ASSIGNMENT 2
CS520

Ravi Teja Pothamsetty
Dipayan Dutta
Harsh Rajkumar Vaish
Vinodh Reddy Velemineti

Questions:

1. How did you represent the board in your program, and how did you represent the information / knowledge that clue cells reveal?

Ans: We represented the board in the form of a matrix, where cells with mine is given a value 1 and non-mine cells are assigned 0.

We represent each constraint with the following data structure

```
Constraint{
        List variables;
        Int value;
}
```

We represent the knowledge in the following forms:

- Safe cells list
- Mine cells list
- Probability matrix --- (this matrix is solely sufficient to store the knowledge form but safe cells list and Mine cells list increase the speed of our algorithm)
- List of constraints.

2. When you collect a new clue, how do you model / process / compute the information you gain from it? i.e., 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?

Ans: We have a list of constraints and knowledge of the cells we inferred to be mine or non-mine. Using this knowledge we formulate the new constraint. Our program doesn't deduce everything it can, because doing that is computationally expensive. We do only one level inference which is good enough as multiple levels is consuming too much time. We can increase the number of levels and get better results.

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? Are there any risks, and how do you face them?

Ans: Our program checks the safe cells list, if it is not empty, then those cells are explored next. We infer safe cells by using the knowledge accumulated. If there are no elements in safe cell list, then our program chooses a cell which has minimum probability to be mine (first we consider the cells which are already in the constraints). If there is a tie between cells we choose the cell that is more connected to the knowledge base (most constrained cell).

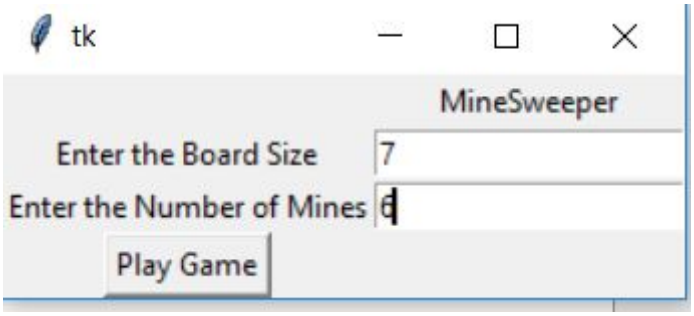
As we don't have much knowledge about the board, we need to make a move within the known region unless all the known region is filled with mines. However, a complete random move might not be logical. Hence, our program makes sure that the next move it makes is an optimal one by choosing the cell with minimum probability in the known region.

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?

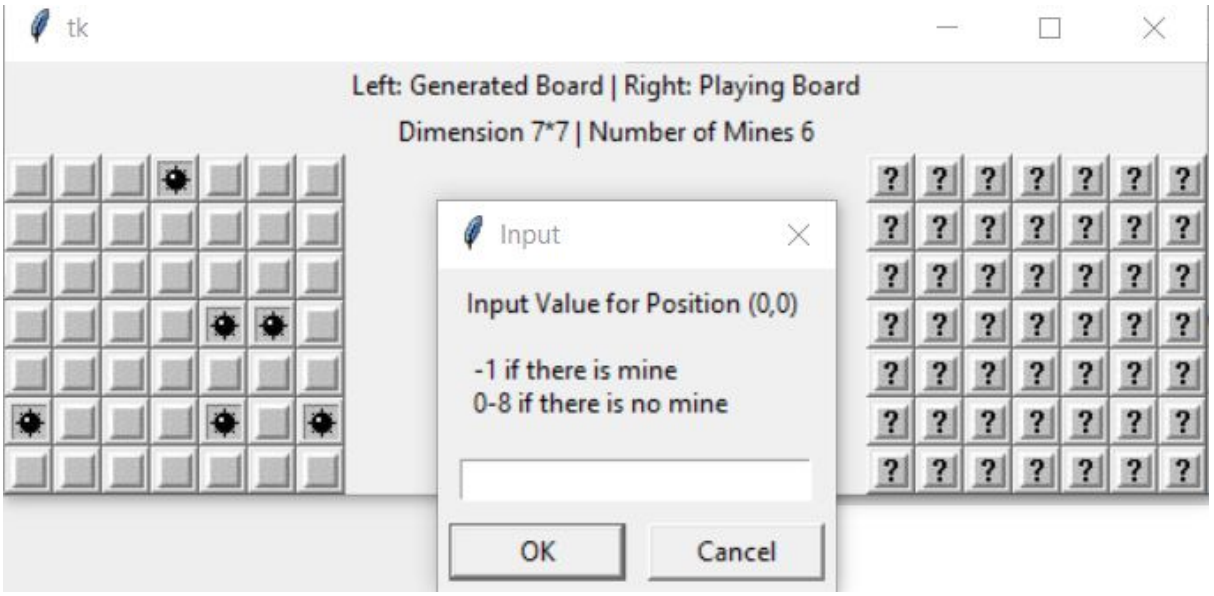
Ans: Yes there are instances where our program makes a decision that I wouldn't agree. This is because exact inference can take exponential time. So, we designed our system in a way that it takes polynomial time. We did one level inference during each addition of a constraint. For this the total time-complexity for inference is O(n^2). The approximation we did is justified because most of the time we explore those cells which are in the neighborhood of already explored cells.
Our program didn't made any decision that surprise us because in all the cases it looks for the most safe cell according to its probability estimate and there is no randomness involved in our algorithm.

Simulation of 7*7 Grid with 6 mines

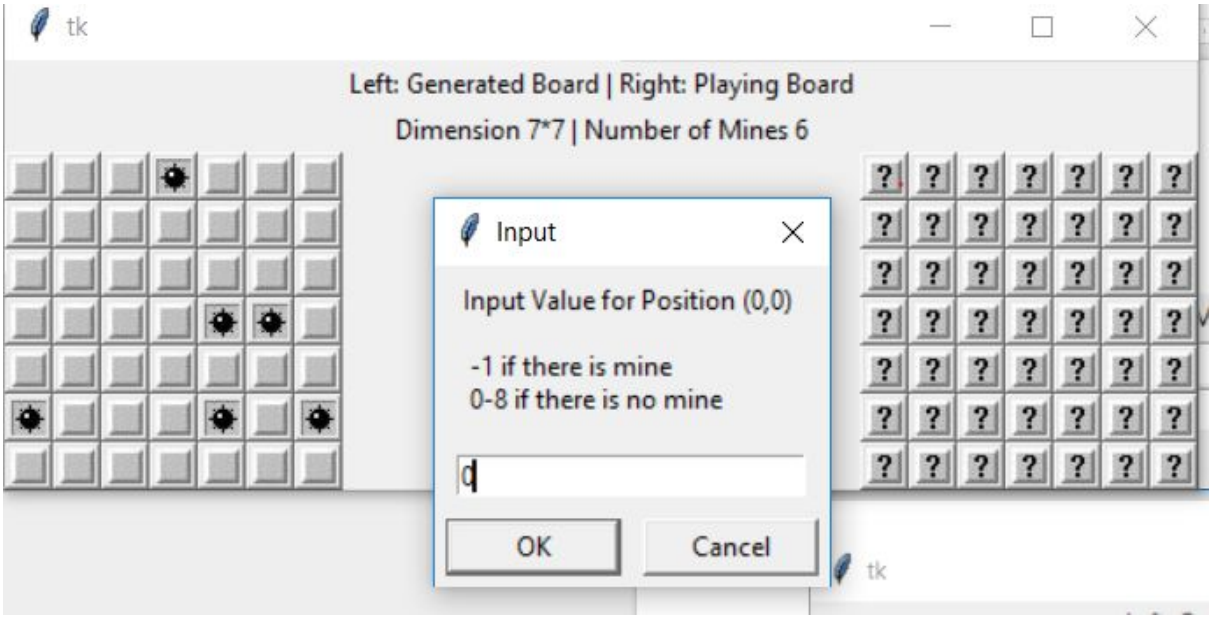
1. Below is the intial screen to take inputs



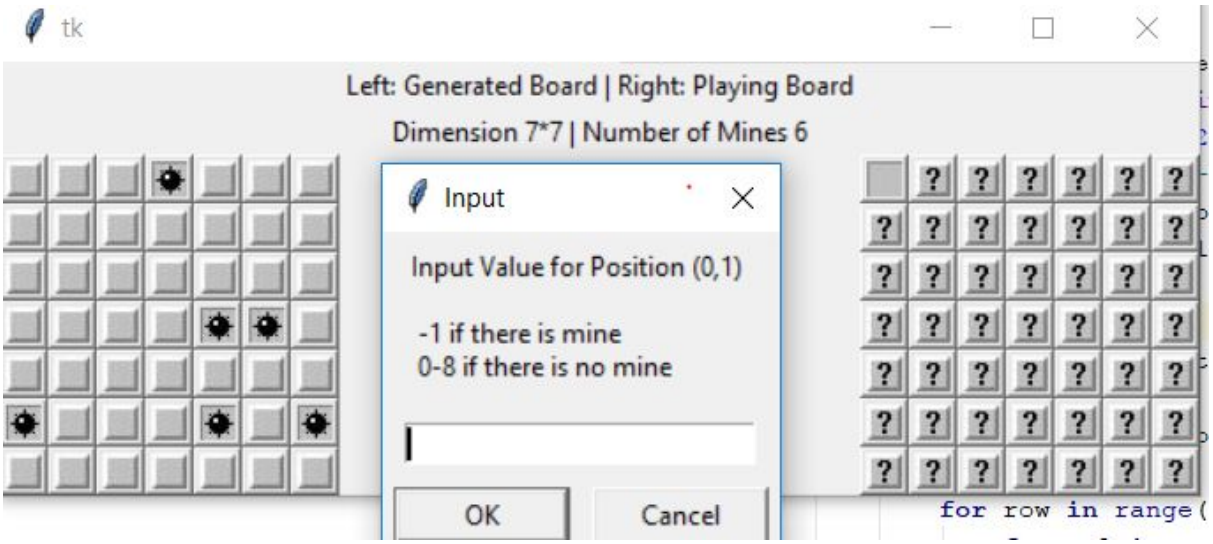
2. Left side is the generated grid. Right side is the grid to be explored. Algorithm asks input for position (0,0)



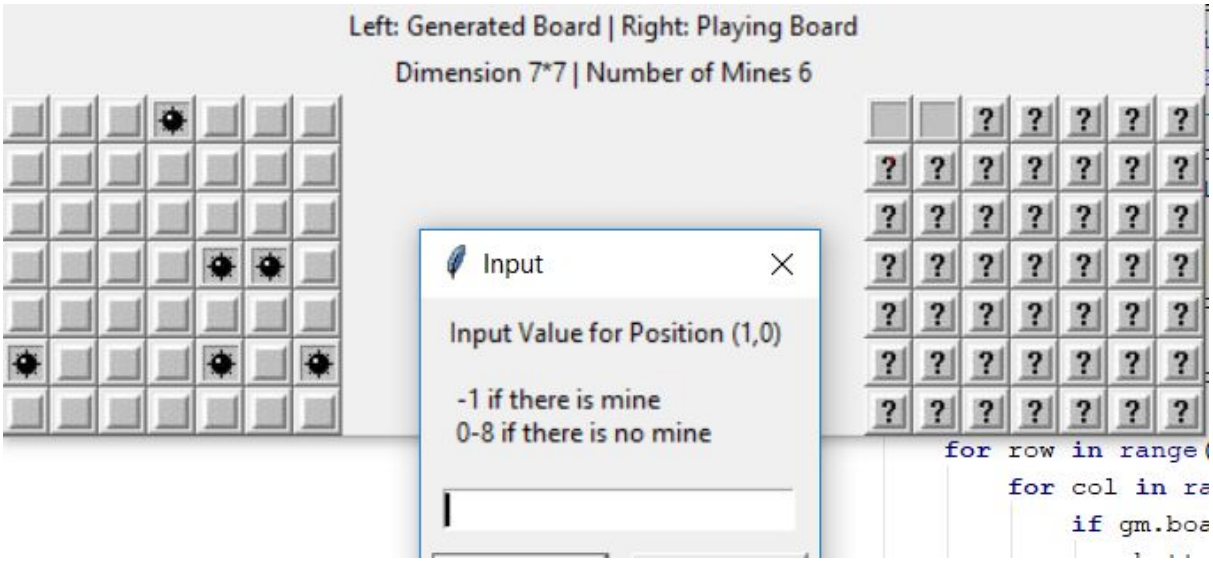
3. Giving the value as zero for position (0,0)



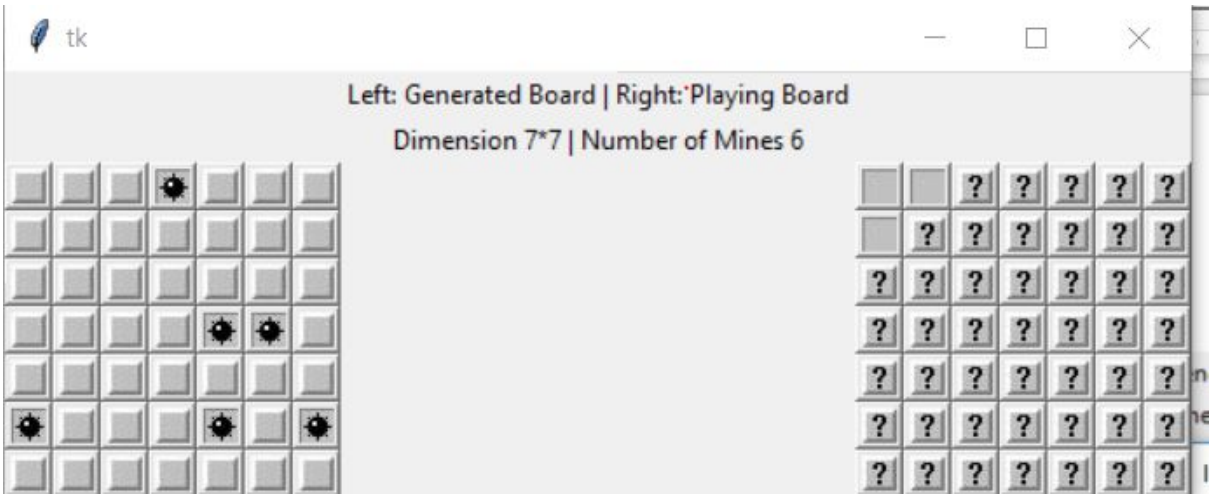
4. Position(0,0) is marked with empty space



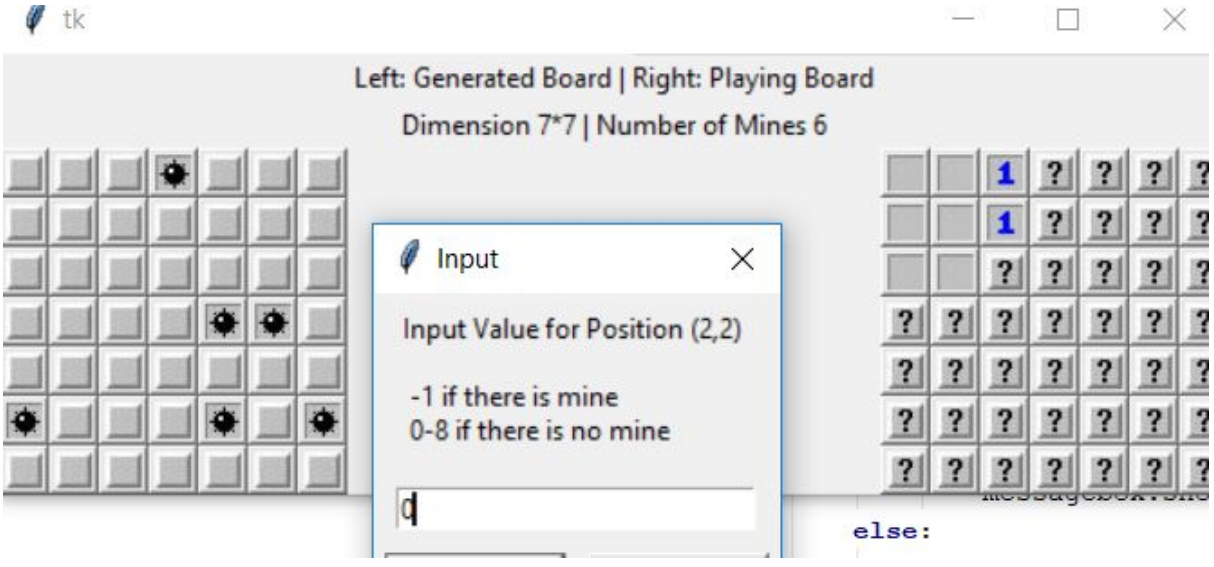
Next it asks to input position (1,0)



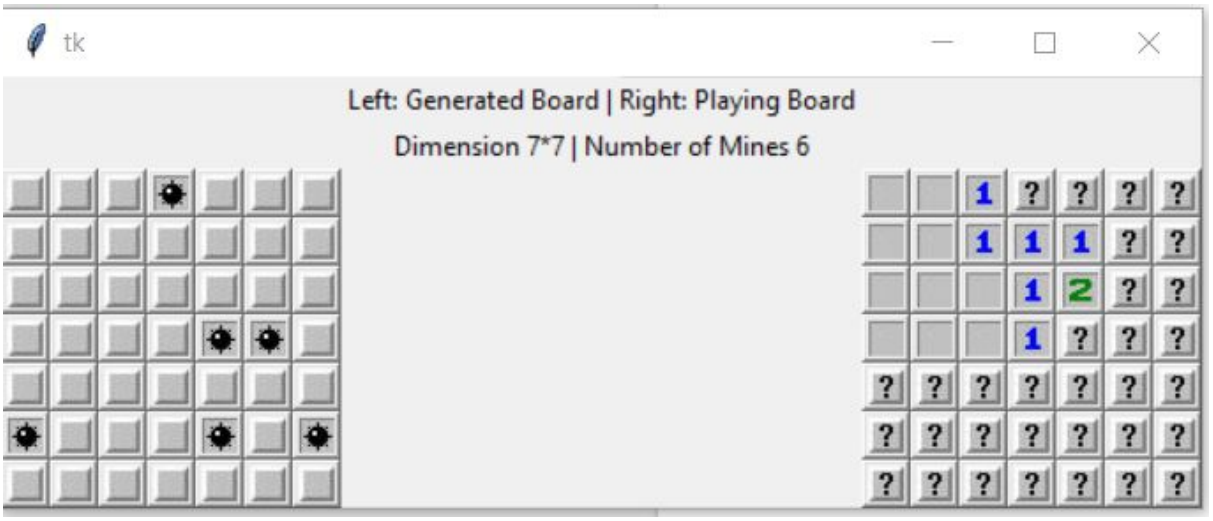
(1,0) is marked with space



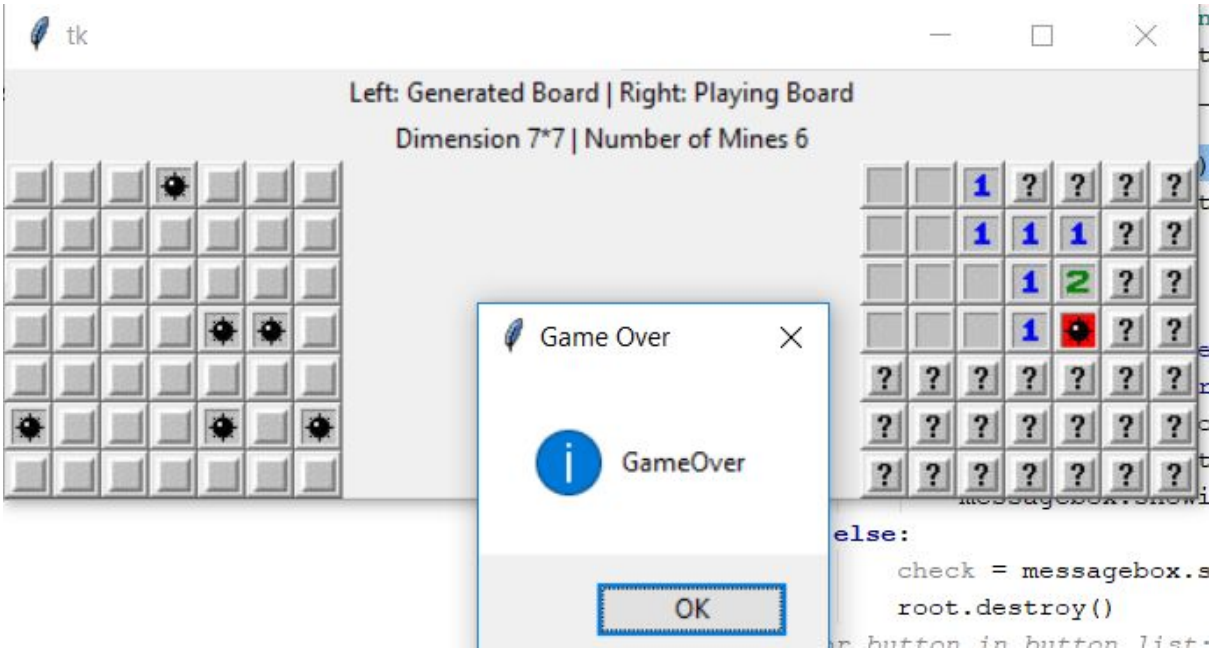
After running for few iterations



After Some iteration current state of grid



We explored grid with mine. So lost



5. Performance: For a fixed, reasonable size of board, what is the largest number of mines that your program can still usually solve? Where does your program struggle?
The program struggles when it won't encounter a zero or a max value. So, if we increase the number of mines the probability of encountering a zero will be reduced and encountering a max value is no increased to that extent. Chances are that the chain of influence will increase as the number of mines increase.(from small values)

6. Efficiency: What are some of the space or time constraints you run into in implementing this program? Are these problem specific constraints, or implementation specific constraints? In the case of implementation constraints, what could you improve on?
Ans:
Space complexity : O(n^2)
Time complexity : O(n^3) (if we can afford an exponential time complexity then the algorithm can be improved further)

Time complexity to get the best result is problem specific constraint. We intentionally compromised on the accuracy to get an algorithm with good time-complexity.

Space is implementation specific constraint, However in worst-case scenario we do require O(n^2) space. We can get better amortized space complexity if we can implement using lists and delete the information that is absolutely unnecessary.

7. Improvements: Consider augmenting your program's knowledge in the following way - when the user inputs the size of the board, they also input the total number of mines on the board. How can this information be modeled and included in your program, and used to inform action? How can you use this information to effectively improve the performance of your program, particularly in terms of the number of mines it can effectively solve.

Ans: If we have the knowledge of number of mines on the board then we know the probability of finding a mine at random on the board. For instance , initially if we encountered a cell with reasonably large number of mines then it is better to explore a cell which is not in the neighborhood of current cell. At any instance we can pick a cell in the vicinity of the explored region or remaining part of the board , we can estimate the probabilities for the cells which are in the vicinity. If minimum of this probabilities is greater than the probability of finding a mine in the rest of the region, then exploring rest of the region is an optimal move to survive after the next move. While exploring rest of the region we assign priority based on the connectivity of the cell to the variables in the constraints list.

Bonus: Chains of Influence

1. Based on your model and implementation, how can you characterize and build this chain of influence?
Ans: In our model, our first move is random. Then, we get some knowledge regarding the surrounding cells of the current cell. Now, we explore an another node where you get more knowledge. Now, in our model at each step we combine the current knowledge with the new knowledge gained to infer useful facts. And every time we encounter a zero or (maximum unknown - mines) in neighbours is equal to the clue value, we can directly infer the individual values of the variables involved and then use this values to update the knowledge base (removing this variables from the constraints)

2. What influences or controls the length of the longest chain of influence when solving a certain board?

Ans: The way we pick the next move controls the longest chain of influence. Since we are choosing the among the neighborhood of the explored territory we are more likely to get a deterministic solution. As there will be common variable in these constraints with the knowledge base. Hence, we are reducing the length of the longest chain of influence.Had we not done that and if we would have chosen a random cell away from the known region then we will likely get a long chain of influence. We could have explored a lot of cells without making a conclusive decision. Thus, increasing the length of the chain of influence.

3. How does the length of the chain of influence, influence the efficiency of your solver?

Ans: Computing new inferences will be costly, according to our algorithm we combine any new information with all the existing knowledge. So, when the chain of influence grows then the number of operations at each step increases. (time taken will be increased drastically)
Also, we have to store more number of constraints in memory.

4. Experiment. Can you find a board that yields particularly long chains of influence? How does this vary with the total number of mines?

Ans: As we increase the number of mines, chain of influence particularly decreases. This is because we can make better inference regarding the safe cells/mine cells after each move. If we have very less number of mines, we have to explore more to get a deterministic solution and thereby increasing the chain of influence.

5. Experiment. Spatially, how far can the influence of a given cell travel?

Ans: We can consider starting from the point (0,0). We gather constraints and move forward. It is quite possible that the mine is present at the extreme far end of the board, let's suppose at (size-1, size-1). Hence, the value obtained after exploring the first cell would affect the value of last cell to be explored. This is the maximum of the chain of influence.

6. Can you use this notion of minimizing the length of chains of influence to inform the decisions you make, to try to solve the board more efficiently?

Ans: We can pick the cells in the which are more connected to the variables . This will minimize the length of chains of influence.

7. Is solving minesweeper hard?

Ans: Yes it is. The one we attempted can be nondeterministic in some cases.(first move is random). We cannot get an optimal solution in polynomial time.

Bonus: Dealing with Uncertainty

1.When a cell is selected to be uncovered, if the cell is 'clear' you only reveal a clue about the surrounding cells with some probability. In this case, the information you receive is accurate, but it is uncertain when you will receive the information.

Ans: If get the information then we follow the algorithm that we implemented but if in some cases we are not receiving any information then we we will pick another cell based on the knowledge we have until we get a cell where we receive any new information to update our knowledge base (something similar rejection sampling). The game might while we are attempting this process !

2. When a cell is selected to be uncovered, the revealed clue is less than or equal to the true number of surrounding mines (chosen uniformly at random). In this case, the clue has some probability of underestimating the number of surrounding mines. Clues are always optimistic.

Ans: If an underestimate of number of surrounding mines is shown then our algorithm can adapt to this situation by changing the way we infer, when we encounter a zero or max number of unknown neighbours. Only when we get underestimate equal to the maximum number of unknown neighbour then only we can conclude all the unknown surrounding cells are mines. When we encounter a zero then we can't conclude anything solely based on this but we can add it our knowledge and hope it infers something useful.

3. When a cell is selected to be uncovered, the revealed clue is greater than or equal to the true number of surrounding mines (chosen uniformly at random). In this case, the clue has some probability of overestimating the number of surrounding mines. Clues are always cautious.

Ans: Here we can only infer when overestimate is 0. In other cases we need to add these facts to knowledge base and hope it infers something useful.