GENREG-manual

GENREG is a program written by Markus Meringer for generating connected simple regular graphs. This version also contains code added by Gunnar Brinkmann to restrict the generation to bipartite graphs, and by Brendan McKay to provide output in the graph6 format and also to have English messages.
To generate the connected k-regular graphs on n vertices call **genreg n k**. To obtain only those graphs with girth $\geq t$, you must append t as another parameter, therefore **genreg n k t**. To control the output you may use the following options **after the numerical parameters**:

-b only generate bipartite graphs

-a The adjacencylists of the generated graphs are written to an ASCII file with suffix `.asc`. There is also the girth, a set of generators of the automorphism group and its order specified. For example when calling **genreg 4 3 -a** you obtain the file `04_3_3.asc` with the content:

```
Graph 1:

1 : 2 3 4
2 : 1 3 4
3 : 1 2 4
4 : 1 2 3
Taillenweite: 3

3 : 1 2 4 3
2 : 1 3 2 4
2 : 1 4 3 2
1 : 2 1 3 4
1 : 3 2 1 4
1 : 4 2 3 1
Ordnung: 24
```

First the adjacency list is given. In every line there are behind the colon the neighbours of the vertex in front of the colon. Then we have the girth and afterwards in every line an automorphism π. Before the colon there is $\min\{j \in \underline{n} \mid \pi(j) \neq j\}$ and behind we have at i-th position $\pi(i)$. The given generators are representatives of the left cosets of a centralizer chain of the automorphism group (Sims chain).

If you declare an additional integer x, only the first x graphs are put on the file. In this case the file will possibly not contain all the graphs for given n, k and t. Therefore the filename is marked by an additional `-U` (for unfinished), e.g. with **genreg 6 3 -a 1** you obtain the file `06_3_3-U.asc`.

If the option **-a** is followed by **stdout**, the output is not written to a file, but to *stdout*.

-s The generated graphs are written to a binary file with the suffix `.scd`. The following coding (shortcode) is used:

One after the other all vertices $1, ..., n$ are considered. Only adjacent vertices with larger number than the considered vertex itself are added to the code. Thus we have for every

edge of the graph exactly one entry in the code. For the example above ($n = 4$, $k = 3$) you get

```
2 3 4 3 4 4
```

as code. To achieve a further compression of the data, we compare the code of the next graph to be constructed with the preceeding and find out, in how many entries at the begining the two codes are equal. Instead of writing the common pieces twice on the file, we only store its length and then the differing entries. Thus as first entry of the file we always have zero. The 4-regular graphs on 7 vertices have the codes

`2 3 4 5 3 4 5 6 7 6 7 6 7 7` and

`2 3 4 5 3 4 6 5 7 6 7 6 7 7`

The file `07_4_3.scd` consists of

```
0 2 3 4 5 3 4 5 6 7 6 7 6 7 7 6 6 5 7 6 7 6 7 7
```

and has length of 24 byte. This kind of compression gets better for big n or $t > 3$. The program *readscd.c* contains easy functions which are able to read shortcode files.

This option can be followed by an integer or **stdout** if only a certain number of graphs are to be stored or you want to use *stdout*.

-g The generated graphs are written to a file in graph6 format.

This option can be followed by an integer or **stdout** if only a certain number of graphs are to be stored or you want to use *stdout*.

-u The generated graphs are not written at all.

-e The parameters n, k and t, the number of generated graphs and the required CPU-time are written to a file with suffix `.erg`. Call of **genreg 21 4 5 -e** produces a file named `21_4_5.erg` containing

```
GENREG - Generator fuer regulaere Graphen
21 Knoten, Grad 4, Taillenweite mind. 5
Erzeugung gestartet...
8 Graphen erzeugt.
Laufzeit:20.9s  0.4 Repr./s
```

As long as the construction is not finished, this file has the name `21_4_5-U.erg` (and of course the last two lines are missing).

If option **-e** is not used, GENREG writes this information to *stderr*.

-c During the execution of the program the generated graphs are counted and the number is written to *stderr* with short intervals. If option **-e** is used, GENREG writes this information to the `.erg` file.

In cases where the computation will be very long, the following modulo-option is available to split the problem into several jobs, so it can be run in *parallel* on different machines.

-m This option must be followed by two integers i and j, $1 \leq i \leq j$. It is used to split the problem into j parts. Call of **genreg 20 3 -s -m 1 2** causes the program to compute only

the first of two parts. The code of the graphs is written to a file named `20_3_3-1.scd`. When the generation is finished, it is renamed to `20_3_3#1.scd`. The remaining graphs can be computed with **genreg 20 3 -s -m 2 2**.

Remarks:

- The program is designed to run on UNIX machines. To preserve compatibility to other operating systems (OS/2, DOS), filenames are chosen to have length at most eight. **genreg 20 3 -s -m 1 2** and **genreg 20 3 -s -m 1 3** produce files with the same name and differing content.
- As long as the number of jobs j is not too large, the single parts should take about equal time and produce about equally much output. For very big problems, especially with girth > 3 deviation from this fact may occur. In such cases contact the author (markus@btm2x2mat.uni-bayreuth.de) to decide whether a little tuning will solve the problem or it is really hopelessly out of range.

Installing GENREG

These instructions are for UNIX only. After unpacking the tar file, edit `makefile` to select a C compiler (`cc` or `gcc`). If English messages are required, add `-DENGLISH` to the definition of `CC`. (Otherwise, messages will be in German.) After editing `makefile`, you can compile the program using `make` .