Learning how to use GAP to count some things - a worksheet

Max Neunhöffer

4 July 2012

This worksheet is intended to be a guide for you to learn about the first steps with GAP (see URL) to count some things.

`http://www.gap-system.org/`

Here is a sequence of exercises building up on each other. Therefore it is sensible to work through them in their given order. Sometimes there are hints. I suggest that you first try to solve each exercise without reading the hint and once you get stuck, try the hint. If you are still stuck, then please ask for help.

1. Start GAP.

   Hint: You find it under "Programs", "Maths" and "GAP4r4p12". Ignore the warning you get at startup.

2. Type in a few commands and compute $123456 \cdot 654321$.

3. Play around with some lists like in the presentation. You can find the log of the presentation here:

   `http://www.cs.st-andrews.ac.uk/~neunhoef/Teaching/sciencesummer12.html`

   and copy/paste should work using the mouse wheel for pasting.

4. Find out how GAP's help system works and learn about the break loop.

   Hint: Type

   ```
   ?break loop
   ```

   and hit enter. Type a question mark (`?`) followed by a word to find a manual section heading beginning with that word. Type two question marks (`??`) followed by a word to find all manual section headings containing that word. You can type

   ```
   1/0;
   ```

   to produce an error that takes you to a break loop. You can type

   ```
   quit;
   ```

   to leave the break loop again.

5. Compute the factorial of 1000, that is $1000! = 1 \cdot 2 \cdots 999 \cdot 1000$.

   Hint: You can use the library function. Later we will write a function to do this on our own.

6. Find out how to edit a text file on your computer, call it "`abc.g`" and read it into GAP using the `Read` command.

   Hint: First download the file "`isprime.g`" from the web page and save it under "`H:\`".

   `http://www.cs.st-andrews.ac.uk/~neunhoef/Teaching/sciencesummer12.html`

Then type

```
Read("H:\\isprime.g");
```

in GAP. You can then use the TextPad program to edit it and read it in as often as you want. For the time being it does not matter what this file does. Follow the same approach with a file called "`abc.g`".

7. It is now time to write your first function, try this one:

```
MyFirst := function(a,b)
  local c;
  c := a^2 + b^2;
  Print("It worked: ",a," and ",b," gives ",c,"\n");
  return c;
end;
```

8. Write a function computing factorials.

   Hints: Read about the `for` and `while` and `if` commands.

9. Put an `Error` statement into the above function.

   Hints: See "`?Error`", if you do

   ```
   Error("Hello");
   ```

   in a program, GAP enters a break loop during execution. You can inspect things and resume the computation using

   ```
   return;
   ```

   This is very useful for debugging.

10. A set in GAP is just a sorted list without duplicates. Find out how to compute the set of the elements occurring in a given list.

    Hint: See "`?Set`".

11. Let's now get back to what the file "`isprime.g`" actually does. It contains a function "`isprime`" which checks whether a given integer n is a prime number or not. Compute the list of primes up to 1 000 000. Is this very efficient?

    Hint: No, it isn't. We only have to do the trial division up to the square root of n, the program in "`isprime2.g`" on the same page is better. Download and try this new program.

12. We now want to use GAP to count some things. How many prime numbers are there between 1 and 10 000 000?

13. Write a function that takes a set L of elements and computes the set $L \times L$ of all pairs of elements of L.

    Hint: A pair is just represented as a list of length 2. Thus your result is a sorted list of pairwise different lists of length 2.

    Hint: First write a function building the set. Then lookup "`?Cartesian`" and compare the performance. Typing

    ```
    time;
    ```

    gives you the time in milliseconds used by the previous GAP command.

14. Count the number of pairs of numbers in the range $1, 2, \ldots, 147$ such that the first entry is divisible by 3, the second is not divisible by 7 and the sum of the two is a prime number.

    Hint: Use your program to find all pairs. Then use a function and "`Filtered`" to enumerate the pairs we are looking for. Count them with "`Length`".

Let n be a positive integer. A **partition** of n is a non-decreasing list of positive integers whose sum is n.

15. Download the file "`partitions1.g`" and run it for a few pairs (n, k) of positive integers.
16. Using the ideas in "`partitions1.g`", write a function that computes all partitions of n whose largest part is exactly equal to k.

    Hint: A similar recursive program does the job. Do the case distinction according to the second largest part.
17. Change "`partitions1.g`" such that it takes an additional argument j and that it only returns the partitions of n whose largest part is at most k and which have at most j parts.

    Hint: Return the empty list of partitions if j is negative and decrease the argument j with every recursive call.
18. How many partitions of 50 are there which do not contain the number 17?
19. How many ways are there to write 20 as a sum of up to 10 natural numbers?

    Hint: Here we distinguish between $20 = 1 + 19$ and $20 = 19 + 1$.
20. Assume you are given a set M of pairs (a, b) of positive integers and an integer N. What is the largest integer L, for which there is a subset S of M such that the sum of the first entries of the pairs in S is less than or equal to N and that the sum of the second entries of the pairs in S is L?

    Write a program to solve this problem. **Hint:** This is the famous knapsack problem and it is difficult.