Nim Game Problem

• Nim is a classic game, of which there are many variants. Here is one:
• Several piles of “tokens” are placed on the table. Players take turns removing a *non-zero* number from just one of the piles. The player who takes the last token wins.

Nim Example

• For brevity, we will represent the piles as a list of the number of tokens in each non-empty pile. Empty piles are not shown.
• Example:
  [3, 6, 9, 2]
  means a pile with 3 tokens, a pile with 6 tokens, one with 9, and one with 2.

Nim Play Example

• player 1: [3, 6, 9, 2] → [3, 6, 7, 2]
• player 2: [3, 6, 7, 2] → [3, 7, 2]
• player 1: [3, 7, 2] → [3, 1, 2]
• player 2: [3, 1, 2] → [1, 2]
• player 1: [1, 2] → [1, 1]
• player 2: [1, 1] → [1]
• player 1: [1] → [] wins
Problem: Construct a good move function for Nim

- We want our function to be in the form of a rex definition:

  \[
  \text{move}(L) = \ldots
  \]

Strategy?

- What is an appropriate strategy?
- Consider a coarser version of the game, in which a player must take all or none of a pile.
- The “strategy” is clear:

  You win if you can leave an even number of piles.

Generalizing the Strategy

- An appropriate generalization is to use the idea of a \text{nim\_sum} of the piles such that:
  - The \text{nim\_sum} of no piles is 0.
  - If the \text{nim\_sum} is not 0, you can always move to leave a \text{nim\_sum} of 0.
  - Every move changes the \text{nim\_sum}

  You can win if you can always leave a \text{nim\_sum} of 0.

Nim Play Showing \text{nim\_sum} in Parens

- player 1: \([3, 6, 9, 2]\) (14) \(\rightarrow\) \([3, 6, 7, 2]\) (0)
- player 2: \([3, 6, 7, 2]\) (0) \(\rightarrow\) \([3, 7, 2]\) (6)
- player 1: \([3, 7, 2]\) (6) \(\rightarrow\) \([3, 1, 2]\) (0)
- player 2: \([3, 1, 2]\) (0) \(\rightarrow\) \([1, 2]\) (3)
- player 1: \([1, 2]\) (3) \(\rightarrow\) \([1, 1]\) (0)
- player 2: \([1, 1]\) (0) \(\rightarrow\) \([1]\) (1)
- player 1: \([1]\) (1) \(\rightarrow\) \([]\) (0) wins
The Mysterious \textit{nim\_sum}

• In order to explain \textit{nim\_sum} and why it works, we need express numbers as \textit{binary numerals}.
• Recall: this simply means as a sum of powers of 2. For example,
\[ 9 = 8 + 0 + 0 + 1 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \]
written \( 1001_2 \).

\textit{nim\_sum} of two numbers

• The \textit{nim\_sum} of two numbers is the number corresponding to the bit-wise exclusive-OR (xor) of their binary representations.
• Example:
\[
\begin{align*}
13 & = 1101_2 \\
14 & = 1110_2 \\
\text{xor} & = 0011_2 = 3
\end{align*}
\]

\textit{nim\_sum} of a list of numbers

• The \textit{nim\_sum} of a list numbers is the reduction of the numbers in the list by the xor on two numbers:
\[ \text{nim\_sum}(L) = \text{reduce}(\text{xor}, 0, L); \]
• It can easily be seen that:
  - xor is associative and commutative
  - 0 is the unit for xor
  - Also, for any \( z \), \( \text{xor}(z, z) = 0 \).

Exercise: Check that these nim\_sums are correct.

• \([3, 6, 9, 2]\) (14)
• \([3, 6, 7, 2]\) (0)
• \([3, 7, 2]\) (6)
• \([3, 1, 2]\) (0)
• \([1, 2]\) (3)
• \([1, 1]\) (0)
• \([1]\) (1)
• \([\ ]\) (0)
Progress So Far

• We decomposed \texttt{nim\_sum} into the use of \texttt{reduce} and \texttt{xor}.
• We now need to show how to compute the move.
• Claim: If the \texttt{nim\_sum} of a list is not 0, it is always possible to get it to 0 by modifying just one pile.

Proof of Claim

• Suppose \texttt{nim\_sum}(L) = s, where \( s \neq 0 \).
• We need to show that there is a valid move that diminishes some pile by \texttt{xor}’ing with \( s \).

\[
\texttt{nim\_sum([a, b, c, d])} = s \\
\texttt{xor(c,s)}
\]

Example

\[
\begin{array}{c}
[3, 6, 9, 2]:\\
0:0|1:1 \\
0:1|1:0 \\
1:0|0:1 \\
0:0|1:0 \\
\hline
\end{array}
\]

\[
\texttt{xor} \\
1:1|1:0
\]

• With which pile can we \texttt{xor} \texttt{1110} to diminish the size of the pile?

Example

\[
\begin{array}{c}
[3, 6, 9, 2]:\\
0:0|1:1 \\
0:1|1:0 \\
1:0|0:1 \\
0:0|1:0 \\
\hline
\end{array}
\]

\[
\texttt{xor} \\
1:1|1:0
\]

• The only pile that qualifies is \texttt{1001}.
\[
\texttt{xor}(1001, 1110) = 0111.
\]
• So we want to change pile 9 to 7 (by taking 2).
Result

- 00110110010
- xor

Generalization

- If the nim_sum is non-zero then there is always a pile which when xor'ed with nim_sum yields a smaller pile.
  - Not necessarily the largest pile. Consider [9, 8, 7]

  - Rationale: The highest-order bit in the nim_sum must have come from one of the piles.
    - Again, not necessarily the largest pile.
    - When we xor the nim_sum to that very pile, this bit becomes 0, meaning that the resulting pile is necessarily smaller.

Completing the move function

- Compute s, the nim_sum of the list of piles.
- If s is not 0:
  - Find a pile p such that xor(p, s) < p (where < means numerically less than) and replace it with xor(p, s).
  - Remove any empty pile.
- Otherwise?