Game Theory

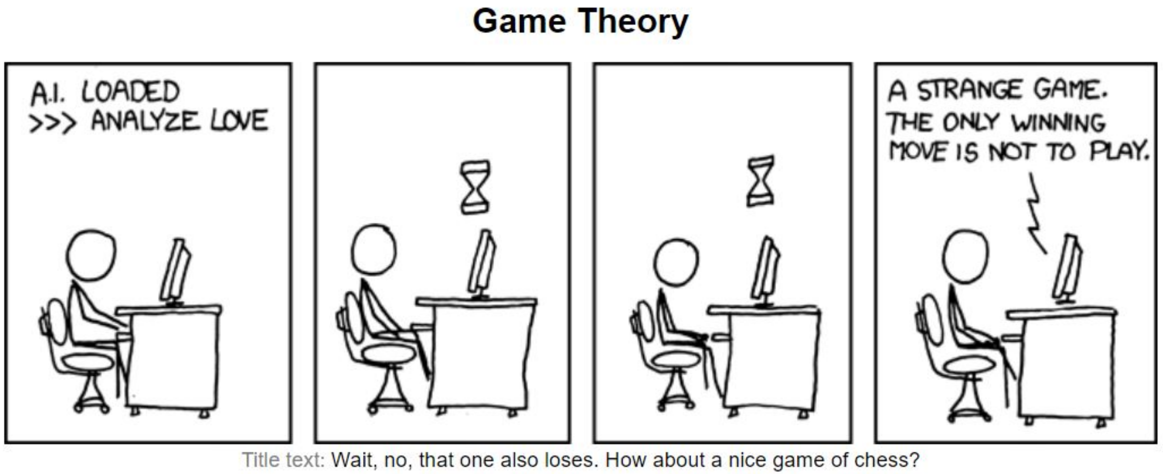
Ethen Yuen {ethening}

2021-04-24

香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Games (?)

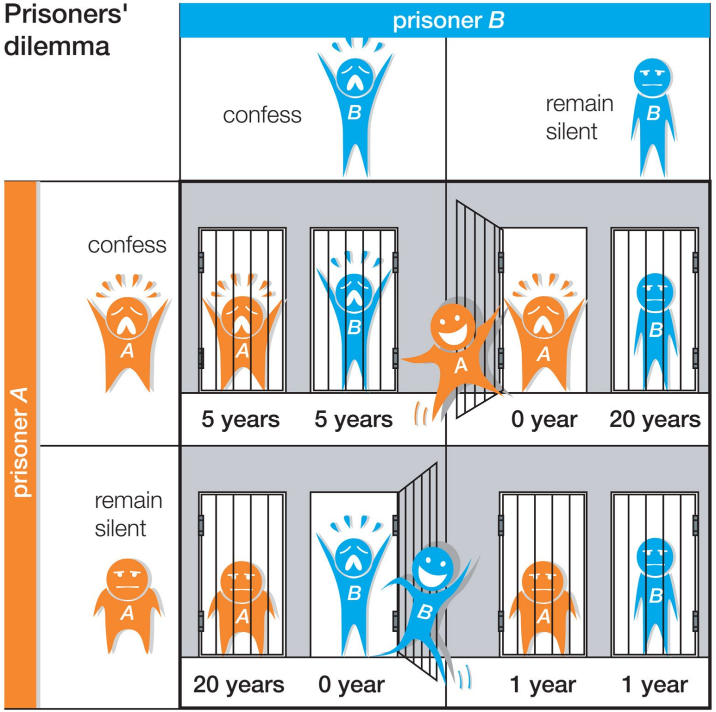
- What kinds of games?



https://xkcd.com/601/

Games Theory

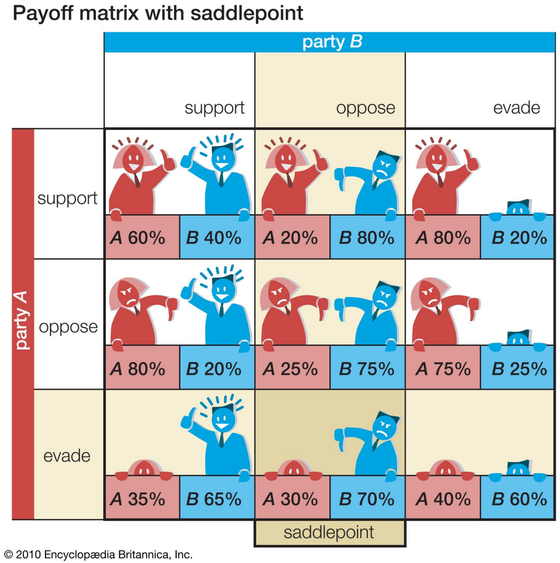
- You may have heard of Prisoner's Dilemma?

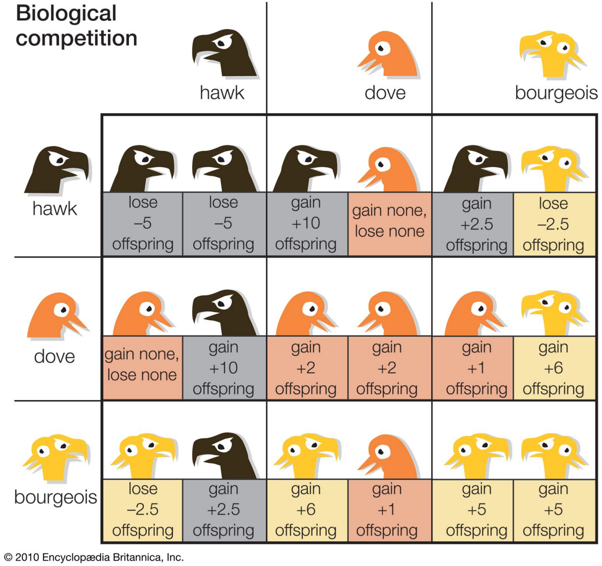


https://www.britannica.com/science/game-theory/The-prisoners-dilemma

Games Theory

- These games are important in various areas (economy, biology, sociology, ...), but not our main focus today.





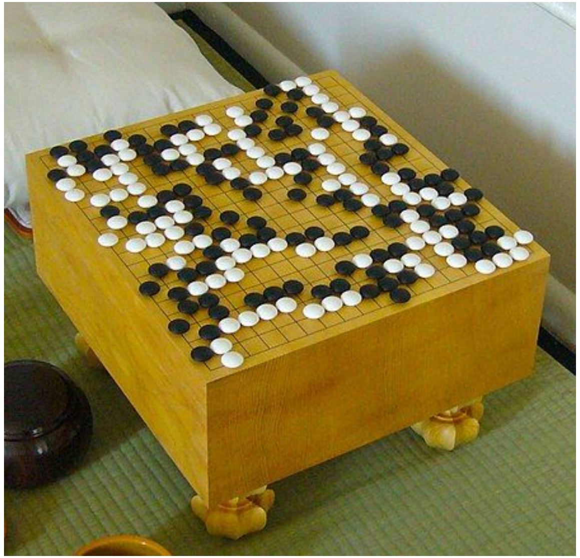
https://www.britannica.com/science/game-theory/The-prisoners-dilemma

Combinatorial Games

- A branch of game theory that frequently appears in competitive programming / computer science
- Usually do not involved probabilities, unlike mainstream game theory

- **Sequential Game**: Players take turns to change the game state with moves
- **Perfect Information**: No hidden or chance moves
- **Progressively Finite**: The game ends in a finite number of moves
- Two Players (usually)

Common examples

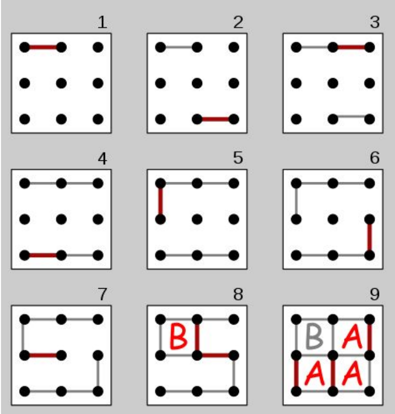


- Tic-tac-toe, Chess, Go

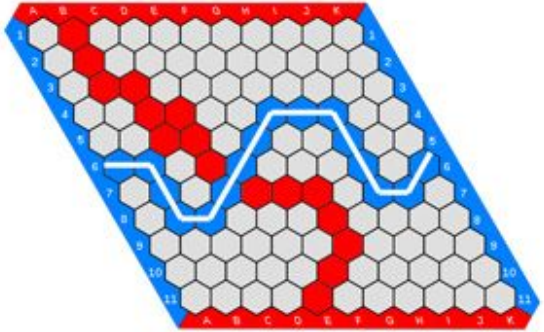
(Not so) Common examples

- Many games appear in *Clubhouse Games: 51 Worldwide Classics* are combinatorial games

Mancala



Dots and Boxes

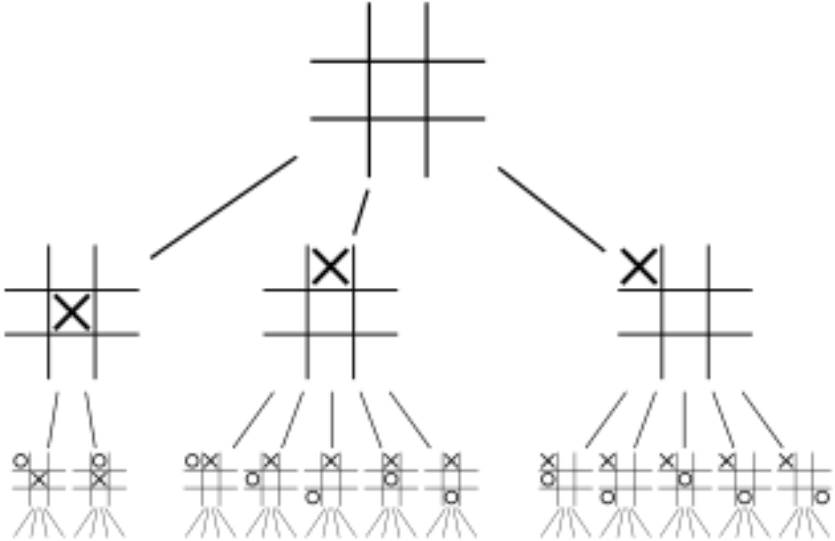


Hex

Beating CPU highest difficulties in these games:

https://www.youtube.com/watch?v=2AXEur7pe_s

Game State & Game Tree

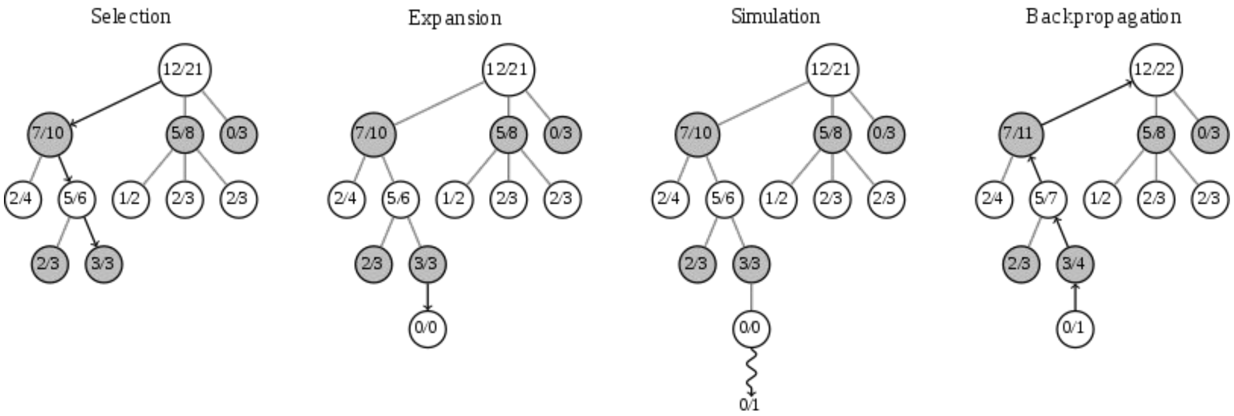


Can be tuned into a DAG

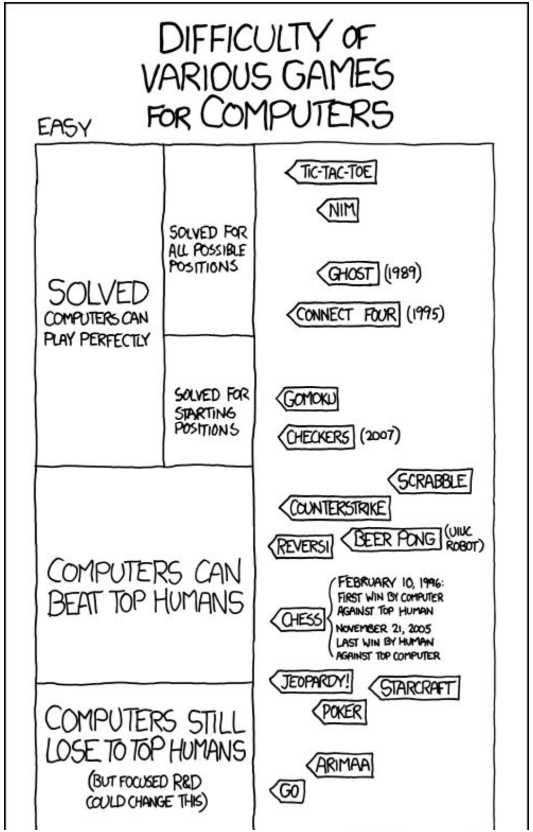
- Each **node** represents a state in game, each **edge** represents a valid move from that state.
- Leaf nodes (Terminal nodes) represent an ending game state: resulting in win / lose / draw.
- For larger games, only a partial game tree may be search to improve chances of picking a best move.

Deep Blue & Alpha Go

- **Deep Blue** (IBM): Chess → Mostly by brute force
- **AlphaGo** (Google Deepmind): Go → Monte Carlo Search Tree



[Monte Carlo Tree Search (MCTS) Tutorial]



(2012) https://xkcd.com/1002/

Solving a game

- Find the outcome of the game (win / lose / draw), considering both players play optimally.
- (Optional) Finding the best possible move(s) of some game states

- Zermelo's theorem:
- **Drawing strategy**: In any finite sequential game with perfect information, at least one of the players has a drawing strategy.
- **Winning strategy**: If a game never ends with a draw, then exactly one of the players has a winning strategy.

Impartial Combinatorial Game

- **Impartial**: At any particular position, both players have the same set of available moves.
  - Nim is impartial.
  - Mancala, Chess, Go and Tic-tac-toe are not.

Normal & Misère game play

- There are (one or more) ending positions in the games we consider.

- **Normal**: first to reach an ending position wins.
- **Misère**: first to reach an ending position loses.

- Unless otherwise specified, we assume the normal game play.

Game 1: Take-away Game

- Start with a pile of **N** stones.
- In each move, a player can remove any number of stones from **1 to K**.

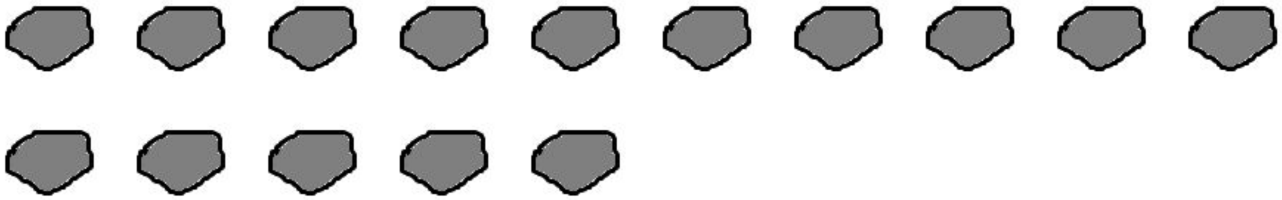
- Player removes the last stone wins.
- Ending position: 0 stones left

Game 1: Sample Play

- N = 15, K = 3

Player 1

Player 2

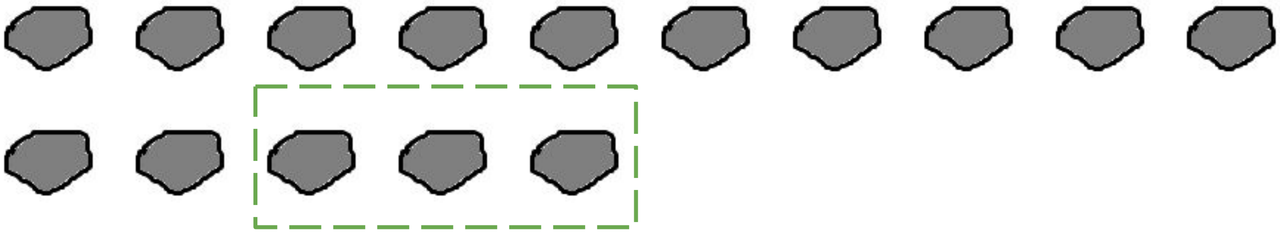


Game 1: Sample Play

- N = 15, K = 3

Player 1

Player 2

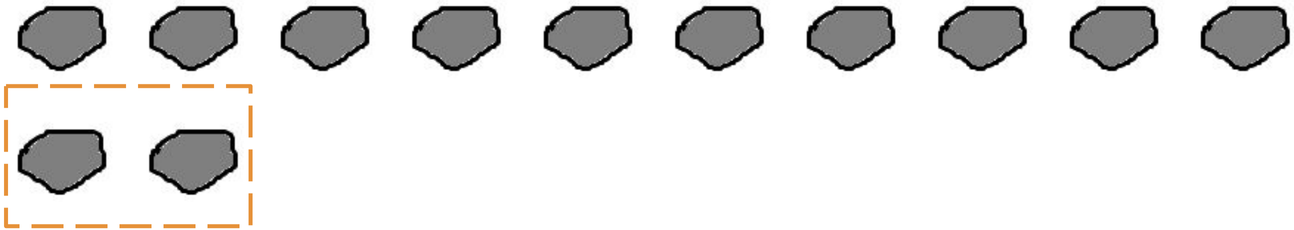


Game 1: Sample Play

- N = 15, K = 3

Player 1

Player 2

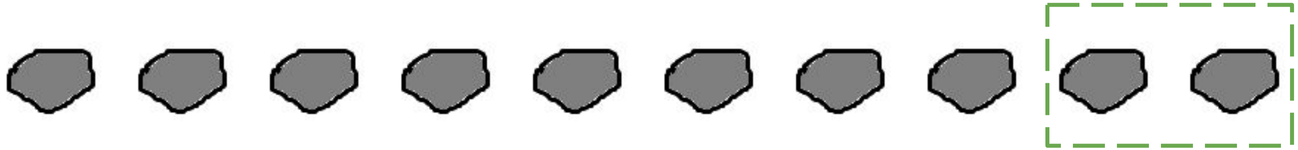


Game 1: Sample Play

- N = 15, K = 3

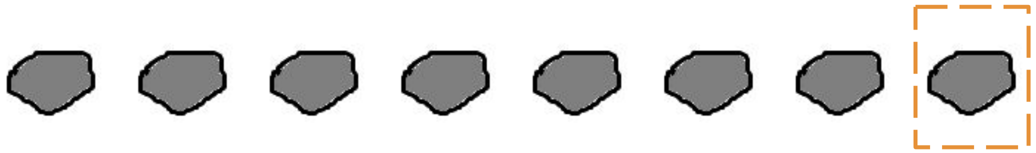
Player 1

Player 2



Game 1: Sample Play

- N = 15, K = 3



Player 1

Player 2

Game 1: Sample Play

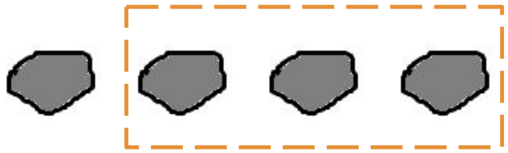
- N = 15, K = 3

Player 1

Player 2

Game 1: Sample Play

- N = 15, K = 3



Player 1

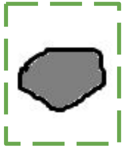
Player 2

Game 1: Sample Play

- N = 15, K = 3

Player 1

Player 2



player 1 wins!!!

- The result has been determined from the initial state (N, K) = (15, 3).
- Under perfect play, Player 1 always win.

Game 1: Try it out

- https://www.ictgames.com/mobilePage/bottleTakeAway/index.html

Game 1: Solution

- Suppose **K** = 3, we can try the game with small **N**.

- Which player has the winning strategy?

N	**0**	1	2	3	**4**	5	6	7	**8**	9	10	11
	P2	P1	P1	P1	**P2**	P1	P1	P1	**P2**	P1	P1	P1

- If N is a **multiple of 4**, Player 2 wins.
- Otherwise, Player 1 wins.

Game 1: Winning Strategy

- If N is a **multiple of 4**, Player 2 wins.
- Otherwise, Player 1 wins.

- Why is this true?
- If N is a multiple of 4,
  - Suppose Player 1 takes **X** stones ($1 \leq X \leq 3$).
  - Player 2 should always respond by taking **(4 - X)** stones.
- After Player 1 turns, number of stones $\neq 0 \pmod 4$, never the ending state
- After Player 2 turns, number of stones $= 0 \pmod 4$

Game 1: Winning Strategy

- If N is a **multiple of 4**, Player 2 wins.
- Otherwise, Player 1 wins.

- If N is a multiple of 4,
  - Suppose Player 1 takes **X** stones ($1 \le X \le 3$).
  - Player 2 should always respond by taking **(4 - X)** stones.
- If $N = 4k + X$ ($1 \le X \le 3$),
  - Player 1 should take **X** stones.
  - Then, use the strategy above.

Game 1: Solution

- If N is a **multiple of 4**, Player 2 wins.
- Otherwise, Player 1 wins.

- General solution:
- If N is a **multiple of (K + 1)**, Player 2 wins.
- Otherwise, Player 1 wins.

P-position and N-position

- **P**-position: The **P**revious player has a winning strategy
- **N**-position: The **N**ext player has a winning strategy.

- Under normal play rule, the player first to reach an ending position wins.
- So, every ending position is a **P**-position.

- If initial position is a **N**-position, Player 1 wins.
- If initial position is a **P**-position, Player 2 wins.

P-position and N-position - Example

- For Take-away game with K = 3:
- Previously we use:

N	0	1	2	3	4	5	6	7	8	9	10	11
	P2	P1	P1	P1	P2	P1	P1	P1	P2	P1	P1	P1

- Now we use:

N	0	1	2	3	4	5	6	7	8	9	10	11
	P	N	N	N	P	N	N	N	P	N	N	N

P-position and N-position

- How do we determine whether a given position is P or N?

- Every ending position is a **P**-position.
- A position that has a way to move to a **P**-position is a **N**-position.
- A position that can **only** move to a **N**-position is a **P**-position.

P-position and N-position - Example

- For Take-away game with K = 3:

N	0	1	2	3	4	5	6	7	8	9	10	11
	P											

- N = 0 is an ending position → P

P-position and N-position - Example

- For Take-away game with K = 3:

N	0	1	2	3	4	5	6	7	8	9	10	11
	P	N	N	N								

- N = 1, 2, 3 has a way to move to P-position (N = 0) → N

P-position and N-position - Example

- For Take-away game with K = 3:

N	0	1	2	3	4	5	6	7	8	9	10	11
	P	N	N	N	P							

- N = 4 can only move to N-position (N = 1, 2, 3) → P

P-position and N-position - Example

- For Take-away game with K = 3:

N	0	1	2	3	4	5	6	7	8	9	10	11
	P	N	N	N	P	N	N	N				

- N = 5, 6, 7 has a way to move to P-position (N = 4) → N

P-position and N-position - Example

- For Take-away game with K = 3:

N	0	1	2	3	4	5	6	7	8	9	10	11
	P	N	N	N	P	N	N	N	P			

- N = 8 can only move to N-position (N = 6, 7, 8) → P

P-position and N-position - Example

- For Take-away game with K = 3:

N	0	1	2	3	4	5	6	7	8	9	10	11
	P	N	N	N	P	N	N	N	P	N	N	N

Blank slide

- Rest / Q&A

Game 2: Nim

- There are **N** piles of stones.
- The **i-th** pile contains **a[i]** stones.

- On each turn, a player may remove any positive number of stones from any non-empty pile.

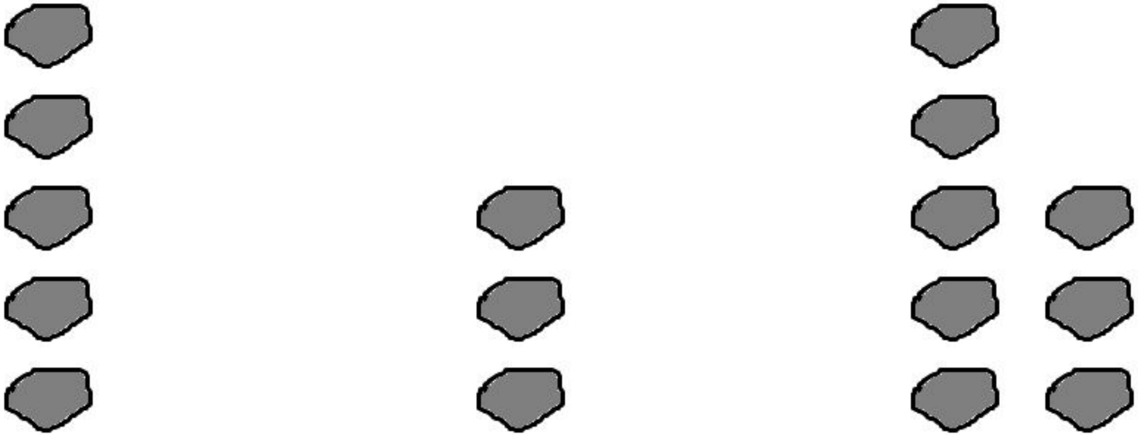
- Player removes the last stone wins.
- Ending position: 0 stones left

Game 2: Sample Play

- N = 3, a[1] = 5, a[2] = 3, a[3] = 8

Player 1

Player 2

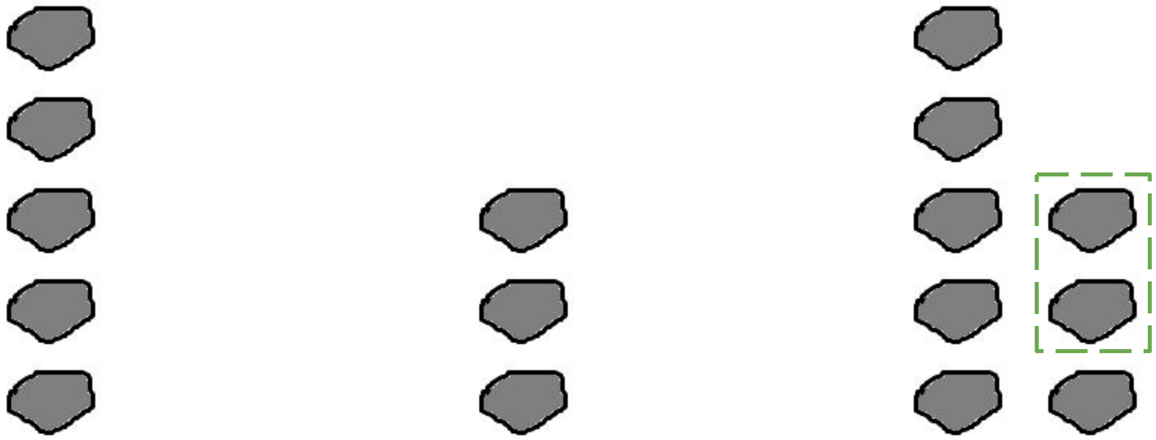


Game 2: Sample Play

- N = 3, a[1] = 5, a[2] = 3, a[3] = 8

Player 1

Player 2

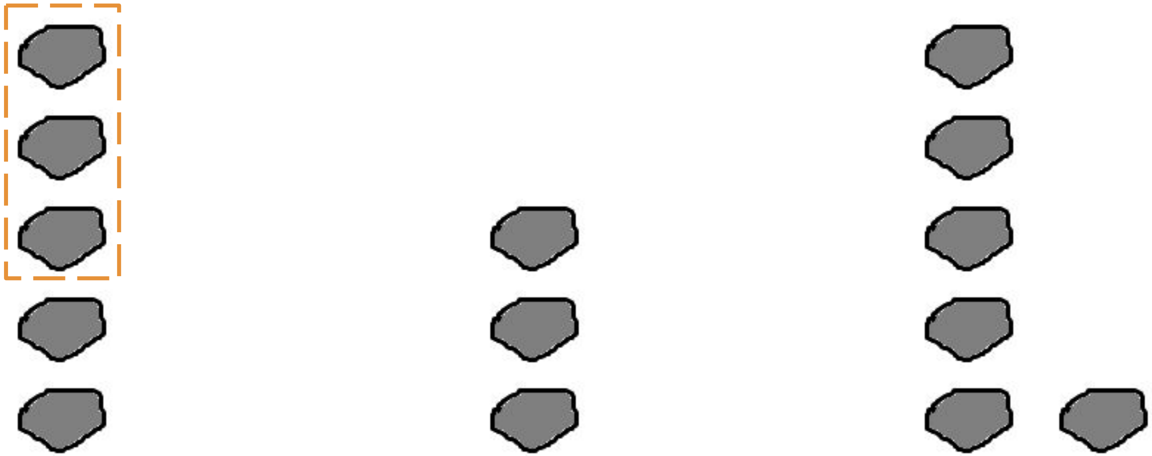


Game 2: Sample Play

- N = 3, a[1] = 5, a[2] = 3, a[3] = 8

Player 1

Player 2

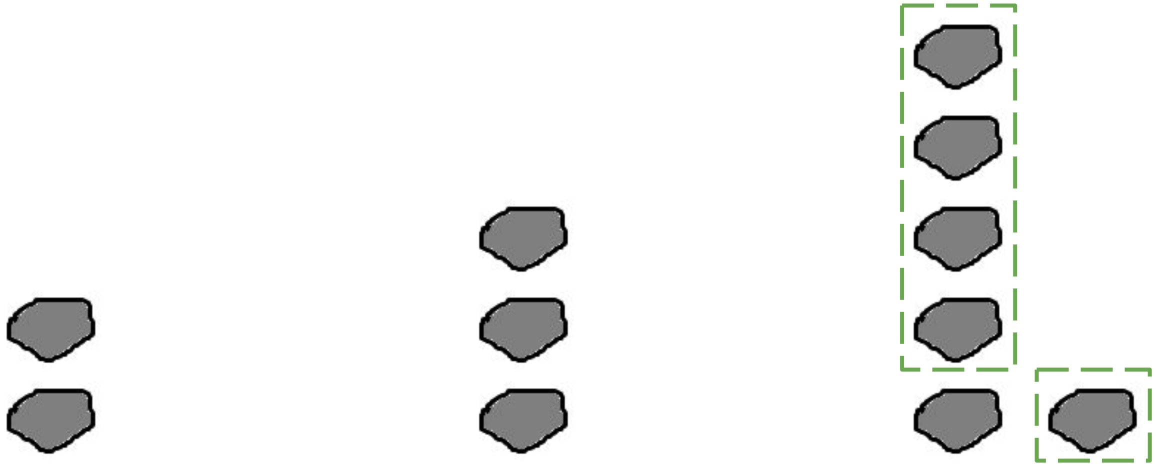


Game 2: Sample Play

- $N = 3$, $a[1] = 5$, $a[2] = 3$, $a[3] = 8$

Player 1

Player 2

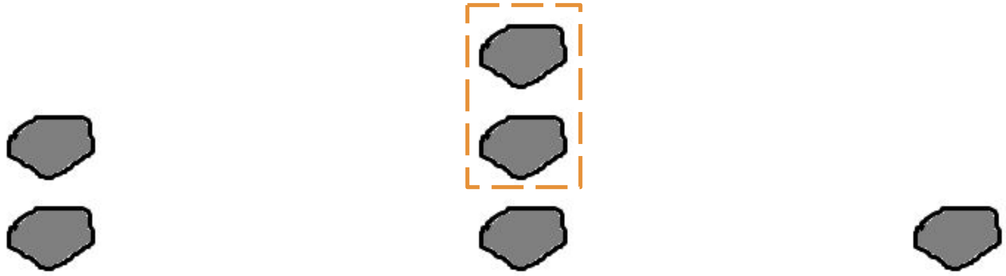


Game 2: Sample Play

- N = 3, a[1] = 5, a[2] = 3, a[3] = 8

Player 1

Player 2



Game 2: Sample Play

- N = 3, a[1] = 5, a[2] = 3, a[3] = 8

Player 1

Player 2

Game 2: Sample Play

- N = 3, a[1] = 5, a[2] = 3, a[3] = 8

Player 1

Player 2

Game 2: Sample Play

- N = 3, a[1] = 5, a[2] = 3, a[3] = 8

Player 1

Player 2

- As with the previous game, the result has been determined from the initial state.
- Under perfect play, Player 1 always win.

player 1 wins!!!

Game 2: Nim

- Consider cases where N is small:

- If **N = 1**, (a[1] > 0,) Player 1 wins.
- If **N = 2**,
  - If a[1] = a[2], Player 2 wins (**Strategy Stealing**: mirror the moves of Player 1)
  - If a[1] != a[2], Player 1 wins (Makes 2 piles the same)

- What about bigger N? Are we able to do case handling for all of those?

Game 2: Nim Sum

- (about to get a bit mathy)
- For each game state **g**, assign to it some integers **G** as its “Nim-value”
- Given two game state **a** and **b**, let **a + b** be the game state with piles of **a** put together with piles of **b**.
  If their Nim-values are **A** and **B**, we define an operator ⊕, where **A ⊕ B** is the Nim-value of state **a + b**.

- We hope there exists some operator that works nicely for ⊕.

Game 2: Nim Sum

- Consider the simplest possible game **N = 1** again.

- Intuitively, since the only information that distinguishes the piles from one another is the number of stones in them, let's say the Nim-value of this game is simply **a[1]**.
- We can see that it is a N-position if **a[1] > 0**, and a P-position if **a[1] = 0**. We would like to reflect this in Nim-value as well → a Nim-value **G** indicates a N-position if **G > 0**, and a P-position if **G = 0**.

Game 2: Nim Sum

- We also can deduce some properties of the operator ⊕ composing Nim-values together.

- $(A \oplus B) \oplus C = A \oplus (B \oplus C)$ and $A \oplus B = B \oplus A$
  Associative and commutative - order shouldn't matter for combining piles.
- $A \oplus 0 = A$
  Operator identity equal 0 - adding a empty pile should not affect the game
- $A \oplus A = 0$
  Inverse of each state is itself - strategy stealing

Game 2: Nim Sum

- $(A \oplus B) \oplus C = A \oplus (B \oplus C)$ and $A \oplus B = B \oplus A$
- $A \oplus 0 = A$
- $\boldsymbol{A \oplus A = 0}$

- Ordered n-tuple of {0, 1} with operation of addition modulo 2 between element?
- $A \oplus B$ is basically the **bitwise XOR** of the two Nim-values (also known as Nim-sum)

Game 2: General Solution

- If $a[1] \oplus a[2] \oplus \ldots \oplus a[N] = 0$,
  - It is a P-position (Player 2 wins).

- If $a[1] \oplus a[2] \oplus \ldots \oplus a[N] \neq 0$
  - It is a N-position (Player 1 wins).

Game 2: Winning Strategy

- If $a[1] \oplus a[2] \oplus \ldots \oplus a[N] = 0$,
  - After Player 1's move, $a[1] \oplus a[2] \oplus \ldots \oplus a[N] \neq 0$
  - There exists a move for Player 2 so that after the move $a[1] \oplus a[2] \oplus \ldots \oplus a[N] = 0$ (Proof next slide)

- If $a[1] \oplus a[2] \oplus \ldots \oplus a[N] \neq 0$
  - Player 1 should move to make $a[1] \oplus a[2] \oplus \ldots \oplus a[N] = 0$
  - Then, use the strategy above.

Game 2: Why does it work?

- If Nim-sum is not zero,

```
  xxxxxxxxxx
  xxxx1xxxxx
  xxxxxxxxxx
⊕ xxxxxxxxxx
--------------------
  00001xxxxx
```

- Find the most significant 1
- There exists a number with ‘1’ in that place
- Reduce that number so that xor-sum is zero

Game 2: Why does it work?

- e.g. 5, 3, 8

```
   0101
   0011
⊕ 1000
-------------
   1110
```

- **Find the most significant 1**
- There exists a number with ‘1’ in that place
- Reduce that number so that xor-sum is zero

Game 2: Why does it work?

- e.g. 5, 3, 8

```
  0101
  0011
⊕ 1000
-------------
  1110
```

- Find the most significant 1
- **There exists a number with '1' in that place**
- Reduce that number so that xor-sum is zero

Game 2: Why does it work?

- e.g. 5, 3, 8

```
    0101
    0011
⊕ 1000 → 0110
-------------
    1110
```

- Find the most significant 1
- There exists a number with ‘1’ in that place
- **Reduce that number so that xor-sum is zero**

Game 2: Mini-test

- In a Nim game, suppose N = 4, a[1] = 35, a[2] = 18, a[3] = 27.

- In order to win, what should Player 2 choose for a[4]?

Game 2: Mini-test Answer

- In a Nim game, suppose N = 4, a[1] = 35, a[2] = 18, a[3] = 27.

- Player 2 wants $a[1] \oplus a[2] \oplus a[3] \oplus a[4] = 0$,
- That is, $a[4] = a[1] \oplus a[2] \oplus a[3] = 35 \oplus 18 \oplus 27 =$ **42**

- Note that the answer is unique.

Importance of Nim Game

- Nim is considered to be the prototypical game among the impartial combinatorial games

- Other games can be analyzed using the idea of Nim!

- Some variant of Nim:

  Fibonacci Nim - https://en.wikipedia.org/wiki/Fibonacci_nim

  Kayles (Circular Nim) - https://en.wikipedia.org/wiki/Kayles

Blank slide

- Rest / Q&A

Game 3: Nim Game with addition

- There are **N** piles of stones. The **i-th** pile contains **a[i]** stones.
- The normal rules of Nim apply, except there is an extra move: instead of removing stones from a pile, the current turn player may choose to **spend their turn adding any number of stones to a pile** instead. They can add as many stones as they want, so long as, again, they are all added to the same pile.
- Assume this addition move may only be done a **finite** number of times per player.

Game 3: Solution

- Turns out this game is exactly the same as normal Nim.

- If the current turn player is in a losing state, under the normal set of Nim rules,
  - If they add **s** stones to some pile, the other player can just spend their turn removing **s** stones from that same pile, effectively undoing the last move.
- A winning player, meanwhile, can just play the game like normal Nim, only deviating from the strategy to **undo any additions** the opposing player makes, as necessary.
- Since the game will **still end after a finite number of moves** (and this is important), the winning player can be determined like normal Nim.

Game 4: Square-number Nim Game

- There are **N** piles of stones. The **i-th** pile contains **a[i]** stones.
- The normal rules of Nim apply, except there is an restriction: a player can only take a perfect square number of stones from a pile.
- A lot more complicated than the original game.

Game 4: Square-number Nim Game

- Common Tweak: Putting limitations / conditions on taking stone.

- Back to our original strategy: decompose the game state into piles, find some properties for each pile, and composing them together to get the answer.

- Consider a single pile with **p** stones first.

Game 4: Directed Graph

- Consider a single pile with **p** stones first.
- How to determine if it is a P-position or N-position?

- Construct a **directed graph** with **p + 1** nodes, labelled **0** to **p**.
- Node = game state, label = how many stones in the pile at the state.
- Add a directed edge from **u to v** if there exist **valid transition** from u stones to v stones.

- In this game, node 11 would have directed edge to node 10, 7, and 2, corresponding to taking 1, 4, or 9 stones.

Game 4: Dynamic Programming

- We could solve this by dynamic programming then.
  - The terminal vertices (outdegree = 0) are P-positions.
  - If a node points to at least one P-position, it is a N-position.
  - If a node **only** points to N-position, it is a P-position.

- If there are no cycles in the graph (the game ends in a finite number of moves), the DP could work by updating in topological order.

- Instead of just assigning P/N-position to node, let's assign some integer value to each node like what we do with Nim-values from earlier.

Sprague-Grundy function

- Usually abbreviated as SG function.

- Suppose current game state is X.
- If X is a terminal state, SG(X) = 0.
- Otherwise,
  - If positions x1, x2,, ..., xk can be reached from the current state X,
  - **SG(X) = mex({SG(x1), SG(x2), ..., SG(xk)})**

MEX (Minimum EXcluded)

- MEX of a set of non-negative integers is the smallest non-negative integer **not** in the set.

Examples:

- mex({0, 2, 4}) = 1
- mex({0, 5, 1, 3}) = 2
- mex({2, 0, 1}) = 3
- mex({1, 3, 5}) = 0
- mex({all positive integers}) = 0
- mex(empty set) = 0

SG-function: Example

- For Take-away game with K = 3:

N	0	1	2	3	4	5	6	7	8	9	10	11
SG	0											

- N = 0 is an ending position → SG(0) = mex({}) = 0

SG-function: Example

- For Take-away game with K = 3:

N	0	1	2	3	4	5	6	7	8	9	10	11
SG	0	1										

- SG(1) = mex({SG(0)}) = mex({0}) = 1

SG-function: Example

- For Take-away game with K = 3:

N	0	1	2	3	4	5	6	7	8	9	10	11
SG	0	1	2									

- SG(2) = mex({SG(0), SG(1)}) = mex({0, 1}) = 2

SG-function: Example

- For Take-away game with K = 3:

N	0	1	2	3	4	5	6	7	8	9	10	11
SG	0	1	2	3								

- SG(3) = mex({SG(0), SG(1), SG(2)}) = mex({0, 1, 2}) = 3

SG-function: Example

- For Take-away game with K = 3:

N	0	1	2	3	4	5	6	7	8	9	10	11
SG	0	1	2	3	0							

- SG(4) = mex({SG(1), SG(2), SG(3)}) = mex({1, 2, 3}) = 0

SG-function: Example

- For Take-away game with K = 3:

N	0	1	2	3	4	5	6	7	8	9	10	11
SG	0	1	2	3	0	1						

- SG(5) = mex({SG(2), SG(3), SG(4)}) = mex({2, 3, 0}) = 1

SG-function: Example

- For Take-away game with K = 3:

N	0	1	2	3	4	5	6	7	8	9	10	11
SG	0	1	2	3	0	1	2	3	0	1	2	3

- SG(6) = mex({SG(3), SG(4), SG(5)}) = mex({3, 0, 1}) = 2
- SG(7) = mex({SG(4), SG(5), SG(6)}) = mex({0, 1, 2}) = 3
- SG(8) = mex({SG(5), SG(6), SG(7)}) = mex({1, 2, 3}) = 0
- SG(9) = mex({SG(6), SG(7), SG(8)}) = mex({2, 3, 0}) = 1
- SG(10) = mex({SG(7), SG(8), SG(9)}) = mex({3, 0, 1}) = 2
- SG(11) = mex({SG(8), SG(9), SG(10)}) = mex({0, 1, 2}) = 3

SG-function: Comparing with P/N-position

- For Take-away game with K = 3:

N	0	1	2	3	4	5	6	7	8	9	10	11
SG	0	1	2	3	0	1	2	3	0	1	2	3

N	0	1	2	3	4	5	6	7	8	9	10	11
	P	N	N	N	P	N	N	N	P	N	N	N

SG function: Properties

- If SG value = 0,
  - It is a P-position (Player 2 wins).
- If SG value ≠ 0
  - It is a N-position (Player 1 wins).
- Any ending position has SG value 0

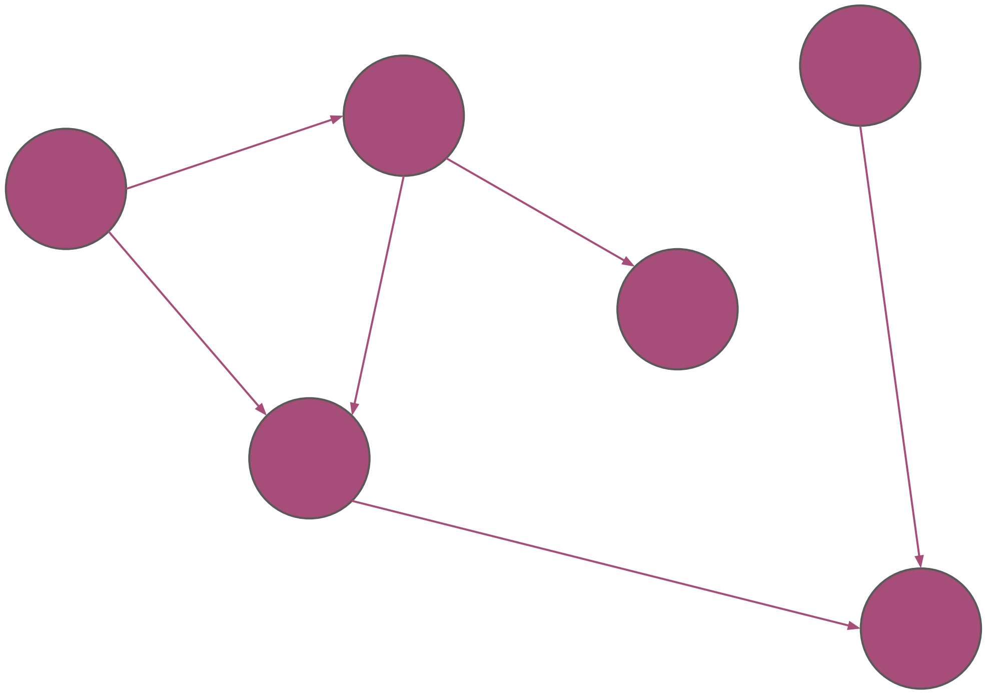
SG function: Mini-test

- Given game represented by a DAG (directed acyclic graph).

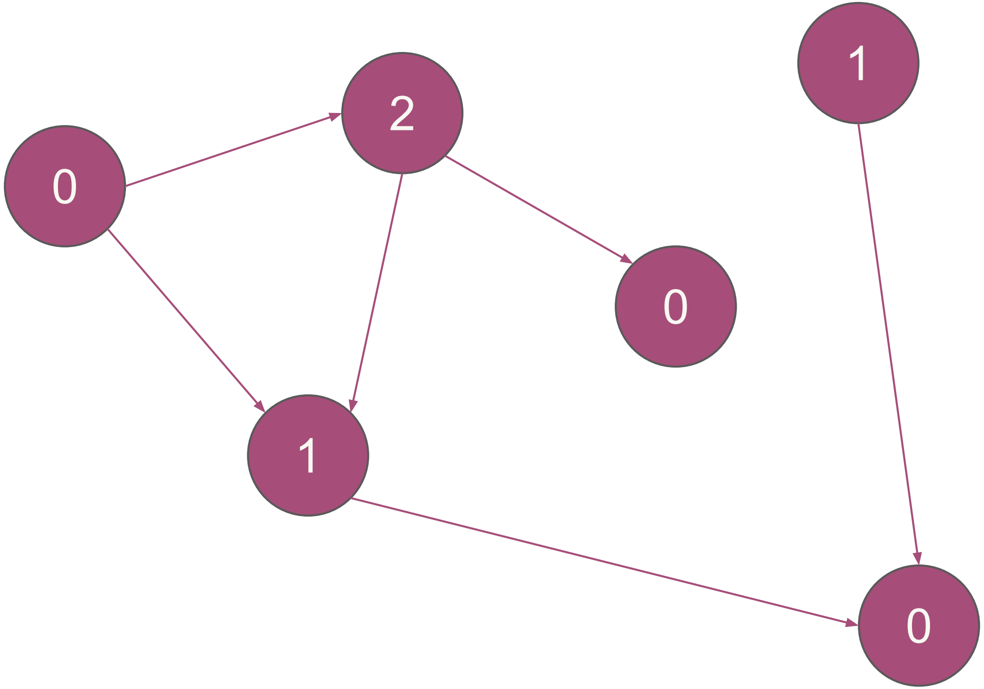
- Node = Game State
- Edge from Node A to B = one can move from state A to state B

- Fill in the SG values of each state

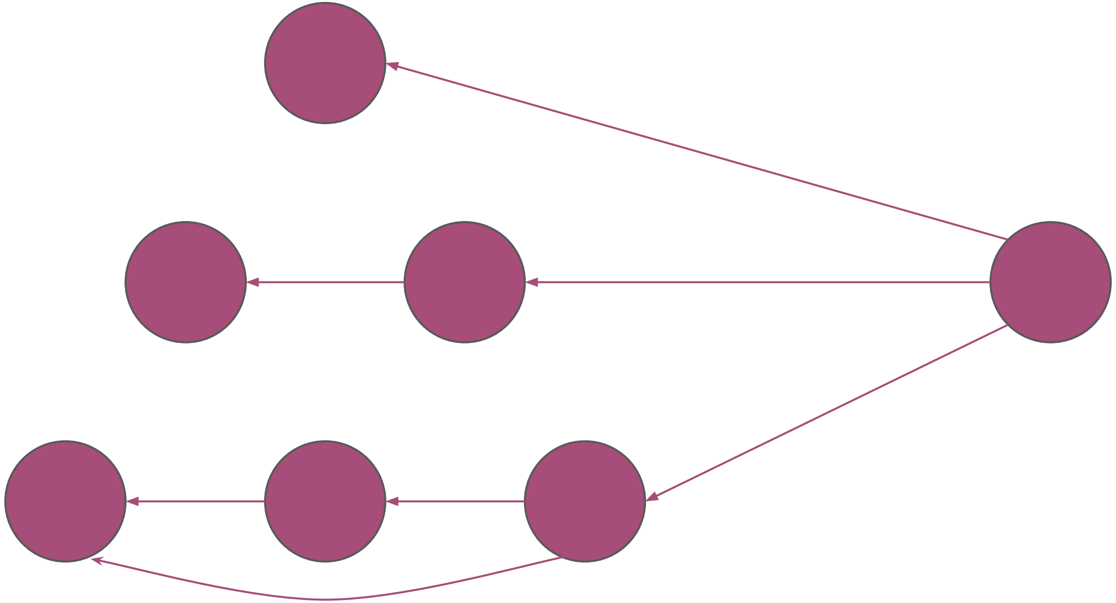
SG function: Mini-test 1



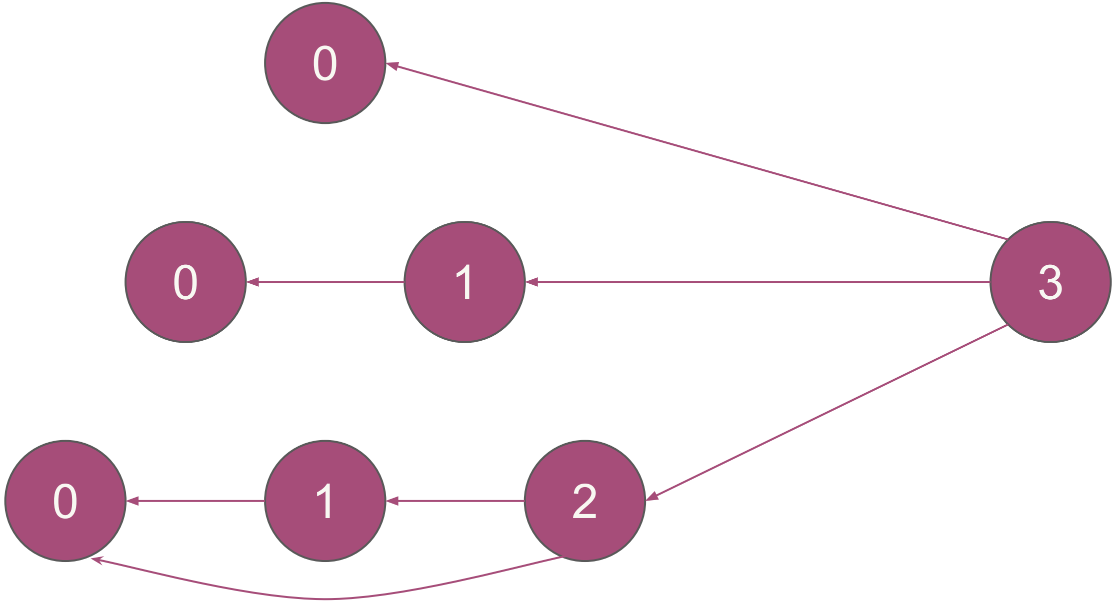
SG function: Mini-test 1 Solution



SG function: Mini-test 2



SG function: Mini-test 2 Solution



SG function: Computation Order

- Any topological ordering works fine.

- In most combinatorial game, the ordering is just 1, 2, 3, ...

Game 4: SG number

- Back to our Square-number Nim Game
- For game with a single pile, we could assign to each state the SG number as denoted above.
- E.g. **SG(5) = mex({SG(5 - 1), SG(5 - 4)})** = mex(SG(4), SG(1)) = mex(2, 1) = **0**, which is a P-position (try work this out without SG number).

- How about when there are multiple piles?
- By Mathemagics, we can actually apply the same strategy with Nim-values: **Taking bitwise XOR of each piles' SG number**.

Sprague-Grundy Theorem

- Suppose we have N independent games, G[1], G[2], ..., G[N], each turn, a player can pick one game and make a valid move in that game.
- Suppose we have analyzed each independent game using SG function.

- Let P[i] be the position in subgame G[i],
- **SG(P[1], P[2], ..., P[N]) = SG(P[1]) ⊕ SG(P[2]) ⊕ ... ⊕ SG(P[N])**

Game 4: SG number

- For the Square-number Nim Game, we can see that,

N	0	1	2	3	4	5	6
SG	0	1	0	1	2	0	1

- If we have a game state with piles of 1, 2, 3, 4, 6 stones, Nim-sum of this game is SG(1) $\oplus$ SG(2) $\oplus$ SG(3) $\oplus$ SG(4) $\oplus$ SG(6) = 1 $\oplus$ 0 $\oplus$ 1 $\oplus$ 2 $\oplus$ 1 = 3
- Player 1 has a winning strategy.
- For normal Nim game, since each state points to every state smaller than it, **SG(p) = p**, with aligned with our results in Game 2.

Game 4: SG number

- Why do this works though?
- The mex function can be viewed as a **promise**: if **SG(p) = g**, that every integer from **0 to g - 1 is available** in the list of valid moves from **p**.

- Then, we can corresponds every pile in our game from **having p stones with weird rules**, to **having SG(p) = g stones but with normal Nim rules**.

Game 4: SG number

- Then, we can corresponds every pile in our game from **having p stones with weird rules**, to **having SG(p) = g stones but with normal Nim rules**.

- This can be done because in Nim, we can transition from **a pile of size p** to **any pile of size less than p**.
- For the transformed SG number world, mex property guarantees that we can transition from **a ‘pile’ of size g** to **any ‘pile’ of size less than g**.
- That way, we can tackle any game like normal Nim, which we already know how to deal with.

Game 4: SG number

- Unlike in Nim, in our transformed game, it's possible to transition from a pile of size g to a pile of size **greater than g**.
- E.g. in Square-number Nim, SG(5) = 0, while SG(5 - 1) = SG(4) = 2, which increase a pile of **size 0 to size 2**.

- Still remember **Game 3** (Nim with addition)?
- If we transition from g to some **larger number h**, by properties of mex, we can visit any numbers from **0 to h - 1**. Since **g < h**, we can always transition back to **g**.
- Same as the undo steps in Game 3.

Impartial Combinatorial Game

- **Any impartial combinatorial game** that can be represented as a DAG can be formulated in terms of SG numbers and reduced to normal **Nim** game.
- It is all Nim after all!

- N independent games = N-pile Nim

Blank slide

- Rest / Q&A

Problem-solving session

- Let's solve some game theory problems!

- Solve = Describe an efficient algorithm to find the answer. Not necessarily something 'mathematical' or elegant.

Strategy

- Look for invariants.
- **Try small cases.**
- Analyse the game using SG function.
- Sum of games can be dealt with by Nim-sum ($\oplus$).
    - No need to worry about "N piles of stones" as long as they are independent.
- If you want to convince yourself, prove your results using induction.

Problem 1

- [Codeforces 705B]
- N (<= 100000) piles of stones.
- i-th pile has a[i] (≤ 1e9) stones.
- Each move = split a pile of X stone into two piles of Y and Z stones, where Y > 0, Z > 0, X = Y + Z

- For each K ≤ N, determine who wins if the game is played with the first K piles.

Problem 2

- [HKOI Junior J163]
- N (<= 50000) piles of stones.
- i-th pile has s[i] (≤ 1e6) stones.
- Each round, a value v[i] is decided.
- Each move,
  - add 1 stone to a pile < v[i] stone
  - take away 1 stone from a pile > v[i] stone

- For each v[i], determine who wins.

Problem 3

- [Codechef June CHCOINSG]
- One pile of N (≤ 1e9) stones.
- Each move = remove p^k stones, where p is prime and k is a non-negative integer.

- Determine who wins.

Problem 4

- [CP-algorithm Crosses-crosses]
- Given a strip of N ($\leq$ 100000) empty cells.
- Each move = put one stone into an empty cell, no two stones can be put into adjacent cells.

- Determine who wins.

Problem 5

- [Codeforces 36D]
- Given two piles of A and B stones (A, B ≤ 1e9).
- Each move = remove one stone from one pile or remove K (≤ 1e9) stones from both piles.

- Determine who wins.

Problem 6

- [Codeforces 1312F]
- N (≤ 300000) of stones.
- i-th pile has a[i] (≤ 1e18) stones.
- Each move = decrease a[i] (a[i] > 0) by X, Y or Z (≤ 5). (If X, Y, Z > a[i], it will set a[i] to 0)
- Additional restriction: Decrease by Y cannot be applied to the same pile continuously (same for Z).

- Determine who wins & the number of options (i, X/Y/Z) the first player can choose in his first move such that the first player can win.

Problem 7

- [AtCoder Grand Contest 010D]
- N (≤ 1e5) positive integers a[1] , a[2] , ..., a[N] (≤ 1e9).
- gcd(a[1] , a[2] , ..., a[N]) = 1.
- Each move = replace an a[i] (a[i] > 1) by a[i] - 1.
- Afterwards, divide each a[i] by gcd(a[1] , a[2] , ..., a[N]).

- Determine who wins.

Summary

- After this training session, you should be able to:

- Identify impartial combinatorial games;
- Use P-N positions/SG function to solve these games;
- Tackle “sum of games” and “splitting into subgames” with ease;
- HAVE FUN solving CGT problems ☺️

References

- HKOI training - Game Theory (2019) by Alex Tung
  https://assets.hkoi.org/training2019/game-theory.pdf

- Codeforces Blog - The Intuition Behind NIM and Grundy Numbers in Combinatorial Game Theory
  https://codeforces.com/blog/entry/66040

Other Fun Games, Puzzles & Related stuff

- Wythoff Game (explained by James Grime)
  https://www.youtube.com/watch?v=pzlpi7lJi4k
  https://www.youtube.com/watch?v=AYOB-6wyK_I
- Coin Flipping Puzzle (explained by Matt Parker & Grant Sanderson)
  https://www.youtube.com/watch?v=as7Gkm7Y7h4
  https://www.youtube.com/watch?v=wTJI_WuZSwE
- Prisoner Hat Puzzle
  https://www.youtube.com/watch?v=N5vJSNXPEwA (intro by TED-Ed)
  https://en.wikipedia.org/wiki/Induction_puzzles (many variants)