gnu

Crowdsourcing project for the database of numbers of isomorphism types of finite groups

What is gnu(n) ?

For an integer n, the number of isomorphism types of finite groups of order n is denoted by gnu(n), where "gnu" stands for the "Group NUmber". The problem of the determination of all groups of a given order up to isomorphism is very interesting and challenging. For example, the sequence (https://oeis.org/A000001) in OEIS is the number of groups of order n, with the first unknown entry being gnu(2048). Known values of gnu(n) for 0 < n < 2048 are summarised in "Counting groups: gnus, moas and other exotica" by John H. Conway, Heiko Dietrich and Eamonn A. O'Brien (https://www.math.auckland.ac.nz/~obrien/research/gnu.pdf) which also discusses some properties of gnu(n) and related functions. Both OEIS and the latter paper derive most of the entries of the gnu(n) table from the GAP Small Groups Library (http://www.gap-system.org/Packages/sgl.html) by Hans Ulrich Besche, Bettina Eick and Eamonn O'Brien. The group numbers in the SmallGroups library are to a large extent cross-checked, being computed using different approaches and also compared with theoretical results, where available (see [Hans Ulrich Besche, Bettina Eick and Eamonn O'Brien. A MILLENNIUM PROJECT: CONSTRUCTING SMALL GROUPS. Int. J. Algebra Comput. 12, 623 (2002), http://dx.doi.org/10.1142/S0218196702001115], in particular 4.1. Reliability of the data).

What is known for n > 2048 ?

For n > 2048, the calculation of gnu(n) is highly irregular. Certain orders, including some infinite series (groups of order p^n for n<=6; groups of order q^n*p where q divides 2^8, 3^6, 5^5 or 7^4 and p is a an arbitrary prime not equal to q; groups of squarefree order; groups or order which is a product of at most three primes) are covered by the GAP Small Groups Library (http://www.gap-system.org/Packages/sgl.html) so the gnu(n) is returned by `NrSmallGroups(n)` . The recently submitted GAP package SglPPow (http://www.gap-system.org/Packages/sglppow.html) by Michael Vaughan-Lee and Bettina Eick adds access to groups of order p^7 for p > 11 and to groups of order 3^8 (it should be loaded with `LoadPackage("sglppow")` . For groups of cube-free order, the Cubefee package (http://www.gap-system.org/Packages/cubefree.html) by Heiko Dietrich calculates gnu(n) with `NumberCFGroups(n)` .

How to calculate gnu(n) for arbitrary n ?

For groups of other orders one could try the GAP package GrpConst by Hans Ulrich Besche and Bettina Eick (http://www.gap-system.org/Packages/grpconst.html) to construct all groups of a given order using `ConstructAllGroups(n)` . As documented at http://www.gap-system.org/Manuals/pkg/grpconst/htm/CHAP003.htm, this function usually returns a list of groups, in which case gnu(n) is the length of this list. However, sometimes this list contains sublists. In this case, one has to check each such sublist contains groups which are pairwise non-isomorphic, or remove duplicates.

The runtime and memory requirements of `ConstructAllGroups` depend very much on n and may vary from minimalistic to practically unfeasible. The website of AG Algebra und Diskrete Mathematik (TU Braunschweig) provides the table containing gnu(n) for many n < 50000: http://www.icm.tu-bs.de/ag_algebra/software/small/number.html. These numbers were taken from the Small Groups Library or calculated with the GrpConst package. There is no information in the table for 1082 orders for which the computation have not yet been completed.

Goals of this package

As we see, currently there is no uniform access to the calculation of gnu(n) in GAP even in the case when it is feasible, since one has to call different functions in a different way, dependently on n. Even finding all known gnu(n) for a list of integers with `NrSmallGroups` is not straightforward, since GAP enters a break loop when the library of groups of order n is not available. Also, these data are accessible only from within the working GAP installation. Next, users who calculate new values of gnu(n) have no easy ways to share their data with others and record provenance details, i.e. who calculated them and when, using which hardware and which versions of GAP and relevant packages, and how much memory and runtime were needed. These missing details hinder verification of the results, while it will be useful to have them easily available to cross-check calculations using different approaches, to check the correctness and performance of new implementations that may emerge in the future, and to check that future versions of GAP do not break these calculations. Also, they may be useful for researchers who wants to calculate all groups of a given order with `ConstructAllGroups` and are interested to know in advance how much time it may take and whether someone else had already attempted this calculation.

The Gnu package addresses these problems by:

- Providing uniform access to the calculation of gnu(n) using a single function.
- Offering both the ability to install package locally and to access it remotely without its local installation.
- Providing remote data via SCSCP (Symbolic Computation Software Composability Protocol) to make them accessible to any SCSCP-compliant software (see the list at http://www.symbolic-computing.org).
- Using GitHub-based development model and storing in the revision history the provenance details such as runtime requirements, details of the software and hardware, etc.

Local installation

To use the package locally, first you have to install the GAP system using the source distribution from http://www.gap-system.org/Releases/. Please ensure that you build packages as described there as well. After that, the Gnu package could be installed in the same way like other GAP packages that do not require compilation. It is suggested to install it as a custom package in the `.gap/pkg` subdirectory of your home directory instead of placing it into the `gap4rN/pkg` directory of your GAP installation. Since the package is regularly updated with new data, you may use git to clone it and subsequently pull changes from the main repository. To do this, change to the `.gap/pkg` directory and call

```
git clone https://github.com/alex-konovalov/gnu.git
```

This will create the directory `gnu` . Later when you will need to pull changes, change to that directory and call

```
git pull
```

Alternatively, if you do not use git, you may download a zip-archive from https://github.com/alex-konovalov/gnu/archive/master.zip and later update it manually by downloading new zip-archive and unpacking it to replace the previous installation of the Gnu package.

After loading the package with `LoadPackage("gnu");` you should be able to use it as follows:

```
gap> Gnu(10000);
4728
gap> GnuExplained(10000);
[ 4728, "precomputed using GrpConst package" ]
gap> NextUnknownGnu(10000);
10080
gap> GnuWishlist([2000..3000]);
[ 2048, 2240, 2496, 2560, 2592, 2688, 2880, 2916 ]
gap> List([105,128,2004,10000,2304,3^8,7^2*5^2*11*19,50000],Gnu);
[ 2, 2328, 10, 4728, 15756130, 1396077, 8, false ]
```

You may see some more examples of explanations how the values of gnu(n) were obtained in the following example:

```
gap> List([105,128,2004,10000,2304,3^8,7^2*5^2*11*19,50000],GnuExplained);
[ [ 2, "using NrSmallGroups and the GAP Small Groups Library" ],
  [ 2328, "using NrSmallGroups and the GAP Small Groups Library" ],
  [ 10, "using NrSmallGroups and the GAP Small Groups Library" ],
  [ 4728, "precomputed using GrpConst package" ],
  [ 15756130, "http://dx.doi.org/10.1016/j.jalgebra.2013.09.028" ],
  [ 1396077, "using NrSmallGroups from SglPPow 1.1" ],
  [ 8, "using NumberCFGroups from CubeFree 1.15" ],
  [ false, "not stored in gnu50000 and no library of groups of size 50000" ] ]
```

Remote connection

It is also possible to access the data without local installation by accessing the dedicated GAP SCSCP server that runs in a Docker container in the Microsoft Azure cloud. This server is periodically restarted to pick up database updates. To access it from GAP, first you need to load the SCSCP package:

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

Note that SCSCP package requires the IO package, and the IO package needs compilation on UNIX systems (for Windows, the GAP distributions comes with compiled binaries for the IO package).

After that, download and read (or copy and paste) the following file into GAP: https://raw.githubusercontent.com/alex-konovalov/gnu/master/lib/gnuclient.g

Now you are able to use remote counterparts of the commands shown in the previous section:

```
gap> GnuFromServer(50016);
1208
gap> GnuExplainedFromServer(50080);
[ 1434, "precomputed using GrpConst package" ]
gap> NextUnknownGnuFromServer(50080);
50112
gap> GnuWishlistFromServer([50000..50100]);
[ 50000, 50048 ]
```

If you have locally installed package, then the functions mentioned in this section will become available after its loading.

Note that the server is restarted periodically and may not contain the latest additions to the database. You may check when the server had been started and which version of the package it uses using the `GetServiceDescription` function from the SCSCP package:

```
gap> GetServiceDescription("scscp.gap-system.org",26133);
rec(
  description := "GAP SCSCP server for numbers of isomorphism types of finite \
groups. Gnu package version given by commit https://github.com/alex-konovalov/\
gnu/commit/6630e86ec7b1633b0afaeb7e35e8045561bb8e60. Server started on Sat Jun\
  4 20:46:10 UTC 2016", service_name := "gnu(n) SCSCP service",
  version := "GAP 4.8.3; CubeFree 1.15; Gnu 6630e86ec7b1633b0afaeb7e35e8045561\
bb8e60; GrpConst 2.5; SCSCP 2.1.4; SglPPow 1.1" )
```

To access the GAP SCSCP server from other SCSCP-compliant systems, follow their documentation for SCSCP client functionality and use the server name scscp.gap-system.org and port number 26133 similarly to the calls in https://github.com/alex-konovalov/gnu/blob/master/lib/gnuclient.g

Accessing provenance information

Using git, you can search in the version control history to find the details about the computation. For example, you can find the commit which adds gnu(4000) with the following command

```
git log --grep="gnu(4000)"
```

which will produce the following output:

```
commit dd9ae55743fe465389324bc44e54197bea146dc7
Author: Alexander Konovalov <alexk@mcs.st-andrews.ac.uk>
Date:   Sat May 21 15:33:01 2016 +0100

    gnu(4000)=6108

    GAP 4.8.3
```

```
GrpConst 2.5
Runtime: 975884 ms
Isomorphic groups eliminated

Submitted by @gnufinder. Validated by @alex-konovalov on
Ubuntu 16.04 on Azure cloud standard DS3 v2 instance (4 cores, 14 GB RAM)

Closes #59.
```

Contributing to the database

You can help to the development of this database with the following contributions:

- submitting new values of gnu(n)
- recording information about partial results to be pursued further (for example, when you run `ConstructAllGroups` but were unable to check non-isomorphism, or the computation was not completed after several days)
- verifying existing entries (possibly using other hardware, operating systems, new releases of GAP and related packages)
- improving the functionality of this package

You can submit new values of gnu(n) as new issues or pull requests to the GitHub repository https://github.com/alex-konovalov/gnu (you will have to create a GitHub account if you don't have one yet).

The template for the new issue/pull request will ask you to check that you provide the following details:

- Version of GAP and critical packages: GrpConst, Cubefree, etc.
- Brief description of the computer used for the calculation (operating system, processor, RAM)
- Runtime required for the calculation
- GAP commands used for the calculation
- Confirm that the output `r` of `ConstructAllGroups` is a *list* of groups (`ForAll(r,IsGroup)` should return `true`), or otherwise confirm that if the output contained lists of groups, then those groups were also shown to be pairwise non-isomorphic.

This information will be used to re-run your calculations and add gnu(n) to the database only after they will be verified.

Group orders that are currently not included in the database can be determined using `NextUnknownGnu` , `GnuWishlist` and their remote procedure call counterparts `NextUnknownGnuFromServer` and `GnuWishlistFromServer` as shown above. It may be also useful to look at currently open issues and pull requests since they may contain newly reported results awaiting to be added to the database after their validation. Yo do not need to worry that you may be duplicating someone's else computation, because in this case **DUPLICATION IS REPLICATION** and by checking that you can reproduce the same result with your GAP installation on your computer you will help to improve the quality of the software used in the experiment.

To submit partial results, please create new issues in this repository and tell what you have tried and at which step the calculation stopped. It will be useful to know, for example, about time-consuming cases that did not finish after substantial amount of time, or run out of memory, or where only the first step of the calculation had been completed, but checking the non-isomorphism has not been done.

You can also help with validating new submissions or rechecking existing ones, and with improving mathematical functionality of the package or its infrastructural part.

You may automatically generate almost all the text to submit using the function `GnuByConstructAllGroups` from the `grpconst.g` script located at https://raw.githubusercontent.com/alex-konovalov/gnu/master/lib/grpconst.g, and also included in the `lib` directory of the package. For example (note the double semicolon usage to suppress the output of the returned list):

```
gap> r:=GnuByConstructAllGroups(50024);;
****************************************
Constructing all groups of order 50024
****************************************
#I  computing groups of order [ 2, 2, 2, 13, 13, 37 ]:

#I  compute Frattini factors:
#I    compute ff groups with socle 2 and size 2
#I    compute ff groups with socle 4 and size 8
#I    compute ff groups with socle 8 and size 8
```

```
#I    compute ff groups with socle 13 and size 52
#I    compute ff groups with socle 26 and size 104
...
...
...
#I   extend candidate number 121 of 123 with size 50024
#I   extend candidate number 122 of 123 with size 50024
#I   extend candidate number 123 of 123 with size 50024
#I  found 187 extensions

****************************************
gnu(50024)=197

GAP      4.8.3
GrpConst 2.5
Runtime: 26335 ms
Isomorphic groups eliminated

In case this value is new, add the next line to data/gnudata.g
GNU_SAVE( 50024, 197, WITH_GC );

****************************************
gap>
```

In this case, instead of filling in the template you can copy and paste the last block of lines from the output into the description of an issue or a pull request with the added call to `GNU_SAVE`, and will only need to add the description of the computer used for the computation. In some cases, when `ConstructAllGroups` returns a list with sublists, the message will also contain report about further isomorphism checks. The function `GnuByConstructAllGroups` also returns a record with the output of `ConstructAllGroups` and timings in case it may require further analysis.

Finally, if you are submitting new values of gnu(n) as GitHub issues, please submit strictly one issue per group order. If you are submitting new values of gnu(n) in a pull request, you may submit one value (in which case the simplest way to submit a pull request is to edit the file https://github.com/alex-konovalov/gnu/blob/master/data/gnudata.g via the GitHub's web-interface) or multiple values, in which case each of them should be in an individual commit with the appropriate commit message looking like the summary produced by `GnuByConstructAllGroups` (see https://github.com/alex-konovalov/gnu/pull/58 for an example). Note however that it may take longer time to review such pull request.

Further details and formatting rules could be found in CONTRIBUTING.md: https://github.com/alex-konovalov/gnu/blob/master/CONTRIBUTING.md

Please take a look, and it will be great if you could be involved!

Alexander Konovalov

May 2016

Acknowledgements

We acknowledge financial support from the OpenDreamKit Horizon 2020 European Research Infrastructures project (#676541). We also acknowledge computational resources that were made available via the Microsoft Azure for Research award.