Mini-course on GAP – Lecture 3

Leandro Vendramin

Universidad de Buenos Aires

Dalhousie University, Halifax
January 2020

We continue with basic GAP programming. Now we are about to learn about loops. Our presentation 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
```

Loops

An equivalent way of doing this uses **`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 uses **`while ... do ... od`**:

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

Loops

Yet another equivalent way of doing this uses **`repeat ... until`**:

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

Loops

Now let us compute (again) Fibonacci numbers. This is better than the method we used before. 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
```

Loops

Is it really better than the previous function for computing Fibonacci numbers? Of course it is!

```
gap> fibonacci(100);
354224848179261915075
gap> fibonacci(1000);
43466557686937456435688527675040625802564660517371780402481729089536555417949051890403879840079255169295922593080322634775209689623239873322471161642996440906533187938298969649928516003704476137795166849228875
```

For computing Fibonacci numbers there an ever better solution! An easy induction exercise shows that (f_n) can be computed using

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

Loops

We use this clever 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)>
```

Loops

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

Loops

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

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

Loops

With **break** one breaks a loop. In the following example we run over the numbers $1, 2, \ldots, 100$ 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
```

Loops

`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.

```
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
```

Now it is time to work with groups. We start with some elementary constructions.

Groups

One constructs groups with the function `Group`. We compute the order of the following groups:

- The group generated by the transposition (12)
- The group generated by the 5-cycle (12345)
- 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
```

Groups

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.

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) ])
```

Recall that a matrix group is a subgroup of $\mathbf{GL}(n, K)$ for some $n \in \mathbb{N}$ and some field K. A permutation group is a subgroup of some Sym_n.

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.

Groups

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) ]
```

Groups

One constructs the symmetric group Sym_n with `SymmetricGroup`.
To construct the alternating group Alt_n one uses `AlternatingGroup`.
The elements of Sym_n are permutations of the set $\{1, \dots, n\}$.

Groups

Let us construct Alt_4 and Sym_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) ]
```

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
```

Let us check that Sym_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 ] ]
```

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 the imaginary unit $i = \sqrt{-1}$ is `E(4)`.

```
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
```

Groups

To check that $G \not\cong \mathbb{D}_8$ we see that G contains a unique element of order two and $\mathbb{D}_8$ has five elements of order two:

```
gap> Number(G, x->Order(x)=2);
1
gap> Number(DihedralGroup(8), x->Order(x)=2);
5
```

The Mathieu group M_{11} is a simple group of order 7920. It can be defined as the subgroup of Sym_{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
```

Groups

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
```

Not always we 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`.

The subgroup of Sym_3 generated by (12) is $\{\mathrm{id}, (12)\}$ and has index three. The subgroup of Sym_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
```

Groups

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. With `IsSubgroup` we check that Alt_4 is a subgroup of Sym_4. With `IsNormal` we see that Alt_4 is a subset of Sym_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 Sym_4 generated by (123) is not normal in Sym_4:

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

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> 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
```

Groups

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
```

Groups

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. We can use the filters `IsPermGroup` (resp. `IsMatrixGroup`) to obtain generalized quaternion groups as permutation (resp. matrix) groups.

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.

Groups

Let us check that Alt_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) ]
```

Direct products of groups are constructed with `DirectProduct`. Example: the groups $C_4 \times C_4$ and $C_2 \times Q_8$ have both order 16, have both 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> Collected(List(C4xC4, Order));
[ [ 1, 1 ], [ 2, 3 ], [ 4, 12 ] ]
gap> Collected(List(C2xQ8, Order));
[ [ 1, 1 ], [ 2, 3 ], [ 4, 12 ] ]
```

Are these two 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
```

Alternatively:

```
gap> IsomorphismGroups(C4xC4,C2xQ8);
fail
```

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

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

`ConjugacyClass` computes a conjugacy class The centralizer can be computed with `Centralizer`.

Let us check that Sym_3 contains three conjugacy classes with representatives id, (12) and (123), so that

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

```
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 $C_{\mathrm{Sym}_3}((123)) = \{\mathrm{id},(123),(132)\}$:

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

In this example we use the function `Representative` to construct a list of representatives of conjugacy classes of Alt_4:

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

Groups

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`.

Let us check that (123) and $(132) = (123)^2$ are not conjugate in Alt_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 Alt_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
```

It is well-known that the converse of Lagrange theorem does not hold. Let us show that Alt_4 has no subgroups of order six.

A naive idea to prove that Alt_4 has no subgroups of order six is to study all the $\binom{12}{6} = 924$ subsets of Alt_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
```

Here we have another idea: if Alt_4 has a subgroup of order six, then the index of this subgroup in Alt_4 is two. With `SubgroupsOfIndexTwo` we check that Alt_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
```

It is enough to construct all conjugacy classes of subgroups!

An exercise on commutators

It is known that the commutator of a finite group is not always equal to the set of commutators. Carmichael's book[1] shows the following example: Let G be the subgroup of Sym_{16} generated by the permutations

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

Show that $[G, G]$ has order 16 and that the set of commutators has 15 elements. In particular, one can show that $cd \in [G, G]$ and that cd is not a commutator.

[1] Introduction to the theory of groups of finite order. Dover Publications, Inc., New York, 1956

An exercise on commutators

Here is the solution:

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