, 0 mines, 1 hidden neighbors, unknown
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0), (5, 0)]
Selected (5, 0); revealed safe (0), 3 safe neighbors, 0 mines, 2 hidden neighbors, probably safe
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0)]
Selected (4, 0); revealed safe (1), 2 safe neighbors, 0 mines, 3 hidden neighbors, probably safe
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (3, 2), (3, 1), (4, 1)]
Selected (4, 1); revealed safe (1), 5 safe neighbors, 0 mines, 3 hidden neighbors, unknown
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (3, 2), (3, 1)]
Selected (3, 1); revealed safe (2), 3 safe neighbors, 0 mines, 5 hidden neighbors, unknown
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (3, 2)]
Selected (3, 2); revealed safe (0), 4 safe neighbors, 0 mines, 4 hidden neighbors, unknown
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (2, 1), (2, 2), (2, 3)]
Selected (2, 3); revealed safe (0), 1 safe neighbors, 0 mines, 7 hidden neighbors, probably safe

Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (2, 1), (2, 2), (1, 2), (1, 3), (1, 4)]
Selected (1, 4); revealed safe (1), 1 safe neighbors, 0 mines, 7 hidden neighbors, probably safe
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (2, 1), (2, 2), (1, 2), (1, 3)]
Selected (1, 3); revealed safe (1), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (2, 1), (2, 2), (1, 2)]
Selected (1, 2); revealed safe (0), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (2, 1), (2, 2), (0, 1), (0, 2), (0, 3), (1, 1)]
Selected (1, 1); revealed safe (3), 1 safe neighbors, 0 mines, 7 hidden neighbors, probably safe
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (2, 1), (2, 2), (0, 1), (0, 2), (0, 3)]
Selected (0, 3); revealed safe (1), 3 safe neighbors, 0 mines, 2 hidden neighbors, probably safe
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (2, 1), (2, 2), (0, 1), (0, 2)]
Selected (0, 2); revealed safe (0), 4 safe neighbors, 0 mines, 1 hidden neighbors, unknown
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (2, 1), (2, 2), (0, 1)]
Selected (0, 1); revealed safe (2), 3 safe neighbors, 0 mines, 2 hidden neighbors, probably safe
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (2, 1), (2, 2)]
Selected (2, 2); revealed safe (0), 6 safe neighbors, 0 mines, 2 hidden neighbors, unknown
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (2, 1)]
Selected (2, 1); revealed safe (3), 5 safe neighbors, 0 mines, 3 hidden neighbors, probably safe
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4), (3, 3)]
Selected (3, 3); revealed safe (0), 6 safe neighbors, 0 mines, 2 hidden neighbors, probably safe
Safe cells: [(2, 6), (2, 5), (2, 4), (3, 4)]
Selected (3, 4); revealed safe (0), 6 safe neighbors, 0 mines, 2 hidden neighbors, probably safe
Safe cells: [(2, 6), (2, 5), (2, 4)]
Selected (2, 4); revealed safe (0), 6 safe neighbors, 0 mines, 2 hidden neighbors, probably safe
Safe cells: [(2, 6), (2, 5), (1, 5)]
Selected (1, 5); revealed safe (3), 2 safe neighbors, 0 mines, 6 hidden neighbors, probably safe
Safe cells: [(2, 6), (2, 5)]
Selected (2, 5); revealed safe (1), 6 safe neighbors, 0 mines, 2 hidden neighbors, unknown
Safe cells: [(2, 6)]
Selected (2, 6); revealed safe (1), 6 safe neighbors, 0 mines, 2 hidden neighbors, unknown
Selected (0, 8); revealed safe (1), 0 safe neighbors, 0 mines, 5 hidden neighbors, unknown
Selected (6, 5); revealed safe (1), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Selected (8, 7); revealed safe (2), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Selected (1, 7); revealed safe (3), 4 safe neighbors, 0 mines, 4 hidden neighbors, unknown

Selected (0, 9); revealed safe (1), 1 safe neighbors, 0 mines, 2 hidden neighbors, unknown
Selected (9, 1); revealed safe (2), 0 safe neighbors, 0 mines, 5 hidden neighbors, unknown
Selected (7, 4); revealed safe (3), 1 safe neighbors, 0 mines, 7 hidden neighbors, unknown
Selected (9, 0); revealed safe (1), 1 safe neighbors, 0 mines, 2 hidden neighbors, unknown
Selected (8, 8); revealed mine, 4 safe neighbors, 0 mines, 4 hidden neighbors, unknown
Selected (8, 0); revealed safe (1), 4 safe neighbors, 0 mines, 1 hidden neighbors, unknown
Selected (9, 4); revealed safe (1), 0 safe neighbors, 0 mines, 5 hidden neighbors, unknown
Selected (9, 6); revealed safe (2), 1 safe neighbors, 0 mines, 4 hidden neighbors, unknown
Selected (1, 6); revealed mine, 5 safe neighbors, 0 mines, 3 hidden neighbors, unknown
Selected (6, 6); revealed safe (1), 5 safe neighbors, 0 mines, 3 hidden neighbors, unknown
Selected (8, 4); revealed safe (2), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Selected (8, 5); revealed mine, 4 safe neighbors, 0 mines, 4 hidden neighbors, unknown
Selected (0, 6); revealed mine, 2 safe neighbors, 1 mines, 2 hidden neighbors, unknown
Selected (0, 4); revealed mine, 4 safe neighbors, 0 mines, 1 hidden neighbors, unknown
Selected (0, 5); revealed safe (3), 2 safe neighbors, 3 mines, 0 hidden neighbors, unknown
Selected (1, 9); revealed safe (1), 4 safe neighbors, 0 mines, 1 hidden neighbors, unknown
Selected (9, 2); revealed safe (2), 1 safe neighbors, 0 mines, 4 hidden neighbors, unknown
Selected (5, 6); revealed mine, 8 safe neighbors, 0 mines, 0 hidden neighbors, unknown
Selected (6, 3); revealed mine, 6 safe neighbors, 0 mines, 2 hidden neighbors, unknown
Selected (7, 6); revealed safe (2), 5 safe neighbors, 1 mines, 2 hidden neighbors, unknown
Selected (7, 5); revealed safe (2), 5 safe neighbors, 1 mines, 2 hidden neighbors, unknown
Selected (9, 5); revealed safe (2), 3 safe neighbors, 1 mines, 1 hidden neighbors, unknown
Selected (8, 2); revealed mine, 4 safe neighbors, 0 mines, 4 hidden neighbors, unknown
Selected (9, 7); revealed safe (2), 2 safe neighbors, 1 mines, 2 hidden neighbors, unknown
Selected (7, 3); revealed mine, 4 safe neighbors, 2 mines, 2 hidden neighbors, unknown
Selected (0, 7); revealed safe (3), 2 safe neighbors, 2 mines, 1 hidden neighbors, unknown
Selected (9, 3); revealed safe (1), 3 safe neighbors, 1 mines, 1 hidden neighbors, unknown
Safe cells: [(8, 3)]
Selected (8, 3); revealed safe (2), 6 safe neighbors, 2 mines, 0 hidden neighbors, probably safe
Selected (9, 9); revealed safe (1), 0 safe neighbors, 1 mines, 2 hidden neighbors, unknown
Safe cells: [(8, 9), (9, 8)]
Selected (9, 8); revealed safe (1), 3 safe neighbors, 1 mines, 1 hidden neighbors, probably safe
Safe cells: [(8, 9)]
Selected (8, 9); revealed safe (1), 4 safe neighbors, 1 mines, 0 hidden neighbors, probably safe
Selected (6, 4); revealed safe (2), 6 safe neighbors, 2 mines, 0 hidden neighbors, unknown
Mine cells: [(0, 0), (1, 0), (2, 0), (3, 0), (8, 1), (1, 8), (8, 6)]
The basic agent scored: 7 / 16

Advanced Agent Play-By-Play:
Selected (0, 6); revealed mine, 0 safe neighbors, 0 mines, 5 hidden neighbors, unknown

Selected (2, 5); revealed safe (1), 0 safe neighbors, 0 mines, 8 hidden neighbors, unknown
Selected (1, 4); revealed safe (1), 1 safe neighbors, 0 mines, 7 hidden neighbors, unknown
Selected (0, 3); revealed safe (1), 1 safe neighbors, 0 mines, 4 hidden neighbors, unknown
Selected (1, 6); revealed mine, 1 safe neighbors, 1 mines, 6 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (3, 6)]
Selected (3, 6); revealed safe (0), 1 safe neighbors, 0 mines, 7 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (3, 7), (4, 5), (4, 6), (4, 7)]
Selected (4, 7); revealed safe (1), 1 safe neighbors, 0 mines, 7 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (3, 7), (4, 5), (4, 6)]
Selected (4, 6); revealed safe (1), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (3, 7), (4, 5)]
Selected (4, 5); revealed safe (1), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (3, 7)]
Selected (3, 7); revealed safe (0), 3 safe neighbors, 0 mines, 5 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (4, 8)]
Selected (4, 8); revealed safe (0), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (5, 9)]
Selected (5, 9); revealed safe (0), 1 safe neighbors, 0 mines, 4 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (4, 4), (5, 4), (5, 5)]
Selected (5, 5); revealed safe (1), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (4, 4), (5, 4)]
Selected (5, 4); revealed safe (1), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (4, 4)]
Selected (4, 4); revealed safe (0), 3 safe neighbors, 0 mines, 5 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (4, 3), (5, 3)]
Selected (5, 3); revealed safe (1), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (4, 3)]
Selected (4, 3); revealed safe (0), 3 safe neighbors, 0 mines, 5 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (3, 2), (4, 2), (5, 2)]
Selected (5, 2); revealed safe (1), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (3, 2), (4, 2)]
Selected (4, 2); revealed safe (0), 3 safe neighbors, 0 mines, 5 hidden neighbors, unknown

Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (3, 2), (3, 1), (4, 1), (5, 1)]
Selected (5, 1); revealed safe (0), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0), (5, 0), (6, 0), (6, 1), (6, 2), (6, 4), (6, 5)]
Selected (6, 5); revealed safe (1), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0), (5, 0), (6, 0), (6, 1), (6, 2), (6, 4)]
Selected (6, 4); revealed safe (2), 4 safe neighbors, 0 mines, 4 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0), (5, 0), (6, 0), (6, 1), (6, 2)]
Selected (6, 2); revealed safe (2), 3 safe neighbors, 0 mines, 5 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0), (5, 0), (6, 0), (6, 1)]
Selected (6, 1); revealed safe (0), 3 safe neighbors, 0 mines, 5 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0), (5, 0), (6, 0), (7, 0), (7, 1), (7, 2)]
Selected (7, 2); revealed safe (4), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0), (5, 0), (6, 0), (7, 0), (7, 1), (7, 4), (7, 5)]
Selected (7, 5); revealed safe (2), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0), (5, 0), (6, 0), (7, 0), (7, 1), (7, 4)]
Selected (7, 4); revealed safe (3), 3 safe neighbors, 0 mines, 5 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0), (5, 0), (6, 0), (7, 0), (7, 1)]
Selected (7, 1); revealed safe (2), 3 safe neighbors, 0 mines, 5 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0), (5, 0), (6, 0), (7, 0)]
Selected (7, 0); revealed safe (1), 2 safe neighbors, 0 mines, 3 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0), (5, 0), (6, 0), (8, 0)]
Selected (8, 0); revealed safe (1), 2 safe neighbors, 0 mines, 3 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0), (5, 0), (6, 0)]
Selected (6, 0); revealed safe (0), 4 safe neighbors, 0 mines, 1 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0), (5, 0)]
Selected (5, 0); revealed safe (0), 3 safe neighbors, 0 mines, 2 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7),

(5, 8), (6, 8), (6, 9), (3, 3), (3, 2), (3, 1), (4, 1), (4, 0)]
Selected (4, 0); revealed safe (1), 2 safe neighbors, 0 mines, 3 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (3, 2), (3, 1), (4, 1)]
Selected (4, 1); revealed safe (1), 5 safe neighbors, 0 mines, 3 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (3, 2), (3, 1)]
Selected (3, 1); revealed safe (2), 3 safe neighbors, 0 mines, 5 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (3, 2)]
Selected (3, 2); revealed safe (0), 4 safe neighbors, 0 mines, 4 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (2, 1), (2, 2), (2, 3)]
Selected (2, 3); revealed safe (0), 2 safe neighbors, 0 mines, 6 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (2, 1), (2, 2), (1, 2), (1, 3)]
Selected (1, 3); revealed safe (1), 3 safe neighbors, 0 mines, 5 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (2, 1), (2, 2), (1, 2), (0, 2), (0, 5)]
Selected (0, 5); revealed safe (3), 1 safe neighbors, 2 mines, 2 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (2, 1), (2, 2), (1, 2), (0, 2)]
Selected (0, 2); revealed safe (0), 2 safe neighbors, 0 mines, 3 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (2, 1), (2, 2), (1, 2), (0, 1), (1, 1)]
Selected (1, 1); revealed safe (3), 1 safe neighbors, 0 mines, 7 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (2, 1), (2, 2), (1, 2), (0, 1)]
Selected (0, 1); revealed safe (2), 2 safe neighbors, 0 mines, 3 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (2, 1), (2, 2), (1, 2)]
Selected (1, 2); revealed safe (0), 6 safe neighbors, 0 mines, 2 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (2, 1), (2, 2)]
Selected (2, 2); revealed safe (0), 6 safe neighbors, 0 mines, 2 hidden neighbors, probably safe

Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3), (2, 1)]
Selected (2, 1); revealed safe (3), 5 safe neighbors, 0 mines, 3 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9), (3, 3)]
Selected (3, 3); revealed safe (0), 6 safe neighbors, 0 mines, 2 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (6, 9)]
Selected (6, 9); revealed safe (0), 1 safe neighbors, 0 mines, 4 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (7, 8), (7, 9)]
Selected (7, 9); revealed safe (1), 1 safe neighbors, 0 mines, 4 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8), (7, 8)]
Selected (7, 8); revealed safe (1), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 8)]
Selected (6, 8); revealed safe (0), 4 safe neighbors, 0 mines, 4 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 7), (7, 7)]
Selected (7, 7); revealed safe (2), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (6, 7)]
Selected (6, 7); revealed safe (1), 3 safe neighbors, 0 mines, 5 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8), (8, 9)]
Selected (8, 9); revealed safe (1), 2 safe neighbors, 0 mines, 3 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7), (5, 8)]
Selected (5, 8); revealed safe (0), 6 safe neighbors, 0 mines, 2 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9), (5, 7)]
Selected (5, 7); revealed safe (1), 6 safe neighbors, 0 mines, 2 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9), (4, 9)]
Selected (4, 9); revealed safe (0), 3 safe neighbors, 0 mines, 2 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (3, 9)]
Selected (3, 9); revealed safe (0), 2 safe neighbors, 0 mines, 3 hidden neighbors, probably

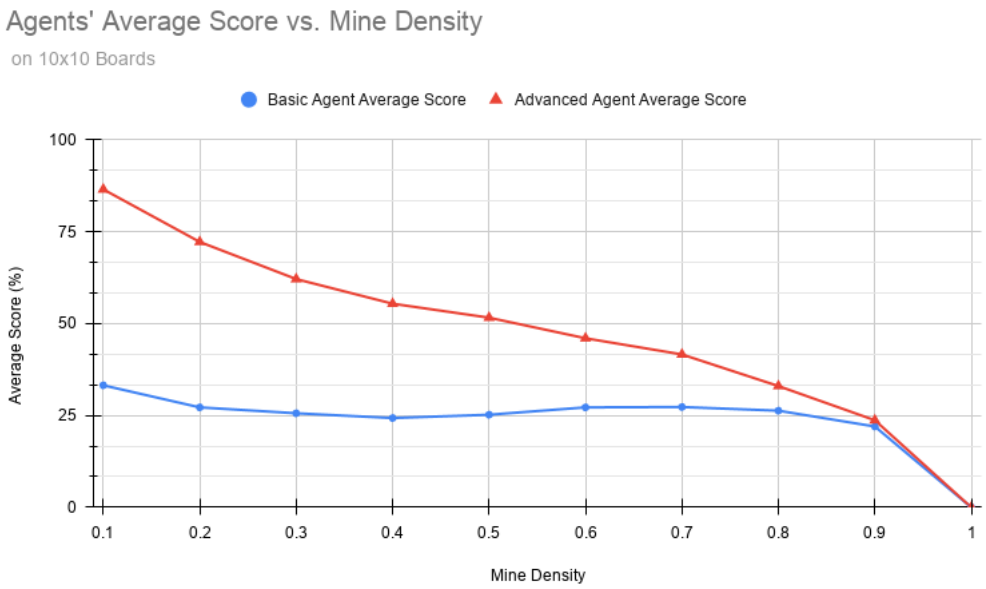
safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8), (2, 9)]
Selected (2, 9); revealed safe (1), 1 safe neighbors, 0 mines, 4 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8), (3, 8)]
Selected (3, 8); revealed safe (0), 6 safe neighbors, 0 mines, 2 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7), (2, 8)]
Selected (2, 8); revealed safe (1), 4 safe neighbors, 0 mines, 4 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5), (2, 7)]
Selected (2, 7); revealed safe (2), 4 safe neighbors, 1 mines, 3 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (3, 5)]
Selected (3, 5); revealed safe (0), 5 safe neighbors, 0 mines, 3 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (1, 7), (1, 9)]
Selected (1, 9); revealed safe (1), 2 safe neighbors, 0 mines, 3 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4), (1, 7)]
Selected (1, 7); revealed safe (3), 2 safe neighbors, 2 mines, 4 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6), (3, 4)]
Selected (3, 4); revealed safe (0), 7 safe neighbors, 0 mines, 1 hidden neighbors, unknown
Safe cells: [(1, 5), (2, 4), (2, 6)]
Selected (2, 6); revealed safe (1), 6 safe neighbors, 1 mines, 1 hidden neighbors, probably safe
Safe cells: [(1, 5), (2, 4)]
Selected (2, 4); revealed safe (0), 7 safe neighbors, 0 mines, 1 hidden neighbors, probably safe
Safe cells: [(1, 5)]
Selected (1, 5); revealed safe (3), 5 safe neighbors, 2 mines, 1 hidden neighbors, probably safe
Selected (8, 7); revealed safe (2), 2 safe neighbors, 0 mines, 6 hidden neighbors, unknown
Selected (9, 6); revealed safe (2), 1 safe neighbors, 0 mines, 4 hidden neighbors, unknown
Selected (9, 8); revealed safe (1), 2 safe neighbors, 0 mines, 3 hidden neighbors, unknown
Selected (6, 6); revealed safe (1), 6 safe neighbors, 0 mines, 2 hidden neighbors, unknown
Selected (7, 6); revealed safe (2), 6 safe neighbors, 0 mines, 2 hidden neighbors, unknown
Safe cells: [(8, 4), (9, 5), (9, 7)]
Selected (9, 7); revealed safe (2), 3 safe neighbors, 0 mines, 2 hidden neighbors, unknown
Safe cells: [(8, 4), (9, 5)]
Selected (9, 5); revealed safe (2), 1 safe neighbors, 0 mines, 4 hidden neighbors, unknown
Safe cells: [(8, 4)]
Selected (8, 4); revealed safe (2), 3 safe neighbors, 0 mines, 5 hidden neighbors, unknown
Selected (9, 0); revealed safe (1), 1 safe neighbors, 0 mines, 2 hidden neighbors, unknown
Selected (9, 4); revealed safe (1), 2 safe neighbors, 0 mines, 3 hidden neighbors, unknown

Selected (0, 8); revealed safe (1), 2 safe neighbors, 0 mines, 3 hidden neighbors, unknown
Selected (9, 9); revealed safe (1), 2 safe neighbors, 0 mines, 1 hidden neighbors, unknown
Selected (9, 2); revealed safe (2), 0 safe neighbors, 0 mines, 5 hidden neighbors, unknown
Selected (9, 3); revealed safe (1), 3 safe neighbors, 0 mines, 2 hidden neighbors, unknown
Selected (0, 7); revealed safe (3), 2 safe neighbors, 2 mines, 1 hidden neighbors, unknown
Selected (9, 1); revealed safe (2), 3 safe neighbors, 0 mines, 2 hidden neighbors, unknown
Selected (0, 9); revealed safe (1), 2 safe neighbors, 0 mines, 1 hidden neighbors, unknown
Mine cells: [(5, 6), (6, 3), (7, 3), (8, 3), (8, 1), (8, 2), (3, 0), (2, 0), (0, 4), (0, 0), (1, 0), (8, 8), (1, 8), (8, 6), (8, 5)]
The advanced agent scored: 14 / 16

From time to time, our advanced agent may unexpectedly select a cell and mark it as a mine, even if that cell is actually safe, thus resulting in a false positive. In our implementation, we make a note of the number of false positives found, and the program moves on. Apart from the false positives, the program makes selection decisions as intended.

2.5 Performance (cont.)

For a fixed, reasonable size of board, plot as a function of mine density the average final score (safely identified mines/total mines) for the simple baseline algorithm and your algorithm for comparison. This will require solving multiple random boards at a given density of mines to get good average score results. Does the graph make sense / agree with your intuition? When does minesweeper become "hard?" When does your algorithm beat the simple algorithm, and when is the simple algorithm better? Why? How frequently is your algorithm able to work out things that the basic agent cannot?



This graph agrees with what we expect. When the mine density is fairly low, the strong inferences that the advanced agent makes allows it to sidestep mines easily, which leads to a score higher than the basic agent. On the other hand, as the mine density increases, the advanced agent becomes on par with the basic agent because even though the advanced agent can make stronger inferences and use probability to its advantage, at the end of the day there are only so many places that it can step to, meaning that it will have as many choices to step towards as the basic agent. Judging by the difference in the two trends in their early stages, the advanced agent is able to outperform the basic agent, and seems to have a slight edge even at higher mine densities. According to the trends, the basic agent never outperforms our algorithm, but can potentially be on par.

In essence, Minesweeper becomes "hard" once there are a lot more mines put on the board. Both the basic and advanced agent will struggle in these environments, as both will inevitably tread over a greater number of mines. The more of them there are, the less safe spots there are to find and make inferences off of.

2.6 Efficiency

What are some of the space or time constraints you run into in implementing this program? Are these problem specific constraints? In the case of implementation constraints, what could you improve upon?

Our worst case running time for our advanced agent is essentially $O(n^2)$ because of

the fact that we are working with the board in a two-dimensional structure, and there can potentially be a scenario in which the agent will have to visit all cells. In addition, since we are using arrays to hold references for the cells to avoid and select, it can potentially become a significant space constraint if the board is very large, as it could potentially grow to hold a large number of mine references.

3 Bonus Questions

3.1 Global Information

Suppose you were told in advance how many mines are on the board. Include this in your knowledge base. How did you model this? Regenerate the plot of mine density vs. average final score with this extra information, and analyze the results.

Given the number of mines left on the board, we could better calculate the risk of each cell being a mine by accounting for every single cell from the very start and update the odds as we mark (or trigger) mines until the very end. We would also know when we have already found all of the mines by tracking the number of mines left on the board.

3.2 Better Decisions

In both the basic and improved agent, when nothing more could be inferred, the agent selects a covered cell at random. Build a better mechanism. How can you justify it? Regenerate the plot of mine density vs. average final score with improved cell selection, and analyze the results.

In addition to selecting low risk cells using probability, we could try and prioritize selecting cells on the inside first, then cells on the edges, then cells in the corner. The three types of cells have eight neighbors, five neighbors, then three neighbors respectively, and the lower the number of neighbors, the higher the odds of tripping a mine and we get less information about safe cells as well. Corners are high risk, low reward compared to inner cells. The more information we can obtain, the faster we can "expand" the frontier before we have to resort to randomness.

Overall, it does not seem to affect the score by too much; sometimes better, sometimes worse, but still very much random as the score ranges around what the original cell selection heuristic was, which was pretty much random.