

## 1 Introduction

GAP (for Groups, Algorithms, and Programming), a powerful package for doing computational discrete algebra, was developed at the Lehrstuhl D fuer Mathematik in Aachen, Germany and is now maintained at the University of St. Andrews, in St. Andrews, Scotland. This project was begun in 1985 and currently consists of a self-contained programming language and over 800 built-in functions for the study of algebraic objects. It is distributed free of charge to the mathematical community. GAP's home page is

<http://www-history.mcs.st-and.ac.uk/gap/>

where you can find further information. If you would like to install it on a machine that you have, I will be willing to help. It is available for PC's running a 32-bit operating system (Like Windows 98), for Macintosh's running MacOS, and for any Unix platform. We will be using the implementation that runs under Linux.

Joachim Neubüser (the original director of the GAP project) had the following credo:

Nobody has ever paid a licence fee for using a proof  
that shows Sylow's subgroups to exist  
Nobody should ever pay a licence fee for using a program  
that computes Sylow's subgroups.

This credo explains, at least in part, why such a huge and useful package as GAP is free for the usage.

## 2 Signing on to GAP

Go to any computer in OM304.

- Place the Linux boot disk in the floppy drive, press Ctrl, Alt, and Del at the same time, click on Shutdown on the next window, and click on Shutdown and Restart on the next window.
- When the boot process has ended you will see something like

```
Panther2178 login:
```

In response to this, enter your username. (You do not need a password for your account the first time you sign on, but after signing on, you should change your password by issuing the command

```
passwd
```

and proceeding as directed.)

- Obtain a copy of the GAP lab for your own use by entering the command

```
cp -r /home1/mat3530/lab1 .
```

Be sure you type this command exactly as shown including the space between `lab1` and the period at the end of the command. This command will copy all of the relevant files to your account. *If you need to return to the lab at a later time to work on this lab, you do not need to get another copy (and you do not need to log on to the same computer).*

- Change to the `lab1` folder of your account by

```
cd lab1
```

- Start the X Windows environment by entering

`startx`

Eventually you will get a screen with a K in the lower left corner. (Respond to any warning messages which appear by clicking on the `Oops` button.)

- Start an XGAP session. Find the icon on the lower menu which looks like a computer monitor with a attached screen and single click on it. Once a terminal window appears enter the command

```
xgap &
```

at the command line. Note: XGAP and GAP differ only in that XGAP has a graphical user interface. When the window labelled `xgap` appears you are ready to go.

- If you want access to the on-line help, start another terminal session as before. In the resulting terminal window enter

```
netscape &
```

at the command line.

- Once Netscape appears, go to the following URL

```
http://www.ux1.eiu.edu/~cfdmb/gap/htm
```

This connects you to on-line documentation. You can bounce between the `xgap` window and the Netscape window by moving the cursor to the appropriate window and left-clicking.

- For directions on signing out, see the section labelled **Finishing up**.

### 3 Language Fundamentals

There is a tremendous amount of power in GAP.

- Commands can be entered directly in GAP. The up-arrow key can then be used to recall previously entered commands which can then be edited.
- Almost every line in GAP must end with a semicolon. The examples below give a few counterexamples to the general rule.
- A basic data structure in GAP is a set. To denote a set, give a comma separated listing of the members enclosed by square braces.
- In GAP, the symbol `:=` is used for assignment, the symbol `=` being reserved for equality.
- The standard GAP prompt is `gap>`. However, if you are entering a multi-line command the prompt becomes `>`. Sometimes, an error will occur and the prompt will change to `brk>`. You can return to the standard prompt by pressing the `Ctrl` and the `D` keys simultaneously.

### 4 Starting out

There is a convenient way to keep a copy of your GAP session. Issuing the command

```
LogTo("session1.gap");
```

causes your entire session to be recorded in the file `session1.gap` which you can edit later on. Before proceeding, read the section **Finishing Up** to see how the session will end .

As a first task in GAP, read in a file which will contains the definitions of functions needed in this lab.

```
Read("./lab1/lab1functions.gap");
```

Listing for this and all other files needed for this lab are given in the Appendix.

## 5 Examples

### 5.1 Loops, Printing, and Equations

You will be analyzing several groups in this lab. Define the first, called by alternating group on 4 letters, by entering the following command

```
Read("./lab1/a4.gap");
```

No output should result from this command, but you will be able to see the group that has been defined when you enter

```
Elements(G);
```

How many elements are there in this set? You could count them by hand or have GAP do the work with

```
Size(Elements(G));
```

Now let's calculate all squares of elements and print them out, one line for each element. This can be done by entering

```
for x in Elements(G) do
  Print(x, " ", x^2, "\n");
od;
```

The spacing and the extra lines are not necessary, but they make the input easier to read. Enter the commands above and see the output. With the prompts, your entry should look like

```
gap> for x in Elements(G) do
>   Print(x, " ", x^2, "\n");
> od;
```

The output from the last command does not look quite right because the squares are more complicated than the original elements. Some simplification is missing. What has happened is that the squares have not been simplified. The `Normal` function does this simplification. In particular `Normal(a^2,G)` will simplify `a^2` to an element of `Elements(G)`. The following commands which you should also enter will compute the squares and print out the elements and their simplified squares.

```
for x in Elements(G) do
  Print(x, " ", Normal(x^2,G), "\n");
od;
```

A note about the `Print` command is in order. The parameters passed to this function can include the names of objects and the values of functions, e.g. `x` and `Normal(x^2,G)`, as well as character strings, e.g. `" "` (which causes a space to be printed in the output) and `"\n"` (which causes a carriage return).

### 5.2 Groups Generated by Subsets

In this section we want to compute the subgroup generated by a subset of elements.<sup>1</sup>

Specifically, we want to compute

$$\{a_{i_1} a_{i_2} \cdots a_{i_k} \mid k \in \mathbb{N}, a_{i_j} \in \{a, bc\}, j = 1, 2, \dots, k\},$$

where `a` and `bc` are as in the last example. This set is closely related to the subgroup generated by `a` and `bc`. In fact, since the group under consideration is finite this set is that subgroup.

First, a set to hold the generating set must be defined

---

<sup>1</sup>Actually, the method of this section only computes the smallest subset of  $G$  which contains the given subset and is closed under multiplication. However, since the group is finite any subset that is closed under multiplication will also be closed under inverses. This is true since in a finite group the inverse of an element is a power of that element.

```
Genset := [a, b*c];
```

Next all products of an element in `Genset` with an element of `Genset` is computed.

```
gap> SetMultiply(Genset,Genset,G);
```

In terms of the notation above, the output of the last command is

$$\{a_{i_1}a_{i_2} \mid a_{i_j} \in \{a, bc\}, j = 1, 2\}.$$

We want to take the union of this set with `Genset` in order to compute

$$\{a_{i_1} \cdots a_{i_k} \mid k \in \{1, 2\}, a_{i_j} \in \{a, bc\}, j = 1, \dots, k\},$$

This can be accomplished by

```
Union(last, Genset);
```

Notice the variable `last` refers to the output of the last command and the function `Union` returns the union of two sets. There are also variables `last2` and `last3` with the obvious meanings. Now we compute the number of elements in  $\{a_{i_1} \cdots a_{i_k} \mid k \in \{1, 2\}, a_{i_j} \in \{a, bc\}, j = 1, \dots, k\}$  by

```
Size(last);
```

We proceed by computing the following sets

$$\begin{aligned} \{a_{i_1} \cdots a_{i_k} \mid k \in \{2, 3\}, a_{i_j} \in \{a, bc\}, j = 1, \dots, k\} \\ \{a_{i_1} \cdots a_{i_k} \mid k \in \{1, 2, 3\}, a_{i_j} \in \{a, bc\}, j = 1, \dots, k\} \\ \{a_{i_1} \cdots a_{i_k} \mid k \in \{2, 3, 4\}, a_{i_j} \in \{a, bc\}, j = 1, \dots, k\} \\ \{a_{i_1} \cdots a_{i_k} \mid k \in \{1, 2, 3, 4\}, a_{i_j} \in \{a, bc\}, j = 1, \dots, k\} \end{aligned}$$

and in the process computing the sizes of those that include `Genset`.

Notice that we stop because the subgroup can have no more than 12 elements. We would also stop if sets including `Genset` stop increasing in size.

```
SetMultiply(last2,Genset,G);
Union(last, Genset);
Size(last);
SetMultiply(last2,Genset,G);
Union(last, Genset);
Size(last);
```

Enter the commands above. Using the up-arrow key to recall previously entered entries will greatly simplify the typing. You will thus compute the subgroup of  $G$  that is generated by the elements  $a$  and  $bc$ .

Before proceeding to the exercises section, use the method above to compute the subgroup of  $G$  that is generated by  $a$  and  $b$ .

## 6 Exercises

In order to do these exercises you will need to read into your GAP session, when you need them, the definitions of some groups. The definitions for the alternating group on 4 letters are contained in

```
./lab1/a4.gap
```

the definitions for the symmetric group on 4 letters are contained in

```
./lab1/s4.gap
```

and the definitions for the dihedral group of order 16 are contained in

```
./lab1/d16.gap
```

1. With  $G$ ,  $a$ , and  $b$  as they are defined in the alternating group on 4 letters, determine whether or not the following equation has a solution for  $x \in G$ :  $ax^3b = x$ . In your solution, print out the elements  $x$  of  $G$  and the simplified forms of  $ax^3b$ , one line for each element.
2. Repeat the previous problem for the symmetric group on 4 letters.
3. Repeat the previous problem for the dihedral group of order 16.
4. Determine subsets of the alternating subgroup on 4 letters which generate subgroups of sizes 2, 3, 4, and 12. Some of this has been done previously. Your work will be much simplified by the function `SubGroupGenerated`. For example, the following command finds the subgroup of the alternating group on 4 letters which is generated by  $a$  and  $b$ .

```
SubGroupGenerated([a,b],G);
```

5. **Extra Credit - 75,000 points** Find a subset of the alternating group on 4 letters which generates a subgroup with exactly 6 elements.<sup>2</sup>
6. Let  $G$  be the symmetric group on 4 letters. Find a set of two elements which generates this group. (The elements of the symmetric group on four letters are products of  $\mathbf{a}$ ,  $\mathbf{b}$ ,  $\mathbf{c}$ , and  $\mathbf{d}$ .)
7. Let  $G$  be the dihedral group of order 16. Find subsets (some will have one element, but none need have more than two) which generate subgroups of orders 2, 4, 8, and 16. (The elements of the dihedral group of order 16 are products of  $\mathbf{a}$  and  $\mathbf{b}$ .)

## 7 Finishing Up

Now you have completed the exercises and you want to hand in your work and keep a record for yourself.

At the end of the session you can issue the command

```
LogTo();
```

which stops the logging to the file. Now you have a file, called `session1.gap` if you used the naming convention above. Now leave GAP by entering

```
quit;
```

Next open a terminal session and edit the file you have created by

```
emacs session1.gap &
```

You can edit the file as desired, adding comments as you wish, and then print out the finished product and hand it in.

When you have finished for the day, remove the Linux bootdisk from the disk drive. Move the cursor to the K in the lower left corner and press the left mouse button. Holding the button down, select Logout from the resulting menu and then release the button. Eventually you will exit the X Windows environment, although there may be some intervening windows. After exiting, press the Ctrl-Alt-Del key combination to reboot.

## 8 What To Hand In

You are to hand in a completed, printed, edited copy of your work in this lab. This is due Friday, February 4.

---

<sup>2</sup>We will show that the number of elements of a subgroup divides the number of elements in the group it is contained in. However, it is not necessarily true that there is a subgroup corresponding to every divisor of the number of elements of the containing group.

## 9 Appendix

### Contents of lab1functions.gap

```
# Function to simplify the word y in terms of the elements of H
Normal := function(y, H)
  local x;
  for x in Elements(H) do
    if x = y then
      return x;
    fi;
  od;
  Print("The element is not in the group \n");
end;

# Function to convert a list/group/set to a set
Convert := function(A)
  local Aset;
  if IsSet(A) then
    Aset := A;
  else
    Aset := AsSet(A);
  fi;
  return Aset;
end;

# Function to multiply two lists/sets, A, B, and return a set consisting of
# all products x*y (simplified by Normal()), where x is in A and
# y is in B and the group is G
SetMultiply := function(A, B, G)
  local Aset, Bset, x, y, Pset;
  Aset := Convert(A);
  Bset := Convert(B);
  Pset := [];
  for x in Aset do
    for y in Bset do
      AddSet(Pset, Normal(x*y,G));
    od;
  od;
  return Pset;
end;

# Function to find the subgroup of the finite group G which is
# generated by the subset A of G.
SubGroupGenerated := function(A, G)
  local Sset, Tset, Subgpset;
  Sset := Convert(A);
  Subgpset := Sset;
  Tset := Union(SetMultiply(Subgpset, Sset, G), Sset);
  while Size(Tset) > Size(Subgpset) do
    Subgpset := Tset;
    Tset := Union(SetMultiply(Subgpset, Sset, G), Sset);
  od;
  return Subgpset;
end;
```

## Contents of a4.gap

```
# Definition of the alternating group on 4 letters, in terms of
# generators and relations.
F := FreeGroup("a","b","c");
G := F/[F.1^2,F.2^2,F.3^3, F.1^F.2/F.1, F.1^F.3/F.2,
        F.2^F.3/(F.1*F.2)];
a := G.1;
b := G.2;
c := G.3;
```

## Contents of d16.gap

```
# Definition of the dihedral group of order 16, in terms of
# generators and relations.
F := FreeGroup("a","b");
G := F/[F.1^8,F.2^2,F.2*F.1*F.2*F.1];
a := G.1;
b := G.2;
```

## Contents of s4.gap

```
# Definition of the symmetric group on 4 letters, in terms of
# generators and relations.
F := FreeGroup("a","b","c","d");
R := [F.1^2,F.2^2,F.3^3,F.4^2,F.1^F.2/F.1, F.1^F.3/F.2];
R := Union(R, [F.2^F.3/(F.1*F.2),F.1^F.4/F.1]);
R := Union(R, [F.2^F.4/(F.1*F.2),F.3^F.4*F.3]);
G := F/R;
a := G.1; b := G.2; c := G.3; d := G.4;
```