24NOV2015 Three Pile Proof

The Game

This game involves three piles of stones. There are two players. One player will take a stone/stones per one turn, then the next player will take his/her turn. One player can take as many stones as he/she wants, but only from one pile per turn. The way to win this game is for a player remove all the last stones or the last stone on their turn, making the losing player not be able to remove any more stones.

The Strategy

Binary Code:

The first thing to understand, in order to understand the strategy is binary code. *For the example of this strategy proof, I will use three piles of 10, 12 and 16 stones.* This is crucial to the rest of the proof, so I will represent how these piles would look, written in binary code, below:

16 → *looks like* → 1 0 0 0 0
12 → *looks like* → 0 1 1 0 0
10 → *looks like* → 0 1 0 1 0

Each place in binary code represents an exponential number, which is important to understand:

So binary code represented for this problem has five places → 0 0 0 0 0

The first and farthest right place stands for 2^0 which equals **one*
The second place from the right stands for 2^1 which equals **two*
The third place from the right stands for 2^2 which equals **four*
The fourth place from the right stands for 2^3 which equals **eight*
The first and farthest left place stands for 2^4 which equals **sixteen*

When there is a one instead of a zero, then the value in bold is represented,
when there is a zero, that place represents zero.

Subpiles:

Each place in binary code will represent a subpile to organize the bigger piles. So the original piles of 16, 12 and 10 would be organized like:

16: one subpile of 16 stones
12: one subpile of 8 stones and one subpile of 4 stones
10: one subpile of 8 stones and one subpile of 2 stones

The subpiles are exponential, so they consist of 1, 2, 4, 8, 16 etc. stones. Below I will include the binary representation of the three piles in binary, and show the strategy. Keep in mind that each column represents a different value subpile.

16 → *looks like* → 1 0 0 0 0
12 → *looks like* → 0 1 1 0 0
10 → *looks like* → 0 1 0 1 0

So when you take the three piles and add up the columns…

$$\begin{array}{r} 1\,0\,0\,0\,0 \\ 0\,1\,1\,0\,0 \\ +\ 0\,1\,0\,1\,0 \\ \hline 1\,2\,1\,1\,0 \end{array}$$

You are left with multiple 1's, representing a lack of paired subpiles. The purpose of the strategy is to ensure there are pairs of subpiles only, or a zero meaning there are complete lack of subpiles, spaced throughout the three major piles. This ensures that whatever move the opposing player makes, it can be countered when you take enough to make all of the subpiles paired up with one another, once again. It is all about equal amount of stones, and since there are three piles, spacing equal subpiles across the three piles reaches that goal.

Therefore, to have ones in the binary code of the piles, represented above, is to have a lack of paired subpiles.

The way to win the game is on your turn, every turn you have, to make a binary code consisting of 2's and 0's only. Absolutely **NO ODD NUMBERS.** So for the example above, the strategy and correct move would be to turn the binary code from 1 2 1 1 0 into something with only 2's and zero's. For example 0 2 2 2 0 would work, making even subpiles throughout the three piles. I will show how this is done below.

16 → *looks like* → 1 0 0 0 0
12 → *looks like* → 0 1 1 0 0
10 → *looks like* → 0 1 0 1 0

So now I will take 10 stones from the pile of 16 stones, above to create:

6 → *looks like* → 0 0 1 1 0
12 → *looks like* → 0 1 1 0 0
10 → *looks like* → 0 1 0 1 0

$$
\begin{array}{r}
0\,0\,1\,1\,0 \\
0\,1\,1\,0\,0 \\
+\;0\,1\,0\,1\,0 \\
\hline
0\,2\,2\,2\,0
\end{array}
$$

These are added up to create 0 2 2 2 0 which contains only zeros and 2's, satisfying the rule of equal subpiles.

The Infamous Why?

The reason you can always change a pile so it fits this rule is simple. There is an "error" in a column, when that column adds up to an odd number such as 1 or 3. This again, means there is not a pair of subpiles. Now, the first column with an error, out of the three piles, will either look like any of the columns listed below, with the error in bold.

1	0	0	**1**
0	**1**	0	**1**
0	0	**1**	**1**

Now the row with the error is the row you will always change, to fix all the errors within the binary code sequence. This is because, for every first row with an error, that row contains a one, which means the value is represented. For the problem below, which was previously used, the first row represents 16. In other words, the largest number is represented. This number, when removed from the first row, can fill into any other place value along the binary code. For instance, shown below the first place value 16, is used to fill in the other place values. Also, for the error column with a 1 in every row, you can start with any of the rows to fix the errors.

1 0 0 0 0

Taking the sixteen where the one is above, you can change the binary sequence to:

0 1 1 1 1

This is because the place values of 1+2+4+8=15

So all the place values before the highest one will add up to less than the highest one.

This allows the first row with the error to be able to fix any problem in that row, being able to make all subpiles pair together or not.

So if the rows look like this:

1 0 0 0 0
0 1 1 0 0
0 1 0 1 0

The correct move would be to manipulate the top row, since it has the highest value of 16, you can move distribute the 16 across the columns with errors. So the error column in **bold** below, you would put a 1 there, signaling a value of four, to make that column represent an equal subpile.

1 0 **0** 0 0 → 0 0 **1** 0 0
0 1 **1** 0 0 → 0 1 **1** 0 0
0 1 **0** 1 0 → 0 1 **0** 1 0

Now the rows only have one error, in **bold** below, which can be solved by distributing the value of 16 to that column representing 2.

0 0 1 **0** 0 → 0 0 1 **1** 0
0 1 1 **0** 0 → 0 1 1 **0** 0
0 1 0 **1** 0 → 0 1 0 **1** 0

Now all rows represent equal subpiles, resembling a correct and winning move.

You will always be able to fix every error in just one row, because the only possible errors are three 1's or one 1. With one row, you could change the 3 1's to two 1's, meaning a paired subpile. And you could change the one 1 into two 1's, meaning a paired subpile. Because that one row can either add one or subtract one to make the three 1's or one 1, two 1's. This means you will always be able to correct the errors in the move, in one move, utilizing just one row.