On the Winning Strategies in Generalizations of Nim

Joshua Xiong

Mentor: Tanya Khovanova

Fourth Annual PRIMES Conference

May 18, 2014

An example: Nim

- Take at least one cookie from any one pile
- The player who takes the last cookie wins

Player 1

Player 2

(2,3,1)

An example: Nim

- Take at least one cookie from any one pile
- The player who takes the last cookie wins

Player 1

Player 2

(2,2,1)

An example: Nim

- Take at least one cookie from any one pile
- The player who takes the last cookie wins

Player 1

Player 2

(2,2,0)

An example: Nim

- Take at least one cookie from any one pile
- The player who takes the last cookie wins

An example: Nim

- Take at least one cookie from any one pile
- The player who takes the last cookie wins

Player 1 Player 2

(1,1,0)

An example: Nim

- Take at least one cookie from any one pile
- The player who takes the last cookie wins

Player 1

Player 2

(0,1,0)

An example: Nim

- Take at least one cookie from any one pile
- The player who takes the last cookie wins

Player 1

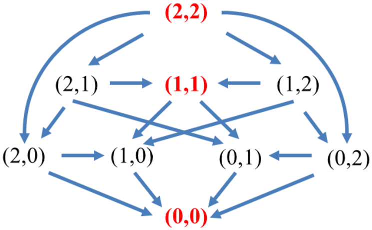
Player 2

(0,0,0)

- Player 2 wins

P-Positions

- The starting position $(2, 3, 1)$ is one where the person to play will always lose assuming optimal play
- We call such positions P-positions (losing positions)
- All other positions are called N-positions (winning positions)
  - Moves from P-positions can only go to N-positions
  - At least one move from every N-position goes to a P-position
  - The zero position $(0, \ldots, 0)$ is a P-position
- Winning strategy is to move to a P-position



Winning Strategy for Nim

Theorem (Bouton's Theorem)

In Nim, $P = (a_1, \ldots, a_n) \in \mathcal{P}$ if and only if $\bigoplus_{i=1}^{n} a_i = 0$.

- The operator $\oplus$ is the bitwise XOR operator, (nim-sum) – represent each of the numbers in binary and add them column-wise modulo 2.

Another example: Wythoff's Game

- Take same number of cookies from two piles or any number from one pile

Player 1

Player 2

(4,3)

Another example: Wythoff's Game

- Take same number of cookies from two piles or any number from one pile

Player 1

Player 2

(2,1)

Another example: Wythoff's Game

- Take same number of cookies from two piles or any number from one pile

Player 1

Player 2

(1,1)

Another example: Wythoff's Game

- Take same number of cookies from two piles or any number from one pile

Player 1

Player 2

(0,0)

Another example: Wythoff's Game

- Take same number of cookies from two piles or any number from one pile

Player 1

Player 2

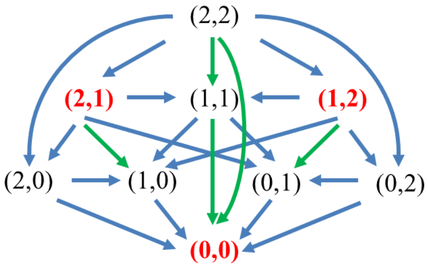
(0,0)

- Player 1 wins

Winning Strategy for Wythoff

Theorem (Wythoff's Theorem)

In Wythoff's game, $P = (a_1, a_2) \in \mathcal{P}$ if and only if $\{a_1, a_2\} = \{\lfloor n\phi \rfloor, \lfloor n\phi^2 \rfloor\}$ for some integer n, where $\phi = \frac{1+\sqrt{5}}{2}$.



Rectangular Games

- Move consists of taking same number of cookies from specified subsets of piles
  - Based on Cookie Monster game
- Adjoins rules onto the Nim rule (taking at least one cookie from exactly one of the piles)
- We are interested in the properties of the P-positions

Proposed Generalizations

- We specify which subsets of piles are legal to take from:
- Examples of games with three piles
  1. One or n game
  2. One or Two game
  3. Consecutive game
  4. Cookie Monster game

(1,2,3)

Proposed Generalizations

- We specify which subsets of piles are legal to take from:
- Examples of games with three piles
  1. One or *n* game
  2. One or Two game
  3. Consecutive game
  4. Cookie Monster game

(0,1,2)

Proposed Generalizations

- We specify which subsets of piles are legal to take from:
- Examples of games with three piles
  1. One or n game
  2. One or Two game
  3. Consecutive game
  4. Cookie Monster game

(1,2,3)

Proposed Generalizations

- We specify which subsets of piles are legal to take from:
- Examples of games with three piles
  1. One or n game
  2. One or Two game
  3. Consecutive game
  4. Cookie Monster game

(0,2,2)

Proposed Generalizations

- We specify which subsets of piles are legal to take from:
- Examples of games with three piles
    1. One or n game
    2. One or Two game
    3. Consecutive game
    4. Cookie Monster game

(1,2,3)

Proposed Generalizations

- We specify which subsets of piles are legal to take from:
- Examples of games with three piles
  1. One or n game
  2. One or Two game
  3. Consecutive game
  4. Cookie Monster game

(1,0,1)

Proposed Generalizations

- We specify which subsets of piles are legal to take from:
- Examples of games with three piles
    1. One or *n* game
    2. One or Two game
    3. Consecutive game
    4. Cookie Monster game

(1,2,3)

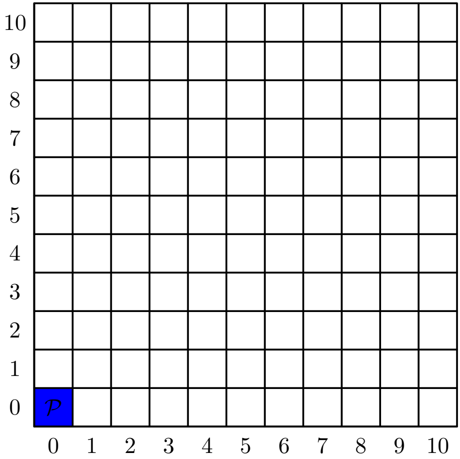
Proposed Generalizations

- We specify which subsets of piles are legal to take from:
- Examples of games with three piles
  1. One or n game
  2. One or Two game
  3. Consecutive game
  4. Cookie Monster game

(0,1,3)

Methods / Programs

- We implement an efficient recursive algorithm for computing P-positions in Java



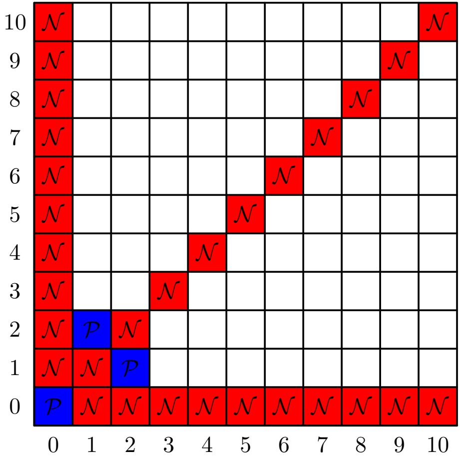
Methods / Programs

- We implement an efficient recursive algorithm for computing P-positions in Java

10	$\mathcal{N}$										$\mathcal{N}$
9	$\mathcal{N}$									$\mathcal{N}$	
8	$\mathcal{N}$								$\mathcal{N}$		
7	$\mathcal{N}$							$\mathcal{N}$			
6	$\mathcal{N}$						$\mathcal{N}$				
5	$\mathcal{N}$					$\mathcal{N}$					
4	$\mathcal{N}$				$\mathcal{N}$						
3	$\mathcal{N}$			$\mathcal{N}$							
2	$\mathcal{N}$		$\mathcal{N}$								
1	$\mathcal{N}$	$\mathcal{N}$									
0	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
	0	1	2	3	4	5	6	7	8	9	10

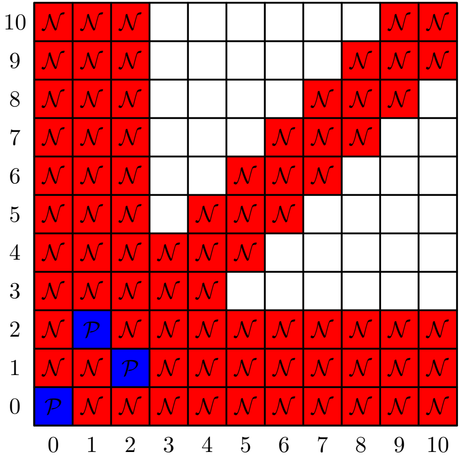
Methods / Programs

- We implement an efficient recursive algorithm for computing P-positions in Java



Methods / Programs

- We implement an efficient recursive algorithm for computing P-positions in Java



Methods / Programs

- We implement an efficient recursive algorithm for computing P-positions in Java

10	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$							$\mathcal{N}$	$\mathcal{N}$
9	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$						$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
8	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$					$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	
7	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$				$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$		
6	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$			$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$			
5	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$				
4	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$					
3	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$					
2	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
1	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
0	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
	0	1	2	3	4	5	6	7	8	9	10

Methods / Programs

- We implement an efficient recursive algorithm for computing P-positions in Java

10	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$		$\mathcal{N}$			$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
9	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$		$\mathcal{N}$		$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
8	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$		$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
7	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$		$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	
6	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$		
5	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
4	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$				
3	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
2	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
1	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
0	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
	0	1	2	3	4	5	6	7	8	9	10

Methods / Programs

- We implement an efficient recursive algorithm for computing P-positions in Java

	0	1	2	3	4	5	6	7	8	9	10
10	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$		$\mathcal{N}$			$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
9	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$		$\mathcal{N}$		$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
8	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$		$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
7	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	
6	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$		
5	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
4	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$			
3	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
2	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
1	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
0	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$

Methods / Programs

- We implement an efficient recursive algorithm for computing P-positions in Java

10	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$		$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
9	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
8	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
7	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
6	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	
5	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
4	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
3	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
2	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
1	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
0	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
	0	1	2	3	4	5	6	7	8	9	10

Methods / Programs

- We implement an efficient recursive algorithm for computing P-positions in Java

10	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
9	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
8	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
7	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
6	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$
5	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
4	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
3	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
2	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
1	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
0	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
	0	1	2	3	4	5	6	7	8	9	10

Odd Game P-positions

Theorem

If we are only allowed to take from an odd number of piles, the P-positions are the same as the ones in Nim.

Main idea of proof:

- Show that the nim-sum of the position has to change when we use the new moves
- Use the strategy in Nim to get back to a position with zero nim-sum
- Since $(0, \ldots, 0) \in \mathcal{P}$, P-positions will be the same

Degrees of Freedom of P-positions

Theorem

For a position with $n-1$ numbers known, and one number unknown: $P = (a_1, \ldots, a_{n-1}, x)$, there is a unique value of x such that $P \in \mathcal{P}$.

- In general, does there exist a function $f(a_1, \ldots a_{n-1}) = x$?
- For Nim, this function is $f_{\mathrm{NIM}}(a_1, \ldots a_{n-1}) = \bigoplus_{i=1}^{n-1} a_i$.

Bounds on P-Positions

- General bound that holds for all rectangular games

Theorem

If $P = (a_1, \ldots, a_n) \in \mathcal{P}$ *then* $2(\sum_{i=1}^{n} a_i - a_j) \geq a_j$.

- If no move allows us to take from exactly two of the piles

Theorem

If $P = (a_1, \ldots, a_n) \in \mathcal{P}$ *then* $\sum_{i=1}^{n} a_i - a_j \geq a_j$.

  - Both proved by strong induction on $\sum_{i=1}^{n} a_i - a_j$

Enumeration of P-positions

- This helps to get a sense of the structure and distribution of the P-positions
- We can enumerate based on total sum
- We calculate this sequence for Nim:
  - Three piles, total sum n: $3^{\mathrm{wt}(n)}$ if n is even, 0 otherwise, where $\mathrm{wt}(n)$ is the number of ones in the binary representation of n.
  - 1, 0, 1, 0, 3, 0, 3, 0, 9, 0, 3, 0, 9, 0, 9, 0, 27, 0, ...
- We also calculate for all other games (we show Cookie Monster game with three piles)
  - Three piles, total sum n: 1, 0, 0, 6, 0, 0, 3, 3, 6, 0, 3, 9, 6, 6, 0, 0, 15, 3, ...

Future Research

- Find an explicit formula for P-positions in our games
- Is there a metric that describes how closely the game behaves to Nim and Wythoff?
- Connection between consecutive game and Lie Algebras
- What happens in the misère version?

Acknowledgments

I would like to thank...

- My mentor, Dr. Tanya Khovanova, for suggesting this project and for her invaluable words of wisdom
- Mr. Wuttisak Trongsiriwat and Mr. Rik Sengupta, for helpful tips and advice on the direction of this project
- The MIT PRIMES program, for providing me with the opportunity to conduct this research
- My parents, for providing transportation as well as continuous support

References

1. M. H. Albert, R. J. Nowakowski, D. Wolfe, *Lessons in Play*, A K Peters, Wellesley, MA, 2007
2. E. R. Berlekamp, J. H. Conway and R. K. Guy, *Winning Ways for your Mathematical Plays*, vol. 1, second edition, A. K. Peters, Natick, MA, 2001.
3. C. L. Bouton. *The Annals of Mathematics*, 2nd Ser., Vol. 3, No. 1/4. (1901 - 1902), pp. 35–39.