Analyzing ELLIE - the Story of a Combinatorial Game

S. Heubach[1] P. Chinn[2] M. Dufour[3] G. E. Stevens[4]

[1]Dept. of Mathematics, California State Univ. Los Angeles

[2]Dept. of Mathematics, Humboldt State University

[3]Dept. of Mathematics, Université du Quebeq à Montréal

[4]Dept. of Mathematics, Hartwick College

May 13, 2009
Math Colloquium, San Jose State University

Overview

Overview

Overview

Overview

Overview

How ELLIE was conceived

- P. Chinn, R. Grimaldi, and S. Heubach, Tiling with Ls and Squares, Journal of Integer Sequences, Vol 10 (2007)
- Phyllis and Silvia talk to Gary - the idea of a game is born
- Matthieu joins in and brings background in combinatorial games

How ELLIE was conceived

- P. Chinn, R. Grimaldi, and S. Heubach, Tiling with Ls and Squares, Journal of Integer Sequences, Vol 10 (2007)
- Phyllis and Silvia talk to Gary - the idea of a game is born
- Matthieu joins in and brings background in combinatorial games

How ELLIE was conceived

- P. Chinn, R. Grimaldi, and S. Heubach, Tiling with Ls and Squares, Journal of Integer Sequences, Vol 10 (2007)
- Phyllis and Silvia talk to Gary - the idea of a game is born
- Matthieu joins in and brings background in combinatorial games

Description of ELLIE

ELLIE is played on a rectangular board of size m-by-n. Two players alternately place L-shaped tiles of area 3. Last player to move wins (normal play).

Questions:

- For which values of m and n is there a winning strategy for Player I?
- What is the winning strategy?

Combinatorial Games

Definition

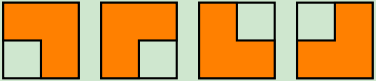
An *impartial combinatorial game* has the following properties:

- no randomness (dice, spinners) is involved, that is, each player has **complete information** about the game and the potential moves
- each player has the **same moves** available at each point in the game (as opposed to chess, where there are white and black pieces).

Working out small examples

Example (The 2×2 board)

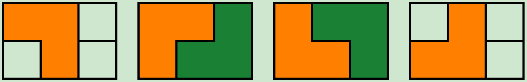
First player obviously wins, since only one L can be placed. In each case, the second player only finds one square left, which does not allow for placement of an L.



Working out small examples

Example (The 2×3 board)

First player's move is orange, second player's move is green.

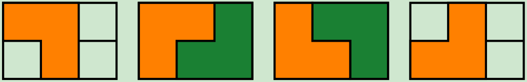


Note that for this board, the outcome (winning or losing) for the first player depends on that player's move. If s/he is smart, s/he makes the first or fourth move. This means that Player I has a winning strategy.

Working out small examples

Example (The 2×3 board)

First player's move is orange, second player's move is green.



Note that for this board, the outcome (winning or losing) for the first player depends on that player's move. If s/he is smart, s/he makes the first or fourth move. This means that Player I has a winning strategy.

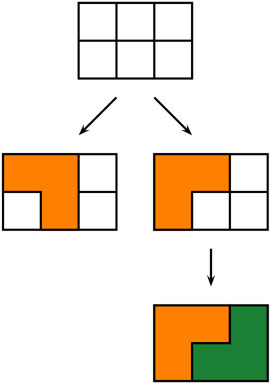
Game trees

Definition

A *position* (or game) in Ellie refers to any of the possible boards that arise in the course of playing the game. A position that arises from a move in the current position or game is called an *option* of the game. The directed graph which has the positions as the nodes and an arrow between a game and its options is called the *game tree*.

Options that are symmetric are usually not listed in the game tree.

Game tree for 2×3 board



Impartial Games

Definition

A position is a *$\mathcal{P}$ position* for the player about to make a move if the $\mathcal{P}$revious player can force a win (that is, the player about to make a move is in a losing position). The position is a *$\mathcal{N}$ position* if the $\mathcal{N}$ext player (the player about to make a move) can force a win.

For impartial games, there are only two outcome classes for any position, namely **winning position** ($\mathcal{N}$ position) or **losing position** ($\mathcal{P}$ position). There are no ties.

Recursive labeling

To find out whether Player I has a winning strategy, we label the nodes of the game tree recursively as follows:

- Leafs of the game tree are always losing ($\mathcal{P}$) positions.

Next we select any position (node) whose options (offsprings) are all labeled. There are two cases:

- The position has at least one option that is a losing ($\mathcal{P}$) position
  $\Rightarrow$ winning position and should be labeled $\mathcal{N}$
- All options of the position are winning ($\mathcal{N}$) positions
  $\Rightarrow$ losing position and should be labeled $\mathcal{P}$

The label of the empty board then tells whether Player I ($\mathcal{N}$) or Player II ($\mathcal{P}$) has a winning strategy.

Recursive labeling

To find out whether Player I has a winning strategy, we label the nodes of the game tree recursively as follows:

- Leafs of the game tree are always losing ($\mathcal{P}$) positions.

Next we select any position (node) whose options (offsprings) are all labeled. There are two cases:

- The position has at least one option that is a losing ($\mathcal{P}$) position $\Rightarrow$ winning position and should be labeled $\mathcal{N}$
- All options of the position are winning ($\mathcal{N}$) positions $\Rightarrow$ losing position and should be labeled $\mathcal{P}$

The label of the empty board then tells whether Player I ($\mathcal{N}$) or Player II ($\mathcal{P}$) has a winning strategy.

Recursive labeling

To find out whether Player I has a winning strategy, we label the nodes of the game tree recursively as follows:

- Leafs of the game tree are always losing ($\mathcal{P}$) positions.

Next we select any position (node) whose options (offsprings) are all labeled. There are two cases:

- The position has at least one option that is a losing ($\mathcal{P}$) position
  $\Rightarrow$ winning position and should be labeled $\mathcal{N}$
- All options of the position are winning ($\mathcal{N}$) positions
  $\Rightarrow$ losing position and should be labeled $\mathcal{P}$

The label of the empty board then tells whether Player I ($\mathcal{N}$) or Player II ($\mathcal{P}$) has a winning strategy.

Recursive labeling

To find out whether Player I has a winning strategy, we label the nodes of the game tree recursively as follows:

- Leafs of the game tree are always losing ($\mathcal{P}$) positions.

Next we select any position (node) whose options (offsprings) are all labeled. There are two cases:

- The position has at least one option that is a losing ($\mathcal{P}$) position
  $\Rightarrow$ winning position and should be labeled $\mathcal{N}$
- All options of the position are winning ($\mathcal{N}$) positions
  $\Rightarrow$ losing position and should be labeled $\mathcal{P}$

The label of the empty board then tells whether Player I ($\mathcal{N}$) or Player II ($\mathcal{P}$) has a winning strategy.

Recursive labeling

To find out whether Player I has a winning strategy, we label the nodes of the game tree recursively as follows:

- Leafs of the game tree are always losing ($\mathcal{P}$) positions.

Next we select any position (node) whose options (offsprings) are all labeled. There are two cases:

- The position has at least one option that is a losing ($\mathcal{P}$) position $\Rightarrow$ winning position and should be labeled $\mathcal{N}$
- All options of the position are winning ($\mathcal{N}$) positions $\Rightarrow$ losing position and should be labeled $\mathcal{P}$

The label of the empty board then tells whether Player I ($\mathcal{N}$) or Player II ($\mathcal{P}$) has a winning strategy.

Recursive labeling

To find out whether Player I has a winning strategy, we label the nodes of the game tree recursively as follows:

- Leafs of the game tree are always losing ($\mathcal{P}$) positions.

Next we select any position (node) whose options (offsprings) are all labeled. There are two cases:

- The position has at least one option that is a losing ($\mathcal{P}$) position $\Rightarrow$ winning position and should be labeled $\mathcal{N}$
- All options of the position are winning ($\mathcal{N}$) positions $\Rightarrow$ losing position and should be labeled $\mathcal{P}$

The label of the empty board then tells whether Player I ($\mathcal{N}$) or Player II ($\mathcal{P}$) has a winning strategy.

Recursive labeling

To find out whether Player I has a winning strategy, we label the nodes of the game tree recursively as follows:

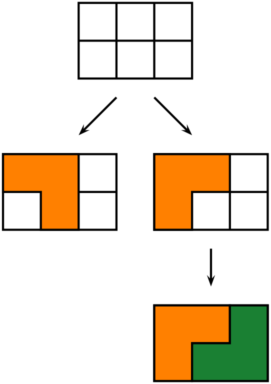
- Leafs of the game tree are always losing ($\mathcal{P}$) positions.

Next we select any position (node) whose options (offsprings) are all labeled. There are two cases:

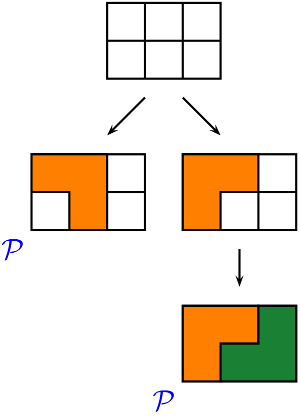
- The position has at least one option that is a losing ($\mathcal{P}$) position $\Rightarrow$ winning position and should be labeled $\mathcal{N}$
- All options of the position are winning ($\mathcal{N}$) positions $\Rightarrow$ losing position and should be labeled $\mathcal{P}$

The label of the empty board then tells whether Player I ($\mathcal{N}$) or Player II ($\mathcal{P}$) has a winning strategy.

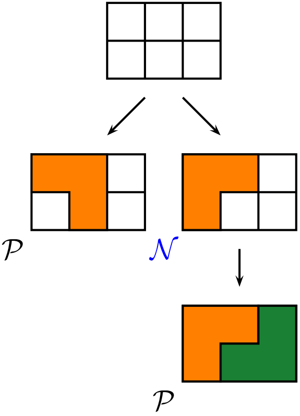
Labeling the game tree for 2×3 board



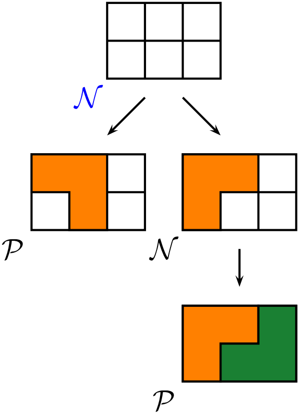
Labeling the game tree for 2 × 3 board



Labeling the game tree for 2 × 3 board



Labeling the game tree for 2 × 3 board

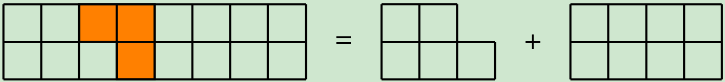


Sums of Games

Definition

If a move splits a game (board) into two smaller sub-boards such that the next player can play in only one of the two sub-boards, then the original game is called the *sum* of the two smaller games.

Example



The Grundy Function

Theorem

The Grundy-value $\mathcal{G}(G)$ of a game G is a measure of the distance to a losing position. If $\mathcal{G}(G) = n$, then for $k \leq n$ there is a sequence of moves that will lead to a losing position in k steps. In particular, G **is in the class** $\mathcal{P}$ **if and only if** $\mathcal{G}(G) = 0$.

So how do we compute the Grundy function???

The Grundy Function

Theorem

The Grundy-value $\mathcal{G}(G)$ of a game G is a measure of the distance to a losing position. If $\mathcal{G}(G) = n$, then for $k \leq n$ there is a sequence of moves that will lead to a losing position in k steps. In particular, G **is in the class $\mathcal{P}$ if and only if $\mathcal{G}(G) = 0$.**

So how do we compute the Grundy function???

Digital Sum and Mex

Definition

The *digital sum* $a \oplus b \oplus \cdots \oplus k$ of of integers $a, b, \ldots, k$ is obtained by translating the values into their binary representation and then adding them without carry-over.

Note that $a \oplus a = 0$.

Definition

The *minimum excluded value* or *mex* of a set of non-negative integers is the least non-negative integer which does not occur in the set. It is denoted by $\text{mex}\{a, b, c, \ldots, k\}$.

Digital Sum and Mex

Example

The digital sum $12 \oplus 13 \oplus 7$ equals 6:

$$
\begin{array}{r|cccc}
12 & 1 & 1 & 0 & 0 \\
13 & 1 & 1 & 0 & 1 \\
7 & & 1 & 1 & 1 \\
\hline
 & 0 & 1 & 1 & 0
\end{array}
$$

Example

$$\text{mex}\{1, 4, 5, 7\} = 0$$
$$\text{mex}\{0, 1, 2, 6\} = 3$$

Digital Sum and Mex

Example

The digital sum $12 \oplus 13 \oplus 7$ equals 6:

12	1	1	0	0
13	1	1	0	1
7		1	1	1
	0	1	1	0

Example

$$\text{mex}\{1,4,5,7\} = 0$$
$$\text{mex}\{0,1,2,6\} = 3$$

Digital Sum and Mex

Example

The digital sum $12 \oplus 13 \oplus 7$ equals 6:

12	1	1	0	0
13	1	1	0	1
7		1	1	1
	0	1	1	0

Example

$$\text{mex}\{1,4,5,7\} = 0$$
$$\text{mex}\{0,1,2,6\} = 3$$

Digital Sum and Mex

Example

The digital sum $12 \oplus 13 \oplus 7$ equals 6:

12	1	1	0	0
13	1	1	0	1
7		1	1	1
	0	1	1	0

Example

$$\text{mex}\{1,4,5,7\} = 0$$
$$\text{mex}\{0,1,2,6\} = 3$$

Computation of the Grundy Function

Theorem

For any impartial games G, H, and J,

- $\mathcal{G}(G) = \text{mex}\{\mathcal{G}(\text{H}) | \text{H } \textit{is an option of } G\}$.
- $G = H + J$ *if and only if* $\mathcal{G}(G) = \mathcal{G}(H) \oplus \mathcal{G}(J)$.

What does this all mean?

- For any given game tree we can recursively label the positions with their Grundy value, then read off the value for the starting board.
- This procedure is scalable if we can find a general rule explaining how a game breaks into smaller games so we can have a computer compute the Grundy function.
- We do not get the winning strategy (unless we look at the trace of the Grundy values), but we can answer the question about existence of a winning strategy.

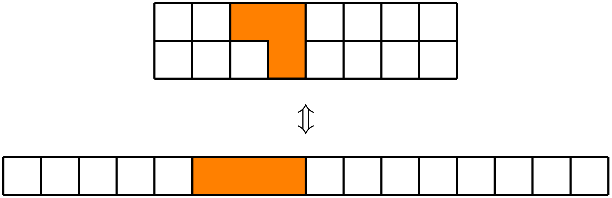
What does this all mean?

- For any given game tree we can recursively label the positions with their Grundy value, then read off the value for the starting board.
- This procedure is scalable if we can find a general rule explaining how a game breaks into smaller games so we can have a computer compute the Grundy function.
- We do not get the winning strategy (unless we look at the trace of the Grundy values), but we can answer the question about existence of a winning strategy.

What does this all mean?

- For any given game tree we can recursively label the positions with their Grundy value, then read off the value for the starting board.
- This procedure is scalable if we can find a general rule explaining how a game breaks into smaller games so we can have a computer compute the Grundy function.
- We do not get the winning strategy (unless we look at the trace of the Grundy values), but we can answer the question about existence of a winning strategy.

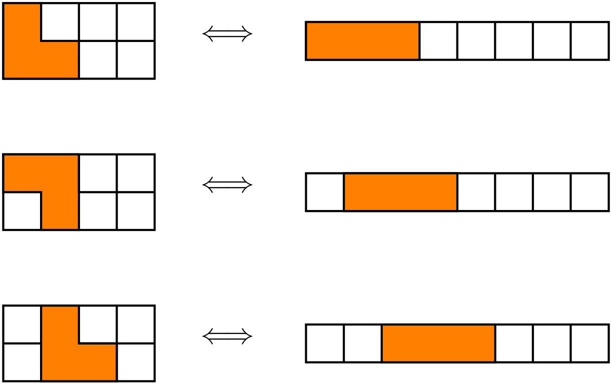
Ellie equivalent



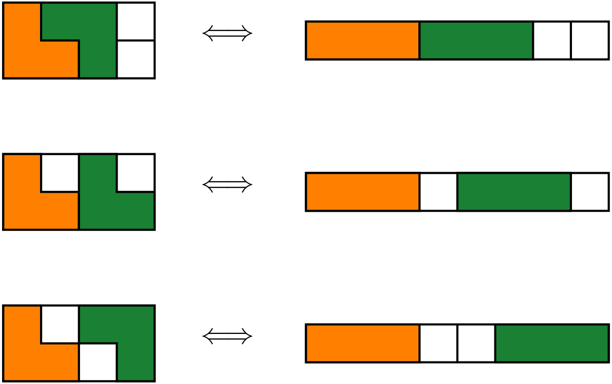
$2 \times n$ board for Ellie $\iff$ $1 \times (2n)$ board with 1×3 tile

Only the number of squares matters, not the geometry!

Ellie equivalent

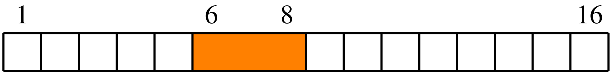


Ellie equivalent



Recursion for Grundy function

- Play at square i splits $1 \times n$ board into two boards of lengths $i - 1$ and $n - i - 2$



- G_n denotes play on a $1 \times n$ board; $G(n, i)$ denotes the game that results from placing 1×3 tile at square i
- $\mathcal{G}(G_0) = \mathcal{G}(G_1) = \mathcal{G}(G_2) = 0$
- $\mathcal{G}(G_n) = \text{mex}\{\mathcal{G}(G(n,i)) | 1 \leq i \leq \lfloor \frac{n}{2} \rfloor\}$
  $= \text{mex}\{\mathcal{G}(G_{i-1}) \oplus \mathcal{G}(G_{n-i-2}) | 1 \leq i \leq \lfloor \frac{n}{2} \rfloor\}$

Values for Grundy function

Let's compute the first 10 or so values of the Grundy function

- $\mathcal{G}(G_0) = \mathcal{G}(G_1) = \mathcal{G}(G_2) = 0$
- $\mathcal{G}(G_n) = \text{mex}\{\mathcal{G}(G_{i-1}) \oplus \mathcal{G}(G_{n-i-2}) | 1 \leq i \leq \lfloor \frac{n}{2} \rfloor\}$
- $\mathcal{G}(G_3) = \text{mex}\{\mathcal{G}(G_0) \oplus \mathcal{G}(G_0)\} = \text{mex}\{0\} = 1$
- $\mathcal{G}(G_4) = \text{mex}\{\mathcal{G}(G_0) \oplus \mathcal{G}(G_1), \mathcal{G}(G_1) \oplus \mathcal{G}(G_0)\} = \text{mex}\{0\} = 1$

n	0	1	2	3	4	5	6	7	8	9	10
$\mathcal{G}(G_n)$	0	0	0	1	1	1	2	2	0	3	3

Values for Grundy function

Let's compute the first 10 or so values of the Grundy function

- $\mathcal{G}(G_0) = \mathcal{G}(G_1) = \mathcal{G}(G_2) = 0$
- $\mathcal{G}(G_n) = \text{mex}\{\mathcal{G}(G_{i-1}) \oplus \mathcal{G}(G_{n-i-2}) | 1 \leq i \leq \lfloor \frac{n}{2} \rfloor\}$

- $\mathcal{G}(G_3) = \text{mex}\{\mathcal{G}(G_0) \oplus \mathcal{G}(G_0)\} = \text{mex}\{0\} = 1$
- $\mathcal{G}(G_4) = \text{mex}\{\mathcal{G}(G_0) \oplus \mathcal{G}(G_1), \mathcal{G}(G_1) \oplus \mathcal{G}(G_0)\} = \text{mex}\{0\} = 1$

n	0	1	2	3	4	5	6	7	8	9	10
$\mathcal{G}(G_n)$	0	0	0	1	1	1	2	2	0	3	3

Values for Grundy function

Let's compute the first 10 or so values of the Grundy function

- $\mathcal{G}(G_0) = \mathcal{G}(G_1) = \mathcal{G}(G_2) = 0$
- $\mathcal{G}(G_n) = \text{mex}\{\mathcal{G}(G_{i-1}) \oplus \mathcal{G}(G_{n-i-2}) | 1 \leq i \leq \lfloor \frac{n}{2} \rfloor\}$

- $\mathcal{G}(G_3) = \text{mex}\{\mathcal{G}(G_0) \oplus \mathcal{G}(G_0)\} = \text{mex}\{0\} = 1$
- $\mathcal{G}(G_4) = \text{mex}\{\mathcal{G}(G_0) \oplus \mathcal{G}(G_1), \mathcal{G}(G_1) \oplus \mathcal{G}(G_0)\} = \text{mex}\{0\} = 1$

n	0	1	2	3	4	5	6	7	8	9	10
$\mathcal{G}(G_n)$	0	0	0	1	1	1	2	2	0	3	3

Values for Grundy function

Let's compute the first 10 or so values of the Grundy function

- $\mathcal{G}(G_0) = \mathcal{G}(G_1) = \mathcal{G}(G_2) = 0$
- $\mathcal{G}(G_n) = \text{mex}\{\mathcal{G}(G_{i-1}) \oplus \mathcal{G}(G_{n-i-2}) | 1 \leq i \leq \lfloor \frac{n}{2} \rfloor\}$

- $\mathcal{G}(G_3) = \text{mex}\{\mathcal{G}(G_0) \oplus \mathcal{G}(G_0)\} = \text{mex}\{0\} = 1$
- $\mathcal{G}(G_4) = \text{mex}\{\mathcal{G}(G_0) \oplus \mathcal{G}(G_1), \mathcal{G}(G_1) \oplus \mathcal{G}(G_0)\} = \text{mex}\{0\} = 1$

n	0	1	2	3	4	5	6	7	8	9	10
$\mathcal{G}(G_n)$	0	0	0	1	1	1	2	2	0	3	3

Values for Grundy function

Let's compute the first 10 or so values of the Grundy function

- $\mathcal{G}(G_0) = \mathcal{G}(G_1) = \mathcal{G}(G_2) = 0$
- $\mathcal{G}(G_n) = \text{mex}\{\mathcal{G}(G_{i-1}) \oplus \mathcal{G}(G_{n-i-2}) | 1 \leq i \leq \lfloor \frac{n}{2} \rfloor\}$

- $\mathcal{G}(G_3) = \text{mex}\{\mathcal{G}(G_0) \oplus \mathcal{G}(G_0)\} = \text{mex}\{0\} = 1$
- $\mathcal{G}(G_4) = \text{mex}\{\mathcal{G}(G_0) \oplus \mathcal{G}(G_1), \mathcal{G}(G_1) \oplus \mathcal{G}(G_0)\} = \text{mex}\{0\} = 1$

n	0	1	2	3	4	5	6	7	8	9	10
$\mathcal{G}(G_n)$	0	0	0	1	1	1	2	2	0	3	3

Values for Grundy function

Let's compute the first 10 or so values of the Grundy function

- $\mathcal{G}(G_0) = \mathcal{G}(G_1) = \mathcal{G}(G_2) = 0$
- $\mathcal{G}(G_n) = \text{mex}\{\mathcal{G}(G_{i-1}) \oplus \mathcal{G}(G_{n-i-2}) | 1 \leq i \leq \lfloor \frac{n}{2} \rfloor\}$

- $\mathcal{G}(G_3) = \text{mex}\{\mathcal{G}(G_0) \oplus \mathcal{G}(G_0)\} = \text{mex}\{0\} = 1$

- $\mathcal{G}(G_4) = \text{mex}\{\mathcal{G}(G_0) \oplus \mathcal{G}(G_1), \mathcal{G}(G_1) \oplus \mathcal{G}(G_0)\} = \text{mex}\{0\} = 1$

n	0	1	2	3	4	5	6	7	8	9	10
$\mathcal{G}(G_n)$	0	0	0	1	1	1	2	2	0	3	3

Values for Grundy function

Let's compute the first 10 or so values of the Grundy function

- $\mathcal{G}(G_0) = \mathcal{G}(G_1) = \mathcal{G}(G_2) = 0$
- $\mathcal{G}(G_n) = \text{mex}\{\mathcal{G}(G_{i-1}) \oplus \mathcal{G}(G_{n-i-2}) | 1 \leq i \leq \lfloor \frac{n}{2} \rfloor\}$

- $\mathcal{G}(G_3) = \text{mex}\{\mathcal{G}(G_0) \oplus \mathcal{G}(G_0)\} = \text{mex}\{0\} = 1$

- $\mathcal{G}(G_4) = \text{mex}\{\mathcal{G}(G_0) \oplus \mathcal{G}(G_1), \mathcal{G}(G_1) \oplus \mathcal{G}(G_0)\} = \text{mex}\{0\} = 1$

n	0	1	2	3	4	5	6	7	8	9	10
$\mathcal{G}(G_n)$	0	0	0	1	1	1	2	2	0	3	3

Values for Grundy function

Let's compute the first 10 or so values of the Grundy function

- $\mathcal{G}(G_0) = \mathcal{G}(G_1) = \mathcal{G}(G_2) = 0$
- $\mathcal{G}(G_n) = \text{mex}\{\mathcal{G}(G_{i-1}) \oplus \mathcal{G}(G_{n-i-2}) | 1 \leq i \leq \lfloor \frac{n}{2} \rfloor\}$

- $\mathcal{G}(G_3) = \text{mex}\{\mathcal{G}(G_0) \oplus \mathcal{G}(G_0)\} = \text{mex}\{0\} = 1$
- $\mathcal{G}(G_4) = \text{mex}\{\mathcal{G}(G_0) \oplus \mathcal{G}(G_1), \mathcal{G}(G_1) \oplus \mathcal{G}(G_0)\} = \text{mex}\{0\} = 1$

n	0	1	2	3	4	5	6	7	8	9	10
$\mathcal{G}(G_n)$	0	0	0	1	1	1	2	2	0	3	3

Values for Grundy function

Let's compute the first 10 or so values of the Grundy function

- $\mathcal{G}(G_0) = \mathcal{G}(G_1) = \mathcal{G}(G_2) = 0$
- $\mathcal{G}(G_n) = \text{mex}\{\mathcal{G}(G_{i-1}) \oplus \mathcal{G}(G_{n-i-2}) | 1 \leq i \leq \lfloor \frac{n}{2} \rfloor\}$

- $\mathcal{G}(G_3) = \text{mex}\{\mathcal{G}(G_0) \oplus \mathcal{G}(G_0)\} = \text{mex}\{0\} = 1$
- $\mathcal{G}(G_4) = \text{mex}\{\mathcal{G}(G_0) \oplus \mathcal{G}(G_1), \mathcal{G}(G_1) \oplus \mathcal{G}(G_0)\} = \text{mex}\{0\} = 1$

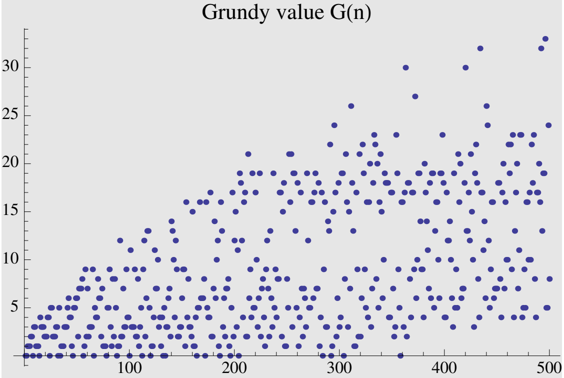
n	0	1	2	3	4	5	6	7	8	9	10
$\mathcal{G}(G_n)$	0	0	0	1	1	1	2	2	0	3	3

Structure of Values

Questions to be answered:

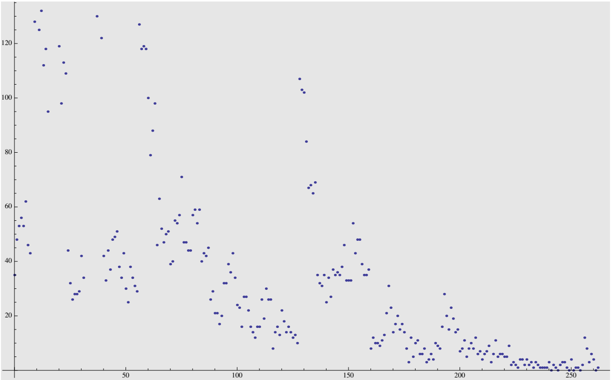
1. Is the sequence of Grundy values $\mathcal{G}(G_n)$ periodic?
2. Is the sequence of Grundy values $\mathcal{G}(G_n)$ ultimately periodic?

Values of $\mathcal{G}(n)$



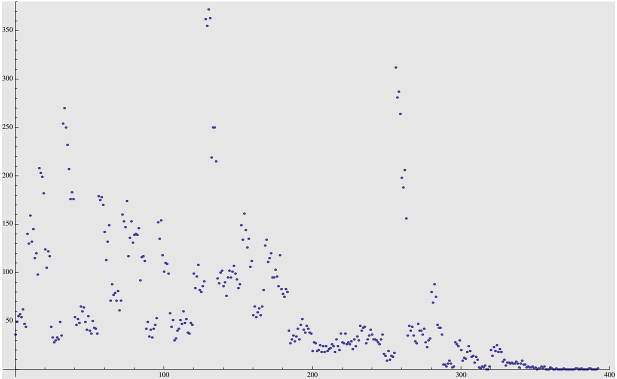
The first 500 values of $\mathcal{G}(n)$

Frequencies of $\mathcal{G}(n)$



10000 values of $\mathcal{G}(n)$; max val = 262; max freq = 202

Frequencies of $\mathcal{G}(n)$



25000 values of $\mathcal{G}(n)$; max val = 392; max freq = 372

Octal Games

Definition

An *octal game* is a 'take-and-break' game identified by a code of the form $.d_1 d_2 d_3 \ldots$ with $0 \leq d_i \leq 7$. A typical move consists of choosing one of the heaps and removing i tokens from the heap, then rearranging the remaining tokens into some allowed number of new heaps. The code describes the allowed moves in the game:

- If $d_i \neq 0$, then an allowed move is to take i tokens from a heap.
- Writing $d_i \neq 0$ in base 2 then shows how the i tokens may be taken: If $d_i = c_2 \cdot 2^2 + c_1 \cdot 2^1 + c_0 \cdot 2^0$, then removal of the i tokens may ($c_j = 1$) or may not ($c_j = 0$) leave j heaps.

Octal Games

Example

The octal game **.17** allows us to take either 1 or 2 tokens.

- $d_1 = 1 = 0 \cdot 2^2 + 0 \cdot 2^1 + \mathbf{1} \cdot 2^0$, therefore we are allowed to leave zero heaps when taking one token, that is, we can take away a heap that consists of a single token.
- $d_2 = 7 = \mathbf{1} \cdot 2^2 + \mathbf{1} \cdot 2^1 + \mathbf{1} \cdot 2^0$, therefore we are allowed to leave either two, one or no heaps when taking two tokens, that is, we can take away a heap that consists of two tokens, we can remove two tokens from the top of a heap (leaving one heap), or can take two tokens and split the remaining heap into two non-zero heaps.

Ellie = ?

Since we can only take three tokens at a time, $d_i = 0$ for $i \neq 3$. When we place a tile, it can be

- at the end (leaving one heap),
- in the middle of the board (leaving two heaps), or
- covering the last three squares, leaving zero heaps.

$\Rightarrow$ **Ellie =**.007

Ellie = ?

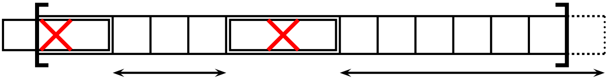
Since we can only take three tokens at a time, $d_i = 0$ for $i \neq 3$. When we place a tile, it can be

- at the end (leaving one heap),
- in the middle of the board (leaving two heaps), or
- covering the last three squares, leaving zero heaps.

$\Rightarrow$ **Ellie =.007**

Treblecross = .007

- Treblecross is Tic-Tac-Toe played on a $1 \times n$ board in which both players use the same symbol, X. The first one to get three X's in a row wins.
- Don't want to place an X next to or next but one to an existing X, otherwise opponent wins immediately
- If only considering sensible moves, one can think of each X as also occupying its two neighbors



What is known about .007

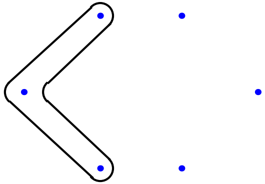
- No complete analysis
- $\mathcal{G}(G_n)$ computed up to $n = 2^{21} = 2,097,152$
- Maximum Grundy value in that range is $\mathcal{G}(1,683,655) = 1,314$
- Last new Grundy value to occur is $\mathcal{G}(1,686,918) = 1,237$
- Most frequent value is 1024, which occurs 63,506 times
- Second most frequent value is 1026, which occurs 62,178 times
- 37 $\mathcal{P}$ positions: 0, 1, 2, 8, 14, 24, 32, 34, 46, 56, 66, 78, 88, 100, 112, 120, 132, 134, 164, 172, 186, 196, 204, 284, 292, 304, 358, 1048, 2504, 2754, 2914, 3054, 3078, 7252, 7358, 7868, 16170

What now????

- Looked at Misère version of the game (last player to move loses), but that is hopeless....
- Tried to see what happens on $3 \times n$ Ellie board - very tough
- Decided to leave Ellie and move on to greener (?) pastures

Circular (n, k) Games

n heaps in a circular arrangement. Select k consecutive heaps and select at least one token from at least one of the heaps



Circular (6,3) game

Question: What is the structure of the set of losing positions?

Variations

- Select a fixed number a from each of the heaps
- Select at least one token from each of the k heaps
- Select at least a tokens from each of the k heaps
- .....

Thank You!

For Further Reading

- Elwyn R. Berlekamp, John H. Conway and Richard K. Guy.
  Winning Ways for Your Mathematical Plays, Vol 1 & 2.
  Academic Press, London, 1982.

- Michael H. Albert, Richard J. Nowakowski, and David Wolfe.
  Lessons in Play.
  AK Peters, 2007.

- I. Caines, C. Gates, R.K. Guy, and R. J. Nowakowski.
  Periods in Taking and Splitting Games.
  American Mathematical Monthly, April:359–361, 1999.

- A. Gangolli and T. Plambeck.
  A Note on periodicity in Some Octal Games.
  International Journal of Game Theory, 18:311–320, 1989.