Circuits, Minesweeper, and NP Completeness

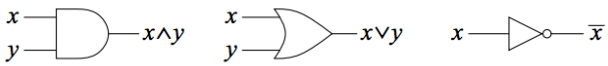
Richard Carini

January 29, 2015

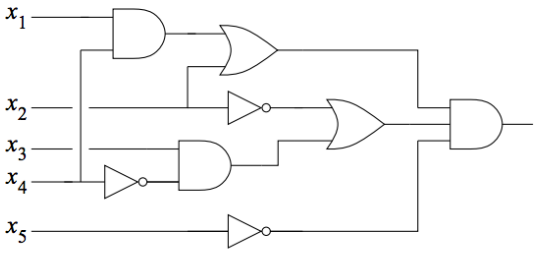
Circuits and NP Completeness

As a short review of what the class has already been introduced to, we know that there exists a classification of problems in NP, problems referred to as NP Complete, that can be used to solve any other problem in NP (provided there exists an algorithm that can solve the original problem). We mentioned that a proof of NP Completeness usually involves reducing the problem to a different problem that we have previously determined is NP Complete. In order to study more and different types of NP Complete problems, we will use the fact that circuit satisfiability is NP Complete.[1] To revisit what we mean by "circuit satisfiability," we want to know whether or not, given an arbitrary circuit, there is an assignment of boolean values (true or false/yes or no/1 or 0) for our starting variables in order for the output to be true. This problem is easily verifiable, given a solution (in NP), and is NP Hard.

A Boolean Circuit[2]



An AND gate, an OR gate, and a NOT gate.



A boolean circuit. Inputs enter from the left, and the output leaves to the right.

Minesweeper

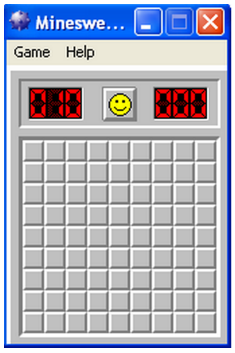
Minesweeper is a game numerous people are familiar with since it comes preinstalled on most versions of Microsoft operating systems. The game is played by starting with a blank grid of squares with a certain number of mines hidden. We begin "exploring" the field by clicking on one of the squares to unveil what is underneath. If the tile was a hidden mine, the game is over and the player has lost. If the tile does not have a mine, the tile will reveal a number indicating how many mines surround that tile

[1] In the previous writeup, we gave an intuitive "proof" that this is true; click here for a reminder on this

[2] Duris, Frantisek

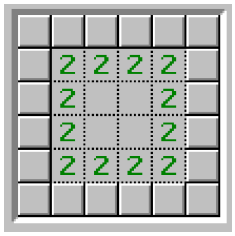
(minimum value is 0 - indicated by a blank; maximum value is an 8). The object of the game is to uncover all tiles that are safe (without a mine) and avoid selecting any squares with mines. There are different versions and variations of the game, but we will be playing the game where each move is not necessarily forced. In other words, there may be some guessing involved or multiple options for the locations of mines.

Example of a Beginner-Level Minesweeper Board[3]

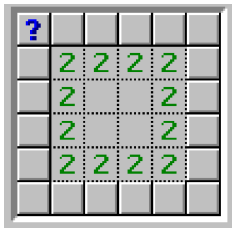


Playing Minesweeper and Minesweeper Satisfiability[4]

Let's begin with an exercise to practice playing Minesweeper.

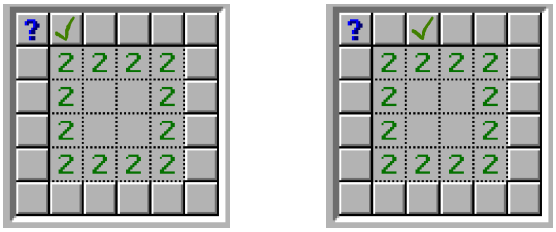


Given the above arrangement, is it possible to find out where each of the mines are without attempting to guess blindly and click a random square?...For that matter, is this arrangement even a viable option that allows for a solvable solution? We begin by analyzing conditions: first, we consider the upper left-hand corner (without loss of generality; this corner can be arbitrarily chosen).

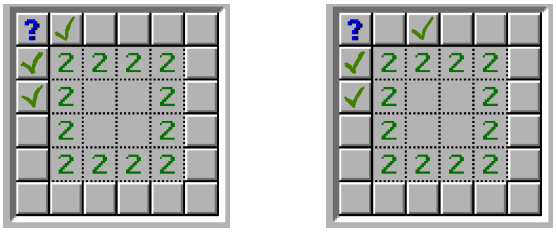


[3]Microsoft
[4]Kaye, Richard

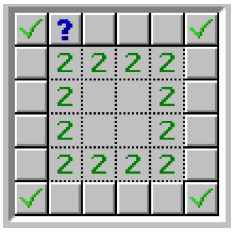
We will use coordinates to refer to the grid as if this were a square matrix. If this upper left-hand square (1,1) was indeed a mine, then there are two possibilities: either the square directly adjacent to it (1,2), OR the square after that (1,3) would be a mine as well (but not both).



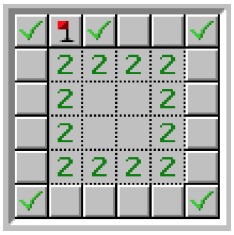
However, either of these configurations would mean that on the adjacent side of the grid, there would be two non-mines that would be assured in positions (2,1) and (3,1).



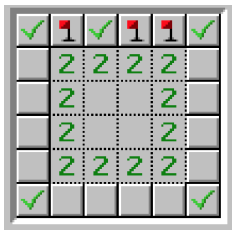
This is a contradiction since the 2 at (3,2) only has one unexplored option that can be a mine...since there are not two mines adjacent to this 2, we know that the corners are not a viable option for a mine. Thus we assert these corner squares are safe and move on to the next square over:



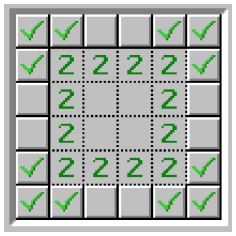
First, we assume the square in question, (1,2), is a mine. We know that the next square over (1,3) must be safe (convince yourself that this is the only viable option: if there were two sequential mines, there would be a contradiction in one of the adjacent sides of the grid in much the same way that we reached our previous contradiction). Therefore, our grid looks like this:



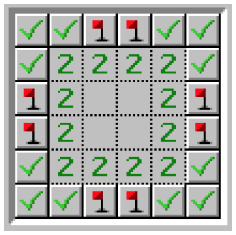
The 2 in the square (2,4) now forces the remaining squares in the first row to be mines, giving



However, this is impossible. Therefore, we know our assumption that there is a mine in (1,2) is false, meaning these squares are also safe:



If we fill in the rest of the squares as mines, we see that this is an actual viable solution, and is a consistent Minesweeper game.



Notice that we tried every possible arrangement of this grid in our cases and showed that each case is impossible except for the solution. Since this solution exists and is unique, we can say that this arrangement was forced. Again, this may not be the case in all of our games of Minesweeper, but it occurs in this specific sample grid. From this example, we can see what is meant when a Minesweeper grid is "consistent."

Definition: A Minesweeper board is *consistent* if there is at least one possible configuration of mines and safe squares that does not result in a contradiction.

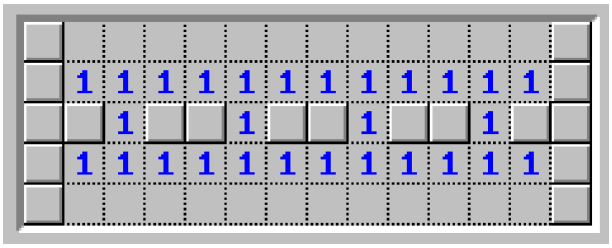
The previous example, through a bit of work, was determined to be consistent because there is a solution to the beginning configuration. Even if there are more than one possible solutions for the arrangements of mines, the grid is still considered to be consistent. The following arrangement, however, is an example of a grid that is not consistent:

We can easily tell that this cannot possibly be a solution: the 6 in the corner only has three neighbors that could potentially be mines. The 2 in the position (1,2) cannot have five mines surrounding this square as is indicated. For this reason, we call this grid "inconsistent." Notice how in our sample problem, even though our grid did end up being forced and did not have any guess work, the process to check was fairly tedious and lengthy. In order to check to see if a grid is consistent, we must check each square, determining whether or not the grid would still be consistent if the square in question was a mine. For this reason, we have yet to develop a clever algorithm to determine a grid's consistency in polynomial time (assuming there even *is* one...)

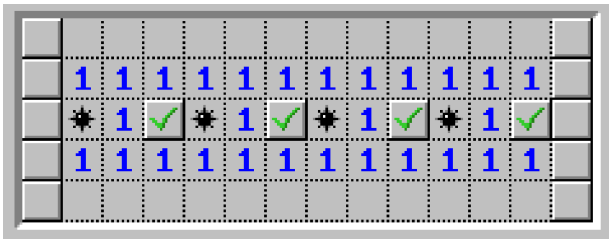
Circuits and Minesweeper

Now that we know how to play Minesweeper and have working definitions for playing the game, we are capable of arriving at a fairly surprising result: Minesweeper is NP Complete! The proof of this result is fairly simple. We start with the knowledge that circuit satisfiability is NP Complete. In an attempt to prove Minesweeper's NP completeness, we begin relating Minesweeper to circuits. We can use the fact that mines' locations and information on one side of the grid can influence information on another side of the grid to our advantage since, essentially, this is what circuits do. The simplest way to carry information in a circuit?

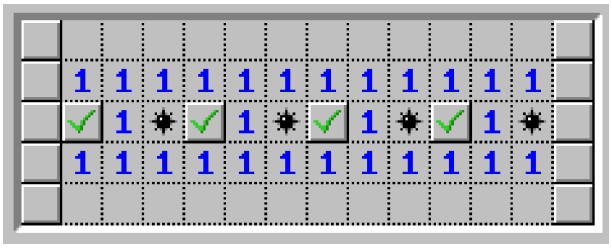
A Wire



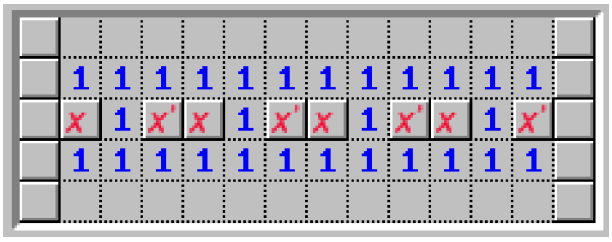
Wires are, fundamentally, a way to transfer the same value from one location to another. In Minesweeper, we can use the above configuration in order to transfer the "truth" value of *mine* or *no mine* down the length of the wire. To see this, let us first assume that the square in position (3,2) has a mine. This assumption would then necessitate that the entire grid look like the following arrangement, carrying the information about the location of the mines down the length of the wire:



Conversely, if the same square was assumed to be safe, the information would still be carried along the length of the wire:



Notationally, we will start to use variables (x, t, u, and the like) to denote the locations where the "truth" value remains the same, and the prime of the variable to indicate the opposite value (x', t', u', etc). Additionally, we give the boolean value of true to any square that has a mine under it.

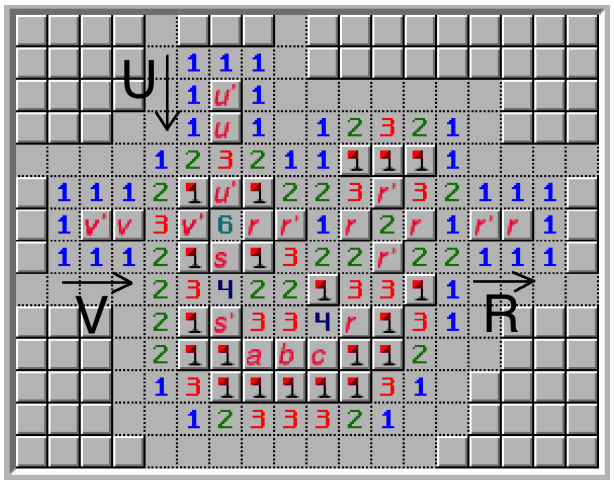


From this labeling, we can say that x is true in our first example arrangement of this wire, and x is false in our second arrangement.

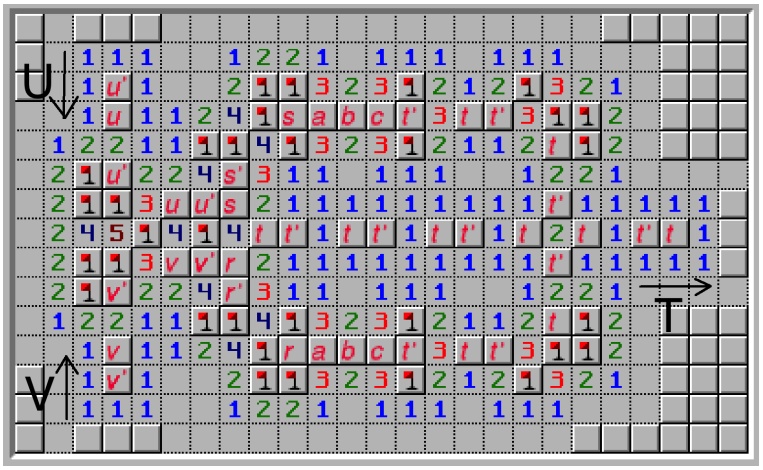
Minesweeper Logic Gates

We are now familiar with the simplest form of information transfer in Minesweeper, but if we are to prove it as NP Complete, we will expect Minesweeper to be able to behave in much more versatile ways than simply retaining information about a square's truth value. Remember that circuits have a series of logic gates (AND, OR, NOT, XOR, etc) that each take in two boolean inputs to output one boolean value. Fortunately, this has been done. In fact, Minesweeper has all of the necessary arrangements for grids that can perform in the same way that these logic gates do.

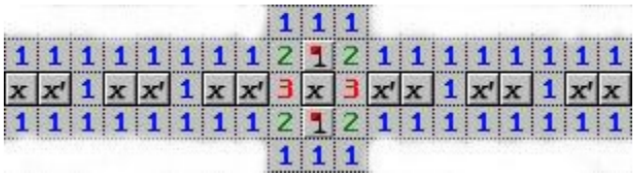
OR Gate



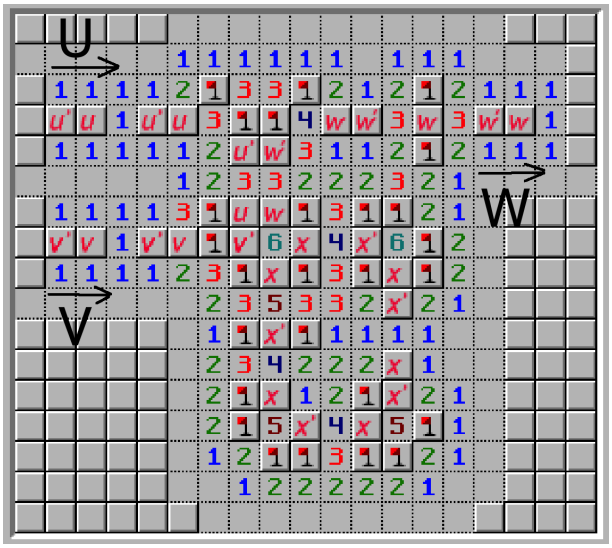
AND Gate



NOT Gate



XOR Gate



This is just a small sampling of the many Minesweeper arrangements that exist to relate Minesweeper to circuits.[5] However, for our purposes, we can remain content with the AND gate, the OR gate, the NOT gate, and the wire. Using these, we will relate boolean circuits to Minesweeper.

Claim: Minesweeper Consistency is NP Complete.

Proof.

- **Minesweeper Consistency is in NP**
  We begin by attempting to show that Minesweeper Consistency is in NP. Suppose we are given a potential solution to an arbitrary Minesweeper grid (in other words, a completely revealed board that has no hidden squares). In order to verify that the solution is consistent, a computer algorithm would step through the grid, one square at a time, verifying that every number has exactly that number's indicated value of mines as neighbors. Each square has a maximum of 8 adjacent squares to check, so given a Minesweeper grid with n squares, the runtime will be at most $8t \cdot n$ (where t is the amount of time necessary to check one square's adjacent partner). Since this algorithm clearly runs in polynomial time, the Minesweeper Consistency problem is in NP.

- **Minesweeper Consistency is NP Hard**
  The fact that Minesweeper is in NP is not enough; we must also prove that this

[5] If you're interested, click here to see even more of these arrangements, including wire crossings, splittings, and more.

problem is NP Hard. Assume we are given an arbitrary boolean circuit that inputs the variables $x_1, x_2, \ldots, x_n$ and outputs the value y. This circuit will have a series of AND, OR, and NOT configurations, which eventually assigns either true or false to y depending on the initial values of our variables. We want an initial boolean value assignment to each variable that will output the value true for y (this, by definition, is our "circuit satisfiability" problem). Since we are able to construct wires, ANDs, ORs, and NOT functions in Minesweeper, we can build a corresponding Minesweeper grid to our circuit. At the "end" or our constructed Minesweeper grid, we assign a mine to the value of y (essentially forcing this value to be true). Assuming we have an algorithm that determines Minesweeper consistency in polynomial time, we can determine a boolean assignment to our initial variables (mine or no mine) that will remain consistent with our assignment of a mine on y. Therefore, circuit satisfiability, in this way, can be determined using Minesweeper Consistency, provided we had an algorithm that runs in polynomial time to check this. Because we know that circuit satisfiability is NP Hard (it can be used to solve any problem in NP), Minesweeper Consistency must therefore be NP Hard as well.

- **Minesweeper Consistency is NP Complete**
  Since Minesweeper Consistency has been shown to be in NP and is NP Hard, by definition, it is NP Complete.

□

Sources:

A very approachable paper and summary on both the Minesweeper consistency problem and P versus NP in general:
Stewert, Ian. *Million Dollar Minesweeper.*
http://www.minesweeper.info/articles/MillionDollarMinesweeper.pdf

Lecture notes for a guide to P, NP, circuits, and SAT:
Frantisek, Duris. *Lecture 16: NP Hard Problems.*
http://www.dcs.fmph.uniba.sk/~fduris/VKTI/3color_HamCycle.pdf

The guide to the proof of Minesweeper's NP Completeness from the discoverer himself:
Kaye, Richard. *Some Minesweeper Configurations.*
http://web.mat.bham.ac.uk/R.W.Kaye/minesw/minesw.pdf

A lecture from Richard Kaye as well, but with more step-by-step approaches to the diagrams as well as great graphics (used in this paper)
Kaye, Richard. *How Complicated is Minesweeper?.*
http://web.mat.bham.ac.uk/R.W.Kaye/minesw/ASE2003.pdf