# CSE835 nauty Instructions

## Compiling nauty

nauty is a program that computes non-isomorphic graphs. It may be downloaded from http://cs.anu.edu.au/~bdm/nauty/. This document refers to version 2.4. To install nauty on Linux, simply unzip the directory somewhere and type:

```
cd nauty24
./configure
make
```

nauty should compile without issue on any of the cse machines. It will compile with some errors using Xcode on OS X (the geng executable compiles anyhow). Compiling on Windows requires the use of a Linux like environment such as Cygwin that has autoconf and make in order to run configure and the makefile. If you encounter a compilation error on any executable besides geng, you can try to fix it, or remove it from the gtools target in the makefile and run make again.

## Installing Cygwin

Cygwin is not an emulator, but a collection of tools and a dll that provide Linux functionality in Windows. It may be downloaded from http://www.cygwin.com/. Download and run setup.exe, keeping the default settings except for the 'Select Packages' page. Here select 'gcc' and 'make' from the 'Devel' category before proceeding. Alternately, perform a complete install, but be warned that it is several GB. Once installation is complete, open a Cygwin Bash shell, copy the folder containing nauty to somewhere such as C:\cygwin\home\Administrator, then run configure and make as before. Cygwin defaults to Unix line endings (newlines) instead of DOS line endings (carriage returns), so make sure to download the appropriate version of nauty, or the configure script may not execute properly. Note that outside executables may be run inside a Cygwin bash shall if they are in your path environment variable, and that executables compiled in the Cygwin environment may be run in a regular Windows command prompt if a copy of cygwin1.dll is put in the C:\WINDOWS directory or another directory included in the path environment variable (the latter trick does not appear to work on Windows 7).

## Using nauty

Compiling nauty creates a number of executables. The one of primary interest to us is geng. Typing './geng 3' generates all non-isomorphic graphs on 3 vertices, './geng -help' displays further options. The executable genbg generates digraphs. Extraneous information is shown in lines beginning with a greater than sign. These lines are printed to stderr rather than stdout, making nauty trivial to use with piping and redirection. If you are unfamiliar with I/O redirection, see http://en.wikipedia.org/wiki/Redirection_(computing). The graphs themselves are displayed as not so human readable bit arrays:

```
>A ./geng -d0D2 n=3 e=0-3
B?
BO
BW
Bw
>Z 4 graphs generated in 0.00 sec
```

Other executables do different things, although you will have no use for most of them. Use '-help' or examine the source code for more details.

## Customizing nauty

Rather than manually editing program files and the makefile, please download the folder custom.zip containing makefile,in, shared.c, et al from the CSE835 website, and place the contents into your nauty folder. Overwrite the existing makefile.in, then run configure and make. If you have already compiled nauty, simply copy these files to your nauty folder, overwriting the existing makefile.in, then rerun configure and make.

Our code creates a number of additional executables, the most useful of which are gengpg and gengadj. These programs are identical in function to geng, except that they output graphs in pg and adjacency matrix formats, respectively. Assuming you had no compilation issues, typing './gengpg 3' will output the following:

```
>A ./gengpg -d0D2 n=3 e=0-3
{0: {}, 1: {}, 2: {}}
{0: {2: 1}, 1: {}, 2: {0: 1}}
{0: {2: 1}, 1: {2: 1}, 2: {0: 1, 1: 1}}
{0: {1: 1, 2: 1}, 1: {0: 1, 2: 1}, 2: {0: 1, 1: 1}}
>Z 4 graphs generated in 0.00 sec
```

Typing './gengadj 3' will output the same graphs as adjacency matrixes, where diagonal entries indicate vertex degree, and non-diagonal entries represent the presence or absence of an edge with -1 and 0, respectively.

## Using nauty with pg

There are two ways to make use of the graphs nauty generates in pg. The first method is to output them to a file, using redirection like './gengpg 3 > graphs.pg'. Since only the graphs are written to stdout and thus graphs.pg, no manual editing of the file is needed. Now create a python program like the following:

```
import pg
f = open('graphs.pg', 'r')
for b in f.xreadlines():
      g = pg.Graph(eval(b[0 : len(b) - 1]))
      ...
f.close()
```

However, this approach requires storing all the graphs on a given number of vertices to a file. There are 274 thousand non-isomorphic graphs on 9 vertices, 12 million non-isomorphic graphs on 10 vertices, 1 billion non-isomorphic graphs on 11 vertices, and so on. A better method is to use piping, like 'gengpg.exe 3 | python test.py'. Simply create a python program like the following:

```
import pg
import sys
while 1:
      b = sys.stdin.readline()
      if not b:
            break
      g = pg.Graph(eval(b[0 : len(b) - 1]))
      ...
```

pg can be used to convert graphs found with nauty to images. See draw_matplotlib() (requires Python 2.6 and matplotlib), and draw_tkz_graph() in pg. Alternately, if you save a single pg graph to a file, you can open it using GMPX.

## nautytopg, pgtonauty, twin, and dh

Our custom code creates a number of executables besides gengpg and gengadj:

1) nautytopg takes as input graphs in graph6 format and outputs them in pg format. This is useful if you want to run the output of geng through another nauty executable before converting it to pg format.

2) pgtonauty takes as input graphs in pg format and outputs them in graph6 format. This is useful if you want to convert the output of pg back to nauty format for some reason.

3) twin takes as input graphs in graph6 format and outputs those that are P-twins, where a graph G is a P-twin for some property P if P holds for the graph and its complement. Typing './geng 8 | ./twin -t' will list all spanning tree twins with 8 vertices. Type './twin -help' for an exhaustive list of properties and output formats.

4) dh takes as input graphs in graph6 format and outputs those that are distance hereditary. Like twin, dh can output graphs in a variety of output formats.