Project Report
Assignment 2
Minesweeper
CS 520

Anirudh Bandi (ab1721) , Shruthi Ramesh (sr1296) ,
Mudita Tandon (mt909) , Nikesh Reddy (nrs113)

Representation:

The board configuration is represented by a 2D array holding the value 9 in each cell representing that the cell is not yet explored. For the first time computer guesses a random location. If for any computer query the user enters -2(which is the number that represents a bomb chosen arbitrarily) the game stops. Any value that the user entered if not -2 is updated in the corresponding location (9 is replaced with that value).

Inference:

The main loop runs until all the locations on the board are cleared. After the first time the computer will query locations which have less probability of being a bomb location among the ones available. If the location guessed is known to be have no bombs around it then all the neighbouring cells around it are explored. If a cell with no bombs is present in the neighborhood of the previous location then the locations around it are also explored recursively. All the locations which are already explored(except locations which don’t have a bomb around them) and not a bomb are checked to see if there any cells that can be unlocked or determined as a bomb.

If the value of the cell is equal to the number of unexplored cells around it then all are considered to be bombs. Similarly if the number of bombs around a cell is equal to the cell value then all the unexplored cells around it be determined as safe. The neighbors of the changed locations mentioned in the previous sentence are tested to see if the change affected their positions causing some cells to be unlocked.

If there aren’t any cells that can be queried directly then the locations with are at the intersection of the neighborhood of two cells are considered and based on the difference between the values of the 2 cells under consideration cells are either cleared or declared as bombs

For example:

		1
A	B	C
	2	

Around the cell 2 , A + B + C = 2 and for the cell 1, B +C = 1 solving the two equations will given that A = 1 i.e there is a bomb in location A. This is implemented using sets in the code, by finding the intersection of neighbors which gives locations B and C between cells 1 and 2.

Subtracting the sets for 1 and 2 gives (A,B,C) - (B,C) = 2-1 = 1.

There is no explicit knowledge base used in the code except the neighborhood of a given location is checked to draw inferences about that location. As more and more cells are unlocked the possible values for locations will decrease.

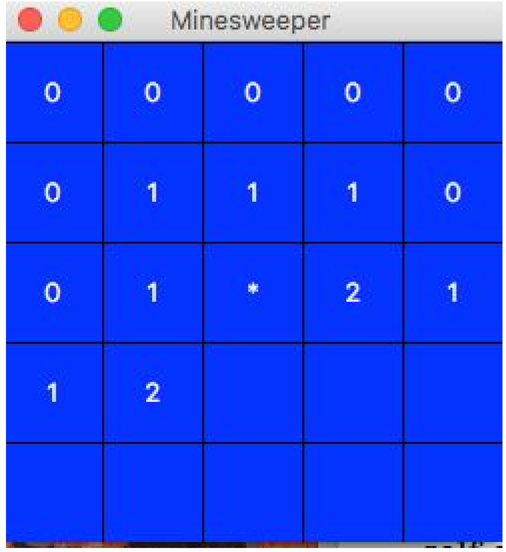
Decisions:

The code will scan through all the locations on the board that aren't completely explored and the cells unlocked in the method mentioned in the previous section. If the system can't find any cells to unlock it'll select a location that has a smallest probability of being a bomb among all the available locations by observing their neighbors.

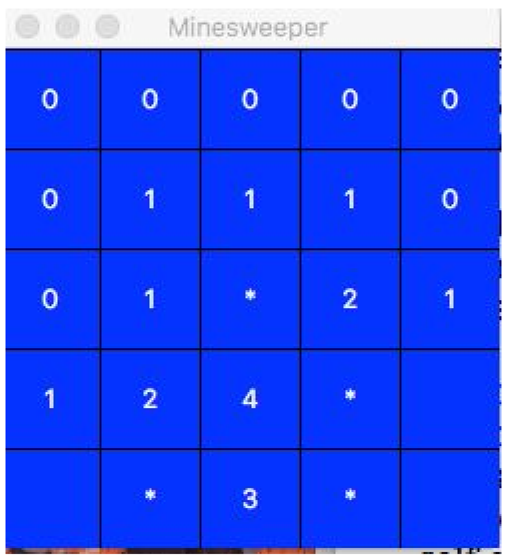
This process takes place until the computer queries a bomb location or it completely finds the location of all bombs

In the initial steps of large boards, the guesses based on probability have a high chance of landing on bomb locations but once a reasonable amount of locations are unlocked the code almost always end in victory. The boards used here are generated in a straightforward way and hence can cause the system to guess a bomb location in the first guess. Most of the failures seen are mostly in first few guesses.

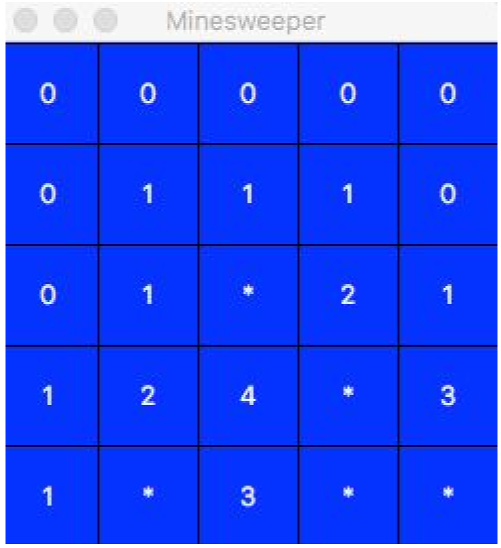
Initial random guess was (0,1) which unlocked all the cells shown here and since neighbours of cell at location (1,2) are all explored except one at location (2,2) and the cell value was 1 location (2,2) is declared as a bomb

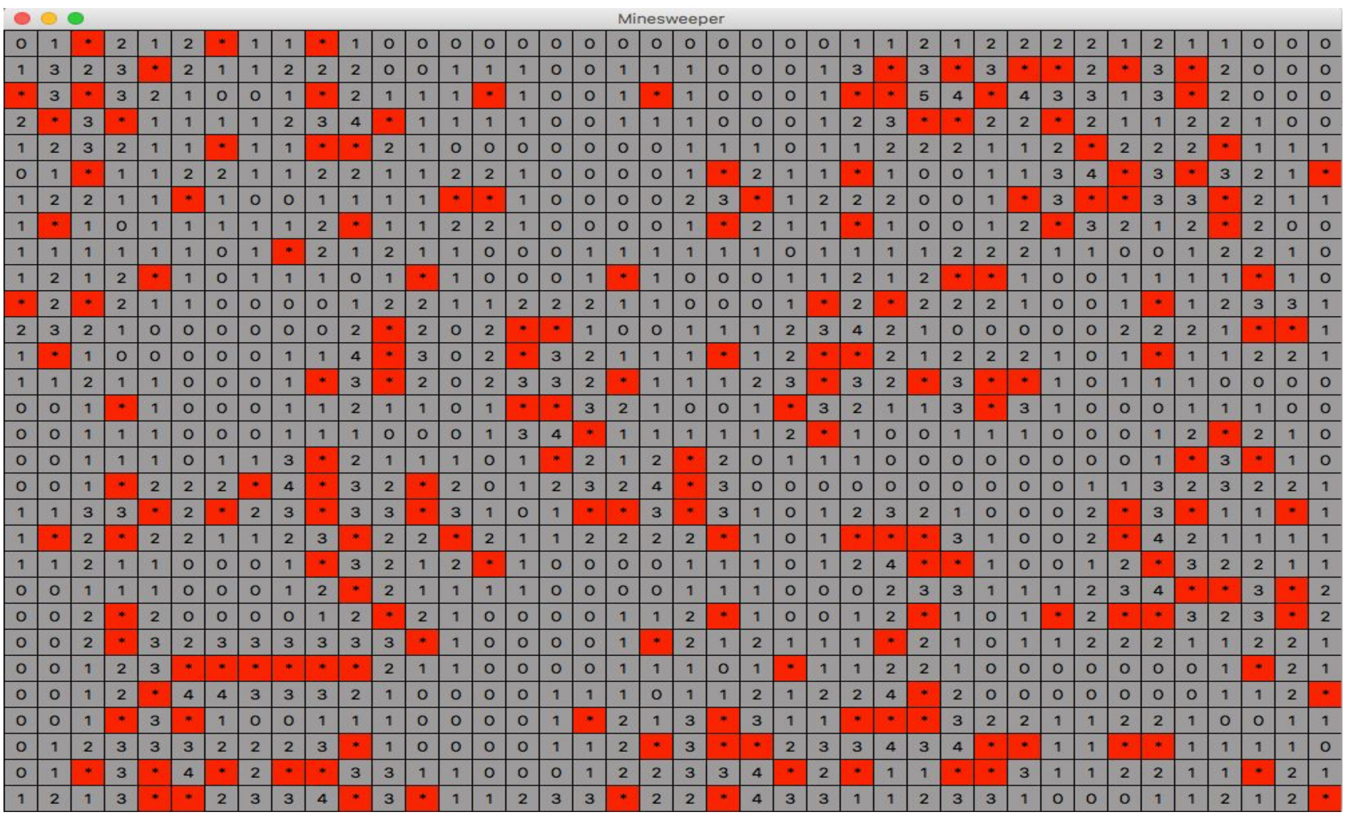


The locations around (2,1) must have one bomb and it is found at (2,2) so the unexplored location at (3,2) must be safe. Writing the equations for locations (3,0) and (3,1) (A+B = 1 , A+B+C = 1 , gives C = 0) C is the location (4,2) which is safe. Around the location (3,2) there should be 4 bombs and 1 is already located at (2,2) and there are three unexplored locations which should be bomb locations.

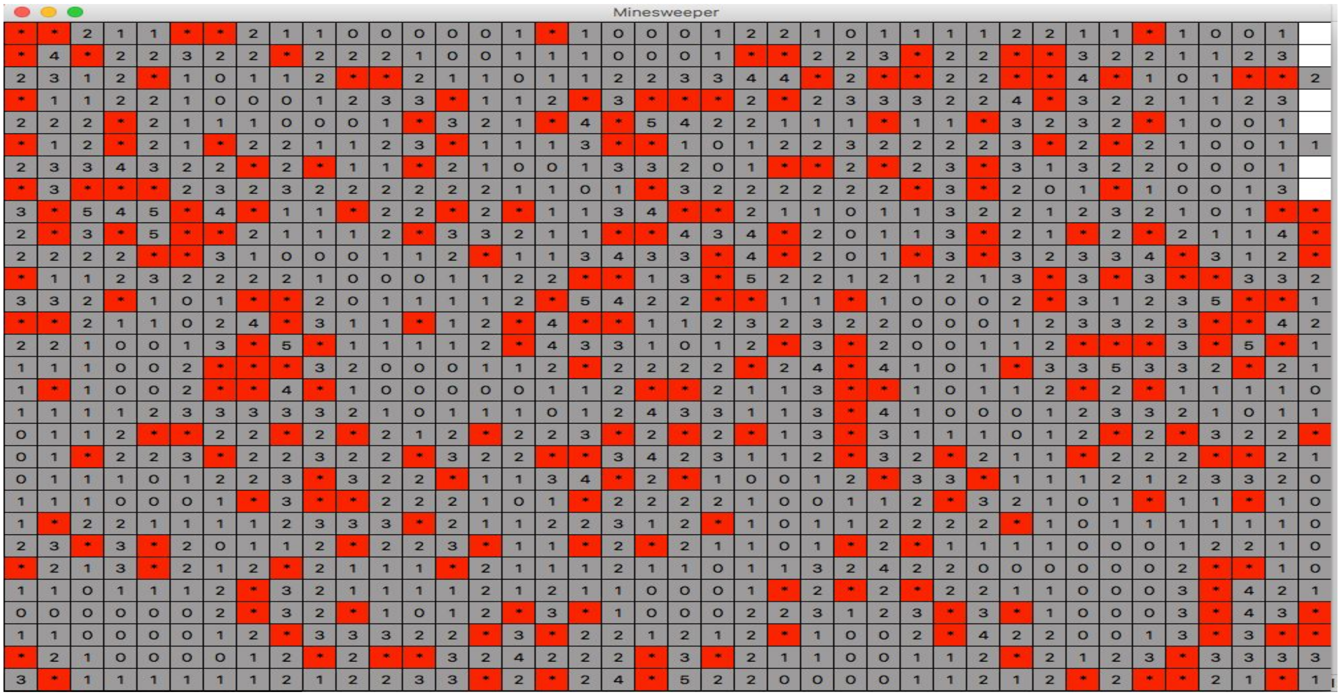


The location (4,0) is cleared as the location (3,0) should have only have 1 bomb around it which is already found at (4,1). Similarly location (3,4) is cleared by using the location (2,4) and based on the location (3,4) which has a value of 3 out of which 2 are already found and the only unexplored cell which is (4,4) is declared as a bomb





The code solves a 40x40 maze with 240 mines(15%) fair amount of times.



For a 40X40 board with 320 mines(20%) it halts at a situation a random guess is the only option and fails. It mostly fails when there is guessing involved.

Efficiency :

The code works at a reasonable speed on a 80x80 board but as the size increases the amount of time taken increases exponentially as many checks are done by looping over all the cells which becomes costly and time consuming as the size increases. If targeted checking can be done that can significantly improve the efficiency of the code.

Improvements:

If the number of mines is provided by the user the risk of choosing a mine can be determined for each mine. It'll help in terminating the search earlier once the given number of mines are found.

Bonus

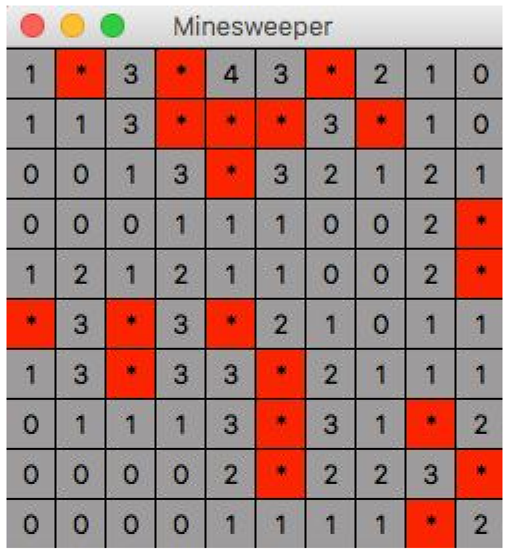
Chain of Influence:

1)When cells are unlocked or declared as mines. The cells in its vicinity are checked to see if it Helps in determining new cells this is done recursively.

2) If there are too many cells with no bombs around them or if the bombs are concentrated in regions the chains of influence are large when compared to dispersed bombs.

3) A board that has long chains of influence can make the job of the solver simpler since only a few random guesses will be enough to unlock a large number of cells due to the chain of influence.

4)



Normally the code almost always fails when the percentage of bombs is around 20%. The given 10x10 board has 20 mines. Since the bombs are placed close together the code was solve the board. These types of boards have long chains of influence hence can be easily solved even when the mines are larger than the limit.

5) is solving minesweeper hard?

Generally speaking since there is randomness involved in solving a minesweeper board. Also the solvability depends on the mine density if it is below a certain level a solution can be found almost always.

Bonus: Dealing with Uncertainty

When the user underestimates the number of mines. Assuming that the misinformation has an underlying probability distribution. Based on it the cell updation can be done. For example if the Probability of the user saying that a cell value is 0 when is there are bombs around it is Xi for i ranging from 0 to 8. We generate a random number between 0 and 1 based on the value we choose the number of bombs that are actually present.

When there occurs a situation where there are conflicting evidence equations. The cell that has majority of the neighbors unlocked can be chosen.

A similar kind of approach can be used for overestimation.

Contribution:
The algorithm and coding was done by the collaborative work of the group.

Citations:

Used for the GUI:
1)https://stackoverflow.com/questions/10020885/creating-a-popup-message-box-with-an-entry-field
2)https://stackoverflow.com/questions/4781184/tkinter-displaying-a-square-grid