Minesweeper Final AI Report

Team name: GukGukAI

Member #1 (name/id) Andrew Tian / 66931790

Member #2 (name/id) Raymond Anggono / 93379391

I. Minimal AI

I.A. Briefly describe your Minimal AI algorithm. What did you do that was fun, clever, or creative?

For the minimal AI, we just applied the "rule of thumb". The algorithm is effective to solve all the super easy boards (5 x 5, 1 mines) and partial easy (8 x 8, 10 mines) and intermediate (16 x 16, 40 mines) boards. The algorithm is risk-free since it only opens a surrounding tile if all possible mines in the neighbor have been marked. For example, if our current tile label is 2 and it has 2 marked neighbors then its remaining neighbors are mine free. On the other hand, we flag the surrounding tile if the number of remaining neighbor cover tiles is equal to the current label number. For instance, if our current label number is 2 and it has 2 remaining cover tiles in its surrounding, then all cover tiles neighbors must be mines. Using the checkNeighbor() and markNeighbor() function, the algorithm decides to flag or open the tiles. If the algorithm decides to open a tile, the algorithm will store the next row and column coordinate to the queue called nextTiles. Otherwise, if the algorithm decides to flag the tile, it will mark the tile with -2 and record it to the persistent board.

I.B Describe your Minimal AI algorithm's performance:

Board Size	Sample Size	Score	Worlds Complete
5x5	100	100	100

8x8	100	30	30
16x16	100	60	30
16x30	0	0	0
Total Summary	300	190	160

II. Final AI

II.A. Briefly describe your Final AI algorithm, focusing mainly on the changes since Minimal AI:

Since Minimal AI only opens and flags tiles based on the rules of thumb, it does not have an ability to open a frontier tile. Therefore, we must create another algorithm that can open and flag a tile if our minimal algorithm has stopped working. The algorithm's idea is to enumerate all possible configurations of mines in the frontier tiles. Using the getFrontierTiles() function, we return the frontier tiles that are adjacent with uncover tiles. Since our algorithm is designated to enumerate all the possible configuration of mines in all the frontier tiles, it will take a lot of time. In order to reduce the runtime, we create a helper function called getGroupedFrontierTiles() to group frontier tiles into a few sections. Then, after we group the frontier tiles into a few sections, we are going to find all possible mine positions in each section respectively using the function called findAllPossibleMine(). Finally, the solver() function will calculate the probability of mine location in each section of frontier tile and compare it with the random guessing. Our algorithm solves 60% of easy board, 50% of intermediate board, and 1% of expert board.

II.B Describe your Final AI algorithm's performance:

Board Size	Sample Size	Score	Worlds Complete
5x5	100	100	100
8x8	100	60	60
16x16	100	100	50
16x30	100	3	1
Total Summary	400	263	211

III. In about 1/4 page of text or less, provide suggestions for improving this project (*this section does <u>NOT</u> count as past of your two-page total limit.*)

Our algorithm has a bad running time because it brute-force all the possible mines in every section. Not only does it have a bad running time, our algorithm has to guess the position of the mines and it will never be as precise as the CSP method. Therefore, to improve this we need to change the algorithm's logic from brute-force method to CSP method. By adding the CSP method to our algorithm, we can improve our score and world's completion and have a faster running time. In order to implement the CSP method, our algorithm will first run the rule of thumb and if there are no more tiles that the rule of thumb could uncover, we will then use the CSP method where we take the frontier nodes and find the possibilities of a mine. However, if there are no more tiles that we can uncover, we will then have to use the brute force method that we implemented and guess which tile will have the least possibility of being a mine.