Winning and losing positions of octal games

av

Joakim Uhlin

2017 - No 43

Winning and losing positions of octal games

Joakim Uhlin

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Oliver Krüger

2017

Abstract

In this thesis we will study a class of impartial games called octal games. One can think informally of such games as a pile of tokens from which each of 2 players alternatingly takes away some number of tokens with every move. Some moves also allow the leftover tokens in the pile to be split up into 2 new piles. The player who cannot take away any more tokens loses.

For any impartial game, in particular any octal game, we can attribute a Grundy-value. These values tell us, roughly speaking, who is winning the game. They are also much easier to compute than a ”brute-force” approach. The Grundy-values of an octal game can be conveniently recorded by a nim-sequence. It is conjectured that for all octal games, the corresponding nim-sequence is ultimately periodic. For several octal games, this has already been proven to be true. However, there still remains several octal games where it is still unknown whether or not the nim-sequence will be ultimately periodic.

Acknowledgements

First and foremost, I would like to thank my supervisor Oliver Krüger, who throughout my work on this thesis gave me excellent support and advice. Secondly, I also would like to thank Jörgen Backelin who is the second reader of this thesis and who contributed with many valuable comments and suggestions. Finally, a big thanks to both Patrik Karlström and Fredrika Agestam for helping with the C++ code.

Contents

1 Introduction

In this section we will present an introduction to *Combinatorial game theory* and in particular the theory for *impartial games*. Unlike many authors who introduce the subject by the game of *Nim*, we will present it from a more abstract point of view. The most important result of this section is the Sprague-Grundy function and how it can be used to compute sums of games which is the foundation of the thesis.

1.1 Combinatorial games

While it is not hard to define combinatorial games formally [1], we will simply give the informal description of how to think of them. Namely, a combinatorial game is a game that satisfies the following conditions:

1. There are two players, who alternate moves.
2. There is no chance involved.
3. The game will eventually end, even if the players do not alternate.
4. Both players have perfect information. This means that the current position, all of the reachable positions and all previous positions, are known to both players at all times.
5. The player who cannot make a move loses.

The third condition is sometimes dropped to allow games where an infinite number of moves is possible. These types of combinatorial games are called **loopy games**. Combinatorial games that are not loopy are called **loopfree games**. Chess is in theory an example of a loopfree game although in practice one of the two players will often claim a draw by the so called 50-move rule.

The last condition is called the **normal play convention**. It may be exchanged by the player who cannot make a move wins', in which case it is called the **misère play convention**. The condition can also be dropped altogether to allow games with draws. Tic-Tac-Toe is an example of a game that can end in a draw.

In this thesis we will only study combinatorial games that are loopfree and satisfies the normal play convention. This categorisation of combinatorial games disqualifies quite a few common games. As noted above, neither Chess nor Tic-Tac-Toe qualifies.

For more on combinatorial games we refer to *Winning Ways* [1] and *On numbers and games* [2].

1.2 Impartial games

We say that a combinatorial game is an *impartial game* if, at any given position, the legal moves of both of the two players are the same. This suggests the following formal definition.

Definition 1.1. (Axioms of impartial games)

i) An *impartial game* is a set of impartial games.

ii) There is no infinite chain of impartial games $G_1 \ni G_2 \ni ...$ ▲

Intuitively speaking, the first axiom tells us that impartial games are completely determined by its moves. It should be noted that this definition of impartial games is not circular as $\emptyset$ vacuously satisfies the first axiom. The second axiom tells us that the impartial game will eventually end, no matter what moves are made. The second axiom is usually called the **Descending Game Condition**.

A combinatorial game that is not impartial is called a **partisan game**. One example of such a game is Tic-Tac-Toe.

Remark 1.1. We will from now on only consider impartial games unless otherwise stated. For convenience, we will thus simply say 'game', when we mean 'impartial game'.

Definition 1.2. Let G be a game.

i) The elements of G are called the *options* of G. An element of G is called an option of G

ii) The *positions* of G are G and all positions of every option of G. An element in the set of positions is called a position of G.

iii) A position which has no option is called *terminal.* ▲

A game is in other words also a position. A position can on the other hand be thought of as a game where we start from the given position. We will, therefore, use the words ‘game’ and ‘position’ interchangeably.

The axioms of impartial games does allow a game to be an infinite set, i.e. a game where we at one point have an infinite amount of moves. Most games studied in Combinatorial Game Theory only has a finite number of moves in each given position, justifying the following definition.

Definition 1.3. Let G be a game. We say that G is a *short game* if it only has a finite number of positions. ▲

We start by presenting the classic theorem of *Conway Induction*[2]. This can be seen as an analogue of normal induction but for impartial games. It is also a very handy tool when working with impartial games, and is more practical to work with than the Descending Game Condition. In fact, it turns out that Conway Induction is equivalent to the Descending Game Condition, which we will also prove.

Theorem 1.1 (Conway Induction)**.** *Let P be a property that a game may have. If any game G has P whenever all options of G has P, then all games have P.*

Proof. By contradiction. Suppose that G_1 is a game that does not have P. Then there is an option of $G_2 \in G_1$ that does not have P since otherwise it would contradict the hypothesis. But then, using an inductive argument, we can create an infinite sequence of games

$$G_1 \ni G_2 \ni \dots$$

such that no games in this sequence has P. Note that none of these games can be terminal because a terminal game must vacuously have P. ■

Theorem 1.2. *Conway Induction implies the Descending Game Condition.*

Proof. By Conway Induction. Let P be the following property of a game G_1: There is no infinite chain of games

$$G_1 \ni G_2 \ni \dots$$

Indeed, suppose G is any game and all options of G has P, then so must G. Thus every game has P. ■

Note that unlike normal induction, Conway Induction does not need a base case to work. This is because a terminal game vacuously satisfies any such property P since it has no option. It is worth mentioning that Conway Induction can actually be generalized to partisan games [2], with an analogue proof.

Since the Descending Game Condition and Conway Induction are equivalent, we could pose the question: Why define impartial games using the Descending Game Condition and not the Conway Induction? There are two main reasons for this. The first one is due to convention, as e.g. Conway[2] defined games in this way. The second one is due to the fact that the Descending Game condition is arguably a more natural and intuitive property than Conway Induction.

The above theorems are in fact special cases of a more general theory about **well-funded posets**. Consider the set $\{P\}$ of positions of a game G together with the partial order $\leq$, where $P_1 \leq P_2$ if P_1 is a position of P_2. Then $(\{P\}, \leq)$ is a poset. Furtheremore, it is well-funded because of the Descending Game Condition. We will not delve more into the theory of posets in this thesis but the poset structure of impartial games can still be useful as an alternative characterization.

An important question about combinatorial games in general is which one of the two players has a winning strategy. For impartial games, we can classify every position in the following way.

Definition 1.4. Let G be a position of a game.

i) G is in $\mathcal{N}$ if there is an option of G which is in $\mathcal{P}$.

ii) G is in $\mathcal{P}$ if every option of G is in $\mathcal{N}$. ▲

We say that a game that lies in $\mathcal{N}$ is a *next player* game. This means that the player to move (the next player) has a winning strategy. Likewise, we call a game that lies in $\mathcal{P}$ a *previous player* game. This means that the player that has just made a move (the *previous player*) has a winning strategy.

Example 1.1. Some simple examples of impartial games follows below.

i) The terminal game $\emptyset$ lies in $\mathcal{P}$ and not in $\mathcal{N}$.

ii) The game $\{\emptyset\}$ lies in $\mathcal{N}$.

iii) The game $\{\{\emptyset\}, \emptyset\}$ lies in $\mathcal{N}$.

★

Before we proceed, we will show that $\mathcal{P}$ and $\mathcal{N}$ are disjoint, i.e. no player can be both winning and losing at the same time. Furthermore, we want to prove that $\mathcal{P}$ and $\mathcal{N}$ are enough to categorize all games. In other words, there is no third alternative like draws.

Theorem 1.3. *Let G be a game, then G is in exactly one of $\mathcal{P}$ and $\mathcal{N}$.*

Proof. By Conway Induction. Let P be the property of a game G: The game G lies in exactly one of $\mathcal{P}$ and $\mathcal{N}$.

Let G be any game. If all options of G satisfies P, then we are in exactly one out of the two below cases. We show that the inductive step holds in both cases.

Case 1: Suppose that all options of G is in $\mathcal{N}$ and no option of G is in $\mathcal{P}$. Then, by definition, G is in $\mathcal{P}$ and not in $\mathcal{N}$.
Case 2: Suppose that there is an option of G that is not in $\mathcal{N}$ and there is an option of G that is in $\mathcal{P}$. Then, by definition, G is in $\mathcal{N}$ and not in $\mathcal{P}$. ■

Note that certain player having a winning strategy does not necessarily mean that they will win no matter what move they make. Rather that there exist moves that will make them win no matter what the other player does. In many examples of combinatorial games, one player will have several options where only some of them will surely result in a win.

1.3 The Sprague-Grundy function

In this section we will define the Sprague-Grundy function, which is a function defined on short games. This function turns out to be a very powerful tool in classifying such games. Before that, we need to define a new function called *mex*.

Definition 1.5. The minimal excluded number of a set $S \subset \mathbf{N}$, denoted *mex*, is the function

$$\operatorname{mex}(S) = \min\{n \in \mathbf{N} \mid n \notin S\}.$$

▲

More informally, mex is the smallest natural number that is not in the given set.

Example 1.2. $\operatorname{mex}(\{0,\ 1,\ 2,\ 5,\ 6,\ 83\}) = 3$ and $\operatorname{mex}(\emptyset)=0$. ★

Definition 1.6. Let G be short game. The *Sprague-Grundy function* $g(G)$, is defined recursively as

$$g(G) = \begin{cases} 0 & \text{if } G \text{ is terminal} \\ \operatorname{mex}\{g(G') \mid G' \in G\} & \text{otherwise} \end{cases}$$

We say that the *Grundy-value* of G is $g(G)$. The set $E = \{g(G')|G' \in G\}$ is called the *excludants of* G and $e \in E$ is called an *excludant of* G. ▲

We need the hypothesis that G is short because mex($\mathbf{N}$) is not defined. It is possible to generalize mex so that mex($\mathbf{N}$) $= \infty$, as can be seen in [3]. This is not needed in this thesis so we settle with this simpler definition instead.

One important consequence of this definition is that no position can have the same Grundy-value as any of its options. Another consequence is that every position with a positive Grundy-value will have at least one option which is a has a Grundy-value equal to 0 and every position that is not terminal with Grundy-value equal to 0 will only have options with positive Grundy-value. These observations are the main tools when working with the Sprague-Grundy function.

Theorem 1.4. *Let G be a short game. Then G is in $\mathcal{N}$ if and only if $g(G) > 0$.*

Proof. By Conway Induction. Let P be the following property of a game G: The game G lies in $\mathcal{N}$ if and only if $g(G) > 0$.

Let G be any game. If all options of G satisfies P, then we are in exactly one out of the two below cases. We show that the inductive step holds in both cases.

Case 1: Suppose that for all options $G' \in G$, we have $g(G') > 0$ and G' is in $\mathcal{N}$. Then, by definition, $g(G) = 0$ and G lies in $\mathcal{P}$.
Case 2: Suppose that that there is an option $G' \in G$ such that $g(G') = 0$ and G' is in $\mathcal{P}$. Then, by definition, $g(G) > 0$ and G is in $\mathcal{N}$. ■

1.4 Sum of games

Definition 1.7. Let G and H be games. The *sum* of the games, denoted $G + H$ is defined as

$$G + H = \{G + H' \mid H' \in H\} \cup \{G' + H \mid G' \in G\}.$$

▲

We may think about the sum of G and H as a new game in which each player can either make a move in G or make a move in H. This game ends when there are no moves left in neither G nor H.

It follows by the definition that the sum of two games is commutative and associative. Therefore, it makes sense to talk about sums of more than two games. We will use the notation

$$\sum_{i=1}^{n} G_i = G_1 + G_2 + \cdots + G_n.$$

Given some finite number of games $G_1, \ldots, G_n$, we want to determine the winner of $\sum_{i=1}^{n} G_i$. The following example shows that the classes $\mathcal{N}$ and $\mathcal{P}$ and not strong enough tools to solve this problem.

Example 1.3. Consider the following statements. We will not prove any of them as they are trivial consequences of the next theorem.

i) If G lies in $\mathcal{P}$ and H lies in $\mathcal{P}$ then $G + H$ lies in $\mathcal{P}$.

ii) If G lies in $\mathcal{P}$ and H lies in $\mathcal{N}$ then $G + H$ lies in $\mathcal{N}$.

iii) If G lies in $\mathcal{N}$ and H lies in $\mathcal{N}$ then $G + H$ lies in either $\mathcal{P}$ or $\mathcal{N}$. For example, consider the games

$$G = \{\emptyset\}, \; H = \{\{\emptyset\}, \emptyset\}.$$

Indeed, G and H lies in $\mathcal{N}$ but $G + G$ lies in $\mathcal{P}$ whereas $G + H$ lies in $\mathcal{N}$.

★

We will now show that the Grundy-function is the tool we need in order to solve the problem of classifying a sum of games into the classes $\mathcal{P}$ and $\mathcal{N}$. However, before we can state the next theorem, we need to define a new binary operator, namely the *nim-sum*.

Definition 1.8. Let $x, y \in \mathbf{N}$ with binary representation $x = (x_n \ldots x_0)_2$ and $y = (y_n \ldots y_0)_2$ for some $n \in \mathbf{N}$. The *nim-sum* of x and y, denoted $x \oplus y$, is defined as

$$x \oplus y = (x_n \ldots x_0)_2 \oplus (y_n \ldots y_0)_2 = (z_n \ldots z_0)_2 = z$$

where $z_i = x_i + y_i \pmod 2$. ▲

The nim-sum is also known as the bitwise XOR of the binary representation of two numbers.

Example 1.4. $14 \oplus 27 = (01110)_2 \oplus (11011)_2 = (10101)_2 = 21.$ ★

The nim-sum is clearly both associative and commutative. It is also clear that $a \oplus 0 = a$ for all $a \in \mathbf{N}$, so 0 is an **identity element**. Another interesting property of the nim-sum is that for any $a, b \in \mathbf{N}$, we have that $a \oplus b = 0$ if and only if $a = b$. This shows that the nim-sum is a **nilpotent operator** and that every element is its own **inverse**. Lastly, note that $a \oplus b \in \mathbf{N}$ for any $a, b \in \mathbf{N}$ so the nim-sum is a **closed operation** in $\mathbf{N}$. This actually shows that that the pair $(\mathbf{N}, \oplus)$ is a **abelian group**. We will not delve deeper into the theory of groups here, but rather refer to [8] for more information.

The last theorem of the section will truly display the power of the Grundy-function. It opens the possibilities for breaking many types of games (including *octal games*, which we will see later) by computational methods. In the proof of the theorem, we will use the notation $[n]_k$ to denote the k:th bit of n in its binary representation (the rightmost bit is the 0:th bit). As an example, we get $[11]_0 = [(1011)_2]_0 = 1$ and $[46]_4 = [(101110)_2]_4 = 0$. We will use that one can write $[a \oplus b]_k = [a]_k \oplus [b]_k$ for any $a, b \in \mathbf{N}$.

Theorem 1.5. *Let G and H be short games. Then $g(G + H) = g(G) \oplus g(H)$.*

Proof. By Conway Induction. Let P be the following property for a game $G + H$:

$$g(G + H) = g(G) \oplus g(H).$$

Note that for all games $G = G + \emptyset$, therefore it actually makes sense to assume that an arbitrary game is of the form $G + H$.

Suppose $G + H$ is any game such that all options of $G + H$ has P. Any option of $G + H$ is either on the form $G' + H$ with $G' \in G$ or on the form $G + H'$ with $H' \in H$. Hence, the excludants of $G + H$ is given by the set

$$E = \{g(G' + H) | G' \in G\} \cup \{g(G + H') | H' \in H\} =$$

$$= \{g(G') \oplus g(H) | G' \in G\} \cup \{g(G) \oplus g(H') | H' \in H\}.$$

Our aim is to show that $\text{mex}(E) = g(G) \oplus g(H)$ by showing that E does not contain $g(G) \oplus g(H)$ but contains every smaller, non-negative integer.

First, note that for any game on the form $G' + H \in G + H$, we have

$$g(G') \neq g(G) \iff g(G') \oplus g(H) \neq g(G) \oplus g(H).$$

An analogue argument can be made about games on the form $G + H'$. This shows that E does not contain $g(G) \oplus g(H)$.

Next, let $n \in \mathbf{N}$ be some number such that $n < g(G + H)$. Thus we can define $k \in \mathbf{N}$ such that k is the biggest number $[g(G + H)]_k = 1$ and $[n]_k = 0$. This means that exactly one of $[g(G)]_k = 1$ or $[g(H)]_k = 1$ holds. WLOG, suppose that $[g(G)]_k = 1$ and $[g(H)]_k = 0$. Then certainly $[g(H) \oplus n]_k = 1$ and furthermore, for any $k' > k$, we have

$$[g(G) \oplus g(H)]_{k'} = [n]_{k'} \iff$$

$$[g(G) \oplus g(H)]_{k'} \oplus [g(H)]_{k'} = [n]_{k'} \oplus [g(H)]_{k'} \iff$$

$$[g(G) \oplus g(H) \oplus g(H)]_{k'} = [g(G) \oplus 0]_{k'} = [g(G)]_{k'} = [n \oplus g(H)]_{k'}.$$

This shows that $g(G) > n \oplus g(H)$, so there is an option $G^* \in G$ such that $g(G^*) = n \oplus g(H)$. It follows that

$$g(G^* + H) = g(G^*) \oplus g(H) = n \oplus g(H) \oplus g(H) = n \oplus 0 = n \in S.$$

Thus E contains every non-negative number smaller than $g(G+H)$. This completes the proof.

■

2 Take-and-Break games

We shall now study a type of impartial games called *Take-and-Break games.* The structure of these games can intuitively be described as a pile of tokens. Two players are alternating moves, and with each move one of them will take away a certain amount of tokens from the pile. In most Take-and-break games the player will have several choices of tokens to remove and sometimes the player will be able to split up the pile into two piles. The first player who cannot make a move loses.

2.1 Code-digits

As a motivating example, we start of this section by studying the following classic[1] game.

Example 2.1. Consider the game of 21 tokens and the legal moves are to take away one, two or three tokens. This game is in $\mathcal{N}$ since the first player can take away one token the first move and then make sure that 4 tokens are removed every two moves. ★

Definition 2.1. Let $\Gamma = d_0 d_1 d_2 \ldots$ be a word (infinite or finite) in the alphabet $\mathbf{N}$. The string $\Gamma = d_0 \cdot d_1 d_2 \ldots$ is called *code-digits*. If there are only a finite number of non-zero characters in Γ and d_k is the last non-zero character, then we simply write $\Gamma = d_0 \cdot d_1 d_2 \ldots d_k$. ▲

We will now give an informal combinatorial definition of *Take-and-Break games* and then a more formal, set theoretic one. We will say that an integer $n = (n_m \ldots n_1 n_0)_2$ **contains** a power of two 2^k if $n_k = 1$. For example, $9 = (1001)_2$ contains 8 but $22 = (10110)_2$ does not.

The rules of Take-and-Break games are characterized by some string $\Gamma = d_0 \cdot d_1 d_2 \ldots$ of code-digits where d_0 does not contain 1 or 2. The game itself can be thought of as a pile of tokens where it is a legal move to remove i tokens and split up the remaining tokens into k non-empty piles if d_i contains 2^k. Play continues until there is no pile with a legal move left.

Naturally, we want these games to obey the axioms of impartial games and as such, we must prohibit "do-nothing"-moves. This is why we must define d_0 to not contain 1 or 2.

Definition 2.2 (Take-and-Break games)**.** Suppose that $\Gamma = d_0 \cdot d_1 d_2 \ldots$ are code-digits such that d_0 does not contain 1 or 2. A Take-and-Break game G_n under Γ is set of sets that is defined recursively as

- The set $G_0 = \emptyset \in G_n$ if d_n contains 1.
- The set $G_k \in G_n$, for $k > 0$, if d_{n-k} contains 2.
- The set $G_{k_1} + G_{k_1} \in G_n$, for $k_1, k_2 > 0$, if $d_{n-(k_1+k_2)}$ contains 4.

$$\vdots$$

- In general, the set $\sum_{i=1}^{j} G_{k_i} \in G_n$, for $k_i > 0$, if $d_{n-\sum_{i=1}^{j} k_i}$ contains 2^j.

If the word is of finite length, we say that Γ is an *finite code-digit string*. Otherwise, it is an *infinite code-digit string*. ▲

Under these conditions, it is clear that Take-and-Break games are short impartial games so it makes sense to talk about the Grundy-value of some pile G_n under the code digits of Γ.

We will often fix some Take-and-Break game $\Gamma = d_0 \cdot d_1 d_2 \dots$ and simply talk about G_n. In this case, it is tacitly understood that we refer to G_n under the code-digits of Γ.

Example 2.2. We end the section with a few examples of different Take-and-Break games.

i) The game $0\cdot 333$ is the game where all moves consist of removing either one, two or three tokens, i.e. the game displayed in Example 2.1. Since we may never split up a pile into two, this is an example of a so called *subtraction game* [1][5]. That is, a game where all the octal digits are either 0 or 3.

ii) The game $0\cdot 176$ is the game where the legal moves are the following

- If the pile consists of one token, take away that token.
- If the pile consists of two or more tokens, take away two tokens. If there are two tokens or more tokens remaining after that, optionally one can split up the pile into two non-empty piles.
- If the pile consists of four or more tokens, take away three tokens. If there are two tokens or more tokens remaining after that, optionally one can split up the pile into two non-empty piles.

iii) The game $0\cdot 01$ is a very boring game. The only legal move is to take away 2 tokens if there are two tokens left. Hence, the game is $\mathcal{N}$ if and only if there are only two tokens.

iv) The game $0\cdot 8$ is the game where the legal moves are those where one token is removed and the remaining tokens are split up into three non-empty piles.

v) The game $0\cdot 333\dots$, more commonly knows as *Nim*, is one the first combinatorial games to be studied. Each move, we may take away any number of tokens from any pile.

$\bigstar$

2.2 Nim-sequences

A convenient way to encode information and study a Take-and-Break is to introduce the *nim-sequence* of the game.

Definition 2.3. Suppose that $\Gamma = d_0 \cdot d_1 d_2 \dots$ is a (finite or infinite) Take-and-break game. We say that sequence

$$g(G_0),\ g(G_1),\ g(G_2), \cdots$$

is the *nim-sequence* of Γ. ▲

Knowing all elements of the nim-sequence means that we have, for all intents and purposes, 'solved' the game. This is because by Theorem 1.3 we know exactly what positions are in $\mathcal{P}$ and what positions are in $\mathcal{N}$.

For a general Take-and-Break game, computing the whole nim-sequnce is very hard. However, for some easier games, it is possible to compute them using combinatorial arguments. We show two examples of such games below.

Example 2.3. We compute the nim-sequence of the game $0 \cdot 333$ that was discussed in the previous examples. We claim that $g(G_n) = n \pmod 4$. This is easily verified for $n = 0, 1, 2$. For any game G_n, $n \geq 3$, we can play to either G_{n-1}, G_{n-2} or G_{n-3}. By the inductive hypothesis, this means that

$$g(G_n) = \text{mex}\,\{G_{n-1}, G_{n-2}, G_{n-3}\} = \begin{cases} \text{mex}\,\{3,2,1\} = 0 & \text{if } n = 0 \pmod 4 \\ \text{mex}\,\{3,2,0\} = 1 & \text{if } n = 1 \pmod 4 \\ \text{mex}\,\{3,1,0\} = 2 & \text{if } n = 2 \pmod 4 \\ \text{mex}\,\{2,1,0\} = 3 & \text{if } n = 3 \pmod 4 \end{cases}.$$

This proves our claim. Therefore, the nim-sequence of the game is

$$012301230123\dots$$

★

Example 2.4. We compute the nim-sequence of the game Nim that was discussed in Example 2.2. We claim that $g(G_n) = n$. This certainly holds for $n = 0$. On the other hand, for G_n with $n > 0$, we can play to $G_0, G_1, \dots, G_{n-1}$. By the inductive hypothesis, this means that

$$g(G_n) = \text{mex}\,\{G_0, G_1, \dots, G_{n-1}\} = \text{mex}\,\{0, 1, \dots, n-1\} = n$$

This proves our claim. Therefore the nim-sequence of the game is

$$0123456789\dots$$

★

For nim-sequences that are repeating i.e. *periodic*, we will sometimes use the convenient notation

$$\dot{n}_1 n_2 \dots \dot{n}_p = n_1 n_2 \dots n_p n_1, \dots n_p n_1 n_2 \dots$$

So the nim-sequence in Example 2.3 could be written as $\dot{0}12\dot{3}$. We shall in later sections give a more detailed analysis of such periodic sequences.

2.3 Octal games

Definition 2.4. Let $\Gamma = d_0 \cdot d_1 d_2 \dots$ be some Take-and-Break game (finite or infinite). If $d_i < 8$ for all i, we say that Γ is an octal game. A finite Take-and-Break game that is an octal game, is called a *finite octal game*. An octal game that is not a finite octal game is called *infinite octal game*. ▲

Informally we can think of octal games as the Take-and-Break games where no moves split up any pile into more than two piles. We note that all games presented in Example 2.2 were octal games except for $0 \cdot 8$.

One could ask why we should study octal games rather than more general theory about Take-and-Break games? This is simply because octal games behave much ”nicer” compared to general Take-and-Break games. One could just as well study games where some legal moves consist of splitting up a pile into three piles, these games are called *hexadecimal games*. Hexadecimals are in general much harder to solve and can behave very ”wild” in comparison to octal games as can be seen in [1][5]. For example, Theorem 2.3 in the next section does not hold if we change ”octal game” to ”hexidecimal game”.

To reduce the computational complexity, it is important to note that due to symmetry, we have

$$\begin{aligned} S &= \{g(G_a) \oplus g(G_b) \mid a, b \in \mathbf{N},\ a + b = n\} \\ &= \{g(G_a) \oplus g(G_b) \mid a, b \in \mathbf{N},\ a \le b,\ a + b = n\}. \end{aligned}$$

Therefore, we only need to compute half of the possible nim-sums in order to establish $\operatorname{mex}(S)$. We show an application of this in the next example.

Example 2.5. We will compute the Grundy-value of G_{10} in the octal game $0{\cdot}176$ that was discussed in Example 2.2. Suppose that we know, from previous computations, that the first 10 numbers in the corresponding nim-sequence is

$$0, 1, 1, 0, 2, 2, 3, 4, 4, 1.$$

Since G_{10} has only a few possible moves (recall Example 2.2) we can determine its Grundy value by some relatively simple computations. Here, we use the sum of games properties that was discussed in section 1.4 to compute the Grundy-values.

$$g(G_{10}) = \operatorname{mex}\left(\left\{\begin{array}{lr} g(G_8) = 4, & g(G_7 + G_1) = 5, \\ g(G_6 + G_2) = 2, & g(G_5 + G_3) = 2 \\ g(G_4 + G_4) = 0, & g(G_7) = 4 \\ g(G_6 + G_1) = 2, & g(G_5 + G_2) = 3, \\ g(G_4 + G_3) = 2 & \end{array}\right\}\right) = 1$$

Since 1 is indeed the smallest integer that does not lie in the above set. ★

Even though the Sprague-Grundy function is a very powerful tool when analyzing octal games, we can sometimes deduce some interesting results just by studying their combinatorial structure. One example of this is shown below.

Theorem 2.1. *Let $\Gamma = d_0.d_1d_2\dots$ be an octal game. If there is i even and a j odd such that d_i and d_j contains 4, then there are only finitely many n such that G_n is in $\mathcal{P}$.*

Proof. Its sufficient to show that there are no $n \geq \max(i, j) + 2$ such that G_n is in $\mathcal{P}$. To prove this, we divide into cases. If n is even, then $a = (n - i)/2$ is a positive integer so $G_a + G_a \in G_n$. This option has Grundy-value

$$g(G_a + G_a) = g(G_a) \oplus g(G_a) = 0$$

so it is in $\mathcal{P}$ by Theorem 1.4. Hence G_n lies in $\mathcal{N}$. If n is odd, then $b = (n - j)/2$ is a positive integer so $G_b + G_b \in G_n$ and a similar argument proves that G_n is in $\mathcal{N}$. This exhausts all cases so we are done. ■

2.4 Periodicity of nim-sequences

Definition 2.5. A sequence $a_0, a_1, a_2 \dots$ is said to be

i) *Ultimately periodic* with *period* $p > 1$ if there exists $n' \in \mathbf{N}$ such that $a_n = a_{n+p}$ for all $n \in \mathbf{N}$, $n \geq n'$ and there is no smaller p such that this holds. If $n' = 0$, we say that the sequence is *periodic.*

ii) *Arithmetic periodic* with *period* $p > 1$ and saltus $s > 0$ if there exists $n' \in \mathbf{N}$ such that $a_n + s = a_{n+p}$ for all $n \in \mathbf{N}$, $n \geq n'$.

The smallest possible such n' is called the *preperiod.* ▲

In Example 3, we showed that the game 0· 33 had a periodic nim-sequence. Below, we show a slightly less trivial example of a game whose nim-sequence is ultimately periodic.

Example 2.6. Once again, consider the game 0· 176 that was discussed previous in examples.

0	1	1	**0**	**2**	**2**	3	4	
4	1	1	6	**2**	**2**	3	4	
4	1	1	6	6	3	3	**2**	
4	1	1	6	6	3	3	4	
4	1	1	6	6	3	3	4	
4	1	1	6	6	3	3	4	
4	1	1	6	6	3	3	4	
4	1	1	6	6	3	3	4	
4	1	1	6	6	3	3	4	...

Note that, except for the values in bold, it appears that the above sequence is ultimately periodic with preperiod 24 and period 8. ★

Without any more theory, we cannot tell if this apparent ultimate periodicity is persistent throughout all values. The following theorem is a key ingredient as it shows that it we can confirm that a nim-sequence is ultimately periodic by just computing a finite number of Grundy-values.

Theorem 2.2 (Guy-Smith Periodicity Theorem)**.** *Let Γ be a finite octal game $\Gamma = d_0 \cdot d_1 d_2 \dots d_k$ (with d_k non-zero), let $n' \in \mathbf{N}$ and $p \in \mathbf{N}^+$. Suppose that for all $n \in \mathbf{N}$ satisfying $n' + p \leq n \leq 2n' + 2p + k$, the following equality holds:*

$$g(G_n) = g(G_{n-p}).$$

If there are no smaller n', p with this property, then $g(G_n) = g(G_{n-p})$ for all $n \geq n' + p$, i.e. G_n is ultimately periodic with period p and preperiod n'.

Proof. We will prove that the equality $g(G_n) = g(G_{n-p})$ holds for $n = 2n'+2p+k+1$. We do this by showing that any excludant of G_n is an excludant of G_{n-p} and vice versa. From this it follows that their Grundy-values must be the same.

First, suppose that we have an option on the form $G_a + G_b \in G_n$ (if an option is to one pile or to no pile at all, then we just let at least one of $a, b = 0$, so it makes sense to assume that an option is on this form). This means that

$$n = 2n' + 2p + k + 1 \leq a + b + k \iff$$

$$2n' + 2p + 1 \leq a + b.$$

So at least one of a and b is at least as big as $n' + p + 1$. WLOG, we may assume that a has this property. Thus, we can safely assume that $G_{a-p} + G_b \in G_{n-p}$ since G_{a-p} must be non-empty pile. Note that $a < n$ since if $a = n$, this would be a "do-nothing"-move. Hence, we get that

$$g(G_a + G_b) = g(G_a) \oplus g(G_b) = g(G_{a-p}) \oplus g(G_b) = g(G_{a-p} + G_b).$$

So any excludant of G_n is an excludant of G_{n-p}.

Next, suppose that we have an option on the form $G_c + G_d \in G_{n-p}$. This means that

$$n - p = 2n' + p + k + 1 \leq c + d + k \iff$$

$$2n' + p + 1 \leq c + d.$$

So at least one of c and d is at least as big as $n' + 1$. WLOG, assume that c has this property. Hence, we get that

$$g(G_c + G_d) = g(G_c) \oplus g(G_d) = g(G_{c+p}) \oplus g(G_d) = g(G_{c+p} + G_d).$$

This shows that every excludant of G_{n-p} is an excludant of G_n. This shows that $g(G_n) = g(G_{n-p})$, where $n = 2n'+2p+k$. Using induction and an analogue argument, we can now prove that the equality holds for all n. This completes the proof. ■

Example 2.7. In Example 2.6, we saw that nim-sequence of $0{\cdot}176$ appeared to be ultimately periodic with preperiod 24 and period 8. By Theorem 1, we need to only to compute the first

$$2 \cdot 24 + 2 \cdot 8 + 3 = 67$$

Grundy-values to confirm that the nim-sequence is ultimately periodic. In the above table, we computed 72, so we have indeed the confirmed that the nim-sequence is ultimately periodic. ★

Example 2.8. We can now give an alternative proof that the nim-sequence of 0· 333 is periodic with period 4. Using Theorem 1, we need only to compute

$$0 \cdot 2 + 4 \cdot 2 + 3 = 11$$

Grundy-values to confirm the periodicity of the nim-sequence. ★

Several questions arise when studying the nim-sequences of octal games in general. Games with very similar rules can have vastly different nim-sequences.

The game 0· 07 has a nim-sequence with preperiod 53 and period 34 [5]. On the other hand, the game 0· 007 has a nim-sequence which we do not yet know if it is ultimately periodic or not. This is despite that Achim Flammenkamp has computed the first 2^{28} Grundy-values of the nim-sequence [6]. It is fascinating that such a small change in the rules of octal games can cause such a big difference. With these motivating examples, we now state the main conjecture of Octal games, originally posed by Richard Guy.

Conjecture 2.1. *The nim-sequence of every finite octal game is ultimately periodic.*

One common, slightly vague, notion used among several authors is that of a **non-trivial finite octal game**, i.e. a game with a preperiod that is not very short (say less than 50). The finite octal games that have mostly been studied this far are those with 3 or less code-digits. There are 79 such non-trivial octal games, 14 of these have been solved this far.

It is worth mentioning that the above conjecture holds for the class of finite subtraction games (recall from Example 2.2). To be more precise, given some subtraction game $\Gamma = d_0.d_1d_2 \dots d_k$ with all $d_i \in \{0, 3\}$, the nim-sequence of Γ will be ultimately periodic. Still, subtraction games are not completely understood yet despite their simplicity. It is an open problem to, given the code-digits of an arbitrary finite subtraction game, determine the preperiod and period of the given subtraction. For more information, see e.g. [1][5].

While proving that the nim-sequence of every finite octal game is ultimately periodic seems to be hard, it can be shown that the nim-sequences can not increase very fast as shown by the below theorem.

Theorem 2.3. *If Γ is a finite octal game, then nim-sequence of Γ is not arithmetically periodic.*

Proof. See [1]. ■

It shall be noted that the condition that the octal game is finite is necessary for the above theorem to hold. As was shown in Example 4, Nim is an infinite octal game which is arithmetically periodic. On the other hand, the condition is not sufficient as the infinite game $\Gamma = 0 \cdot 111\ldots$ has the nim-sequence $0111\ldots$. As was discussed in the previous section, the above theorem does not hold if "octal game" is changed into "hexadecimal game". One counter-example is shown in [5]. There, Aaron Siegel proves that $0 \cdot 8$ is arithmetically periodic with nim-sequence

$$0000111222333444\ldots$$

2.5 Sparse spaces

When computing the nim-sequence of some finite octal game Γ, the naive method to compute G_n is to determine the set

$$Ewww = \{g(G_a) \oplus g(G_b) \mid G_a + G_b \in G_n\}$$

Even for quite small values of n, this requires massive amounts of computations. However, for some octal games, a much more convenient method exists.

Computations of the Grundy-values of $0 \cdot 77$ reveals that the game is periodic with period 12 and preperiod 71. The values in the period are $1, 2, 4, 7$ and 8. The values $0, 3, 5$ and 6 only appears a few times in the preperiod. The difference between these two sets of numbers are that the first ones has an odd number of ones in their binary representation (the **odious numbers**) and the other ones has an even number of ones in their binary representation (the **evil numbers**).

We note that the set of numbers that have an even number of ones is a set closed under the operation of the nim-sum. While we cannot make the exact same division of every octal game it is often times still possible to make a distinction of some subset of $\mathbf{N}$ that is closed under the nim-sum.

Definition 2.6. Let Γ be an octal game and $\mathcal{S} \subseteq \mathbf{N}$. We say that $\mathcal{S}$ is a *sparse space* if the two below conditions are satisfied.

i) The following inequality hold

$$\#\{n \in \mathbf{N} \mid g(G_n) \in \mathcal{S}\} < \infty.$$

ii) For all $r_1, r_2 \in \mathcal{S}$ and all $c_1, c_2 \in \mathbf{N} \setminus \mathcal{S} = \mathcal{C}$, the following algebraic properties holds:

$$r_1 \oplus r_2 \in \mathcal{S} \qquad c_1 \oplus r_1 \in \mathcal{C}$$

$$c_1 \oplus c_2 \in \mathcal{S} \qquad r_2 \oplus c_2 \in \mathcal{C}.$$

We say that $\mathcal{C}$ is the common coset of $\mathcal{S}$. The elements in $\mathcal{S}$ are called *rare* and elements in $\mathcal{C}$ are called *common*. ▲

An equivalent formulation to the first condition is that '$g(G_n) \in \mathcal{S}$ only for finitely many n' or 'there are only finitely many elements in the nim-sequence of G_n that is in $\mathcal{S}$'.

Remark 2.1. It is easily seen that for every sparse space, 0 will always be a rare value. Indeed, if $a \in \mathcal{S}$ then $0 \oplus a \in \mathcal{S}$. Likewise if $a \in \mathcal{C}$, then $0 \oplus a \in \mathcal{C}$.

It might not be immediately apparent how sparse spaces "look like". It follows, of course, from the definition that sparse spaces are proper subsets of $\mathbf{N}$. Our aim is to prove a classification theorem that yields a very useful alternative representation of sparse spaces.

Recall that we discussed the group-structure of $(\mathbf{N}, \oplus)$ in section 1.4. If $\mathcal{S} \subset \mathbf{N}$ is a sparse space, it is easily seen that $\mathcal{S}$ is a subgroup of $(\mathbf{N}, \oplus)$. Indeed, it contains the identity 0, it has inverses of every element and it is closed under $\oplus$. In fact, it is an **index-2 subgroup** (see [8] for more details). This is because for any $n \in \mathbf{N}$, we have that either $n \oplus \mathcal{S} = \mathcal{S}$ or $n \oplus \mathcal{S} = \mathcal{C}$.

We are now ready to state the structure theorem of sparse spaces. Denote the set of two-powers

$$2^{\mathbf{N}} = \left\{2^i | i \in \mathbf{N}\right\}.$$

Theorem 2.4 (Aaron Siegel)**.** *There is a 1-to-1 correspondence between index-2 subgroups of $(\mathbf{N}, \oplus)$ and proper subsets of $2^{\mathbf{N}}$, given by the map $\mathcal{S} \mapsto \mathcal{S} \cap 2^{\mathbf{N}}$.*

Proof. See [5]. ■

Theorem 2.3 gives a convenient way to represent a sparse space by the set $\mathcal{T}$ of powers of two of $\mathcal{S}$. Another equivalent but more compact way is to represent the sparse space by a binary string with a non-zero number of 0:s. For example, the sets $\mathcal{T}_1 = \{8, 2, 1\}$ and $\mathcal{T}_2 = \{32, 16, 8, 2\}$ naturally correspond to the two binary strings $b_1 = \ldots 1110100$ and $b_2 = \ldots 111000101$.

Definition 2.7. Let $x, y \in \mathbf{N}$ with binary representation $x = (x_n \dots x_1 x_0)_2$ and $y = (y_n \dots y_1 y_0)_2$. The *bitwise AND* of x and y, denoted $x \wedge y$, is defined as

$$x \wedge y = (x_n \dots x_1 x_0)_2 \wedge (y_n \dots y_1 y_0)_2 = (z_n \dots z_1 z_0)_2 = z$$

where

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

▲

Example 2.9. $27 \wedge 35 = (011011)_2 \wedge (100011)_2 = (000011)_2 = 3$ ★

One useful application of Theorem 2.4 is that we get an alternative characterization of rare values in terms of bitwise AND. Given a number $r = (r_n \dots r_0)_2 \in \mathbf{N}$ and a sparse space $\mathcal{S}$, we get the following equivalences:

$$\begin{aligned} r \text{ is rare } &\iff \# \left\{ 2^i | r_i = 1,\, 2^i \notin \mathcal{S} \right\} \text{even} \iff \\ &\iff \# \left\{ i | r_i \ \wedge \ b_i \right\} \text{ is even} \end{aligned}$$

where $b = b_n \dots b_1 b_0$ is the bit string corresponding to the set of powers of two $\mathcal{T} \subset \mathcal{S}$. This means that the theorem also gives a way to determine a potential sparse space for an octal game. Indeed, for some octal game Γ, take index-2 subgroup of $(\mathbf{N}, \oplus)$ such that $g(G_i) \in \mathcal{S}$ for as few i as possible.

Example 2.10. The set of evil numbers is a sparse space associated to the game 0· 77. The corresponding binary string is simply

$$\dots 111111$$

★

Example 2.11. We once again study the game 0· 176. If this game has a sparse space, it must be one where the values can be divided into rare and common in the following manner

rare	common
0	1
2	3
	4
	6

$$\ldots 1111101$$

★

Since we already knew that the nim-sequences of $0\cdot 77$ and $0\cdot 176$ were ultimately periodic, it was easy to confirm that they had a sparse space. For an octal game in general, one would have to know every element of its nim-sequence in order to confirm that it has a sparse space. However, often times an apparent sparse space will appear long before the nim-sequence is actually periodic.

The game $0\cdot 007$ is a good example of this. Even though this game is not yet shown to be periodic, one could presume that its sparse space is the one characterized by the bit-string

$$\ldots 11110111000$$

Computations by Flammenkamp[6] shows that while 2^{28} Grundy-values have been computed in $0\cdot 007$, only 22476 of those have been in this presumed sparse space. It is therefore reasonable to believe that this regularity will continue.

A good reason to believe that an apparent sparse space will remain to be an apparent sparse space is that for sufficiently large n, most options of G_n will be on the form $G_a + G_b$, where both G_a and G_b have common Grundy-values. Thus, most Grundy-values of these options will be

$$\text{common value} \oplus \text{common value} = \text{rare value.}$$

So mostly rare values will be excluded and therefore there is a high chance of the next Grundy-value also being common.

The last theorem of this section might describes the above phenomena in a more precise way. It tells us that games that appear to have a sparse space likely will be ultimately periodic.

Theorem 2.5. *Let Γ be a finite octal game. If Γ has a sparse space, then its nim-sequence is ultimately periodic.*

We start by proving the following Lemma.

Lemma 2.1. *Let $\Gamma = d_0.d_1d_2\dots d_k$ be a finite octal game. If Γ has a sparse space, then its nim-sequence is bounded.*

proof of Lemma 2.1. Since Γ has a sparse space, it only has a finite amount of rare values in its nim-sequence. So denote

$$\begin{aligned} n_0 &= \text{ The last index such that } g(G_{n_0}) \text{ is rare} \\ r_0 &= \text{ The \# of rare values in the nim-sequence} \\ a &= \text{ The \# of digits } d_i \text{ that contains 2} \\ b &= \text{ The \# of digits } d_i \text{ that contains 4.} \end{aligned}$$

It is enough to show that $g(G_n)$ is bounded for all $n \geq 2n_0 + k + 1$, so for the rest of the proof, we assume that n satisfies this inequality. We will show that there is a common value c_0 so that $g(G_n) \leq c_0$. We know that the Grundy value of G_n is common so it is determined only by the common excludants of G_n. Write any option of G_n as $G_a + G_b$. Since $a + b + k$ is at least as big as n, we can write

$$2n_0 + k + 1 \leq a + b + k \iff 2n_0 + 1 \leq a + b$$

so at most one of a, b is at least as small as n_0. Hence, at most one of a, b is rare. Assume WLOG that a has this property. Using the algebraic properties of sparse spaces we conclude that the set of common excludants can be written as

$$E = \{g(G_a) \oplus g(G_b) \text{ common} : G_a + G_b \in G_n\}$$

$$= \{g(G_a) \oplus g(G_b) : G_a + G_b \in G_n \text{ where } g(G_a) \text{ rare, } g(G_b) \text{ common}\}.$$

For each digit d_i containing 2, we allow at most 1 move to a position whose Grundy-value is in E. Likewise, for each digit d_i containing 4, we allow at most r_0 moves to positions whose Grundy-values are in E. Hence, the options $G_a + G_b \in G_n$ can admit at most $a + br_0$ different common values. We claim that

$$c_0 = \text{ The } (a + br_0 + 1)\text{:th common value}$$

is the bound. To see why, note that if one of the first $a + br_0$ common values is not excluded, then the Grundy-value of G_n is certainly smaller than c_0. On the other hand, if all first $a + br_0$ common values are excluded then $g(G_n) = c_0$. This proves the claim, so we are done.

■

proof of Theorem 2.4. By Lemma 2.1, there is an integer N so that $N \geq g(G_n)$ for all n. There are only a finite number of $n_0 + k$-tuples with non-negative integers smaller than N so there exist $m_1, m_2 \in \mathbf{N}$, $m_1 < m_2$ such that

$$g(G_{m_1+i}) = g(G_{m_2+i}) \tag{$*$}$$

for all i satisfying $0 \leq i < n_0 + k$. There is an infinite amount of m_1, m_2 with this property so we may further assume that $n_0 \leq m_1, m_2$. We claim that this implies that $(*)$ holds for all $i \geq n_0 + k$. We prove this using an inductive argument. For the base case, note that $G_{m_1+n_0+k}$ and $G_{m_2+n_0+k}$ must both have common Grundy-values. Hence, it is enough to show that they have the same common excludants.

Using the algebraic properties of the sparse space, we deduce that any option $G_a + G_b \in G_{m_1+n_0+k}$ with common Grundy-value, exactly one of a, b needs to be rare. WLOG assume that a is rare so $a \leq n_0$ and thus $m_1 \leq b \leq m_1 + n_0 + k$. Rewriting the previous inequality, we get that $0 \leq b - m_1 \leq n_0 + k$. Using this, we can write

$$g(G_a) \oplus g(G_b) = g(G_a) \oplus g(G_{m_2+(b-m_1)})$$

which is an excludant of $G_{m_2+n_0+k}$. Similarly, we can show that every common excludant of $G_{m_2+n_0+k}$ is an excludant of $G_{m_1+n_0+k}$. This proves the base step. The inductive step is done in an analogue fashion. Thus, the nim-sequence of Γ is ultimately periodic with preperiod m_1 and period $m_2 - m_1$. ■

A more careful analysis of the proof of Lemma 2.1 and Theorem 2.4 gives an upper bound to how long it takes before the nim-sequence of a given finite octal game Γ with a sparse space. This bound is, however, rather enormous. Indeed, if x, y, k, r_0 and n_0 are chosen the same way as in the proof of Lemma 2.4, we get that the sum of the period and the preperiod is bounded by [6]

$$(x + r_0 y + 1)^{r_0+k}.$$

Example 2.12. Let $\Gamma = 0.56$. This octal game has a sparse space characterized by the bit string ...11111011011. It has 46 rare value and its last rare value is at index 1795. Its actual preperiod is 326640 and its period 144. However, the upper bound for the sum of its preperiod and period given by Theorem 2.4 is

$$(1 + 2 \cdot 46 + 1)^{(1795+2)} = 48^{1797} \approx 1.55 \cdot 10^{3021}.$$

★

The above example shows that upper bound is simply too rough to have any practical impact. At the moment, there is no known method for making any significant improvement of the given upper bound. However, none of finite octal games with a sparse space that has been shown to be ultimately periodic so far, seems to come close to the upper bound (compare e.g. with the above example). It is therefore possible that a more detailed analysis could improve the upper bound by a lot.

For general octal games, a higher pile of tokens means more possible moves. It seems intuitive that more moves would "increase the chance" for the first player to have a winning strategy. The below corollary shows that this is the case for all octal games with a sparse space.

Corollary 2.1. *Let Γ be a finite octal game. If Γ has a sparse space, then there are only a finite number of positions G_n such that G_n lies in $\mathcal{P}$.*

Proof. Let $\mathcal{S}$ be the given sparse space. We know that 0 was always a rare value. So if there is a finite number of positions G_n such that $g(G_n) \in \mathcal{S}$, in particular there is only a finite number of positions G_n such that $g(G_n) = 0$ which is equivalent to there being only a finite number of position G_n in $\mathcal{P}$ by Theorem 1.3. ■

With this is mind, one could hope to solve the main conjecture by trying to determine the sparse space of every octal game. Unfortunately, this is not enough as it turns out that some octal games does not have a sparse space. An easy counter-example is the game $0{\cdot}33$ which had 0 in its period and thus cannot have a finite number of rare values, no matter what space $\mathcal{S}$ we choose.

3 Algorithms and Applications

In this section we discuss applications of the theory of octal games. We show how it can be used to determine ultimate periodicity of nim-sequences of finite octal games. We discuss three different algorithms for computing Grundy-values of finite octal games, namely:

i) The **naive algorithm**.

ii) The **sparse space algorithm**, abbreviated as the **Sps-algorithm**.

iii) The **cheating sparse space algorithm**, abbreviated as the **CSpS-algorithm**.

We also display an algorithm that determines whether, given some Grundy values of an octal game, we can deduce ultimate periodicity using the Guy-Smith Periodicity Theorem.

Some of the algorithms are fairly self-explanitory and are presented without much comment. Others are more complex and are explained more thoroughly. All algorithms are given in general pseudocode, without a specific programming language in mind.

3.1 Some basic Complexity theory

Complexity theory is the study of the asymptotic behavior of functions and algorithms. Intuitively, we can think of it as a rough measure of how many steps a particular algorithm needs to terminate relative to its input size. This makes it an important concept when studying combinatorial game theory in general as any game with only finitely many possible moves will have a brute force-solution. These algorithms are, however, very slow and so we seek to find faster algorithms.

We introduce the part of the theory needed to discuss octal games. For a more thorough explanation concerning Complexity theory we refer to [7].

Definition 3.1 (The Big-O notation)**.**
Let $f(n), g(n)$ be two non-negative functions. Suppose there is some integer n' and a positive constant c such that

$$f(n) \leq cg(n), \text{ for all } n' \leq n.$$

Then we say that $f(n)$ is of *order* $g(n)$. More compactly, we will write $f(n) = \mathcal{O}(g(n))$ to denote this.

▲

Example 3.1. Let $f(n) = n^3 + 17n^2 + 3n + 78$. Then, for $1 \leq n$, we write

$$\begin{aligned} f(n) =& n^3 + 17n^2 + 3n + 78 \leq \\ \leq& n^3 + 17n^3 + 3n^3 + 78n^3 = 99n^3. \end{aligned}$$

This shows that $f(n) = \mathcal{O}(n^3)$ ★

Typically, one rarely uses the methods in above example to show that some function $f(n)$ is of order $g(n)$. Rather, one usually uses shortcuts or good enough estimates. The following theorem displays two of the most important properties of the Big-O notation.

Theorem 3.1. *Let $f(n), g(n)$ be two non-negative real-valued functions such that $f(n) = \mathcal{O}(f'(n))$ and $g(n) = \mathcal{O}(g'(n))$. Let c be a positive constant. Then the following properties of the Big-O notation hold:*

i) Absorbing constants
$$\mathcal{O}(cf(n)) = \mathcal{O}(f(n)).$$

ii) Rule of sum
$$f(n) + g(n) = \mathcal{O}(f'(n) + g'(n)).$$

iii) Rule of product
$$f(n)g(n) = \mathcal{O}(f'(n)g(n)).$$

iv) Rule of transitivity. If $f(n) = \mathcal{O}(g(n))$ and $g(n) = \mathcal{O}(h(n))$ then.
$$f(n) = \mathcal{O}(h(n)).$$

v) Taking away smaller terms. If there is some n' such that $f(n) \leq g(n)$ for all $n' \leq n$, then
$$\mathcal{O}(f(n) + g(n)) = \mathcal{O}(g(n)).$$

Note that the above theorem gives a much faster proof that the function in Example 3.1, as we can simply use the absorptivity of constants and take away smaller terms.

The big-O notation is a key ingredient when analyzing how fast an algorithm works. Let $T(n)$ be an upper bound of the time (i.e. the number of steps) it takes for an algorithm to finish, given an input of size n. We say that the algorithm is a

- **constant time algorithm** if $T(n) = \mathcal{O}(1)$.
- **linear time algorithm** if $T(n) = \mathcal{O}(n)$.
- **quadratic time algorithm** if $T(n) = \mathcal{O}(n^2)$.

3.2 The naive algorithm

We start by examining the ”naive” algorithm in order to compute Grundy values, that is, an algorithm that does not use the theory of sparse spaces. For games with a sparse space, this algorithm will be inferior to the SpS-algorithm in terms of computational complexity. However, as we discussed in the last section, some octal games lack a sparse space (e.g. 0·33 and 0·104) thus making the naive algorithm necessary for a complete analysis of octal games.

The idea of the Naive algorithm is straightforward. We create a list of Grundy-values L and let $g(G_0) = 0$ be the first element of L. Then we compute $g(G_n)$ recursively using mex. We then let $g(G_n)$ be the n:th element of L. It is less straightforward to determine the time-complexity of the Naive algorithm, which is what we will do in this section.

First, we present an algorithm to compute the mex of a list L of integers. The idea of the algorithm is to create a help list H of the same length as L, where $H[i] = 1$ if i is an element of L and $H[i] = 0$ otherwise

Procedure 1 Minimal excluded number

Input: A list L of non-negative integers.
Output: A non-negative integer.

```
function MEX($T$)
    $m \leftarrow \max(L)$
    $l \leftarrow \text{length}(L)$
    if $l = 0$ then
        return 0
    end if
    $H \leftarrow \text{zeros}(m+1)$                    ▷ Creates a list of $m+1$ zeros

    for all $i$ from 0 to $l-1$ do
        $H[L[i]] \leftarrow 1$
    end for

    for all $i$ from 0 to $m$ do
        if $H[i] = 0$ then
            return $i$
        end if
    end for
    return $m+1$
end function
```

Lemma 3.1. *Procedure* 1 *returns the mex of a list of non-negative integers. Fur-*

thermore, it terminates in linear time.

Proof. The correctness of the algorithm is trivial. To show that it is a linear time algorithm, note that all steps are either takes constant time or linear time, thus in total the algorithm must take linear time to terminate. ■

Procedure 2 The naive algorithm

Input: An integer $n \in N$.

Output: A vector with the first $n + 1$ Grundy-values of the nim-sequence of the octal game Γ.

```
function NAIVE($n$)
    GV-vec← [0]

    if n = 0 then
        return GV-vec
    end if
    for all $i$ from 1 to $n$ do

        $K \leftarrow [\,]$                                  ▷ $K$ is an empty list.
        for all $G_a + G_b \in G_n$ do
            $K \leftarrow K + [g(G_a) \oplus g(G_b)]$
        end for
        $L \leftarrow \text{mex}(K)$
    end for
    return GV-vec
end function
```

We need the following Lemma in order to give a rough bound of the number of options in a finite octal game.

Lemma 3.2. *Let $\Gamma = d_0 \cdot d_1 d_2 \dots d_k$ be some finite octal game, then the function*

$$Op_\Gamma(n) = \#\{\textit{Options of } G_n\} \leq (k+1)n$$

Proof. WLOG, we may assume that

$$\Gamma = 4 \cdot \underbrace{77\dots7}_{k \text{ digits}}$$

since no other octal games with the same number of digits will have more options. Thus, if the statement holds for this choice of Γ, it will hold for any choice of Γ with $k+1$ digits. It is easily verifiable that the inequality holds for cases $n = 0, 1$, so suppose $2 \leq n$.

For any j such that $0 \leq j \leq k$, the number of ways to take away j tokens and either leave only one pile or split the pile into two non-empty piles is (up to symmetry)

$$\#\left\{G_0 + G_{n-j},\, G_1 + G_{n-j-1},\, \dots,\, G_{\lfloor (n-j)/2 \rfloor} + G_{n-j-\lfloor (n-j)/2 \rfloor}\right\} =$$

$$=\#\left\{0,\, 1,\, 2,\, \dots,\, \lfloor (n-j)/2 \rfloor - 1,\, \lfloor (n-j)/2 \rfloor\right\} \leq$$

$$\leq \#\left\{0,\, 1,\, 2,\, \dots,\, \lfloor n/2 \rfloor - 1,\, \lfloor n/2 \rfloor\right\} \leq \frac{n+2}{2}.$$

Summing over all possible values for j now yields

$$\mathrm{Op}_\Gamma(n) \leq \sum_{j=0}^{k} \frac{n+2}{2} = (k+1)\frac{n+2}{2} \leq (k+1)n$$

Note that the last equality follows from the fact that $2 \leq n$, it thus follows that $\mathrm{Op}_\Gamma(n) \leq (k+1)n$, which finishes the proof. ■

Combining the two previous Lemmas, we can conclude the following.

Theorem 3.2. *Suppose that $\Gamma = d_0 \cdot d_1 d_2 \dots d_k$ is some finite octal game. Then the naive algorithm terminates in quadratic time.*

Proof. Let $T(n)$ be the running time of the algorithm, i.e. the number of steps it takes to compute $n+1$ Grundy-values. We will prove that $T(n) = \mathcal{O}(n^2)$ by using Theorem 3.1

In each step of the for-loop in steps $6-12$, we create a list K of the Grundy values of the options. By Lemma 3.2, the number of options $\mathrm{Op}_\Gamma(n) \leq (k+1)n = \mathcal{O}(n)$. Then we compute the mex of all the $\mathrm{Op}_\Gamma(n)$ numbers of K, so it also takes $\mathcal{O}(n)$ steps. By the rule of sum, each step of the algorithm takes $\mathcal{O}(n+n) = \mathcal{O}(n)$ steps. The algorithms runs through a for-loop with n steps, so the total running time is

$$T(n) = (n+1)\mathcal{O}(n) = \mathcal{O}(n(n+1)) = \mathcal{O}(n^2+n) = \mathcal{O}(n^2).$$

■

3.3 Algorithms using sparse spaces

For octal games with a presumed sparse space $\mathcal{S}$, the SpS-algorithm can heavily speed up the computations of Grundy-values. The idea is to compute G_n recursively by first determining the set

$$S = \{g(G_a + G_b) : G_a + G_b \in G_n,\, g(G_a) \in \mathcal{S}\}\,.$$

Most of the Grundy-values in the above set will be common values because G_b most of the time will be a common value. Furthermore, no other options of G_n can have Grundy-values that are common. Hence, a very good candidate for the smallest excluded Grundy-value is $c = \min(\mathcal{C} \setminus S)$. To confirm that c is in fact $g(G_n)$, we need only to compute enough Grundy-values so that

$$\{\text{Computed Grundy-values}\} \supseteq \{0, 1, \ldots, c-1\} \cap \mathcal{S}.$$

We will refer to c as the **candidate value** and the rare values less than as c as **leftover rares**. An option is said to **cover the leftover rare** if its Grundy-value is equal to that rare value. In the event that some leftover rare r is not covered by any option of G_n, we get $g(G_n) = r$. However, this happens very seldom.

We are ready to display the SpS-algorithm. For simplicity sake, we will assume that for the given finite octal game $\Gamma = d_0{\cdot}\, d_1 d_2 \ldots d_k$, the list L in the input is sufficiently long. That is, it contains all values of the nim-sequence up to index $k+1$.

Procedure 3 The SpS-algorithm

Input: A list L of sufficiently many Grundy-values of some nim-sequence, a sparse space $\mathcal{S} \subset \mathbf{N}$, a non-negative integer n.

Output: A list L of $n + 1$ Grundy-values.

```
1: function CSpS($L, \mathcal{S}, n$)
2:     $l \leftarrow$ length($L$)

3:     for all $i$ from $l$ to $n$ do
4:         $H \leftarrow [\,]$                                   ▷ $H$ is an empty list.
5:         for all $G_a + G_b \in G_n$ such that $g(G_a) \in \mathcal{S}$ do
6:             $H \leftarrow H + [g(G_a) \oplus g(G_b)]$          ▷ Appends an element to $H$.
7:         end for

8:         $c \leftarrow$ mec($H$)          ▷ mec($H$) is the smallest common not in $H$.
9:         $K \leftarrow \{0, 1, \ldots, c-1\} \cap \mathcal{S}$     ▷ $K$ is the set of leftover rares.

10:        for all $G_a + G_b \in G_n$ such that $g(G_a), g(G_b) \in \mathcal{C}$ do
11:            $K \leftarrow K \setminus \{g(G_a) \oplus g(G_b)\}$
12:            if $K = \emptyset$ then               ▷ All leftover rares have been covered.
13:                $L \leftarrow L + [c]$
14:                break
15:            end if
16:        end for

17:        if $K \neq \emptyset$ then
18:            $L \leftarrow L + [\text{mer}(K)]$   ▷ mer($K$) is the smallest rare value not in $K$.
19:        end if

20:    end for

21:    return $L$
22: end function
```

The sparse space algorithm relies on the fact that we usually will not need very many computations cover all the leftover rares. One may wonder exactly how many computations that is needed in order to establish to confirm that the candidate value. Not much is known about this problem in general. The table below shows some solved, nontrivial octal games and gives some insight about the upper bound of computations needed to cover all the leftover rares.

Define the number i_n to be the smallest number such that

$$\{g(G_a) \oplus g(G_b) : G_a + G_b \in G_n \text{ and } a \leq i_n\} \text{ covers all leftover rares.}$$

The **depth** of an octal game with a (presumed) sparse space is $\max_n \{i_n\}$. The **average leftovers** indicates how many interations (on avarage) in the for-loop in steps 10-16 of the SpS-algorithm that are needed in order to cover all of the leftover rares. Thus it is a relevant statistic when trying to quantify how much faster the SpS-algorithm is than the Naive algorithm. It should be noted that the avarage leftovers can differ somewhat depending on the implementation of the algorithms used. For avarage leftovers, we used a new implementation of the algorithm.

Table 1: Solved, non-trivial octal games with a sparse space [6].

Game	rare	depth	average leftovers
0·45	11	37	15.8632...
0·156	15	243	27.1894
0·055	6	20	13.1154...
0·644	31	604	80.9368...
0·356	7	19	20.6758...
0·56	46	7405	93.2393...
0·16	53	21577	129.2356...
0·376	510	505866	8137.3728...
0·454	17	4858	–
0·054	38	16284	–
0·354	132	705	–

We see that the number of avarage leftovers is typically very small. With this in mind, we may heuristically argue that most/all games with a sparse space should have, on average, a constant number of computation needed to cover all the leftover rares. Furthermore, if the number of rare values dies off quickly, the number of computations needed to determine the candidate value is bounded by a constant. Thus, to compute n Grundy-values, we need n steps in a loop, where each step in the loop takes $\mathcal{O}(1)$-time. Thus, the SpS-algorithm should be an $\mathcal{O}(n)$-algorithm, making it a much more efficient algorithm than the naive algorithm.

Unfortunately, there seem to exist some finite octal games that are too wild to be analyzed using only computations. For example, Achim Flammenkamp computed 2^{25} Grundy-values of the game $0{\cdot}163$. It is believed that this game has a sparse space characterized by the bit string $\ldots 111110111000$. However, it has 2800497 rare values, making about 8% of the Grundy-values rare. This is a very high ratio of rare values to common values compared to the solved games. Take the game 0.354 for instance, which has a preperiod of 10061916 but only 132 rare values. Hence, the rare values are only about 0.00013% of the total number of the Grundy-values in the preperiod. It seems that the number of rare values is a good indicator of how "soon" a nim-sequence is getting periodic thus making it unlikely that $0{\cdot}163$ will be solved anytime soon using these methods.

3.4 A cheating algorithm

We start this section by examining a statistic of octal games with sparse spaces, namely the *last index*. This motivates the idea of the next algorithm.

Definition 3.2. Let Γ be some octal game. Suppose Γ has a sparse space $\mathcal{S}$. The *last index* of Γ is defined as

$$\max\{n \in \mathbf{N} \mid g(G_n) \in \mathcal{S}\}.$$

▲

Table 2: Solved, non-trivial octal games with a sparse space.

Game	Preperiod	Last index	Last index/Preperiod
$0{\cdot}45$	498	198	0.3976...
$0{\cdot}156$	3479	357	0.1026...
$0{\cdot}055$	259	43	0.2704...
$0{\cdot}644$	3256	511	0.1569...
$0{\cdot}356$	7315	43	0.0059...
$0{\cdot}56$	326640	1795	0.0055...
$0{\cdot}16$	105351	13935	0.1323...
$0{\cdot}376$	2268248	1140540	0.5028...
$0{\cdot}454$	160949019	124	$10^{-7} \cdot 7.704\ldots$
$0{\cdot}054$	193235616	796	$10^{-6} \cdot 4.119\ldots$
$0{\cdot}354$	10061916	3227	$10^{-4} \cdot 3207\ldots$

The above table suggest that for most octal games with a sparse space, the rare values die out long before the nim-sequence actually becomes periodic. Furthermore, it seems that games with longer preperiod also have a lower "Last index/Preperiod"-ratio, with 0· 376 being the main exception.

If this suspected pattern continues, one could use previous computations of unsolved octal games with apparent sparse spaces in order to conjecture a last index. For example, Flammenkamp [6] has computed 2^{37} Grundy-values of the nim-sequence of 4· 045. The game has an apparent sparse space with only 34 rare values, the last one is at index 497. It is therefore likely that 497 is the very last rare value of the nim-sequence.

The idea of the CSpS-algorithm is to use this observation in order to speed up computations of Grundy-values $g(G_n)$ when n is big enough. Indeed, if there is a n' such that $n \geq n'$, the Grundy-values of G_n are all common, then we need only to compute the candidate value.

Procedure 4 The CSpS-algorithm

Input: A list L that contains all rare values of some nim-sequence, a sparse space $\mathcal{S} \subset \mathbf{N}$, a non-negative integer n.

Output: A list L of $n + 1$ Grundy-values.

```
1:  function CSPS($L, \mathcal{S}, n$)
2:      $l \leftarrow$ length($L$)

3:      for all $i$ from $l$ to $n$ do
4:          $H \leftarrow [\,]$                                  ▷ $H$ is an empty list.
5:          for all $G_a + G_b \in G_n$ such that $g(G_a) \in \mathcal{S}$ do
6:              $H \leftarrow H + [g(G_a) \oplus g(G_b)]$     ▷ Appends an element to $H$.
7:          end for
8:          $L \leftarrow L + [\text{mec}(H)]$                ▷ Appends an element to $L$.
9:      end for

10:     return $L$
11: end function
```

Theorem 3.3. *The CSpS-algorithm terminates in linear time.*

Proof. Let Γ be any octal game with sparse space $\mathcal{S}$. Then number of rare values in its nim-sequence is bounded by c_0. Hence, the number of options on the form $G_a + G_b$ is also bounded. Thus the for-loop in steps $5-7$ takes $\mathcal{O}(1)$ steps. Using Theorem 3.1, we get that the number of the steps in for-loop in steps $3-9$ is

$$T(n) = \mathcal{O}((n-l)\mathcal{O}(1)) = \mathcal{O}(\mathcal{O}(n-1l)) = \mathcal{O}(n-l) = \mathcal{O}(n).$$

■

The obvious downside of the CSpS-algorithm is that assumes that L contains all the rare values of the nim-sequence and will thus generate the wrong Grundy-values if this is not the case. Still, one can heuristically argue that games such as $4\cdot 045$ probably has no rare values after index 497.

This also raises the question of how big gaps there can be between rare values. Unfortunately, not much is known about this in general. However, relatively big gaps do occur in some nim-sequences. One example of such a game is the *James Bond Game* $0\cdot 007$. This game is not yet solved but Achim Flammenkamp has computed the first 2^{28} Grundy-values of its nim-sequnce, which shows that the game likely has a sparse space[6]. Aaron Siegel noted in [5] that the nim-sequence of $0\cdot 007$ has 1284 rare values in its nim-sequence. The 1271:st of these is at index 82860 whereas the 1272:nd is at index $47,461,861$.

3.5 Determining periodicity

The Guy-Smith periodicity theorem tells us that it is possible to confirm ultimate periodicity of a nim-sequence of some finite octal game with only a finite number of known values. In this section, we ask if this can be achieved in a reasonable time. To tackle this problem, our idea is to first determine the (potential) period of the nim-sequence. After that, determining the preperiod will be simple.

To give some motivation as to why a good algorithm is needed, let us first study a naive algorithm. Suppose L of Grundy-values of length $l+1$. **The naive algorithm** tests every possible period p' and run as long as possible by comparing $L[i]$ and $L[i-p']$ for $i = l, l-1, \ldots, p'$.

If $L[i] \neq L[i-p']$, we instead define $p' \leftarrow p'+1$ and try $i = l, l-1, \ldots, p'$ again. In practice, such an algorithm works well on a sequence that is "random enough". However, we will see that there are cases when the naive algorithm is ineffective.

Example 3.2. Consider an edge case sequence such as

$$\mathbf{3}212121\ldots 2121212\mathbf{3}212121\ldots 212121$$

where there is a very big number of 1:s and 2:s between the 3:s. If we would use a naive algorithm then every odd p' would fail on the first comparison. The first attempt would be $p' = 2$, the next $p' = 4$, then $p' = 6$ etc. etc. up to $p' = n/2$. If the list has length n, this algorithm would need roughly

$$\#\{\text{possible } p'\} \cdot \#\{\text{number of computations per } p'\} \approx \frac{n}{4} \cdot \frac{n}{4} = \frac{n^2}{16}$$

steps to complete. This means that the algorithm would need **quadratic time** to complete. ★

There are, however, better algorithms to determine periodicity of a sequence. For instance, the *Z-algorithm*.

Definition 3.3. Suppose that $a_0, a_1, \ldots, a_n$ is some finite sequence. Then we define the function $Z : \mathbf{N}^+ \to \mathbf{N}$ by

$$Z(i) = \text{The biggest } m \in \mathbf{N} \text{ so that } a_j = a_{j+i} \text{ for } j = 0, 1, \ldots, m.$$

Denote $Z_i = Z(i)$ and define the vector $Z = (Z_1, Z_2, \ldots, Z_n)$. ▲

Example 3.3. Let $L = 123412341211123$. Then the corresponding Z-vector is

$$Z = (0, 0, 0, 6, 0, 0, 0, 2, 0, 0, 1, 1, 3, 0, 0).$$

★

Indeed, the number Z_i tells us how much the start of the sequence "resembles" the sequence starting at index i. We can use this when working proving ultimate periodicity of nim-sequences. Suppose that $a_0, a_1, \ldots, a_n$ are the first $n+1$ elements of a sequence, denote the Z-**vector of the reversed sequence** $a_n, \ldots, a_1, a_0$ by $\hat{Z} = (\hat{Z}_1, \hat{Z}_2, \ldots, \hat{Z}_n)$.

Example 3.4. Let $L = 2123121212$ be a sequence. Then the corresponding $\hat{Z}$-vector is

$$\hat{Z} = (0, 4, 0, 2, 0, 0, 3, 0, 1).$$

★

Now, suppose that some finite sequence L of length $n+1$ is a subsequence of a ultimately periodic sequence with period p and preperiod n'. Let $\hat{Z} = (\hat{Z}_1.\hat{Z}_2, \ldots, \hat{Z}_n)$ be the corresponding $\hat{Z}$-vector. Then if $p, n' \leq n$, we have that

$$\hat{Z}_p > 0 \text{ and } n' = n - \hat{Z}_p - p + 1.$$

Furthermore, in order to confirm the ultimate periodicity, we need the given p, n to satisfy the Guy-Smith periodicity theorem. Hence, we need only to go through the vector and see if some choice of $Z_p > 0$ yields a p and n' satisfying the inequality $n \geq 2n' + 2p + k$ where k is the last non-zero digit of the finite octal $\Gamma = d_0.d_1d_2 \ldots d_k$ with the given nim-sequence. There are at most n possible values for p so given such a list L of $n+1$ elements and the vector $\hat{Z}$, we need only $\mathcal{O}(n)$ steps to determine p and n'. So to prove that we can determine p and n in linear time, it is sufficient to show that we can determine the Z-vector linear time.

A naive approach yields that it is possible to determine the Z-vector in $\mathcal{O}(n^2)$ steps. We can do better than this, however, as is shown by the algorithm below.

Procedure 5 Computes the Z-vector of some finite subsequence

Input: A list L.

Output: A vector Z of non-negative integers.

```
function Z-ALGORITHM($L$)
    l = length($L$)
    $Z \leftarrow [-1]$                              ▷ The first element of $Z$ is irrelevant
    $L \leftarrow 0$
    $R \leftarrow 0$

    for all $i$ from 1 to $l$ do
        if i > R then
            $L \leftarrow i$
            $R \leftarrow i$

            while $R < l$ and L[R-L] == L[R] do
                $R \leftarrow R + 1$                 ▷ $R$ increases until there is a mismatch.
            end while

            $Z \leftarrow Z + [R - L]$
            $R \leftarrow R - 1$
        else
            $k \leftarrow R - L$
            if $Z[k] < R - i + 1$ then
                $Z \leftarrow Z + [Z[k]]$
            else
                $L \leftarrow i$

                while $R < l$ and L[R-L] == L[R] do
                    $R \leftarrow R + 1$             ▷ $R$ increases until there is a mismatch.
                end while

                $Z \leftarrow Z + [Z[R - L]]$
                $R \leftarrow R - 1$
            end if
        end if
    end for

end function
```

Theorem 3.4. *The Z-algorithm terminates in linear time.*

Proof. It is sufficient to prove that the for-loop in steps $6 - 26$ takes linear time as everything else runs in constant time.

Let l be the length of the list L. Note that the only steps in the for-loop that are not constant time operations are the two while-loops where we compare indexes. So the time spent in the for-loop but outside the while-loops is $\mathcal{O}(l)$. It remains to show that the algorithm spends only $\mathcal{O}(l)$-time inside the while-loops.

Now, whenever inside one of the while loops, there can be at most l matches, since for each i, R will never decrease. On the other hand, we can only mismatch once for each i so there can be at most l mismatches. So in total we have

$$T(l) = \mathcal{O}(l) + \mathcal{O}(l) + \mathcal{O}(l) = 3\mathcal{O}(l) = \mathcal{O}(3l) = \mathcal{O}(l).$$

■

4 Results

Using the methods described in the previous section, we were able to confirm many of the computational results of Achim Flammenkamp using the Cheating Sparse Space algorithm. One example of such an implementation in C++ can be found at:

https://gist.github.com/JoakimUhlin/cb340d2dba0a2a9412f11a322de46138

Table 3: Non-trivial octal games with a sparse space that we managed to solve.

Game	Preperiod	Period	Bitstring
0·45	498	20	...111111111
0·156	3479	349	...111111011
0·055	259	148	...111111111
0·644	442	3256	...111111110
0·356	7315	142	...111111011
0·56	326640	144	...111011011
0·16	105351	149459	...111111110
0·376	2268248	4	...111111110
0·454	160949019	4	...111110111
0·54	193235616	796	...111111011
0·354	10061916	3227	...110111111

Unfortunately, we were only able to compute around 10^9 values of the nim-sequences of each respective game due to computational constraints. Observing the computational data provided by Flammenkamp [6] suggest that some of octal games in the table below could have a relatively small preperiod and preperiod.

Table 4: Octal games that might have a relatively short preperiod and period

Game	Current index	Presumed bitstring	Current rares	Current last index	Current max
0.167	2^{37}	$\ldots 11111101$	60	1303	64
0.174	2^{39}	$\ldots 11111011$	57	674	84
0.245	2^{35}	$\ldots 11111010$	151	1325	142
4.045	2^{37}	$\ldots 11011111$	34	497	93
4.367	2^{36}	$\ldots 11111110$	142	3508	101

Current index is the number of Grundy-values computed by Flammenkamp. **Presumed bitstring** is the bitstring that characterizes the presumed sparse space. **Current rares** is the number of rare values in the nim-sequence with index equal to or less than current index. **Current last index** is the $n \leq$ current index for which $g(G_n)$ is a presumed rare value, with regards to the presumed bit string. **Current max** is the maximal $g(G_n)$ for $n \leq$ current last index.

References

[1] E. R. Berlekamp, J. H. Conway, R.K. Guy, Winning ways,
1- 2 Academic Press, London (1982). Second edition,
1-4. A. K. Peters, Wellesley/MA (2001/03/03/04).

[2] John H. Conway: On numbers and games. Academic Press, London (1976). Second edition: A. K. Peters, Wellesley/MA (2001).

[3] Aviezri S. Fraenkel, Ofer Rahat
Infinite cyclic impartial games, (1998)

[4] J. Sjöstrand, IMPARTIAL GAMES AND SPRAGUE-GRUNDY THEORY — LECTURE NOTES,
https://www.math.kth.se/matstat/gru/sf2972/2013/lecture9_2013.pdf.

[5] A. N. Siegel, Combinatorial Game Theory, (2013).

[6] A. Flammenkamp, Sprague-Grundy Values of Octal-Games
http://wwwhomes.uni-bielefeld.de/achim/octal.html

[7] S. Arora, B. Barak, Computational Complexity: A Modern Approach,
Cambridge University Press; 1 edition, (2009)

[8] J. A. Beachy, W. D. Blair, Abstract Algebra
Waveland Press, Inc.; 3 edition (January 1, 2006)