E.T.G. Schlebusch

Grundy's Game

Master's thesis

March 31st, 2015

Supervisor: Dr. F.M. Spieksma

Mathematical Institute, Leiden University

Contents

1. Introduction

In 1939, P. Grundy defined a new game [6], which is now named after him: Grundy's Game. He describes it as a two-person game with a pile of matches, where a move consists of taking any pile and dividing it into two unequal parts. The last player to make a move wins. This game is an example of a so called impartial combinatorial game, which we will define in Section 2. In that section we will also provide examples of other impartial combinatorial games, like Kayles.

In mathematics, and in particular in game theory, we are of course interested in finding an optimal strategy for a game. For impartial combinatorial games, we can determine whether a state of a game is a winning position for the **p**revious player or for the **n**ext player, or, in other words, whether the state is a P- or an N-position. We describe these P- and N-positions, and how to determine them in Section 3.

However, it is usually easier to work with the so called Sprague-Grundy values of the different states of a game, instead of P- and N-positions. In the literature that we found, this Sprague-Grundy function, and the corresponding Nim-addition, is consistently posed without any reasoning why it is used. In this thesis, we try to give a more intuitive approach to the Sprague-Grundy theorem, which will hopefully give the reader a better understanding of the theorem and why it is used.

We start by explaining the need for the Sprague-Grundy function in Section 4. In Section 5 we will show how Nim-addition leads to the Spargue-Grundy theorem, which allows us to work with combinations, or sums, of more than one game. In Section 6 we explain how one can use the Sprague-Grundy function and theorem to play optimally in an impartial combinatorial game.

The Sprague-Grundy function uses recursion to give a value to each state of a game. This means that it takes a long time to calculate the Sprague-Grundy value for a 'large' state (for example, Grundy's Game with a lot of matches). To avoid this, we try to find some kind of periodicity in the Sprague-Grundy values of games. The Sprague-Grundy values of Kayles, which we give in Section 7, become periodic relatively fast. However, for Grundy's Game no periodicity has been found yet, even though values have been calculated up to piles of 2^{35} matches [5]. We give the first 30 values in Section 8.

To gain more insight into the Sprague-Grundy values of Grundy's Game, we will study periodicity for several adaptations of Grundy's Game in Section 9. In Section 10 we deduce some periodicity properties. For instance, we prove that the Sprague-Grundy values cannot have periodicity 3 (Lemma 10.2), even though Berlekamp, Conway and Guy write that 'the strong tendency to period 3 continues as far as values have been calculated' [1].

In Section 11 we divide the Sprague-Grundy values of Grundy's Game into rare and common values. This division was first made by Berlekamp, Conway and Guy, and makes it more intuitive that eventually the Sprague-Grundy values of Grundy's Game will become periodic [1]. In Sections 10 and 11 we also give two upper limits (Proposition 10.1 and Remark 11.8), such that, if values are calculated up till there and satisfy the given conditions, we can prove that the Sprague-Grundy values do eventually become periodic.

2. Impartial combinatorial games

Definition 2.1 (Combinatorial games)**.** For combinatorial games, we use the definition given by Ferguson [4]. Combinatorial games are games that satisfy the following conditions.

- There are two players that alternate moving.
- There is a finite set of possible states of the game.
- The rules of the game specify, for both players and each state, the set of feasible moves.
- The game ends when a position is reached from which no moves are possible for the player whose turn it is to move.
- The game always ends in a finite number of moves.

Definition 2.2 (Normal and misère play)**.** Combinatorial games can be played with the normal and the misère rule [1] [4]. With the normal rule, the last player to make a move wins, and with the misère rule, the last player to make a move loses. From now on, we will only consider games played with the normal rule.

Definition 2.3 (Impartial game)**.** We call a combinatorial game impartial, if both players have the same set of feasible moves for any given state S [1] [4]. We say $H(S)$ is the set of states that can be reached by making a feasible move from state S.

2.1. **Graph representation.**

Every impartial combinatorial game can be represented by a finite directed graph $G(V, E)$ without directed cycles. Each state of the game is represented by a node $v \in V$ and legal moves from one state to another are represented by directed edges $(v, w) \in E$. This means that $H(v)$ for $v \in V$ is defined as follows:

$$H(v) = \{w \in V : (v, w) \in E\}.$$

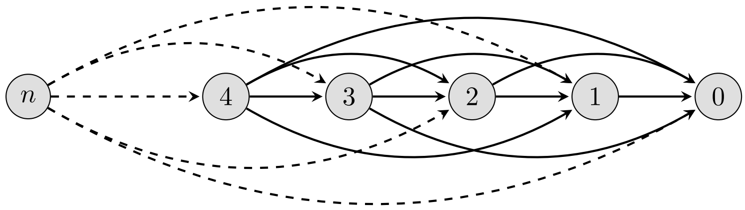
Since both players have the same available moves, the graph is the same for both players and the possible moves (edges) only depend on what state (node) the player is in. Because the game always ends in a finite number of moves, there cannot be any cycles in the graph. The graph does not uniquely determine the game, since there might be more games with the same graph, but where the nodes represent different states. It is easy to see that every finite directed graph without cycles represents at least one impartial combinatorial game.

2.2. **Game of Nim.**

The most widely known impartial combinatorial game is the game of Nim.

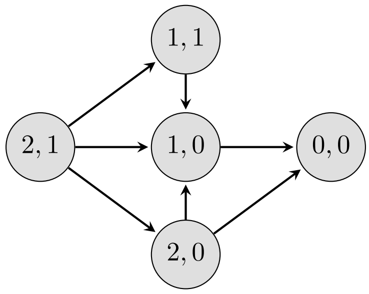
In the game of Nim there are two players and a number of stones distributed over several piles. A turn for one of the players consists of removing a number of stones (at least 1 and at most all the stones in the pile) from one pile. A player wins, when he or she removes the last stone in the game.

Example 2.4. The graph-representation of the game of Nim with one pile of n stones is given in the following picture:



Note that this is a very easy game to win for the first player: just take all the stones in the pile.

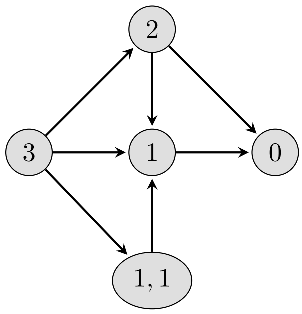
Example 2.5. The graph-representation of the game of Nim with two piles of respectively 2 and 1 stones is given in the following picture:



Here, a winning strategy for the first player would be to move to state $(1, 1)$. Then the second player can only move to state $(1, 0)$, after which the first player can make the last move.

2.3. **Kayles.**
Another example of an impartial combinatorial game is Kayles, first described by Henry Ernest Dudeney [3]. This game starts with n bowling pins, all standing next to each other. A move consists of knocking down one pin or two adjacent pins.

Example 2.6. The graph-representation of Kayles starting with 3 pins is given in the following picture:

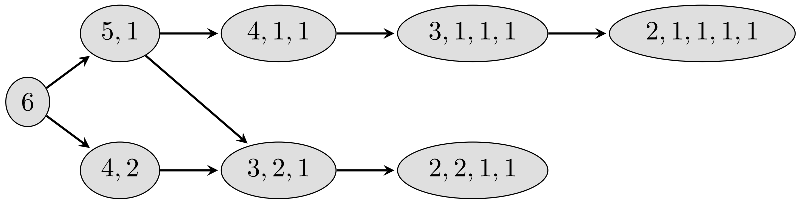


Here $(1, 1)$ represents the state where the middle bowling pin has been knocked down. From this state, it is not possible to knock down both pins and move to state (0), since the pins are not adjacent.

A winning strategy for the first player, would be to move to the state $(1, 1)$. Then the second player can only move to state (1), after which the first player can make the last move.

2.4. **Grundy's Game.**
In 1939, Patrick Michael Grundy wrote a paper [6] in which he defined the following game. Like in the game of Nim, the game consists of stones (or matches, marbles, etcetera) distributed over several piles. A turn for one of the players consists of dividing one of these piles into two different-sized piles. For example, a pile consisting of 6 stones can be divided into a pile of 5 and a pile of 1, or a pile of 4 and a pile of 2. However, it cannot be divided into two piles of 3. Whoever cannot divide any more piles (because all piles contain 1 or 2 stones) loses the game.

Example 2.7. The graph-representation of Grundy's Game starting with a pile of 6 stones is as follows.



A winning strategy for the first player, would be to move to state $(4, 2)$. Then the second player can only move to state $(3, 2, 1)$, after which the first player can make the last move.

3. P- and N-positions

For impartial, combinatorial games, we can calculate the optimal strategy for both players, and we can thus always calculate which one of the two players will win, provided that both players play optimally.

Definition 3.1 (P- and N-positions)**.** We call a state a P-position if it is a winning position for the **P**revious player and an N-position if it is a winning position for the **N**ext player [1] [4].

So how do we know what P- an N-positions are? Note that the terminal positions, which are states in which there are no more possible moves, are always winning positions for the previous player, and thus P-positions. Now let us assume that we are in a state S (that is not a terminal position), and player 1 is the next player to make a move. After making a move, player 1 will be the previous player. So, if player 1 can make a move to a state that is a P-position, player 1 will be the winner of the game. Since in our current state S, player 1 is the next player, S is then an N-position. If, however, it is not possible to move to a state that is a P-position, it means that the next state is an N-position and thus a winning position for player 2. In our current state S, player 2 is the previous player, and it is thus a P-position.

We can describe whether a state x is a P or an N-position with the following labelling function:

Definition 3.2 (Labelling function)**.** The labelling function $h : V \to \{P, N\}$ is defined as follows.

$$h(x) = \begin{cases} P & \text{if } \forall y \in H(x) : h(y) = N \\ N & \text{else.} \end{cases}$$

Lemma 3.3. *The function h is uniquely defined for all nodes on any finite, directed graph without cycles.*

We will prove this lemma by just giving an algorithm to label nodes in a finite, directed graph without cycles by P for a P-position and by N for an N-position. Then we will show the correctness of the algorithm.

Algorithm 3.4 (Labelling algorithm)**.**

(1) Let $L = \{\emptyset\}$.
(2) Choose $v \in V \setminus L$ with $H(v) \subseteq L$. $L := L \cup \{v\}$.
If $\exists w \in H(v)$ with w labeled P: give v label N.
Else: give v label P.
(3) If $L = V$: stop.
Else: Go to step 2.

Proof. We will now prove that the algorithm above is correct, by showing that it gives the correct labelling for every finite, directed graph without cycles. First note that, if all nodes are labelled, they are all labelled correctly, since a node v is labelled P, if and only if $h(v) = P$.

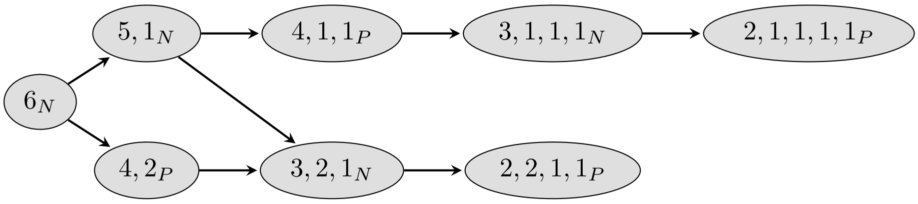
Now we will show that the algorithm labels all nodes in every finite, directed graph without cycles. Note that only labelled nodes are added to L, so if $L = V$ we have labelled all nodes. We thus need to show that all nodes are added to L, and thus that, for $L \neq V$, there is always a node $v \notin L$ that satisfies the condition for step 2: $H(v) \subseteq L$.

Let $G' = (V', E') \subseteq G$, where $V' = V \setminus L \neq \emptyset$ and $E' = \{(x, y) : (x, y) \in E : x, y \in V'\}$. We need a node $v \in V'$, such that, for each $w \in V'$, $(v, w) \notin E'$. Now assume that there is no such node v. This means, that for each node $v \in V'$, there is a $w \in V'$, such that $(v, w) \in E'$. This means that we can move through the graph G' for an infinite amount of steps. G' is a subgraph of G, and thus also finite. This means that we must reach one node at least twice during our tour. This means that G' has to contain a cycle, but it does not, since it is a subgraph of G. We thus have a contradiction and, as long as $L \neq V$, there is always a node $v \notin L$ that satisfies the condition for step 2: $H(v) \subseteq L$. □

Corollary 3.5. *Lemma 3.3 and the fact that each impartial combinatorial game can be represented by a finite directed graph without cycles, imply that for each impartial combinatorial game, we can determine whether each state is a P- or an N-position.*

Remark 3.6 (Optimal move)**.** This labelling of the states of a game also provides optimal moves for each state. If the state is a P-position, there is no optimal move for the next player, and if the state is an N-position, an optimal move is to move the game to a P-position (note that there might be more then one optimal move).

Example 3.7. We will show the P- and N-positions for Grundy's Game, starting with a pile of six stones. As described above, we start with the terminal positions, which we will label P. Then we work backwards, where states that can reach at least one P-position will be labelled N and states that can only reach N-positions will be labelled P.



So when playing Grundy's Game with a pile of six stones, the player that makes the first move can always win, since (6) is an N-position. This player will win by first moving to a P-position, thus splitting the pile of 6 stones into a pile of 4 and a pile of 2. The other player then only has one possible move, namely to split the pile of 4 stones into a pile of 3 and a pile of 1. The first player will then split the pile of 3 stones into a pile of 2 and a pile of 1, and the other player will have no move left, since all remaining piles will consist of either 1 or 2 stones.

4. Sprague-Grundy function

In the game of Nim with one pile, we have seen already that the only P-position is the empty pile, and all other states are N-positions. This leads to the following labelling function $h_{Nim} : \mathbb{N} \to \{P, N\}$.

$$h_{Nim}(x) = \begin{cases} P & \text{if } x = 0 \\ N & \text{else.} \end{cases}$$

This labelling function is very easy, and the game of Nim with one pile is uncomplicated as well. An optimal move is always to take all stones that are left. Because this one-pile Nim-game is so easy, it would make sense to associate states of other impartial combinatorial games to states of a one-pile Nim-game.

So how is this done? We want to associate a state S of a game G in such a way, that all moves that are available in one-pile Nim, are also available in G. If that is the case, we can mimic the moves in one-pile Nim in the game G. In other words, the state S of G is associated with the Nim-game of n stones, if and only if it can reach a state that is associated with the Nim-game of m stones, for all $m < n$, and it cannot reach a state that is associated with the Nim-game of n stones. This leads to the following definition of the Sprague-Grundy function.

Definition 4.1 (Sprague-Grundy function)**.** The Sprague-Grundy function $g : V \to \mathbb{N}$ for a state x is defined as follows:

$$g(x) = \min\{n \in \mathbb{N}_{\geq 0} : n \neq g(y), \forall y \in H(x)\}.$$

Remark 4.2. This function indeed associates states of the game of Nim with one pile of stones with itself. Since every amount of stones can be reduced to every smaller amount of stones, we have:

$$\begin{aligned} g(n) &= \min\{m \in \mathbb{N}_{\geq 0} : m \neq g(y), \forall y \in H(n)\} \\ &= \min\{m \in \mathbb{N}_{\geq 0} : m \neq g(y), \forall 0 \leq y < n\} \\ &= n. \end{aligned}$$

Corollary 4.3. *If S is a winning state for the previous player, then $h_{Nim}(g(S)) = P$ and if S is a winning state for the next player, then $h_{Nim}(g(S)) = N$, or in other words, $h_{Nim} \circ g = h$.*

Proof. First of all, it is easy to see that a terminal position x gets Sprague-Grundy value 0, and thus has $h_{Nim}(g(x)) = h_{Nim}(0) = P$:

$$\begin{aligned} g(x) &= \min\{n \in \mathbb{N}_{\geq 0} : n \neq g(y), \forall y \in H(x)\} \\ &= \min\{n \in \mathbb{N}_{\geq 0} : n \neq g(y), \forall y \in \emptyset\} \\ &= 0. \end{aligned}$$

Now note that the Sprague-Grundy function does the same as the labelling algorithm (Algorithm 3.4). Each state S that can only reach N-positions (with Sprague-Grundy value unequal to 0) gets Sprague-Grundy value 0, and thus $h_{Nim}(g(S)) = h_{Nim}(0) = P$. Each state S that can reach at least one P-position (with Sprague-Grundy value equal to 0), gets a Sprague-Grundy value unequal to 0 and thus $h_{Nim}(g(S)) = P$. □

5. Sum of games

The Sprague-Grundy function associates states of games with states of one-pile Nim. However, since the function uses recursion, it can take a very long time to calculate the Sprague-Grundy values for a game. This would go a lot faster if there is a way to combine the Sprague-Grundy values of subgames of a game. For example, a game of Nim with x piles, contains x subgames of Nim with one pile.

Definition 5.1 (Sum of games)**.** The sum G of the games $G_1, \dots, G_n$ is the game consisting of all games G_i combined, or, in other words, playing all games G_i at the same time, where at each turn a player has to make a move in exactly one of the games G_i. The game G ends when all subgames G_i have ended.

Now let G be the sum of two games G_1 and G_2, where G_1 is in state A, with $g(A) = a$, and G_2 is in state B, with $g(B) = b$. What we want, is to find a function $F(a, b)$ that associates the state (A, B) with a one pile Nim-game, without having to use the recursive Sprague-Grundy function. In other words, we want to find a non-recursive expression for the following:

$$F(a, b) = g(A, B) = \min\{n \in \mathbb{N}_{\geq 0} : n \neq F(a', b), \forall a' < a \text{ and } n \neq F(a, b'), \forall b' < b\}.$$

In the sum of two games, the game is in a P-position when both subgames are equivalent to Nim-games of the same amount of stones. In this case, the previous player can always copy what the next player does. If the next player moves one of the games to a state that is equivalent to a Nim-game of n stones, the previous player can do the same in the other game, until both games are in a terminal position. If both piles are equivalent to Nim-games of a different amount of stones, it is an N-position, since the next player can move the game to a state where both piles are equivalent to Nim-games of the same amount of stones, which is a P-position. This means that our function $F : \mathbb{N}_{\geq 0} \times \mathbb{N}_{\geq 0} \to \mathbb{N}_{\geq 0}$, needs to satisfy the following, for $a, b \in \mathbb{N}_{\geq 0}$:

$$h_{Nim}(F(a, b)) = \begin{cases} P & \text{if } a = b \\ N & \text{else.} \end{cases}$$

It follows that

$$F(a, b) \begin{cases} = 0 & \text{if } a = b \\ \neq 0 & \text{else.} \end{cases}$$

We also want to be able to recursively extend F for more than 2 piles. We define $F(a_1, \dots, a_n) = F(F(a_1, \dots, a_{n-1}), a_n)$. Since the order of piles should not matter, we want $F(a, b) = F(b, a)$ and $F(F(a, b), c) = F(a, F(b, c))$.

From the above conditions, it follows that for each $a \in \mathbb{N}_{\geq 0}$, we have

$$0 = F(0, 0) = F(0, F(a, a)) = F(F(0, a), a).$$

And thus $F(0, a) = a$.

Note that we have described an Abelian group [11] $\mathbb{N}_{\geq 0}$, with group operation $F(a, b)$, identity element 0, associativity, an inverse element a^{-1} for each element a, such that $F(a, a^{-1} = 0$ $(a^{-1} = a)$ and commutativity.

Lemma 5.2. *Each element is its own inverse in an Abelian group $(A, \oplus)$ if and only if the group is a vector space over $\mathbb{F}_2$.*

Proof. First we prove that, if each element is its own inverse in an Abelian group A, with operation $\oplus$, then the group is a vector space over $\mathbb{F}_2$, with the standard scalar multiplication on $mathbbF_2$.
To be a vector space over $\mathbb{F}_2$, $(A, \oplus)$ has to satisfy the following axioms [10].

(1) $(A, \oplus)$ is an Abelian group. This is satisfied by assumption.
(2) For all $\lambda, \mu \in \mathbb{F}_2$ and $a \in A$, we have $\lambda \times (\mu \times a) = (\lambda \cdot \mu) \times a$. This is easily checked, since $\lambda, \mu \in \{0, 1\}$:
  - $0 \times (0 \times a) = 0 \times 0 = 0$ and $(0 \cdot 0) \times a = 0 \times a = 0$.
  - $0 \times (1 \times a) = 0 \times a = 0$ and $(0 \cdot 1) \times a = 0 \times a = 0$.
  - $1 \times (0 \times a) = 1 \times 0 = 0$ and $(1 \cdot 0) \times a = 0 \times a = 0$.
  - $1 \times (1 \times a) = 1 \times a = a$ and $(1 \cdot 1) \times a = 1 \times a = a$.
(3) For all $a \in A$, we have $1 \times a = a$. This is satisfied.
(4) For all $\lambda \in \mathbb{F}_2$ and $a, b \in A$, we have $\lambda \times (a \oplus b) = (\lambda \times a) \oplus (\lambda \times b)$. This is easily checked, since $\lambda \in \{0, 1\}$:
  - $0 \times (a \oplus b) = 0$, and $(0 \times a) \oplus (0 \times b) = 0 \oplus 0 = 0$.
  - $1 \times (a \oplus b) = a \oplus b$, and $(1 \times a) \oplus (1 \times b) = a \oplus b$
(5) For all $\lambda, \mu \in \mathbb{F}_2$ and $a \in A$, we have $(\lambda + \mu) \times a = (\lambda \times a) \oplus (\mu \times a)$. This is easily checked, since $\lambda, \mu \in \{0, 1\}$:
  - $(0 + 0) \times a = 0 \times a = 0$, and $(0 \times a) \oplus (0 \times a) = 0 \oplus 0 = 0$.
  - $(0 + 1) \times a = 1 \times a = a$, and $(0 \times a) \oplus (1 \times a) = 0 \oplus a = a$.
  - $(1 + 0) \times a = 1 \times a = a$, and $(1 \times a) \oplus (0 \times a) = a \oplus 0 = a$.
  - $(1 + 1) \times a = 0 \times a = 0$, and $(1 \times a) \oplus (1 \times a) = a \oplus a = 0$.

So since $(A, \oplus)$ satisfies all these conditions, it is a vector space over $\mathbb{F}_2$.

Now we need to prove that if $(A, \oplus)$ is a vector space over $\mathbb{F}_2$, it is an Abelian group where each element is its own inverse. From axiom 1 for vector spaces we know that $(A, \oplus)$ is an Abelian group. Now we just need to show that in $(A, \oplus)$ each element a is its own inverse. We have

$$\begin{aligned} a \oplus a &= (1 \times a) \oplus (1 \times a) \\ &= (1 + 1) \times a \\ &= 0 \times a \\ &= 0. \end{aligned}$$

So for each element a, we have $a \oplus a = 0$, and thus a is its own inverse and $(A, \oplus)$ is an Abelian group where each element is its own inverse. □

So we need to define F such that $(\mathbb{N}_{\geq 0}, F)$ is a vector space over $\mathbb{F}_2$. We will do so by taking the bitwise addition on the binary notation of elements of $\mathbb{N}_{\geq 0}$. It is easily checked that this does indeed satisfy the conditions. This bitwise addition on the binary notation is, appropriately, called Nim-addition.

Definition 5.3 (Binary notation)**.** For $x \in \mathbb{N}_{\geq 0}$, we use the binary notation $(\ldots 0 x_{k(x)} \ldots x_1 x_0)$, where $x = x_{k(x)} 2^{k(x)} + \ldots + x_1 2^1 + x_0 2^0$ and $x_i \in \{0, 1\}$ for all $i \geq 0$. In this notation, we have $k(x) = \lfloor {}^2\log(x) \rfloor$, such that $2^{k(x)} \leq x < 2^{k(x)+1}$. Note that $x_i = 0$ for $i > k(x)$. We often omit these leading zeroes, and start the notation with $x_{k(x)}$.

Definition 5.4 (Nim-addition)**.** Let x and y be two natural numbers, larger or equal to 0, and let $(\ldots x_{k(x)} \ldots x_0)$ and $(\ldots y_{k(y)} \ldots y_0)$ be their binary representations. Then their Nim-sum $z = x \oplus y$ is defined as follows, where $(\ldots z_{k(z)} \ldots z_0)$ is the binary representation of z:

$$z_i = \begin{cases} 0, & \text{if } x_i = y_i \\ 1, & \text{if } x_i \neq y_i. \end{cases}$$

Example 5.5. For example, let us add 13 and 11 by Nim-addition.

We have $13 = (1110)$ and $11 = (1011)$. Adding these gives:

$$\begin{array}{r} 1110 \\ 1011 \\ \hline 0101 \end{array}$$

Since $(0101) = 5$, we have $13 \oplus 11 = 5$.

What is left, is checking whether this Nim-addition is indeed the right group-operation, since there might be other group-operations that also satisfy the conditions for a vector space over $\mathbb{F}_2$. Remember that we wanted our operation to satisfy

$$F(a,b) = g(A,B) = \min\{n \in \mathbb{N}_{\geq 0} : n \neq F(a',b), \forall a' < a \text{ and } n \neq F(a,b'), \forall b' < b\}.$$

Since this expression is the Sprague-Grundy value for the state (A,B), we need to show that the Nim-sum of the Sprague-Grundy values of these two states, $g(A) \oplus g(B)$, is equal to the Sprague-Grundy value of the combination of these games, $g(A,B)$. If we extend this to a combination of n games, we get the Sprague-Grundy theorem.

Theorem 5.6 (Sprague-Grundy Theorem)**.** *Let g_i be the Sprague-Grundy function of the game G_i for $i = 1,\dots,n$. Let G be the sum of all games G_i, where a game G_i is in state x_i. Then the Sprague-Grundy function $g(x_1,\dots,x_n)$ of G is equal to $g_1(x_1) \oplus \dots \oplus g_n(x_n)$.*

Proof. We use a similar proof as Ferguson in [4].
We have to prove two things:

- For every $b < a = g_1(x_1) \oplus \dots \oplus g_n(x_n)$ there is a state $(x'_1,\dots,x'_n)$ that can be reached from $(x_1,\dots,x_n)$, with $g_1(x'_1) \oplus \dots \oplus g_n(x'_n) = b$.
- There is no state $(x'_1,\dots,x'_n)$ that can be reached from $(x_1,\dots,x_n)$, with $g_1(x'_1) \oplus \dots \oplus g_n(x'_n) = a = g_1(x_1) \oplus \dots \oplus g_n(x_n)$.

First we show the first condition. Let $a = g_1(x_1) \oplus \dots \oplus g_n(x_n)$. Now take an arbitrary $b < a$. We have to show that there is a state $(x'_1,\dots,x'_n)$ that can be reached from $(x_1,\dots,x_n)$, with $g_1(x'_1) \oplus \dots \oplus g_n(x'_n) = b$.

Take $c = a \oplus b$. For all $l > k(c)$, we have $a_l = b_l$. Also, $a_{k(c)} \neq b_{k(c)}$. Since $a > b$, this means that $a_{k(c)} = 1$ and $b_{k(c)} = 0$.

Since $a_{k(c)} = 1$, there has to be an m such that $g_m(x_m)_{k(c)} = 1$ as well. Since Nim-addition is commutative, we can assume, without loss of generality, that this is in the first game, thus $g_1(x_1)_{k(c)} = 1$. Since $c_{k(c)} = g_1(x_1)_{k(c)} = 1$ and $c_l = 0$ for $l > k(c)$, $c \oplus g_1(x_1) < g_1(x_1)$. Since $c \oplus g_1(x_1)$ is strictly smaller than $g_1(x_1)$, we know that there is an x'_1 reachable from x_1, with $g_1(x'_1) = c \oplus g_1(x_1)$. So an allowed move would be to go from $(x_1, x_2, \dots, x_n)$ to $(x'_1, x_2, \dots, x_n)$. For the second state, the following holds.

$$\begin{aligned} g_1(x'_1) \oplus g_2(x_2) \oplus \dots \oplus g_n(x_n) &= c \oplus g_1(x_1) \oplus g_2(x_2) \oplus \dots \oplus g_n(x_n) \\ &= c \oplus a \\ &= b. \end{aligned}$$

So for every $b < a = g_1(x_1) \oplus \dots \oplus g_n(x_n)$ there is a state $(x'_1,\dots,x'_n)$ that can be reached from $(x_1,\dots,x_n)$, with $g_1(x'_1) \oplus \dots \oplus g_n(x'_n) = b$.

Now we prove the second condition by contradiction. Assume there is a $(x'_1,\dots,x'_n)$ that can be reached from $(x_1,\dots,x_n)$, with $g_1(x'_1) \oplus \dots \oplus g_n(x'_n) = a = g_1(x_1) \oplus \dots \oplus g_n(x_n)$. We know that there is exactly one i for which x'_i is different from x_i, since we can only make a move in one of the games. Since Nim-addition is commutative, we can assume, without loss of generality, that this is x'_1. This means that for $m = 2, \ldots, n$, we have $x'_m = x_m$. So we have

$$\begin{aligned} g_1(x'_1) \oplus g_2(x_2) \oplus \ldots \oplus g_n(x_n) &= g_1(x_1) \oplus g_2(x_2) \oplus \ldots \oplus g_n(x_n) \\ \Rightarrow \qquad g_1(x'_1) &= g_1(x_1). \end{aligned}$$

And this is not possible, since x'_1 is reachable from x_1, and can thus not have the same Sprague-Grundy value. So there is no state $(x'_1, \ldots, x'_n)$ that can be reached from $(x_1, \ldots, x_n)$, with $g_1(x'_1) \oplus \ldots \oplus g_n(x'_n) = a = g_1(x_1) \oplus \ldots \oplus g_n(x_n)$.

Because we have proved both the first and the second condition to be true, we have proven the Sprague-Grundy Theorem, and thus the Sprague-Grundy values of different games can be added with Nim-addition. □

Lemma 5.7. *For all $a, b \in \mathbb{N}_{\geq 0}$, $a \oplus b \leq a + b$.*

Proof. Compared to normal (binary) addition, Nim-addition does not 'carry the 1'. Since it does not do this, it means that Nim-addition can only result in an equal or smaller number than regular addition. □

Lemma 5.8. *For all $a, b \in \mathbb{N}_{\geq 0}$, if $a \oplus b = c$, then $a \oplus c = b$ and $b \oplus c = a$. In other words, Nim-subtraction is the same as Nim-addition.*

Proof. The following holds:

$$\begin{aligned} a \oplus b &= c \\ \Rightarrow \quad a \oplus a \oplus b &= a \oplus c \\ \Rightarrow \quad b &= a \oplus c. \end{aligned}$$

The equality $b \oplus c = a$ follows analogous. □

6. Optimal play

In this section, we will discuss how one can use the Sprague-Grundy function and theorem to play optimally in an impartial combinatorial game G in state S, consisting of n subgames. Recall that, if the game is in a P-position, in other words, if $g(S) = 0$, there is no optimal move for the next player. If the game is in an N-position, or in other words, if $g(S) = s \neq 0$, the optimal move is to move to a P-position, thus to a state S' with $g(S') = 0$. We will give an algorithm that describes how to move from an N-position to a P-position.

Algorithm 6.1 (Optimal move)**.**
Step 1. Consider a subgame G_m, in state S_m, with $g(S_m) = x$, such that $x_{k(s)} = 1$.
Step 2. Define $\hat{x}$, such that

$$\hat{x}_i = \begin{cases} x_i & \text{if} \quad i > k(s) \\ s_i \oplus x_i & \text{if} \quad i \leq k(s). \end{cases}$$

Step 3. Move the subgame G_m to a state $\hat{S}_m$, with $g(\hat{S}_m) = \hat{x}$.

Lemma 6.2. *For a game G, in state S, with $g(S) \neq 0$, the optimal move algorithm gives an optimal move, or in other words, gives a move to a state S', with $g(S') = 0$.*

Proof. We will prove the correctness of this algorithm by showing that the specified move is feasible and leads to a P-position (and is thus optimal).

We have to show that there is indeed a game G_m in state S_m, with $g(S_m) = x$, such that $x_{k(s)} = 1$ (Step 1). This exists, because otherwise $s_{k(s)} = g(S_1)_{k(s)} \oplus \ldots \oplus g(S_n)_{k(s)} = 0 \oplus \ldots \oplus 0 = 0$, and $s_{k(s)} = 1$ by definition of $k(s)$.

Next we show that a move in game G_m from state S_m to state $\hat{S}_m$ is feasible (Steps 2 and 3). In the binary representation of $\hat{x}$, we have $\hat{x}_i = x_i$ for $i > k(s)$, and $\hat{x}_{k(s)} = 0$, whereas $x_{k(s)} = 1$. This means that $\hat{x} < x$ and thus a move from a state S_m, with $g(S_m) = x$ to a state $\hat{S}_m$, with $g(\hat{S}_m) = \hat{x}$ is feasible.

Finally, we will show that this is indeed an optimal move. Let S' be the new state of the complete game. We have $s' = g(S') = g(S) \oplus x \oplus \hat{x}$. This means that

$$\begin{aligned} s'_i &= s_i \oplus x_i \oplus \hat{x}_i \\ &= \begin{cases} s_i \oplus x_i \oplus x_i & \text{if} \quad i > k(s) \\ s_i \oplus x_i \oplus s_i \oplus x_i & \text{if} \quad i \leq k(s). \end{cases} \\ &= \begin{cases} s_i & \text{if} \quad i > k(s) \\ 0 & \text{if} \quad i \leq k(s). \end{cases} \\ &= 0. \end{aligned}$$

The last equality holds, because $s_i = 0$ for all $i > k(s)$.
So now we have shown that $g(S') = 0$, which means S' is a P-position, and the move was an optimal move. □

For more on the game of Nim, like an optimal strategy for the misère version, we refer to Bouton [2], containing a complete mathematical theory for Nim.

7. Sprague-Grundy values of Kayles

As an example, we will now calculate the Sprague-Grundy values for Kayles. In this game, the only terminal position is when there are no more bowling pins left. Therfore $g(0) = 0$. The Sprague-Grundy function for Kayles is as follows:

$$g(x) = \min \left\{ \quad n \in \mathbb{N}_{\geq 0} : \quad \begin{array}{l} n \neq g(y) \oplus g(x-y-1), \forall 0 \leq y \leq \left\lfloor \frac{x-1}{2} \right\rfloor \\ n \neq g(y) \oplus g(x-y-2), \forall 0 \leq y \leq \left\lfloor \frac{x-2}{2} \right\rfloor \end{array} \right\}.$$

The table below contains the first 10 Sprague-Grundy values of Kayles, calculated with the C++-program in appendix A.

Amount of pins	Sprague-Grundy value
0	0
1	1
2	2
3	3
4	1
5	4
6	3
7	2
8	1
9	4

Now it is interesting to investigate whether these Sprague-Grundy values eventually become periodic.

Definition 7.1 (Periodic)**.** We say the Sprague-Grundy values of a game are periodic with period p, starting at a, if for all values $y \geq a + p$ we have $g(y) = g(y - p)$.

In fact, if we look at the first 84 Sprague-Grundy values for Kayles, displayed here from left to right and top to bottom, it seems to work towards a period of 12:

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
4	1	2	**3**	1	4	7	2	1	8	2	7
4	1	2	8	1	4	7	2	1	**4**	2	7
4	1	2	8	1	4	7	2	1	8	**6**	7
4	1	2	8	1	4	7	2	1	8	2	7

The values that are not in line with period 12 have been made bold. It turns out that the Sprague-Grundy values for Kayles are periodic with period 12, starting at 71. This was first proven by Richard Guy and Cedric Smith [9] by describing Kayles as an octal game, which we will not discuss in this thesis, since Grundy's Game cannot be described as an octal game. Below, we will give a slightly different proof from the one from Guy and Smith, which does not need the theory of octal games.

Proposition 7.2. *Starting at $a = 71$, Kayles is periodic with period $p = 12$.*

Proof. By calculating the values up to $n = 166$, we see that, in these values, the last irregularity occurs at $n = 70$. After that we have a pattern with a period of $p = 12$, like the last row of the table above. We will show that, if Kayles is periodic with period $p = 12$, starting at $a = 71$ up to 166, it will remain periodic.
We will prove this by induction.
The values from $g(71)$ to $g(166)$ are periodic, thus for $x \in \{83, \ldots, 166\}$, we have

$g(x) = g(x-12)$.
Now let us assume that for certain $n > 166$, all values from $g(71)$ up to $g(n-1)$ are periodic with period 12. This is the induction hypothesis.
We will now show that this implies that $g(n) = g(n-12)$. This amounts to the following:

(1) $\forall x < g(n-12)$: (a) $\exists 0 \le y \le \lfloor \frac{n-1}{2} \rfloor$ s.t. $g(y) \oplus g(n-y-1) = x$ or
(b) $\exists 0 \le y \le \lfloor \frac{n-2}{2} \rfloor$ s.t. $g(y) \oplus g(n-y-2) = x$.

(2) (a) $\forall 0 \le y \le \lfloor \frac{n-1}{2} \rfloor : g(y) \oplus g(n-y-1) \ne g(n-12)$ and
(b) $\forall 0 \le y \le \lfloor \frac{n-2}{2} \rfloor : g(y) \oplus g(n-y-2) \ne g(n-12)$.

We will start with condition 1. Then we have, by assumption,

$$\forall x < g(n-12): \quad (a) \quad \exists 0 \le y \le \left\lfloor \frac{n-13}{2} \right\rfloor \text{ s.t. } g(y) \oplus g(n-y-13) = x$$
$$\text{or} \quad (b) \quad \exists 0 \le y \le \left\lfloor \frac{n-14}{2} \right\rfloor \text{ s.t. } g(y) \oplus g(n-y-14) = x.$$

Lets first check it for x satisfying $1(a)$.
Since $0 \le y \le \lfloor \frac{n-13}{2} \rfloor$, the following holds for $n-y-1$:

$$\begin{aligned}
& n - \left\lfloor \frac{n-13}{2} \right\rfloor - 1 \le && n-y-1 && \le n-1 \\
\Rightarrow \quad & n - \frac{n-13}{2} - 1 \le && n-y-1 && \le n-1 \\
\Rightarrow \quad & \frac{n+11}{2} \le && n-y-1 && \le n-1 \\
\Rightarrow \quad & \frac{166+11}{2} < && n-y-1 && \le n-1 \\
\Rightarrow \quad & 88 < && n-y-1 && \le n-1.
\end{aligned}$$

It follows that $83 \le n-y-1 \le n-1$. The induction hypothesis implies that $g(n-y-1) = g(n-y-13)$. Thus, there exists $0 \le y \le \lfloor \frac{n-13}{2} \rfloor \le \lfloor \frac{n-1}{2} \rfloor$, such that $g(y) \oplus g(n-y-13) = g(y) \oplus g(n-y-1) = x$.

Now let us check it for x satisfying $1(b)$.
Since $0 \le y \le \lfloor \frac{n-14}{2} \rfloor$, the following holds for $n-y-2$:

$$\begin{aligned}
& n - \left\lfloor \frac{n-14}{2} \right\rfloor - 2 \le && n-y-2 && \le n-2 \\
\Rightarrow \quad & n - \frac{n-14}{2} - 2 \le && n-y-2 && < n-1 \\
\Rightarrow \quad & \frac{n+10}{2} \le && n-y-2 && < n-1 \\
\Rightarrow \quad & \frac{166+10}{2} < && n-y-2 && < n-1 \\
\Rightarrow \quad & 88 < && n-y-2 && < n-1.
\end{aligned}$$

It follows that $83 \le n-y-2 \le n-1$. The induction hypothesis implies that $g(n-y-2) = g(n-y-14)$. Thus, there exists $0 \le y \le \lfloor \frac{n-14}{2} \rfloor \le \lfloor \frac{n-2}{2} \rfloor$, such that $g(y) \oplus g(n-y-14) = g(y) \oplus g(n-y-2) = x$.

So since for all $x < g(n-12)$ either $1(a)$ or $1(b)$ holds, we have proven that condition 1 is true.

Now let us check condition 2. We will first prove $2(a)$:

$$\forall 0 \leq y \leq \left\lfloor \frac{n-1}{2} \right\rfloor : g(y) \oplus g(n-y-1) \neq g(n-12).$$

First, let $0 \leq y \leq \left\lfloor \frac{n-13}{2} \right\rfloor$. Then, as before, this means that $g(y) \oplus g(n-y-1) = g(y) \oplus g(n-y-13)$. By definition of $g(n-12)$, $g(y) \oplus g(n-y-1) = g(y) \oplus g(n-y-13) \neq g(n-12)$.
Next, consider $\left\lfloor \frac{n-13}{2} \right\rfloor < y \leq \left\lfloor \frac{n-1}{2} \right\rfloor$. The following holds for $n-y-1$:

$$\begin{aligned}
& n - \left\lfloor \frac{n-1}{2} \right\rfloor - 1 \leq & n-y-1 & < n - \left\lfloor \frac{n-13}{2} \right\rfloor - 1 \\
\Rightarrow \quad & \left\lceil \frac{n-1}{2} \right\rceil \leq & n-y-1 & < n - \frac{n-14}{2} - 1 \\
\Rightarrow \quad & \frac{n-1}{2} \leq & n-y-1 & < n - \frac{n-14}{2} - 1 \\
\Rightarrow \quad & \frac{166-1}{2} < & n-y-1 & < n-1-\frac{166-14}{2} \\
\Rightarrow \quad & 82 < & n-y-1 & < n-1.
\end{aligned}$$

Since $n-y-1$ is integer, this implies $83 \leq n-y-1 \leq n-1$. From the induction hypothesis, it follows that $g(n-y-1) = g(n-y-13)$.
Now, let $x := n-y-13$. Then we have

$$g(y) \oplus g(n-y-13) = g(n-x-13) \oplus g(x).$$

So now we have the following:

$$\begin{aligned}
& \left\lfloor \frac{n-13}{2} \right\rfloor < & n-x-13 & \leq \left\lfloor \frac{n-1}{2} \right\rfloor \\
\Rightarrow \quad & \left\lfloor \frac{n-13}{2} \right\rfloor < & n-x-13 & \leq \frac{n-1}{2} \\
\Rightarrow \quad & \left\lfloor \frac{n-13}{2} \right\rfloor - n + 13 < & -x & \leq \frac{n-1}{2} - n + 13 \\
\Rightarrow \quad & n - 13 - \frac{n-1}{2} \leq & x & < n - 13 - \left\lfloor \frac{n-13}{2} \right\rfloor \\
\Rightarrow \quad & \frac{n-25}{2} \leq & x & < \left\lceil \frac{n-13}{2} \right\rceil \\
\Rightarrow \quad & \frac{166-25}{2} < & x & < \left\lceil \frac{n-13}{2} \right\rceil \\
\Rightarrow \quad & 0 < & x & < \left\lceil \frac{n-13}{2} \right\rceil.
\end{aligned}$$

Since x is integer, this implies $0 < x \leq \left\lfloor \frac{n-13}{2} \right\rfloor$. As before, by definition of $g(n-12)$, this means that $g(x) \oplus g(n-x-13) \neq g(n-12)$. So, for $\left\lfloor \frac{n-13}{2} \right\rfloor < y \leq \left\lfloor \frac{n-1}{2} \right\rfloor$, it follows that $g(y) \oplus g(n-y-1) = g(y) \oplus g(n-y-13) = g(x) \oplus g(n-x-13) \neq g(n-12)$.

Part $2(b)$ follows analogously:

$$\forall 0 \leq y \leq \left\lfloor \frac{n-2}{2} \right\rfloor : g(y) \oplus g(n-y-2) \neq g(n-12).$$

First, let $0 \leq y \leq \lfloor \frac{n-14}{2} \rfloor$. As before, $g(y) \oplus g(n-y-2) = g(y) \oplus g(n-y-14)$. By definition of $g(n-12)$, $g(y) \oplus g(n-y-2) = g(y) \oplus g(n-y-14) \neq g(n-12)$. Next, consider $\lfloor \frac{n-14}{2} \rfloor < y \leq \lfloor \frac{n-2}{2} \rfloor$. The following holds for $n-y-2$:

$$\begin{aligned}
& n - \left\lfloor \frac{n-2}{2} \right\rfloor - 2 \leq && n-y-2 && < n - \left\lfloor \frac{n-14}{2} \right\rfloor - 2 \\
\Rightarrow \quad & \left\lceil \frac{n-2}{2} \right\rceil \leq && n-y-2 && < n - \frac{n-15}{2} - 2 \\
\Rightarrow \quad & \frac{n-2}{2} \leq && n-y-2 && < n - \frac{n-15}{2} - 2 \\
\Rightarrow \quad & \frac{166-2}{2} < && n-y-2 && < n-2-\frac{166-15}{2} \\
\Rightarrow \quad & 82 < && n-y-2 && < n-1.
\end{aligned}$$

Since $n-y-2$ is integer, this implies $83 \leq n-y-2 \leq n-1$. From the induction hypothesis, it follows that $g(n-y-2) = g(n-y-14)$.
Now, let $x := n-y-14$. Then we have

$$g(y) \oplus g(n-y-14) = g(n-x-14) \oplus g(x).$$

So now we have the following:

$$\begin{aligned}
& \left\lfloor \frac{n-14}{2} \right\rfloor < && n-x-14 && \leq \left\lfloor \frac{n-2}{2} \right\rfloor \\
\Rightarrow \quad & \left\lfloor \frac{n-14}{2} \right\rfloor < && n-x-14 && \leq \frac{n-2}{2} \\
\Rightarrow \quad & \left\lfloor \frac{n-14}{2} \right\rfloor - n + 14 < && -x && \leq \frac{n-2}{2} - n + 14 \\
\Rightarrow \quad & n - 14 - \frac{n-2}{2} \leq && x && < n - 14 - \left\lfloor \frac{n-14}{2} \right\rfloor \\
\Rightarrow \quad & \frac{n-26}{2} \leq && x && < \left\lceil \frac{n-14}{2} \right\rceil \\
\Rightarrow \quad & \frac{166-26}{2} < && x && < \left\lceil \frac{n-14}{2} \right\rceil \\
\Rightarrow \quad & 0 < && x && < \left\lceil \frac{n-14}{2} \right\rceil.
\end{aligned}$$

Since x is integer, this implies $0 < x \leq \lfloor \frac{n-14}{2} \rfloor$. As before, by definition of $g(n-12)$, this means that $g(x) \oplus g(n-x-14) \neq g(n-12)$. So, for $\lfloor \frac{n-14}{2} \rfloor < y \leq \lfloor \frac{n-2}{2} \rfloor$, it follows that $g(y) \oplus g(n-y-2) = g(y) \oplus g(n-y-14) = g(x) \oplus g(n-x-14) \neq g(n-12)$.

We have now proven condition 1 and condition 2 to be true, and therefore, Kayles is periodic with period 12, starting at 71. □

8. Sprague-Grundy values of Grundy's Game

In Grundy's games, piles of 1 and 2 stones are the terminal positions and thus have Sprague-Grundy value equal to 0. The Sprague-Grundy function for Grundy's Game is as follows:

$$g(x) = \min\{n \in \mathbb{N}_{\geq 0} : n \neq g(y) \oplus g(x-y), \forall 1 \leq y \leq \left\lfloor \frac{x-1}{2} \right\rfloor\}.$$

Below, we give the first 30 Sprague-Grundy values of Grundy's Game, calculated with the C++-program in Appendix B.

Amount of stones	Sprague-Grundy value
1	0
2	0
3	1
4	0
5	2
6	1
7	0
8	2
9	1
10	0
11	2
12	1
13	3
14	2
15	1
16	3
17	2
18	4
19	3
20	0
21	4
22	3
23	0
24	4
25	3
26	0
27	4
28	1
29	2
30	3

Interestingly though, it is not known if these values eventually become periodic [7] [8], which is the case for many other games, like Kayles (Proposition 7.2). The first 2^{35} Sprague-Grundy values of Grundy's Game have been calculated by Achim Flammenkamp [5], but no periodicity has been found yet.

9. Adaptations to Grundy's Game

As it is not yet known if Grundy's Game becomes periodic, it might yield some insight to consider different adaptations to Grundy's Game. We will check whether these adaptations do become periodic, and may even get some insights into why Grundy's Game does not become periodic relatively fast.

9.1. Grundy's Game with equal divisions.

A natural adaptation to Grundy's Game would be to also admit divisions into two equal piles. Then only piles of one stone are a terminal position.
In the table below we give the first 10 Sprague-Grundy values of this game, calculated with the C++-program in Appendix C.

Amount of stones	Sprague-Grundy value
1	0
2	1
3	0
4	1
5	0
6	1
7	0
8	1
9	0
10	1

Proposition 9.1. *The Sprague-Grundy values of the adapted Grundy's Game, that allows dividing a pile into two equal piles, are periodic with period 2.*

Proof. We will prove this by induction.
A pile of one stone has Sprague-Grundy value 0, since it is terminal. We can then calculate the Sprague-Grundy value of a pile of two stones. A pile of two stones can only be divided into two piles of one stone, and this has Sprague-Grundy value $0 \oplus 0 = 0$. This gives the pile of two stones the Sprague-Grundy value of

$$\begin{aligned} g(2) &= \min\{n \in \mathbb{N}_{\geq 0} : n \neq g(y) \forall y \in H(2)\} \\ &= \min\{n \in \mathbb{N}_{\geq 0} : n \neq g(1,1)\} \\ &= \min\{n \in \mathbb{N}_{\geq 0} : n \neq 0\} \\ &= 1. \end{aligned}$$

Now let us assume that the Spraque-Grundy value is equal to 1 for all even numbers smaller than n, and equal to 0 for all odd numbers smaller than n, for a certain $n > 2$. This is the induction hypothesis.
There are two options: n is even or n is odd.
n **is even.** If n is even, the n stones can be divided into two piles of both an even amount of stones, this has Sprague-Grundy value $1 \oplus 1 = 0$, or two piles of both an odd amount of stones, this has Sprague-Grundy value $0 \oplus 0 = 0$ as well. This gives the pile of n stones a Sprague-Grundy value of

$$\begin{aligned} g(n) &= \min\{n \in \mathbb{N}_{\geq 0} : n \neq g(y) \forall y \in H(n)\} \\ &= \min\{n \in \mathbb{N}_{\geq 0} : n \neq 0\} \\ &= 1. \end{aligned}$$

n **is odd.** If n is odd the n stones can only be divided into one pile of an even amount of stones and one pile of an odd amount of stones, giving Sprague-Grundy value $1 \oplus 0 = 1$. This gives the pile of n stones a Sprague-Grundy value of

$$\begin{aligned} g(n) &= \min\{n \in \mathbb{N}_{\geq 0} : n \neq g(y) \forall y \in H(n)\} \\ &= \min\{n \in \mathbb{N}_{\geq 0} : n \neq 1\} \\ &= 0. \end{aligned}$$

This means that each pile with an odd amount of stones has Sprague-Grundy value 1 and each pile with an even amount of stones has Sprague-Grundy value 0, and thus the Sprague-Grundy values for this game are periodic, with period 2. □

9.2. **Grundy's Game, with $g(2) = 1$.**
The previous adaptation reduced Grundy's Game to a very simple, immediately periodic game. Therefore we will now consider some games that are a mixture between Grundy's Game and the previous adaptation. For example, maybe the problem with Grundy's Game is that the pile of two stones cannot be divided. Therefore we will consider Grundy's Game, with the additional rule that a pile of two stones can be divided into two piles of one stone, so $g(2) = 1$.
In the table below we give the first 10 Sprague-Grundy values of this game, calculated with the C++-program in Appendix D.

Amount of stones	Sprague-Grundy value
1	0
2	1
3	0
4	1
5	0
6	1
7	0
8	1
9	0
10	1

Note that these are exactly the same values as we had with our previous adaptation. The proof for periodicity is analogous.

9.3. **Grundy's Game with equal divisions, but without division of a pile of two stones.**
So is it really the division of the pile of two stones that makes Grundy's Game so difficult? We will now consider the game that does allow divisions into two equal piles for piles larger than 2, but does not allow the division of a pile of two stones.
In the table below we give the first 10 Sprague-Grundy values of this game, calculated with the C++-program in Appendix E.

Amount of stones	Sprague-Grundy value
1	0
2	0
3	1
4	2
5	0
6	1
7	2
8	3
9	1
10	2

This does look a bit different from the previous adaptations. In fact, it does not appear to become periodic anytime soon; looking at the first 1000 Sprague-Grundy values we cannot find any periodicity.

9.4. **Grundy's Game, where a division into two piles differing one is not allowed.**
To really check if this is the problem with Grundy's Game, we can think of more games that do not allow the division of a pile of two stones. For example, we can consider the game that is like Grundy's Game, but in addition does not allow divisions into two piles where the difference is one.
In the table below we give the first 16 Sprague-Grundy values of this game, calculated with the C++-program in Appendix F.

Amount of stones	Sprague-Grundy value
1	0
2	0
3	0
4	1
5	0
6	2
7	1
8	3
9	0
10	2
11	1
12	3
13	0
14	2
15	1
16	3

Though it is not directly obvious from these first 16 values, the Sprague-Grundy values for this game do become periodic pretty fast.

Proposition 9.2. *The Sprague-Grundy values of the adapted Grundy's Game, that does not allow dividing a pile into two piles with one stone difference, are periodic, starting at 5, with period 4.*

Proof. We will prove this by induction.
Let us assume that for a certain $n > 16$, the Spraque-Grundy value for $x \in \{5, \ldots, n-1\}$ is equal to

$$g(x) = \begin{cases} 3 & \text{if} \quad x \equiv 0 \mod 4 \\ 0 & \text{if} \quad x \equiv 1 \mod 4 \\ 2 & \text{if} \quad x \equiv 2 \mod 4 \\ 1 & \text{if} \quad x \equiv 3 \mod 4. \end{cases}$$

This is the induction hypothesis. We will now prove that, if this is the Sprague-Grundy function for all $x \in \{5, \ldots, n-1\}$, it is also the Sprague-Grundy function for n.
From now on, when we talk about a pile of $x \mod 4$ stones, we assume this pile has more than 4 stones. We consider four different options.

n mod 4 = 1.

If $n \mod 4 = 1$, the stones can be divided into the following piles:

First pile	Second pile	Sprague-Grundy value
1	$0 \mod 4$	$0 \oplus 3 = 3$
2	$3 \mod 4$	$0 \oplus 1 = 1$
3	$2 \mod 4$	$0 \oplus 2 = 2$
4	$1 \mod 4$	$1 \oplus 0 = 1$
$1 \mod 4$	$0 \mod 4$	$0 \oplus 3 = 3$
$2 \mod 4$	$3 \mod 4$	$2 \oplus 1 = 3$

This gives:

$$\begin{aligned} g(n) &= \min\{n \in \mathbb{N}_{\geq 0} : n \neq g(y) \forall y \in H(n)\} \\ &= \min\{n \in \mathbb{N}_{\geq 0} : n \neq 1, 2, 3\} \\ &= 0. \end{aligned}$$

n mod 4 = 2.

If $n \mod 4 = 2$, the stones can be divided into the following piles:

First pile	Second pile	Sprague-Grundy value
1	$1 \mod 4$	$0 \oplus 0 = 0$
2	$0 \mod 4$	$0 \oplus 3 = 3$
3	$3 \mod 4$	$0 \oplus 1 = 1$
4	$2 \mod 4$	$1 \oplus 2 = 3$
$1 \mod 4$	$1 \mod 4$	$0 \oplus 0 = 0$
$2 \mod 4$	$0 \mod 4$	$2 \oplus 3 = 1$
$3 \mod 4$	$3 \mod 4$	$1 \oplus 1 = 0$

This gives:

$$\begin{aligned} g(n) &= \min\{n \in \mathbb{N}_{\geq 0} : n \neq g(y) \forall y \in H(n)\} \\ &= \min\{n \in \mathbb{N}_{\geq 0} : n \neq 0, 1, 3\} \\ &= 2. \end{aligned}$$

n mod 4 = 3.

If $n \mod 4 = 3$, the stones can be divided into the following piles:

First pile	Second pile	Sprague-Grundy value
1	$2 \mod 4$	$0 \oplus 2 = 2$
2	$1 \mod 4$	$0 \oplus 0 = 0$
3	$0 \mod 4$	$0 \oplus 3 = 3$
4	$3 \mod 4$	$1 \oplus 1 = 0$
$1 \mod 4$	$2 \mod 4$	$0 \oplus 2 = 2$
$3 \mod 4$	$0 \mod 4$	$1 \oplus 3 = 2$

This gives:

$$\begin{aligned} g(n) &= \min\{n \in \mathbb{N}_{\geq 0} : n \neq g(y) \forall y \in H(n)\} \\ &= \min\{n \in \mathbb{N}_{\geq 0} : n \neq 0, 2, 3\} \\ &= 1. \end{aligned}$$

n mod 4 = 0.
If $n \mod 4 = 0$, the stones can be divided into the following piles:

First pile	Second pile	Sprague-Grundy value
1	$3 \mod 4$	$0 \oplus 1 = 1$
2	$2 \mod 4$	$0 \oplus 2 = 2$
3	$1 \mod 4$	$0 \oplus 0 = 0$
4	$0 \mod 4$	$1 \oplus 3 = 2$
$1 \mod 4$	$3 \mod 4$	$0 \oplus 1 = 1$
$2 \mod 4$	$2 \mod 4$	$2 \oplus 2 = 0$
$0 \mod 4$	$0 \mod 4$	$3 \oplus 3 = 0$

This gives:

$$\begin{aligned} g(n) &= \min\{n \in \mathbb{N}_{\geq 0} : n \neq g(y) \forall y \in H(n)\} \\ &= \min\{n \in \mathbb{N}_{\geq 0} : n \neq 0, 1, 2\} \\ &= 3. \end{aligned}$$

Hence, if, for n larger than 16, we have the following Sprague-Grundy values for all values smaller than n and larger than 4:

$$g(n) = \begin{cases} 3 & \text{if} \quad n \equiv 0 \mod 4 \\ 0 & \text{if} \quad n \equiv 1 \mod 4 \\ 2 & \text{if} \quad n \equiv 2 \mod 4 \\ 1 & \text{if} \quad n \equiv 3 \mod 4. \end{cases}$$

Then it is is also true for n. Thus the Sprague-Grundy values of the adapted Grundy's Game, that does not allow dividing a pile into two piles with one stone difference, are periodic, starting at 5, with period 4. □

9.5. **Grundy's Game, where a division into two piles differing one or two is not allowed.**
Adding an extra restriction to Grundy's Game (not allowing separation into piles differing one), actually makes it periodic. So what if we add more restrictions to Grundy's Game and do not allow separation into piles differing one or two?
In the table below we give the first 12 Sprague-Grundy values of this game, calculated with the C++-program in Appendix G.

Amount of stones	Sprague-Grundy value
1	0
2	0
3	0
4	0
5	1
6	0
7	2
8	1
9	3
10	0
11	4
12	2

It does not seem as if this game will become periodic. In fact, looking at even more restrictions, disallowing not only piles differing one and two, but also three, or even more, still does not give a game for which the Sprague-Grundy values appear to become periodic.

9.6. **Grundy's Game with division into three piles.**
The last adaptation that we will consider is the adaptation where we divide a pile into three piles. Naturally, this gives three different options: all piles have to consist of a different amount of stones, only two piles can consist of the same amount of stones and all piles can consist of the same amount of stones.

9.6.1. *All piles have to consist of a different amount of stones.*
In the table below we give the first 10 Sprague-Grundy values of this game, calculated with the C++-program in Appendix H.

Amount of stones	Sprague-Grundy value
1	0
2	0
3	0
4	0
5	0
6	1
7	1
8	1
9	2
10	2

It seems as though this game keeps going like this, with Sprague-Grundy value $g(n) = \left\lfloor \frac{n-3}{3} \right\rfloor$, however, this goes wrong for some values, for example $g(13) = 0$. We have calculated the values up to $g(1750)$, and it appears as though the 'wrong' values keep occurring.

9.6.2. *Two piles may consist of the same amount of stones.*
In the table below we give the first 10 Sprague-Grundy values of this game, calculated with the C++-program in Appendix I.

Amount of stones	Sprague-Grundy value
1	0
2	0
3	0
4	1
5	1
6	2
7	2
8	3
9	3
10	4

It seems as though this game keeps going like this, with Sprague-Grundy value $g(n) = \left\lfloor \frac{n-2}{2} \right\rfloor$, however, like before, this goes wrong for some values, for example $g(12) = 1$. We have calculated the values up to $g(1750)$ for this game as well, and it appears as though the 'wrong' values keep occurring.

9.6.3. *All piles may consist of the same amount of stones.*
In the table below we give the first 10 Sprague-Grundy values of this game, calculated with the C++-program in Appendix J.

Amount of stones	Sprague-Grundy value
1	0
2	0
3	1
4	1
5	2
6	2
7	3
8	3
9	4
10	4

Proposition 9.3. *The Sprague-Grundy values for this game are as follows:*

$$g(n) = \left\lfloor \frac{n-1}{2} \right\rfloor .$$

Proof. We will prove this by induction.
We know that it holds up to $g(10)$.
Now let us assume that for certain $n > 10$, $g(x) = \left\lfloor \frac{x-1}{2} \right\rfloor$, for all $x \leq n-1$. This is the induction hypothesis.
We will now show that it also holds for $g(n)$.
By definition of the Sprague-Grundy function, we have, for all $1 \leq x < \frac{n}{2}$, or $1 \leq x \leq \left\lfloor \frac{n-1}{2} \right\rfloor$

$$g(n) \neq g(n-2x) \oplus g\left(\frac{n-2x}{2}\right) \oplus g\left(\frac{n-2x}{2}\right).$$

Since, with Nim-addition, each element is its own inverse, the following holds for all $1 \leq x \leq \left\lfloor \frac{n-1}{2} \right\rfloor$

$$g(n) \neq g(n-2x).$$

By the induction hypothesis, we have, for all $1 \leq x \leq \left\lfloor \frac{n-1}{2} \right\rfloor$

$$\begin{aligned} g(n-2x) &= \left\lfloor \frac{n-2x-1}{2} \right\rfloor \\ &= \left\lfloor \frac{n-1}{2} \right\rfloor - x. \end{aligned}$$

This takes on every value between 0 and $\left\lfloor \frac{n-1}{2} \right\rfloor - 1$. So we have, for all $0 \leq z \leq \left\lfloor \frac{n-1}{2} \right\rfloor - 1$,

$$g(n) \neq z.$$

By definition of the Sprague-Grundy function, it follows that

$$g(n) \geq \left\lfloor \frac{n-1}{2} \right\rfloor .$$

Now we have to show that there for any $x, y \geq 1$, with $x + y \leq n-1$, $g(x) \oplus g(y) \oplus g(n-x-y) \neq \left\lfloor \frac{n-1}{2} \right\rfloor$.

Let $x, y \geq 1$ and $x + y \leq n - 1$. From the induction hypothesis, it follows that

$$\begin{aligned} g(x) \oplus g(y) \oplus g(n-x-y) &= \left\lfloor \frac{x-1}{2} \right\rfloor \oplus \left\lfloor \frac{y-1}{2} \right\rfloor \oplus \left\lfloor \frac{n-x-y-1}{2} \right\rfloor \\ &\leq \left\lfloor \frac{x-1}{2} \right\rfloor + \left\lfloor \frac{y-1}{2} \right\rfloor + \left\lfloor \frac{n-x-y-1}{2} \right\rfloor \\ &\leq \frac{x-1}{2} + \frac{y-1}{2} + \frac{n-x-y-1}{2} \\ &\leq \frac{n-3}{2} \\ &< \left\lfloor \frac{n-1}{2} \right\rfloor . \end{aligned}$$

Note that we used Lemma 5.7. It follows that, if $g(x) = \left\lfloor \frac{x-1}{2} \right\rfloor$ for $x \in \{1, \ldots, n-1\}$, also $g(n) = \left\lfloor \frac{n-1}{2} \right\rfloor$. □

10. Analyzing Grundy's Game

In this section we will see whether we can say anything about the periodicity of Grundy's Game. We will start with a proposition.

Proposition 10.1. *If the Sprague-Grundy values of Grundy's Game become periodic with period p ($p \geq 1$), starting at a certain value a ($a \geq 1$), and stay periodic up until $b = 2a + 3p - 2$, then the Sprague-Grundy values of Grundy's Game will remain periodic.*

Proof. We will prove this by induction. Note that the proof is very similar to the proof of Proposition 7.2.
It is given that the values from $g(a)$ to $g(2a + 3p - 2)$ are periodic, and thus for $x \in \{a + p, \ldots, 2a + 3p - 2\}$, we have $g(x) = g(x - p)$.
Now let us assume that for certain $n > 2a + 3p - 2$, all values from $g(a)$ up to $g(n - 1)$ are periodic with period p. This is the induction hypothesis.
We will now show that this implies $g(n) = g(n - p)$. This amounts to showing the following:

(1) $\forall x < g(n-p) : \exists 1 \leq y \leq \lfloor \frac{n-1}{2} \rfloor$ s.t. $g(y) \oplus g(n-y) = x$
(2) $\forall 1 \leq y \leq \lfloor \frac{n-1}{2} \rfloor : g(y) \oplus g(n-y) \neq g(n-p)$.

We will start with condition 1. Then we have, by assumption:

$$\forall x < g(n-p) : \exists 1 \leq y \leq \left\lfloor \frac{n-p-1}{2} \right\rfloor \text{ s.t. } g(y) \oplus g(n-y-p) = x.$$

Since $1 \leq y \leq \lfloor \frac{n-p-1}{2} \rfloor$, the following holds for $n - y$:

$$\begin{aligned}
& n - \left\lfloor \frac{n-p-1}{2} \right\rfloor \leq && n - y && \leq n - 1 \\
\Rightarrow \quad & n - \frac{n-p-1}{2} \leq && n - y && \leq n - 1 \\
\Rightarrow \quad & \frac{n+p+1}{2} \leq && n - y && \leq n - 1 \\
\Rightarrow \quad & \frac{2a+4p-1}{2} < && n - y && \leq n - 1 \\
\Rightarrow \quad & a + p < && n - y && \leq n - 1.
\end{aligned}$$

From the induction hypothesis, it follows that $g(n-y) = g(n-y-p)$. Thus, there exists $1 \leq y \leq \lfloor \frac{n-p-1}{2} \rfloor \leq \lfloor \frac{n-1}{2} \rfloor$, such that $g(y) \oplus g(n-y-p) = g(y) \oplus g(n-y) = x$. Since this holds for all $x < g(n - p)$, we have proven that condition 1 is true.

Now let us check condition 2:

$$\forall 1 \leq y \leq \left\lfloor \frac{n-1}{2} \right\rfloor : g(y) \oplus g(n-y) \neq g(n-p).$$

First, let $1 \leq y \leq \lfloor \frac{n-p-1}{2} \rfloor$. Now, as before, $g(y) \oplus g(n-y) = g(y) \oplus g(n-y-p)$. By definition of $g(n-p)$, it holds that $g(y) \oplus g(n-y) = g(y) \oplus g(n-y-p) \neq g(n-p)$.

So now, let us consider $\lfloor \frac{n-p-1}{2} \rfloor < y \leq \lfloor \frac{n-1}{2} \rfloor$. The following holds for y:

$$\begin{aligned} \left\lfloor \frac{n-p-1}{2} \right\rfloor < \quad & y \quad \leq \left\lfloor \frac{n-1}{2} \right\rfloor \\ \Rightarrow \quad \left\lfloor \frac{2a+2p-3}{2} \right\rfloor < \quad & y \quad \leq \frac{n-1}{2} \\ \Rightarrow \quad a+p-1 < \quad & y \quad \leq n-1 \\ \Rightarrow \quad a+p \leq \quad & y \quad < n. \end{aligned}$$

From the induction hypothesis, it follows that $g(y) = g(y-p)$. Thus, for $\lfloor \frac{n-p-1}{2} \rfloor < y \leq \lfloor \frac{n-1}{2} \rfloor$, it holds that

$$g(y) \oplus g(n-y) = g(y-p) \oplus g(n-y) = g(y-p) \oplus g(n-p-(y-p)).$$

Also, for $y-p$,

$$\begin{aligned} \left\lfloor \frac{n-p-1}{2} \right\rfloor - p < \quad & y-p \quad \leq \left\lfloor \frac{n-1}{2} \right\rfloor - p \\ \Rightarrow \quad \left\lfloor \frac{2a+2p-3}{2} \right\rfloor - p < \quad & y-p \quad \leq \frac{n-1}{2} - p \\ \Rightarrow \quad a-1 < \quad & y-p \quad \leq \frac{n-1}{2} - \frac{p}{2} \\ \Rightarrow \quad 1 \leq \quad & y-p \quad \leq \left\lfloor \frac{n-p-1}{2} \right\rfloor. \end{aligned}$$

Thus, for $y-p$, we have $g(y-p) \oplus g(n-p-(y-p)) \neq g(n-p)$. It follows that, for $\lfloor \frac{n-p-1}{2} \rfloor < y \leq \lfloor \frac{n-1}{2} \rfloor$,

$$g(y) \oplus g(n-y) = g(y-p) \oplus g(n-p-(y-p)) \neq g(n-p).$$

We have proven both condition 1 and condition 2 to be true, and thus the Sprague-Grundy values of Grundy's game will remain periodic with period p, if they are periodic starting at a up to $b = 2a+3p-2$. □

Lemma 10.2. *Grundy's Game cannot have period p if there exists a $k \in \mathbb{N}_{>0}$ with $g(kp) = 0$.*

Proof. Suppose the lemma is not true and Grundy's Game eventually has period p, and there exists a $k \in \mathbb{N}_{>0}$ with $g(kp) = 0$.

Since Grundy's Game has period p, we have $g(n) = g(n-kp)$ for large enough n.

By definition of the Sprague-Grundy function, $g(n) \neq g(n-kp) \oplus g(kp)$. However, since $g(kp) = 0$, it follows that $g(n) \neq g(n-kp)$.

This is a contradiction and thus Grundy's Game cannot have period p if there exists a $k \in \mathbb{N}_{>0}$ with $g(kp) = 0$. □

This lemma allows us to eliminate a lot of possible periods.

Corollary 10.3. *The period of Grundy's Game cannot be 3.*

Proof. Take $k = 90$, then we have $g(3k) = g(270) = 0$. From the lemma it now follows that 3 cannot be the period of Grundy's Game. □

This is very interesting, since, as Berlekamp, Conway and Guy write, 'the strong tendency to period 3 continues as far as values have been calculated' [1]. Although the proof is relatively simple, we did not find Lemma 10.2 in the literature.

The largest value that has been calculated [5] that has Sprague-Grundy value equal to 0 is 1222. Since we know all values below 1222 that have Sprague-Grundy value equal to 0, we can make a list of values that cannot be the period of Grundy's Game.

Corollary 10.4. *The values in the following table cannot be the period of the Sprague-Grundy values of Grundy's Game.*

p	kp
1	1
2	2
3	270
4	4
5	10
6	270
7	7
8	288
9	270
10	10
11	682
12	276
13	26
14	392
15	270
16	288
17	340
18	270
19	285
20	20
21	273
22	682
23	23
24	288
25	50

p	kp
26	26
27	270
28	392
30	270
31	682
32	288
34	340
35	630
36	288
39	273
42	630
45	270
46	276
47	282
48	288
49	392
50	50
52	676
53	53
54	270
56	392
57	285
62	682
63	630
68	340

p	kp
69	276
70	630
71	639
72	288
73	365
79	316
85	340
90	270
91	273
92	276
94	282
95	285
96	288
98	392
105	630
106	636
126	630
135	270
137	685
138	276
141	282
144	288
158	316
159	636
167	334

p	kp
169	676
170	340
173	346
181	362
193	386
196	392
210	630
211	633
212	636
213	639
233	932
270	270
273	273
276	276
282	282
283	566
285	285
288	288
315	630
316	316
318	636
334	334
337	337
338	676
340	340

p	kp
341	682
346	346
359	359
362	362
365	365
386	386
389	389
392	392
463	926
466	932
566	566
611	1222
630	630
633	633
636	636
639	639
673	673
676	676
682	682
685	685
923	923
926	926
929	929
932	932
1222	1222

So the smallest value that can still be the period of Grundy's Game is 29. However, if we keep to our original intuition that the period has a tendency towards 3, it could also still be 33, which is at least divisible by 3.

11. Sparse set

In *Winning Ways for your Mathematical Plays* [1] the notion of a sparse set in the Sprague-Grundy values of games is introduced, with the objective of giving a plausible reasoning that the Sprague-Grundy values will eventually become bounded and thus periodic.

Definition 11.1 (Sparse set)**.** A sparse set contains only values that do not occur often (we will call these values rare) and satisfy the following conditions (we will call values that are not rare common).

(1) Adding two rare values by Nim-addition gives a rare value.
(2) Adding two common values by Nim-addition gives a rare value.
(3) Adding a rare and a common value by Nim-addition gives a common value.

We compute the Sprague-Grundy value by taking the first number that is not excluded. Since most values are common, and adding two common values gives a rare value, we exclude a lot of rare values. Therefore, the chance that the lowest number that is not excluded is a common value, is a lot larger than the chance that it is a rare value. Rare values will thus appear less and less and hopefully die out completely, which will give us more insight into the Sprague-Grundy values.

Assumption 11.2. *If values from a sparse set do not occur often, they will eventually die out completely.*

So now we have to define these sparse and common sets for Grundy's Game.

Lemma 11.3. 0 *is contained in any sparse set of Grundy's Game.*

Proof. We know that $x \oplus x = 0$ for each $x \in \mathbb{N}_{\geq 0}$. There are two options: x is common or x is rare. However, in both cases, it follows from the definition of the sparse set, that 0 is rare, and thus contained in any sparse set for Grundy's Game. □

However, we cannot say much about other values and whether they are rare or common. What we can do, is calculate the frequencies of $0, 1, \ldots, 25$ in the first 10.000 values of the Sprague-Grundy values of Grundy's Game.

Value	Frequency
0	41
1	54
2	359
3	336
4	250
5	232
6	13
7	20
8	167

Value	Frequency
9	157
10	57
11	48
12	73
13	62
14	163
15	153
16	1018
17	1046

Value	Frequency
18	21
19	13
20	25
21	26
22	532
23	568
24	14
25	7

Now we have to make a decision which values we will call common (meaning they appear a lot) and which values we will call rare. Let us take the (arbitrary) division of values that appear 100 or more times and values that appear less than 100 times. This gives as common values $2, 3, 4, 5, 8, 9, 14, 15, 16, 17, 22$ and 23. We have to study what these values have in common.

First, we recall that Nim-addition is done based on the binary expansion of numbers. It thus would make sense to look at the binary representations. We also notice that whether an odd value is rare or common seems to depend on the even value before it. If an even value n is rare, than the odd value $n+1$ is rare as well. This would imply that the last digit of the binary representation does not influence whether a value is rare or common.

In fact, we can make a division into two sets, a rare and a common set, by saying that the rare set for Grundy's Game consists of the values, that in their binary expansions, have an even number of ones, excluding the last digit.

Although this may seem like an arbitrary division into a rare and a common set, it seems to hold, also when more values are calculated. For example, when studying the 2^{35} values that have been calculated up to now [5], we can see that this separation into a rare and common set makes a lot of sense.

value	frequency	rare\common	value	frequency	rare\common
0	42	rare	39	812035154	common
1	54	rare	40	10	rare
2	235895639	common	41	12	rare
3	235922007	common	42	289917958	common
4	280693375	common	43	289907373	common
5	280895124	common	44	207300631	common
6	13	rare	45	207296837	common
7	20	rare	46	15	rare
8	211258329	common	47	12	rare
9	211373705	common	48	14	rare
10	62	rare	49	10	rare
11	51	rare	50	1227313476	common
12	97	rare	51	1227222541	common
13	77	rare	52	693170502	common
14	358751606	common	53	693248697	common
15	359055301	common	54	3	rare
16	1503739408	common	55	1	rare
17	1503699812	common	56	290176385	common
18	21	rare	57	290145505	common
19	13	rare	58	12	rare
20	25	rare	59	12	rare
21	26	rare	60	22	rare
22	825636790	common	61	14	rare
23	825656992	common	62	226602151	common
24	19	rare	63	226630568	common
25	13	rare	64	478307463	common
26	319548008	common	65	478250973	common
27	319478576	common	66	28	rare
28	185149233	common	67	32	rare
29	185115516	common	68	31	rare
30	18	rare	69	28	rare
31	17	rare	70	465913616	common
32	1432455778	common	71	465941487	common
33	1432555345	common	72	16	rare
34	26	rare	73	20	rare
35	23	rare	74	368783312	common
36	32	rare	75	368840406	common
37	32	rare	76	275915447	common
38	811959642	common	77	275950754	common

Definition 11.4 (Rare set for Grundy's Game)**.** The rare set for Grundy's Game consists of the values that in their binary expansions have an even number of ones, excluding the last digit.

For example, 0, 1, 6, 7, 10 and 11 are contained in the rare set for Grundy's Game, while 2, 3, 4, 5, 8 and 9 are not.

Lemma 11.5. *The rare set for Grundy's Game satisfies the conditions for a sparse set.*

Proof. Since we do not consider the last digit of the binary expansion, and Nim-addition is a bitwise addition, we can disregard the last digit completely. Now let us denote the number of ones in the binary expansion of x by $\mathbb{1}_x$ and the number of ones that occur on the same place in the binary expansions of x and y by $\mathbb{1}_{xy}$. Since $1 \oplus 1 = 0 \oplus 0 = 0$ and $1 \oplus 0 = 1$, we have for $z = x \oplus y$ that $\mathbb{1}_z = \mathbb{1}_x + \mathbb{1}_y - 2 \times \mathbb{1}_{xy}$.
Condition 1.
Let x and y be two rare values. All terms in $\mathbb{1}_z = \mathbb{1}_x + \mathbb{1}_y - 2 \times \mathbb{1}_{xy}$ are even, and thus $\mathbb{1}_z$ is even, and thus z is rare.
Condition 2.
Let x and y be two common values. The first two terms in $\mathbb{1}_z = \mathbb{1}_x + \mathbb{1}_y - 2 \times \mathbb{1}_{xy}$ are odd and the last term is even, and thus $\mathbb{1}_z$ is even, and thus z is rare.
Condition 3.
Let x be a common value and y be a rare value. The first term in $\mathbb{1}_z = \mathbb{1}_x + \mathbb{1}_y - 2 \times \mathbb{1}_{xy}$ is odd and the last two terms are even, and thus $\mathbb{1}_z$ is odd, and thus z is common. □

Lemma 11.6. *If the rare values of Grundy's Game eventually die out, the values of Grundy's Game will remain bounded.*

Proof. Assume there are n values x, for which $g(x)$ is rare.

A common value is only achieved when we add a rare value and a common value, and thus the Sprague-Grundy values of the set $H(x) = \{y : y \text{ can be reached from } x\}$ can contain at most n common values for each x (since there are only n rare values). This means that $g(x) = \min\{n \in \mathbb{N}_{\geq 0} : n \neq g(y) \forall y \in H(x)\}$ is at most the $n+1$st common value. So, if there are n rare values, the values of Grundy's Game will be bounded by the $n+1$st common value. □

Lemma 11.7. *If in Grundy's Game the rare values die out, it will become periodic.*

Proof. Let R be the largest state for which $g(R)$ is rare. Let $m \in \mathbb{N}$, such that $g(n) \leq m$ for all $n > R$ (this m exists, by Lemma 11.6). Since $g(n)$ can only be a common value for $n > R$, let us assume that there are k common values c for which $c \leq m$.

Since the rare values have died out, we only have to check which common values are excluded to compute the next value. Common values are only achieved by adding a rare and a common value and thus we only have to consider $g(x) \oplus g(n-x)$, where $x \leq R$.

Now consider the sequences $(g(R+1), \dots, g(2R)), (g(R+2), \dots, g(2R+1)), \dots,$ $(g(R+k^R+1), \dots g(2R+k^R)))$. These k^R+1 sequences each have length R, and since for all $n > R$, there are k possible values for $g(n)$, there are exactly k^R different possible sequences of length R. We have k^R+1 sequences, which means that at least two of these sequences are equal, lets say $(g(R+y), \dots, g(2R+y-1))$ and $(g(R+z), \dots, g(2R+z-1))$. Since these sequences are equal, $g(2R+y)$ and $g(2R+z)$ are equal too, since these values only depend on $g(x) \oplus g(n-x)$, where $1 \leq x \leq R$

and n is equal to $2R+y$ and $2R+z$ respectively. These values $g(x) \oplus g(n-x)$ are equal in the two sequences, since $g(2R+y-x) = g(2R+z-x)$ for $1 \leq x \leq R$. Now the sequences $(g(R+y+1), \dots, g(2R+y))$ and $(g(R+z+1), \dots, g(2R+z))$ are equal as well, and thus $g(2R+y+1) = g(2R+z+1)$ in the same way. This holds for all subsequent values, which implies periodicity. □

Remark 11.8. If $g(R)$ is a rare value and all values from $g(R+1)$ to $g(2R+k^R)$ are common, the sequence becomes periodic in the way described above, and thus $g(R)$ is the last rare value. However, the last rare value that was found by Achim Flammenkamp [5] is $g(48399022) = 259$ and he found a total of 1287 rare values. This would give $R = 48399022$ and $k = 1288$. This means we would have to calculate the Sprague-Grundy values up to $2R+k^R$ and those numbers are way to large to actually calculate.

The introduction of rare values also gives a faster way to compute new Sprague-Grundy values [1]. Where before, to calculate the Sprague-Grundy value for a pile of n stones, we had to calculate $\lfloor \frac{n}{2} \rfloor$ Nim-sums (besides calculating all Sprague-Grundy values for piles of less than n stones), we can now use the fact that $g(n)$ will probably be a common value. First we will calculate the first common value that is not excluded, with the following formula:

$$g_c(x) = \min \left\{ \begin{array}{ll} n \in \mathbb{N}_{\geq 0} : & n \text{ is common and} \\ & n \neq g(y) \oplus g(n-y), \forall y < n, \text{ with } g(y) \text{ is rare} \end{array} \right\}.$$

Now we just have to check that there is no rare value smaller than this common value, that is also not excluded. This can be achieved by computing the other Nim-sums $(g(y) \oplus g(n-y)$ with $1 \leq y \leq \lfloor \frac{n-1}{2} \rfloor)$ until either (1) we have excluded all rare values smaller than the common value, or (2) we have calculated all Nim-sums and there is still at least one rare value smaller than the common value left, that has not been excluded. In the first case, $g(n)$ is equal to our calculated common value and we made less calculations than in our original calculation. In the second case, $g(n)$ is equal to the smallest rare value not excluded and we have made the same amount of calculations as in our original calculation.

12. Discussion and conclusion

We started out working on the subject of Grundy's Game, hoping to solve the periodicity question. Sadly, we were not able to do that, although the chances of that were probably quite slim. In the end, we spent a lot of time on fully understanding the Sprague-Grundy function, Nim-addition and the Sprague-Grundy theorem. Hopefully, we have made the need for these more insightful in this thesis.

Besides that, we have been able to come up with a few new insights into the periodicity of Grundy's Game. We considered some adaptations of Grundy's Game, but we also ruled out certain periods, including a period of 3.

For further research, one could check out more adaptations of Grundy's Game and maybe pinpoint exactly why the Sprague-Grundy values for Grundy's Game are all over the place. However, the easiest, although incredibly time consuming, way to prove periodicity would probably be to actually calculate enough values, such that either Proposition 10.1 or Remark 11.8 holds.

References

[1] Berlekamp, E. R., Conway, J. H., & Guy, R. K. (2001). *Winning Ways for Your Mathematical Plays* (2nd ed., Vol. 1). Wellesley, MA: A K Peters.

[2] Bouton, C. L. (1901). Nim, A Game with a Complete Mathematical Theory. *The Annals of Mathematics, 3*(1/4), 35-39.

[3] Dudeney, H. E. (1908). *The Canterbury puzzles and other curious problems.* New York: E. P Dutton and Company.

[4] Ferguson, T. S. (2014). *Game Theory* (2nd ed.). Retrieved from https://www.math.ucla.edu /~tom/Game_Theory/Contents.html

[5] Flammenkamp, A. (2002, May 23). Sprague-Grundy Values of Grundy's Game. Retrieved from http://wwwhomes.uni-bielefeld.de/achim/grundy.html.

[6] Grundy, P. (1939). Mathematics and Games. *Eureka, 2*, 6-8.

[7] Guy, R. K. (1996). Unsolved Problems in Combinatorial Games. In R. Nowakowski (Ed.), *Games of No Chance* (pp. 475-491). Berkeley, CA: MSRI Publications.

[8] Guy, R. K., & Nowakowski, R. J. (2009). Unsolved Problems in Combinatorial Games. In M. H. Albert & R. J. Nowakowski (Eds.), *Games of No Chance 3* (pp. 465-489). Berkeley, CA: MSRI Publications.

[9] Guy, R. K., & Smith, C. A. (1956). The G-values of various games. *Mathematical Proceedings of the Cambridge Philosophical Society, 52*(03), 514-526.

[10] Van Luijk, R. (2015). *Linear Algebra I.* Retrieved from http://pub.math.leidenuniv.nl /~luijkrmvan/linalg/2015/LinearAlgebra1_2015.pdf

[11] Stevenhagen, P. (2016). *Algebra I.* Retrieved from http://websites.math.leidenuniv.nl /algebra/algebra1.pdf

Appendix A. C++-code to calculate Sprague-Grundy values for Kayles

In this program, we calculate the Sprague-Grundy value for Kayles with n pins by calculating $g(y) \oplus g(n-y-1)$ for all values $0 \leq y \leq \lfloor \frac{n-1}{2} \rfloor$, which we place in the first $\lfloor \frac{n-1}{2} \rfloor$ places of an array. Next we calculate $g(y) \oplus g(n-y-2)$ for all values $0 \leq y \leq \lfloor \frac{n-2}{2} \rfloor$, which we place in the same array, in the places $\lfloor \frac{n-1}{2} \rfloor + 1$ up to $n-2$. Next we sort these values (with the quicksort algorithm) and check which is the smallest value larger or equal to 0, that is not in the array. We do this by first checking whether the first value in the array is equal to 0, and then (if it is) checking what the first case is where the difference between two consecutive values is larger than 1.

```
#include <iostream>
#include <math.h>
#include <fstream>
using namespace std;

int SGsom (int a, int b){
    int x = 0;
    int y = 2;
    int w;
    while (a !=0 || b!=0){
          if (a % y == b % y){
                w=0;
          } /* end if */
          else{
               w=1;
          } /* end else */
          x = x + w*(y/2);
          a = (a - a % y);
          b = (b - b % y);
          y = 2*y;
    } /* end while */
    return x;
} /* end SGsom */

void quickSort(int arr[], int left, int right) {
      int i = left, j = right;
      int tmp;
      int pivot = arr[(left + right) / 2];
      while (i <= j) {
            while (arr[i] < pivot){
                  i++;
            } /* end while */
            while (arr[j] > pivot){
                  j--;
            } /* end while */
```

```
            if (i <= j) {
                  tmp = arr[i];
                  arr[i] = arr[j];
                  arr[j] = tmp;
                  i++;
                  j--;
            } /* end if */
      } /* end while */
      if (left < j){
            quickSort(arr, left, j);
      } /* end if */
      if (i < right){
            quickSort(arr, i, right);
      } /* end if */
} /* end quickSort */

int main (){
  int m;
  int k;
  int l;
  cout << "Up until where do you want to know the Sprague-Grundy values?" << endl;
  cin >> m;
  int n = m+1;
  int x[n];
  x[0]=0;
  x[1]=1;
  int y[n-1];
  for (int i=2; i <=n-1; i++){
      k = 0;
      l = 1;
      for (int j=0; j <= (i-1)/2; j++){
          y[j] = SGsom(x[j],x[i-j-1]);
      } /* end for */
      for (int j=0; j <= (i-2)/2; j++){
          y[j+(i+1)/2] = SGsom(x[j],x[i-j-2]);
      } /* end for */
      y[i]=i+1;
      quickSort(y,0,i-1);
      while (k != 1){
            if (y[0] != 0){
               k++;
               x[i]=0;
            } /* end if */
            else if (y[l]-y[l-1] <= 1){
                l++;
            } /* end if */
            else {
                 k++;
                 x[i]=y[l-1]+1;
            } /* end else */
      } /* end while */
  } /* end for */
```

```
   cout << "These are the Sprague-Grundy values up until " << m << ":" << endl;
   for (int i=0; i <= m; i++){
         cout << i << "---" << x[i] << endl;
   } /* end for */
   ofstream myfile;
   myfile.open ("Kayles.csv");
   myfile << "i;g(i)\n";
   for (int i=1; i <= m; i++){
         myfile << i << ";" << x[i] <<"\n";
   } /* end for */
   myfile.close();
   return 0;
 } /* end main */
```

For Grundy's Game and all its adaptations, the red part of the code above will be pretty much the same. Therefore we will show only the black part, which is the only part that significantly differs, for Grundy's Game and its adaptations.

Appendix B. C++-code to calculate Sprague-Grundy values for Grundy's Game

In this program, we calculate the Sprague-Grundy value for n stones by calculating $g(y) \oplus g(n-y)$ for all values $0 \leq y \leq \left\lfloor \frac{n-1}{2} \right\rfloor$, which we place in the first $\left\lfloor \frac{n-1}{2} \right\rfloor$ places of an array. Next we sort these values (using the quicksort algorithm) and check which is the smallest value larger or equal to 0 that is not in the array. We do this by first checking whether the first value in the array is equal to 0, and then checking what the first case is where the difference between two consecutive values is larger than 1.

Note that we only show the code that differs significantly from the code in Appendix A.

```
   x[0]=0;
   int y[(n-2)/2];
   for (int i=1; i <=n-1; i++){
       k = 0;
       l = 1;
       for (int j=1; j <= (i-1)/2; j++){
           y[j-1] = SGsom(x[j],x[i-j]);
       } /* end for */
       y[(i-1)/2]=(i-1)/2 + 1;
       quickSort(y,0,(i-3)/2);
```

Appendix C. C++-code to calculate Sprague-Grundy values for Grundy's Game, where a division into two equal piles is allowed.

Note that we only show the code that differs significantly from the code in Appendix A.

```
int y[(n-1)/2];
for (int i=0; i <=n-1; i++){
    k = 0;
    l = 1;
    for (int j=1; j <= i/2; j++){
        y[j-1] = SGsom(x[j],x[i-j]);
    } /* end for */
    y[i/2]=i/2 + 1;
    quickSort(y,0,(i-2)/2);
```

Appendix D. C++-code to calculate Sprague-Grundy values for Grundy's Game, with $g(2) = 1$.

Note that we only show the code that differs significantly from the code in Appendix A.

```
x[0]=0;
x[1]=0;
x[2]=1;
int y[(n-2)/2];
for (int i=3; i <=n-1; i++){
    k = 0;
    l = 1;
    for (int j=1; j <= (i-1)/2; j++){
        y[j-1] = SGsom(x[j],x[i-j]);
    } /* end for */
    y[(i-1)/2]=(i-1)/2 + 1;
    quickSort(y,0,(i-3)/2);
```

Appendix E. C++-code to calculate Sprague-Grundy values for Grundy's Game, with equal divisions, but without division of a pile of two stones.

Note that we only show the code that differs significantly from the code in Appendix A.

```
x[0]=0;
x[1]=0;
x[2]=0;
int y[(n-1)/2];
for (int i=3; i <=n-1; i++){
    k = 0;
    l = 1;
    for (int j=1; j <= i/2; j++){
        y[j-1] = SGsom(x[j],x[i-j]);
    } /* end for */
    y[i/2]=i/2 + 1;
    quickSort(y,0,(i-2)/2);
```

Appendix F. C++-code to calculate Sprague-Grundy values for Grundy's Game, where a division into two piles differing one is not allowed.

Note that we only show the code that differs significantly from the code in Appendix A.

```
x[0]=0;
x[1]=0;
int y[(n-3)/2];
for (int i=2; i <=n-1; i++){
    k = 0;
    l = 1;
    for (int j=1; j <= (i-2)/2; j++){
        y[j-1] = SGsom(x[j],x[i-j]);
    } /* end for */
    y[(i-2)/2]=(i-2)/2 + 1;
    quickSort(y,0,(i-4)/2);
```

Appendix G. C++-code to calculate Sprague-Grundy values for Grundy's Game, where a division into two piles differing one or two is not allowed.

Note that we only show the code that differs significantly from the code in Appendix A.

```
x[0]=0;
x[1]=0;
x[2]=0;
int y[(n-4)/2];
for (int i=3; i <=n-1; i++){
    k = 0;
    l = 1;
    for (int j=1; j <= (i-3)/2; j++){
        y[j-1] = SGsom(x[j],x[i-j]);
    } /* end for */
    y[(i-3)/2]=(i-3)/2 + 1;
    quickSort(y,0,(i-5)/2);
```

Appendix H. C++-code to calculate Sprague-Grundy values for Grundy's Game, with division into three piles, where all piles have to have a different amount of stones.

Note that we only show the code that differs significantly from the code in Appendix A.

```
int y[(n-4)/3*(n-2)/2];
int z;
for (int i=0; i <=n-1; i++){
    z = 0;
    k = 0;
    l = 1;
    for (int j=1; j <= (i-3)/3; j++){
        for (int h=j+1; h<= (i-j-1)/2; h++){
             y[z] = SGsom(SGsom(x[j],x[h]),x[i-j-h]);
             z++;
        } /* end for */
    } /* end for */
    y[z]=z + 1;
    quickSort(y,0,z-1);
```

Appendix I. C++-code to calculate Sprague-Grundy values for Grundy's Game, with division into three piles, where two piles may have the same amount of stones.

Note that we only show the code that differs significantly from the code in Appendix A.

```
int y[(n-2)/3*(n-1)/2];
int z;
for (int i=0; i <=n-1; i++){
    z = 0;
    k = 0;
    l = 1;
    for (int j=1; j <= (i-1)/3; j++){
        for (int h=j; h<= (i-j)/2; h++){
             y[z] = SGsom(SGsom(x[j],x[h]),x[i-j-h]);
             z++;
        } /* end for */
    } /* end for */
    y[z]=z + 1;
    quickSort(y,0,z-1);
```

Appendix J. C++-code to calculate Sprague-Grundy values for Grundy's Game, with division into three piles, where all piles may have the same amount of stones.

Note that we only show the code that differs significantly from the code in Appendix A.

```
int y[(n-1)/3*(n-1)/2];
int z;
for (int i=0; i <=n-1; i++){
    z = 0;
    k = 0;
    l = 1;
    for (int j=1; j <= i/3; j++){
        for (int h=j; h<= (i-j)/2; h++){
             y[z] = SGsom(SGsom(x[j],x[h]),x[i-j-h]);
             z++;
        } /* end for */
    } /* end for */
    y[z]=z + 1;
    quickSort(y,0,z-1);
```