FACULTY OF SCIENCE AND TECHNOLOGY

BACHELOR'S THESIS

Study programme / specialisation: Advanced teacher education for 8-13 levels	The *spring* semester, *2023* Open
Authors: Tiril Abrahamsen Morten Eriksen	
Supervisor at UiS: Tyson Ritter	
Thesis title: The card game SET and finite geometry	
Credits (ECTS): 10	
Keywords: SET Finite geometry Counting Simulation Teaching Card game Modular arithmetic	Pages: 52 + appendix: 11 Stavanger, *15.05.23*

Approved by the Dean 30 Sep 21
Faculty of Science and Technology

The card game SET and finite geometry

Tiril Abrahamsen and Morten Eriksen

Spring 2023

Contents

1 Abstract

This thesis explores the underlying mathematics of the card game SET, which is a fast-paced pattern recognition game played with 81 cards. The game involves making SETs of three cards with certain compatibility conditions based on their color, shape, filling, and number properties. The first section gives an introduction to the game and its rules. The subsequent sections cover various mathematical concepts related to SET, such as counting SETs, modular arithmetic, and finite affine geometry. Section seven describes how simulations can be used to deal with a number of problems or questions related to the game. The thesis concludes by discussing how the game SET can be used in the classroom to develop cognitive and social skills in students. Through its exploration of the mathematical principles underlying the game, this thesis exposes the strong connection between SET and finite geometry, and offers insights into the use of SET as an educational tool.

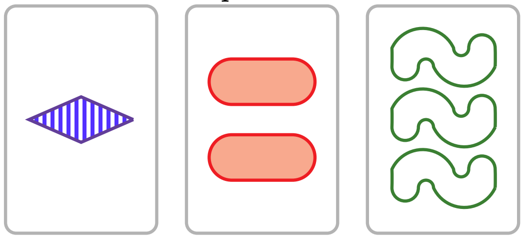
2 Acknowledgement

We would like to give our sincere gratitude to our supervisor, Tyson Ritter. Tyson supported and guided us on our journey in writing this thesis. We are truly grateful for his dedication in providing constructive feedback, insightful suggestions, and nurturing environment that encouraged us to explore new ideas and overcome challenges.

3 Introduction

The card game SET is a fastpaced pattern recognition card game. The entire deck consists of 81 cards and the purpose of the game is to make different SETs. One SET consists of three different cards where all three cards satisfy a certain very interesting compatibility condition. Each card has four different properties: Color, shape, filling and number, and each property has three values.

Example of a SET



The SET shown above is special in the way that it consists of all the different values within each property. All the number of objects are present $(1, 2, 3)$. All the different colors are in the SET (purple, red, green). The different fillings are present (striped, full, empty), and all the different shapes are in the SET (diamond, oval, worm). We will see later why this is a SET, and what other types of SETs we can form.

The game starts by laying twelve cards down on a table in a rectangle. Each player starts looking for SETs; there are no turns. When a SET is found the participants shout out "SET" and take the three cards away from the table. Each time a person says "SET" the SET is checked by the other players to make sure the SET is valid. Three new cards are placed on the table if a SET has been collected from the layout or if no SETs are found in the layout. The game continues until either all the cards are used up or it is not possible to make more SETs from the remaining cards on the table. At this point the person who has collected the most SETs has won the game.

There is only one rule about what constitutes a valid SET, but even though the game is based upon one single rule, there are some advanced underlying mathematics behind the game. In particular, the connection between the game and finite geometry is very strong. The mathematics of SET includes the study of the different properties as lines and points, similar to the mathematics in finite geometry. Using this geometry we can explore different problems using combinatorics and probability. In this thesis we are going to take a deeper look into these underlying mathematics of the card game SET.

Our thesis consists of six sections, which involves the following subjects:

- The **first** section is about the rules of SET, the definition of a SET, notation in regards to the game and the fundamental theorem of SET.
- The **second** section consists of things to count in SET, which includes probabilities, number of SETs and number of different types of SETs.
- The **third** section is about modular arithmetic and how it can be used to determine SETs. It also involves two techniques to determine a card, which was hidden at the start of the game, and how we can use vectors to find SETs.
- The **fourth** section is about finite affine geometry, which involves looking at lines, planes and cubes in SET, introducing parallel SETs and direction vectors in SET.

- The **fifth** section consists of simulations and coding in SET, where simulations was used to determine probabilities of different scenarios in SET.
- In the **sixth** and last section we discuss how the card game SET can be used in a classroom. The section talks about how to introduce the game to students and how the students can develop useful skills, such as cognitive and social skills, by playing the game.

4 Rules of the game

In this chapter we will talk about the rules of the game SET. The definition of a SET will be revealed, some notation to the game will be presented and a theorem regarding how to determine a SET from just two cards will be formed.

4.1 What is a SET?

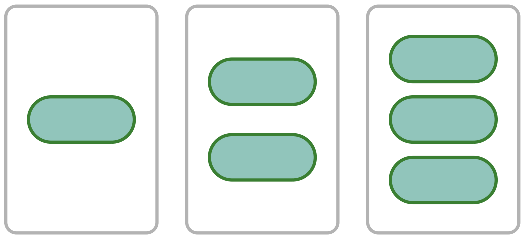
A SET consists of exactly three cards such that for each property the values of that property on the three cards are either all equal or all different. It is allowed to have a combination of properties for which the cards have the same values, as well as properties for which the cards have distinct values, within a single SET. In the case that the values of a property on each card are all different, each of the three possible values must then appear exactly once in the SET. The different values within each property are described in Table 1 below.

Table 1: Values and properties of the game

Number	Filling	Color	Shape
1	Hollow	Green	Oval
2	Striped	Purple	Diamond
3	Full	Red	Worm

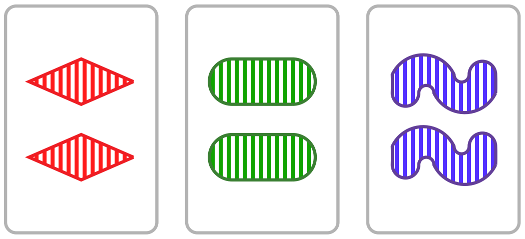
Here are some examples of SETs with an explanation to why it is a SET:

SET with **three** shared properties



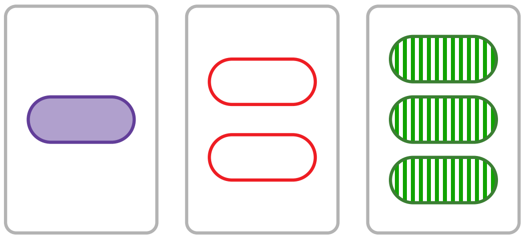
Explanation: The cards have identical values for three properties and distinct values for the remaining property. They have the same color (green), the same shape (oval), and the same filling (full). They all have a different number of objects: 1, 2 and 3. Since they all have the same value for color, shape, and filling, and take all possible values for the number, this is a SET.

SET with **two** shared properties



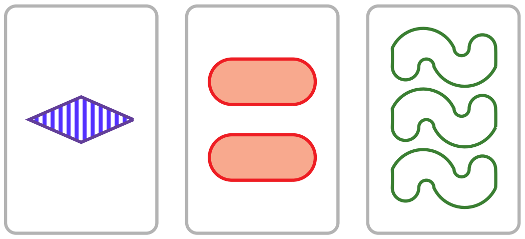
Explanation: These cards have identical values for two properties and distinct values for the remaining two properties. They all have the same filling (stripped), and the same number of objects (2). Since they all have the same value for filling and number, and take all possible values for color and shape, this is a SET.

SET with **one** shared property



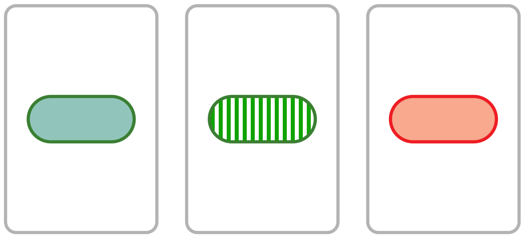
Explanation: The cards have identical values for one property and distinct values for the remaining three properties. They are all oval shaped objects. Since they all have the same value for shape (oval), and take all possible values for color, number and filling, this is a SET.

SET with **zero** shared properties



Explanation: This is the SET from the introduction, and these cards all have distinct values within all of the four properties. Since they take all possible values for color, number, filling, and shape, this is a SET.

We now give an example of three cards which do not form a SET:

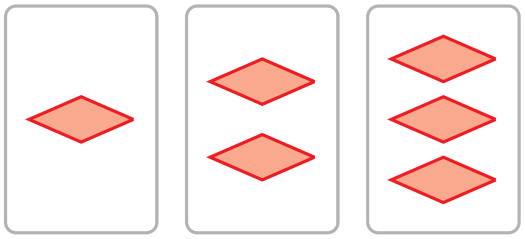


Explanation: Lets check why this is not a valid SET. In the properties number and shape the three cards meet the requirements to be a valid SET. This is because all the three cards share the same value in these two properties: They are all oval and has only one figure on the card. But when it comes to the last two properties, filling and color, the cards do not meet the requirements to be a valid SET. The first two cards are both green, but the third is red. Also when we check the filling, two of the cards are full, but one card is striped. That goes against the rules that a SET consists of three cards in which each of the card's properties either are shared on each card or are different on each card. This is why these three cards are **not** a valid SET.

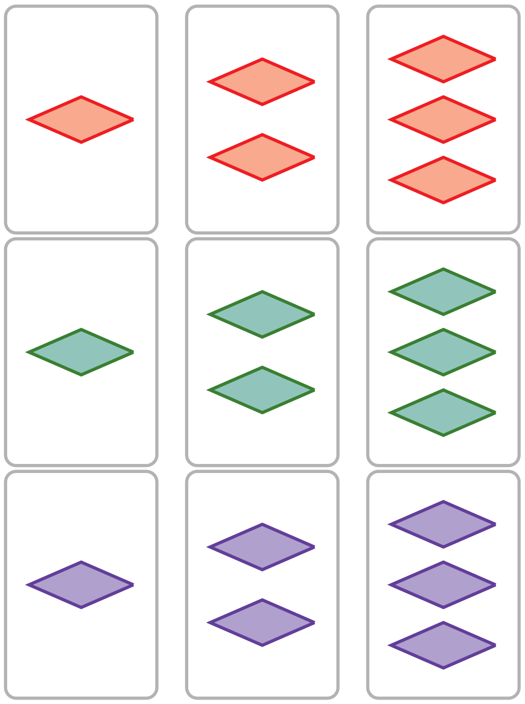
4.2 Notation

As we have seen the game consists of a stack of cards that have pictures of different figures on them. Within the deck exists four different properties: color, shape, filling and quantity. Refering to Table 1 we can see that each of these four properties has three different values. Each and every combination of the four different properties can only occur once, which means that there is only one card for each combination in the deck. Each card is completely unique and can only be used to create a SET once during normal gameplay.

If we try to calculate the total number of cards in the deck, we can try simplifying the problem. Imagine having a deck, which only had one property (the number of objects). This deck would consist of 3 cards: A card with one object, a card with two objects and a card with three objects as shown below.



Now we add a property, color. This deck would consist of $3 + 3 + 3 = 3 \cdot 3 = 3^2 = 9$ cards, since for each red card we would have to add one of each other color. This deck of cards would look like this:



If we add yet another property (shape) the deck would consist of $9 + 9 + 9 = 3 \cdot 9 = 3^3 = 27$ cards, since we would have to add the same nine cards as shown above, but with oval shape, and another nine cards with worm shapes. Similarly, if we add another property (shading), we would have $27 + 27 + 27 = 3 \cdot 27 = 3^4 = 81$ cards. This is the case for the card game SET, and this is the reason why there is 81 cards in the deck.

If we were to have a different number of properties, then the total number of cards in the deck, would be 3^p, where p is the number of properties. We will later see how this can be used to count various things in the game of SET.

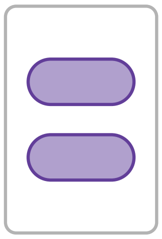
Now we want to introduce some notation.

Table 2: Definition of values and properties

Notation	Number	Filling	Color	Shape
0	3	Hollow	Green	Oval
1	1	Striped	Purple	Diamond
2	2	Full	Red	Worm

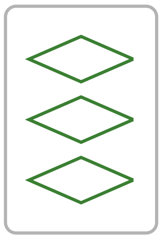
The table above shows the notation we have created to make it easier to talk about the different type of cards throughout the game. The assignment of values $0, 1, 2$ does not matter, as long as we keep the same notation throughout the thesis. Similar, it does not matter which property we chose to have for the first number in the notation. We have chosen to make the notation 0 as the number 3, since it would make perfect sense to use 1 as the notation for one object and 2 as the notation for two objects. Furthermore 3 is 0 mod 3. What modular arithmetic has to do with the game, will be discussed later in the thesis.

Let us take this card as an example: $(2, 2, 1, 0)$. Looking at Table 2 we can easily find out which card this is. We simply just follow the table from left to right in order to determine the right value for each property. In this case the card would have two objects, full filling, purple color and oval shape. Which looks like this:



$(2, 2, 1, 0)$

Let us take another example: $(0, 0, 0, 1)$. Using Table 2 from left to right, we get: three objects, hollow filling, green color and diamond shaped. Which is the card



$(0, 0, 0, 1)$

4.3 The Fundamental Theorem of SET

Players with some experience with the game will start to create different methods for finding SETs during the game. When players start playing the game for the first time, a common method to easily find a SET is to analyse two cards first. After the player has seen what the cards have in common and their differences, the player will see if there exists a third card on the table that will match the first two cards. This naturally leads to people asking if every pair of cards can be completed to a SET, and if a pair can be completed to a SET, how many choices are there for the third card? The answer to these questions comes in the form of a theorem, which we call the fundamental theorem of SET.

Theorem 1: Given two cards, A and B, then there is a unique card C such that the cards A, B, C form a SET.

Proof. Given two cards $A = (a_1, a_2, a_3, a_4)$ and $B = (b_1, b_2, b_3, b_4)$ where $a_i,\ b_i \in \{0, 1, 2\}$ we suppose there is a card $C = (c_1, c_2, c_3, c_4)$ which forms the SET A, B, C. Consider a_i and b_i. If $a_i = b_i$ then we set $c_i = a_i = b_i$. If $a_i \neq b_i$ then they are distinct values in the set $\{0, 1, 2\}$, and we let c_i be the remaining value, so $\{a_i, b_i, c_i\} = \{0, 1, 2\}$.

It is now clear that the cards A, B, C form a SET.

We know that the card C exists since all the cards are made up of all the combinations of the properties, but we need to show that it was not already selected (i.e. that it is still in the deck).

Suppose that $C = A$. Then $a_i = c_i$ for each i, so it must be that $b_i = a_i = c_i$ also, for each i. Then $A = B$, which is a contradiction, since all the cards are unique. $\square$

We will use this theorem a lot.

5 Counting in SET

In this chapter we will look at things to count within the game SET. Probabilities and number of SETs among different selection of cards will be explored. We will also explore different *types* of SETs, where we divide SETs into groups from the number of shared properties.

5.1 Basic Counting in SET

In the game SET there are some basic things to count. Below are displayed two basic things to count. The probability of three random cards being a SET, the different number of 12 card deals and the total number of SETs in the game will be determined.

Probability of making a SET from three random cards

Imagine picking three random cards from the deck. What is the probability that these form a SET? Start by picking two random cards. Referring to Theorem 1, only one unique card forms a SET with the first two. Since there is 79 cards left in the stock the probability of making a SET from three random cards is the probability of picking the unique card, out of the remaining 79 cards.

$$P = \frac{1}{79}\ .$$

To generalize this formula to a deck with p properties and 3^p cards, we get

$$P = \frac{1}{3^p - 2}\ .$$

Number of different twelve cards deals

When playing SET there should always be at least twelve cards laid down on the table. Let us look at the different twelve card layouts in the beginning of the game. Since the order of the 12 cards on the table is irrelevant the total number of different twelve card deals is

$$\binom{81}{12} = \frac{81!}{12!69!} = 7.07 \cdot 10^{13}\ .$$

This is a very large number, which makes every game unique, and the odds of having the same twelve card deals in two games in very small.

If we want to generalize the formula to a deck of 3^p cards, where p is the number of properties, we get the total of 12 card deals to be

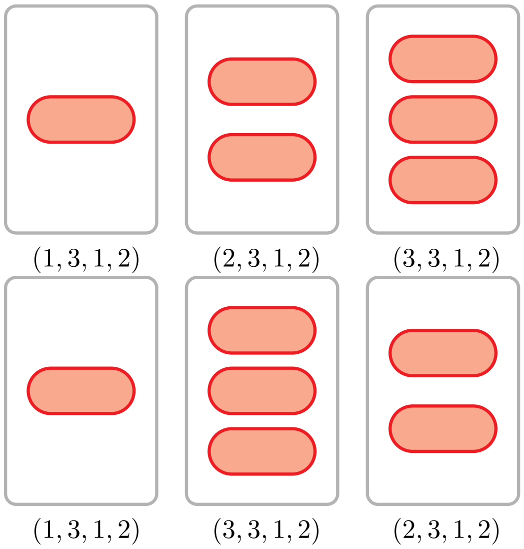
$$\binom{3^p}{12} = \frac{3^p!}{12!(3^p - 12)!}\ .$$

5.2 Interlude: Ordered and Unordered SETs

In mathematics a set is a collection of distinct elements, and we distinguish between sets and sequences without repetition. The elements of a set can be arranged in a particular sequence, or they can be unordered sets. Sets that maintain the order of their elements are called sequences without repetition, while sets that do not maintain the order of their elements are called unordered sets. In the card game SET the order the three cards does not matter, as long as the three cards together meet the requirements to be a valid SET. That means when we talk about SETs in this card game, we only talk about unordered SETs.

Ordered and unordered SETs is a relevant topic, when we try to count the total number of different SETs in the game. We want to distinguish between ordered and unordered SETs, since the result of our counting typically will give us the number of ordered SETs. Since the ordering of the cards in a SET does not matter, we have to divide by the permutation of three elements, 3!, to get the number of unordered SETs.

Below you can see one SET, that are laid in two different orders. The three cards meet the requirements to be a valid SET: All the cards share three of the four properties. They are all red, oval and the filling is full. In the last property that is number of figures on the card, they all have different values. This is just an example to show that this only counts as one SET and not two.



These two SETs are the same SET, but different as ordered sequences. There are 3! different orderings of the same three cards.

In summary, the key difference between ordered and unordered SETs is whether the order of their elements matters or not. Ordered SETs maintain the order of their elements, while unordered SETs do not. And in this card game, the order the cards does not matter and we are therefore only talking about unordered SETs further in this thesis.

5.3 More Basic Counting in SET

Now we will go back to some basic counting.

Number of SETs in total

Number of ordered SETs is

$$81 \cdot 80 \cdot 1 = 6480 \ .$$

This equation comes from picking one card from the 81 cards, then there is 80 cards left, where we pick the second card. Referring to Theorem 1 only one card can form a SET with the first and second card, so we multiply by 1.

In the game SET, we only see SETs as unordered SETs, so to find the number of unordered SETs we divide the number of ordered SETs with the permutations of three cards, which is 3!. Hence the number of unordered SETs is

$$\frac{81 \cdot 80 \cdot 1}{3!} = 1080 \ .$$

As shown we can go back and forth between ordered and unordered SETs using the number 3!. If you want to go from unordered SETs to ordered SETs you simply multiply by 3!, and if you divide by 3! you go from ordered SETs to unordered SETs.

Using the formula above we can calculate the total number of SETs in a deck with p properties:

$$\frac{3^p \cdot (3^p - 1) \cdot 1}{3!} \ .$$

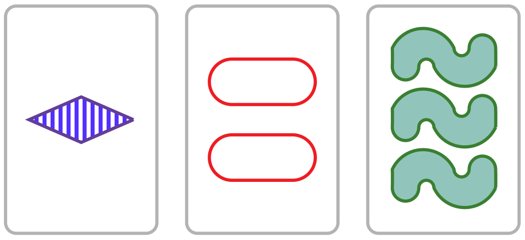
5.4 Advanced Counting in SET

We have now gone through some different basic things to count in SET. Now we will do some more advanced counting in SET.

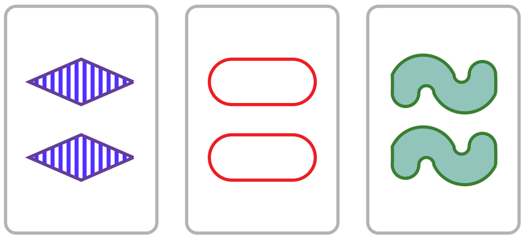
Number of SETs of each type

Now that we have counted the number of total SETs in the game, let us try to partition these SETs into 0 similarities, 1 similarity, 2 similarities and 3 similarities SETs. Shown below is an example of each type of SET.

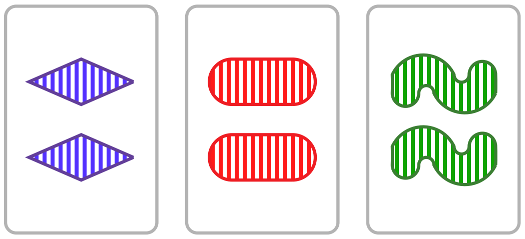
0 similarities



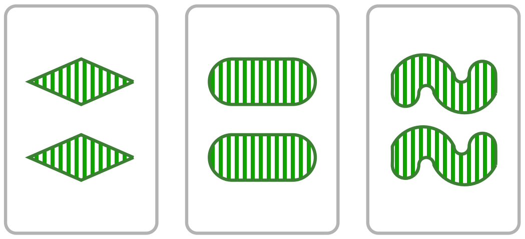
1 similarity



2 similarities



3 similarities



With x shared properties and 4 total properties we get the formula

$$\frac{81 \cdot \binom{4}{x} \cdot 2^{4-x} \cdot 1}{3!}\ .$$

Explanation of the formula: Start by picking a random card out of the 81 cards in deck. Then with x shared properties among four properties, we have $\binom{4}{x}$ cards where the properties take the same value as the first card. Now we consider the properties that should be different on the second card. There are two possible values for each property that are different from the first card, and we need $4 - x$ of the properties to take one of these two values. Hence, the total number of possible combinations is 2^{4-x}. Referring to the fundamental theorem there is only one unique card, which forms a SET with the first and the second card, so we multiply by one.

The formula from the previous page is used to generate the table shown below.

Table 3: Number of SETs of each type

Type	Formula	# of SETs	Probability
0 similarities	$\frac{81 \cdot \binom{4}{0} \cdot 2^{4-0} \cdot 1}{3!}$	216	20%
1 similarities	$\frac{81 \cdot \binom{4}{1} \cdot 2^{4-1} \cdot 1}{3!}$	432	40%
2 similarities	$\frac{81 \cdot \binom{4}{2} \cdot 2^{4-2} \cdot 1}{3!}$	324	30%
3 similarities	$\frac{81 \cdot \binom{4}{3} \cdot 2^{4-3} \cdot 1}{3!}$	108	10%
Total		1080	100%

As expected we get 1080 SETs in total. Notice that there is no 4 similarities type, since this would mean, that all the cards would be the same card, but there is only one of each card in the deck.

Using the formula we can generalize it with x shared properties and p total properties:

$$\frac{81 \cdot \binom{p}{x} \cdot 2^{p-x} \cdot 1}{3!} .$$

How many SETs can we create containing a given card?

The number of different SETs, where a given card occurs is

$$\frac{1 \cdot 80 \cdot 1}{2!} = 40 .$$

The logic behind this equation is, that we have given one card A and then pick a random card from the remaining 80 cards, B. Now we use the fundamental theorem, that there exists a unique card C, such that A, B, C forms a SET. We divide by 2! since the SET A, B, C is the same SET as A, C, B.

Generalizing this number to a deck with p properties we get

$$\frac{1 \cdot (3^p - 1) \cdot 1}{2!} .$$

How many SETs in n cards?

When calculating the number of SETs in n cards, we do not look at $n < 3$, since a SET consists of 3 cards. Furthermore when $n = 3$ it is obvious there can only exist one SET. So we would like to investigate for $n > 3$. The theoretical maximum number of SETs in n cards can be calculated using the following formula

$$\frac{n \cdot (n-1) \cdot 1}{3!} .$$

This formula looks similar to the equation where we found the number of SETs in 81 cards. This makes sense, since we first pick one of the n cards, and second we pick another card, which is the $n-1$th card, and from the fundamental theorem the third card is a unique card. We also have to divide with the number of times we can rearrange 3 cards, that is 3!. This formula gives us a theoretical maximum number of SETs in n cards, which we will typically not be able to achieve, even when it is an integer. Since in some cases of n this will not be a whole number, and for $n = 6$ this equation tells us that the theoretical maximum number of SETs is 5, which is not the case.

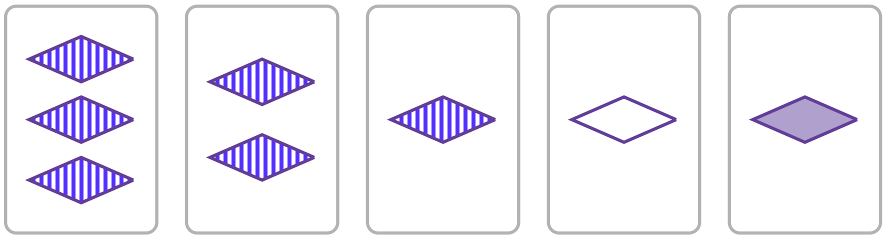
$n = 4$

Consider having four different cards A, B, C and D. We suppose A, B and C forms a SET. We consider whether it is possible to create a SET including D. But if D is in a SET, the other two cards must be taken from the cards A, B, C. Suppose, without loss of generality, that A, B forms a SET with D. This is a contradiction since the fundamental theorem tells us that C is a unique card which forms the SET A, B, C. Hence A, B, D cannot be a SET. Therefore it is not possible to make a SET including the card D. Thus, there can only exist one SET in 4 cards.

$n = 5$

Given five cards A, B, C, D and E. We suppose A, B, C and C, D, E forms a SET. We consider whether it is possible to form a SET using A or B as the first card and D or E as the second card. But if A, D is in a SET, the other card must be taken from the cards B, C, E. Suppose A, D, E forms a SET. This is a contradiction since the fundamental theorem dictates that C is the only card, which forms a SET with D and E. The same argument is used for picking A and E, B and D and B and E. Hence the maximum number of SETs in five cards is two.

Here is an example of five cards with two SETs:



$n = 6$

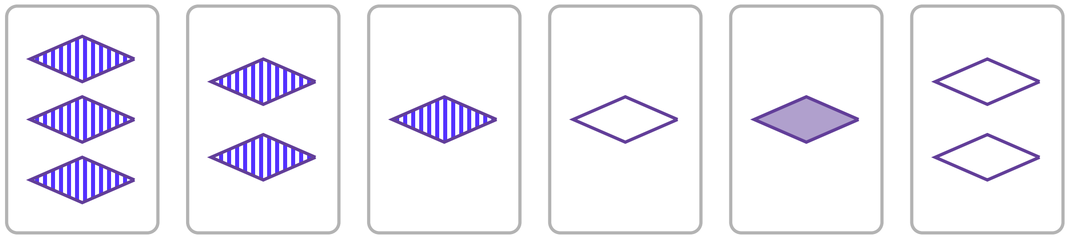
The key is that for any given two cards, there is exactly one third card that completes the set. Beginning with the simple construction of three SETs we have:

Given three unique cards A, B, C, which do not form a SET. We find the unique cards D, E, F such that ABD, ACE, and BCF are each SETs. We now have three SETs in six cards.

Imagine that there exists a fourth SET. We know that it cannot contain any two cards that are both in one of our existing SETs, since then it would have to also contain the same third card per the fundamental theorem. We can therefore combine D with C, E, F; E with B, D, F; and F with A, D, E. The only three-card SET that could follow this rule is DEF. Notably, A, B, C is NOT a SET.

Since A, B, C is not a SET, pick a property that causes them to not be a SET. As an example we choose the color. In this case, two of these cards are red and one is green - assume A and B are red and C is green. Since ACE and BCF are SETs then E and F must be purple. And since ABD is a SET then D must be red. This means that DEF cannot form a SET, since D is red and E and F are both purple. Since that was the only possible option, there can be no other SETs. Hence, the maximum number of SETs in six cards is three SETs.

Here is an example of six cards with three SETs:



A table is presented below with the answers to "How many SETs in n cards?".

Table 4: Number of SETs in n cards

n	Number of SETs
3	1
4	1
5	2
6	3

The reason we have not included higher numbers of cards, is that it will get very complicated.

5.5 Changing the rules of the game

When we address changing the rules of the game, we mean that we change the number of properties or the number of values within each property. Would it make the game more interesting? Would it make the game more complicated or less complicated?

Four properties and two values

Recall in the original game we have four properties and three values within each property. Let us try to keep the four properties, but eliminate one value. The total number of cards in the deck is therefore $2^4 = 16$ cards.

The concept of a SET stays the same, but will now consist of only two cards, and any two cards you pick from the total 16 cards will form a SET, since any two cards will either share values within the properties or the value will be different within each property. Even though the game is very trivial, we would still like to do some counting.

To find the total number of SETs we pick one card out of the total 16 cards, and the since any two cards forms a SET, we pick another card from the remaining 15 cards. We divide by the permutation of two elements, which is 2! and get the total number of SETs to be

$$\frac{16 \cdot 15}{2!} = 120 \ .$$

Four properties and four values

Let us try to add one value to the original game. Our choices for the new values is presented in the table below.

Table 5: Definition of values and properties with four values

Number	Filling	Color	Shape
1	Striped	Purple	Diamond
2	Full	Red	Worm
3	Hollow	Green	Oval
4	Dots	Pink	Star

The total number of cards in the deck is easy to calculate and is $4^4 = 256$ cards. But what is a SET? Suppose we still say that a SET is three cards, the fundamental theorem would not hold, since we would have another value to take into account. Now suppose a SET consists of four cards. We can pick any two cards and make a SET from those, but the third card has to be chosen wisely in order to make a SET of four cards.

In other words this game would first of all consist of a lot of cards, and second of all the game would be very complicated.

Five properties and three values

Imagine we create a similar game of SET, but instead of four properties we have five properties. A SET still consists of three cards and the proof for the fundamental theorem still holds. In fact the proof for the fundamental theorem still holds for p properties, as long as there is three values. The requirements for the three cards to be a valid SET is still the same as in the original game. The property of the cards must either be shared on each cards or be different on each card. The only difference is that the players have to look at five properties instead of four.

We decided to add background as the fifth property with the colors white, black and grey. The different values and properties is presented in the table below:

Table 6: Definition of values and properties with five properties

Notation	Number	Filling	Color	Shape	Background
0	3	Hollow	Green	Oval	White
1	1	Striped	Purple	Diamond	Black
2	2	Full	Red	Worm	Grey

The total number of cards in the deck is $3^5 = 243$. So the deck is three times bigger, than the original game. That means the game will take three times longer to play. Since a SET still consists of three cards, and since Theorem 1 still holds we want to do some counting.

The number of SETs is calculated using the general formula for number of SETs in a deck with $p = 5$ properties. Hence the number of SETs is

$$\frac{3^5 \cdot (3^5 - 1) \cdot 1}{3!} = 9801 \ .$$

This means the number of SETs in a deck with five properties is over nine times higher than the original game. This game would take a long time to finish, and having to take into account another property while searching for SETs might be exhausting. In conclusion we say that a game with five properties is not optimal as a game, because of the time to finish a game.

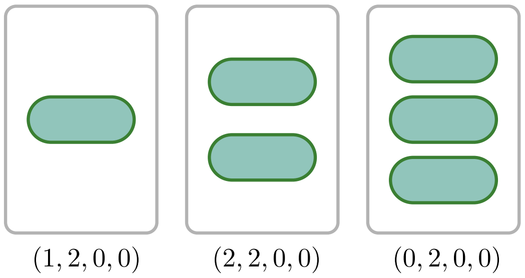
6 Modular arithmetic

In modular arithmetic we deal with remainders. The definition reads that $a \equiv b \bmod m$ if $a - b$ is divisible by m. It is a mathematical technique that involves calculating the remainder of a number when it is divided by another number called the modulus. For example $8 \equiv 2 \bmod 3$, since dividing 8 by 3 gives us a remainder of 2. Another notation for this is $8 = 2$ in $\mathbb{Z}_3$. Here $(\mathbb{Z}_3, +)$ is a group with the operation addition and the elements within the group is the numbers $\{0, 1, 2\}$.

6.1 Determine if a SET is valid using modular arithmetic

In the section "Notation" we used the notation $A = (a_1, a_2, a_3, a_4)$ for a card, where a_i can take the value 0, 1 or 2. If we take a closer look at SETs and try to add the respective values together, we can use this number to determine if it is a SET or not.

Given the cards presented below with the respective notations



we can calculate the sum of the respective values. We get

$$(6, 6, 0, 0) \equiv (0, 0, 0, 0) \bmod 3 \ .$$

Theorem 2: Three cards $A = (a_1, a_2, a_3, a_4)$, $B = (b_1, b_2, b_3, b_4)$ and $C = (c_1, c_2, c_3, c_4)$ form a SET if and only if $A + B + C \equiv (0, 0, 0, 0) \bmod 3$.

Proof. Table 7 clearly shows that taking the sum of each respective value in a SET will give $(0, 0, 0, 0)$ in $\mathbb{Z}_3$. For three cards to form a SET each property must be one of the four SET combinations from Table 7. These combinations all sum to 0 mod 3. Furthermore all the non-SET combinations are equivalent to 1 or 2. This concludes that all combinations where the sum of the three cards are $(0, 0, 0, 0)$ is a SET. $\square$

A table that supports Theorem 2 along with the proof is made by looking at the possible combinations of SETs and non-SETs:

Table 7: Combinations of SETs and non-SETs

SET combinations	Sum	Non-SET combinations	Sum
$\{0, 0, 0\}$	0	$\{0, 0, 1\}$	1
$\{1, 1, 1\}$	$3 \equiv 0$	$\{0, 1, 1\}$	2
$\{2, 2, 2\}$	$6 \equiv 0$	$\{0, 0, 2\}$	2
$\{0, 1, 2\}$	$3 \equiv 0$	$\{0, 2, 2\}$	$4 \equiv 1$
		$\{1, 1, 2\}$	$4 \equiv 1$
		$\{1, 2, 2\}$	$5 \equiv 2$

To verify that the table is complete, we look at all the combinations of SETs and non-SETs, we know that we have all combinations present, since the number of possible ways to place r elements n different places, when the ordering does not matter and repetition is allowed, is given by

$$\binom{n+r-1}{r} .$$

We have three different numbers $\{0,1,2\}$ we try to place three different places. This gives us

$$\binom{3+3-1}{3} = \binom{5}{3} = 10 .$$

Since there is 10 different combinations present in our table, we know we have all the possible combinations.

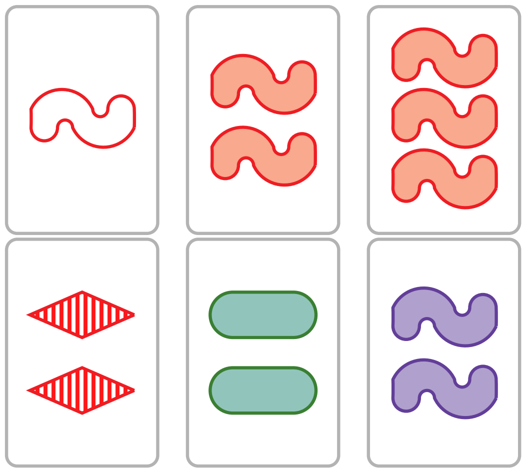
6.2 The End Game

Since it is possible to make SETs using all the cards in the stock, we know that the sum of all the cards has to be $(0,0,0,0)$ in $\mathbb{Z}_3$. This can easily be illustrated by arranging the cards such that they share all but one condition. Let us take the number of objects as the property that they do not share. Now we make SETs that only differ in the number of objects. By doing this we will have used up the entire deck. This concludes that the entire deck must sum to $(0,0,0,0)$ in $\mathbb{Z}_3$.

When playing the game we collect SETs, which we know have the sum of $(0,0,0,0)$ mod 3. This means that regardless of how many SETs we collect from the stock, the rest of the stock and the cards on the table will still have the sum of $(0,0,0,0)$ mod 3. In the end of the game the amount of remaining cards cannot be 3, since if there is 3 remaining cards on the table these would sum to $(0,0,0,0)$ mod 3, and would therefore form a SET, which we collect, and then we end up with zero cards. This gives that the amount of remaining cards is $3n$ for $n \geq 2$.

This means that in the end of the game we can determine the sum of all the remaining cards' respective values and if we do not get $(0,0,0,0)$ mod 3 we know that we have made a mistake while playing the game. If we get $(0,0,0,0)$ mod 3, most of the time, it will mean that we did not make any mistakes, but it is possible we made ≥ 2 mistakes during the game, but again very unlikely.

Here is an example of two non-SETs, where the mistakes cancel and it will not be possible to detect any error occurred during play:

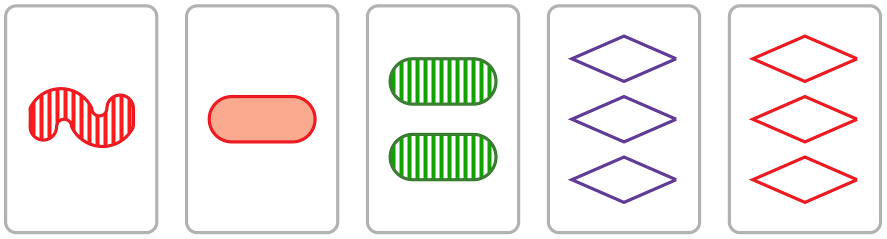


If we sum the values of the cards of each non-SET we get $(0,1,0,0)$ mod 3 for the first SET and $(0,2,0,0)$ mod 3 for the second SET. If we sum these SETs we get $(0,0,0,0)$ mod 3, and this means that these two mistakes cancel and it will not be possible to detect that any error occurred during play.

Hidden card

To make the game more interesting we start by placing a card to the side face-down without looking at it. Then the game is played as usual with 80 cards. In the end of the game the amount of remaining cards is $3n-1$ for $n \geq 1$, since we removed a card in the start of the game. Assuming no mistakes were made playing the game, we can determine exactly which card the hidden card is. If there is only two cards left a player can yell out "SET!" and grab the remaining two cards in addition to the hidden card. But if the endgame consists of more than two cards it is still possible to determine the hidden card. As explained earlier the remaining cards in a normal game sum up to $(0,0,0,0)$ mod 3. We can use this knowledge to determine the hidden card.

An example of an endgame with 5 cards is presented below.



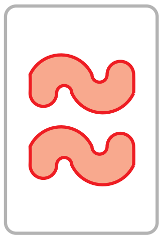
The first step of the process of finding the hidden card is to write down the notation of each of the remaining cards. In this case from left to right it is $(1,1,2,2)$, $(1,2,2,0)$, $(2,1,0,0)$, $(0,0,1,1)$ and $(0,0,2,1)$. Then we sum each respective value of the remaining cards and make the sum mod 3. For this example it becomes

$$(1,1,2,2)+(1,2,2,0)+(2,1,0,0)+(0,0,1,1)+(0,0,2,1)=(4,4,7,4)\equiv(1,1,1,1) \text{ mod } 3\ .$$

The hidden card has to satisfy

$$(1,1,1,1)+(a_1,a_2,a_3,a_4)=(0,0,0,0) \text{ mod } 3$$

which implies that our hidden card is the card with the notation $(2,2,2,2)$ (2 elements, **full** filling, red color and worm shape):



Another way to determine the Hidden Card, comes with a more visual approach where we look at each property separately. This technique involves determining each property of the Hidden Card, by looking at the remaining cards. The steps will be explained using the example above.

1. Step: **The number of objects:** Observe there is two cards with one object, two cards with three objects and only one cards with two objects. This concludes the hidden card consists of two objects.
2. Step: **The color:** The remaining cards consists of three red cards, one green and one purple. If we try to form SETs only looking at the color, we can conclude that the color of the hidden card is red.
3. Step: **The filling:** The filling of the remaining cards is two empty cards, two striped cards and one full card. This means that the hidden card has a full filling.
4. Step: **The shape:** We see that the number of shapes in the remaining cards is two diamond shaped cards, two oval shaped cards and one worm shaped card. This implies that the shape of the hidden card is worm.

Using this method we can conclude that the hidden card is two full red worms, which is the same result we got using the method involving modular arithmetic.

These procedures are valid for higher amounts of remaining cards as well. Here it is important to note, that it is possible that the hidden card can form a SET with two of the remaining cards.

6.3 Using vectors to find SETs

Recall earlier in this chapter we created notation for each property and their values written in modular arithmetic. We can use this notation to find the third unique card that complete a SET when having the first two cards with modular arithmetic. But we can also find this unique third card by using vectors:

Given two points P and Q in a vector space we can determine the vector between the points: $\vec{PQ} = Q - P$. Let the points P and Q be the first two cards in a SET. To determine the remaining card, W, to make these 3 cards form a SET we can find the midpoint:

$$\begin{aligned} W =& P + 2^{-1}\vec{PQ} \\ =& (p_1, p_2, p_3, p_4) + 2^{-1}(q_1 - p_1, q_2 - p_2, q_3 - p_3, q_4 - p_4) \\ =& 2^{-1}(p_1 + q_1, p_2 + q_2, p_3 + q_3, p_4 + q_4) \ . \end{aligned}$$

Since we are working modular 3, we need to determine what the inverse of 2, 2^{-1}, is modular 3. In other words we need to determine the number, in which you multiply by 2 and get 1 mod 3.

$$2 \cdot 2 = 4 \equiv 1 \text{ mod } 3 \text{ hence } 2^{-1} \equiv 2 \text{ mod } 3 \ .$$

Observe that 2 is then inverse of 2 mod 3. Knowing this we have the final formula

$$W = 2(p_1 + q_1, p_2 + q_2, p_3 + q_3, p_4 + q_4) = 2(P + Q) \ .$$

To prove that W is the unique card, which forms a SET with P and Q, we use Theorem 2, which tells us that P, Q, W forms a SET if and only if $P + Q + W \equiv 0$ mod 3. And since $W = 2(P + Q)$ then we have

$$P + Q + W = P + Q + 2(P + Q) = 3P + 3Q \equiv 0 \text{ mod } 3 \ .$$

Hence P, Q and W forms a SET.

Let us try to determine the last card, W, which forms the SET P, Q, W given two cards with the coordinates $P = (1, 2, 2, 1)$ and $Q = (1, 0, 2, 0)$:

$$W = 2(P + Q) = 2(1 + 1, 2 + 0, 2 + 2, 1 + 0) = 2(2, 2, 4, 1) = (4, 4, 8, 2) \equiv (1, 1, 2, 2) \bmod 3 .$$

Using Theorem 2, we check if P, Q, W forms a SET:

$$(1, 2, 2, 1) + (1, 0, 2, 0) + (1, 1, 2, 2) = (3, 3, 6, 3) \equiv (0, 0, 0, 0) \bmod 3 .$$

Hence P, Q and W forms a SET.

Note it does not depend on which choice of two cards we started with. So every point is "between" the other two. Like 3-points on a circular "line". We will make this precise in the next section.

7 Finite Affine Geometry

Finite affine geometry is a branch of geometry that generalizes Euclidean geometry. Instead of having infinitely many points we have a finite number of points. In Euclidean geometry a line is defined as having infinite number of points, where in finite geometry a line has a finite number of points.

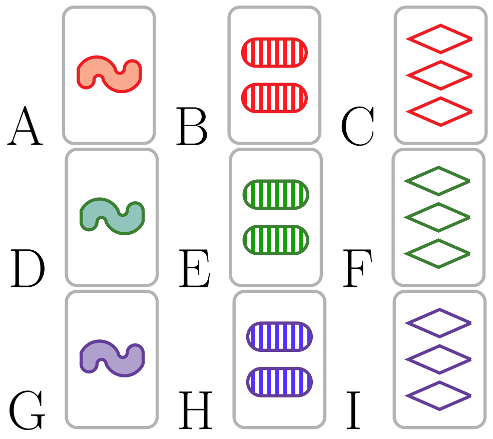
In this chapter we will talk about planes and cubes and how these can relate to the game of SET. We will also further discuss how the four axioms of finite affine plane geometry relate to SET, and introduce a new term, called parallel SETs.

7.1 Lines and planes

Lines and planes are fundamental concepts in geometry, and the card game SET provides a fun and engaging way to explore some of the basic concepts of geometry. The card game SET is related to geometry since the game involves certain geometric properties. Geometry is a branch of mathematics that is concerned with the study of points, lines, angles and planes. It describes a point as zero-dimensional, a line as one dimensional and a plane as a two-dimensional surface.

We observe that a valid SET of three cards resembles three points as a straight line in four-dimensional space. The number of dimensions comes from the number of properties, which also corresponds to the number of coordinates. In the game SET there is four properties, therefore a SET can be said to be a line with three points in a four dimensional space. A valid SET will always be a straight line. That means when we have three cards that do not form a SET, the non-SET does not create a line. We know that it always is possible to create a line from two points. From the fundamental theorem we know that, given two cards, there is a unique card which forms a SET with the two given cards. In other words these two points, or cards, determine a distinct line, or a distinct SET. If we have three cards, that do not form a SET, we can create a plane of cards. A plane of cards in SET, is defined as having nine cards in a square, where all possible SETs are present. If we form a SET from two random cards from the plane, the last card which makes it a SET exists in the plane.

Here is a example of a plane of cards in SET:



In each plane there exists 12 SETs. Some are more obvious than others. We can split the SETs up into vertical, horizontal, increasing diagonal and decreasing diagonal SETs. The horizontal SETs consists of $\{A, B, C\}$, $\{D, E, F\}$ and $\{G, H, I\}$. The vertical SETs are $\{A, D, G\}$, $\{B, E, H\}$ and $\{C, F, I\}$. If we start from the bottom the increasing diagonal SETs are $\{G, E, C\}$, $\{H, F, A\}$ and

$\{I, D, B\}$. Starting from the top we get the decreasing diagonal SETs to be $\{A, E, I\}$, $\{B, F, G\}$ and $\{C, D, H\}$.

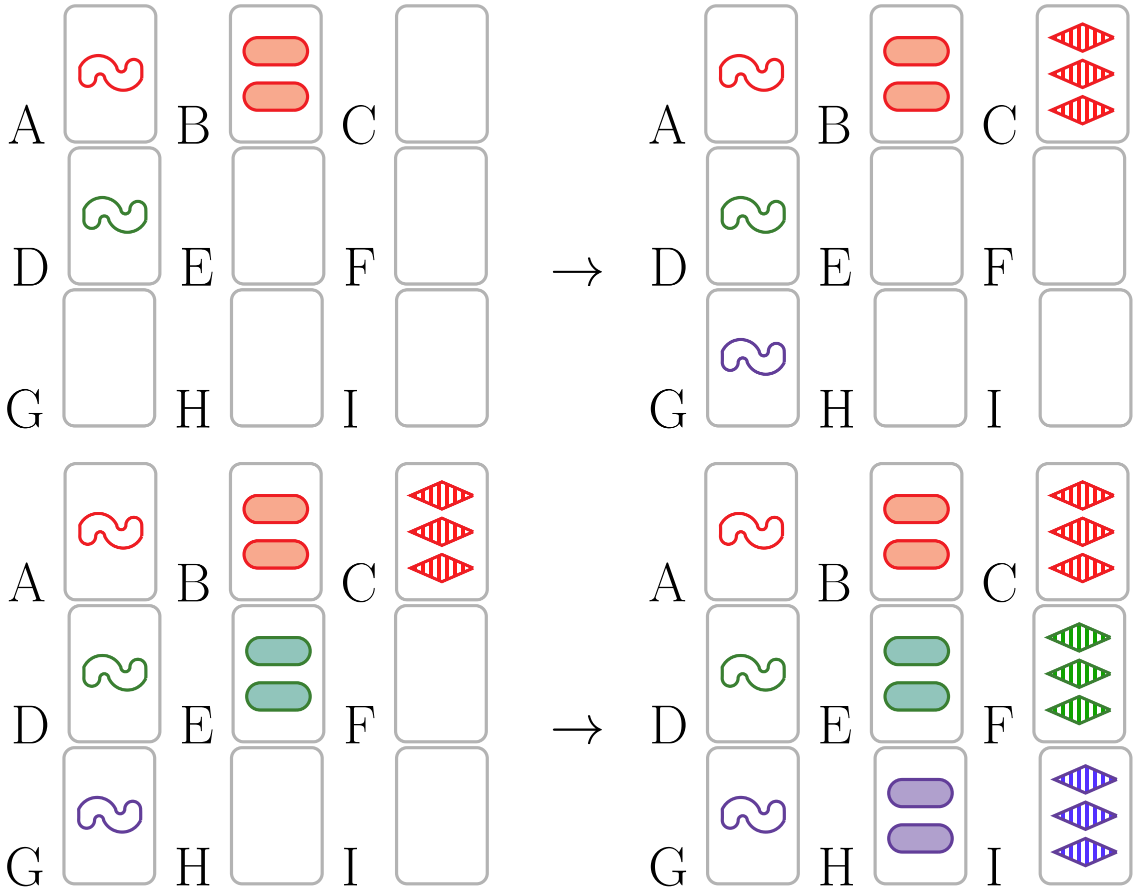
If we look at the theoretical maximum number of SETs in 9 cards, we use the formula from the "Counting" section.

$$\frac{9(9-1)\cdot 1}{3!} = 12\ .$$

Since there is 12 SETs in a plane of cards all possible SETs must exist in the plane. In other words a plane is "closed" under the operation of completing SETs.

Theorem 3: Given any three cards that do not form a valid SET, there exist a unique plane containing them.

Proof. Given three cards that do not form a SET, A, B and D. The fundamental theorem is used to determine the rest of the plane. An example of this procedure is presented below.



We chose two cards on a vertical, horizontal or diagonal line and find the last unique card using the fundamental theorem. In this case we chose A and B to find C and A and D to find G. Then we use C and G to find E. From this point there is various of ways to find the last remaining three cards. An example of this is using D and E to find F and B and E to find H and lastly we can use G and H to find I. From this the six remaining cards are determined by the original three cards, and we have a plane of nine cards with 12 SETs.

This plane is unique, since the fundamental theorem states that only a unique card can form a SET with two given cards.

$\square$

In the case that A, B and D forms a SET, then if we try to make a plane using these three cards, we come to a contradiction, since A, B, C must form a SET, so $D = C$, which is not possible since there only exists one of each card in the deck.

Let us try to count the number of planes in the deck. First we need to determine how many ways we can choose three cards that do not form a valid SET. First we pick two random cards, and ways to pick these two cards is $81 \cdot 80$. Then Theorem 1 tells us that there exists only one unique card that forms a SET with the two first cards. We cannot choose this specific card to be the third card for constructing the plane, but we can take any other card from the last 78 cards left in the stock. the different ways to pick out three cards that do not form a SET is:

$$81 \cdot 80 \cdot 78 = 505440 \ .$$

This number is the number of ways to make a plane if the ordering is taken into account. In SET the ordering does not matter, so we need to find the number of planes where the ordering is irrelevant: The first card can be placed nine different places, and the second card can be placed eight different places. As described earlier the third card cannot complete a SET with the first two cards. Therefore it cannot be placed in a line with the two other cards, but anywhere else in the plane. So the third card can be placed six different places. The number of planes in the deck is therefore:

$$\frac{81 \cdot 80 \cdot 78}{9 \cdot 8 \cdot 6} = 1170 \ .$$

7.2 Finite affine plane geometry

Finite affine plane geometry has four axioms:

Axiom 1. There are at least 3 non-collinear points (non-collinear simply means not on the same line).
Axiom 2. Every line has at least two points.
Axiom 3. Two distinct points determine a unique line.
Axiom 4. For every line l and point $P \notin l$ there is exactly one line l' such that $P \in l'$ and $l \cap l' = \phi$. l' is said to be parallel to l.

Axiom 1 and 2 are quite simple, and exist for the reason to ensure the plane geometry is not trivial. When there exist three non-collinear points we can get more dimension in the plane than just points and lines. Axiom 2 states that we need at least two points to create a line.

Axiom 3 is fundamental in finite affine plane geometry in that case that it ensures that there exist a unique line between any two points.

Axiom 4 is known as the parallel postulate in Euclidean geometry. It is important to distinguish between parallel lines and lines that do not intersect, because in higher dimensional space not all lines that do not intersect are parallel. Skew lines is one example of lines that do not intersect but are not parallel to each other.

Finite Affine Plane Geometry in SET

These axioms turn out to translate well into the different rules of the card game SET. As mentioned the points and lines in affine geometry can represent different things, and in this case the points and lines can be really good illustrators for the cards and SETs in the game. We can see points as cards, and lines of three points as a SET. The four axioms from finite affine geometry regenerated to the rules of SET is:

Axiom 1. There are at least three cards that are not in the same SET.
Axiom 2. Every SET has at least two cards.
Axiom 3. Two cards determine a unique SET (the fundamental theorem).
Axiom 4. For any SET and any card not in the SET, there exists only one SET containing this card, which is parallel to the first SET.

Axiom 1 and 2 are self-explanatory. We can easily find three cards that does not form a SET, we simply find a SET, and substitute one of the cards with any other card, and the three cards will not form a SET. Furthermore we know that a SET consists of three cards.

Axiom 3 is explained with the fundamental theorem of SET-there exist a unique third card for any given two cards that makes it a SET.

Axiom 4 tells us that for a card not in a SET there exists a unique parallel SET containing this card. In the following section we will look further into what parallel SETs are.

7.3 Parallel SETs and direction vectors

Lets disregard from the game just for a second to talk about the term parallel in finite euclidean geometry. Two lines are said to be parallel if they are in the same plane but do not have any points that intersect. In other words: two lines is said to be parallel if they share a direction vector.

The direction vector in the card game SET is something worth discussing. We will use vector translation to define parallel SETs and show how to use vector calculations to find SETs and parallel SETs.

Interlude: Cycles and sequences

In our attempt to define what parallel SETs are, we need to talk about **cycles**. A cycle of objects is a sequence placed in a circular order and in mathematics we write $(0, 1, 2)$ for the cycle of the numbers 0, 1 and 2. This cycle is equivalent to the cycles $(1, 2, 0)$ and $(2, 0, 1)$. It is possible to write another cycle of the same numbers, which is the counterclockwise sequence of the numbers. For this instance it becomes $(0, 2, 1) \equiv (2, 1, 0) \equiv (1, 0, 2)$. This means there is two different cycles of three numbers.

Here is an illustration of two different cycles. The left one goes clockwise, and the right one goes counterclockwise.
The importance of talking about cycles comes in relation to parallel SETs, which we will cover following this.

When we talk about parallel SETs we have the following conditions:

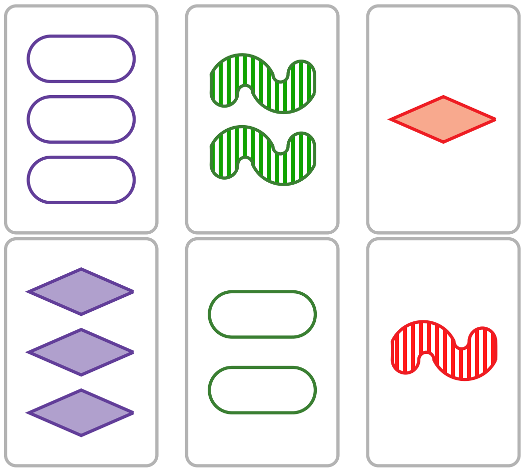
- If a value is the same within a property in one SET then the other SET also has the same value within the same property. For instance, if the first SET contains only green cards, then the other SET contains only cards of the same color.

$$\begin{tikzcd}
& 0 \arrow[dr, bend left] & & & 0 \arrow[dl, bend right] & \\
2 \arrow[ur, bend left] & & 1 \arrow[ll, bend left] & 2 \arrow[rr, bend right] & & 1 \arrow[ul, bend right]
\end{tikzcd}$$

- If a value is all different in one SET then the value is all different in the other SET. For instance, if the first SET contains the number of elements 1, 2 and 3, then the other SET also contains these number of elements. Furthermore if the cycling order of the value in the SETs is the same within one property, then it is the the same cycle of values within the other properties. For instance, if the first SET cycles the number of elements $(1, 2, 3)$ and the color (red, green, purple), then the other SET cycles these two properties the same way.

Notice that in some cases these cycles can be different, but as long as the cycles within each property of each SET is the same, the two SETs are parallel.

Given the two SETs:



In order to look at the cycle of each property within each SET, we write each SET as the cycle of each coordinate. The first SET cycle is

$$\begin{array}{cccc} \text{Number} & \text{Filling} & \text{Color} & \text{Shape} \end{array}$$

$$[(0, 2, 1), (0, 1, 2), (1, 0, 2), (0, 2, 1)]$$

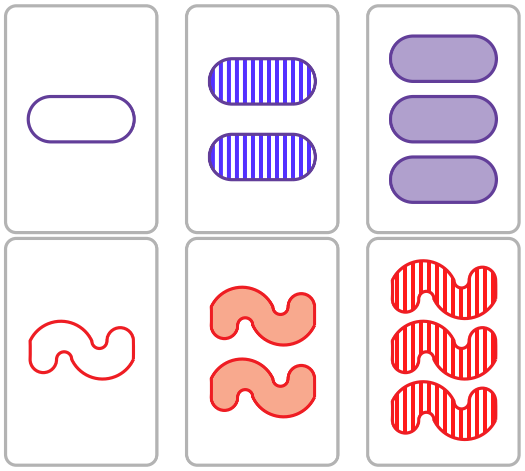
And the cycle of each property within the second SET is

$$\begin{array}{cccc} \text{Number} & \text{Filling} & \text{Color} & \text{Shape} \end{array}$$

$$[(0, 2, 1), (2, 0, 1), (1, 0, 2), (1, 0, 2)]$$

The cycle of each property within each SET is the same, since the cycles $(0, 1, 2) \equiv (2, 0, 1)$ and $(0, 2, 1) \equiv (1, 0, 2)$. Hence these two SETs are parallel SETs.

Given two SETs:



We want to check if these two SETs are parallel, so we do the same procedure as before; we look at the cycle of the values within each property of the first SET:

$$\begin{array}{cccc} \text{Number} & \text{Filling} & \text{Color} & \text{Shape} \end{array}$$

$$[(1,2,0),(0,1,2),(1,1,1),(0,0,0)]\ .$$

And the cycle of the values within each property of the second SET:

$$\begin{array}{cccc} \text{Number} & \text{Filling} & \text{Color} & \text{Shape} \end{array}$$

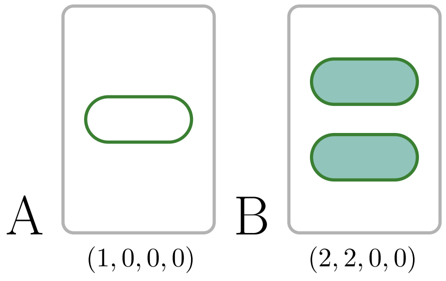
$$[(1,2,0),(0,2,1),(2,2,2),(2,2,2)]\ .$$

The cycle of the number is the same, but the cycle of the filling is different $(0,1,2) \not\equiv (0,2,1)$, which is why these two SETs are not parallel SETs.

These conditions are visual and easy to check if you are presented with two SETs. But as all mathematicians we will like to check if two SETs are parallel or find parallel SETs using calculations and mathematics. This can be done using vector calculations, and presenting the cards with their respective coordinates.

In vector space we know that the the direction vector, $\vec{v}$ of a line is $\vec{v} = B - A$ where the points A and B is on the line. This translates well into SET, since we can represent the cards with coordinates.

Given the two cards



we can calculate the direction vector of the cards

$$\vec{v} = B - A = (2,2,0,0) - (1,0,0,0) = (1,2,0,0) \ .$$

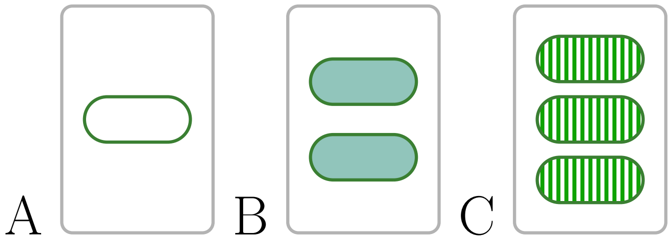
We can use this to determine the last card, C:

$$A + \vec{v} = B, \quad A + 2\vec{v} = C \ .$$

In this example it gives us the last card, C to be

$$C = (1,0,0,0) + 2(1,2,0,0) = (0,1,0,0) \text{ mod } 3 \ .$$

This gives us the last card, C, which makes A, B and C a SET.



From the equation above we can show that given a card A and direction vector $\vec{v}$, then A, $A + \vec{v}$, $A + 2\vec{v}$ form a SET with direction $\vec{v}$.

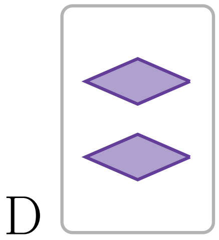
$$A + B + C = A + A + \vec{v} + A + 2\vec{v} = 3A + 3\vec{v} = 0 \text{ mod } 3 \ .$$

Theorem 4: Two SETs are parallel if and only if their corresponding direction vectors are a multiple of each other.

Proof. Look at page 202 in [4]. □

From this we can try to create a parallel SET to our previous SET, (A, B, C) by picking a random card, D, and using the direction vector to find the other two cards which leaves us with two parallel SETs.

Picking the card, D, with the coordinates $(2,2,1,1)$.

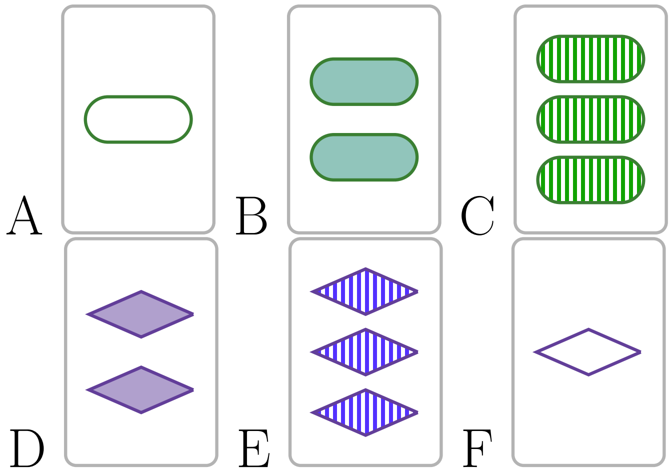


Now we find a parallel SET by using the direction vector $\vec{v}$:

$$E = D + \vec{v} = (2,2,1,1) + (1,2,0,0) = (0,1,1,1) \text{ in } \mathbb{Z}_3 \ .$$

$$F = D + 2\vec{v} = (2,2,1,1) + (2,1,0,0) = (1,0,1,1) \text{ in } \mathbb{Z}_3 \ .$$

This gives us the two parallel SETs A, B, C and D, E, F:



Regenerating this term of parallel into the card game SET, we can talk about parallel SETs. Using what we have learned from the fourth section about modular arithmetic, we know that when a SET is valid we get $(0, 0, 0, 0)$ in mod 3 when adding the three cards together. This means the three cards/points lays in the same line. Having two of these lines that do not intersect, its said that the lines are parallel to each other. The different planes we can create, will reveal which SETs that are parallel to each other.

Its also possible to see which SETs are parallel simply by pattern recognition.

7.4 Cube

In this thesis we have talked about one and two dimensions as SETs $\mathbb{Z}_3^1$ and as planes $\mathbb{Z}_3^2$. But can we talk about finite affine geometry in one dimension higher, i.e. $\mathbb{Z}_3^3$? Moving on from planes that consists of nine cards in a 3x3 square, we can now create a cube from three planes laying on top each other. The form of a cube will therefore be a three dimensional cube with cards laying in a 3x3x3 shape. This means that a cube consists of 27 cards.

SETs in a cube

To find out how many SETS we can find in the 27 cards in a cube we can use the formula we already created in section 6.5:

$$\frac{n \cdot (n-1) \cdot 1}{3!} \qquad \Longrightarrow \qquad \frac{27 \cdot (27-1) \cdot 1}{3!} = 117 \; .$$

This means its nearly 10 times more SETs in a cube than in a plane. This is because in addition to the $12 \cdot 3$ SETs that exists in the three planes alone, we can find SETs all different ways horizontal, vertical and diagonal in the cube. So instead of using the formula we created to find the total number of sets, we can also simply count them:

First we have the $12 \cdot 3$ SETs from the three planes = 36 SETs.

The other option we have is to create SETs with one card from each of the three planes laying on top of each other. Lets remind us what Theorem 1 said: “To create a SET from two cards there exists a unique third card which makes it a SET”. This means we can take any card from plane one and plane two, and the card we have to pick from plane three to form a SET is determined by the first two cards. From plane one we can pick between nine different cards, and the same applies from plane two. The total number of this kind of SETs are therefore: $9 \cdot 9 \cdot 1 = 81$.

Hence the total number of SETs in a cube is $36 + 81 = 117$.

Planes in a cube

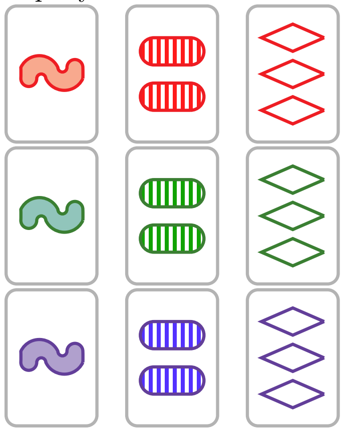
Lets find out how many planes there exists in a cube. We already know about the three planes that lays horizontal on top of each other.

The easiest way to find the remaining planes in a cube is to use what we have leaned about parallel SETs. First we choose any valid SET we want from the cube, lets choose one SET from the top layer. Then we have to find another SET that is parallel to the first SET from the middle layer. These two SETs from the top and middle layer will determine which third SET we have to pick from the bottom layer. This SET will also be parallel to the first two. We can first choose between 12 different SETs from the top layer. Further there will only be three SETs in the middle layer that will be parallel to the first SET we chose, since there is three SETs of each type in each plane, i.e. horizontal, vertical, increasing diagonal and decreasing diagonal SETs. One SET in the bottom layer will be parallel to the two SETs that we choose. The total number of this kind of planes in a cube is therefore $12 \cdot 3 \cdot 1 = 36$.

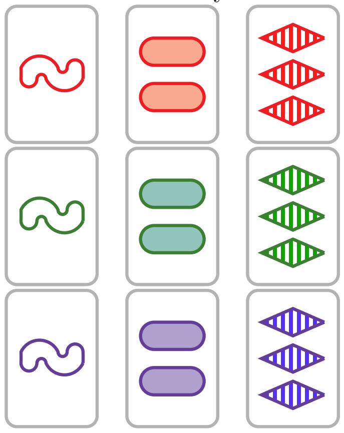
Hence the total number of planes in a cube is $3 + 36 = 39$.

In the following page we have presented an example of a cube of SETs to easier visualize the different SETs and planes in a cube.

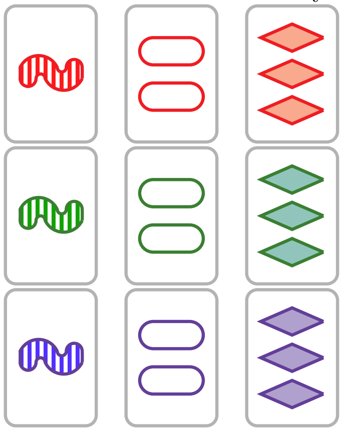
Top layer



Middle layer



Bottom layer



Can we take it one more step and talk about the card game SET in one dimension higher than a cube? This would be four dimensional which would be the whole deck, and we have already discussed its mathematics in the Section "Counting in SET".

8 Simulations and coding in SET

In this section we describe the four classes of simulations we have carried out and for each of them some conclusions. Then, important elements from the codebase behind the simulations are briefly described. But before that, we need to answer the questions of why we do simulations.

The card game SET has shown to have many complicated mathematical features and probabilities. In this section we are going to talk about how we can use programming and coding to determine (with high accuracy) probability of certain events happening in the game.

8.0.1 The necessity of simulations

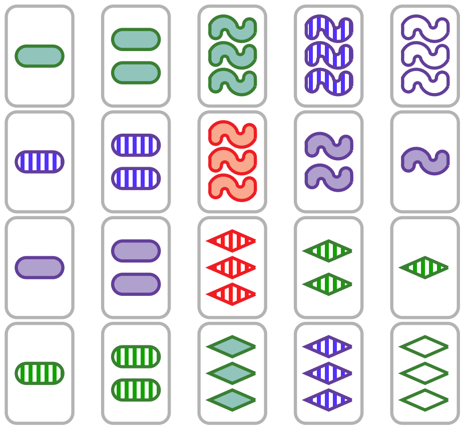
Simulations are necessary in this context, because of the large number of combinations that can be involved when we want to answer questions, such as "What is the smallest n, such that every subset of n cards contains a SET?". In theory, a computer could be used to calculate this, but since the number of subsets of the 81 cards is extremely large, this is not practically possible. A small practical example of this; suppose we want to examine whether all subsets with 21 cards contain a SET. An algorithm for this could be that for each subset, we will examine whether it contains a SET or not. The problem here is that there are $\binom{81}{21} = 1,36 \cdot 10^{19}$ subsets of size 21. With a modern computer, it takes about two minutes to examine approximately 1.000.000 subsets. It then gives approximately $4,5 \cdot 10^{11}$ hours to examine all subsets. This corresponds to approximately 50 million years. With such large numbers, it will not be of any help to use more powerful or several computers.

8.1 Simulations

Before describing our simulations we will introduce a terminology for whether a set of cards contains a SET. We will use the term *reducible* for the fact that a set of cards contains a SET. A set of cards that do not contain a SET is called non-reducible or irreducible.

Simulation can be a shortcut to get answers to essential questions like "What is the smallest n such that any subset with n cards is reducible?". The weakness of the simulations we have run is that they correspond to taking random samples. This clearly weakens the truth value of the conclusions we can draw from simulations. For example, our simulations show that all subsets of 18 cards will contain a SET. We have examined well over 100.000.000 subsets of size 18 and found that they all contained a SET. But we know that there are sets of 20 cards that do not contain a SET [3].

A set of 20 cards with no SETs is presented below:



The three simulations we have completed are designed to answer the following questions:

1. If you simulate the game, how many cards are left when the game ends, i.e. when the remaining amount of cards is non-reducible?
2. What is the probability that a randomly chosen set of n cards is non-reducible?
3. How many SETs can be formed from a subset of n cards?

All three questions have been attempted to be elucidated through simulations. These are described below.

8.1.1 Number of cards in the end game

As described we simulated the game, and looked at the remaining cards at the end of the game. Here is the result of the simulation, presented as the number of total cases that ended up with the respective number of cards along with the probability:

Table 8: Number of cards in the end game

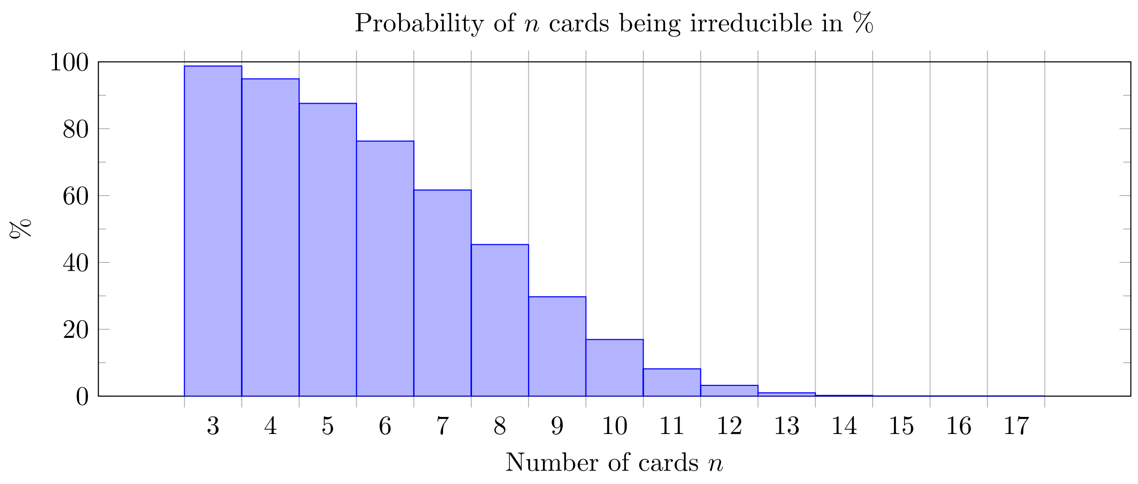
Number of simulations	0 cards	6 cards	9 cards	12 cards	15 cards	18 cards
$50 \cdot 10^6$	765.970	$24,5 \cdot 10^6$	$21,3 \cdot 10^6$	$3.4 \cdot 10^6$	34060	21
Probability	$1,5\%$	$49,0\%$	$42,6\%$	$6,8\%$	$0,07\%$	$4,2 \cdot 10^{-7}\%$

In these simulations, we start each game with all 81 cards. Then the game is played normally and SETs are formed randomly. When the cards left at the end of the game cannot be further reduced, this run is over and the result is how many cards are left.

As shown earlier in the thesis it is not possible to end up with 3 cards in the end of the game, which is why 3 cards is not in the table. It is clear that ending up with 6 or 9 cards is by far the most common, while ending up with 18 cards is very unlikely. Next time you play a game of SET, you now have the knowledge of the probability of ending up with a certain amount of cards, and when you come to the end of the game, you know how likely or unlikely your scenario is.

8.1.2 Distribution of reducible SETs

This simulation deals with finding an estimate of the probability that a randomly chosen set of n cards is irreducible. In other words, what is the probability that n cards does not form at least one SET. The results are presented in a histogram where the x-axis present the number of cards n and the y-axis present the probability in percentages.



In order to see the exact values, a table is created with our results for this simulation.

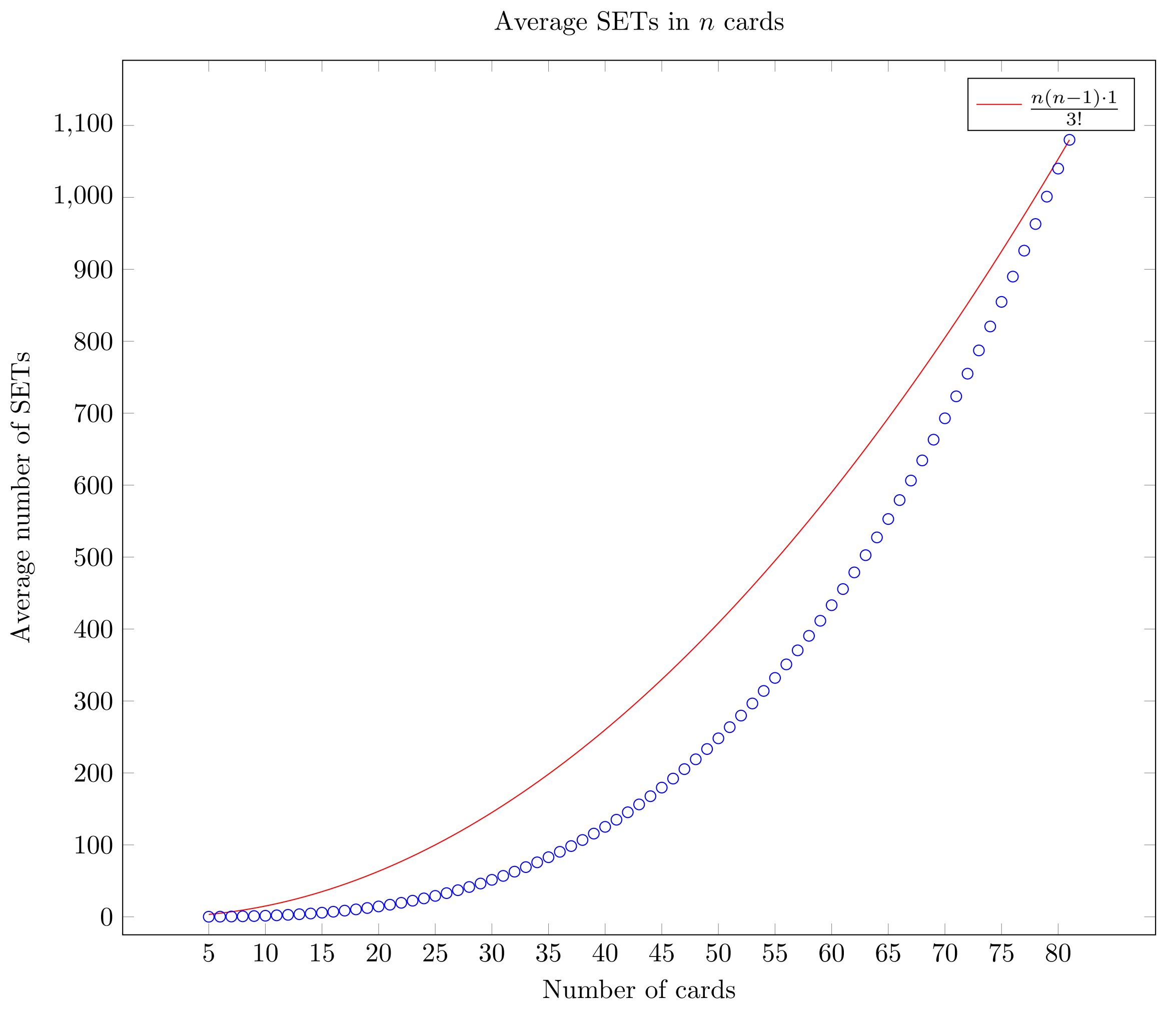
Table 9: Probabilty of n cards not having a SET

Number of cards n	Probability of no SET
3	$98,7\%$
4	$94,9\%$
5	$87,6\%$
6	$76,3\%$
7	$61,7\%$
8	$45,4\%$
9	$29,7\%$
10	$16,7\%$
11	$8,2\%$
12	$3,2\%$
13	$1,0\%$
14	$0,2\%$
15	$0,04\%$
16	$3,3 \cdot 10^{-3}\%$
17	$1,4 \cdot 10^{-4}\%$

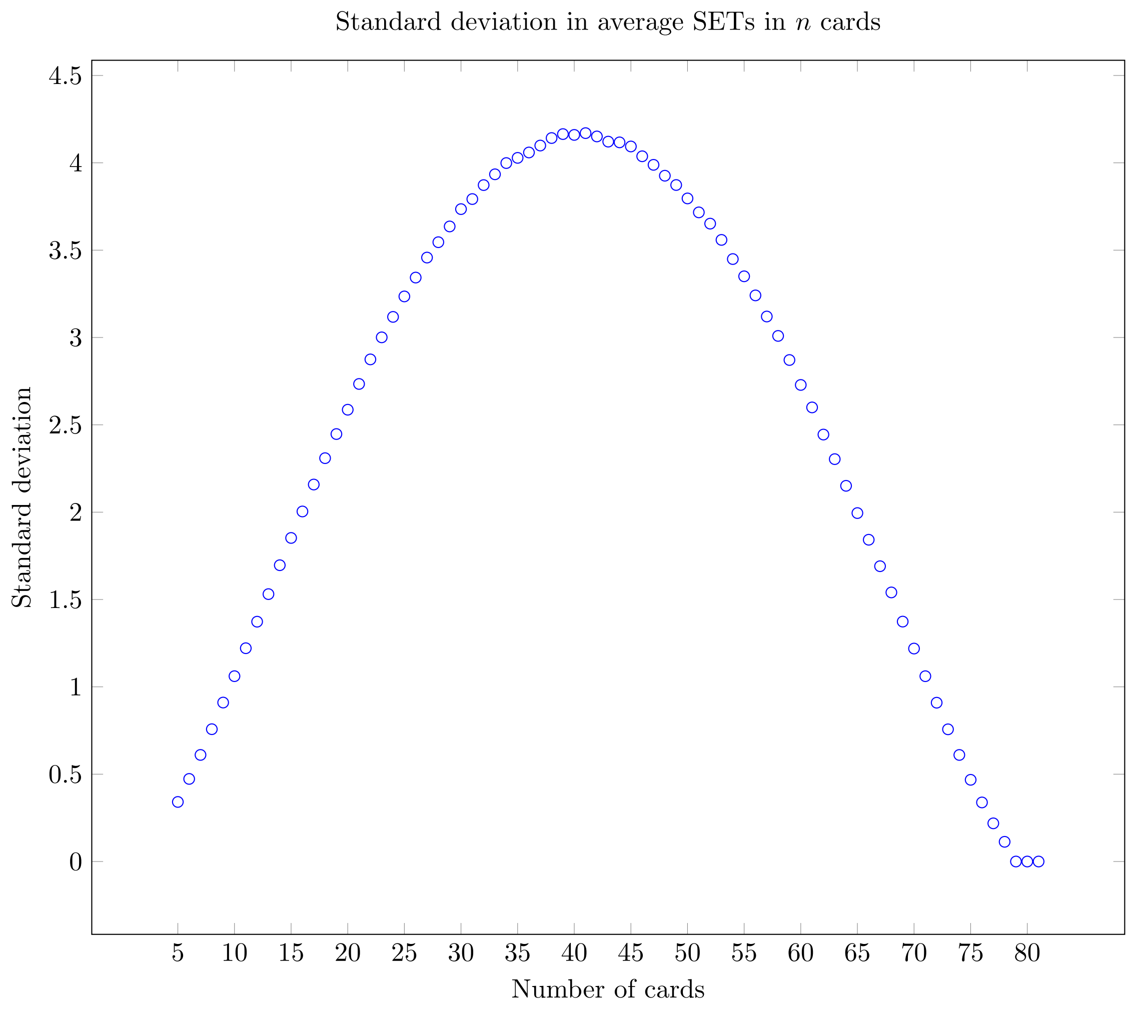
For each n between 3 and 18, the simulation consisted of 30.000.000 random subsets of n cards that we tried to reduce. The percentage shows the proportion of non-reducible quantities. From the result we can estimate that if you choose 10 random cards, the probability that it does not contain any SET is $16,7\%$. In the simulations, we found no non-reducible sets with more than 17 cards. Since it is a simulation, however, we cannot infer that all sets with at least 18 cards contain a SET and, as mentioned earlier, we know that there exists non-reducible sets of 20 cards.

8.1.3 SETs in n cards and probability

The point of our last simulation was to estimate the number of SETs in n cards. For each n, we have found 100.000 random sets of n cards and computed the number of SETs. The first graph shows the average number of SETs in n cards, and the second graph shows the standard deviation as a function of n cards.



The graph shows that the number of average SETs drastically increases with the number of cards. In theory this makes sense, since adding a card will drastically increase the number of average SETs. The red line shows the theoretical maximum number of SETs existing in n cards. As expected the red line is above the blue points. But at the start and at the end of the graph the two graphs meet. This implies that the theoretical maximum number of SETs from n cards is the same as the average number of SETs in n cards, when $n \leq 2$ and $n = 81$. This makes sense, since for $0, 1$ and 2 cards we will never be able to form a SET. And in the whole deck there is exactly 1080 SETs, which the data shows.



The standard deviation of average SETs in n cards follows a normal distribution, as shown from the graph. The standard deviation tells us how precise the average of SETs in n cards is. In other words, the standard deviation peaks when we use half the deck of cards, and is at the lowest when $n \leq 2$ and $n \geq 79$.

8.2 Programming in Python

In this section, the most important elements of the code base for the three simulations above are described.

Data representation of a card

A card is represented as a string of 4 characters, where each character is '0', '1' or '2'. Each of the four characters corresponds to a shape, a color, a number or a pattern. Since there are 3 shapes, 3 colors, 3 numbers (1, 2 or 3) and 3 fillings, each character must have one out of 3 possible values. Here 0, 1 and 2 are selected, but it is not important and it is only used where the output from parts of the simulation is to be used directly in LaTeX.

Structure of Code Base

The code base is structured into three layers: the bottom layer contains basic set functionality such as calculating how many SETs a given set of cards contains. This layer is called SetLogic. The next level contains functionality called Controller. It contains functionality for performing the essential simulations and returning the result as data. Finally, we have the app level that is doing the overall simulation by setting up parameters, and writing the result to output. The mapping between layers and Python files is:

Table 10: Structure of code base

Layer	Files
App	main1.py main2.py main3.py
Controller	simulation1.py simulation2.py simulation3.py
SetLogic	setfunc.py

In the following section the code for selected functions in the SetLogic layer is described.

Code for SET Logic

The rule for whether 3 cards constitute a SET is that for all 4 character positions the characters on the 3 cards must either be the same or all different. The following function can decide whether or not three cards is a set:

```python
# return true iff a, b and c is a set.
def isSet(a,b,c):
    for i in range(4):
        if a[i] == b[i] and a[i] == c[i]:
            continue
        if a[i] != b[i] and a[i] != c[i] and b[i] != c[i]:
            continue
        return False
    return True
```

This function can be used to compute all the SETs contained in a set of cards. The function used for our results is:

```python
# return a list of sets from data with replacement.
def ComputeAllSetsWithReplacement(data):
    n = len(data)
    result = []
    for i in range(n):
        for j in range(i+1,n):
            for k in range(j+1,n):
                if isSet(data[i], data[j], data[k]):
                    result.append([data[i], data[j], data[k]])
    return result
```

A variant of this is to find SETs in a set of cards where a given card can only be used in one SET. The code for this is:

```python
# return a list of sets from data without replacement.
def ComputeAllSets(data):
    n = len(data)
    used = []
    result = []
    for i in range(n):
        for j in range(i+1,n):
            for k in range(j+1,n):
                if used.count(data[i]) == 0 and used.count(data[j]) == 0 and used.count(data[k]) == 0:
                    if isSet(data[i], data[j], data[k]):
                        result.append([data[i], data[j], data[k]])
                        used.append(data[i])
                        used.append(data[j])
                        used.append(data[k])
    return result
```

The last function to be explained here is the function that calculates an array with all 81 cards. This will be used in all three simulations. Here we could just have hard-coded the cards in the code, but we have done it in a much more generic way:

```python
# Will add all cards/strings of length 4 to container
def GenerateAllCards(prefix, container):
    if (len(prefix) < 4):
        for ch in symbols:
            GenerateAllCards(prefix + str(ch), container)
    else:
        container.append(prefix)
```

The implementation of the function illustrates the power of recursion, because it can easily be used if we add another property or change the number of different values a given property can have. The use of the function can be illustrated by showing some code for the controller for the third simulation (What is the number of SETs in n cards?):

```python
# perform one simulation with m random cards. Return
# the numbers of sets that can be drawn with replacement
# from the cards
# can be done.
def OneSimulation(m):
    data = []
    GenerateAllCards("", data)
    random.shuffle(data)
    subset = data[slice(m)]
    sets = ComputeAllSetsWithReplacement(subset)
    return len(sets)
```

The entire code is displayed in “Appendix”.

8.3 Probability of a plane in n cards

In this subsection we will describe our last and fourth simulation, in which we have used simulations to estimate the probability that n random cards contain a plane. The essential functionality of such a simulation is a function that, given a set of cards as a parameter, can decide whether or not the set contains a plane.

Our first approach was to use a kind of brute-force algorithm. We go through all possible subsets of size 9, and for each of them count the number of sets, and if that turns out to be 12, then we have a plane. This approach is not practical due to the enormous computation time.

The second approach is based on the algorithm described in the section "Lines and planes". Recall the algorithm consists of four steps:

1. **Step:** Find three cards that do not constitute a SET.
2. **Step:** Calculate two of the cards using two of the original cards and check that these are in the set of n cards.
3. **Step:** Calculate the card in the middle based on the cards calculated from step 2 and check to see if that card is present.
4. **Step:** Calculate the remaining three cards and check that they are present in the set of n cards.

If some check fails, then the algorithm will continue with step one to find the next three cards that do not form a SET. This continues until the algorithm has used up all possible combinations of three cards not being a SET. This algorithm is much more effective and solved the problems related to computation time. The result obtained is presented in a graph along with a table.

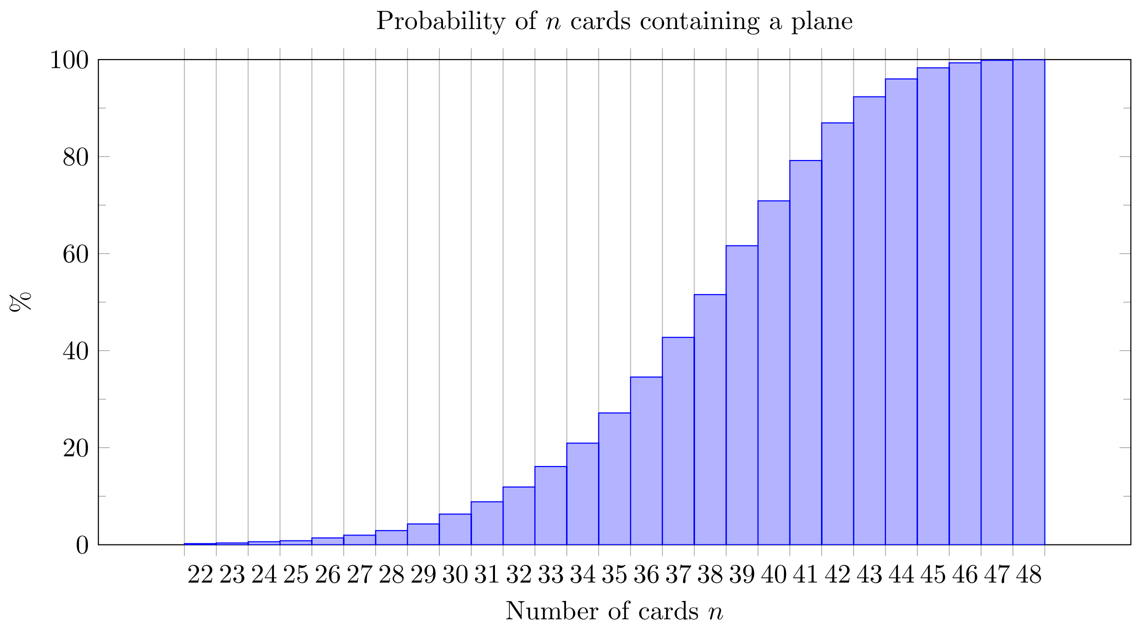


Table 11: Probabilty of n cards containing a plane

Number of cards n	Probability of a plane in %
16	0.004
17	0.015
18	0.025
19	0.055
20	0.072
21	0.136
22	0.228
23	0.366
24	0.62
25	0.83
26	1.406
27	1.968
28	2.918
29	4.284
30	6.314
31	8.86
32	11.9
33	16.124
34	20.936
35	27.162
36	34.562
37	42.738
38	51.56
39	61.634
40	70.882
41	79.194
42	86.952
43	92.334
44	96.01
45	98.3
46	99.332
47	99.854
48	99.964
49	100.0

The graph shows that the probability of n cards containing a SET increases as the number of cards n increases. In the simulation we ran 100.000 simulations for each n, and did not find any planes in 15 cards or less, although we know that it is possible to find a plane in 9 cards, since a plane consists of exactly 9 cards.

The probability of a plane in 9 cards is

$$\frac{1170}{\binom{81}{9}} \approx 4,5 \cdot 10^{-11}\%$$

since there is a total of 1170 planes in the deck (from the section "Finite Affine Geometry"), and the number of different ways to pick 9 cards out of 81 cards is $\binom{81}{9}$.

All our simulations found a plane in $n > 48$ cards, so it seems that for ≥ 49 cards the set of cards will always contain a plane. Furthermore you would need ≥ 38 cards to have more than 50% probability to find a plane.

9 Using SET in a classroom

The popular card game, SET, is more than just a source of fun and entertainment for families and friends worldwide. It has also proven to be an effective tool for educators in teaching math concepts to students. By playing SET, students can better understand the different branches of math, making it an excellent addition to any math class. Apart from its educational benefits, SET also improves a lot of skills that are essential in everyday life, not just in school. One such skill is problem-solving, which is a fundamental skill that can be applied in various real-world scenarios. Because SET is easy to learn and captivating, it catches the attention of most people, making it an excellent tool for motivation and engagement in learning mathematics. Thus, incorporating SET into your classroom can be one of the best decisions you make as a teacher. It not only makes math lessons more enjoyable, but it also improves students problem-solving skills and encourages their interest in math. This is a guide for how to include SET in your classroom in the best way possible.

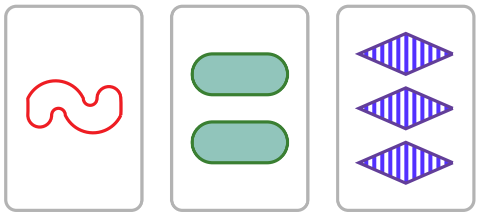
9.1 What SET is and how to play

We assume that you as a reader are familiar with the basic features and rules of the game, but here is a short recap just to be sure:

The card game SET is a fast-paced pattern recognition card game consisting of 81 cards. Each card has a combination of four different properties, each taking one of three possible values:

Number: One, Two, Three
Color: Green, Purple, Red
Shape: Oval, Diamond, Worm
Filling: Empty, Striped, Full

Here is one example of a SET:



A valid SET consists of three cards where for each property, the values within the SET must be the same or completely different on all three cards. Looking at the valid SET above- the three cards does not share any properties. The number, color, shape and filling is completely different on all three cards. This makes it a valid set.

The purpose of the game is to make different SETs of three cards. The game starts with twelve cards placed on a table in a square formation. Players then search for SETs. When a SET is found the participants shout out “SET” and take the three cards away from the table and the person gets one point. Each time a person says “SET” the SET is checked by the other players to make sure the SET is valid. Three new cards are placed on the table and the game continues until all the cards are used up or when it is impossible to make more SETs from the remaining cards. If no SETs is found in the layout during the game, three new cards are placed on the table. At the end of the game, the player who has collected the most SETs is declared the winner.

9.2 How to introduce the game to students

It is recommend to use at least one and a half hours when introducing the game to students for the first time.

The easiest way to introduce SET to your classroom is simply by telling them all of the information you got in the Section “What SET is and how to play” above, but that will not teach the students anything. A much more powerful way to introduce the game is simply by letting the students explore the deck on their own. The most important thing to remember when introducing this game to the class is that you as a teacher are supposed to interacting with the students to support their problem solving skills. Simply try to not tell the students information about the game, just ask them questions to guide them to the answer.

Start with putting the students in groups of two or three. Try to avoid bigger groups, for the reason that we want a lot of oral activity from all students. Give every group a copy of the deck, and simply tell the groups:

“Take a look at the cards. What do you notice, and what do you wonder about?”

This is a smart way to start the students thought process around mathematics without noticing it themselves. Give the groups some time to discuss what they see. After the groups has discussed for a while it is now time to discuss what the students has found out in plenary. The goal is that the students now have discovered the different properties and their values:

Number: One, Two, Three
Color: Green, Purple, Red
Shape: Oval, Diamond, Worm
Filling: Empty, Striped, Full

It could be an advantage to write up the different properties and values on the board in a systematic way as the students find them. This could be a smart way to find the missing properties and values, because the students will most likely notice a pattern of three values per property. If necessary, draw attention to any missing properties and values.

Now ask the students:

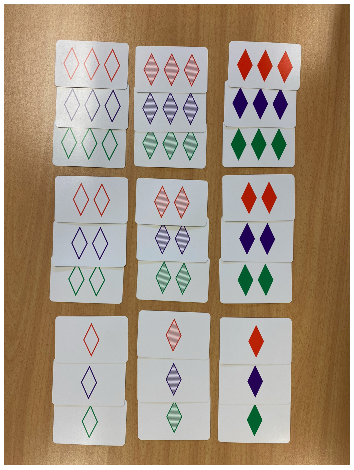
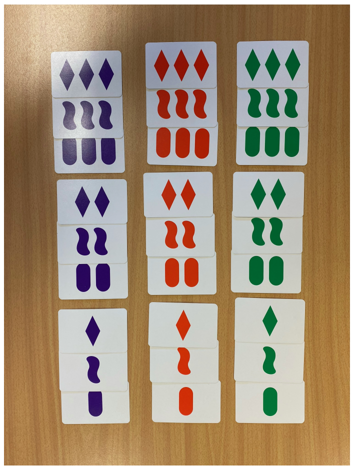
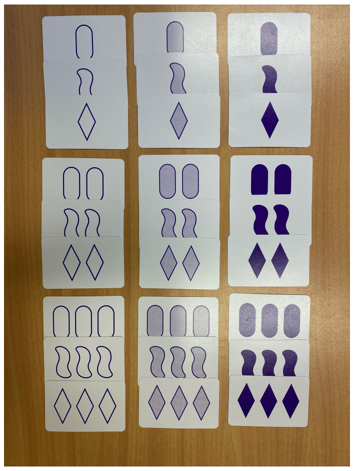
“How many of each card are there, and are all of them there?”

If the students start to answer by guessing, try to lead them by asking questions back. Ask questions like: “How can you be sure? Is there any card missing? Can you explain why?”

Have an open discussion with the class about what they think they have to do. It is smart to come to the conclusion together that the students have to find a way to organize the cards to figure this out. We do this to activate the creative sides of the students. They have to find a way to organize the cards in a systematic way to show this. The students in each group will definitely have different thoughts on what they think will be the best way to organize the cards. The students will now have to practice different social skills when discussing, and make sure they are collaborative within the group. Walk around and guide the different groups, and make sure the groups use their own unique way to organize the cards. We want the groups to organize in different ways to open up for a discussion. It is therefore allowed to give the student hints. Questions to guide the students can be: “In which ways can you organize the cards? Do you need to organize all of the cards together, or can you organize them separately? Could you organize the cards in terms of a value or property?”

After the groups have organized the cards in different ways, walk around together with the class. Discuss the different methods to organize. [1]

Here are three examples of different ways to organize the cards. In the first picture are the cards organized by shape- it is all of the diamond shaped cards in the deck. In The next two pictures are the cards organized by filling and color. Use this for inspiration. [1]

It is now time to find out how many cards the deck consists of. Give the students the following task:

"How many cards are in the deck?

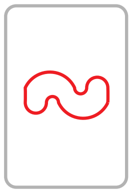
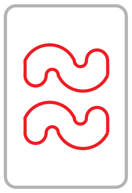
Finally- this is where math and counting comes in. The groups should be discouraged from counting the whole deck one card at a time. The students should use a calculation to find the total number of cards. The calculation will be different between the groups, because they have organized the cards in different ways. But they will all use The Multiplication Principle in some way:

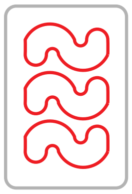
"If there is x possible ways A can occur, and y possible ways B can occur, then the occurrence of both events happens simultaneously are $x * y$" [5]

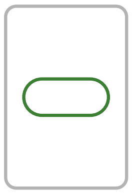
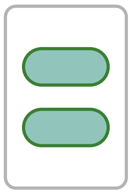
Let every group explain out loud to the rest of the class how they calculated the total number of cards in the deck. It is very beneficial to practice talking using mathematical notation. The students can also learn a lot from each other when they speak using mathematical notation. This is because the students can explain things easier or differently from the teacher, and give the rest of the class a better understanding. When every student has understood why the deck has 81 cards, move on to the next task. The students should by now have a good understanding of the content in the deck, and all the different cards in the deck. But before they start to play the actual game, they should practice how to make a SET. Do not tell the students what the rule for a valid set is yet. We want the student to discover this by themselves. A good idea is to play: What is the final card? Find different SETs from the deck and show the students card A and B, and simply ask them: [1]

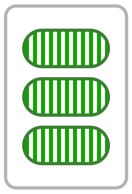
"What is the final card?"

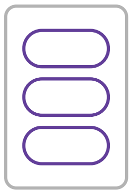
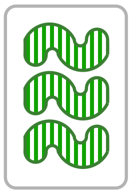
Start with giving the student a SET where it is easy to find the last card. This could be a SET with three properties in common and only one property that are different on every card. After that find SETs that are more and more complicated, so the students has to use more and more time to find card C. Some examples that can be smart to give the student in ascending order is:

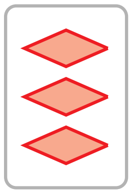
A  B 

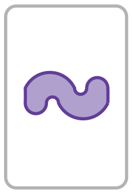
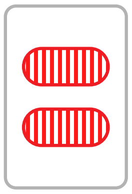
C 

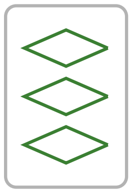
A  B 

C 

A  B 

C 

A  B 

C 

Go carefully through one example at a time. Show the students card A and B. Tell the students to discuss in groups what card A and B have in common, and what they think card C must be. When discussing each example with the whole class go carefully through one property at a time. Start with for example the property color. Ask the students what color card A and B have, and what color they think card C must have. Move on to one of the three remaining properties, for example filling. Ask what filling card A and B have, and what filling they think card C must have. Do this exact same thing with the last two properties, which is shape and number. When you have gone through all of the properties and agreed with the students on which values card C must have, reveal the last card C in the SET.

Come to the conclusion together that every property is either the same or different on every card, and that a valid set is defined as :

"A SET consists of three cards, where each property must be the same or different on every card".

Explain to the students that this is how SETs are found while playing. You always look at two cards and notice which properties they share and not, and then look if the third card that completes the SET are on the table. Explain to the students that this is what we call the fundamental theorem of the card game SET:

Theorem 1: Given two cards, A and B, then there is a unique card C such that the cards A, B, C form a SET.

It is now finally time to let the students play the game. They have explored the whole deck, and know how to make a SET. Use some minutes to explain how the game works, and put together groups of preferred size and let the students play. For first-time players, I would recommend small groups- for example just two students. Try to avoid groups bigger than 4 students, because the game will be quite stressful then. ENJOY!

9.3 Skills developed by playing SET

SET develops a lot of unique skills that are very useful for students, both in school and in everyday life. In Section 7.2 above I made a guide on how to introduce SET to students and explained shortly why the different methods were beneficial for the students. In this chapter, I will explain more about the different skills SET helps students to develop. [6]

9.3.1 Cognitive Development

Cognitive skills are the core skills that our brains use to think, read, remember, learn and pay attention. SET is a unique game that helps to further develop these skills and qualities. [6]

The game is fast paced and has no turns. The players must therefore pay constant attention throughout the whole game. This means SET is perfect to develop concentration and attention skills. [6]

The students have to constantly find SETs during the whole game. The students will create different methods to most easily find SETs, but sometimes will their method not work. The students must therefore think creatively and change their problem solving method. Some of the SETs are not that obvious, and the player have to think creative and in new ways to find these kinds of SETs. [6]

One of the valuable cognitive skills that SET can develop is memory. When playing SET, students have to remember all the cards that are already on the table, and when the dealer puts out three new cards, they need to quickly recognize if any of the new cards can complete a SET with two of the old cards. By remembering some of the cards that are already in a SET and removed from the table, students can also determine which SETs are no longer possible to make. [6]

9.3.2 Social skills

Although the game only requires one player, is SET mostly played in groups by two or more. This means that SET very much is a social game. Using SET in a classroom means that the students have to practice a lot of different social skills. They have to play in groups and collaborate with their classmates, maybe someone they do not get along with. [6]

SET also plays a significant role in developing verbal communication and proper behavior among students. While playing the game, students may experience frustration and must learn to communicate appropriately with their classmates. This is an excellent opportunity for students to practice expressing themselves effectively while respecting others feelings. Additionally, at the end of the game, the group must determine who has the most SETs and declare the winner. Conversely, there will also be losers in the game. Both winning and losing can lead to inappropriate verbal communication, making it crucial for students to practice proper behavior in both scenarios. They learn how to handle success and failure gracefully, which is an essential life skill that can help them in various situations. [6]

9.4 Different ways to play

Let‘s find some different ways to use SET in a classroom.

Original Way
The students should get some experience with the game by playing with the original rules for a while. It is possible to make some changes- for example, switch up the sizes of the groups. This is an excellent way to meet the different needs of each student. A smaller playgroup will better suit students who do not handle stress very well or need more time to master the game. Bigger groups will suit the students who easily find SETs and need more competition. Also, the more students playing with one deck, the more stressful will the game be.

Big cards- Team
Print out the whole deck in a much bigger size- up to the size of a normal sheet of paper. Divide the students into two groups of three. The students will now play in groups of three against each other. Put the deck on a table, and the groups on different sides of the table. The rules of the game are just the same as the original game: you put out twelve cards on the table and try to find SETs, but we add two rules. It is not allowed to talk during the game, and a SET must be found within the group. Let‘s explain this further: [2]

When one person in one of the group finds a SET they have to point at one of the three cards in the SET. Then it is up to the rest of the group. One other person in the group has to figure out what SET their team player has discovered. Once the person has figured this out, they have to point at a second card in this SET. Then the last person has to find the third and final card that makes the three cards a SET. The group takes the cards and gets one point. The team with the most points when the game is finished has won. [2]

It is possible to add one more rule if the students want more challenge. After the first student in one of the groups has found a SET and pointed at one of the cards, then any player from both teams can point at the second card. Let’s say that group one points at the first card, then group two points at the second card. Now can anybody from both teams point at the third and last card. The group who points at the most cards in the SET win the SET and therefore the point. [2]

Even harder: It is allowed for both teams to point at the three cards in a SET, but to get the point all three members of the group have to point. A group can therefore destroy for the other group by pointing on one of the cards. If this happens will neither of the groups get the point. [2]

Online
SET has its own homepage: www.setgame.com, where they publish a daily SET puzzle with 6 different SETs from 12 cards. Using this website will be the easiest way to use SET in a classroom. There is no need of having several decks on hand and requires no planning prior.

This website can be used in many different ways. All the students can play by themselves on their own computers, but the best way is letting the students work together in pairs on one computer. The pairs works together to find all six SETs and the pair who has first found all six SETs win. It will be smart to make the winning pair explain the different SETs they found out loud to the rest of the class. This is a smart way for the students to practice speaking using mathematical notation out loud in front of the class.

References

[1] Albrecht. A. *Counting in unexpected ways.* URL: `https://amiealbrecht.com/2016/08/18/counting-in-unexpected-ways/`. (accessed: 06.05.2023).

[2] D. Butler. *TEAM SET.* URL: `https://blogs.adelaide.edu.au/maths-learning/2019/06/06/team-set/`. (accessed: 06.05.2023).

[3] L. B. Davis and D Maclagan. *THE CARD GAME SET.* URL: `https://web.archive.org/web/20130605073741/http://www.math.rutgers.edu/~maclagan/papers/set.pdf`. (accessed: 04.05.2023).

[4] L. McMahon et al. *The joy of SET: The many mathematical dimensions of a seemingly simple card game.* Princeton University Press, 2017.

[5] Rule of product. *Wikipedia.* URL: `https://en.wikipedia.org/wiki/Rule_of_product`. (accessed: 14.05.2023).

[6] SET Skill Connection for Teachers. *Playmonster.* URL: `https://www.playmonster.com/wp-content/uploads/2019/10/2017-11-07-SET-Skill-Connections-for-Teachers.pdf`. (accessed: 06.05.2023).

10 Appendix

10.1 Python Code

Here the Python code used in the section Simulations and Coding is presented.

main1.py

```
    from simulation import *
import time

for n in [10000000]:
    startTime = time.time()
    result = DoSimulate(n)
    used = time.time() - startTime
    print("Size: " + str(n) + ". Time: " + str(used))
    print("Result:")
    for i in range(81):
        if (result[i] > 0):
            print(str(i) + ": " + str(result[i]) )
    print("----------------")
```

main2.py

```
    from simulation2 import *
import time

first = 3
last = 10
n = 100000
startTime = time.time()
result = DoSimulate(n, range(first,last))
used = time.time() - startTime
print("Size: " + str(n) + ". Time: " + str(used))
print("Result:")
for i in range(81):
    if (first <= i and i < last):
        print(str(i) + ": " + str(100*result[i]/n) + " (" + str(result[i]) + ")" )
print("----------------")
```

main3.py

```
    from simulation3 import *
import time

first = 5
last = 82
n = 100000
startTime = time.time()
for m in range(first, last):
    result = DoSimulate(n, m)
    print(str(m) + ": Avg: " + str(result[0]) +", stdev: " + str(result[1]))
used = time.time() - startTime
print("used: " + str(used) + "----------------")
```

main4.py

```
    from planecomputation import *
import time

first = 48
last = 53
n = 10000
startTime = time.time()
print("No of simulations per n: " + str(n))
print("n, %, # planes, average # of planes")
for m in range(first, last):
    result = OneSimulation(n, m)
    print(str(m) + ", " + str(result[0]*100/n) + ", " + str(result[0]) + ", " + str(result[1]) )
used = time.time() - startTime
print("used: " + str(used) + "----------------")
```

main4-1.py

```
    from simulation4 import *
import time

m = 4
n = 1000000
startTime = time.time()
for i in range(m):
    result = DoSimulate(n)
    print("Of " + str(n) + " we found " + str(result) + " planes")
used = time.time() - startTime
print("used: " + str(used) + "----------------")
```

planecombutation.py

```
    from setfunc import *
import random

# will simulate noOfRun times, each with
# noOfCards random cards.
# It will return a pair:
# result[0] = the precentage of simulations with a plane
# result[1] = the average number of planes in the simulations
#             with a plane
def OneSimulation(noOfRun, noOfCards):
    planeCount = 0
    planeSum = 0
    data = []
    GenerateAllCards("", data)

    for i in range(noOfRun):
        random.shuffle(data)
        subset = data[slice(noOfCards)]
        noOfPlanes = countPlanes(subset)
        if noOfPlanes > 0:
            planeCount = planeCount + 1
            planeSum = planeSum + noOfPlanes
    avgPlane = 0
    if planeCount > 0:
        avgPlane = planeSum/planeCount
    return [planeCount, avgPlane]

# data is a set of cards.
# return the number of planes in data
def countPlanes(data):
    result = 0
    n = len(data)
    for i in range(n):
        for j in range(i+1,n):
            for k in range(j+1,n):
                if not isSet(data[i], data[j], data[k]):
                    canBeAPlane = tryWith(data, data[i], data[j], data[k])
                    if canBeAPlane:
                        result = result + 1
    result = result / 72
    return result

# data is set of cards, and c1, c2 and c4 are cards form data
# that does not define a set.
# Return true iff there is a plane with c1, c2 and c4 in data.
def tryWith(data, c1, c2, c4):
    m = [[c1,c2,''], [c4,'',''], ['','','']]
    c7 = ThirdCard(c1,c4)
    if data.count(c7) == 0 :
        return False
    m[2][0] = c7
    c3 = ThirdCard(c1,c2)
    if data.count(c3) == 0:
        return False
    m[0][2] = c3
    c5 = ThirdCard(c3,c7)
    if data.count(c5) == 0:
        return False
    m[1][1] = c5
    c6 = ThirdCard(c4,c5)
    if data.count(c6) == 0:
        return False
    m[1][2] = c6
    c8 = ThirdCard(c2,c5)
    if data.count(c8) == 0:
        return False
    m[2][1] = c8
    c9 = ThirdCard(c7,c8)
    if data.count(c9) == 0:
        return False
    m[2][2] = c9
    #validate(m)
    #plane = [c1,c2,c3,c4,c5,c6,c7,c8,c9]
    #print("plane " + str(set(plane)) + " in " + str(data))
    # check that there are 12 sets
    #valid = len(ComputeAllSetsWithReplacement(plane)) == 12
    #if not valid:
    #    print("Invalid plane: " + str(plane))
    # check that they are all different
    #s = set(plane)
    #if len(s) != 9:
    #    print("Invalid plane with duplicates " + str(plane))
    return True

def validate(m):
    #rows
    for r in range(3):
        if not isSet(m[r][0], m[r][1], m[r][2]):
            print("Invalid set " + str(m))
    #cols
    for c in range(3):
        if not isSet(m[0][c], m[1][c], m[2][c]):
            print("Invalid set " + str(m))

    if not isSet(m[0][0], m[1][1], m[2][2]):
        print("Invalid set " + str(m))

    if not isSet(m[0][2], m[1][1], m[2][0]):
        print("Invalid set " + str(m))

    if not isSet(m[0][1], m[1][0], m[2][2]):
        print("Invalid set " + str(m))

    from setfunc import *
import random

# will simulate noOfRun times, each with
# noOfCards random cards.
# It will return a pair:
# result[0] = the precentage of simulations with a plane
# result[1] = the average number of planes in the simulations
#             with a plane
def OneSimulation(noOfRun, noOfCards):
    planeCount = 0
    planeSum = 0
    data = []
    GenerateAllCards("", data)

    for i in range(noOfRun):
        random.shuffle(data)
        subset = data[slice(noOfCards)]
        hasPlane = containsPlane(subset)
        if hasPlane:
            planeCount = planeCount + 1
    return planeCount
```

```
# data is a set of cards.
# return the number of planes in data
def countPlanes(data):
    result = 0
    n = len(data)
    for i in range(n):
        for j in range(i+1,n):
            for k in range(j+1,n):
                if not isSet(data[i], data[j], data[k]):
                    canBeAPlane = tryWith(data, data[i], data[j], data[k])
                    if canBeAPlane:
                        result = result + 1
    return result

# data is a set of cards.
# return true iff data contains a plane
def containsPlane(data):
    n = len(data)
    for i in range(n):
        for j in range(i+1,n):
            for k in range(j+1,n):
                if not isSet(data[i], data[j], data[k]):
                    canBeAPlane = tryWith(data, data[i], data[j], data[k])
                    if canBeAPlane:
                        return True
    return False
# data is set of cards, and c1, c2 and c4 are cards form data
# that does not define a set.
# Return true iff there is a plane with c1, c2 and c4 in data.
def tryWith(data, c1, c2, c4):
    m = [[c1,c2,''], [c4,'',''], ['','','']]
    c7 = ThirdCard(c1,c4)
    if data.count(c7) == 0 :
        return False
    m[2][0] = c7
    c3 = ThirdCard(c1,c2)
    if data.count(c3) == 0:
        return False
    m[0][2] = c3
    c5 = ThirdCard(c3,c7)
    if data.count(c5) == 0:
        return False
    m[1][1] = c5
    c6 = ThirdCard(c4,c5)
    if data.count(c6) == 0:
        return False
    m[1][2] = c6
    c8 = ThirdCard(c2,c5)
    if data.count(c8) == 0:
        return False
    m[2][1] = c8
    c9 = ThirdCard(c7,c8)
    if data.count(c9) == 0:
        return False
    m[2][2] = c9
    plane = [c1,c2,c3,c4,c5,c6,c7,c8,c9]
    #validate(m, plane)

    return True

def validate(m, plane):
    #rows
    for r in range(3):
        if not isSet(m[r][0], m[r][1], m[r][2]):
            print("Invalid set " + str(m))
    #cols
    for c in range(3):
        if not isSet(m[0][c], m[1][c], m[2][c]):
            print("Invalid set " + str(m))

    if not isSet(m[0][0], m[1][1], m[2][2]):
        print("Invalid set " + str(m))

    if not isSet(m[0][2], m[1][1], m[2][0]):
        print("Invalid set " + str(m))

    if not isSet(m[0][1], m[1][0], m[2][2]):
        print("Invalid set " + str(m))
    # check that there are 12 sets
    valid = len(ComputeAllSetsWithReplacement(plane)) == 12
    if not valid:
        print("Invalid plane: " + str(plane))
    # check that they are all different
    s = set(plane)
    if len(s) != 9:
        print("Invalid plane with duplicates " + str(plane))
```

setfunc.py

```
    import random

symbols = ['0', '1', '2']

# return true iff a, b and c is a set.
def isSet(a,b,c):
    for i in range(4):
        if a[i] == b[i] and a[i] == c[i]:
            continue
        if a[i] != b[i] and a[i] != c[i] and b[i] != c[i]:
            continue
        return False
    return True
```

```
# a and b are different cards.
# Return the unique third card that together with a and b
# will be a set.
def ThirdCard(a,b):
    result = ""
    for i in range(4):
        if a[i] == b[i]:
            result += str(a[i])
        else:
            for ch in symbols:
                if a[i] != ch and b[i] != ch:
                    result += str(ch)
    return result

# data is a list of cards.
# Return true iff data contains a set.
def ContainsSet(data):
    n = len(data)
    for i in range(n):
        for j in range(i+1,n):
            for k in range(j+1,n):
                if isSet(data[i], data[j], data[k]):
                    return True
    return False

# data is a list of cards.
# Return true iff data contains a set.
def ContainsSetV2(data):
    n = len(data)
    for i in range(n):
        for j in range(i+1,n):
            third = ThirdCard(data[i], data[j])
            if data.count(third) > 0:
                return True
    return False

def printAllSets(data, withReplace):
    n = len(data)
    used = []
    for i in range(n):
        for j in range(i+1,n):
            for k in range(j+1,n):
                if (withReplace):
                    if isSet(data[i], data[j], data[k]):
                        print("A set is (" + data[i] + "," + data[j] + "," + data[k] + ")")
                else:
                    if used.count(data[i]) == 0 and used.count(data[j]) == 0 and used.count(data[k])
                        if isSet(data[i], data[j], data[k]):
                            print("A set is (" + data[i] + "," + data[j] + "," + data[k] + ")")
                            used.append(data[i])
                            used.append(data[j])
                            used.append(data[k])
```

```
# return a list of sets from data with replacement.
def ComputeAllSetsWithReplacement(data):
    n = len(data)
    result = []
    for i in range(n):
        for j in range(i+1,n):
            for k in range(j+1,n):
                if isSet(data[i], data[j], data[k]):
                    result.append([data[i], data[j], data[k]])
    return result

# return a list of sets from data without replacement.
def ComputeAllSets(data):
    n = len(data)
    used = []
    result = []
    for i in range(n):
        for j in range(i+1,n):
            for k in range(j+1,n):
                if used.count(data[i]) == 0 and used.count(data[j]) == 0 and used.count(data[k]) ==
                    if isSet(data[i], data[j], data[k]):
                        result.append([data[i], data[j], data[k]])
                        used.append(data[i])
                        used.append(data[j])
                        used.append(data[k])
    return result

def ComputeAllSetsV2(data):
    n = len(data)
    used = []
    result = []
    for i in range(n):
        for j in range(i+1,n):
            if used.count(data[i]) == 0 and used.count(data[j]) == 0:
                third = ThirdCard(data[i], data[j])
                if data.count(third) > 0 and used.count(third) == 0:
                    result.append([data[i], data[j], third])
                    used.append(data[i])
                    used.append(data[j])
                    used.append(third)
    return result

def ComputeAllSetsV3(data):
    n = len(data)
    used = []
    result = []
    rest = data.copy()
    for i in range(n):
        for j in range(i+1,n):
            if used.count(data[i]) == 0 and used.count(data[j]) == 0:
                third = ThirdCard(data[i], data[j])
                if data.count(third) > 0 and used.count(third) == 0:
                    result.append([data[i], data[j], third])
                    used.append(data[i])
                    used.append(data[j])
                    used.append(third)
                    rest.remove(data[i])
                    rest.remove(data[j])
                    rest.remove(third)
    if len(rest) < 18:
        return result
    elif len(rest) == 18:
        file = open("len_18.txt", "a")
        file.write(str(rest)+ "\n")
        file.close()
    elif len(rest) == 21:
        file = open("len_21.txt", "a")
        file.write(str(rest)+ "\n")
        file.close()
    return result

# Will add all cards/strings of length 4 to container
def GenerateAllCards(prefix, container):
    if (len(prefix) < 4):
        for ch in symbols:
            GenerateAllCards(prefix + str(ch), container)
    else:
        container.append(prefix)
```

simulation.py

```
    from setfunc import *
import random

# perform one reduction of all 81 cards. Return
# the numbers of cards left, when no more reduction
# can be done.
def OneSimulation():
    data = []
    GenerateAllCards("", data)
    random.shuffle(data)
    sets = ComputeAllSetsV3(data)
    return 81-3*len(sets)

# perform n simulations. Return an array a of size 81,
# foreach i: a[i] is the number of simulations ending with
# a non-reducable set of size i.
def DoSimulate(n):
    result = [0]*81
    for i in range(n):
        rest = OneSimulation()
        result[rest] += 1
    return result
```

simulation2.py

```
    from setfunc import *
import random


def OneSimulation(noOfRun, noOfCards):
    searchCount = 0
    data = []
    GenerateAllCards("", data)

    for i in range(noOfRun):
        random.shuffle(data)
        subset = data[slice(noOfCards)]
        if not ContainsSetV2(subset):
            searchCount = searchCount + 1
    return searchCount

def DoSimulate(noOfRun, theRange):
    result = [0]*81
    for noOfCards in theRange:
        res = OneSimulation(noOfRun, noOfCards)
        result[noOfCards] = res
    return result
```

simulation3.py

```
    from setfunc import *
import random
import statistics

# perform one simulation with m random cards. Return
# the numbers of sets that can be drawn with replacement
# from the cards
# can be done.
def OneSimulation(m):
    data = []
    GenerateAllCards("", data)
    random.shuffle(data)
    subset = data[slice(m)]
    sets = ComputeAllSetsWithReplacement(subset)
    return len(sets)

# perform n simulations with m cards. Return a list with two numbers:
# the first one is the average number of sets in the n simulations,
# the second is the standard derivation
def DoSimulate(n, m):
    result = []
    for i in range(n):
        noOfSets = OneSimulation(m)
        result.append(noOfSets)
    avg = sum(result)/len(result)
    stdev = statistics.stdev(result)
    return [avg, stdev]
```

simulation4.py

```
    from setfunc import *
import random
import statistics

# perform one simulation with m random cards. Return
# the numbers of sets that can be drawn with replacement
# from the cards
# can be done.
def OneSimulation(m):
    data = []
    GenerateAllCards("", data)
    random.shuffle(data)
    subset = data[slice(m)]
    sets = ComputeAllSetsWithReplacement(subset)
    return len(sets)

# perform n simulations with 9 cards. Return the number of planes
def DoSimulate(n):
    result = 0
    for i in range(n):
        noOfSets = OneSimulation(9)
        if noOfSets == 12:
            result = result + 1
    return result
```