CSCI-761 Homework 2

Tristan Miller

February 2026

Part 1

To generate triangle-free graphs on 4 vertices I used `geng -lt 4 > n4.g6`. The contents of `n4.g6` are

```
C?
C@
CB
CF
C‘
CR
Cr
```

Here is a command to generate the file `n$((n+1)).g6` from the file `n$n.g6`:

```
<n$n.g6 addptg -lj1: \
    | pickg -k:2 -h:4 \
    | sort -u >n$((n+1)).g6
```

The first line reads the file `n$n.g6` and uses `addptg` to add a new point to each graph in every possible way. `-j1:` is used to allow the new point to have any number of edges greater than or equal to 1.

The second line selects all the new graphs that have no triangles (`-k:2`, i.e. no cliques larger than 2) and no independent sets of size 5 (`-h:4`).

The final line removes duplicate graphs using `sort -u` and writes the result to the new file.

We can repeat this procedure up to `n14.g6`, which is empty as there are no graphs on 14 vertices that are free of both triangles and independent sets of size 5.

The contents of `n12.g6` (12 groups):

```
K‘aAAGUEpRDo
K@AAHWYoYwTO
K‘aAIOiDWsCh
K‘?CGtDIkwL_
K?CkQMp[cgL@
K?GTa\cUDGrC
```

```
KoCIHaO@XDHB
K‘‘@OkcEICoL
KQ‘?pMCQ?bcU
Ks_GagjLASko
Ks_HIGZKQSm_
K?_YPMQoPokc
```

And of `n13.g6` (only 1 group):

```
Ls‘?XGRQR@B‘Kc
```

I wrote and used the following script to generate these files:

```
#!/bin/sh

# Generate the canoncal labelings for all triangle-
    free graphs on 4 vertices
geng -lt 4 > n4.g6

# Generate the next file of graphs
next() {
    n=$1  # number of vertices
    cl=$2 # avoid cliques of this size
    is=$3 # avoid independent sets of this size

    infile=n$n.g6
    outfile=n$((n+1)).g6

    echo "generating␣$infile␣->␣$outfile"

    # as explained above
    <$infile addptg -lj1:
        | pickg -k:$((cl-1)) -h:$((is-1))
        | sort -u >$outfile
}

# Repeat the above procedure to generate all the
    required files
for i in $(seq 4 13); do
    next $i 3 5
done
```

Part 2

N13

We can use `dreadnaut` to find information about the automorphism group of the sole graph in `n13.g6`, `Ls‘?XGRQR@B‘Kc`:

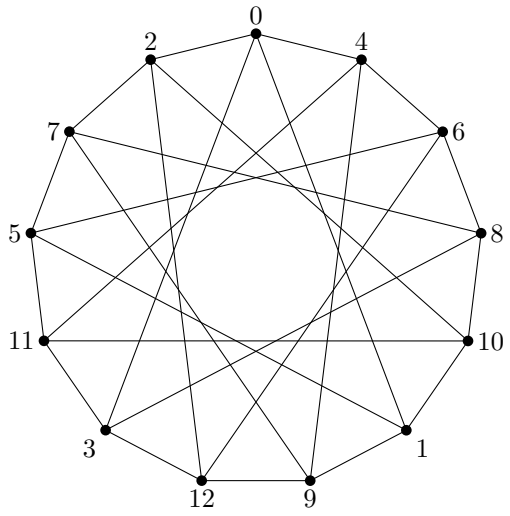
```
$ listg -d n13.g6 > n13d
$ dreadnaut
Dreadnaut version 2.9.3 (64 bits).
> <n13d
> x
(1 2 3 4)(5 7 8 6)(9 10 12 11)
level 2:  4 orbits; 1 fixed; index 4
(0 1)(2 9)(3 5)(4 10)(6 8)(7 12)
level 1:  1 orbit; 0 fixed; index 13
1 orbit; grpsize=52; 2 gens; 6 nodes; maxlev=3
cpu time = 0.00 seconds
```

We can see that the graph has automorphism group size 52 and two generators:

$$(1\ 2\ 3\ 4)(5\ 6\ 7\ 8)(9\ 10\ 11\ 12), \text{ and}$$

$$(0\ 1)(2\ 9)(3\ 5)(4\ 10)(6\ 8)(7\ 12).$$

The graph may be drawn like so:



From the drawing, we can immediately see the graph has a symmetry of order 13 (rotation) and a symmetry of order 2 (reflection). A less obvious symmetry is "turning the graph inside-out", where the inner star pattern becomes the outer polygon and vice versa.

N12

The two most symmetric graphs in `n12.g6` are `Ks_GagjLASko` and `K‘aAIOiDWsCh`.

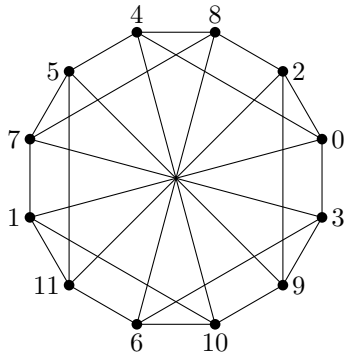
`Ks_GagjLASko` has group size 48 and is generated by four generators:

$$(1\ 2)(3\ 4)(5\ 6)(7\ 8)(9\ 10)$$

(1 2)(7 9)(8 10)

(0 1)(2 11)(3 7)(4 10)(5 9)(6 8)

(0 3)(1 7)(2 9)(4 6)(5 11)(8 10)



This drawing makes the 60° rotational symmetry and reflections obvious. We can also exchange pairs of vertices with an edge-crossing next to them (eg. 6 and 10). The first two generators, however, are not obviously symmetries of the graph.

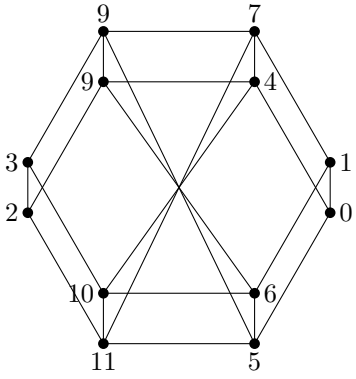
`K‘aAIOiDWsCh` has group size 16 and is also generated by four generators:

(2 3)(4 5)(6 7)(8 9)(10 11)

(2 3)(8 10)(9 11)

(0 1)(4 6)(5 7)

(0 2)(1 3)(4 8)(5 11)(6 10)(7 9)



This drawing shows two reflectional symmetries (left-right and top-bottom) and two 180° rotational symmetries. There is also a symmetry given by rotating just the left or right half of the graph 180°.