Minesweeper construction & solving algorithm (April 2019)

Siddharth Garg (18BCB0038), BTech student, VIT Vellore

***Abstract*—Minesweeper is a single person puzzle implemented as a game. One of the most popular and successful version of the puzzle has been the one which came along Windows Entertainment Pack 1 in 1990.**

The player is initially presented with a grid of undifferentiated squares. Some randomly selected squares, unknown to the player, are designated to contain mines. Typically, the size of the grid and the number of mines are set in advance by the user, either by entering the numbers or selecting from defined skill levels, depending on the implementations.

2	3	*	2	2	*	2	1
*	*	5			4	*	2
	?	*	*	*		4	*
*	6		6	*	*		2
2	*	*		5	5		2
1	3	4		*	*	4	*
0	1	*	4			*	3
0	1	2	*	2	3	*	2

Figure 1: This is a sample minesweeper grid, if all cells of minesweeper grid are revealed.

The game is played by revealing squares of the grid by clicking or otherwise indicating each square. If a square containing a mine is revealed, the player loses the game. If no mine is revealed, a digit is instead displayed in the square, indicating how many adjacent squares contain mines; if no mines are adjacent, the square becomes blank, and all adjacent squares will be recursively revealed. The player uses this information to deduce the contents of other squares, and may either safely reveal each square or mark the square as containing a mine.

In 2000, Richard Kaye published a proof that it is NP-complete to determine whether a given grid of uncovered, correctly flagged, and unknown squares, the labels of the foremost also given, has an arrangement of mines for which it is possible within the rules of the game. The argument is constructive, a method to quickly convert any Boolean circuit into such a grid that is possible if and only if the circuit is satisfiable; membership in NP is established by

using the arrangement of mines as a certificate. If, however, a minesweeper board is already guaranteed to be consistent, solving it is not known to be NP-complete, but it has been proven to be co-NP-complete.

Kaye also proved that infinite minesweeper is Turing complete.

This project contains two modules, the game and the solver. The game implementation presents a command line interface for the user to interact with the puzzle and solve it. The solver serves as a hint for the player solving the puzzle when needed. The solver essentially analyzes the same grid shown to the user and makes an optimum move which reveals a covered tile.

I. The Minesweeper Problem

The Minesweeper is a one-person board game played on a rectangular grid of size k by l. Let such a game be of size n where n = kl. Also choose m, such that m < n, to be the number of hidden mines on the grid. Initially all the squares on the grid are empty, and it is the aim of the game to uncover all the squares that do not contain a mine and mark (or leave blank) all the squares containing a mine. At each move the player must either choose an unlabeled square to 'probe' or mark a square as containing a mine. If the probed square contains a mine the game is over with a loss for the player; otherwise the number of mines immediately adjacent to it is revealed to the player in form of a number (0–8). This number will remain the label of the probed square for the remainder of the game. If the player uncovers the last square not containing a mine the game finishes with a win for the player.

N-consistency

Given an n-dimensional grid partially marked with numbers and/or mines, some squares being blank, determine if there is some pattern of mines in the blank squares that give rise to the numbers seen.

It is intuitive that by adding extra dimensions to the playing area the game becomes harder, so it would be expected that 3-consistency is NP-hard.

In fact, n-consistency is NP-complete for any $n \geq 2$. An interesting situation arises however when reducing the dimension to one, since it is not immediately clear how to construct logic gates as in the 2-dimensional variation.

n-dimensional Minesweeper is NP-complete

n-consistency can be inductively reduced to (n+1) consistency. At a high level, the reduction is to construct a wire, an and gate and a not gate by

'padding' the existing configuration with mines and increasing the labels appropriately to ensure consistency. Finally, the required labels around the outer layer of mines are added.

II. Working of Constructed Minesweeper Game

1) Setting up of grid

The grid-size and number of mines are set before initializing the game. It first creates an empty grid (2D Array) with all elements having NULL value. These represent uncovered tiles. After the setup the program goes into an infinite loop until the user exits the game.

2) Taking input from the user

The user is asked to probe a tile of his/her choice by mentioning the row and column number of the desired tile to be uncovered. This uniquely identifies the user's choice and uncovers the required tile. The input is checked through a regular expression by which if the user does not enter the tile row and column location in an appropriate manner, it asks user to re iterate its choice.

The user is given an option to ask for an optimum hint for the next move and also an option to solve the rest of the puzzle by the solving algorithm itself.

3) Setting mines

While initializing the grid the number of mines are set up obtaining a random cell each time and checking whether it doesn't contain a mine already.

4)Setting number

Once the mines of the board are set the algorithm iterates through every cell and calculates number of mines in the neighboring cells. The value of that cell is set to that number for the algorithm which is not shown to the user unless it probes certain uncovered tiles.

5)Revealing cells on user's probe

The user's choice is analyzed and some tiles are uncovered. If the chosen cell is already shown then the user is asked again for input. For each cell doesn't contain a flag, its neighbors are revealed. This process is continued recursively until each neighboring cell

which doesn't contain a flag is revealed.

III. Working of Constructed Minesweeper Solver

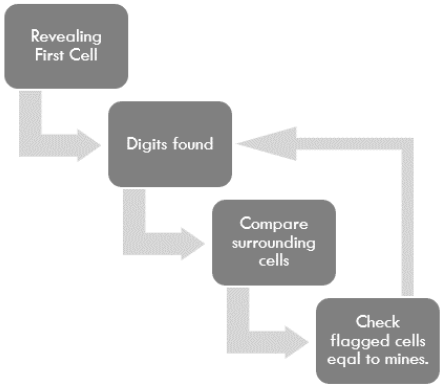


Fig. 2. Flow chart showing approximately the working of the algorithm

1) Checking if the grid is empty

If the grid given is empty then any choice of tile is equally probable. Therefore the first tile of the first column is chosen and revealed.

2)Finding the sure shot cell

The algorithm iterates through each revealed cell and analyzes its neighbors. It uses the following algorithm to determine for sure if the cell is to be flagged or to be revealed:

If the sum of unrevealed neighbors and flagged neighbors is equal to the value of the cell then flag the neighbors.

If the number of flagged neighbors is equal to cell value then reveal the neighbors.

3)The probabilistic approach using minimum cell ratio

While iterating through each revealed cell if there is no sure shot neighbor calculated then there is a cell calculated which has the minimum probability of containing a mine using the following formula

Ratio=(cell value – number of flagged neighbors) /number of unrevealed neighbors

While analyzing each revealed cell if sure-shot method fails then the ratio of that cell's neighbors is set. And at end if no sure shot answer is found then the

IV. Advantages

1) Fast solving algorithm for quicker results

2) Users solving the puzzle are aided by the solving algorithm when they are stuck, by giving them a hint option.

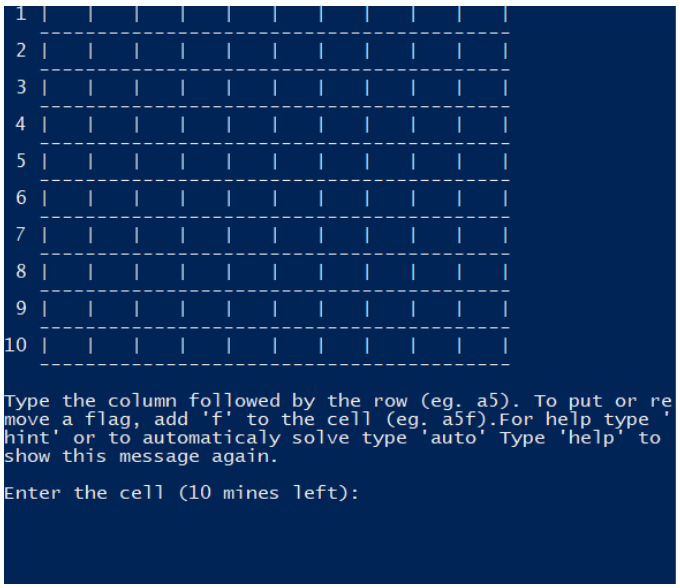
3) Polynomial complexity of $O(n^2)$

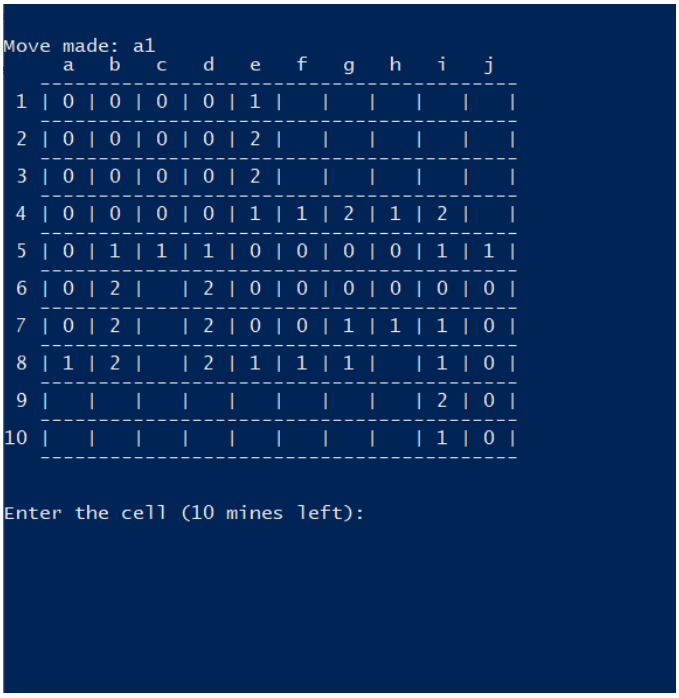
4)Low space complexity.

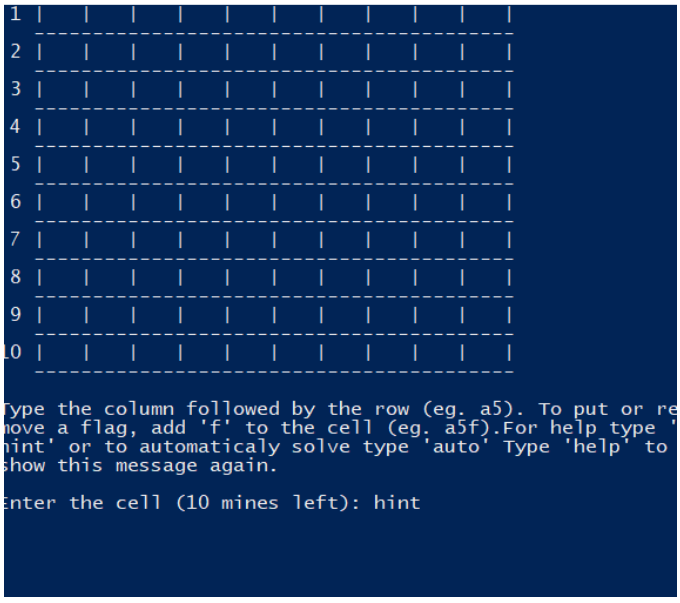
V. Disadvantage

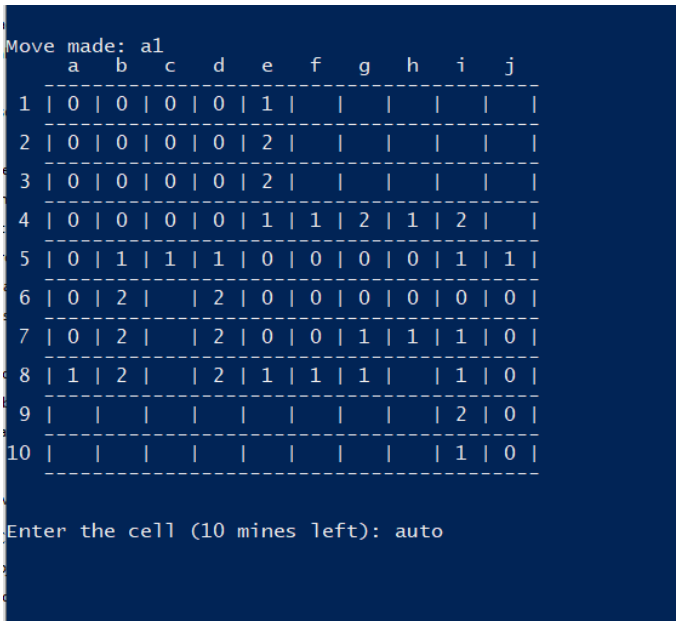
1) Made for regular aid of users trying to solve puzzles, accuracy might decrease for unpractical increase of n.

VI. Sample Test Cases









```
You Win. It took you 1 minutes and 8 seconds.

Move made: h10f
    a   b   c   d   e   f   g   h   i   j
  ---------------------------------------
 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
  ---------------------------------------
 2 | 0 | 0 | 0 | 0 | 2 | x | 3 | 1 | 1 | 0 |
  ---------------------------------------
 3 | 0 | 0 | 0 | 0 | 2 | x | 3 | x | 2 | 1 |
  ---------------------------------------
 4 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | 1 | 2 | x |
  ---------------------------------------
 5 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
  ---------------------------------------
 6 | 0 | 2 | x | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
  ---------------------------------------
 7 | 0 | 2 | x | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
  ---------------------------------------
 8 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | x | 1 | 0 |
  ---------------------------------------
 9 | 1 | x | 1 | 1 | x | 1 | 2 | 2 | 2 | 0 |
  ---------------------------------------
10 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | x | 1 | 0 |
  ---------------------------------------

Play again? (y/n):
```

REFERENCES

[1] RICHARD KAYE, School of mathematics and sciences, The University of Birmingham, Birmingham, B15 2TT. "Some Minesweeper Configurations" Available: http://web.mat.bham.ac.uk/R.W.Kaye/minesw/minesw.pdf

[2] IAN STEWART "Ian Stewart on Minesweeper" Available: https://www.claymath.org/sites/default/files/minesweeper.pdf

AUTHOR

Siddharth Garg (18BCB0038) is a student at Vellore Institute of Technology, currently pursuing 1st year of Bachelors of Technology. This project is created by him as part of his course 'Data Structures & Algorithms' under the guidance of Dr. (Prof.) Boominathan