The Game of Nim

H. Ronald,
MS19167

IISER Mohali

March 22, 2023

Combinatorial games

Usually have the following properties

Combinatorial games

Usually have the following properties

1. There are two players (say P1 and P2).

Combinatorial games

Usually have the following properties

1. There are two players (say P1 and P2).
2. Players alternate making moves.

Combinatorial games

Usually have the following properties

1. There are two players (say P1 and P2).
2. Players alternate making moves.
3. The game ends when a player can't make a move.

Combinatorial games

Usually have the following properties

1. There are two players (say P1 and P2).
2. Players alternate making moves.
3. The game ends when a player can't make a move.
4. It's a finite game (there are finitely many game positions and the game eventually ends).

Nim

How to play

There are a finite number of heaps of coins.

Nim

How to play

There are a finite number of heaps of coins. P1 and P2 alternate taking off any number of coins from a pile until there are no coins left.

There are a finite number of heaps of coins. P1 and P2 alternate taking off any number of coins from a pile until there are no coins left. The player who makes the last move wins (normal play).

Nim

Positions

To represent a game position, we can use the notation $(x_1, x_2, \ldots, x_n)$, where there are x_i coins in the ith heap and so on.

To represent a game position, we can use the notation $(x_1, x_2, \ldots, x_n)$, where there are x_i coins in the ith heap and so on.

A game is in a **P-position** if it secures a win for the Previous player (the one who just moved).

To represent a game position, we can use the notation $(x_1, x_2, \ldots, x_n)$, where there are x_i coins in the ith heap and so on.

A game is in a **P-position** if it secures a win for the Previous player (the one who just moved).

A game is in a **N-position** if it secures a win for the Next player.

Nim

Positions

To represent a game position, we can use the notation $(x_1, x_2, \ldots, x_n)$, where there are x_i coins in the ith heap and so on.

A game is in a **P-position** if it secures a win for the Previous player (the one who just moved).

A game is in a **N-position** if it secures a win for the Next player.

For instance, $(0, 0, 1)$ is an N-position and $(1, 1, 0)$ is a P-position.

Backward induction

To find whether a Nim position is N or P

Backward induction

To find whether a Nim position is N or P

A **terminal position** is one from which no possible moves are left.

Backward induction

To find whether a Nim position is N or P

A **terminal position** is one from which no possible moves are left.

Backward induction works as follows:

1. Label every terminal position as P.

Backward induction

To find whether a Nim position is N or P

A **terminal position** is one from which no possible moves are left.

Backward induction works as follows:

1. Label every terminal position as P.
2. Label every position that can reach a P position as N.

Backward induction

To find whether a Nim position is N or P

A **terminal position** is one from which no possible moves are left.

Backward induction works as follows:

1. Label every terminal position as P.
2. Label every position that can reach a P position as N.
3. For positions that only move to N positions, label P.

Backward induction

To find whether a Nim position is N or P

A **terminal position** is one from which no possible moves are left.

Backward induction works as follows:

1. Label every terminal position as P.
2. Label every position that can reach a P position as N.
3. For positions that only move to N positions, label P.
4. Return to step 2 and repeat the process until all positions are labeled.

Solving Nim

XOR operation and Nim-sum

We define XOR operation on two binary numbers: $x \oplus y$ as the bitwise XOR operation, that is, an odd number of 1s give 1 and an even number of 1s give 0.

Solving Nim

XOR operation and Nim-sum

We define XOR operation on two binary numbers: $x \oplus y$ as the bitwise XOR operation, that is, an odd number of 1s give 1 and an even number of 1s give 0. For instance,

3	011
4	100
5	101
2	010

Solving Nim

XOR operation and Nim-sum

We define XOR operation on two binary numbers: $x \oplus y$ as the bitwise XOR operation, that is, an odd number of 1s give 1 and an even number of 1s give 0. For instance,

$$\begin{array}{cc} 3 & 011 \\ 4 & 100 \\ 5 & 101 \\ \hline 2 & 010 \end{array}$$

At any position, the **Nim-sum** is the XOR of all heap sizes. So, the Nim-sum of $(3, 4, 5)$ is 2.

Solving Nim

Lemma

If a position has Nim-sum 0*, the next move changes it to some non-zero value.*

Solving Nim

Lemma

If a position has Nim-sum 0, the next move changes it to some non-zero value.

Proof.

Consider a position $(x_1, x_2, \ldots, x_n)$ with $s = x_1 \oplus \cdots \oplus x_n$.

Solving Nim

Lemma

If a position has Nim-sum 0, the next move changes it to some non-zero value.

Proof.

Consider a position $(x_1, x_2, \ldots, x_n)$ with $s = x_1 \oplus \cdots \oplus x_n$. The next move changes the position to $(y_1, y_2, \ldots, y_n)$ with $t = y_1 \oplus \cdots \oplus y_n$.

Solving Nim

Lemma

If a position has Nim-sum 0*, the next move changes it to some non-zero value.*

Proof.

Consider a position $(x_1, x_2, \ldots, x_n)$ with $s = x_1 \oplus \cdots \oplus x_n$. The next move changes the position to $(y_1, y_2, \ldots, y_n)$ with $t = y_1 \oplus \cdots \oplus y_n$. Since only one heap can be modified after a move, we have $x_k \neq y_k$ for some k and $x_i = y_i$ for $i \neq k$.

Solving Nim

Lemma

If a position has Nim-sum 0*, the next move changes it to some non-zero value.*

Proof.

Consider a position $(x_1, x_2, \ldots, x_n)$ with $s = x_1 \oplus \cdots \oplus x_n$. The next move changes the position to $(y_1, y_2, \ldots, y_n)$ with $t = y_1 \oplus \cdots \oplus y_n$. Since only one heap can be modified after a move, we have $x_k \neq y_k$ for some k and $x_i = y_i$ for $i \neq k$. Then

$$t = t \oplus s \oplus s = (y_1 \oplus x_1) \oplus \cdots \oplus (y_n \oplus x_n) \oplus s = y_k \oplus x_k \oplus s.$$

Solving Nim

Lemma

If a position has Nim-sum 0, the next move changes it to some non-zero value.

Proof.

Consider a position $(x_1, x_2, \ldots, x_n)$ with $s = x_1 \oplus \cdots \oplus x_n$. The next move changes the position to $(y_1, y_2, \ldots, y_n)$ with $t = y_1 \oplus \cdots \oplus y_n$. Since only one heap can be modified after a move, we have $x_k \neq y_k$ for some k and $x_i = y_i$ for $i \neq k$. Then

$$t = t \oplus s \oplus s = (y_1 \oplus x_1) \oplus \cdots \oplus (y_n \oplus x_n) \oplus s = y_k \oplus x_k \oplus s.$$

The term $y_k \oplus x_k$ is never zero. If $s = 0$, then $t \neq 0$. □

Solving Nim

Lemma

If a position has a Nim-sum of non-zero value, it is always possible to make it zero by the next move.

Solving Nim

Lemma

If a position has a Nim-sum of non-zero value, it is always possible to make it zero by the next move.

Proof.

Choose a heap x_k such that its most significant bit is in the same position as that of the most significant bit in s (one must always exist, the most significant bit of s must come from the most significant bit of any of the heaps).

Solving Nim

Lemma

If a position has a Nim-sum of non-zero value, it is always possible to make it zero by the next move.

Proof.

Choose a heap x_k such that its most significant bit is in the same position as that of the most significant bit in s (one must always exist, the most significant bit of s must come from the most significant bit of any of the heaps). Make the new value of the heap $y_k = s \oplus x_k$ by removing $x_k - y_k$ coins from the heap.

Solving Nim

Lemma

If a position has a Nim-sum of non-zero value, it is always possible to make it zero by the next move.

Proof.

Choose a heap x_k such that its most significant bit is in the same position as that of the most significant bit in s (one must always exist, the most significant bit of s must come from the most significant bit of any of the heaps). Make the new value of the heap $y_k = s \oplus x_k$ by removing $x_k - y_k$ coins from the heap. The new New-sum will be:

$$t = y_k \oplus x_k \oplus s = s \oplus x_k \oplus x_k \oplus s = 0.$$

□

Solving Nim

Theorem

Any position which has a Nim-sum 0 is a P-position. All other positions are N-positions.

Solving Nim

Theorem

Any position which has a Nim-sum 0 *is a P-position. All other positions are N-positions.*

Proof.

If we start off by making our first move so that the Nim-sum is 0, then on each turn our opponent will disturb the sum, and we will in turn set it back to 0.

Solving Nim

Theorem

Any position which has a Nim-sum 0 is a P-position. All other positions are N-positions.

Proof.

If we start off by making our first move so that the Nim-sum is 0, then on each turn our opponent will disturb the sum, and we will in turn set it back to 0. Eventually on our turn there will be no coins left with a Nim-sum of 0, meaning that we are the winner!

Solving Nim

Theorem

Any position which has a Nim-sum 0 is a P-position. All other positions are N-positions.

Proof.

If we start off by making our first move so that the Nim-sum is 0, then on each turn our opponent will disturb the sum, and we will in turn set it back to 0. Eventually on our turn there will be no coins left with a Nim-sum of 0, meaning that we are the winner! Of course, if the Nim-sum starts off at 0 and we go first, then we must hope for our opponent to make a mistake, since they will have the winning strategy. □

Winning Nim

A more practical strategy

Winning Nim

A more practical strategy

1. Whenever possible, reduce the heaps to two non-zero heaps containing the same number of coins each. This obviously has a nim-sum of 0.

Winning Nim

A more practical strategy

1. Whenever possible, reduce the heaps to two non-zero heaps containing the same number of coins each. This obviously has a nim-sum of 0. Now just mimic your opponent's move each time on the opposite heap to keep the two heaps equal until you are able to take the final coin.

Winning Nim

A more practical strategy

1. Whenever possible, reduce the heaps to two non-zero heaps containing the same number of coins each. This obviously has a nim-sum of 0. Now just mimic your opponent's move each time on the opposite heap to keep the two heaps equal until you are able to take the final coin.
2. An easy way to think about making the Nim-sum 0 is to always leave even subpiles of the powers of 2, starting with the largest power possible, where a subpile is a pile group of coins within a nim-heap.

Winning Nim

A more practical strategy

1. Whenever possible, reduce the heaps to two non-zero heaps containing the same number of coins each. This obviously has a nim-sum of 0. Now just mimic your opponent's move each time on the opposite heap to keep the two heaps equal until you are able to take the final coin.
2. An easy way to think about making the Nim-sum 0 is to always leave even subpiles of the powers of 2, starting with the largest power possible, where a subpile is a pile group of coins within a nim-heap. So for example, leave an even number of subpiles of 2, 4, 8, 16, etc. Any time there are an even number of piles of each power of 2, the nim-sum must be 0.

References

Theory of Impartial Games. Available at: https://web.mit.edu/sp.268/www/nim.pdf