Leandro Vendramin

Algebra with GAP

– Monograph –

April 20, 2021

Preface

Acknowledgments Several colleagues have helped by providing examples, counterexamples, questions, exercises, solutions and references. Among them, we should especially thank Thomas Brauer, Fernando Fantino, Marco Farinati, César Galindo, Gastón García, István Heckenberger, Alexander Hulpke, Mitja Mastnak, Mikhail Kotchetov, Peter Selinger.

Buenos Aires, 00/00/0000 *Leandro Vendramin*

Contents

Part I Basic theory

1 First steps 3
- 1.1 The very first steps 3
- 1.2 Basic arithmetics 4
- 1.3 Basic programming 9
- 1.4 Permutations 22
- 1.5 Matrices 24
- 1.6 Polynomials 28
- 1.7 Problems 30

2 Basic group theory 37
- 2.1 Basic constructions 37
- 2.2 Homomorphisms 47
- 2.3 SmallGroups 50
- 2.4 Finitely presented groups 54
- 2.5 Problems 58

3 Representations 63
- 3.1 Constructing representations 63
- 3.2 The McKay conjecture 65
- 3.3 Problems 67

4 Advanced group theory 69
- 4.1 Conjugacy classes 69
- 4.2 Simple groups 72
- 4.3 Problems 72

References 73

Index 75

Part I
Basic theory

Chapter 1
First steps

1.1 The very first steps

Immediately after running GAP we will see some information related to the distribution we have installed. This information includes software version number and the packages loaded in memory. We will also see that GAP is ready:

```
gap>
```

To close GAP one uses **`quit`**:

```
gap> quit;
```

Every command should end with the symbol `;` (semicolon). The symbol `;;` (double semicolon) also is used to end a command but it means that no screen output will be produced.

```
gap> 2+5;;
gap> 2+5;
7
```

To see information related commands or functions, tutorials and manuals one uses the symbol `?` (question mark). Here we have some examples:

```
gap> ?tutorial
gap> ?sets
gap> ?help
gap> ?permutations
gap> ?Eigenvalues
gap> ?CyclicGroup
gap> ?FreeGroup
gap> ?SylowSubgroup
```

To write comments one uses the symbol `#` (number sign or hash) as the following example shows:

```
gap> # This is just a comment!
```

To make the command line more readable one could use the symbol \ (backslash):

```
gap> # Let us compute 1+2+3
gap> 1\
> +2\
> +3;
6
```

The function `LogTo` saves the subsequent interaction to be logged to a file. It means that everything you see on your terminal will also appear in this file. This is extremely useful. When the function `LogTo` is called with no parameters GAP will stop writing a log file.

```
gap> # Save the output to the file mylog
gap> LogTo("mylog");
gap> # Stop saving the output
gap> LogTo();
```

`NamesUserGVars` returns the names of the global variables created by the user. `MemoryUsage` returns the bytes used by a variable. Example:

```
gap> p := 2^30;;
gap> NamesUserGVars();
[ "p" ]
gap> s := "a string";;
gap> NamesUserGVars();
[ "p", "s" ]
gap> MemoryUsage(p);
8
```

`SaveWorkspace` saves an image of the memory into a file. Later one can recover this image by calling GAP as follows:

```
gap -L <workspace>
```

1.2 Basic arithmetics

One can do basic arithmetic operations with rational numbers:

```
gap> 1+1;
2
gap> 2*3;
6
gap> 8/2;
4
gap> (1/3)+(2/5);
11/15
gap> 2^(-4);
1/16
```

```
gap> 2*(-6)+4;
-8
gap> (1-5^2)^2-2*(2+4*2)^2;
376
gap> NumeratorRat(3/5);
3
gap> DenominatorRat(3/5);
5
```

One uses **mod** to obtain the remainder after division of *a* by *m*, where *a* is the dividend and *m* is the divisor. Examples:

```
gap> 6 mod 4;
2
gap> -6 mod 5;
4
```

There are several functions that one can use for specific purposes. For example `Factors` returns the factorization of an integer and `IsPrime` detects whether an integer is prime or not.

```
gap> Factors(10);
[ 2, 5 ]
gap> Factors(18);
[ 2, 3, 3 ]
gap> Factors(1800);
[ 2, 2, 2, 3, 3, 5, 5 ]
gap> IsPrime(1800);
false
gap> Factors(37);
[ 37 ]
gap> IsPrime(37);
true
```

Other useful functions: `Sqrt` computes square roots, `Factorial` computes the factorial of a positive integer, `Gcd` computes the greatest common divisor of a finite list of integers, `Lcm` computes the least commom multiple.

```
gap> Sqrt(25);
5
gap> Factorial(15);
1307674368000
gap> Gcd(10,4);
2
gap> Lcm(10,4,2,6);
60
```

We can also work in cyclotomic fields. `CF` creates a cyclotomic field. To create primitive roots of 1 one uses the function `E`. More precisely: `E(n)` returns $e^{2\pi i/n}$. Tipically, cyclotomic numbers will be represented as rational linear combinations of primitive roots of 1. Examples:

```
gap> E(6) in Rationals;
false
```

```
gap> E(6) in Cyclotomics;
true
gap> E(3) in CF(3);
true
gap> E(3) in CF(4);
false
gap> E(3)^2+E(3);
-1
gap> E(5)^5-E(5);
-2*E(5)-E(5)^2-E(5)^3-E(5)^4
gap> E(6);
-E(3)^2
```

`Inverse` (resp. `AdditiveInverse`) returns the multiplicative (resp. additive) inverse of an element.

```
gap> AdditiveInverse(2/3);
-2/3
gap> Inverse(2/3);
3/2
gap> AdditiveInverse(E(7));
-E(7)
gap> Inverse(E(7));
E(7)^6
```

GAP can work with very large numbers. Let us see some nice examples:

Example 1.1. One can easily prove that $n = 164$ is the largest integer such that 7^n divides 1000!:

```
gap> Factorial(1000) mod 7^164;
0
gap> Factorial(1000) mod 7^165 = 0;
false
```

Example 1.2. Let us compute 999^{179} mod 1793:

```
gap> 999^179 mod 1763;
1219
```

The following example appears in `Mathoverflow`, question #282035. Can you prove the result without using a computer software?

Example 1.3. The sum of digits of 3^{1000} is divisible by 7:

```
gap> Sum(ListOfDigits(3^1000)) mod 7;
0
```

The following example shows that it is possible to perform calculations with the ring $\mathbb{Z}[i]$ of Gauss integers.

Example 1.4. In this example we will solve the following exercises:

1. Prove that $1+3i \mid 7+i$.
2. Apply the division algorithm to $\alpha = 4+5i$ and $\beta = 3$.
3. Compute $\gcd(4+i,3)$.
4. Is $3+2i$ prime in $\mathbb{Z}[i]$?
5. Factorize 2 and $7+17i$ in $\mathbb{Z}[i]$.
6. Find all factorizations of 65 as a sum of squares.

First we need to do the following:

```
gap> i := E(4);;
```

To solve the first exercise we observe that one cannot use the operator **mod**. However, we can perform the usual division. For the first exercise we see that $1+3i \mid 7+i$ since $\frac{7+i}{1+3i} \in \mathbb{Z}[i]$:

```
gap> (7+i)/(1+3*i);
1-2*E(4)
```

Alternatively, we could do the following calculation:

```
gap> EuclideanRemainder(7+i,1+3*i);
0
```

Recall that the norm of $\alpha \in \mathbb{Z}[i]$ is defined as the real number $N(\alpha) = \alpha\overline{\alpha}$, where $\overline{\alpha}$ denotes the complex conjugate of α. The division algorithm states that there exist q and r in $Z[i]$ such that $\alpha = q\beta + r$ with $N(r) < N(\beta)$. For the second exercise we find that $4+5i = 3(1+2i)+(1-i)$ and $N(1-i) = 2 < N(3) = 9$. In fact, we compute

```
gap> EuclideanQuotient(4+5*i,3);
1+2*E(4)
gap> EuclideanRemainder(4+5*i,3);
1-E(4)
```

For the third exercise we compute the greatest common divisor of the numbers and, moreover, somehow for the same prize we find elements u and v in $\mathbb{Z}[i]$ such that $(4+5i)u+3v = \gcd(4+5i,3)$. Indeed,

```
gap> Gcd(4+5*i,3);
1
gap> GcdRepresentation(4+5*i,3);
[ 2+2*E(4), 1-6*E(4) ]
```

The fourth exercise is easy once we know how to recognize the ring where we need to work:

```
gap> IsPrime(GaussianIntegers,3+2*i);
true
```

Factorizing numbers is also quite easy once we know where we need to work. To solve the exercise we perform the following calculation:

```
gap> Factors(7+17*i);
[ 1-E(4), 2+3*E(4), 2+3*E(4) ]
gap> Factors(GaussianIntegers, 7+17*i);
[ 1-E(4), 2+3*E(4), 2+3*E(4) ]
```

Here, to find the factors of the number $7+17i$ we do not need to specify the ring, as it is clear from the context. However, the situation is quite different if we want to factorize the number 2 in $\mathbb{Z}[i]$.

```
gap> Factors(2);
[ 2 ]
gap> Factors(GaussianIntegers, 2);
[ 1-E(4), 1+E(4) ]
```

Finite fields

To create the finite field of p^n elements (here p is a prime number) we use the function `GF`. The characteristic of a field can be obtained with `Characteristic`.

```
gap> GF(2);
GF(2)
gap> GF(4);
GF(2^2)
gap> GF(9);
GF(3^2)
gap> Characteristic(Rationals);
0
gap> Characteristic(CF(3));
0
gap> Characteristic(CF(4));
0
gap> Characteristic(GF(2));
2
gap> Characteristic(GF(9));
3
```

Let p be a prime number and let F denote the field with $q := p^n$ elements, for some $n \in \mathbb{N}$. The subset

$$\{x \in F : x \neq 0\}$$

is a cyclic group of size $q-1$; say generated by ζ. Then $F = \{0, \zeta, \zeta^2, \dots, \zeta^{q-1}\}$, so each non-zero element of F is then a power of ζ.

In GAP each non-zero element of the finite field `GF(q)` will be a power of the generator `Z(q)`. The zero of `GF(q)` will be `0*Z(q)` or equivalently `Zero(GF(q))`. `One(GF(q))` will be the multiplicative neutral element of `GF(q)`.

```
gap> Size(GF(4));
4
gap> Elements(GF(4));
```

```
[ 0*Z(2), Z(2)^0, Z(2^2), Z(2^2)^2 ]
gap> Z(4);
Z(2^2)
gap> Inverse(Z(4));
Z(2^2)^2
gap> Zero(GF(4));
0*Z(2)
gap> 0 in GF(4);
false
gap> Zero(Rationals);
0
gap> One(GF(4));
Z(2)^0
gap> 1 in GF(4);
false
gap> One(Rationals);
1
```

To recognize elements in finite fields with a prime number of elements one uses the function `Int`. Examples:

```
gap> Elements(GF(5));
[ 0*Z(5), Z(5)^0, Z(5), Z(5)^2, Z(5)^3 ]
gap> Int(Z(5)^0);
1
gap> Int(Z(5)^1);
2
gap> Int(Z(5)^2);
4
gap> Int(Z(5)^3);
3
```

1.3 Basic programming

Objects and variables

An object is something that we can assign to a variable. So an object could be either a number, a string, a group, a field, an element of a group, a group homomorphism, a ring, a matrix, a vector space... To assign an object to a variable one uses the operator := as the following example shows:

```
gap> p := 32;;
gap> p;
32
gap> p = 32;
true
gap> p := p+1;;
gap> p;
33
```

```
gap> p = 32;
false
```

Remark 1.1. The symbols `=` (conditional) and `:=` (assignment operator) are different!

Remark 1.2. What if I forgot to assign the result of a calculation for further use? We can do the following:

```
gap> 2*(5+1)-6;
6
gap> n := last;
6
```

One also has `last2` and `last3`.

Conditionals

There are three very important operators: **`not`**, **`and`**, **`or`**. We also have comparison operators; for example the expression `x<>y` returns **`true`** if `x` and `y` are different, and **`false`** otherwise. Here we have some easy examples:

```
gap> x := 20;; y := 10;;
gap> x <> y;
true
gap> x > y;
true
gap> (x > 0) or (x < y);
true
gap> (x > 0) and (x < y);
false
gap> (2*y < x);
false
gap> (2*y <= x);
true
gap> not (x < y);
true
```

Example 1.5. Let us check that $100^{300} > 300^{100}$:

```
gap> 100^300 > 300^100;
true
```

Functions

There are two equivalent ways of constructing functions. For example, to construct the map $x \mapsto x^2$ either we can use the *one-line definition*

```
gap> square := x->x^2;
function( x ) ... end
```

or the *classical*

```
gap> square := function(x)
> return x^2;
> end;
function( x ) ... end
```

In both cases we will obtain the same result; with any definition:

```
gap> square(4);
16
gap> square(-4);
16
gap> square(-5);
25
```

One can also define functions with no arguments. A classical example is:

```
gap> hi := function()
> Display("Hello world");
> end;
function(  ) ... end
gap> hi();
Hello world
```

Example 1.6. Let us write a function to compute the map

$$f\colon n \mapsto \begin{cases} n^3 & \text{si } n \equiv 0 \bmod 3,\\ n^5 & \text{si } n \equiv 1 \bmod 3,\\ 0 & \text{otherwise.}\end{cases}$$

Here is the code and some experiments:

```
gap> f := function(n)
> if n mod 3 = 0 then
> return n^3;
> elif n mod 3 = 1 then
> return n^5;
> else
> return 0;
> fi;
> end;
function( n ) ... end
gap> f(10);
100000
gap> f(5);
0
gap> f(4);
1024
```

Example 1.7. The Fibonacci sequence f_n is defined recursively as $f_1 = f_2 = 1$ and

$$f_{n+1} = f_n + f_{n-1}$$

for $n \geq 2$. The following function computes Fibonacci numbers:

```
gap> fibonacci := function(n)
> if n = 1 or n = 2 then
> return 1;
> else
> return fibonacci(n-1)+fibonacci(n-2);
> fi;
> end;
function( n ) ... end
gap> fibonacci(10);
55
```

Question: Can you compute f_{100} with this method?

Example 1.8 (Collatz conjecture). For $n \in \mathbb{N}$ let

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ es even,} \\ 3n+1 & \text{if } n \text{ is odd.} \end{cases}$$

The conjecture is that no matter what number n you start with, there is $m \in \mathbb{N}$ such that $f^m(n) = 1$, where $f^m = f \circ \cdots \circ f$ (m-times). Let us test the conjecture for $n = 5$:

```
gap> f := function(n)
> if n mod 2 = 0 then
> return n/2;
> else
> return 3*n+1;
> fi;
> end;
function( n ) ... end
gap> f(f(f(f(f(5)))));
1
```

Exercise: Write a function that for each n returns the smallest integer m such that $f^m(n) = 1$.

A possible solution uses recursive functions:

```
gap> g:=function(n)
> if f(n)=1 then
> return 1;
> else
> return 1+g(f(n));
> fi;
> end;
function( n ) ... end
```

A different approach, which uses ideas we will develop on page 20, can be given as follows:

```
gap> g:=function(n)
> m:=1;
> while f(n)>1 do
> m:=m+1;
> n:=f(n);
> od;
> return m;
> end;
> function( n ) ... end
```

Strings

A string is an expression delimited by the symbol `"` (quotation mark):

```
gap> string := "hello world";
hello world
```

To extract one character one uses the expression `string[position]`; to extract substrings `string{positions}`. Examples:

```
gap> string[1];
'h'
gap> string[3];
'l'
gap> string{[1,2,3,4,5]};
"hello"
gap> string{[7,8,9,10,11]};
"world"
gap> string{[11,10,9,8,7,6,5,4,3,2,1]};
"dlrow olleh"
```

There are several functions that allow us to work with strings. `String` converts anything into a string of characters.

```
gap> String(1234);
"1234"
gap> String(01234);
"1234"
gap> String([1,2,3]);
"[ 1, 2, 3 ]"
gap> String(true);
"true"
```

The function `ReplacedString` replace substrings:

```
gap> ReplacedString("Hello world", "world", "all");
"Hello all"
```

`Print` allows us to print data in the screen.

```
gap> string := "Hello world";;
gap> Print(string);
Hello world
```

Let us see another example:

```
gap> n := 100;;
gap> m := 5;;
gap> Print(n, " times ", m, " is ", n*m);
100 times 5 is 500
```

The function `Print` can be used with some special characters. For example, `\n` means “new line”. Examples:

```
gap> Print("Hello\nworld");
Hello
world
gap> Print("To write \\...");
To write \...
```

The functions `PrintTo` and `AppendTo` work as `Print` but the output goes to a file. It is important to remark that `PrintTo` will overwrite an existing file!

Lists

A list is an ordered sequence of objects (maybe of different type), including empty places. Lists are written using square brackets. Examples:

```
gap> IsList([1, 2, 3]);
true
gap> IsList([1, 2, 3, "abc"]);
true
gap> IsList([1, 2,, "abc"]);
true
gap> 2 in [1, 2, 5, 4, 10];
true
gap> 3 in [0,10,"abc"];
false
```

Example 1.9. Let us create a list with the first six prime numbers. `Size` or `Length` return the number of non-empty elements of the list.

```
gap> primes := [2, 3, 5, 7, 11, 13];
[ 2, 3, 5, 7, 11, 13 ]
gap> Size(primes);
6
```

To access to an element inside a list one should refer to the position.

```
gap> primes[1];
2
gap> primes[2];
3
```

Let us obtain the sublist consisting of the elements in the second, third and fifth position:

```
gap> primes;
[ 2, 3, 5, 7, 11, 13 ]
gap> primes{[2,3,5]};
[ 3, 5, 11 ]
```

Another example (to avoid confusion):

```
gap> list := ["a", "b", "c", "d", "e", "f"];
[ "a", "b", "c", "d", "e", "f" ]
gap> list{[1,3,5]};
[ "a", "c", "e" ]
```

To find elements inside a list one uses `Position`. If the element we are looking for does not belong to the list, `Position` will return `fail`; otherwise it will return the first place where the element appears. Let us look at some examples:

```
gap> Position([5, 4, 6, 3, 7, 3, 7], 5);
1
gap> Position([5, 4, 6, 3, 7, 3, 7], 1);
fail
gap> Position([5, 4, 6, 3, 7, 3, 7], 7);
5
```

`Add` and `Append` are used to add elements at the end of a list. To remove elements from a list one uses `Remove`; examples:

```
gap> primes;
[ 2, 3, 5, 7, 11, 13 ]
gap> # Add 19 at the end of the list
gap> Add(primes, 19);
gap> primes;
[ 2, 3, 5, 7, 11, 13, 19 ]
gap> # Add the prime 17 at position 7
gap> Add(primes, 17, 7);
gap> primes;
[ 2, 3, 5, 7, 11, 13, 17, 19 ]
gap> # Add 23 and 29 at the end
gap> Append(primes, [23, 29]);
gap> primes;
[ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 ]
gap> # Remove the first element of the list
gap> Remove(primes, 1);;
gap> primes;
[ 3, 5, 7, 11, 13, 17, 19, 23, 29 ]
```

`Concatenation` concatenates two or more lists. This function returns a new list consisting of the lists used in the argument.

```
gap> Concatenation([1,2,3],[4,5,6]);
[ 1, 2, 3, 4, 5, 6 ]
```

Remark 1.3. Is there any difference between `Append` and `Concatenation`? Yes! The function `Concatenation` does not modify the lists used in the argument. `Append` does.

`Collected` returns a new list where each element of the original list appears with multiplicity. Example:

```
gap> Factors(720);
[ 2, 2, 2, 2, 3, 3, 5 ]
gap> Collected(last);
[ [ 2, 4 ], [ 3, 2 ], [ 5, 1 ] ]
```

To make a copy of a list one should use the function `ShallowCopy`. The following example shows the difference between `ShallowCopy` and the assignment operator.

```
gap> a := [1, 2, 3, 4];;
gap> b := a;;
gap> c := ShallowCopy(a);;
gap> Add(a, 5);
gap> a;
[ 1, 2, 3, 4, 5 ]
gap> b;
[ 1, 2, 3, 4, 5 ]
gap> c;
[ 1, 2, 3, 4 ]
gap> Add(b, 10);
gap> a;
[ 1, 2, 3, 4, 5, 10 ]
gap> b;
[ 1, 2, 3, 4, 5, 10 ]
```

The function `Reversed` returns a list containing the elements of our list in reversed order. In the following example the variable `list` will not be modified by the function `Reversed`:

```
gap> list := [2, 4, 7, 3];;
gap> Reversed(list);
[ 3, 7, 4, 2 ]
gap> list;
[ 2, 4, 7, 3 ]
```

`SortedList` returns a new list where the elements are sorted with respect to the operator `<=`. In the following example one sees that `SortedList` will not modify the value of the variable `list`:

```
gap> list := [2, 4, 7, 3];;
gap> SortedList(list);
```

```
[ 2, 3, 4, 7 ]
gap> list;
[ 2, 4, 7, 3 ]
```

`Sort` sorts a list in increasing order. Can you recognize the difference between `Sort` and `SortedList`?

```
gap> list := [2, 4, 7, 3];;
gap> Sort(list);
gap> list;
[ 2, 3, 4, 7 ]
```

Remark 1.4. Say that we want to apply `SortedList` or `Sort` to a given list. In this case, all the elements of the list must be of the same type and comparable with respect to the operator `<=`.

`Filtered` allows us to obtain the elements of a list that satisfy a particular given property. The function `Number` returns the number of elements of a list that satisfy a given property. `First` returns the first element of a list that satisfy a given property. Some examples are as follows:

```
gap> list := [1, 2, 3, 4, 5];;
gap> Filtered(list, x->x mod 2 = 0);
[ 2, 4 ]
gap> Number(list, x->x mod 2 = 0);
2
gap> Filtered(list, x->x mod 2 = 1);
[ 1, 3, 5 ]
gap> First(list, x->x mod 2 = 0);
2
```

Example 1.10. Let us compute how many powers of 2 divide 18000. This number is four, as the following code shows:

```
gap> Factors(18000);
[ 2, 2, 2, 2, 3, 3, 5, 5, 5 ]
gap> Collected(Factors(18000));
[ [ 2, 4 ], [ 3, 2 ], [ 5, 3 ] ]
gap> Number(Factors(18000), x->x=2);
4
```

There are very nice ways to create lists. The following examples need no further explanations.

```
gap> List([1, 2, 3, 4, 5], x->x^2);
[ 1, 4, 9, 16, 25 ]
gap> List([1, 2, 3, 4, 5], IsPrime);
[ false, true, true, false, true ]
```

Ranges

Ranges are lists where the difference between two consecutive integers is a constant. Examples:

```
gap> Elements([1,3..11]);
[ 1, 3, 5, 7, 9, 11 ]
gap> Elements([1..5]);
[ 1, 2, 3, 4, 5 ]
gap> Elements([0,-2..-8]);
[ -8, -6, -4, -2, 0 ]
gap> IsRange([1..100]);
true
gap> IsRange([1,3,5,6]);
false
```

We can use `Elements` to list all the elements in a given range. Conversely, `ConvertToRangeRep` converts (if possible) a list into a range. Examples:

```
gap> list := [ 1, 2, 3, 4, 5 ];;
gap> ConvertToRangeRep(list);;
gap> list;
[ 1 .. 5 ]
gap> list := [ 7, 11, 15, 19, 23 ];
gap> IsRange(list);
true
gap> ConvertToRangeRep(list);
gap> list;
[ 7, 11 .. 23 ]
```

Sets

A set is a particular type of ordered list that contains no gaps with no repetitions. To convert a list to a set one uses `Set`.

```
gap> list := [1, 2, 3, 1, 5, 6, 2];;
gap> IsSet(list);
false
gap> Set(list);
[ 1, 2, 3, 5, 6 ]
```

To add elements use `AddSet` and `UniteSet`. To remove them, `RemoveSet`. Examples:

```
gap> set := Set([1, 2, 4, 5]);;
gap> # Let us add the number 10
gap> AddSet(set, 10);
gap> set;
[ 1, 2, 4, 5, 10 ]
gap> # Let us remove the number 4
gap> RemoveSet(set, 4);
gap> set;
[ 1, 2, 5, 10 ]
gap> UniteSet(set, [1, 1, 5, 6]);
gap> set;
[ 1, 2, 5, 6, 10 ]
```

To perform basic set operations one uses `Union`, `Intersection`, `Difference` and `Cartesian`. Examples:

```
gap> S := Set([1, 2, 8, 11]);;
gap> T := Set([2, 5, 7, 8]);;
gap> Intersection(S, T);
[ 2, 8 ]
gap> Union(S, T);
[ 1, 2, 5, 7, 8, 11 ]
gap> Difference(S, T);
[ 1, 11 ]
gap> Difference(T, S);
[ 5, 7 ]
gap> Difference(S, S);
[  ]
gap> Cartesian(S, T);
[ [ 1, 2 ], [ 1, 5 ], [ 1, 7 ], [ 1, 8 ], [ 2, 2 ],
  [ 2, 5 ], [ 2, 7 ], [ 2, 8 ], [ 8, 2 ], [ 8, 5 ],
  [ 8, 7 ], [ 8, 8 ], [ 11, 2 ], [ 11, 5 ],
  [ 11, 7 ], [ 11, 8 ] ]
```

Records

Records allow us to put several objects in the same structure. Let us say that we want to create a structure for the point $(1,2)$ of the plane:

```
gap> point := rec(x := 1, y := 2);;
gap> point.x;
1
gap> point.y;
2
gap> RecNames( point );
[ "x", "y" ]
```

The function **`IsBound`** can be used to test whether our structure contains a given component. The function **`Unbind`** removes a variable from the memory. Examples:

```
gap> point := rec( x := 1, y := 2 );;
gap> IsBound(point.z);
false
gap> point.z := 3;;
gap> IsBound(point.z);
true
```

```
gap> point;
rec( x := 1, y := 2, z := 3 )
gap> Unbind(point.z);
gap> point;
rec( x := 1, y := 2 )
```

Loops

Our section of loops will be based on the following very simple problem. We want to check that

$$1+2+3+\cdots+100=5050.$$

Of course we can use `Sum`, which sums all the elements of a list:

```
gap> Sum([1..100]);
5050
```

An equivalent way of doing this would be using **`for ... do ... od`**:

```
gap> s := 0;;
gap> for k in [1..100] do
> s := s+k;
> od;
gap> s;
5050
```

Yet another equivalent way of doing this is the following: **`while ... do ... od`**:

```
gap> s := 0;;
gap> k := 1;;
gap> while k<=100 do
> s := s+k;
> k := k+1;
> od;
gap> s;
5050
```

Yet another equivalent way of doing this is the following: **`repeat ... until`**:

```
gap> s := 0;;
gap> k := 1;;
gap> repeat
> s := s+k;
> k := k+1;
> until k>100;
gap> s;
5050
```

Now let us compute (again) Fibonacci numbers. This is better than the method of Example 1.7.

Example 1.11. Let us write a non-recursive function to compute Fibonacci numbers.

```
gap> fibonacci := function(n)
> local k, x, y, tmp;
> x := 1;
> y := 1;
> for k in [3..n] do
>   tmp := y;
>   y := x+y;
>   x := tmp;
> od;
> return y;
> end;
function( n ) ... end
gap> fibonacci(100);
354224848179261915075
gap> fibonacci(1000);
434665576869374564356885276750406258025646605173\
717804024817290895365554179490518904038798400792\
551692959225930803226347752096896232398733224711\
616429964409065331879382989696499285160037044761\
37795166849228875
```

Example 1.12. Divisors of a given integer can be obtained with `DivisorsInt`. In this example we run over the divisors of 100 and print only those that are odd.

```
gap> Filtered(DivisorsInt(100), x->x mod 2 = 1);
[ 1, 5, 25 ]
```

Similarly

```
gap> for d in DivisorsInt(100) do
> if d mod 2 = 1 then
> Display(d);
> fi;
> od;
1
5
25
```

With **`continue`** one can skip iterations. An equivalent (but less elegant) approach to the problem of Example 1.12 is the following:

```
gap> for d in DivisorsInt(100) do
> if d mod 2 = 0 then
> continue;
> fi;
> Display(d);
> od;
1
5
25
```

With **break** one breaks a loop. In the following example we run over the numbers $1,2,\dots,100$ and stop when a number whose square is divisible by 20 appears.

```
gap> First([1..100], x->x^2 mod 20 = 0);
10
```

Similarly:

```
gap> for k in [1..100] do
> if k^2 mod 20 = 0 then
> Display(k);
> break;
> fi;
> od;
10
```

`ForAny` returns **true** if there is an element in the list satisfying the required condition and **false** otherwise. Similarly `ForAll` returns **true** if all the elements of the list satisfy the required condition and **false** otherwise. Examples:

```
gap> ForAny([2,4,6,8,10], x->x mod 2 = 0);
true
gap> ForAll([2,4,6,8,10], x->(x > 0));
true
gap> ForAny([2,3,4,5], IsPrime);
true
gap> ForAll([2,3,4,5], IsPrime);
false
```

1.4 Permutations

Let $n \in \mathbb{N}$. A **permutation** in n letters is a bijective map $\sigma\colon \{1,\dots,n\} \to \{1,\dots,n\}$. For example, the permutation $\binom{1234}{3124}$ is the bijective map $1 \mapsto 3$, $2 \mapsto 1$, $3 \mapsto 2$ and $4 \mapsto 4$. Usually one writes a permutation as a product of disjoint cycles. For example:

$$\begin{pmatrix}1234\\2413\end{pmatrix} = (1243), \qquad \begin{pmatrix}12345\\21435\end{pmatrix} = (12)(34)(5) = (12)(34).$$

The permutation $\binom{12345}{21435} = (12)(34)$ in GAP is `(1,2)(3,4)`. The function `IsPerm` checks whether some object is a permutation. Let us see some examples:

```
gap> IsPerm((1,2)(3,4));
true
gap> (1,2)(3,4)(5)=(1,2)(3,4);
true
gap> (1,2)(3,4)=(3,4)(2,1);
true
gap> IsPerm(25);
false
```

```
gap> IsPerm([1,2,3,4]);
false
```

The image of an element `i` under the natural right action of a permutation `p` is `i^p`. The preimage of the element `i` under `p` can be obtained with `i/p`. In the following example we compute the image of 1 and the preimage of 3 by the permutation (123):

```
gap> 2^(1,2,3);
3
gap> 2/(1,2,3);
1
```

Composition of permutations will be performed from left to right. For example

$$(123)(234) = (13)(24)$$

as the following code shows:

```
gap> (1,2,3) * (2,3,4);
(1,3)(2,4)
```

To obtain the inverse of a permutation one uses `Inverse`:

```
gap> Inverse((1,2,3));
(1,3,2)
gap> (1,2,3)^(-1);
(1,3,2)
```

Let σ be a permutation written as a product of disjoint cycles. The function `ListPerm` returns a list containing $\sigma(i)$ at position i. Conversely, any list of this type can be transformed into a permutation with the function `PermList`. Examples:

```
gap> # The permutation (12) in two letters
gap> ListPerm((1,2));
[ 2, 1 ]
gap> # The permutation (12) in four letters
gap> ListPerm((1,2), 4);
[ 2, 1, 3, 4 ]
gap> ListPerm((1,2,3)(4,5));
[ 2, 3, 1, 5, 4 ]
gap> ListPerm((1,3));
[ 3, 2, 1 ]
gap> PermList([1,2,3]);
()
gap> PermList([2,1]);
(1,2)
```

The **sign** of a permutation σ is the number $(-1)^k$, where $\sigma = \tau_1 \cdots \tau_k$ is some factorization of σ as a product of transpositions. To compute the sign of a permutation one uses the function `SignPerm`. Examples:

```
gap> SignPerm(());
1
```

```
gap> SignPerm((1,2));
-1
gap> SignPerm((1,2,3,4,5));
1
gap> SignPerm((1,2)(3,4,5));
-1
gap> SignPerm((1,2)(3,4));
1
```

Example 1.13. For a given n we will construct the permutation $\sigma \in \mathbb{S}_n$ given by $\sigma(j) = n - j + 1$, we will write σ as a product of disjoint cycles and we will compute its sign:

```
gap> n := 5;;
gap> p := PermList(List([1..n], j->n-j+1));
(1,5)(2,4)
gap> SignPerm(p);
1
```

1.5 Matrices

For us a matrix will be just a rectangular array of numbers. The size of a matrix can be obtained with `DimensionsMat`. Sometimes (for example if one has an integer matrix) the function `Display` shows matrices in a nice way. `LaTeX` returns the LaTeX command[1] needed to write a matrix. Examples:

```
gap> m := [[1,2,3],[4,5,6]];;
gap> Display(m);
[ [  1,  2,  3 ],
  [  4,  5,  6 ] ]
gap> LaTeX(m);
"\\left(\\begin{array}{rrr}%\n1&2&3\\
\\%\n4&5&6\\\\%\n\\end{array}\\right)%\n"
gap> m[1][1];
1
gap> m[1][2];
2
gap> m[2][1];
4
gap> DimensionsMat(m);
[ 2, 3 ]
```

Let us play with row vectors:

Example 1.14. Let $v = (1,2,3)$ and $w = (0,5,-7)$ be row vectors of $\mathbb{Q}^3$. Let us check that $-5v = (-5,-10,-15)$ and $2v - w = (2,-1,13)$. We also check that the inner product between v and w is $v \cdot w = -11$.

[1] The function `LaTeX` works for other GAP objects as well.

```
gap> v := [1,2,3];;
gap> w := [0,5,-7];;
gap> IsRowVector(v);
true
gap> IsRowVector(w);
true
gap> -5*v;
[ -5, -10, -15 ]
gap> 2*v-w;
[ 2, -1, 13 ]
gap> v*w;
-11
```

We now perform some elementary calculations with matrices:

Example 1.15. Let

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 2 \\ 6 & 1 \\ 0 & 2 \end{pmatrix}.$$

Let us compute A^3, BC, CB, $A+CB$ and $2A-5CB$:

```
gap> A := [[1,1,1],[0,1,1],[0,0,1]];;
gap> B := [[1,4,7],[2,5,8]];;
gap> C := [[1,2],[6,1],[0,2]];;
gap> Display(A^3);
[ [  1,  3,  6 ],
  [  0,  1,  3 ],
  [  0,  0,  1 ] ]
gap> Display(B*C);
[ [  25,  20 ],
  [  32,  25 ] ]
gap> Display(C*B);
[ [   5,  14,  23 ],
  [   8,  29,  50 ],
  [   4,  10,  16 ] ]
gap> Display(A+C*B);
[ [   6,  15,  24 ],
  [   8,  30,  51 ],
  [   4,  10,  17 ] ]
gap> Display(2*A-5*C*B);
[ [   -23,   -68,  -113 ],
  [   -40,  -143,  -248 ],
  [   -20,   -50,   -78 ] ]
```

To construct a null matrix one uses the function `NullMat`. The identity is constructed with the function `IdentityMat`. To construct diagonal matrices one uses `DiagonalMat`. Examples:

```
gap> Display(NullMat(2,3));
```

```
[ [  0,  0,  0 ],
  [  0,  0,  0 ] ]
gap> Display(IdentityMat(3));
[ [  1,  0,  0 ],
  [  0,  1,  0 ],
  [  0,  0,  1 ] ]
gap> Display(DiagonalMat([1,2]));
[ [  1,  0 ],
  [  0,  2 ] ]
```

We know that `matrix[i][j]` returns the (i,j)-element of our matrix. To extract submatrices from a matrix one uses `matrix{rows}{columns}` such as the following example shows:

```
gap> m := [\
> [1, 2, 3, 4, 5],\
> [6, 7, 8, 9, 3],\
> [3, 2, 1, 2, 4],\
> [7, 5, 3, 0, 0],\
> [0, 0, 0, 0, 1]];
gap> m{[1..3]}{[1..3]};
[ [ 1, 2, 3 ], [ 6, 7, 8 ], [ 3, 2, 1 ] ]
gap> m{[2,4,5]}{[1,3]};
[ [ 6, 8 ], [ 7, 3 ], [ 0, 0 ] ]
```

It is possible to work with matrices with coefficients in arbitrary rings. Let us start working with matrices over the finite field $\mathbb{F}_5$ of five elements:

```
gap> m := [[1,2,3],[3,2,1],[0,0,2]]*One(GF(5));
[ [ Z(5)^0, Z(5), Z(5)^3 ],
  [ Z(5)^3, Z(5), Z(5)^0 ],
  [ 0*Z(5), 0*Z(5), Z(5) ] ]
gap> Display(m);
 1 2 3
 3 2 1
 . . 2
```

Now let us work with 3×3 matrices with coefficients in the ring $\mathbb{Z}/4$. Let us compute the identity of $M_3(\mathbb{Z}/4)$:

```
gap> m := IdentityMat(3, ZmodnZ(4));;
gap> Display(m);
matrix over Integers mod 4:
[ [  1,  0,  0 ],
  [  0,  1,  0 ],
  [  0,  0,  1 ] ]
```

One uses the function `Inverse` to compute the inverse of an invertible (square) matrix. This function returns `fail` if the matrix is not invertible. `IsIdentityMat` returns either **`true`** if the argument is the identity matrix or **`false`** otherwise. We also use `TransposedMat` to compute the transpose of a matrix:

```
gap> m := [[1, -2, -1], [0, 1, 0], [1, -1, 0]];;
gap> Display(Inverse(m));
```

```
[ [   0,   1,   1 ],
  [   0,   1,   0 ],
  [  -1,  -1,   1 ] ]
gap> Inverse([[1,0],[2,0]]);
fail
gap> IsIdentityMat(m*Inverse(m));
true
gap> Display(TransposedMat(m)*m);
[ [   2,  -3,  -1 ],
  [  -3,   6,   2 ],
  [  -1,   2,   1 ] ]
```

Creating random matrices is easy. `RandomMat` returns a random rectangular matrix over a given ring, which defaults to $\mathbb{Z}$. With `RandomInvertibleMat` (resp. with `RandomUnimodularMat`) one creates a random square matrix with integer entries invertible (resp. over the integers).

```
gap> RandomMat(2,2);
[ [ 2, 0 ], [ 4, -1 ] ]
gap> RandomInvertibleMat(2);
[ [ 1, -2 ], [ -1, 0 ] ]
gap> Inverse(last);
[ [ 0, -1 ], [ -1/2, -1/2 ] ]
gap> RandomUnimodularMat(3);
[ [ 5, -15, 28 ], [ -2, 6, -11 ], [ -11, 32, -60 ] ]
gap> Inverse(last);
[ [ -8, -4, -3 ], [ 1, 8, -1 ], [ 2, 5, 0 ] ]
```

Example 1.16. An easy induction exercise shows that the Fibonacci sequence (f_n) can be computed using

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n = \begin{pmatrix} f_{n-1} & f_n \\ f_n & f_{n+1} \end{pmatrix}, \quad n \geq 1.$$

We use this trick to compute (very efficiently) Fibonacci numbers:

```
gap> fibonacci := function(n)
> local m;
> m := [[0,1],[1,1]]^n;;
> return m[1][2];
> end;
function( n ) ... end
gap> fibonacci(10);
55
gap> fibonacci(100000);
<integer 259...875 (20899 digits)>
```

One can easily compute the characteristic and minimal polynomial of a matrix with `CharacteristicPolynomial` and `Minimal Polynomial`, respectively.

Let A be an $m \times n$ and let b be an $1 \times n$ matrix (say, a row vector of length n). `SolutionMat` returns (if possible) some x of size $1 \times m$ such that $xA = b$. In case there is no x such that $xA = b$, the function returns `fail`. Let us see some examples:

```
gap> # Only one solution
gap> SolutionMat([[1,2],[3,4]], [13,2]);
[ -23, 12 ]
gap> # Infinite solutions
gap> SolutionMat([[1,2],[2,4]], [1,2]);
[ 1, 0 ]
gap> # No solutions
gap> SolutionMat([[1,2],[2,4]], [1,3]);
fail
```

The trace of a square matrix is computed with `Trace`. The determinant with `Determinant`. The rank of a matrix is computed with `Rank`. Examples:

```
gap> m := [[1, 2, 3],[5, 4, 3],[0, 0, 2]];;
gap> Determinant(m);
-12
gap> Trace(m);
7
gap> Rank(m);
3
gap> Rank(NullMat(2, 2));
0
gap> Rank([[1, 2, 3],[4, 5, 6],[6, 7, 8]]);
2
gap> Rank([[1, 2, 3],[4, 5, 6],[6, 7, 9]]);
3
```

The function `NullspaceMat` computes the vector space generated by the solutions of $xA = 0$, where x is a matriz of size $1 \times m$ and A is a matrix of size $m \times n$.

1.6 Polynomials

One can define polynomial rings with `PolynomialRing`. Indeterminates are defined with `IndeterminatesOfPolynomialRing`. As an example, we define the polynomial $\mathbb{Q}[x]$ in one variable x over the ring $\mathbb{Q}$s:

```
gap> A := PolynomialRing(Rationals,["x"]);;
gap> x := IndeterminatesOfPolynomialRing(A)[1];
x
```

Let us see some operations on polynomials. One uses the function `Value` to evaluate polynomials. With `Factors` one obtains the factorization of a polynomial over the default ring where the polynomial is defined. `RootsOfPolynomial` returns all the roots of the polynomial. Here we have some examples:

```
gap> x := Indeterminate(Rationals);;
gap> DefaultRing(x);
Rationals[x]
gap> IsPolynomial(x^2+x-2);
true
```

```
gap> IsPolynomial((x^2+x-2)/(x-1));
true
gap> AsPolynomial((x^2+x-2)/(x-1));
x+2
gap> Value(x^2+x-2, 0);
-2
gap> Value(x^2+x-2, 1);
0
gap> Value(x^2+x-2, -1);
-2
gap> Factors(x^2+x-2);
[ x-1, x+2 ]
gap> RootsOfPolynomial(x^2+x-2);
[ 1, -2 ]
```

The degree of a polynomial can be obtained with `Degree`. The coefficients with the function `CoefficientsOfUnivariatePolynomial`. The example below needs no further explanation:

```
gap> x := Indeterminate(Rationals);;
gap> f := x^5+2*x^3+3*x^2+4;;
gap> Degree(f);
5
gap> CoefficientsOfUnivariatePolynomial(f);
[ 4, 0, 3, 2, 0, 1 ]
gap> LeadingCoefficient(f);
1
```

Example 1.17. Let $f = 21x^2 + 9$ and $g = 20x^4 + 10x$ be polynomials in $\mathbb{Z}/30[x]$. Let us check that $fg = 0$:

```
gap> x := Indeterminate(Integers mod 30);;
gap> f := 21*x^2+9;;
gap> g := 20*x^4+10*x;;
gap> f*g;
ZmodnZObj(0,30)
gap> IsZero(f*g);
true
```

Example 1.18. Let us check that $2x^2 + 1$ divides $6x^3 + 10x^2 + 3x + 5$ in $\mathbb{Q}[x]$:

```
gap> x := Indeterminate(Rationals);;
gap> (6*x^3+10*x^2+3*x+5) mod (2*x^2+1);
0
gap> (6*x^3+10*x^2+3*x+5)/(2*x^2+1);
3*x+5
```

Example 1.19. Let us factorize in $\mathbb{Q}[x]$ the polynomial $f = 2x^5 + 3x^4 - x^2 - 2x + 1$ and prove that

$$2x^5 + 3x^4 - x^2 - 2x + 1 = (2x - 1)(x^2 + x - 1)(x^2 + x + 1).$$

Here is the code:

```
gap> x := Indeterminate(Rationals);;
gap> Factors(2*x^5+3*x^4-x^2-2*x+1);
[ 2*x-1, x^2+x-1, x^2+x+1 ]
```

Now we see that a cubic root of one is a root of our f. Let us try to factorize f over $\mathbb{Q}(\omega)$, where ω is a cubic root of one:

```
gap> Factors(PolynomialRing(Field(E(3)),"x"),\
> 2*x^5+3*x^4-x^2-2*x+1);
[ 2*x-1, x+(-E(3)), x+(-E(3)^2), x^2+x-1 ]
```

Example 1.20. Let $f = x^2 + 5x + 2$ and $g = x^4 + 1$. We check that $3f - 2g$ is irreducible (over the integers):

```
gap> x:=Indeterminate(Integers,"x");;
gap> y:=Indeterminate(Integers,"y");;
gap> f := x^2+5*x+2;;
gap> g := x^4+1;;
gap> 3*f-2*g;
-2*x^4+3*x^2+15*x+4
gap> IsIrreducible(last);
true
```

Example 1.21. The polynomial $x^2 - x + 41$ gives a prime for $x \in \{0,\dots,40\}$. Let us check this:

```
gap> x := Indeterminate(Rationals);;
gap> p := x^2-x+41;;
gap> List([0..40], j->Value(p, j));
[ 41, 41, 43, 47, 53, 61, 71, 83, 97, 113, 131, 151,
  173, 197, 223, 251, 281, 313, 347, 383, 421, 461,
  503, 547, 593, 641, 691, 743, 797, 853, 911, 971,
  1033, 1097, 1163, 1231, 1301, 1373, 1447, 1523,
  1601 ]
gap> Filtered(last, j->not IsPrime(j));
[  ]
```

1.7 Problems

1.1. Use `ChineseRem` to find (if possible) the smallest solution of the following congruences:

$$\begin{cases} x \equiv 3 \bmod 10, \\ x \equiv 8 \bmod 15, \\ x \equiv 5 \bmod 84 \end{cases} \qquad \begin{cases} x \equiv 29 \bmod 52, \\ x \equiv 19 \bmod 72. \end{cases}$$

1.2. Compute the gcd of 42823 and 6409. Find $x, y \in \mathbb{Z}$ such that

$$\gcd(5033464705, 3138740337) = 5033464705x + 3138740337y.$$

1.3. Describe the following sequence: $a_1 = 3$, $a_{n+1} = 3^{a_n} \bmod 100$.

1.4. Use `Product` to compute $2 \cdot 4 \cdot 6 \cdots 20$.

1.5. Prove that

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \cdots + \frac{1}{9999} - \frac{1}{10000} = \frac{1}{5001} + \frac{1}{5002} + \cdots + \frac{1}{10000}.$$

1.6. Find the last two digits of 3^{400}.

1.7. Find the roots of $x^2 + x + 7 \equiv 0 \bmod m$ for $m \in \{15, 189\}$.

1.8. Write a function that returns the binary expansion of an integer. Could you do this for other bases?

1.9. Use `MinimalPolynomial` to compute the minimal polynomial of $3 + \sqrt{5}$ over the rational numbers.

1.10. Compute the first 100 Fibonacci numbers.

1.11. For $k \in \mathbb{N}$ let a_n be given by $a_1 = \cdots = a_{k+1} = 1$ and $a_n = a_{n-k} + a_{n-k-1}$ for all $n > k+1$. Write a function depending on k that constructs the sequence a_n. For more information see `http://oeis.org/A103379`.

1.12 (Somos sequence). Write a function that returns the n-th term of a_n, where $a_0 = a_1 = a_2 = a_3 = 1$ and

$$a_n = \frac{a_{n-1}a_{n-3} + a_{n-2}^2}{a_{n-4}}$$

for all $n \geq 4$. For more information see `http://oeis.org/A006720`.

1.13. Write a function that given a list `lst` of words and a letter `x`, returns a sublist of `lst` where every word starts with `x`.

1.14. Write a function that returns the number of prime numbers $\leq n$.

1.15. Use the function `Permuted` to write a function that shows all the anagrams of a given word.

1.16. Given a list of non-negative numbers, write a function that displays the histogram associated with this list. For example, if the argument is the list `[1,4,2]`, the function should display

```
X
XXXX
XX
```

1.17. Write a function that given a list of words returns the longest one.

1.18. Write a function that transforms a given number of seconds in days, hours, minutes and seconds.

1.19. Write a function that returns the average value of a given list of numbers.

1.20. Write a function that, given a letter, returns **`true`** if the letter is a vowel and **`false`** otherwise.

1.21. Use `CharacteristicPolynomial` to compute the characteristic polynomial of the matrix $\begin{pmatrix} 0 & -1 & 1 \\ 1 & 2 & -1 \\ 1 & 1 & 0 \end{pmatrix}$. Can you compute the minimal polynomial?

1.22. Use the function `QuotientRemainder` to compute the quotient and the remainder of $f = 2x^4 + 3x^3 + 2x + 4$ and $g = 3x^2 + x + 2$ in the ring $\mathbb{Z}_5[x]$.

1.23. Compute $3x^{101} - 15x^{16} - 2x^7 - 5x^4 + 3x^3 + 2x^2 + 1 \bmod x^3 + 1$.

1.24. Prove that $x = 2$ is the only root in $\mathbb{Z}_5$ of $x^{1000} + 4x + 1 \in \mathbb{Z}_5[x]$.

1.25. Factorize $x^4 - 1$ in $\mathbb{Z}/5[x]$ and in $\mathbb{Z}/7[x]$.

1.26. Prove that $x^2 - 79x + 1601$ gives a prime number for $x \in \{0, 1, \dots, 79\}$.

1.27. Write the first 50 twin primes.

1.28. FRACTRAN is a programming language invented by J. Conway. A FRACTRAN program is simply an ordered list of positive rationals together with an initial positive integer input n. The program is run by updating the integer n as follows:

- For the first rational f in the list for which $nf \in \mathbb{Z}$, replace n by nf.
- Repeat this rule until no rational in the list produces an integer when multiplied by n, then stop.

Write an implementation of the FRACTRAN language.

Starting with $n = 2$, the program

$$\frac{17}{65}, \frac{133}{34}, \frac{17}{19}, \frac{23}{17}, \frac{2233}{69}, \frac{23}{29}, \frac{31}{23}, \frac{74}{341}, \frac{31}{37}, \frac{41}{31}, \frac{129}{287}, \frac{41}{43}, \frac{13}{41}, \frac{1}{13}, \frac{1}{3}$$

produces the sequence

$$2, 15, 825, 725, 1925, 2275, 425, 390, 330, 290, 770 \dots$$

In 1987, J. Conway proved that this sequence contains the set $\{2^p : p \text{ prime}\}$. See `https://oeis.org/A007542` for more information.

1.29. The first terms of Conway's "look and say" sequence are the following:

```
1
11
21
1211
111221
312211
```

After guessing how each term is computed, write a script to create the first terms of the sequence.

1.30. Write

$$\begin{pmatrix}123456\\253461\end{pmatrix}, \qquad \begin{pmatrix}123456789\\234517896\end{pmatrix}, \qquad \begin{pmatrix}12345\\32451\end{pmatrix},$$

as a product of disjoint cycles.

1.31. Write the permutations $(123)(45)(1625)(341)$ and $(12)(245)(12)$ as product of disjoint cycles.

1.32. Find a permutation τ such that

1. $\tau(12)(34)\tau^{-1} = (56)(13)$.
2. $\tau(123)(78)\tau^{-1} = (257)(13)$.
3. $\tau(12)(34)(567)\tau^{-1} = (18)(23)(456)$.

1.33. Compute $\tau\sigma\tau^{-1}$ in the following cases:

1. $\sigma = (123)$ and $\tau = (34)$.
2. $\sigma = (567)$ and $\tau = (12)(34)$.

1.34. Let $\sigma \in \mathbb{S}_9$ be given by $\sigma(i) = 10 - i$ for all $i \in \{1,\dots,9\}$. Write σ as a product of disjoint cycles.

1.35. Find (if possible) three permutations $\alpha,\beta,\gamma \in \mathbb{S}_5$ such that $\alpha\beta = \beta\alpha$, $\beta\gamma = \gamma\beta$ and $\alpha\gamma \neq \gamma\alpha$.

1.36. Use the function `PermutationMat` to write the elements of $\mathbb{S}_3$ as 3×3 matrices.

1.37. For $A = \begin{pmatrix}1&2&3\\4&5&6\\0&2&3\end{pmatrix}$ compute

$$I + A + \frac{1}{2!}A^2 + \frac{1}{3!}A^3 + \frac{1}{4!}A^4.$$

1.38. Write the funcion

$$(n,A) \mapsto I + A + \frac{1}{2!}A^2 + \frac{1}{3!}A^3 + \cdots + \frac{1}{n!}A^n.$$

1.39. For $n \in \mathbb{N}$ the Hilbert matrix H_n is defined as

$$(H_n)_{ij} = \frac{1}{i+j-1}, \quad i,j \in \{1,\dots,n\}.$$

Write the function $n \mapsto H_n$.

1.40. Use the function `KroneckerProduct` to compute

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \otimes \begin{pmatrix} 5 & 6 & 7 \\ 2 & 1 & 0 \\ 0 & 1 & 9 \end{pmatrix}.$$

1.41. Let S be the vector space (over the rationals) generated by $(0,1,0)$ and $(0,0,1)$ and T be generated by $(1,2,0)$ and $(3,1,2)$. Use `VectorSpace` to create these vector spaces and compute $\dim S$, $\dim T$, $\dim(S \cap T)$ and $\dim(S+T)$.

1.42. Write the coordinates of the vector $(1,0,1)$ in the basis given by $(2i,1,0)$, $(2,-i,1)$, $(0,1+i,1-i)$.

1.43. Walsh matrices $H(2^k)$, $k \geq 1$, are defined recursively as follows:

$$H(2) = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad H(2^k) = H(2) \otimes H(2^{k-1}), \quad k \geq 1,$$

Construct the function $n \mapsto H(2^n)$.

1.44. Use the functions `Eigenvalues` and `Eigenvalues` to compute the eigenvalues and eigenvectors of the matrix

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 6 & 7 & 8 \end{pmatrix} \in \mathbb{Q}^{3\times 3}.$$

The function `Eigenvectors` returns generators of the eigenspaces, where $v \neq 0$ is an eigenvector of A with eigenvalue λ if and only if $vA = \lambda v$. This is because actions are on the right!

1.45. Use the function `NullspaceMat` to compute the nullspace of the matrix A from problem 1.44.

The nullspace of A is defined as the set of vectors v such that $vA = 0$. This is because actions are on the right!

1.46.

1. Prove that $1+3i \nmid 4+i$.
2. Apply the division algorithm to $\alpha = 2+7i$ and $\beta = 1+2i$.
3. Compute $\gcd(7+17i, 8-14i)$.

4. Is $2+3i$ prime in $\mathbb{Z}[i]$?
5. Is 2 prime in $\mathbb{Z}[i]$?
6. Is 3 prime in $\mathbb{Z}[i]$?
7. Factorize $8-14i$ and $-39+48i$ in $\mathbb{Z}[i]$.

Chapter 2
Basic group theory

2.1 Basic constructions

A **matrix group** is a subgroup of $\mathbf{GL}(n,\mathbb{K})$ for some $n \in \mathbb{N}$ and some field $\mathbb{K}$. A **permutation group** is a subgroup of some $\mathbb{S}_n$. One constructs groups with the function `Group`.

Example 2.1. With `Order` we compute the order of the following groups: a) the group generated by the transposition (12), b) the group generated by the 5-cycle (12345), and c) the group generated by the permutations $\{(12),(12345)\}$:

```
gap> Order(Group([(1,2)]));
2
gap> Order(Group([(1,2,3,4,5)]));
5
gap> Order(Group([(1,2),(1,2,3,4,5)]));
120
```

For $n \in \mathbb{N}$ let C_n be the (multiplicative) cyclic group of order n. One construct cyclic groups with `CyclicGroup`. With no extra arguments, this function returns an abstract presentation of a cyclic group.

Example 2.2. Let us construct the cyclic group C_2 of size two as an abstract group, as a matrix group and as a permutation group.

```
gap> CyclicGroup(2);
<pc group of size 2 with 1 generators>
gap> CyclicGroup(IsMatrixGroup, 2);
Group([ [ [ 0, 1 ], [ 1, 0 ] ] ])
gap> CyclicGroup(IsPermGroup, 2);
Group([ (1,2) ])
```

For $n \in \mathbb{N}$ the **dihedral group** of order $2n$ is the group

$$\mathbb{D}_{2n} = \langle r,s : srs = r^{-1}, s^2 = r^n = 1\rangle.$$

To construct dihedral groups we use `DihedralGroup`. With no extra arguments, the function returns an abstract presentation of a dihedral group. As we did before for cyclic groups, we can construct dihedral groups as permutation groups.

Example 2.3. Let us construct $\mathbb{D}_6$, compute its order and check that this is an abelian group.

```
gap> D6 := DihedralGroup(6);;
gap> Order(D6);
6
gap> IsAbelian(D6);
false
```

To display the elements of the group we use `Elements`:

```
gap> Elements(DihedralGroup(6));
[ <identity> of ..., f1, f2, f1*f2, f2^2, f1*f2^2 ]
gap> Elements(DihedralGroup(IsPermGroup, 6));
[ (), (2,3), (1,2), (1,2,3), (1,3,2), (1,3) ]
```

One constructs the symmetric group $\mathbb{S}_n$ with `SymmetricGroup`. Of course, the elements of $\mathbb{S}_n$ are permutations of the set $\{1,\dots,n\}$ To construct the alternating group $\mathbb{A}_n$ one uses `AlternatingGroup`.

Example 2.4. Let us construct $\mathbb{A}_4$ and $\mathbb{S}_4$ and display their elements.

```
gap> S4 := SymmetricGroup(4);;
gap> A4 := AlternatingGroup(4);;
gap> Elements(A4);
[ (), (2,3,4), (2,4,3), (1,2)(3,4), (1,2,3), (1,2,4),
  (1,3,2), (1,3,4), (1,3)(2,4), (1,4,2), (1,4,3),
  (1,4)(2,3) ]
gap> Elements(S4);
[ (), (3,4), (2,3), (2,3,4), (2,4,3), (2,4), (1,2),
  (1,2)(3,4), (1,2,3), (1,2,3,4), (1,2,4,3), (1,2,4),
  (1,3,2), (1,3,4,2), (1,3), (1,3,4), (1,3)(2,4),
  (1,3,2,4), (1,4,3,2), (1,4,2), (1,4,3), (1,4),
  (1,4,2,3), (1,4)(2,3) ]
```

Now let us check that

```
gap> (1,2,3) in A4;
true
gap> (1,2) in A4;
false
gap> (1,2,3)(4,5) in S4;
false
```

Example 2.5. Let us check that $\mathbb{S}_3$ has two elements of order three and three elements of order two. One computes order of elements with `Order`.

```
gap> S3 := SymmetricGroup(3);;
gap> List(S3, Order);
[ 1, 2, 3, 2, 3, 2 ]
gap> Collected(List(S3, Order));
[ [ 1, 1 ], [ 2, 3 ], [ 3, 2 ] ]
```

Example 2.6. Let us show that

$$G = \left\langle \begin{pmatrix} 0 & i \\ i & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \right\rangle$$

is a non-abelian group of order eight not isomorphic to a dihedral group. Recall that for GAP, the imaginary unit $i = \sqrt{-1}$ is `E(4)`. To check that $G \not\simeq \mathbb{D}_8$ we see that G contains a unique element of order two and $\mathbb{D}_8$ has five elements of order two:

```
gap> a := [[0,E(4)],[E(4),0]];;
gap> b := [[0,1],[-1,0]];;
gap> G := Group([a,b]);;
gap> Order(G);
8
gap> IsAbelian(G);
false
gap> Number(G, x->Order(x)=2);
1
gap> Number(DihedralGroup(8), x->Order(x)=2);
5
```

Example 2.7. The Mathieu group M_{11} is a simple group of order 7920. It is defined as the subgroup of $\mathbb{S}_{11}$ generated by

$$(1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11), \quad (3\,7\,11\,8)(4\,10\,5\,6).$$

Let us construct M_{11} and check with `IsSimple` that M_{11} is simple:

```
gap> a := (1,2,3,4,5,6,7,8,9,10,11);;
gap> b := (3,7,11,8)(4,10,5,6);;
gap> M11 := Group([a,b]);;
gap> Order(M11);
7920
gap> IsSimple(M11);
true
```

Example 2.8. The function `Group` can also be used to construct infinite groups. Let us consider two matrices with finite order and such that their product has infinite order.

```
gap> a := [[0,-1],[1,0]];;
gap> b:= [[0,1],[-1,-1]];;
gap> Order(a);
4
```

```
gap> Order(b);
3
gap> Order(a*b);
infinity
gap> Order(Group([a,b]));
infinity
```

Remark 2.1. Not always GAP will be able to determine whether an element has finite order or not!

With `Subgroup` we construct the subgroup of a group generated by a list of elements. The function `AllSubgroups` returns the list of subgroups of a given group. The index of a subgroup can be computed with `Index`.

Example 2.9. Let us check that the subgroup of $\mathbb{S}_3$ generated by (12) is $\{\mathrm{id},(12)\}$ and has index three, and the subgroup of $\mathbb{S}_3$ generated by (123) is $\{\mathrm{id},(123),(132)\}$ and has index two:

```
gap> S3 := SymmetricGroup(3);;
gap> Elements(Subgroup(S3, [(1,2)]));
[ (), (1,2) ]
gap> Index(S3, Subgroup(S3, [(1,2)]));
3
gap> Elements(Subgroup(S3, [(1,2,3)]));
[ (), (1,2,3), (1,3,2) ]
gap> Index(S3, Subgroup(S3, [(1,2,3)]));
2
```

Recall that a subgroup K of G is said to be normal if $gKg^{-1} \subseteq K$ for all $g \in G$. If K is normal in G, then G/K is a group.

Example 2.10. With `IsSubgroup` we check that $\mathbb{A}_4$ is a subgroup of $\mathbb{S}_4$. With `IsNormal` we see that $\mathbb{A}_4$ is a subset of $\mathbb{S}_4$ under conjugation:

```
gap> S4 := SymmetricGroup(4);;
gap> A4 := AlternatingGroup(4);;
gap> IsSubgroup(S4,A4);
true
gap> IsNormal(S4,A4);
true
gap> Order(S4/A4);
2
```

The subgroup of $\mathbb{S}_4$ generated by (123) is not normal in $\mathbb{S}_4$:

```
gap> IsNormal(S4, Subgroup(S4, [(1,2,3)]));
false
```

Example 2.11. Let us show that in $\mathbb{D}_8$ there are subgroups H and K such that K is normal in H, H is normal in G and K is not normal in G.

```
gap> D8 := DihedralGroup(IsPermGroup, 8);;
gap> Elements(D8);
[ (), (2,4), (1,2)(3,4), (1,2,3,4),
  (1,3), (1,3)(2,4), (1,4,3,2),
  (1,4)(2,3) ]
gap> K := Subgroup(D8, [(2,4)]);;
gap> Elements(K);
[ (), (2,4) ]
gap> H := Subgroup(D8, [(1,2,3,4)^2,(2,4)]);;
gap> Elements(H);
[ (), (2,4), (1,3), (1,3)(2,4) ]
gap> IsNormal(D8, K);
false
gap> IsNormal(D8, H);
true
gap> IsNormal(H, K);
true
```

Example 2.12. Let us compute the quotients of the cyclic group C_4. Since every subgroup of C_4 is normal, we can use `AllSubgroups` to check that C_4 contains a unique non-trivial proper subgroup K. The quotient C_4/K has two elements:

```
gap> C4 := CyclicGroup(IsPermGroup, 4);;
gap> AllSubgroups(C4);
[ Group(()), Group([ (1,3)(2,4) ]),
  Group([ (1,2,3,4) ]) ]
gap> K := last[2];;
gap> Order(C4/K);
2
```

Recall that for $n \in \mathbb{N}$ the generalized quaternion group is the group

$$Q_{4n} = \langle x, y \mid x^{2n} = y^4 = 1, x^n = y^2, y^{-1}xy = x^{-1} \rangle.$$

We use `QuaternionGroup` to construct generalized quaternion groups. As we did before, we can use the filters `IsPermGroup` (resp. `IsMatrixGroup`) to obtain generalized quaternion groups as permutation (resp. matrix) groups.

Example 2.13. Let us check that each subgroup of the quaternion group Q_8 of order eight is normal and that Q_8 is non-abelian:

```
gap> Q8 := QuaternionGroup(IsMatrixGroup, 8);;
gap> IsAbelian(Q8);
false
gap> ForAll(AllSubgroups(Q8), x->IsNormal(Q8,x));
true
```

If G is a group, its center is the subgroup

$$Z(G) = \{x \in G : xy = yx \text{ for all } y \in G\}.$$

The commutator of two elements $x,y \in G$ is defined as $[x,y] = x^{-1}y^{-1}xy$. The commutator subgroup, or derived subgroup of G, is the subgroup $[G,G]$ generated by all the commutators of G.

Example 2.14. Let us check that $\mathbb{A}_4$ has trivial center and that its commutator is the group $\{\mathrm{id}, (12)(34), (13)(24), (14)(23)\}$:

```
gap> A4 := AlternatingGroup(4);;
gap> IsTrivial(Center(A4));
true
gap> Elements(DerivedSubgroup(A4));
[ (), (1,2)(3,4), (1,3)(2,4), (1,4)(2,3) ]
```

To construct direct products of groups one uses the function `DirectProduct`.

Example 2.15. Let us check that $C_4 \times C_4$ and $C_2 \times Q_8$ have order 16, have three elements of order two and twelve elements of order four.

```
gap> C4 := CyclicGroup(IsPermGroup, 4);;
gap> C2 := CyclicGroup(IsPermGroup, 2);;
gap> Q8 := QuaternionGroup(8);;
gap> C4xC4 := DirectProduct(C4, C4);;
gap> C2xQ8 := DirectProduct(C2, Q8);;
gap> List(C4xC4, Order);
[ 1, 4, 2, 4, 4, 4, 4, 4, 2, 4, 2, 4, 4, 4, 4, 4 ]
gap> Collected(List(C4xC4, Order));
[ [ 1, 1 ], [ 2, 3 ], [ 4, 12 ] ]
gap> List(C2xQ8, Order);
[ 1, 4, 4, 2, 4, 4, 4, 4, 2, 4, 4, 2, 4, 4, 4, 4 ]
gap> Collected(List(C2xQ8, Order));
[ [ 1, 1 ], [ 2, 3 ], [ 4, 12 ] ]
```

Are these groups isomorphic? No. An easy way to see this is the following: $C_4 \times C_4$ is abelian and $C_2 \times Q_8$ is not:

```
gap> IsAbelian(C4xC4);
true
gap> IsAbelian(C2xQ8);
false
```

Recall that if G is a group and $g \in G$, the **conjugacy class** of g in G is the subset $g^G := \{x^{-1}gx : x \in G\}$. The **centralizer** of g in G is the subgroup

$$C_G(g) = \{x \in G : xg = gx\}.$$

To compute conjugacy classes we have the following functions: `ConjugacyClasses` and `ConjugacyClass`. The centralizer can be computed with `Centralizer`.

Example 2.16. Let us check that $\mathbb{S}_3$ contains three conjugacy classes with representatives id, (12) and (123), so that

$$(12)^{\mathbb{S}_3} = \{(12),(13),(23)\}, \qquad (123)^{\mathbb{S}_3} = \{(123),(132)\}.$$

```
gap> S3 := SymmetricGroup(3);;
gap> ConjugacyClasses(S3);
[ ()^G, (1,2)^G, (1,2,3)^G ]
gap> Elements(ConjugacyClass(S3, (1,2)));
[ (2,3), (1,2), (1,3) ]
gap> Elements(ConjugacyClass(S3, (1,2,3)));
[ (1,2,3), (1,3,2) ]
```

Let us check that $C_{\mathbb{S}_3}((123)) = \{\mathrm{id}, (123), (132)\}$:

```
gap> Elements(Centralizer(S3, (1,2,3)));
[ (), (1,2,3), (1,3,2) ]
```

Example 2.17. In this example we use the function `Representative` to construct a list of representatives of conjugacy classes of $\mathbb{A}_4$:

```
gap> A4 := AlternatingGroup(4);;
gap> List(ConjugacyClasses(A4), Representative);
[ (), (1,2)(3,4), (1,2,3), (1,2,4) ]
```

With the function `IsConjugate` we can check whether two elements are conjugate. If two elements g and h are conjugate, we want to find an element x such that $g = x^{-1}hx$. For that purpose we use `RepresentativeAction`.

Example 2.18. Let us check that (123) and $(132) = (123)^2$ are not conjugate in $\mathbb{A}_4$:

```
gap> A4 := AlternatingGroup(4);;
gap> g := (1,2,3);;
gap> IsConjugate(A4, g, g^2);
false
```

Now we check that (123) and (134) are conjugate in $\mathbb{A}_4$. We also find an element $x = (234)$ such that $(134) = x^{-1}(123)x$:

```
gap> h := (1,3,4);;
gap> IsConjugate(A4, g, h);
true
gap> x := RepresentativeAction(A4, g, h);
(2,3,4)
gap> x^(-1)*g*x=h;
true
```

Example 2.19. It is well-known that the converse of Lagrange theorem does not hold. The following example is based on [2]. Let us show that $\mathbb{A}_4$ has no subgroups of order six. A naive idea to prove that $\mathbb{A}_4$ has no subgroups of order six is to study all the $\binom{12}{6} = 924$ subsets of $\mathbb{A}_4$ of size six and check that none of these subsets is a group:

```
gap> A4 := AlternatingGroup(4);;
gap> k := 0;;
gap> for x in Combinations(Elements(A4), 6) do
> if Size(Subgroup(A4, x))=Size(x) then
> k := k+1;
> fi;
> od;
gap> k;
0
```

This is an equivalent way of doing the same thing:

```
gap> ForAny(Combinations(Elements(A4), 6),\
> x->Size(Subgroup(A4, x))=Size(x));
false
```

Now we use a similar idea. We use that every subgroup of order six contains exactly five elements besides the unit 1. So we see that none of the $\binom{11}{5} = 462$ subsets of $\mathbb{A}_4$ with five elements generates a subgroup of order six. In the following code we do not use `Combinations`. Combinations will be generated by using an iterator.

```
gap> k := 0;;
gap> for t in IteratorOfCombinations(\
> Filtered(A4, x->not x = ()), 5) do
> if Size(Subgroup(A4, t))=Size(t)+1 then
> k := k+1;
> fi;
> od;
gap> k;
0
```

Here we have another idea: if $\mathbb{A}_4$ has a subgroup of order six, then the index of this subgroup in $\mathbb{A}_4$ is two. With `SubgroupsOfIndexTwo` we check that $\mathbb{A}_4$ has no subgroups of index two:

```
gap> SubgroupsOfIndexTwo(A4);
[  ]
```

Of course we can construct all subgroups and check that there are no subgroups of order six:

```
gap> List(AllSubgroups(A4), Order);
[ 1, 2, 2, 2, 3, 3, 3, 3, 4, 12 ]
gap> 6 in last;
false
```

Moreover, it is enough to construct all conjugacy classes of subgroups:

```
gap> c := ConjugacyClassesSubgroups(A4);;
gap> List(c, x->Order(Representative(x)));
[ 1, 2, 3, 4, 12 ]
gap> 6 in last;
false
```

Another approach is to use conjugacy classes of elements in $\mathbb{A}_4$. Indeed, the conjugacy classes of $\mathbb{A}_4$ are:

$$\begin{aligned} &\{\text{id}\}, && \{(243),(123),(134),(142)\},\\ &\{(12)(34),(13)(24),(14)(23)\}, && \{(234),(124),(132),(143)\}. \end{aligned}$$

This is how we construct the conjugacy classes of $\mathbb{A}_4$:

```
gap> ConjugacyClasses(A4);
[ ()^G, (1,2)(3,4)^G, (1,2,3)^G, (1,2,4)^G ]
gap> Elements(ConjugacyClass(A4, ()));
[ () ]
gap> Elements(ConjugacyClass(A4, (1,2)(3,4)));
[ (1,2)(3,4), (1,3)(2,4), (1,4)(2,3) ]
gap> Elements(ConjugacyClass(A4, (1,2,3)));
[ (2,4,3), (1,2,3), (1,3,4), (1,4,2) ]
gap> Elements(ConjugacyClass(A4, (1,2,4)));
[ (2,3,4), (1,2,4), (1,3,2), (1,4,3) ]
```

Assume that $\mathbb{A}_4$ has a subgroup K of order six. Then K has index two in $\mathbb{A}_4$ and hence it is normal in $\mathbb{A}_4$. This means that K is a union of conjugacy classes of $\mathbb{A}_4$ and that $\{1\} \subseteq K$. This is a contradiction!

Let us now use the commutator to prove that $\mathbb{A}_4$ has no subgroups of order six. If there exists a subgroup K of order six, then K is normal in $\mathbb{A}_4$ and the quotient $\mathbb{A}_4/K$ is cyclic of order two. This implies that

$$[\mathbb{A}_4,\mathbb{A}_4] = \{\text{id},(12)(34),(13)(24),(14)(23)\},$$

is contained in K, a contradiction since 4 does not divide 6:

```
gap> DerivedSubgroup(A4);
Group([ (1,4)(2,3), (1,3)(2,4) ])
gap> Elements(last);
[ (), (1,2)(3,4), (1,3)(2,4), (1,4)(2,3) ]
```

One more variation. If K is a subgroup of $\mathbb{A}_4$ of order six, then there are two possibilities: either $K \simeq \mathbb{S}_3$ or $K \simeq C_6$. The group $\mathbb{A}_4$ has no elements of order six:

```
gap> Filtered(A4, x->Order(x)=6);
[  ]
```

Then $K \simeq \mathbb{S}_3$ and hence K contains three elements of order two. Thus

$$\{\text{id},(12)(34),(13)(24),(14)(23)\}.$$

Do you have another proof of the fact that $\mathbb{A}_4$ has no subgroups of order six?

Example 2.20. If G is a finite abelian group, the structure theorem states that there exist $n_1,\dots,n_k \in \mathbb{N}$ such that $G \simeq C_{n_1} \times \cdots \times C_{n_k}$. To construct finite abelian groups one uses `AbelianGroup`. Let us construct $C_2 \times C_3$ and check that this group is isomorphic to C_6. For that purpose, we see that $C_2 \times C_3$ contains an element of order six:

```
gap> C2xC3 := AbelianGroup(IsPermGroup, [2, 3]);;
gap> First(C2xC3, x->Order(x)=6);
(1,2)(3,4,5)
```

To study isomorphisms between finite group one uses `IsomorphismGroups`. This function returns `fail` if the groups are not isomorphic or some isomorphism otherwise.

Example 2.21. Let us construt a group G such that $G = \langle a\rangle \times \langle b\rangle$ with $C_4 \simeq \langle a\rangle$ and $C_2 \simeq \langle b\rangle$. We also prove that

$$\langle a^2\rangle \simeq \langle b\rangle, \qquad G/\langle a^2\rangle \not\simeq G/\langle b\rangle.$$

Here is the code:

```
gap> G := AbelianGroup(IsPermGroup, [4,2]);
Group([ (1,2,3,4), (5,6) ])
gap> K := Subgroup(G, [(5,6)]);;
gap> L := Subgroup(G, [(1,2,3,4)^2]);;
gap> IsomorphismGroups(K, L);
[ (5,6) ] -> [ (1,3)(2,4) ]
gap> IsomorphismGroups(G/K,G/L);
fail
```

One can show that:

$$\langle a^2\rangle \simeq \langle b\rangle \simeq C_2, \qquad G/\langle a^2\rangle \simeq C_4, \qquad G/\langle b\rangle \simeq C_2 \times C_2.$$

We can also work with classic groups. Use

```
gap> ?classical groups
```

to get more information.

Example 2.22. One can use the function `GL` to construct the groups $\mathbf{GL}(n,\mathbb{Z})$, $\mathbf{GL}(n,\mathbb{Z}/m)$ and $\mathbf{GL}(n,\mathbb{F}_q)$:

```
gap> Order(GL(2,Integers));
infinity
gap> Order(GL(2,ZmodnZ(4)));
96
gap> Order(GL(2,GF(4)));
180
gap> Order(GL(3,GF(4)));
181440
```

Similarly, with `SL` one constructs $\mathbf{SL}(n,\mathbb{Z})$, $\mathbf{SL}(n,\mathbb{Z}/m)$ and $\mathbf{SL}(n,\mathbb{F}_q)$:

```
gap> Order(SL(2,GF(3)));
24
gap> Order(SL(2,Integers));
infinity
```

```
gap> Order(SL(2,ZmodnZ(4)));
48
gap> Order(SL(2,GF(4)));
60
```

It is known that the commutator of a finite group is not always equal to the set of commutators. The following example is based on [5].

Example 2.23. Let G be the subgroup of $\mathbb{S}_{16}$ generated by the permutations

$$\begin{aligned} a &= (13)(24), & b &= (57)(6,8),\\ c &= (911)(10,12), & d &= (13,15)(14,16),\\ e &= (13)(5,7)(9,11), & f &= (12)(3,4)(13,15),\\ g &= (56)(7,8)(13,14)(15,16), & h &= (9\ 10)(11\ 12). \end{aligned}$$

We show that $[G,G]$ has order 16:

```
gap> a := (1,3)(2,4);;
gap> b := (5,7)(6,8);;
gap> c := (9,11)(10,12);;
gap> d := (13,15)(14,16);;
gap> e := (1,3)(5,7)(9,11);;
gap> f := (1,2)(3,4)(13,15);;
gap> g := (5,6)(7,8)(13,14)(15,16);;
gap> h := (9,10)(11,12);;
gap> G := Group([a,b,c,d,e,f,g,h]);;
gap> D := DerivedSubgroup(G);;
gap> Size(D);
16
```

We now show that the set of commutators has 15 elements. In particular, we show that $cd \in [G,G]$ and that cd is not a commutator:

```
gap> Size(Set(List(Cartesian(G,G), Comm)));
15
gap> c*d in Difference(D,\
> Set(List(Cartesian(G,G), Comm)));
true
```

2.2 Homomorphisms

Now we work with group homomorphisms. There are several ways to construct group homomorphisms. The function `GroupHomomorphismByImages` returns the group homomorphism constructed from a list of generators of the domain and the value of the image at each generator. Properties of group homomorphisms can be studied with `Image`, `IsInjective`, `IsSurjective`, `Kernel`, `PreImage`, `PreImages`, etc.

Example 2.24. The map $\mathbb{S}_4 \to \mathbb{S}_3$ that maps each transposition of $\mathbb{S}_4$ into (12) extends to a group homomorphism f. This homomorphism f is not injective ($\ker f$ has twelve elements) and it is not surjective (for example $(123) \notin f(\mathbb{S}_4)$):

```
gap> S4 := SymmetricGroup(4);;
gap> S3 := SymmetricGroup(3);;
gap> f := GroupHomomorphismByImages(S4, S3,\
> [(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)],\
> [(1,2),(1,2),(1,2),(1,2),(1,2),(1,2)]);
[ (1,2), (1,3), (1,4), (2,3), (2,4), (3,4) ] ->
[ (1,2), (1,2), (1,2), (1,2), (1,2), (1,2) ]
gap> Size(Kernel(f));
12
gap> IsInjective(f);
false
gap> Size(Image(f));
2
gap> IsSurjective(f);
false
gap> (1,2,3) in Image(f);
false
```

If K is a normal subgroup of G, the canonical map $G \to G/K$ can be constructed with the function `NaturalHomomorphismByNormalSubgroup`.

Example 2.25. Let us construct $C_{12} = \langle g : g^{12} = 1 \rangle$ as a group of permutations, the subgroup $K = \langle g^6 \rangle$ and the quotient C_{12}/K. We also construct the canonical (surjective) map $C_{12} \to C_{12}/K$:

```
gap> g := (1,2,3,4,5,6,7,8,9,10,11,12);;
gap> C12 := Group(g);;
gap> K := Subgroup(C12, [g^6]);;
gap> f := NaturalHomomorphismByNormalSubgroup(C12, K);
[ (1,2,3,4,5,6,7,8,9,10,11,12) ] -> [ f1 ]
gap> Image(f, g^6);
<identity> of ...
```

The function `AutomorphismGroup` computes the automorphism group of a finite group. If G is a group, the automorphisms of G of the form $x \mapsto g^{-1}xg$, where $g \in G$, are the **inner automorphisms** of G. The function `IsInnerAutomorphism` checks whether a given automorphism is inner.

Example 2.26. Let us check that $\mathrm{Aut}(\mathbb{S}_3)$ is a non-abelian group of six elements:

```
gap> aut := AutomorphismGroup(SymmetricGroup(3));
<group of size 6 with 2 generators>
gap> IsAbelian(aut);
false
```

Example 2.27. Let us prove that for $n \in \{2,3,4,5\}$ each automorphism of $\mathbb{S}_n$ is inner. Here is the code:

```
gap> for n in [2..5] do
> G := SymmetricGroup(n);;
> if ForAll(AutomorphismGroup(G),\
> x->IsInnerAutomorphism(x)) then
> Print("Each automorfism of S",\
> n, " is inner.\n");
> fi;
> od;
Each automorphism of S2 is inner.
Each automorphism of S3 is inner.
Each automorphism of S4 is inner.
Each automorphism of S5 is inner.
```

It is known that in $\mathbb{S}_6$ there are non-inner automorphisms:

```
gap> S6 := SymmetricGroup(6);;
gap> f := First(AutomorphismGroup(S6),\
> x->IsInnerAutomorphism(x)=false);
[ (1,2,3,4,5,6), (1,2) ] -> [ (2,3)(4,6,5), (1,2)(3,5)(4,6) ]
```

The automorphism of $\mathbb{S}_6$ such that $(123456) \mapsto (23)(465)$ and $(12) \mapsto (12)(35)(46)$ is not inner. Let us compute this homomorphism in the transpositions:

```
gap> for t in ConjugacyClass(S6, (1,2)) do
> Print("f(", t, ")=", Image(f,t), "\n");
> od;
f((1,2))=(1,2)(3,5)(4,6)
f((1,3))=(1,6)(2,5)(3,4)
f((1,4))=(1,4)(2,3)(5,6)
f((1,5))=(1,5)(2,4)(3,6)
f((1,6))=(1,3)(2,6)(4,5)
f((2,3))=(1,3)(2,4)(5,6)
f((2,4))=(1,5)(2,6)(3,4)
f((2,5))=(1,6)(2,3)(4,5)
f((2,6))=(1,4)(2,5)(3,6)
f((3,4))=(1,2)(3,6)(4,5)
f((3,5))=(1,4)(2,6)(3,5)
f((3,6))=(1,5)(2,3)(4,6)
f((4,5))=(1,3)(2,5)(4,6)
f((4,6))=(1,6)(2,4)(3,5)
f((5,6))=(1,2)(3,4)(5,6)
```

With `AllHomomorphisms` one constructs the set of group homomorphisms between two given groups. Similarly, one uses `AllEndomorphisms` to compute endomorphisms.

Example 2.28. We prove that there are ten endomorphisms of $\mathbb{S}_3$:

```
gap> S3 := SymmetricGroup(3);;
gap> Size(AllEndomorphisms(S3));
10
```

Example 2.29. The center of $C_2 \times \mathbb{S}_3$ is not stable under endomorphisms of $C_2 \times \mathbb{S}_3$. We see that $Z(C_2 \times \mathbb{S}_3) = \{\mathrm{id}, (12)\}$ and that there exists at least one endomorphism of $C_2 \times \mathbb{S}_3$ that permutes the non-trivial element of the center:

```
gap> C2 := CyclicGroup(IsPermGroup, 2);;
gap> S3 := SymmetricGroup(3);;
gap> C2xS3 := DirectProduct(C2, S3);;
gap> Center(C2xS3);
Group([ (1,2) ])
gap> ForAll(AllEndomorphisms(C2xS3),\
> f->Image(f,(1,2)) in [(), (1,2)]);
false
```

With `InnerAutomorphismsAutomorphismGroup` one constructs the inner automorphism group of a given group.

Example 2.30. Let us check that $\mathrm{Aut}(\mathbb{S}_6)/\mathrm{Inn}(\mathbb{S}_6) \simeq C_2$:

```
gap> S6 := SymmetricGroup(6);;
gap> A := AutomorphismGroup(S6);;
gap> Size(A);
1440
gap> I := InnerAutomorphismsAutomorphismGroup(A);;
gap> Order(A/I);
2
```

2.3 SmallGroups

GAP contains a database with all groups of certain small orders. The groups are sorted by their orders and they are listed up to isomorphism. This database is part of a library named `SmallGroups`. It contains the following groups: a) those of order ≤ 2000 except order 1024, b) those of cube-free order ≤ 50000, c) those of order p^7 for $p \in \{3,5,7,11\}$, d) those of order p^n for $n \leq 6$ and all primes p, e) those of order $q^n p$ for q^n dividing 2^8, 3^6, 5^5 or 7^4 and all primes p with $p \neq q$, f) those of square-free order, and g) those whose order factorizes into at most three primes.

The library was written by H, Besche, B. Eick and E. O'Brien.

As one can image, this library is a very useful tool when one needs to look for examples and counterexamples.

Example 2.31. Let us see what `SmallGroups` knows about groups of order twelve:

```
gap> SmallGroupsInformation(12);

  There are 5 groups of order 12.
    1 is of type 6.2.
    2 is of type c12.
    3 is of type A4.
    4 is of type D12.
    5 is of type 2^2x3.

  The groups whose order factorises in at most 3 primes
  have been classified by O. Hoelder. This classification is
  used in the SmallGroups library.

  This size belongs to layer 1 of the SmallGroups library.
  IdSmallGroup is available for this size.
```

Some of the examples of this section are from [10].

Example 2.32. Let us check that there exist non-abelian groups of odd order and that the smallest of this group has order 21:

```
gap> First(AllSmallGroups(Size, [1, 3..21]),\
> x->not IsAbelian(x));;
gap> Size(last);
21
```

Example 2.33. In one line we check that there are no simple groups of order 84. We use the filter `IsSimple` with the function `AllSmallGroups`:

```
gap> AllSmallGroups(Size, 84, IsSimple, true);
[  ]
```

With the function `StructureDescription` one explores the structure of a given group. The function returns a short string which gives some insight into the structure of the group.

Example 2.34. Let us see how the groups of order twelve look like:

```
gap> List(AllSmallGroups(Size, 12),\
> StructureDescription);
[ "C3 : C4", "C12", "A4", "D12", "C6 x C2" ]
```

The group `C3 : C4` denotes the semidirect product $C_3 \rtimes C_4$.

Example 2.35. Let us explore more group homomorphisms. We know that the symmetric group $\mathbb{S}_4$ is generated by the transpositions (12), (23) and (34). We let f be the group homomorphism $\mathbb{S}_4 \to \mathbb{S}_3$ given by $(12) \mapsto (12)$, $(23) \mapsto (23)$ and $(34) \mapsto (12)$. Let us perform some calculations related to this group homomorphism:

```
gap> S4 := SymmetricGroup(4);;
gap> S3 := SymmetricGroup(3);;
gap> f := GroupHomomorphismByImages(S4, S3, [(1,2),(2,3),(3,4)],\
>  [(1,2),(2,3),(1,2)]);;
gap> K := Kernel(f);;
gap> StructureDescription(K);
"C2 x C2"
gap> IsInjective(f);
false
```

```
gap> StructureDescription(S4/K);
"S3"
gap> StructureDescription(Image(f));
"S3"
gap> IsSurjective(f);
true
```

It is important to remark that the string returned by `StructureDescription` is not an isomorphism invariant: non-isomorphic groups can have the same string value and two isomorphic groups in different representations can produce different strings.

Example 2.36. There are two groups of order 20 that can be written as a semidirect product $C_5 \rtimes C_4$. `StructureDescription` will not distinguish such groups:

```
gap> List(AllSmallGroups(Size, 20),\
> StructureDescription);
[ "C5 : C4", "C20", "C5 : C4", "D20", "C10 x C2" ]
```

To identify groups in the database `SmallGroups` one uses the function `IdGroup`. Here we have some examples:

```
gap> IdGroup(SymmetricGroup(3));
[ 6, 1 ]
gap> IdGroup(SymmetricGroup(4));
[ 24, 12 ]
gap> IdGroup(AlternatingGroup(4));
[ 12, 3 ]
gap> IdGroup(DihedralGroup(8));
[ 8, 3 ]
gap> IdGroup(QuaternionGroup(8));
[ 8, 4 ]
```

Example 2.37. In [9], T. Lam and D. Leep proved that each index-two subgroup of $\mathrm{Aut}(\mathbb{S}_6)$ is isomorphic either to $\mathbb{S}_6$, $\mathbf{PGL}_2(9)$ or to the Mathieu group M_{10}. Let us check this claim using the function `IdGroup`:

```
gap> autS6 := AutomorphismGroup(SymmetricGroup(6));;
gap> lst := SubgroupsOfIndexTwo(autS6);;
gap> List(lst, IdGroup);
[ [ 720, 764 ], [ 720, 763 ], [ 720, 765 ] ]
gap> IdGroup(PGL(2,9));
[ 720, 764 ]
gap> IdGroup(MathieuGroup(10));
[ 720, 765 ]
gap> IdGroup(SymmetricGroup(6));
[ 720, 763 ]
```

Example 2.38. Now we prove a theorem of R. Guralnick [8]. The theorem states that the smallest finite group G such that $\{[x,y] : x,y \in G\} \neq [G,G]$ has order 96.

```
gap> G := First(AllSmallGroups(Size, [1..100]),\
> x->Order(DerivedSubgroup(x))<>Size(\
> Set(List(Cartesian(x,x), Comm))));;
gap> Order(G);
96
gap> IdGroup(G);
[ 96, 3 ]
```

With `IdGroup` (or with `IsomorphismGroups`) we check that

$$G \simeq \langle (1\,3\,5)(2\,4\,6)(7\,11\,9)(8\,12\,10), (3\,9\,4\,10)(5\,8)(6\,7)(11\,12)\rangle.$$

How did we find this isomorphism? We have constructed our group G. Then we use the function `IsomorphismPermGroup` to construct a faithful representation of G as a permutation group. With `SmallerDegreePermutationRepresentation` we construct (if possible) an isomorphic permutation group of smaller degree. Be aware that this new degree may not be minimal. After some attempts, we obtain an isomorphic copy of G inside $\mathbb{S}_{12}$. To construct a set of generators we then use `SmallGeneratingSet`. Again, be aware that this set may not be minimal.

For a finite group G let $\operatorname{cs}(G)$ denote the set of sizes of the conjugacy classes of G, that is

$$\operatorname{cs}(G) := \{|g^G| : g \in G\}.$$

Let us write a function to compute cs. With this function we show that

$$\operatorname{cs}(\mathbb{S}_3) = \{1,2,3\}, \quad \operatorname{cs}(\mathbb{A}_5) = \{1,12,15,20\}, \quad \operatorname{cs}(\mathbf{SL}_2(3)) = \{1,4,6\}.$$

Here is the code:

```
gap> cs := function(group)
> return Set(List(ConjugacyClasses(group), Size));
> end;
function( group ) ... end
gap> cs(SymmetricGroup(3));
[ 1, 2, 3 ]
gap> cs(AlternatingGroup(5));
[ 1, 12, 15, 20 ]
gap> cs(SL(2,3));
[ 1, 4, 6 ]
```

We will write $G_{n,k}$ to denote the k-th group of size n in the database, thus $G_{n,k}$ is a group with `IdGroup` equal to `[ n, k ]`.

Example 2.39. This example is taken from [13, Theorem A] and answers a question made by R. Brauer [4, Question 2(ii)]. G. Navarro proved that there exist finite groups G and H such that G is solvable, H is not solvable and $\operatorname{cs}(G) = \operatorname{cs}(H)$.

Let $G = G_{240,13} \times G_{960,1019}$ and $H = G_{960,239} \times G_{480,959}$. Then G is solvable and H is not. Moreover, $\operatorname{cs}(G) = \operatorname{cs}(H)$ as the following code shows:

```
gap> U := SmallGroup(960,239);;
gap> V := SmallGroup(480,959);;
gap> L := SmallGroup(960,1019);;
gap> K := SmallGroup(240,13);;
gap> UxV := DirectProduct(U,V);;
gap> KxL := DirectProduct(K,L);;
gap> IsSolvable(UxV);
false
gap> IsSolvable(KxL);
true
```

One could try to compute $\mathrm{cs}(U \times V)$ directly. However, this calculation seems to be hard. The trick is to use that $\mathrm{cs}(U \times V) = \{nm : n \in \mathrm{cs}(U), m \in \mathrm{cs}(V)\}$.

Here is the code:

```
gap> cs(KxL)=Set(List(Cartesian(cs(U),cs(V)), x->x[1]*x[2]));
true
```

Example 2.40. This example appeared in [13]. It answers another R. Brauer question [4, Question 4(ii)]. G. Navarro proved that there exist finite groups G and H such that G is nilpotent, $Z(H) = 1$ and $\mathrm{cs}(G) = \mathrm{cs}(H)$.

The groups are $G = \mathbb{D}_8 \times G_{243,26}$ and $H = G_{486,36}$. Here is the code:

```
gap> K := DihedralGroup(8);;
gap> L := SmallGroup(243,26);;
gap> H := SmallGroup(486,36);;
gap> IsTrivial(Center(H));
true
gap> G := DirectProduct(K,L);;
gap> cs(G)=cs(H);
true
gap> IsNilpotent(G);
true
```

2.4 Finitely presented groups

Let us start working with free groups. The function `FreeGroup` construct the free group in a finite number of generators.

Example 2.41. We create the free group F_2 in two generators and we create some random elements with the function `Random`:

```
gap> f := FreeGroup(2);
<free group on the generators [ f1, f2 ]>
gap> f.1^2;
f1^2
gap> f.1^2*f.1;
f1^3
```

```
gap> f.1*f.1^(-1);
<identity ...>
gap> Random(f);
f1^-3
```

Example 2.42. The function `Length` can be used to compute the length of words in a free group. In this example we create 10000 random elements in F_2 and compute their lengths.

```
gap> f := FreeGroup(2);;
gap> Collected(List(List([1..10000], x->Random(f)), Length));
[ [ 0, 2270 ], [ 1, 1044 ], [ 2, 1113 ],
  [ 3, 986 ], [ 4, 874 ], [ 5, 737 ],
  [ 6, 642 ], [ 7, 500 ], [ 8, 432 ],
  [ 9, 329 ], [ 10, 248 ], [ 11, 189 ],
  [ 12, 152 ], [ 13, 119 ], [ 14, 93 ],
  [ 15, 68 ], [ 16, 57 ], [ 17, 34 ],
  [ 18, 30 ], [ 19, 23 ], [ 20, 19 ],
  [ 21, 16 ], [ 22, 8 ], [ 23, 3 ], [ 24, 4 ],
  [ 25, 4 ], [ 26, 2 ], [ 27, 2 ], [ 28, 1 ],
  [ 31, 1 ] ]
```

Some of the functions we used before can also be used in free groups. Examples of these functions are `Normalizer`, `RepresentativeAction`, `IsConjugate`, `Intersection`, `IsSubgroup`, `Subgroup`.

Example 2.43. Here we perform some elementary calculations in F_2, the free group with generators a and b. We also compute the automorphism group of F_2.

```
gap> f := FreeGroup("a", "b");;
gap> a := f.1;;
gap> b := f.2;;
gap> Random(f);
b^-1*a^-5
gap> Centralizer(f, a);
Group([ a ])
gap> Index(f, Centralizer(f, a));
infinity
gap> Subgroup(f, [a,b]);
Group([ a, b ])
gap> Order(Subgroup(f, [a,b]));
infinity
gap> AutomorphismGroup(f);
<group of size infinity with 3 generators>
gap> GeneratorsOfGroup(AutomorphismGroup(f));
[ [ a, b ] -> [ a^-1, b ],
  [ a, b ] -> [ b, a ],
  [ a, b ] -> [ a*b, b ] ]
```

We now check that the subgroup S generated by a^2, b and aba^{-1} has index two in F_2. We compute $\mathrm{Aut}(S)$ and check that it is not a free group:

```
gap> S := Subgroup(f, [a^2, b, a*b*a^(-1)]);
Group([ a^2, b, a*b*a^-1 ])
gap> Index(f, S);
2
gap> A := AutomorphismGroup(S);
<group of size infinity with 3 generators>
gap> IsFreeGroup(A);
false
```

Example 2.44. Let $n \geq 3$ and $p \geq 2$ be integers. An amazing result of Coxeter [6] states that the group generated by $\sigma_1, \dots, \sigma_{n-1}$ and

$$\begin{aligned} &\sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1} && \text{if } i \in \{1,\dots,n-2\}, \\ &\sigma_i\sigma_j = \sigma_j\sigma_i && \text{if } |i-j| \geq 2, \\ &\sigma_i^p = 1 && \text{if } i \in \{1,\dots,n-1\}\rangle, \end{aligned}$$

is finite if and only if $(p-2)(n-2) < 4$.

We study the case $n = 3$. Let

$$G = \langle a, b : aba = bab,\, a^p = b^p = 1\rangle.$$

We claim that

$$G \simeq \begin{cases} \mathbb{S}_3 & \text{if } p = 2, \\ \mathbf{SL}_2(3) & \text{if } p = 3, \\ \mathbf{SL}_2(3) \rtimes C_4 & \text{if } p = 4, \\ \mathbf{SL}_2(3) \times C_5 & \text{if } p = 5: \end{cases}$$

Here is the proof:

```
gap> f := FreeGroup(2);;
gap> a := f.1;;
gap> b := f.2;;
gap> p := 2;;
gap> while p-2<4 do
> G := f/[a*b*a*Inverse(b*a*b), a^p, b^p];;
> Display(StructureDescription(G));
> p := p+1;
> od;
S3
SL(2,3)
SL(2,3) : C4
C5 x SL(2,5)
```

Example 2.45. For $l, m, n \in \mathbb{N}$, we define the **von Dyck group** (or triangular group) of type (l, m, n) as the group

$$G(l,m,n) = \langle a, b : a^l = b^m = (ab)^n = 1\rangle.$$

It is known that $G(l,m,n)$ is finite if and only if

$$\frac{1}{l}+\frac{1}{m}+\frac{1}{n}>1.$$

We claim that

$$G(2,3,3)\simeq\mathbb{A}_4,\quad G(2,3,4)\simeq\mathbb{S}_4,\quad G(2,3,5)\simeq\mathbb{A}_5.$$

Here is the code:

```
gap> f := FreeGroup(2);;
gap> a := f.1;;
gap> b := f.2;;
gap> StructureDescription(f/[a^2,b^3,(a*b)^3]);
"A4"
gap> StructureDescription(f/[a^2,b^3,(a*b)^4]);
"S4"
gap> StructureDescription(f/[a^2,b^3,(a*b)^5]);
"A5"
```

Example 2.46. This example is taken from [7]. Let us check that the group

$$\langle a,b,c : a^3=b^3=c^4=1, ac=ca^{-1}, aba^{-1}=bcb^{-1}\rangle$$

is trivial. For that purpose we use `IsTrivial`:

```
gap> f := FreeGroup(3);;
gap> a := f.1;;
gap> b := f.2;;
gap> c := f.3;;
gap> G := f/[a^3, b^3, c^4, c^(-1)*a*c*a, \
> a*b*a^(-1)*b*c^(-1)*b^(-1)];;
gap> IsTrivial(G);
true
```

Example 2.47. In [11] it is proved that for $n\in\mathbb{N}$,

$$\langle a,b : a^{-1}b^na=b^{n+1},\ a=a^{i_1}b^{j_1}a^{i_2}b^{j_2}\cdots a^{i_k}b^{j_k}\rangle,$$

is trivial if $i_1+i_2+\cdots i_k=0$. As an example let us see that

$$\langle a,b : a^{-1}b^2a=b^3, a=a^{-1}ba\rangle$$

is the trivial group:

```
gap> f := FreeGroup(2);;
gap> a := f.1;;
gap> b := f.2;;
gap> G := f/[a^(-1)*b^2*a*b^(-3),a*(a^(-1)*b*a)];;
gap> IsTrivial(G);
true
```

For each $n \geq 2$ the Burnside group $B(2,n)$ is defined as the group

$$B(2,n) = \langle a, b : w^n = 1 \text{ for all word } w \text{ in the letters } a \text{ and } b\rangle.$$

Example 2.48. We prove that the group $B(2,3)$ is a finite group of order 27. Let F be the free group of rank two. We divide F by the normal subgroup generated by $\{w_1^3,\dots,w_{10000}^3\}$, where $w_1,\dots,w_{10000}$ are some randomly chosen words of F. The following code shows that $B(2,3)$ is finite:

```
gap> f := FreeGroup(2);;
gap> rels := Set(List([1..10000], x->Random(f)^3));;
gap> G := f/rels;;
gap> Order(G);
27
```

Example 2.49. It is known that $B(2,4)$ is a finite group. Here we present here a computational proof. We use the same trick as before. The following code shows that $B(2,4)$ is finite and has order ≤ 4096:

```
gap> f := FreeGroup(2);;
gap> rels := Set(List([1..10000], x->Random(f)^4));;
gap> B24 := f/rels;;
gap> Order(B24);
4096
```

2.5 Problems

2.1. Compute the order of the group generated by

$$\begin{pmatrix}1&0\\0&1\end{pmatrix}, \qquad \begin{pmatrix}-1&0\\0&-1\end{pmatrix}, \qquad \begin{pmatrix}-1&0\\0&1\end{pmatrix}, \qquad \begin{pmatrix}1&0\\0&-1\end{pmatrix}.$$

Can you recognize this group?

2.2. Contruct the Heisenberg group $H(\mathbb{Z}/3)$.

2.3. Construct the Klein group as a group of permutations.

2.4. Prove that all subgroups of $C_4 \times Q_8$ are normal.

2.5. Let G be the set of matrices of the form

$$\begin{pmatrix}1&c\\0&d\end{pmatrix}, \quad c,d \in \mathbb{F}_4,\ c \neq 0.$$

Prove that G is a group and compute its order.

2.6. Use the function `IsomorphicSubgroups` to prove that $\mathbb{A}_6$ does not contain a subgroup isomorphic to $\mathbb{S}_5$ and that $\mathbb{A}_7$ contains a subgroup isomorphic to $\mathbb{S}_5$.

2.7. Prove that $\mathbb{A}_6$ does not contain subgroups of prime index.

2.8. Prove that $\mathbf{SL}_2(3)$ has a unique normal subgroup of order eight.

2.9. Find a subgroup of $\mathbf{SL}_2(5)$ isomorphic to $\mathbf{SL}_2(3)$.

2.10. Use the functions `SylowSubgroup` and `ConjugacyClassSubgroups` to construct all Sylow subgroups of $\mathbb{A}_4$ and $\mathbb{S}_4$.

2.11. Prove that $\mathbb{S}_5$ has a 2-Sylow subgroup isomorphic to the dihedral group of eight elements.

2.12. Can you recognize the structure of 2-Sylow subgroups of $\mathbb{S}_6$?

2.13. Use the function `Normalizer` to compute the number of conjugates of 2-Sylow subgroups of $\mathbb{A}_5$.

2.14. Find all Sylow subgroups of C_{27}, $\mathbf{SL}_2(5)$, $\mathbb{S}_7$, $\mathbb{S}_3 \times \mathbb{A}_4$ and $\mathbb{S}_3 \times C_{20}$.

2.15. Compute the conjugacy classes of subgroups of $\mathbb{S}_3 \times \mathbb{S}_3$ and find three p-Sylow subgroups, say A, B, C, such that $A \cap B = 1$ and $A \cap C \neq 1$.

2.16. Prove that the group

$$\left\{ \begin{pmatrix} 1 & b \\ 0 & d \end{pmatrix} : b, d \in \mathbb{F}_{19},\ d \neq 0 \right\}$$

is not simple.

2.17. Let G be the group generated by the permutations $(1\,2)(6\,11)(8\,12)(9\,13)$, $(5\,13\,9)(6\,10\,11)(7\,8\,12)$ and $(2,4,3)(5,8,9)(6\,10\,13)(7\,11\,12)$. How many elements of G are commutators?

2.18. Prove that $\mathbb{A}_4 \times C_7$ does not contain subgroups of index two.

2.19. Prove that $\mathbb{A}_5$ does not contain subgroups of order $8, 15, 20, 24, 30, 40$.

2.20. It is known that an abelian subgroup of $\mathbb{S}_n$ has order $\leq 3^{\lfloor n/3 \rfloor}$. How good is this bound? For $n \in \{5, 6, 7, 8\}$ find (if possible) an abelian subgroup of $\mathbb{S}_n$ of order $3^{\lfloor n/3 \rfloor}$.

2.21. Prove that $\mathbf{SL}_2(5)$ does not contain subgroups isomorphic to $\mathbb{A}_5$.

2.22. Prove that for each d that divides 24 there exists a subgroup of $\mathbb{S}_4$ of order d.

2.23. Prove that $\mathbf{SL}_2(3)$ contains a unique element of order two. Prove that $\mathbf{SL}_2(3)$ does not have subgroups of order 12.

2.24. Prove that the derived subgroup of $\mathbf{SL}_2(3)$ is isomorphic to Q_8.

2.25. Can you recognize the group $\mathbf{SL}_2(3)/Z(\mathbf{SL}_2(3))$?

2.26. Are the groups $\mathbb{S}_5$ and $\mathbf{SL}_2(5)$ isomorphic?

2.27. Let $\langle r^4 = s^2 = 1, srs = r^{-1}\rangle$ be the dihedral group of eight elements. Find all subgroups containing $\langle 1, r^2\rangle$.

2.28. Find all the group homomorphisms $\mathbb{S}_3 \to \mathbf{SL}_2(3)$.

2.29. Are there any surjective homomorphism $\mathbb{D}_{16} \to \mathbb{D}_8$? What about $\mathbb{D}_{16} \to C_2$?

2.30. Prove that $\mathrm{Aut}(\mathbb{A}_4) \simeq \mathbb{S}_4$.

2.31. Prove that $\mathrm{Aut}(\mathbb{D}_8) \simeq \mathbb{D}_8$ and that $\mathrm{Aut}(\mathbb{D}_{16}) \not\simeq \mathbb{D}_{16}$.

2.32. Compute the order of the group $\mathrm{Aut}(C_{11} \times C_2 \times C_3)$.

2.33. Prove that $\mathbb{D}_{12} \simeq \mathbb{S}_3 \times C_2$.

2.34. Prove that every group of order < 60 is solvable.

2.35. Let G be a group of order twelve such that $G \not\simeq \mathbb{A}_4$. Prove that G contains an element of order six.

2.36. Prove that a group of order 455 is cyclic.

2.37. Let G be a simple group of order 168. Compute the number of elements of order seven of G.

2.38. Prove that there are no simple groups of order 2540 and 9075.

2.39. Find a group G of order 3^6 such that $\{[x,y] : x,y \in G\} \neq [G,G]$.

2.40. Find a group G of order 2^7 such that $\{[x,y] : x,y \in G\} \neq [G,G]$

2.41. Prove that a group of order 15, 35 or 77 is cyclic.

2.42. Prove that a simple group of order 60 is isomorphic to $\mathbb{A}_5$.

2.43. Prove that the only non-abelian simple group of order < 100 is $\mathbb{A}_5$.

2.44. Is the following true? For any finite group G the set $\{x^2 : x \in G\}$ is a subgroup of G.

2.45. Prove the following theorem of Guralnick [8]. There exists a group G of order $n \leq 200$ such that $[G,G] \neq \{[x,y] : x,y \in G\}$ if and only if

$$n \in \{96, 128, 144, 162, 168, 192\}.$$

2.46. Prove the following extension of Guralnick's theorem (Problem 2.45). There exists a group G of order $n < 1024$ such that $[G,G] \neq \{[x,y] : x,y \in G\}$ if and only if n is one of the following numbers: 96, 128, 144, 162, 168, 192, 216, 240, 256, 270, 288, 312, 320, 324, 336, 360, 378, 384, 400, 432, 448, 450, 456, 480, 486, 504, 512, 528, 540, 560, 576, 594, 600, 624, 640, 648, 672, 702, 704, 720, 729, 744, 750, 756, 768, 784, 792, 800, 810, 816, 832, 840, 864, 880, 882, 888, 896, 900, 912, 918, 936, 960, 972, 1000, 1008.

2.47. Compute the list of normal subgroups of $\mathbf{GL}_2(3)$.

2.48. Compute the list of minimal subgroups of $\mathbb{A}_4$.

2.49. Compute the socle and the list of minimal normal subgroups of $\mathbb{A}_4$.

2.50. Compute the Fitting and the Frattini subgroup of $\mathbf{SL}_2(3)$.

2.51. Compute the list of all maximal normal subgroups of $\mathbf{SL}_2(3)$.

2.52. Prove that $\mathbf{PSL}_2(7)$ has a maximal subgroup of order 16.

2.53. Let G be a finite group and H be a subgroup. The **Chermak–Delgado** measure of H is the number $m_G(H) = |H||C_G(H)|$. Write a function to compute the Chermak–Delgado.

2.54. Compute $m_G(H)$ for $G \in \{\mathbb{S}_3, \mathbb{D}_8\}$ and H a subgroup of G.

2.55. Compute order the holomorph of $\mathbb{A}_4$. Find a permutation representation of small degree and find some minimal normal subgroup of order four. This is an exercise of [12].

2.56. Prove that the group $\langle (123\cdots7), (26)(34) \rangle$ is simple, has order 168 and acts transitively on $\{1,\dots,7\}$.

2.57. Compute the order of the group $\langle a,b : a^2 = b^2 = (bab^{-1})^3 = 1 \rangle$.

2.58. Prove that $\langle a,b : a^2 = aba^{-1}b = 1 \rangle$ is an infinite group.

2.59. Compute the order of the group $\langle a,b : a^8 = b^2a^4 = ab^{-1}ab = 1 \rangle$.

2.60. Can you recognize the group $\langle a,b : a^5 = 1, b^2 = (ab)^3, (a^3ba^4b)^2 = 1 \rangle$?

2.61. Prove that the group $\langle a,b : a^2 = b^3 = a^{-1}b^{-1}ab = 1 \rangle$ is finite and cyclic.

2.62. Prove that the group $\langle a,b : a^2 = b^3 = 1 \rangle$ is non-abelian.

2.63. Compute the order of the group

$$\langle a,b,c : a^3 = b^3 = c^3 = 1,\, aba = bab,\, cbc = bcb,\, ac = ca \rangle.$$

2.64. Prove that the group $\langle a,b,c : bab^{-1} = a^2,\, cbc^{-1} = b,\, aca^{-1} = c^2 \rangle$ is trivial. This is an exercise of Serre's book [14, §1]:

2.65. Prove that $B(2,3)$ is isomorphic to the Heisenberg group $H(\mathbb{Z}/3)$.

2.66. Find a permutation representation of the group $B(2,3)$.

2.67. Prove that $B(3,3)$ is a finite group of order ≤ 2187.

2.68. Let G be a finite group with k conjugacy classes. It is known that the probability that two elements of G commute is equal to $\mathrm{prob}(G) = k/|G|$. Compute this probability for $\mathbf{SL}_2(3)$, $\mathbb{A}_4$, $\mathbb{A}_5$, $\mathbb{S}_4$ and Q_8.

Chapter 3
Representations

3.1 Constructing representations

Let us construct irreducible representations of finite groups.

Example 3.1. Let us construct the representation ρ of $\mathbb{A}_4$ given by

$$(12)(34) \mapsto \begin{pmatrix} 0 & 1 & -1 \\ 1 & 0 & -1 \\ 0 & 0 & -1 \end{pmatrix}, \quad (123) \mapsto \begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & -1 \\ 1 & 0 & -1 \end{pmatrix}.$$

We use the function `GroupHomomorphismByImages`.

```
gap> A4 := AlternatingGroup(4);;
gap> a := [[0,1,-1],[1,0,-1],[0,0,-1]];;
gap> b := [[0,0,-1],[0,1,-1],[1,0,-1]];;
gap> rho := GroupHomomorphismByImages(A4,\
> [ (1,2)(3,4), (1,2,3) ], [ a, b ]);;
gap> IsGroupHomomorphism(rho);
true
```

This is indeed a faithful representation of $\mathbb{A}_4$.

```
gap> IsTrivial(Kernel(rho));
true
```

Just to see how it works, let us compute $\rho_{(132)}$, the image of $(132) \in \mathbb{A}_4$ under ρ. We are working with 3×3 matrices so it is better to use the function `Display`.

```
gap> Display(Image(rho, (1,3,2)));
[ [  -1,   0,   1 ],
  [  -1,   1,   0 ],
  [  -1,   0,   0 ] ]
```

Now we construct the character χ of ρ. We also check that ρ is irreducible since

$$\langle \chi, \chi \rangle = \frac{1}{12} \sum_{x \in \mathbb{A}_4} \chi(x)\chi(x^{-1}) = 1.$$

```
gap> chi := x->TraceMat(x^rho);;
gap> 1/Order(A4)*Sum(List(A4, x->chi(x)*chi(x^(-1))));
1
```

In [1, Problem 1], R. Brauer asked what algebras are group algebras. This question might be impossible to answer. However, we can play with some particular examples.

Example 3.2. Is $\mathbb{C}^5 \times M_5(\mathbb{C})$ a (complex) group algebra? The answer is no. To prove our claim, we can compute the degrees of the irreducible characters using `CharacterDegrees`. We list the degrees of irreducible characters of groups of order 30. We see that there are four groups of order 30 and none of them has a group algebra isomorphic to $\mathbb{C}^5 \times M_5(\mathbb{C})$.

```
gap> n := 30;;
gap> for G in AllGroups(Size, n) do
> Print(CharacterDegrees(G), "\n");
> od;
[ [ 1, 10 ], [ 2, 5 ] ]
[ [ 1, 6 ], [ 2, 6 ] ]
[ [ 1, 2 ], [ 2, 7 ] ]
[ [ 1, 30 ] ]
```

In fact, this shows that the groups algebras of groups of order 30 are

$$\mathbb{C}^{10} \times M_2(\mathbb{C})^5, \qquad \mathbb{C}^6 \times M_2(\mathbb{C})^6, \qquad \mathbb{C}^2 \times M_2(\mathbb{C})^7, \qquad \mathbb{C}^{30}.$$

How can we construct irreducible representations of a given group? This can be done with the package `Repsn`, written by Vahid Dabbaghian. Now let us play with examples:

Example 3.3. Let us construct the irreducible representations of $\mathbb{S}_3$. The irreducible characters of a finite group can be constructed with `Irr`:

```
gap> S3 := SymmetricGroup(3);;
gap> irr := Irr(S3);
[ Character( CharacterTable( Sym( [ 1 .. 3 ] ) ), [ 1, -1, 1 ] ),
  Character( CharacterTable( Sym( [ 1 .. 3 ] ) ), [ 2, 0, -1 ] ),
  Character( CharacterTable( Sym( [ 1 .. 3 ] ) ), [ 1, 1, 1 ] ) ]
```

To construct irreducible representations we need to load the package `repsn`:

```
gap> LoadPackage("repsn");
```

The package contains `IrreducibleAffordingRepresentation`. This function produces irreducible representations from irreducible characters. Since we are working with $\mathbb{S}_3$, we will only need to consider the character of degree two. We will produce the faithful represention $\mathbb{S}_3 \to \mathbf{GL}(2,\mathbb{C})$ given by

$$(123) \mapsto \begin{pmatrix} \omega^2 & 0 \\ 0 & \omega \end{pmatrix}, \quad (12) \mapsto \begin{pmatrix} 0 & \omega \\ \omega^2 & 0 \end{pmatrix}.$$

Here is the code:

```
gap> f := IrreducibleAffordingRepresentation(irr[2]);
[ (1,2,3), (1,2) ] -> [ [ [ E(3)^2, 0 ], [ 0, E(3) ] ],
  [ [ 0, E(3) ], [ E(3)^2, 0 ] ] ]
gap> Image(f, (1,2,3));
[ [ E(3)^2, 0 ], [ 0, E(3) ] ]
gap> Display(Image(f, (1,2,3)));
[ [  E(3)^2,       0 ],
  [       0,    E(3) ] ]
gap> Display(Image(f, (1,2)));
[ [       0,    E(3) ],
  [  E(3)^2,       0 ] ]
```

3.2 The McKay conjecture

For a finite group G and a prime p such that p divides $|G|$ one defines

$$\mathrm{Irr}_{p'}(G) = \{\chi \in \mathrm{Irr}(G) : p \nmid \chi(1)\}.$$

In 1970 J. McKay made the following amazing conjecture: if $P \in \mathrm{Syl}_p(G)$, then

$$|\,\mathrm{Irr}_{p'}(G)| = |\,\mathrm{Irr}_{p'}(N_G(P))|.$$

Example 3.4 (Testing the McKay conjecture). It is believed that the McKay conjecture is true. Let us check the conjecture in some small examples. We first write a function that checks the conjecture.

```
gap> McKay := function(G, p)
> local N, n, m;
> N := Normalizer(G, SylowSubgroup(G, p));
> n := Number(Irr(G), x->Degree(x) mod p <> 0);
> m := Number(Irr(N), x->Degree(x) mod p <> 0);
> if n = m then
> return n;
> else
> return false;
> fi;
> end;
function( G, p ) ... end
```

With this function is now easy to check the conjecture in several small examples. Let us check that the McKay conjecture is true for $\mathbf{SL}_2(3)$. This group has order 24, so we need to check the conjecture for 2 and 3:

```
gap> McKay(SL(2,3), 2);
4
gap> McKay(SL(2,3), 3);
6
```

For $k \in \mathbb{Z}$ and a finite group G let

$$M_k(G) = |\{\chi \in \operatorname{Irr} p'(G) : \chi(1) \equiv \pm k \bmod p\}|.$$

The Isaacs–Navarro conjecture states that

$$M_k(G) = M_k(N_G(P)).$$

Example 3.5 (Testing the Isaacs–Navarro conjecture). Here we have a function that checks the Isaacs–Navarro conjecture:

```
gap> IsaacsNavarro := function(G, k, p)
> local mG, mN, N;
> N := Normalizer(G, SylowSubgroup(G, p));
> mG := Number(Filtered(Irr(G), x->Degree(x)\
> mod p <> 0), x->Degree(x) mod p in [-k,k] mod p);
> mN := Number(Filtered(Irr(N), x->Degree(x)\
> mod p <> 0), x->Degree(x) mod p in [-k,k] mod p);
> if mG = mN then
> return mG;
> else
> return false;
> fi;
> end;
function( G, k, p ) ... end
```

Let us check that the Isaacs–Navarro conjecture is true for the group $\mathbf{SL}_2(3)$. We only need to check the conjecture for $k \in \{1,2\}$ and $p \in \{2,3\}$.

```
gap> IsaacsNavarro(SL(2,3), 1, 2);
4
gap> IsaacsNavarro(SL(2,3), 1, 3);
6
gap> IsaacsNavarro(SL(2,3), 2, 2);
0
gap> IsaacsNavarro(SL(2,3), 2, 3);
6
```

In 1951 Ore conjectured that in each finite non-abelian simple group G every element is a commutator. The conjecture was proved by Liebeck, O'Brien, Shalev and Tiep in 2010. The following example proves Ore's conjecture is true for sporadic simple groups.

Example 3.6 (Ore's conjecture for sporadic simple groups). A well-known result of Burnside states that for a finite group G, an element $g \in G$ is a commutator if and only if

$$\sum_{\chi \in \operatorname{Irr}(G)} \frac{\chi(g)}{\chi(1)} \neq 0.$$

We show a function that checks wether every element of a given group is is a commutator. The input is the character table of the group and the function returns **`true`** if every element of the group is a commutator or **`false`** otherwise.

```
gap> Ore := function(char)
> local s,f,k;
> for k in [1..NrConjugacyClasses(char)] do
> s := 0;
> for f in Irr(char) do
> s := s+f[k]/Degree(f);
> od;
> if s<=0 then
> return false;
> fi;
> od;
> return true;
> end;
function( char ) ... end
```

We use this function to verify Ore's conjecture for the five Mathieu groups:

```
gap> Ore(CharacterTable("M11"));
true
gap> Ore(CharacterTable("M12"));
true
gap> Ore(CharacterTable("M22"));
true
gap> Ore(CharacterTable("M23"));
true
gap> Ore(CharacterTable("M24"));
true
```

Similarly one verifies Ore's conjecture for all other sporadic simple groups.

3.3 Problems

3.1. Describe the groups algebras $\mathbb{C}[G]$ for G a finite group of order 28.

3.2. Construct the irreducible representations of $\mathbb{D}_8$ and $\mathbf{SL}_2(3)$.

3.3. Construct the irreducible representations of $\mathbb{A}_4$, $\mathbb{S}_4$ and $\mathbb{A}_5$.

3.4. Verify the McKay conjecture for the Mathieu Group M_{11}.

3.5. Verify the Ore conjecture for all the sporadic simple groups.

Chapter 4
Advanced group theory

4.1 Conjugacy classes

For a finite group G, let $\mathrm{cs}(G)$ denote the set of sizes of the conjugacy classes of G.

Example 4.1. Let us write a function to compute cs. With this function we show that

$$\mathrm{cs}(\mathbb{S}_3) = \{1,2,3\}, \quad \mathrm{cs}(\mathbb{A}_5) = \{1,12,15,20\}, \quad \mathrm{cs}(\mathbf{SL}_2(3)) = \{1,4,6\}.$$

Here is the code:

```
gap> cs := function(group)
> return Set(List(ConjugacyClasses(group), Size));
> end;
function( group ) ... end
gap> cs(SymmetricGroup(3));
[ 1, 2, 3 ]
gap> cs(AlternatingGroup(5));
[ 1, 12, 15, 20 ]
gap> cs(SL(2,3));
[ 1, 4, 6 ]
```

The following example is taken from [13, Theorem A] and answers a question made by R. Brauer [4, Question 2(ii)].

Example 4.2. G. Navarro proved that there exist finite groups G and H such that G is solvable, H is not solvable and $\mathrm{cs}(G) = \mathrm{cs}(H)$.

To prove this, let $G = G_{240,13} \times G_{960,1019}$ and $H = G_{960,239} \times G_{480,959}$. Then G is solvable and H is not. Moreover, $\mathrm{cs}(G) = \mathrm{cs}(H)$, as the following code shows:

```
gap> U := SmallGroup(960,2390);;
gap> V := SmallGroup(480,959);;
gap> L := SmallGroup(960,1019);;
gap> K := SmallGroup(240,13);;
gap> UxV := DirectProduct(U,V);;
gap> KxL := DirectProduct(K,L);;
gap> IsSolvable(UxV);
false
gap> IsSolvable(KxL);
true
```

One could try to compute $\text{cs}(U \times V)$ directly. However, this calculation seems to be hard. The trick is to use the fact that $\text{cs}(H) = \{uv : u \in \text{cs}(U), v \in \text{cs}(V)\}$. Here is the code:

```
gap> cs(G)=Set(List(Cartesian(cs(U),cs(V)), x->x[1]*x[2]));
true
```

The following result was proved by G. Navarro in [13]. It answers another question posed by R. Brauer [4, Question 4(ii)].

Example 4.3. G. Navarro proved that there exist finite groups G and H such that G is nilpotent, $Z(H) = 1$ and $\text{cs}(G) = \text{cs}(H)$.

The groups discovered by Navarro are $G = \mathbb{D}_8 \times G_{243,26}$ and $H = G_{486,36}$. Here is the code:

```
gap> K := DihedralGroup(8);;
gap> L := SmallGroup(243,26);;
gap> H := SmallGroup(486,36);;
gap> IsTrivial(Center(H));
true
gap> G := DirectProduct(K,L);;
gap> cs(G)=cs(H);
true
gap> IsNilpotent(G);
true
```

GAP contains several useful group databases. One of these databases is the `SmallGroups` library we found in Section 2.3 but this library is not the only one. It is also possible to work with transitive or primitive groups of small degree. Let us see some applications.

Example 4.4. Let us prove the following claim: There is no sharply 4-transitive group of degree seven or nine, see [3, Exercise 1.18]. We first create a list of all transitive groups of degrees seven and nine and we keep only those groups that are at least 4-transitive:

```
gap> l := Filtered(AllTransitiveGroups(NrMovedPoints, [7,9]), \\
> x->Transitivity(x)>3);
[ A7, S7, A9, S9 ]
```

Finally, we check that none of these groups are sharply 4-transitive. For that purpose, we see that the action on tuples is never regular:

```
gap> true in List(l, x->IsRegular(x, \\
> Arrangements([1..NrMovedPoints(x)], 4), OnTuples));
false
```

Now let us see some examples that use the primitive groups library:

Example 4.5. Let us prove the following claim: Primitive groups of degree eight are double transitive. First we note that there are seven primitive groups of degree eight:

```
gap> l := AllPrimitiveGroups(NrMovedPoints, 8);
[ AGL(1, 8), AGammaL(1, 8), ASL(3, 2),
  PSL(2, 7), PGL(2, 7), A(8), S(8) ]
```

To check that all these groups are indeed at least 2-transitive, we use the function `Transitivity`:

```
gap> List(l, Transitivity);
[ 2, 2, 3, 2, 3, 6, 8 ]
gap> ForAll(last, x->x>1);
true
```

The *commuting probability* of a finite group G is defined as the probability that a randomly chosen pair of elements of G commute, and it is thus equal to $k(G)/|G|$. In Problem 2.68 we asked to write a function that computes the commuting probability of a finite group. We first present a solution to this problem:

```
gap> p := x->NrConjugacyClasses(x)/Order(x);
function( x ) ... end
```

In 1970 J. C. Dixon observed that the commuting probability of a finite non-abelian simple group is $\leq 1/12$. This bound is attained for the alternating simple group $\mathbb{A}_5$:

```
gap> p(AlternatingGroup(5));
1/12
```

One can find Dixon's proof in a 1973 volume of the *Canadian Mathematical Bulletin*. The proof we present here is based on a proof due to Iván Sadofschi Costa. We first assume that the commuting probability of G is $> 1/12$. Since G is a non-abelian simple group, the identity is the only central element. Let us assume first that there is a conjugacy class of G of size m, where m is such that $1 < m \leq 12$. Then G is a transitive subgroup of $\mathbb{S}_m$. For these groups the problem is easy: we show that there are no non-abelian simple groups that act transitively on sets of size $m \in \{2,\dots,12\}$ with commuting probability $> 1/12$. To do this, we list these transitive groups and their commuting probabilities and verify that all commuting probabilities are $\leq 1/12$:

```
gap> l := AllTransitiveGroups(NrMovedPoints, [2..12], \\
> IsAbelian, false, \\
> IsSimple, true);;
[ A5, L(6) = PSL(2,5) = A_5(6), A6,
  L(7) = L(3,2), A7, L(8)=PSL(2,7), A8,
  L(9)=PSL(2,8), A9, A_5(10), L(10)=PSL(2,9),
  A10, L(11)=PSL(2,11)(11), M(11), A11, A_5(12),
  L(2,11), M_11(12), M(12), A12 ]
```

```
gap> List(l, p);
[ 1/12, 1/12, 7/360, 1/28, 1/280, 1/28, 1/1440,
  1/56, 1/10080, 1/12, 7/360, 1/75600, 2/165,
  1/792, 31/19958400, 1/12, 2/165, 1/792, 1/6336,
  43/239500800 ]
gap> ForAny(l, x->p(x)>1/12);
false
```

Now assume that all non-trivial conjugacy class of G have at least 13 elements. Then the class equation implies that

$$|G| \geq \frac{13}{12}|G| - 12,$$

and therefore $|G| \leq 144$. Thus one needs to check what happens with groups of order ≤ 144. But we know that the only non-abelian simple group of size ≤ 144 is the alternating simple group $\mathbb{A}_5$.

```
gap> AllGroups(Size, [2..144], \\
> IsAbelian, false, \\
> IsSimple, true);
[ Alt( [ 1 .. 5 ] ) ]
```

4.2 Simple groups

What if we want to work with some simple groups?

4.3 Problems

References

1. R. Brauer. Representations of finite groups. In *Lectures on Modern Mathematics, Vol. I*, pages 133–175. Wiley, New York, 1963.
2. M. Brennan and D. Machale. Variations on a Theme: A_4 Definitely Has no Subgroup of Order Six! *Math. Mag.*, 73(1):36–40, 2000.
3. P. J. Cameron. *Permutation groups*, volume 45 of *London Mathematical Society Student Texts*. Cambridge University Press, Cambridge, 1999.
4. A. R. Camina and R. D. Camina. The influence of conjugacy class sizes on the structure of finite groups: a survey. *Asian-Eur. J. Math.*, 4(4):559–588, 2011.
5. R. D. Carmichael. *Introduction to the theory of groups of finite order*. Dover Publications, Inc., New York, 1956.
6. H. S. M. Coxeter. *Kaleidoscopes*. Canadian Mathematical Society Series of Monographs and Advanced Texts. John Wiley & Sons, Inc., New York, 1995. Selected writings of H. S. M. Coxeter, Edited by F. Arthur Sherk, Peter McMullen, Anthony C. Thompson and Asia Ivić Weiss, A Wiley-Interscience Publication.
7. P. de la Harpe. *Topics in geometric group theory*. Chicago Lectures in Mathematics. University of Chicago Press, Chicago, IL, 2000.
8. R. M. Guralnick. Commutators and commutator subgroups. *Adv. in Math.*, 45(3):319–330, 1982.
9. T. Y. Lam and D. B. Leep. Combinatorial structure on the automorphism group of S_6. *Exposition. Math.*, 11(4):289–308, 1993.
10. D. MacHale. Minimum counterexamples in group theory. *Math. Mag.*, 54(1):23–28, 1981.
11. C. F. Miller, III and P. E. Schupp. Some presentations of the trivial group. In *Groups, languages and geometry (South Hadley, MA, 1998)*, volume 250 of *Contemp. Math.*, pages 113–115. Amer. Math. Soc., Providence, RI, 1999.
12. R. F. Morse. On the application of computational group theory to the theory of groups. In *Computational group theory and the theory of groups*, volume 470 of *Contemp. Math.*, pages 1–19. Amer. Math. Soc., Providence, RI, 2008.
13. G. Navarro. The set of conjugacy class sizes of a finite group does not determine its solvability. *J. Algebra*, 411:47–49, 2014.
14. J.-P. Serre. *Trees*. Springer Monographs in Mathematics. Springer-Verlag, Berlin, 2003. Translated from the French original by John Stillwell, Corrected 2nd printing of the 1980 English translation.

Index

Automorphism group, 48

Brauer, R., 64
Burnside group, 58, 62

Center, 41
Centralizer, 42
Chermak–Delgado subgroup, 61
Collatz
  conjecture, 12
Commutator, 42
Conjugacy class, 42
Conway, J., 32
Coxeter, H., 56

Eigenvalue, 34
Eigenvector, 34

Fibonacci sequence, 12, 21, 27
Fractran programming language, 32
Free group, 54

Group
  alternating, 38
  cyclic, 37
  dihedral, 37
  homomorphism, 47
Guralnick, R., 60, 61

Heisenberg group, 62
Hilbert matrix, 34

Index, 40
Inner automorphisms, 48
Isaacs, M., 66

Kronecker product, 34
Kronecker, L., 34

Lam, T., 52
Leep, D., 52
Look and say sequence, 32

Matrix
  basic operations, 25
  determinant of a, 28
  diagonal, 25
  null, 25
  number of columns, 24
  number of rows, 24
  over a finite field, 26
  rank of a, 28
  submatrices of a, 26
  trace of a, 28
McKay conjecture, 65
McKay, J., 65

Navarro, G., 53, 66, 69

Order
  of elements, 39
Ore
  conjecture, 66

Permutation, 22
  sign of a, 23

Subgroups, 40

Tensor product
  of matrices, 34
Theorem
  of Liebeck–O'Brien–Shalev–Tiep, 66

Vectors
  basic operations, 24

Walsh matrix, 34