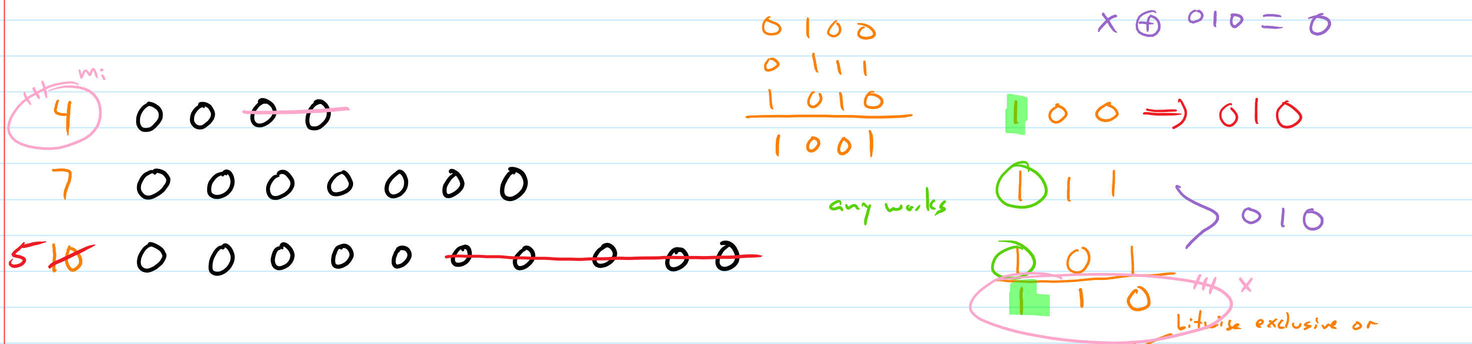
For Nim, there is a winning move if and only if the bitwise exclusive or of the number of stones left in each row is non-zero, and the winning moves are the ones that make the bitwise exclusive or 0.



Lemma: If $m_1, \dots, m_n$ stones left in each row and $m_1 \oplus m_2 \oplus \cdots \oplus m_r = 0$ then all moves $m_i \to m_i'$ make Nim-sum non-zero. m_i = # stones in row i

Proof: Suppose $m_1 \oplus \cdots \oplus m_r = 0$. Let $m_i \to m_i'$ be a legal move. So $m_i' < m_i$.

Then resulting Nim-sum is $m_1 \oplus \cdots \oplus m_{i-1} \oplus m_i' \oplus m_{i+1} \oplus \cdots \oplus m_r$

bitwise exclusive or

$$= m_1 \oplus \cdots \oplus m_{i-1} \oplus m_i' \oplus m_{i+1} \oplus \cdots \oplus m_r \oplus (m_i \oplus m_i)$$

$$= \underbrace{m_1 \oplus \cdots \oplus m_{i-1} \oplus m_i \oplus m_{i+1} \oplus \cdots \oplus m_r}_{} \oplus (m_i' \oplus m_i)$$

$$= \underbrace{0} \oplus (m_i' \oplus m_i) = m_i' \oplus m_i \neq 0$$ since $m_i' \neq m_i$

Lemma: If $m_1, \dots, m_n$ stones left in each row and $m_1 \oplus m_2 \oplus \cdots \oplus m_r \neq 0$ then there is a move $m_i \to m_i'$ make Nim-sum zero.

Proof: Suppose $m_1 \oplus \cdots \oplus m_r = x \neq 0$. Find most significant bit msb of x.
Find i s.t. m_i has that bit set. Let $m_i' = m_i \oplus x$ (must be 1 for bit in xor to be 1)
Choose move that reduces m_i to m_i'. $m_i' < m_i$, so legal! ⟶ one bit changed $1 \to 0$, nothing to left changed
Nim-sum of result is $m_1 \oplus \cdots \oplus m_{i-1} \oplus m_i' \oplus m_{i+1} \oplus \cdots \oplus m_r$

$$= m_1 + \cdots + m_{i-1} + m_i \oplus x \oplus m_{i+1} \oplus \cdots \oplus m_r$$

$$= m_1 + \cdots + m_{i-1} + m_i \oplus m_{i+1} \oplus \cdots \oplus m_r \oplus x$$

$$= x \oplus x = 0$$

THM: Player has a winning move in Nim iff Nim-sum at start of turn is non-zero.

Proof: (strong induction on # of stones left)

Base case ($n=0$): The only game with 0 stones is already over, previous player took last stone and won, so no winning moves.

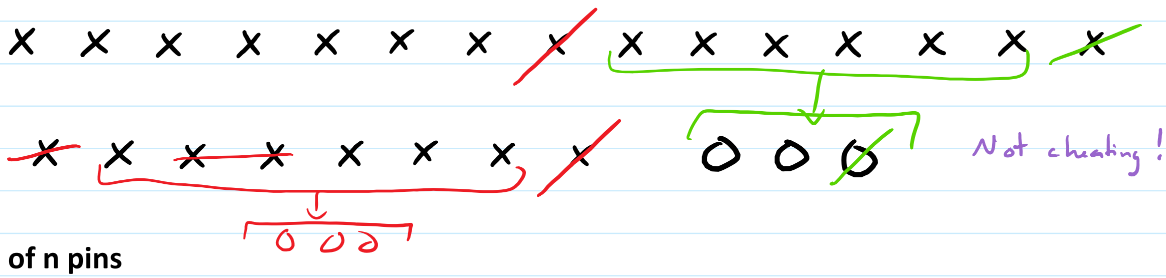
Ind step: Suppose $k > 0$ and all positions with i stones, $0 \le i < k$
have winning moves if and only if Nim-sum is non-zero

Two cases: a) Nim-sum is non-zero
Then there is a move to Nim-sum zero position with $k' < k$ stones
Other player then has no winning move
Move is a winning move

b) Nim-sum is zero
All moves are to Nim-sum non-zero positions with $k' < k$ stones
Other player has a winning response to all moves.
Original position is a losing position.

Kayles

0 5 10 15 20 25 30 35
0, 1, 2, 3, 1, 4, 3, 2, 1, 4, 2, 6, 4, 1, 2, 7, 1, 4, 3, 2, 1, 4, 6, 7, 4, 1, 2, 8, 5, 4, 7, 2, 1, 8, 6, 7



Start with row of n pins

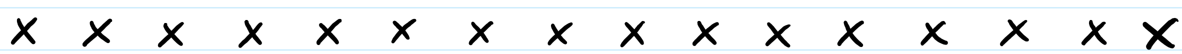
On each turn, take 1 or 2 adjacent pins

If no possible moves, you lose

(normal

last move loses = misère)

impartial! possible moves don't depend on turn



Using Sprague-Grundy

Sprague-Grundy Theorem: every finite, normal, impartial combinatorial game is equivalent to some form of 1-row Nim.

Nimber *n = game of 1-row Nim w/ n stones

Corollary: If G is equivalent to *n and H is equivalent to *m then G+H is equivalent to *(n⊕m)

For finite, normal, impartial games:

- game-over position = *0
- for each other position P in order of increasing length (max # moves to end)
  ↳ or any order s.t. when you get to a position, have already iterated through resulting positions
  - start with S = empty set
  - for each move
    - determine resulting position P'
    - look up what P' is equivalent to, add to S
  - compute mex(S), save that as equivalent to P
    ↳ minimum excludant = min non-neg integer not in set

Moving coins:

- start with coins in row of spaces 0, ..., n-1; at most one coin per space
- players take turns moving any coin to an unoccupied space to the left
- last move wins

PLEASE DOUBLE-CHECK !

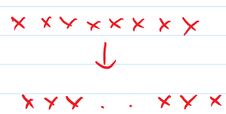
---OO

OO--- *0

≤ 1 move	O-O--	{OO---} {*0}	*1
≤ 2 moves	-OO--	{O-O--,OO---} {*1, *0}	*2
	O--O-	{OO--, O-O--} {*0, *1}	*2
≤ 3 moves	-O-O-	{O--O-, OO---, -OO--} {*2, *0, *2}	*1
	--OO-	{O--O-, -O-O-, O-O--, -OO--} {*2, *1, *1, *2}	*0
	O---O	{OO---, O-O--, O--O-} {*0, *1, *2}	*3
	-O--O	{O---O, OO---, -OO--, -O-O-} {*3, *0, *2, *1}	*4
	--O-O	{O---O, -O--O, O-O--, -OO--, --OO-} {*3, *4, *1, *2, *0}	*5
	---OO	{O---O, -O--O, --O-O, O--O-, -O-O-, --OO-} {*3, *4, *5, *2, *1, *0}	*6

For Kayles/Nim-like games (reducing number of objects in a pile or splitting piles;
can't move on more than one pile at a time [easily decomposed into subgames])

- game-over position = *0
- for each other initial pile P in order of increasing size
  - start with S = empty set
  - for each move
    - determine resulting position P', write as p1 + p2 + ... + pn (objects left in each pile)
    - look up what each pi is equivalent to, compute exclusive-or of all; add result to S
  - compute mex(S), save that as equivalent to P



Finding winning move

Finding winning move

0, 1, 2, 3, 1, 4, 3, 2, 1, 4, 2, 6, 4, 1, 2, 7, 1, 4,
3, 2, 1, 4, 6, 7, 4, 1, 2, 8, 5, 4, 7, 2, 1, 8, 6, 7

① compute table as above up to size of largest group

x x x x x · x x x x x x · x x x · x x

② look up equivalence for each group

③ compute xor

④ find group that has 1 in same place as MSB of xor

⑤ compute xor of that group and result from ③

x x x x x   x x x x x x   x x x x x x x x x x x

· · x x x x x x x x x

x · · x x x x x x x x

x x · · x x x x x x x

x x x · · x x x x x x