On the Game of Nim

David Liu

December 2015

1 Introduction

Nim is a simple but fascinating strategy and logic game. The game revolves around two players and piles of objects, known as *nim-heaps*, or just heaps. The players take turns and alternate removing a discrete number of objects from these distinct heaps. Players may remove as many objects as they wish, provided they all come from the same heap, but they must remove at least one object, and cannot pass on their turn. Depending on the variation of nim played, there can be any number of heaps, and the objective can either be to remove the last possible object or forcing the other player to take the last possible object.

The history of Nim supposedly stretches back thousands of years to ancient China or other Asian civilizations, but the current name was given by a Harvard professor, Charles Bouton, in 1901.

Nim is an impartial game, meaning the possible moves are identical for each player in any position. Games like Chess and Go are not, for example, because depending on the position of the board and how many pieces are taken, the players can have different possible moves. The only difference between moves in an impartial games is which players turn it is. Nim is a game that has been mathematically solved, meaning that given the starter number of heaps and objects, one can mathematically determine the winner assuming that both players play optimally. As a game of logic, conclusions based on the study of Nim have had wide reaching application. This paper will cover how to solve Nim and how the method of solving it is important to the overarching Sprague-Grundy Theorem.

2 A Sample Game

Suppose I decide to play a game of Nim with my evil alter-ego, Divad. There are three heaps containing one, three, and five objects. The goal is to take the last object.

Table 1: A Game of Nim

Turn	**Heap 1**	**Heap 2**	**Heap 3**	**Move**
0	1	3	5	The game begins
1	1	3	2	Divad removes 3 from Heap 3
2	1	3	1	I remove 1 from Heap 3
3	1	0	1	Divad removes 3 from Heap 2
4	0	0	1	I remove 1 from Heap 1
5	0	0	0	Divad removes 1 from Heap 3

Despite my best efforts, the evil Divad claims victory, and I lie hopeless in defeat. But what allowed Divad to prevail, and could I have done anything to prevent him from winning?

3 Solving Nim

At the end of the sample game, I was stuck in a position where I was guaranteed to lose, since after my move, there was only one object left in the third heap, which Divad would take. Let us use the notation a, b, c to represent this position, where a, b, and c represent how many objects are left in heaps 1, 2, and 3 respectively. Thus, at the end of the previous game, I was in position 0, 0, 1. Let us label this a "N-position," which means the player who moves next is guaranteed a win. Likewise, if we look at the position before this, {1, 0, 1}, it is obvious that this position guarantees a win for the player who went previously, since the only possible move results in an N-position. Let us label this an "P-position."

P-positions guarantee a win for the previous player; the player at the current P-position can only move to an N-position.

N-positions guarantee a win for the next (current) player; the (next) player at the current N-position can move to a P-position, forcing the other player to move to another N-position.

Theorem: Moving to a P-position every turn will result in a win in the Game of Nim.

{0, 0, 1} was a N-position. Since Divad would move "next," I would lose. {0, 1, 1}, the scenario on the previous term, was a P-position, since my only move would change the game to an N-position, so I would lose in that scenario as well. Using these definitions, and by seeing whether I could move a current state to a P-position, I can label each state either a P-position or an N-position. For instance, {0, 1, 4} would be an N-position, since the person who moves on that turn can change the state to {0, 1, 1}, a P-position, forcing the other players move and therefore a win. A list of all P-positions and N-positions for our simple three heap game of Nim is shown below.

Table 2: Nim Game Positions

{a, b, 0}	**{a, b, 1}**	**{a, b, 2}**	**{a, b, 3}**	**{a, b, 4}**	**{a, b, 5}**
{0, 0, 0} : P	{0, 0, 1} : N	{0, 0, 2} : N	{0, 0, 3} : N	{0, 0, 4} : N	{0, 0, 5} : N
{0, 1, 0} : N	{0, 1, 1} : P	{0, 1, 2} : N	{0, 1, 3} : N	{0, 1, 4} : N	{0, 1, 5} : N
{0, 2, 0} : N	{0, 2, 1} : N	{0, 2, 2} : P	{0, 2, 3} : N	{0, 2, 4} : N	{0, 2, 5} : N
{0, 3, 0} : N	{0, 3, 1} : N	{0, 3, 2} : N	{0, 3, 3} : P	{0, 3, 4} : N	{0, 3, 5} : N
{1, 0, 0} : N	{1, 0, 1} : P	{1, 0, 2} : N	{1, 0, 3} : N	{1, 0, 4} : N	{1, 0, 5} : N
{1, 1, 0} : P	{1, 1, 1} : N	{1, 1, 2} : N	{1, 1, 3} : N	{1, 1, 4} : N	{1, 1, 5} : N
{1, 2, 0} : N	{1, 2, 1} : N	{1, 2, 2} : N	{1, 2, 3} : P	{1, 2, 4} : N	{1, 2, 5} : N
{1, 3, 0} : N	{1, 3, 1} : N	{1, 3, 2} : P	{1, 3, 3} : N	{1, 3, 4} : N	{1, 3, 5} : N

Each position has edges that lead to other potential positions after a legal move. In order to guarantee a win, the current player must move to a P-position, since this forces the other player to move to another N-position. Continuing along this train of thought would eventually lead to the {0,0,0} P-position, winning the game.

Drawing the graph might be feasible for small games with a small number of heaps and objects, but how can we guarantee a win for larger games, on the fly, without having the time to write everything out?

If we examine the list of P-positions and N-positions, we can see that for every P-position, XORing the number of objects in the heaps results in 0. For instance, $1 \oplus 2 \oplus 3 = 0$. But for every N-position, XORing the number of objects in the heaps results in some number greater than 0. For instance, $0 \oplus 1 \oplus 2 = 3$.

The sum that results from XORing all the remaining objects in the heaps is known as a nim-sum. XOR, otherwise known as an exclusive-or is a logical operation. Essentially, it is equivalent to bitwise addition in mod 2. As an example, say we want to XOR the two numbers 9 and 7. 9 can be written in base 2 as 1001, while 6 can be written in base 2 as 0111.

$$\begin{array}{r} 1001 \\ +0111 \\ \hline 1110 \end{array}$$

We add the bits in each column in base 2, treating them individually, dropping the carry out bit unlike regular addition. So 9 XOR 7 produces 14.

We want to prove that the nim-sum should equal 0 for every P-position; that is, in order to win a game of Nim, we want to make every move such that the position after we move has a nim-sum of 0.

3.1 Proof of P-positions and N-positions

3.1.1 Moving from a P-position results in an N-position

Let P denote the set of all P-positions in a game of Nim. If $\{x_1, x_2...x_n\}$ is in P, then $x_1 \oplus x_2 \oplus ...x_n$ equals the nim-sum of some P-position in P. Removing objects from a heap would thus change some heap x_1 to $x_1' < x_1$. The nim-sum of this new position would equal $x_1' \oplus x_2 \oplus ...x_n$. We can prove that these nim-sums are different by contradiction.

Assume that the nim-sums are identical. Then $x_1' \oplus x_2 \oplus ...x_n = x_1 \oplus x_2 \oplus ...x_n$. Since XOR is commutative, we can simplify the above expression to $x_1' = x_1$. But we know that $x_1' < x_1$, so this contradicts our original statement. Clearly, the nim-sum always changes after a move. Therefore, if the nim-sum is equivalent to 0, and the game is in a P-position, the next move will change the nim-sum, which will no longer be equal to 0. The game moves to an N-position, completing this part of the proof.

3.1.2 From an N-position, a move exists to a P-position

Now that we know that moving from a P-position results in an N-position, we only need to show that a player can always move to a P-position if they are currently in an N-position.

Let N denote the set of all P-positions in a game of Nim. If $\{y_1, y_2...y_n\}$ is in P, then $y_1 \oplus y_2 \oplus ...y_n$ equals the nim-sum of some N-position in N. Removing objects from a heap would thus change some heap y_1 to $y_1' < y_1$. The nim-sum of this new position would equal $y_1' \oplus y_2 \oplus ...y_n$.

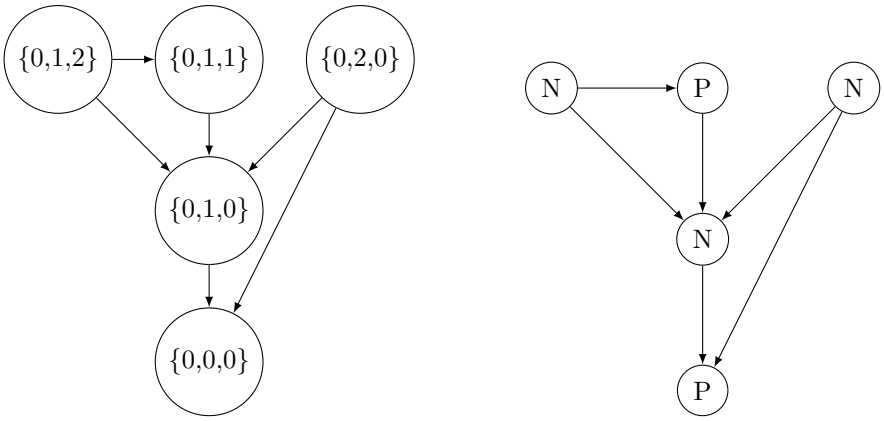
Since XOR is associative, we can write $y_1 \oplus y_2 \oplus ...y_n$ as $y_1' \oplus (y_2 \oplus ...y_n)$. Assume that y_1 is the largest heap.

We know that $y_2 \oplus ...y_n$ cannot possibly exceed the value of y_1, because of how XOR works, and it also cannot be equal to y_1, because then we would be XORing two of the same thing, and the nim-sum would equal 0. From this, we can conclude that we can also remove a certain number of objects from the

largest nim-heap in order to make $y_1' = y_2 \oplus ...y_n$, which would make the nim-sum of the new position equal to 0. By the first part of our proof, this new position is a P-position, completing the rest of the proof.

3.2 The Sprague-Grundy Theorem

The concept of Nim can be universally applied to almost all impartial games. The Sprague-Grundy Theorem is a generalization of the solving of Nim and other impartial games. First, let us return to our previous table above, which writes out every possible position in our game of Nim. However, we could also write it in the form of a tree-like graph, linking positions that one couuld possibly move by edges and representing positions with nodes. A small chunk of the list of positions is shown below.



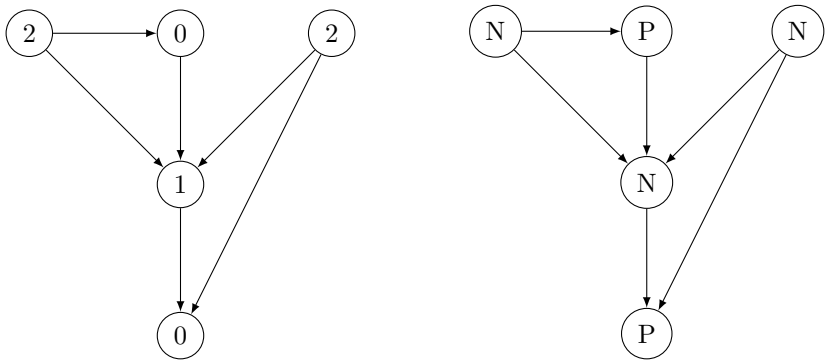
Potentially, we could model any impartial game on a graph like this one. Let us define this graph as $G = (X, F)$, where X is the set of all possible positions, and F is a function that returns a subset that contains the list of possible nodes to move to. If the subset is empty, the game is over and the player who must move on that turn loses.

In the above graph, we labeled the graph with the according N and P positions. Let us now introduce the mex function, or Minimum Excluded Value function.

$$g(x) = min\{n \geq 0 : n \neq g(y) \text{ for } y \in F(x)\}$$

This function takes a node in the graph, examines the node and the nodes it could potentially move to, and labels the node with a value that its followers does not already have. Thus, this is a recursive function that can be used to assign values to every node in the graph. For instance, take the {0,0,0} node in

our graph. It has no places to move, so we assign it the value of 0, since that is the least value that has not yet been used. Next {0,1,0} can be labeled with 1, since its only follower, {0,0,0}, already has 0 taken. {0,2,0} would be labeled 2, since its two followers have already taken 0 and 1. We can continue to do this throughout the entire graph. The sample graph here is labeled with the mex values.



The mex values of 0 seem to match up with P-positions.

Next, graphs can be added. Say we have $G_1 = (X1, F1)$, and $G_2 = (X2, F2)$. The sum of these games will be $G(X, F) = G_1 + G_2$.

The Sprague-Grundy function states that the mex function for a sum of games on a graph is just the Nim sum of the mex functions of its components.

3.3 Conclusion

We've thus proven that our notion of P-positions and N-positions seems to hold true for any position in any game of Nim. Thus, assuming perfect play, going first will assure a victory if the starting configuration is an N-position, since the first player has the power to move into a P-position and set up for a win. Otherwise, if the starting configuration is in a P-position, the player who moves second can guarantee a win. The sample game above started in an N-position, since $1 \oplus 3 \oplus 5 = 7 \neq 0$, meaning that since Divad went first, he was guaranteed a win assuming he played perfectly, which he did.

3.4 References

"Theory of Impartial Games." 3 Feb. 2009. Web. 11 Dec. 2015.

Davis, Dean. "The Game of Nim Expository Paper." 1 July 2006. Web. 11 Dec. 2015.

Wikipedia. Wikimedia Foundation. Web. 11 Dec. 2015.