Combinatorial Game Theory:
An analysis of the games Nim and Cutcake

Rhoda Durodola

31135714

Supervisor: Dr Athanassios N. Avramidis

Abstract

Combinatorial game theory studies the strategies of games involving two players with perfect information and no randomisation (for example, rolling dice). This is not to be confused with the classical game theory which is a partition of economics in which players making decisions simultaneously as opposed to alternating moves. This is the main distinction between the two theories. The objective of this project is to review the optimal strategies for some of these games that have been discovered, mainly Nim and to discuss the implications of these, for example, in regards to whether these strategies are subject to the winning player making the first move. As I progress in exploring this game, I hope to gain an better understand of what it means to solve a game and the methods used to do so. I will then apply this knowledge to attempt to find a winning - or at the very least, non-losing - strategy to the game Cutcake.

Contents

List of Figures

1 Introduction

A combinatorial game typically involves two players that rotate play in well-defined moves. Consequently, it is entirely strategic: how to play the game optimally facing a perfect adversary. It is usually assumed that the game ends after a finite amount of operations, and the outcome would be a single winner. Although, there can be special cases for example, games similar to Chess can be played indefinitely, and others like tic-tac-toe define ties in some instances.

Numerous approaches for investigating multi-player combinatorial games have been developed in past years. Li [6] is one of many who endeavours to characterise n-player games by establishing prohibitive assumptions surrounding each player's actions. Loeb [7] expands on this by proposing the concept of a solid prevailing alliance, where a member of this group is assured a win. The key distinction of n-player combinatorial games is that each participant plays in a fixed, sequential order. This paper does not discuss further any more details regarding $n > 2$ player games, where n is the number of players.

Returning to the main focus of 2 player combinatorial games, the winner of such game may be regarded as the last player to make a move - this is defined as *normal play*. *Misère play*, on the contrary, occurs when the first player unable to move is labelled the winner. If it is viable to return to a previous position in a game, it is called a loopy game (for example chess). Lastly, if each player is given the same plays accessible from a certain position, the game is deemed *impartial*; otherwise, it is known as *partisan*. For example, chess is partisan as only one player can move the white pieces and likewise for the black pieces.

The book Winning Ways is perhaps the most holistic source for the topic that is combinatorial game theory. Throughout this paper, I explore the optimal strategy discovered for certain games, in addition running these on python with a brief analysis of different scenarios. I hope to extract relevant details from this text alongside.

We start of with a more formal and elaborate discussion of combinatorial games in Section 2 before divulging into a detailed analysis of the game of Nim in Section 3 with implementation in python. Following this will be a rigorous investigation in hope to discover winning strategy for Cutcake.

2 Key concepts

2.1 Definition of a Game

Mathematicians define a game recursively, in terms of positions rather than games itself. A position is regarded as a set, that takes the form $\{L \mid R\}$, where L and R are the positions that can be reached in one move by the Left and Right players respectively. Therefore, each game may be written as

$$G = \{G^{L_1}, G^{L_2}, ... \mid G^{R_1}, G^{R_2}, ...\}.$$

This can be simplified to

$$G = \{G^L \mid G^R\}$$

where G^L and G^R are sets of left and right options which can be infinite or empty for example. An empty set represents *end game*, where neither Left nor Right has an option to move [1].

2.1.1 Game addition

The *sum* or *disjunctive compound* of a game is equivalent to playing multiple games concurrently. If you have a game with more than 2 elements, a player can chose to make a move in one of these components; this can be defined as

$$G + H = \{G^L + H, G + H^L \mid G^R + H, G + H^R\}[1]$$

$G^L + H$ for example, indicates the set of moves a player can make $(G^{L_1} + H, G^{L_2} + H, ...)$ within the chosen components.

2.2 Definition of a Combinatorial Game

This paper will focus on a subset of games known as combinatorial games. According to Berlekamp et al.[1], combinatorial game is one that (almost always) satisfies the following conditions:

1. There can only be two players, usually called Left and Right.
2. There are a finite number of positions, and often a specific starting point.
3. There are clear, defined rules which specify the moves that Left and Right are able to make from a given position.
4. Players Left and Right take turns in the game.
5. In normal play, a player that can not move loses (as opposed to misère play where a player unable to move wins).
6. The rules are such that a play will always end if some player is unable to move. This is called the ending position. There are no games which are drawn by repetition of moves.
7. Both players have complete information.
8. There are no moves of chance.

The most appreciable constraint is that no stochastic events are permitted — in other words, games involving dice and other such random sources are not combinatorial but rather deductive, where the players have complete knowledge.

From these set of rules it is easy to see that a game such as poker would not be classified as combinatorial as the allure of the game violates conditions 7 - players are unaware of their opponents cards, in addition to condition 8, and 1 - possibility of coalitions.

3 Nim

3.1 A brief history

The game of Nim is the most well-known unbiased combinatorial game. There are several accounts of Nim originating in China. Coins, stones and other miscellaneous items were used to play the game. Nim is eponymous of the dated English word meaning to steal or remove and was given it's name by Charles Bouton, a Harvard professor, circa 1901. Bouton provided a proof outlining a winning strategy for the game in 1901. His technique was built around the binary number system [3], which we shall discuss in further depth in this section. A lot of the foundations of combinatorial game theory has emerged from the analysis of the game of Nim.

3.2 The rules of Nim

The game starts with p piles with an arbitrary number of coins (or stones etc) in each pile. Two players, Left and Right, alternate removing ('nimming') any amount of coins from any one pile on each turn until there are no coins left. In normal play, the player to take the last coin wins, although it is typically played misère.

Nim may be viewed as the sum of multiple one-pile Nim games.

3.3 Examples

3.3.1 Example 1

A game of Nim with only one pile of n coins may be elucidated by the following graphical representation:

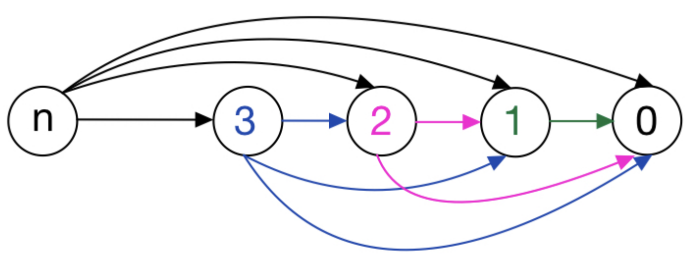


Figure 1: Nim with one pile

This graph displays all the possible moves the first player can make. Each of the n coins are represented by the different states. Each arrows delineates the possible moves that the players can make. For example, player one is able to remove enough coins to leave 3 remaining, this is showed by the arrow from state n to state 3.

It is worth noting that this game is comfortably won by the player that takes the initial turn - they would simply collect $n-1$ coins in the heap giving their opponent no choice but remove the last coin.

3.3.2 Example 2

The following is another diagram with instead two piles of 1 and 2 coins:

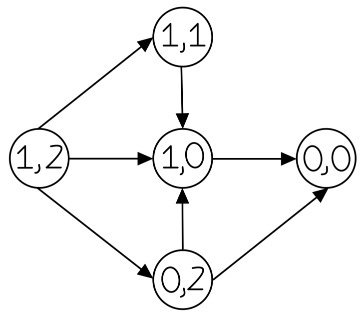


Figure 2: Nim with two piles

Each state has two values, which signifies the number of coins that each pile contains. The beginning state has 1 coin in the first pile and 2 coins in the second pile. As in Example 1, each arrow illustrates the possible moves that a player can make. For example, at the start of the game, Left (player 1) has three options: the first is to remove 1 coin from the pile with 2 coins and thus arriving at the state (1,1), the second is to remove the 2 coins from the second pile arriving at the state (1,0), and the third is to remove 1 coin from the first pile, arriving at the state (0,2).
Here, we can observe that it would be ideal for Left to move to state (1,0), which in turn limits Right (player two) to take the last coin and move to a state (0,0), hence allowing Left to win the game.

3.4 Game Positions

A game is in *P-position* if it is a winning position for the Previous player (the player that has just moved).
A game is in *N-position* if it is a winning position for the Next player (the player that is about to move).

To identify whether a state is in position N or P, we use a method known as *backwards induction* to work form the end of the game, to the start. The following process is for misère play:

1. Label every terminal position as N.
2. Label every positions that reaches an N position as P.
3. Label every position that can reach a P position as N.
4. All positions should be labeled, otherwise return to step 2 and repeat.

For normal play, we may simply begin by labelling a terminal position as P instead, and continue from step 2.
Exploring Example 2 again, let us label the terminal position, state (0,0), as N. Following the above guidelines, state (1,0) will be P. If Left chose to go to state (1,1), this would have resulted in an N position, allowing Right to win.

3.5 Nimbers

Nimbers (also know as Grundy Numbers) are defined as the values of the piles in Nim. In other words, they are simply the number of coins in a pile.

Binary addition without carrying is the crucial gambit in working out the strategy of Nim. To add Nimbers, we first encode them to binary numbers, then take the *Exclusive Or* (XOR - a binary operation) of the two numbers (this is adding the binary numbers without carrying). The result is referred to as the *Nim sum*. Note that in the XOR operation, we can write the addition of two numbers x and y as $x \bigoplus y$.

Value	Binary	Value	Binary	Value	Binary
5	101	2	010	2	010
$\bigoplus 3$	011	$\bigoplus 4$	100	$\bigoplus 2$	010
—	—	—	—	—	—
6	110	6	110	0	000

The last example demonstrates one of the notable features of nim addition. A nim number x is its own inverse, so $x + x = 0$.

It can be proven that nim sum is:

- Commutative: $x \bigoplus y = y \bigoplus x$
- Associative: $(x \bigoplus y) \bigoplus z = x \bigoplus (y \bigoplus z)$, and
- $x \bigoplus y = 0$ iff $x = y$

For reference, the Nim addition table is shown in figure 3.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	3	2	5	4	7	6	9	8	11	10	13	12	15	14
2	3	0	1	6	7	4	5	10	11	8	9	14	15	12	13
3	2	1	0	7	6	5	4	11	10	9	8	15	14	13	12
4	5	6	7	0	1	2	3	12	13	14	15	8	9	10	11
5	4	7	6	1	0	3	2	13	12	15	14	9	8	11	10
6	7	4	5	2	3	0	1	14	15	12	13	10	11	8	9
7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
9	8	11	10	13	12	15	14	1	0	3	2	5	4	7	6
10	11	8	9	14	15	12	13	2	3	0	1	6	7	4	5
11	10	9	8	15	14	13	12	3	2	1	0	7	6	5	4
12	13	14	15	8	9	10	11	4	5	6	7	0	1	2	3
13	12	15	14	9	8	11	10	5	4	7	6	1	0	3	2
14	15	12	13	10	11	8	9	6	7	4	5	2	3	0	1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 3: Nim addition table [5]

This table can be red as such: If we pick a value along the first row and add it to a number down the first column, the nim value is the number that these values correspond to. An example is $7 \bigoplus 3 = 4$.

3.6 Nim Solution

Now let us analyse a Nim game where we begin with 3 piles of 3, 4 and 5 coins. In binary, these would coincide with the values 011, 100 and 101 respectively.

Value	Binary
3	011
$\bigoplus$ 4	100
$\bigoplus$ 5	101
—	—
2	010

From the table above, we can see that the nim sum is 2 or 010. In order to be in a winning position for normal play, a player should always ensure that the nim sum on their move is 0. So player 1 can procure this by removing 2 coins from pile one as shown below:

Value	Binary
1	001
$\bigoplus$ 4	100
$\bigoplus$ 5	101
—	—
0	000

Thus, we can conclude that any game position where the nim sum is 0 is indeed a P position, otherwise it is an N position. From this point, we can assume that whatever move the opponent makes, Left can return to a P position, because any move Right makes, changes the nim value from 0, so Left is able to return the nim value back to 0.

Of course, in practise it may be difficult to keep track of the nim sum in one's head. Primarily, it is quite clear that whenever possible, a player should minimise the game to two piles of the same number of coins, as this will obviously have a nim sum of 0. At this point, the player would just mimic the opponents move until they are able to take the final coin. In other scenarios, a player can consciously leave an even number of 'sub' piles that each contain a power of 2 coins. For example if we had four piles with 1, 3, 5, and 7, their nim sum can be viewed as $1 + (2 + 1) + (4 + 1) + (4 + 2 + 1)$. Without even calculating their binary values we can see that there are an even number of 1s, 2s and 4s so their nim sum is automatically 0. If the last pile had 8 coins, Left would simply need to remove 1 coin from this pile to achieve the desired nim sum.

The strategy for misère play is exactly the same as above with a slight modification. This is that a player must ensure there is an odd number of piles containing a singular coin, so that the opponent is left with the last coin.

A final context to consider is if the game already begins with a nim sum of 0. In this case, Left has no choice but to play hope their adversary makes a mistake and play randomly in the meanwhile. The worst case scenario is to prolong defeat as much as possible! However, for Right this situation would be ideal as they can easily return to a P position where the nim sum is 0, assuring them a win from the very start.

3.7 Nim in Python

Appendix A contains the code for creating a Nim game in Python. The code generates a random number of piles ranging from 2 to 4, each of which can hold up to 10 coins. Two players are able to take turns removing any quantity of coins from a pile. The loser is determined by the player who retrieves the last coin.

3.7.1 Game 1

The first game produced has a nim sum of 2. For Left to win, they need to ensure that there is an odd number of piles with only one coin, aforementioned. So Left removes two coins from pile three in order to reduce the nim sum to 0, bringing the game to a P position.
Right retaliates by removing four coins from pile one, bringing the nim sum to 4:

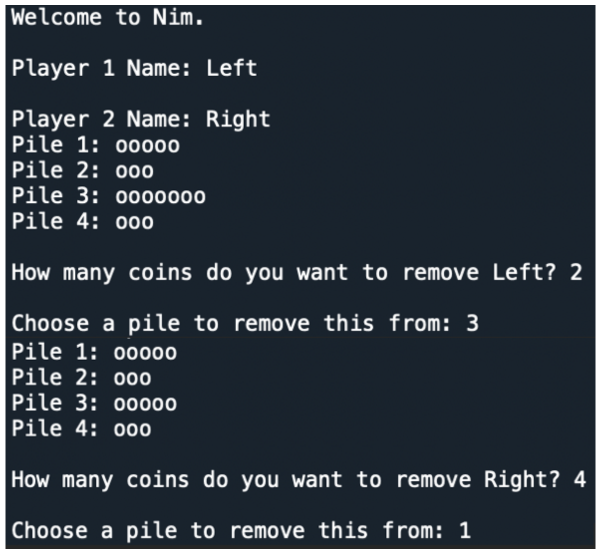


Figure 4: Game 1 - Part A

Left then mimics Right and retrieves four coins from pile three to maintain a nim sum of 0.

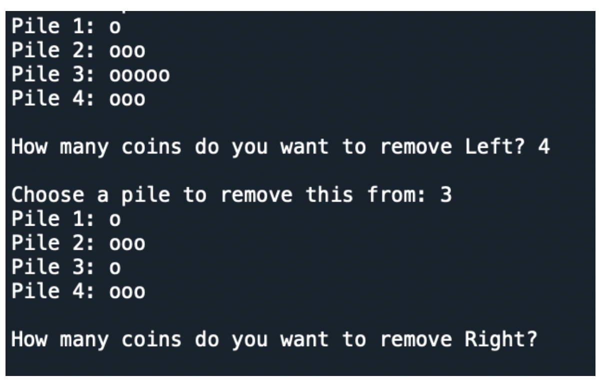


Figure 5: Game 1 - Part B

From this point, it is somewhat clear that Left has won the game (that is if they continue to play optimally!), regardless of the four moves Right can make, which are:

1. Remove one coin from pile one (or three).
2. Remove one coin from pile 2 (or four).
3. Remove two coins from pile 2 (or four).
4. Remove three coins from pile 2 (or four).

For the sake of completion, let us go with option 3. After this move, Left in turn, takes three coins from pile three, leaving an odd number of single coin piles and thus ensuring that Right is unfortunately left with the last coin.

```
How many coins do you want to remove Right? 2

Choose a pile to remove this from: 2
Pile 1: o
Pile 2: o
Pile 3: o
Pile 4: ooo

How many coins do you want to remove Left? 3

Choose a pile to remove this from: 3
Try again as you entered an invalid value

How many coins do you want to remove Left? 3

Choose a pile to remove this from: 4
Pile 1: o
Pile 2: o
Pile 3: o
Pile 4:

How many coins do you want to remove Right? 1

Choose a pile to remove this from: 1
Pile 1:
Pile 2: o
Pile 3: o
Pile 4:
```

Figure 6: Game 1 - Part C

```
How many coins do you want to remove Left? 1

Choose a pile to remove this from: 2
Pile 1:
Pile 2:
Pile 3: o
Pile 4:

How many coins do you want to remove Right? 1

Choose a pile to remove this from: 3
Pile 1:
Pile 2:
Pile 3:
Pile 4:
Oh no you took the last coin. Better luck next time Right.

Would you like to play again? Enter Y if Yes and anything else if No: N

In [4]:
```

Figure 7: Game 1 - Part D

The method used in this game by Left of mimicing every one of Right's moves is called the *Tweedledee and Tweedledum* strategy [2] and is used in many other games.

3.7.2 Game 2

This time, we begin with a nim sum of 0, so all Left is able to do is play randomly and hope that Right makes an error at any point during the game. Left proceeds by taking four coins from pile one, and Right imitates this by taking four coins from pile four.

```
Welcome to Nim.

Player 1 Name: Left

Player 2 Name: Right
Pile 1: ooooo
Pile 2: oooo
Pile 3: oooo
Pile 4: ooooo

How many coins do you want to remove Left? 4

Choose a pile to remove this from: 1
Pile 1: o
Pile 2: oooo
Pile 3: oooo
Pile 4: ooooo

How many coins do you want to remove Right? 4

Choose a pile to remove this from: 4
Pile 1: o
Pile 2: oooo
Pile 3: oooo
Pile 4: o
```

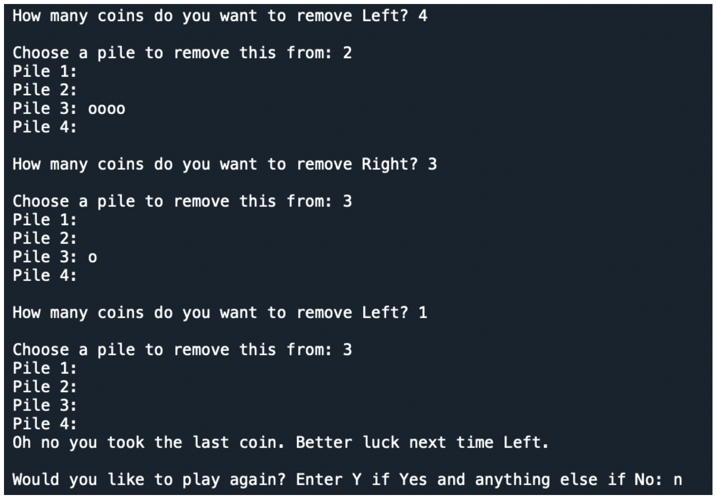
Figure 8: Game 2 - Part A

Left then takes the remainder of pile one, and again Right does the same with pile four.

```
How many coins do you want to remove Left? 1

Choose a pile to remove this from: 1
Pile 1:
Pile 2: oooo
Pile 3: oooo
Pile 4: o

How many coins do you want to remove Right? 1

Choose a pile to remove this from: 4
Pile 1:
Pile 2: oooo
Pile 3: oooo
Pile 4:
```

Figure 9: Game 2 - Part B

Now, Left removes all of pile two, with the hope that Right remains consistent with their Tweedledee and Tweedledum strategy. Unluckily, Right does not give credence to this and instead picks three coins from the last pile, leaving the final coin to Left.



```
How many coins do you want to remove Left? 4

Choose a pile to remove this from: 2
Pile 1:
Pile 2:
Pile 3: oooo
Pile 4:

How many coins do you want to remove Right? 3

Choose a pile to remove this from: 3
Pile 1:
Pile 2:
Pile 3: o
Pile 4:

How many coins do you want to remove Left? 1

Choose a pile to remove this from: 3
Pile 1:
Pile 2:
Pile 3:
Pile 4:
Oh no you took the last coin. Better luck next time Left.

Would you like to play again? Enter Y if Yes and anything else if No: n
```

Figure 10: Game 2 - Part C

The results of this study indicate that the nim sum of a game is the key variable in predicting the winner of said game, not necessarily the player that starts first, as is the case in many other games. We can say that Nim can be won by player two if and only if the nim-sum of the remaining piles is 0.

4 Sprague-Grundy Theorem

Recall that a game is considered to be impartial if each of the two players, Left and Right have the same moves available to them for any given state S (refer to Definition of a Combinatorial Game). The Sprague-Grundy theory [4, 8] states that any such impartial game that is also finite, is equivalent to a single game of Nim which is distinguished by a single natural number n. This theory has been further generalised to all impartial games by generalizing Nim to all ordinals. It's important to clarify that this theorem is applicable only to games of normal play [1]. As a result of this, we will proceed with analysing the game Cutcake in normal play.

5 Cutcake

5.1 Background

The game of Cutcake can be found in Winning Ways [2], and is described as a game that is played on a rectangular grid. Each player takes a turn cutting the rectangle into smaller pieces. Left is permitted only to cut from north to south, while Right cuts from east to west. The player that is unable to move (i.e. make a cut) loses.

This game is partisan because Left can only cut vertical lines and Right, horizontal. We will change this rule so that both players can cut in any direction, we do this so that we can apply the Sprague-Grundy theorem of impartial games. More formally, we can state the rules of this modified version of the game in the following section.

5.2 The rules of Cutcake

Say we have an m x n grid. Each player, Left and Right, take turns removing an a x a section of the grid until it is no longer possible to do so. Individual cells of the grid removed together is what we refer to as a section. The player that remove the last cell with a legal move, wins the game. We also have $a < m$ and $a < n$.

We can imagine the grid to be a cake or cookie (as stated in Winning Ways [2]) where the cuts are made using an a x a pastry cutter which we may refer to as size a, and does not have to completely cover a section of the grid.

5.3 Examples

5.3.1 Example 1

The first example shows a 2x2 grid and illustrates the different cuts that the first player (Left) can make on their initial turn, in the form of $G = \{G^{L_1}, G^{L_2}, G^{L_3}, G^{L_4} \mid G^R\}$ (Right's moves are not presented here).

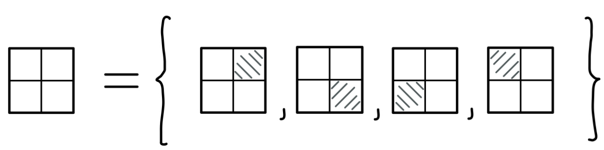


Figure 11: Example 1 of Cutcake

Left begins with a cutter of size 1 so we can see the different positions or moves that Left can take. These arise simply by a removal of any cell from the four quadrants of the initial grid.

5.3.2 Example 2

This second example displays a Cutcake game in full, where Left and Right take turns cutting out pieces using a cutter of size 2 from the grid until no more cuts can be made.

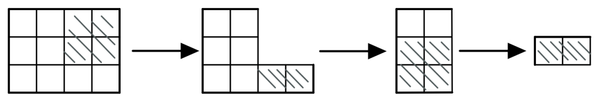


Figure 12: Example 2 of Cutcake

Left makes their first move by cutting out a 2x2 section in the top right corner, which is shown by the shaded area. Right then removes a 1x2 section from the bottom right shown in the second image. Next Left makes a cut leaving the top row available for Right to claim. Thus Right is labelled the winner.

5.4 Cutcake solution

We will deduce a winning strategy for Cutcake by looking at multiple scenarios consisting of different sized grids. The cutter will be fixed at size 2 in all instances. We also restrict the number of cells that can be removed on a player's turn to be in the set $s = \{2, 4\}$.

5.4.1 Case 1 - even x even grid

In this first case, we have an m x n gird where m and n are even numbers and $m = n$.

We begin by looking at a 4x4 grid. This is a game that can easily lead to a win by player 1, depending on their initial move. The strategy is for Left to make a cut in the centre of the grid, then employ the tweedledee and tweedledum strategy against Right in a symetrical manner (either horizontal or vertical) until they successfully make the final cut. Figure 13 provides an example of the sequence of moves that induces a victory for player 1. Left moves are shown by the blue shading, which represents a cut made, and Right, red.

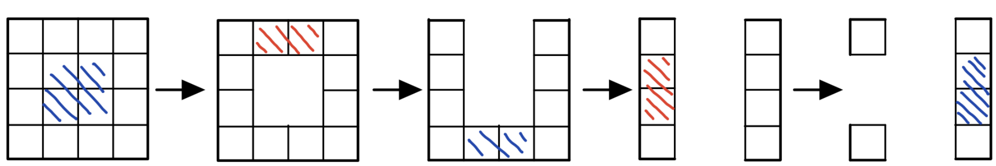


Figure 13: Example of a 4x4 grid

At the point of the final move shown, we have reached end game, where Left has won as there are no more cuts of two or four cells that can be made with a cutter of size 2; there are only singular cells remaining - and removing these are illegal moves. As a result, we may name this a terminal position, which we label a P-position (we are playing in normal convention). From this, we may work backwards to determine that the starting position is an N-position.

Now let us test this strategy on a larger, 6x6 grid.

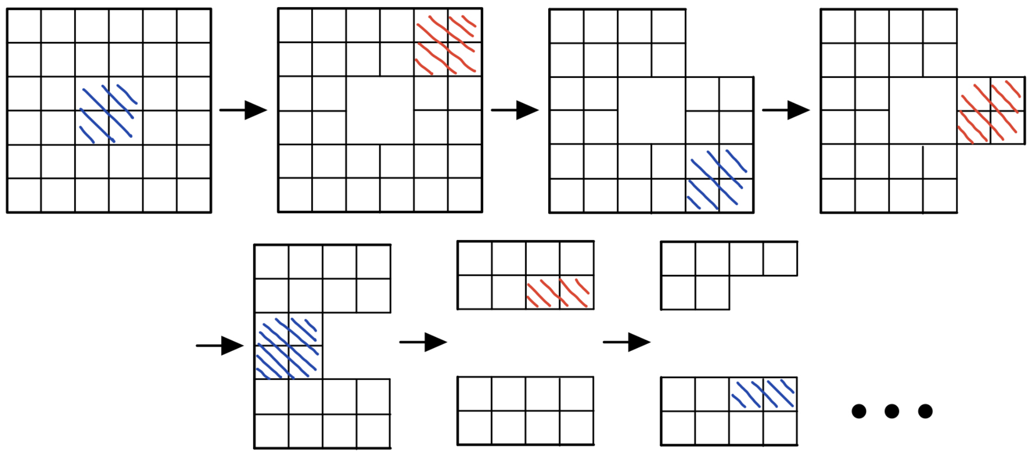


Figure 14: Example of a 6x6 grid

This game is incomplete, but we can see that Left is sure to win, regardless of what move Right plays as we have a symmetrical shape on the horizontal axis.

5.4.2 Case 2 - even x odd grid

In this case, we have an mxn gird where m and n are now odd numbers. We will implement the same strategy used in case 1 in this scenario.

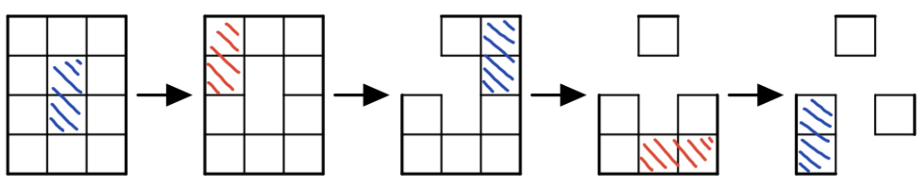


Figure 15: Example of a 4x3 grid

From figure 16, we can see that Left wins again under the same strategy. This is quite straightforward to understand, because an even number multiplied by an odd number is always an even number; and this is the same as case 1: even x even = even.

This example also shows that there can be some instances where player 1 is unable to mimic player 2's moves due to the current state of the game, shown in the penultimate stage. As there remains just one legal move, Left is still crowned winner.

In actuality, if we adhere to the original basis proposed, the commencing move played by Left would be illicit, due to the nature of a pastry cutter. Nonetheless, this move is permitted as by the rules of the game.

5.4.3 Case 3 - odd x odd grid

In the final case, we have an mxn gird. Having trialled many games, there doesn't seem to be a clear strategy for either players, as out of 10 trials roughly 50% of these are won by either player. This is due to the fact that there is no exact center with which player 1 might make the first move. Here are some examples:

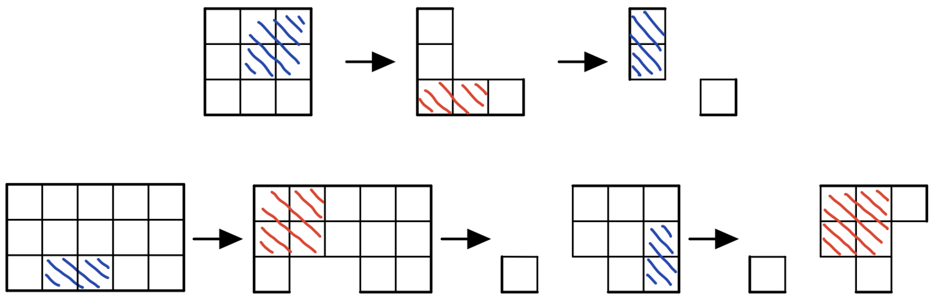


Figure 16: Example of a 3x3 and 3x5 grid

In the top row we are presented with a 3x3 grid where Left wins, and on the bottom we have a 3x5 grid where Right is now the winner.

It seems that an odd x odd grid is the best option for players that desire to play a game where the winner is not predetermined.

5.4.4 Argumentation

The Sprague-Grundy theorem suggests that any Cutcake position in normal play, is comparable to a nim pile of size n. The approach of imitating your opponent's move by symmetry suggests that even x even grids have a nim value of zero. We can recollect back to the game of Nim where a nim value of zero guarantees a win for the second player. However, this is not the case in Cutcake, simply because of the first move that player 1 is advised to make. The symmetry model also indicates that an even x odd (or odd x even) grid position holds a nim value of ≥ 1, which corresponds with a first player win.

6 Future work

As I delved into the project, I discovered there were many layers to the problem of trying to find a winning strategy of a combinatorial game. Time constraints limited me from testing all possible grid and cutter sizes for Cutcake to confirm the legitimacy of the strategy discovered.

An algorithm would be a useful tool to develop, in order to accurately predict whether the Cutcake strategy is applicable for larger grid sizes. This will provide an efficient method to examine all conceivable circumstances in each game to see whether the approach needs to be adapted.

Secondly, if a third player is brought into play, it will be fascinating to observe how the winning strategy may alter. Moreover, we could analyse the potential of alliances, in which two of the three players band together to avoid losing, and what tactics they would employ in such situation.

Lastly, it is also possible to explore in great depth a strategy for variations of both games. In regards to Nim, this could take the form suggesting that players remove same number of coins from more than one pile at a time. Concerning Cutcake, we may discuss scenarios with grids of various shapes, rather than limiting this to rectangles.

References

[1] Berlekamp, E.R., Conway, J.H., Fraenkel, A.S., Nowakowski, R.J. and Pless, V. (1990). *Combinatorial Games.* American Mathematical Society providence, Rhode Island.

[2] Berlekamp, E.R., Conway, J.H. and Guy, R.K. (1982). *Winning Ways for your mathematical plays.* New York: Academic Press Inc.

[3] Bouton, C.L. (1901). Nim, A Game with a Complete Mathematical Theory. *Annals of Mathematics*, [online] 3(1/4), pp.35-39. doi:10.2307/1967631.

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

[5] Guy, R.K. (1996). *Games of no chance.* Cambridge ; New York: Cambridge University Press, p.65.

[6] Li, S.-Y. . R. (1978). N-person Nim and n-person Moore's Games. *International Journal of Game Theory*, 7(1), pp.31–36. doi:10.1007/bf01763118.

[7] Loeb, D.E. (1996). *Games of no chance.* Cambridge ; New York: Cambridge University Press, pp.451–471.

[8] Sprague, R. (1935). Über Mathematische Kampfspiele. *Tohoku Mathematical Journal*, [online] 41, pp.434–444. Available at: `https://www.jstage.jst.go.jp/article/tmj1911/41/0/41_0_438/_article/-char/en\` [Accessed 24 Feb. 2022].

Appendices

Appendix A - Nim in python code

```
import random
print("Welcome to Nim.")

def start():
    coinlist = []
    piles = random.randint(2, 4)
    coins = random.randint(1, 10)
    player1, player2 = get_players()

    currentplayer = player1 # player 1 starts first

    initialsetup(coinlist, piles, coins, currentplayer)

    newgame(coinlist, piles, coins, player1, player2, currentplayer)

def get_players():
    return input("Player 1 Name: "), input("Player 2 Name: ")

#printing a randam set of piles of coins for the player
def initialsetup(coinlist, piles, coins, currentplayer):
    for i in range(0, piles):
        coins = random.randint(1, 7)
        print('Pile {}: {}'.format(i + 1, 'o' * coins))
        coinlist.append(coins)


def inputcheck(coinlist, piles, currentplayer):

    #loop that makes sure player 1 enters a valid input
    #(will keep asking if they haven't)
    #ctr = coins to remove, ptrf = piles to remove from
    while True:
        ctr = input\
            ('How many coins do you want to remove {}? '.format(currentplayer))
        ptrf = input('Choose a pile to remove this from: ')

        #loop is broken if all the conditions are satisfied:
        if (ctr and ptrf) and (ctr.isdigit()) and (ptrf.isdigit()):
            if (int(ctr) > 0) and (int(ptrf) <= len(coinlist)) and \
              (int(ptrf) > 0):
                if (int(ctr) <= coinlist[int(ptrf) - 1]):
                    if (int(ctr) != 0) and (int(ptrf) != 0):
                        break
```

```
        #does this if conditions aren't met
        print("Try again as you entered an invalid value")

    # updates the new coinlist from whichever pile is chosen
    coinlist[int(ptrf) - 1] -= int(ctr)
    #shows new piles after current player moves
    cont(coinlist, piles, currentplayer)

def cont(coinlist, piles, currentplayer):
    for i in range(0, piles):
        print("Pile {}: {}".format(i + 1, 'o' * coinlist[i]))

# prints loser of game form who took last coin (misere play),
# asks if players want to play game again,
def newgame(coinlist, piles, coins, player1, player2, currentplayer):
    # loop allows players to keep starting a new game
    while True:
        inputcheck(coinlist, piles, currentplayer)
        if coinlist == [0] * len(coinlist):
            print("Oh no you took the last coin.")
            print("Better luck next time {}. ".format(currentplayer))
            print("Would you like to play again?")
            user = input("Enter Y if Yes and anything else if No: ")

            if user.upper() == 'Y':
                start()
            else:
                break

        # players can take turns
        if currentplayer == player1:
            currentplayer = player2

        else:
            currentplayer = player1

start()
```