CDMTCS
Research
Report
Series

An Investigation of Algorithms to Aesthetically Draw Cayley Graphs

Alastair A. Abbott

Michael J. Dinneen

Department of Computer Science,
University of Auckland,
Auckland, New Zealand

CDMTCS-318
March 2008

Centre for Discrete Mathematics and
Theoretical Computer Science

An Investigation of Algorithms to Aesthetically Draw Cayley Graphs

Alastair A. Abbott and Michael J. Dinneen
(aabb009@ec.auckland.ac.nz & mjd@cs.auckland.ac.nz)

Dept. of Computer Science, University of Auckland
Private Bag 92019, Auckland, New Zealand

March 14, 2008

Abstract

Graph visualisation is an important field in Computer Science. The visualisation of groups in the form of Cayley graphs has applications in the layout of interconnected networks and mathematics. By using theoretical results from group theory, we present two algorithms that take as input a Cayley graph (G, S) and draws it in a layout that highlights the symmetry of the group and is easily readable.

1 Introduction

Graph visualisation is an important and quickly growing field in Computer Science. Graphs can be used to represent various types of data, from family trees and simple flow charts to subway systems or mapping the universe [1]. The main issue is taking the raw mathematical information that describes the graph, and transforming it into a format that is easy to understand and read. A set of aesthetic criteria have been created that sets out some key factors that a graph layout needs to take into account to create a drawing that is desirable. The main factors are minimising the number of edge crossovers, keeping the total edge length to a minimum, maximising symmetry, maximising the compactness of the graph and maximising the angles between edges. The importance of these aesthetic properties has been supported by psychological research [1, 2, 3]. These criteria are not inflexible, as different situations require different graphs. For example, in an electronic wiring diagram, minimising edge length may be especially important, as wire costs money, and other criteria may need to be compromised [1].

Strictly minimising aspects such as the number of crossovers and the compactness of a graph are not simple, and have been shown NP-complete [2]. As a result, force-directed algorithms, otherwise known as spring algorithms have become increasingly popular. A general spring algorithm works by placing springs between vertices, and iterating towards a minimum energy situation. There are many spring algorithms, such as Kamada and Kawai's algorithm, or Tutte's algorithm [3]. Spring algorithms are particularly good at showing symmetry in a graph. In particular, it has been shown that for each geometric automorphism in a graph representing a symmetry, there is a stable state of the spring system that displays that symmetry [3].

We are looking in particular at the visualisation of a group, in the form of a Cayley graph.

A group is a nonempty set and a binary operation $(G, \cdot)$ in which $\cdot$ is associative, G contains an identity element and each element in G has an inverse element in G [4]. Any group can be represented by a Cayley graph in several ways. A Cayley graph $Cay(S : G)$ consists of a group G with generating set S. Each element of G is a vertex in $Cay(S : G)$, and for x and y in G, there is a (directed) edge (x, y) if and only if $x \cdot s = y$ for some s in G. Often colours are used to represent different generators of the Cayley graph [4]. If S is closed under inverses ($S = S \cup S^{-1}$) then we can view $Cay(S : G)$ as an undirected graph since we would have connections both ways ($x \cdot s = y \implies y \cdot s^{-1} = x \cdot s \cdot s^{-1} = x$).

While there has been research done on visualisation of many different types of data, there has been very little work done towards aiding visualisation of a group or even Cayley graphs in general, short of implementing a spring algorithm.

In the research of interconnected networks, Cayley graphs are often used as good models of the networks, due to their desirable properties such as vertex transitivity and high fault tolerance, so figuring out a better way to draw them could be of great use in that field [5]. In this situation, showing a high degree of symmetry is important, as it allows the same routing scheme to be implemented easily across the network and at each node. Drawing better diagrams would allow engineers to assemble networks more easily saving time and money, and having a drawing that shows symmetry well it should be easier to add wire connections between nodes. It is important to note that since every group is isomorphic to a permutation group, every group is a subgroup of a symmetric group, so being able to create good drawings would also aid pure mathematicians with understanding the symmetry and structure of group [4].

We plan to utilise the structure of groups and their generators to create an algorithm that will significantly aid in the drawing of Cayley graphs. Because of the importance of the symmetry in Cayley graphs, especially in many applications of the field, we will place extra emphasis on the symmetric properties of the graph, although other aesthetic criteria such as minimising edge crossovers will still need to be taken into account to avoid creating graphs that are too difficult to read.

2 Method Used for Developing a Good Drawing Algorithm

The algorithm that we were attempting to create needed to be able to take in an arbitrary group G, and a set $S \subseteq G$ of generators. The generators need not generate the whole of G, and the algorithm must be able to deal with any number of generators. We chose to colour code our generators (at least those with 8 or less generators) as is often the case in Cayley graphs as it allows easier understanding of the graph, and the ability to deduce which edges relate to which generators [2, 4]. Our algorithm can deal with a set of generators, S, that is not closed under inverses, meaning that the graph is strictly directed.

For any generator s the nodes generated by s starting at any node, say g, would result in a cycle gs, gs^2, $\dots gs^n = g$ because if G is a finite group then $s^n = e$, the identity [4]. This set of nodes generated by s will either span all of G, or will form a subset of G, trivially. From group theory we can recognise that this set is a coset of H in G containing s, where H is the set of elements in the word generated by s [4]. This realization is important, and leads us to a number of findings that we can use to aid our drawing. For a coset Hs and a group G, $|H|$ divides $|G|$, meaning that a generator s will either generate all of G, or a number of cosets all of equal order [4]. Furthermore, $|s| = |H|$ since H is simply the set of elements generated in the walk consisting of the powers of s, hence a generator generate $|G|/|s|$ cosets, all of order $|s|$.

It is desirable to tackle the set of generators in some logical order. From our previous findings, it seems wise that we would initially wish to draw the cycles of the largest length. These largest cycles are direct representations of the cosets of s, so it follows that the generator with the largest order will create the largest cycles. We use this finding to decide to rank S by element order, descending. If two elements have equal order, then there is no need to differentiate between them, so we can leave their order untouched after the sorting algorithm is run.

Generators beyond the first generator will lead us to one of two cases. A generator $s = s_i$ with $i \geq 2$, will either form a coset that is a subset of exactly one coset formed by s_1, or will create a coset that intersects multiple cosets of s_1. In the first case, the geometric result is a sub-cycle of a cycle generated by s_1. That is, it is a cycle of less length, but with each node in it also being in one larger cycle. In this case, the generator should have little weighting, if any, towards the final shape of the drawing, as the cycle shape should be dictated by the larger cycle. In the second case, the coset will join cycles that were created by s_1 and can be seen as 'spokes' between cycles. These are important in the final graph layout, and we should attempt to apply the aesthetics criteria to these spokes to create a desirable layout. We would also like to attempt to exploit the inherent symmetry of a group to easily show the symmetric characteristics in the final drawing, as is especially desirable in this situation due to the importance of symmetry in groups [4].

Next we turned our attention to creating a base layout for the graph. The two main possibilities we decided to explore based on examination of existing Cayley graphs [4, 10] were a **'radial'** layout and a **'nested'** layout, and we developed each of these methods into separate algorithms.

The radial layout involved placing the initial cycles generated by s_1 as nodes in a larger cycle centered on the center of the drawing. The remaining edges generated by the remaining generators where just added without changing the node position as calculated by the first generator. This seemed logical and nice in some situations, but looked messy even on some basic Cayley graphs based on Abelian (commutative) groups such as $Z_4 \times Z_3 \times Z_3$ (see Figure 1), mainly because it resulted in an excessive number of crossovers, which made for a messy and difficult to read drawing.

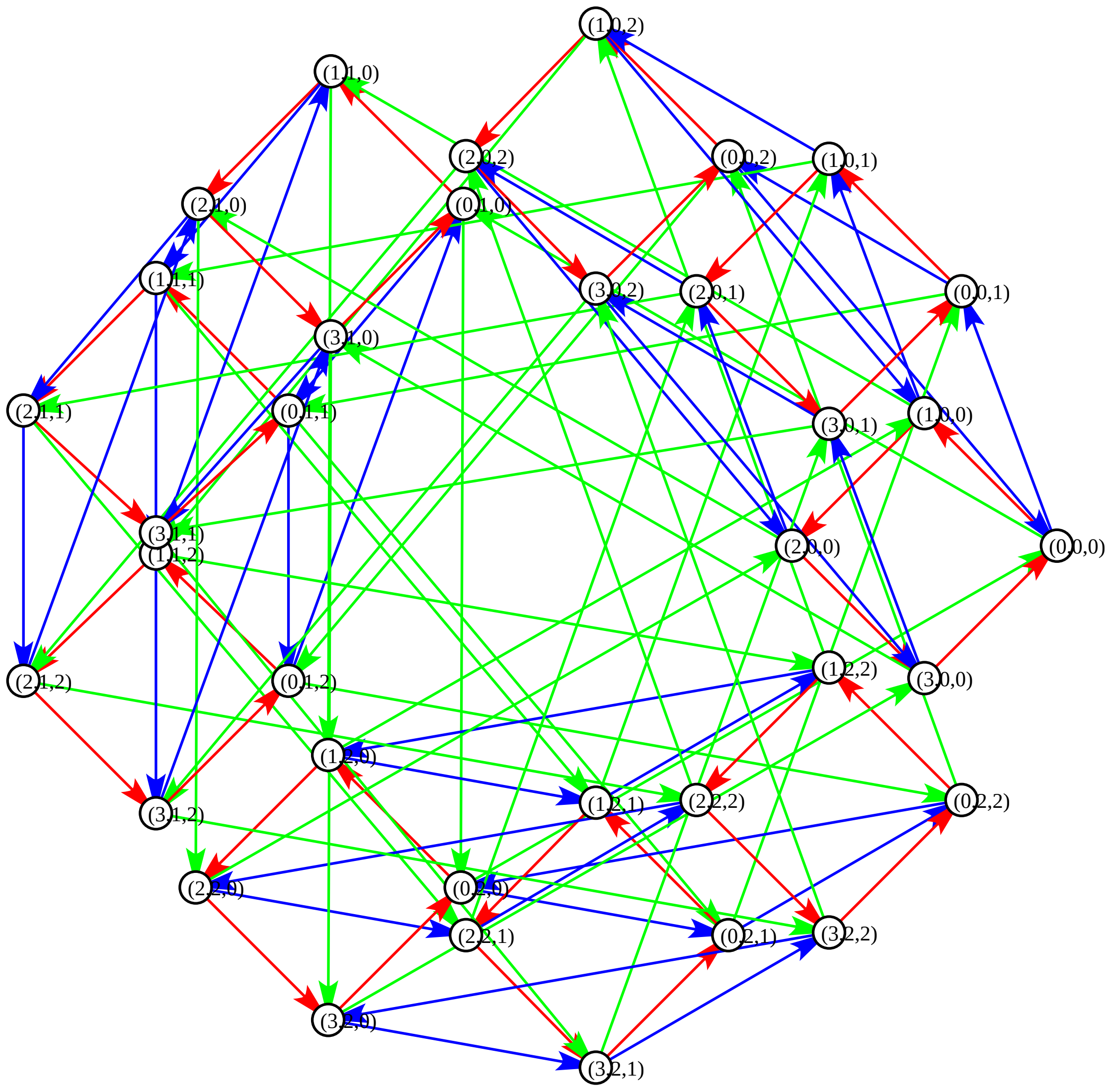


Figure 1: A cluttered radial drawing of an Abelian Cayley graph.

The nested algorithm was similar, but involved placing the initial cycles inside of each other, and adding the remaining edges as in the radial method. This showed much more success, with a low number of crossovers and a logical order that made it easy to visualise the group, especially in examples such as dihedral groups, where one cycle directly represented rotations of itself, and the spokes between the cycles represented a possible reflection at each position [4]. It also successfully showed the symmetry of the group in a highly desirable and obvious way, especially the rotational symmetry [3]. The main problem with this method was that it led to multiple edges being on top of each other, which was highly misleading in some situations. We countered this by rotating each alternate cycle by a few degrees, although in cases where the overlapping edges corresponded to a generator of which the inverse generator was also in S this would have been unnecessary. This lead to slightly more crossovers but was a huge improvement in aesthetics of the layout and the symmetry of the underlying group was still highly visible.

In the creation of our programs, we decided to implement a generalised group class to aid the visualisation of groups. Any group can be trivially represented by an adjacency list of its Cayley graph G, with the identity element 0. The generating set is $\{x \mid (0, x) \in E(G)\}$, and each multiplication required in the creation of the Cayley graph can be trivially computed. This allowed us to deal with the visualisation of Cayley graphs without the underlying group structure to be known, as is the case in the more general example. Using this as our trivial case, the program structure allows for the creation of more complicated classes to represent a class of groups, provided an invertible function that maps the elements of the group onto the integers 0 to $|G| - 1$ exists. For examples, a class of symmetric groups can be created, using the mapping described in [6] to map the permutations to the integers. This structure allows a user to create a class to represent a group of any objects, and visualise it using our algorithms.

Having implemented both the nested and radial methods, we examined the results that we had obtained by drawing the 10 example graphs presented in Appendix A. By examination, with the aesthetic criteria in mind, we found the nested algorithm to be producing significantly better results than the radial algorithm.

Because Cayley graphs are not necessarily planar [7] crossovers are inevitable, and as the size of the generating set S grows, the number of crossovers is bound to rise. While the nested method seemed to be superior to the radial method in terms of most aesthetic criteria, it still had the possibility for the number of crossovers and the final layout to be improved.

We decided to introduce spring algorithm concepts into our algorithms using a base layout as a structured and promising starting position for the nodes. A spring algorithm seemed like a good idea because they are good at showing symmetry, as is our base layouts, and should help to reduce the number of crossovers. We decided to implement the Kamada and Kawai's algorithm [8]. Rather than just running a standard spring algorithm using the nested/radial starting positions for vertices, which would dramatically reduce performance, we decided to implement it in stages. After each generator s_i, we ran a spring algorithm with decreasing sensitivity to moving vertices after each successive generator. We achieved this by decreasing the number of iterations, and also tried decreasing the spring force.

The spring algorithm methods gave mixed results. As we wished, the number of crossovers were reduced, and symmetry was maintained. However, the structure of the group that the nested layout by itself effectively presented was somewhat lost, and while symmetry was still visible it did not represent the symmetry of the group itself, as the nested method did. Because of this, it turned out that the nested algorithm, without any integrated springs, provided the picture that was the best aesthetically, having taken into account the importance of the group structure and symmetry.

3 Results

The pseudocode for the **nested algorithm** that we developed is given as follows:

```
1. Rank the generating set $S$ by the order of the generators descending

2. Calculate number $c$ of cosets as $|G|/s_1$.

$\boxed{\text{Draw the edges generated by the } 1^{st} \text{ generator and position nodes.}}$
3.    while there is an unseen node $g$ do
              until the current node, $g$, reaches back at the start of this coset:
              a) Position $g$ on a circle around the center which has a radius
              proportional to the number of the current coset, at $i/c$ of the way
              around the circle, where $i$ is the node number within the coset.
              b) Add edge $(g, g \cdot s_1)$
              c) Let $g = g \cdot s_1$

$\boxed{\text{Add the edges generated by the remaining generators.}}$
4. for each generator $s_i$ in $S$ do
      for each node $g$ in $G$ do
          Add edge $(g, g \cdot s_i)$
```

To test our algorithm we chose 10 example Cayley graphs, including some Abelian and non-Abelian groups, as well as some very dense and large graphs. These are detailed in Table 1, and the full adjacency lists and generators are in Appendix A.The notation we used for the groups is standard, with the quaternion group $Q = Q_4$. The groups are described fully in [11]. The semi-direct products (of cyclic groups) used are explained in more detail in [9].

Table 1: Basic attributes of the 10 key example graphs studied.

Group Name	**Order**	**Is Abelian?**	**Number of Generators**
D_{28}	56	No	4
$\mathrm{Z}_4 \times_\sigma \mathrm{Z}_{29}$	116	No	5
$Z_2 \times_\sigma Z_{12}$	48	No	3
$Z_2 \times Z_2 \times Z_3 \times Z_4$	48	Yes	4
$Z_9 \times Z_3$	27	Yes	7
$Q_8 \times Z_2$	32	No	4
S_4	24	No	4
$Q \times Z_3$	24	No	4
Q_{10}	20	No	5
A_4	12	No	4

We present our drawings of these test cases in Appendix B. Figures (A_4 nested) and (A_4 radial) show the Cayley graph of the group A_4. The drawing (A_4 nested) shows the graph produced by our algorithm using the nested algorithm of drawing the graph. The cosets and relations are clearly visible, and it is easy to use the symmetry of the graph to get an idea on the structure of the group. The drawing (A_4 radial) on the other hand shows the same Cayley graph drawn using the radial algorithm. This method produces a graph that lacks the symmetry and niceness of the nested graph, and this result was seen on virtually all the graphs using this method, with the exception of those graphs where the highest generator order was 2. In this case, the result was superior to the nested algorithm, due to the lack of cycles of length 3 or more, which resulted in unnecessary crossovers. This can be seen in the difference between the drawings ($Z_2 \times Z_{12}$ nested) and ($Z_2 \times Z_{12}$ radial). The drawing (A_4 nested spring) shows the same Cayley graph of A_4 drawn using the nested method but with the spring algorithm applied as discussed in Section 2, with decreasing sensitivity to moving vertices. This was the nicest of the spring drawings I was able to produce, and it shows the symmetry and main features well.

Consider the nested method for drawing another non-Abelian group, $Q_8 \times Z_2$. This large, order 32 graph shows is symmetry well and has a low number of undesirable edge crossings given its size. The benefit of adding twists to alternate concentric cycles can be seen along the diagonals. Without this twist all of the close blue lines would misleadingly overlap. The spring-based nested drawing of this same Cayley graph shows less structure. Unlike the drawing for A_4, it produced an undesirable drawing that lost much of the symmetry that we desire in Cayley graphs, and is difficult to read.

Now consider the drawing ($Z_9 \times Z_3$ nested). This is a very dense graph drawn with these generators, and unlike the previous two, it is based on an Abelian group. While it is cluttered due to the density, the symmetry is still startlingly obvious, and by maintaining this structure it is still possible to examine the group easily by closely examining the graph. Because of the density of the graph, it will contain a lot of crossovers when drawn with any algorithm, but my algorithm draws it very nicely and uses the group structure to produce a much better picture than would be possible with other algorithms.

A comparison of all the 10 main examples we used are available in Appendix B, including those drawn with some of the inferior methods along the way. All these graphs are structured well and are easy to interpret. As we initially stated as being desirable, they show the inherent symmetry of the groups, and produce equally good drawings for Abelian and non-Abelian groups.

4 Discussion

The nested algorithm that we found successful works well for a number of reasons. Because we are dealing with finite groups, every element g of G has finite order [4]. We can prove that the set of elements generated by an element s forms a coset:

Since $|H| = |s|$ and $|H|$ divides $|G|$, we have $|G|/|s|$ cosets of size $|s|$. By ordering the group S by element order, we ensure that $|s|$ is a maximum. This means we have a minimum number of cosets in our graph, where each coset is viewed as a cycle of length $|s|$. By ordering S, we ensure that the maximum number of edges are in these cycles, and the minimum number of edges go between these cycles. The edges drawn by the remaining generators go between these cycles and are the edges that have potential to cross other edges. By minimising this number we improve the final drawing of the graph.

The nested algorithm generally produces nicer layouts than the radial one because it effectively shows the symmetry of the graphs, which helps make a graph much easier to interpret [3, 1]. The radial method results in regions of the graph being very dense and others rather sparse due to the asymmetric placing of nodes within cosets in relation to other cosets. This creates lopsided graphs that are not easy to make sense of. Because the nested method arranges the cosets concentrically around each other, it displays rotationally symmetry of the graph, and for Abelian groups it also reflective symmetry as defined in [3]. This makes a graph easy to interpret and shows the structure of the group well, with the bonus that the symmetry displayed directly relates to the symmetry of the group.

Our algorithm is much more efficient than a spring-type algorithm. This is because spring algorithms use a large number of iterations to converge on a stable and good-looking drawing. By exploiting the symmetry of the groups, our algorithm does not do this at all. We were able to draw graphs with more than 100 vertices in a fraction of a second, where a spring algorithm took more than 30 minutes to draw the same graph. This is a significant advantage for the applications of this field as networks and groups that need to be visualised are often very large.

Our program has the potential to be easily adapted in many situations due to the flexibility of its accommodation of custom group classes. By creating a class that describes one type of group, it can be instantly used to visualise similar groups of any size or selection of generators, as well as the possibility for customisation of other features of the group. This could speed up the visualisation of a large number of Cayley graphs simultaneously.

The way our algorithms[1] work should be an advantage in the field of interconnected networks. Because the symmetry is strongly shown, it would be particularly easy to set up routing schemes at each node. By creating an algorithm that draws a Cayley graph in such a way that is aesthetic with a strong emphasis on the display of symmetry we have achieved this goal.

5 Future Work

While the algorithm we have described efficiently and aesthetically draws Cayley graphs, there are areas that can be looked into in order to, with sufficient time for development, further improve the quality of drawing produced.

The algorithm could be improved to cope better with edges that go in both directions, i.e. edges (u, v) and (v, u) both exist for some u, v in V. This could be done at the drawing time, or the algorithm could possibly be expanded to recognise when a generator is its own inverse, or the inverse of another generator and deal with the cases accordingly. This also has the potential for an increase in efficiency, because it is not necessary to calculate the edges generated by a generator g and its inverse g^{-1}; they can both be done from one of the other.

The nested algorithm uses produces unsatisfactory results when all the generators have order 2, with all nodes falling on a straight line. The radial algorithm produces much more desirable results in this situation, but and could be used instead. However, the results are still not always ideal and better ways of dealing with this situation should be explored.

There seem to be cases in some graphs such as the simple example (Cayley graph for group S_3) of Figure 2 in which the graph the algorithm produces are nice and symmetric, but an edge crosses unnecessarily. For denser graphs it is harder to tell if this is happening, but it is likely that it is. It should be possible to algorithmically swap around certain vertices in graphs to eliminate theses crossovers. This would create graves with less edges crossing, which is desirable when drawing nice graphs, as set out by [1].

While our experimentation with working spring-algorithm techniques into our methods yielded some success, it seems likely that it could be used in certain situations or in limited ways to achieve nice drawings. A possible technique to look further into

[1]The programs are freely available for noncommercial use by request from the authors.

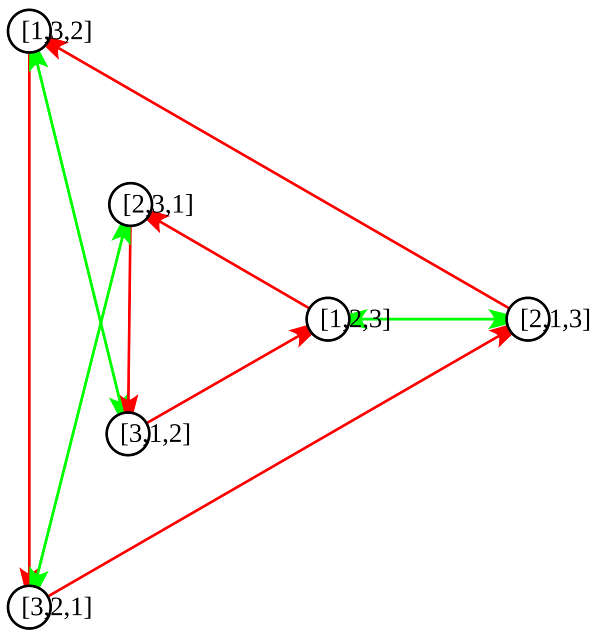


Figure 2: Example drawing of S_3 where we get too many edge crossings.

is creating a subgraph only containing the initial cosets created by the first generator, as well as any edges that go between 2 of these cosets, i.e. edges (u, v) where u from *coset1*, v from *coset2*, *coset1* $\neq$ *coset2*. This could work because a lot the mess in the graphs seem to come from these edges only, as these are the only edges with the potential to cross other edges. It would also be a lot more efficient that calculating the spring forces for all the other edges when they would already be desirably connected.

6 Conclusion

In our research we explored a number of options to work towards creating an algorithm that creates better drawings of Cayley graphs. By using theories and aspects of group theory, we were able to come up with some algorithms that are specifically suited towards drawing Cayley graphs. Our algorithms use group theory to create a drawing that shows the symmetry of a Cayley graph, is easy to interpret and does a decent job of creating a graph drawing that is aesthetically good by minimizing edge crossovers and illustrating structures of a good graph. The nested algorithm proved to be superior to the radial algorithm in almost all cases. Our algorithms will be of use when it is necessary to visualise a group or draw a Cayley graph in an easy to interpret fashion. This could be of use in many situations such as in the planning of interconnected networks.

A Adjacency Lists of Example Graphs

The first three adjacency lists are semi-direct products of cyclic groups (see [9]). The final seven lists are taken from the group tables given in [11]. The Cayley graph generators $g_1, g_2, \ldots, g_\Delta$ for each case are given as the neighbors of vertex 0 (the identity element). The i-th neighbor in each adjacency list for row j corresponds to the product $j \cdot g_i$, where g_i is the i-th generator.

$Z_4 \times_\sigma Z_{29}$:

0:	78	71	55	109	58
1:	79	72	56	110	59
2:	80	73	57	111	60
3:	81	74	29	112	61
4:	82	75	30	113	62
5:	83	76	31	114	63
6:	84	77	32	115	64
7:	85	78	33	87	65
8:	86	79	34	88	66
9:	58	80	35	89	67
10:	59	81	36	90	68
11:	60	82	37	91	69
12:	61	83	38	92	70
13:	62	84	39	93	71
14:	63	85	40	94	72
15:	64	86	41	95	73
16:	65	58	42	96	74
17:	66	59	43	97	75
18:	67	60	44	98	76
19:	68	61	45	99	77
20:	69	62	46	100	78
21:	70	63	47	101	79
22:	71	64	48	102	80
23:	72	65	49	103	81
24:	73	66	50	104	82
25:	74	67	51	105	83
26:	75	68	52	106	84
27:	76	69	53	107	85
28:	77	70	54	108	86
29:	95	98	80	3	87
30:	96	99	81	4	88
31:	97	100	82	5	89
32:	98	101	83	6	90
33:	99	102	84	7	91
34:	100	103	85	8	92
35:	101	104	86	9	93
36:	102	105	58	10	94
37:	103	106	59	11	95
38:	104	107	60	12	96
39:	105	108	61	13	97
40:	106	109	62	14	98
41:	107	110	63	15	99
42:	108	111	64	16	100
43:	109	112	65	17	101
44:	110	113	66	18	102
45:	111	114	67	19	103
46:	112	115	68	20	104
47:	113	87	69	21	105
48:	114	88	70	22	106
49:	115	89	71	23	107
50:	87	90	72	24	108
51:	88	91	73	25	109
52:	89	92	74	26	110
53:	90	93	75	27	111
54:	91	94	76	28	112
55:	92	95	77	0	113
56:	93	96	78	1	114
57:	94	97	79	2	115
58:	9	16	90	36	0
59:	10	17	91	37	1
60:	11	18	92	38	2
61:	12	19	93	39	3
62:	13	20	94	40	4
63:	14	21	95	41	5
64:	15	22	96	42	6
65:	16	23	97	43	7
66:	17	24	98	44	8
67:	18	25	99	45	9
68:	19	26	100	46	10
69:	20	27	101	47	11
70:	21	28	102	48	12
71:	22	0	103	49	13
72:	23	1	104	50	14
73:	24	2	105	51	15
74:	25	3	106	52	16
75:	26	4	107	53	17
76:	27	5	108	54	18
77:	28	6	109	55	19
78:	0	7	110	56	20
79:	1	8	111	57	21
80:	2	9	112	29	22
81:	3	10	113	30	23
82:	4	11	114	31	24
83:	5	12	115	32	25
84:	6	13	87	33	26
85:	7	14	88	34	27
86:	8	15	89	35	28
87:	50	47	7	84	29
88:	51	48	8	85	30
89:	52	49	9	86	31
90:	53	50	10	58	32
91:	54	51	11	59	33
92:	55	52	12	60	34
93:	56	53	13	61	35
94:	57	54	14	62	36
95:	29	55	15	63	37
96:	30	56	16	64	38
97:	31	57	17	65	39
98:	32	29	18	66	40
99:	33	30	19	67	41
100:	34	31	20	68	42
101:	35	32	21	69	43
102:	36	33	22	70	44
103:	37	34	23	71	45
104:	38	35	24	72	46
105:	39	36	25	73	47
106:	40	37	26	74	48
107:	41	38	27	75	49
108:	42	39	28	76	50
109:	43	40	0	77	51
110:	44	41	1	78	52
111:	45	42	2	79	53
112:	46	43	3	80	54
113:	47	44	4	81	55
114:	48	45	5	82	56
115:	49	46	6	83	57

D_{28}:

0:	1	55	7	45
1:	2	0	12	50
2:	3	1	9	47
3:	4	2	14	52
4:	5	3	11	49
5:	6	4	16	54
6:	7	5	13	51
7:	8	6	18	0
8:	9	7	15	53
9:	10	8	20	2
10:	11	9	17	55
11:	12	10	22	4
12:	13	11	19	1
13:	14	12	24	6
14:	15	13	21	3
15:	16	14	26	8
16:	17	15	23	5
17:	18	16	28	10
18:	19	17	25	7
19:	20	18	30	12
20:	21	19	27	9
21:	22	20	32	14
22:	23	21	29	11
23:	24	22	34	16
24:	25	23	31	13
25:	26	24	36	18
26:	27	25	33	15
27:	28	26	38	20
28:	29	27	35	17
29:	30	28	40	22
30:	31	29	37	19
31:	32	30	42	24
32:	33	31	39	21
33:	34	32	44	26
34:	35	33	41	23
35:	36	34	46	28
36:	37	35	43	25
37:	38	36	48	30
38:	39	37	45	27
39:	40	38	50	32
40:	41	39	47	29
41:	42	40	52	34
42:	43	41	49	31
43:	44	42	54	36
44:	45	43	51	33
45:	46	44	0	38
46:	47	45	53	35
47:	48	46	2	40
48:	49	47	55	37
49:	50	48	4	42
50:	51	49	1	39
51:	52	50	6	44
52:	53	51	3	41
53:	54	52	8	46
54:	55	53	5	43
55:	0	54	10	48

$Z_2 \times_\sigma Z_{12}$:

0:	19	23	12
1:	18	22	23
2:	17	21	22
3:	16	20	21
4:	15	19	20
5:	14	18	19
6:	13	17	18
7:	12	16	17
8:	23	15	16
9:	22	14	15
10:	21	13	14
11:	20	12	13
12:	7	11	0
13:	6	10	11
14:	5	9	10
15:	4	8	9
16:	3	7	8
17:	2	6	7
18:	1	5	6
19:	0	4	5
20:	11	3	4
21:	10	2	3
22:	9	1	2
23:	8	0	1

$Z_2 \times Z_2 \times Z_3 \times Z_4$:

0:	1	4	24	12
1:	2	5	25	13
2:	3	6	26	14
3:	0	7	27	15
4:	5	8	28	16
5:	6	9	29	17
6:	7	10	30	18
7:	4	11	31	19
8:	9	0	32	20
9:	10	1	33	21
10:	11	2	34	22
11:	8	3	35	23
12:	13	16	36	0
13:	14	17	37	1
14:	15	18	38	2
15:	12	19	39	3
16:	17	20	40	4
17:	18	21	41	5
18:	19	22	42	6
19:	16	23	43	7
20:	21	12	44	8
21:	22	13	45	9
22:	23	14	46	10
23:	20	15	47	11
24:	25	28	0	36
25:	26	29	1	37
26:	27	30	2	38
27:	24	31	3	39
28:	29	32	4	40
29:	30	33	5	41
30:	31	34	6	42
31:	28	35	7	43
32:	33	24	8	44
33:	34	25	9	45
34:	35	26	10	46
35:	32	27	11	47
36:	37	40	12	24
37:	38	41	13	25
38:	39	42	14	26
39:	36	43	15	27
40:	41	44	16	28
41:	42	45	17	29
42:	43	46	18	30
43:	40	47	19	31
44:	45	36	20	32
45:	46	37	21	33
46:	47	38	22	34
47:	44	39	23	35

$Z_9 \times Z_3$:

0:	3	17	23	5	1	9	19
1:	4	15	21	3	2	10	20
2:	5	16	22	4	0	11	18
3:	6	20	26	8	4	12	22
4:	7	18	24	6	5	13	23
5:	8	19	25	7	3	14	21
6:	9	23	2	11	7	15	25
7:	10	21	0	9	8	16	26
8:	11	22	1	10	6	17	24
9:	12	26	5	14	10	18	1
10;	13	24	3	12	11	19	2
11:	14	25	4	13	9	20	0
12:	15	2	8	17	13	21	4
13:	16	0	6	15	14	22	5
14:	17	1	7	16	12	23	3
15:	18	5	11	20	16	24	7
16:	19	3	9	18	17	25	8
17:	20	4	10	19	15	26	6
18:	21	8	14	23	19	0	10
19:	22	6	12	21	20	1	11
20:	23	7	13	22	18	2	9
21:	24	11	17	26	22	3	13
22:	25	9	15	24	23	4	14
23:	26	10	16	25	21	5	12
24:	0	14	20	2	25	6	16
25:	1	12	18	0	26	7	17
26:	2	13	19	1	24	8	15

$Q_8 \times Z_2$:

0:	1	8	16	23
1:	2	9	17	16
2:	3	10	18	17
3:	4	11	19	18
4:	5	12	20	19
5:	6	13	21	20
6:	7	14	22	21
7:	0	15	23	22
8:	15	4	24	25
9:	8	5	25	26
10:	9	6	26	27
11:	10	7	27	28
12:	11	0	28	29
13:	12	1	29	30
14:	13	2	30	31
15:	14	3	31	24
16:	17	24	0	7
17:	18	25	1	0
18:	19	26	2	1
19:	20	27	3	2
20:	21	28	4	3
21:	22	29	5	4
22:	23	20	6	5
23:	16	31	7	6
24:	31	20	8	9
25:	24	21	9	10
26:	25	22	10	11
27:	26	23	11	12
28:	27	16	12	13
29:	28	17	13	14
30:	29	18	14	15
31:	30	19	15	8

S_4:

0:	16	12	6	20
1:	17	13	7	21
2:	18	14	4	22
3:	19	15	5	23
4:	20	16	11	12
5:	21	17	10	13
6:	22	18	9	14
7:	23	19	8	15
8:	12	20	1	16
9:	13	21	0	17
10:	14	22	3	18
11:	15	23	2	19
12:	9	0	20	6
13:	8	1	21	7
14:	11	2	22	4
15:	10	3	23	5
16:	2	4	12	11
17:	3	5	13	10
18:	0	6	14	9
19:	1	7	15	8
20:	7	8	16	1
21:	6	9	17	0
22:	5	10	18	3
23:	4	11	19	2

$Q \times Z_3$:

0:	1	4	8	22
1:	2	5	9	23
2:	3	6	10	20
3:	0	7	11	21
4:	7	2	12	16
5:	4	3	13	17
6:	5	0	14	18
7:	6	1	15	19
8:	9	12	16	6
9:	10	13	17	7
10:	11	14	18	4
11:	8	15	19	5
12:	15	10	20	0
13:	12	11	21	1
14:	13	8	22	2
15:	14	9	23	3
16:	17	20	0	14
17:	18	21	1	15
18:	19	22	2	12
19:	16	23	3	13
20:	23	18	4	8
21:	20	19	5	9
22:	21	16	6	10
23:	22	17	7	11

Q_{10}:

0:	1	10	17	14	4
1:	2	11	18	15	5
2:	3	12	19	16	6
3:	4	13	10	17	7
4:	5	14	11	18	8
5:	6	15	12	19	9
6:	7	16	13	10	0
7:	8	17	14	11	1
8:	9	18	15	12	2
9:	0	19	16	13	3
10:	19	5	8	1	16
11:	10	6	9	2	17
12:	11	7	0	3	18
13:	12	8	1	4	19
14:	13	9	2	5	10
15:	14	0	3	6	11
16:	15	1	4	7	12
17:	16	2	5	8	13
18:	17	3	6	9	14
19:	18	4	7	0	15

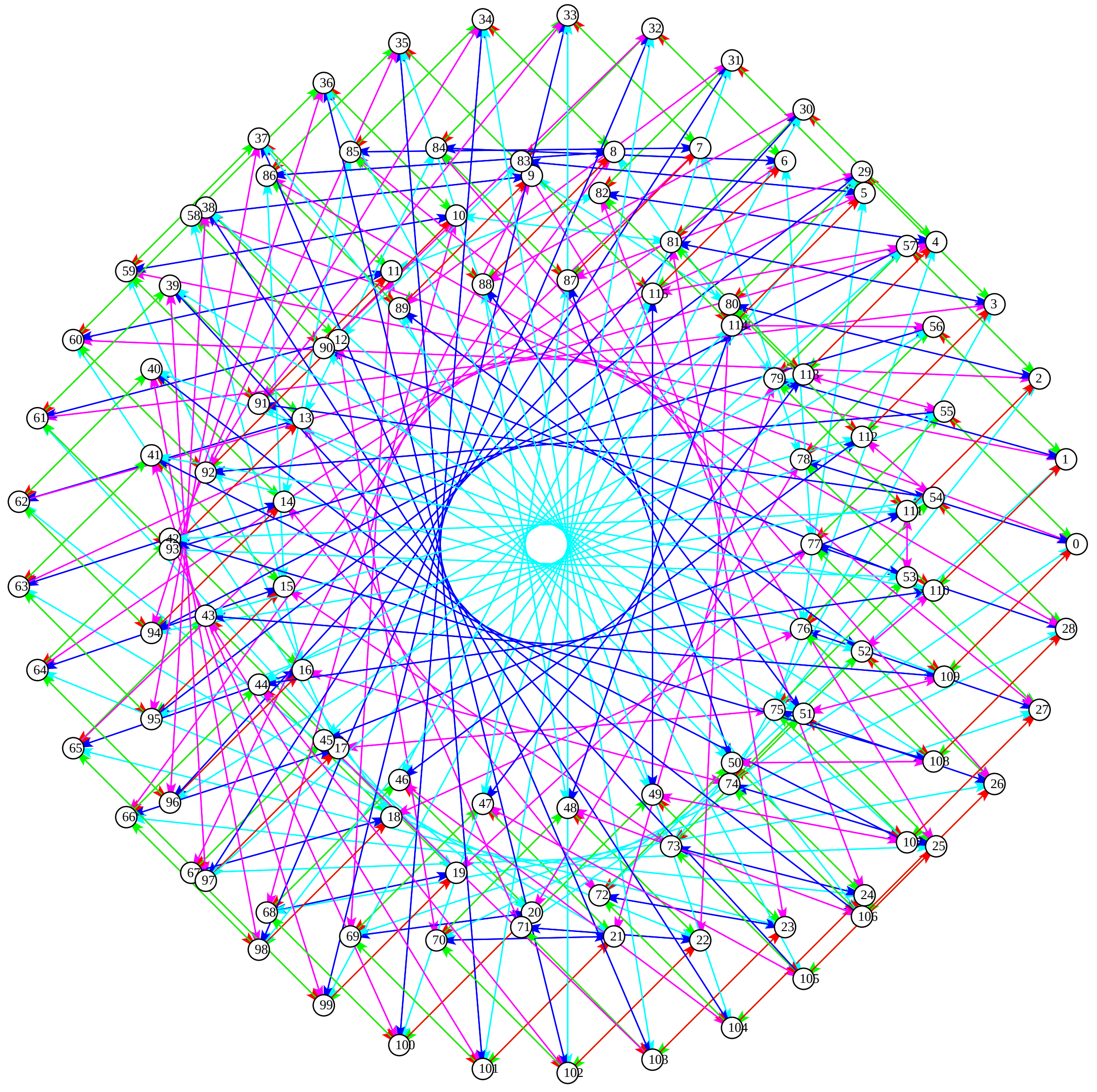
A_4:

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

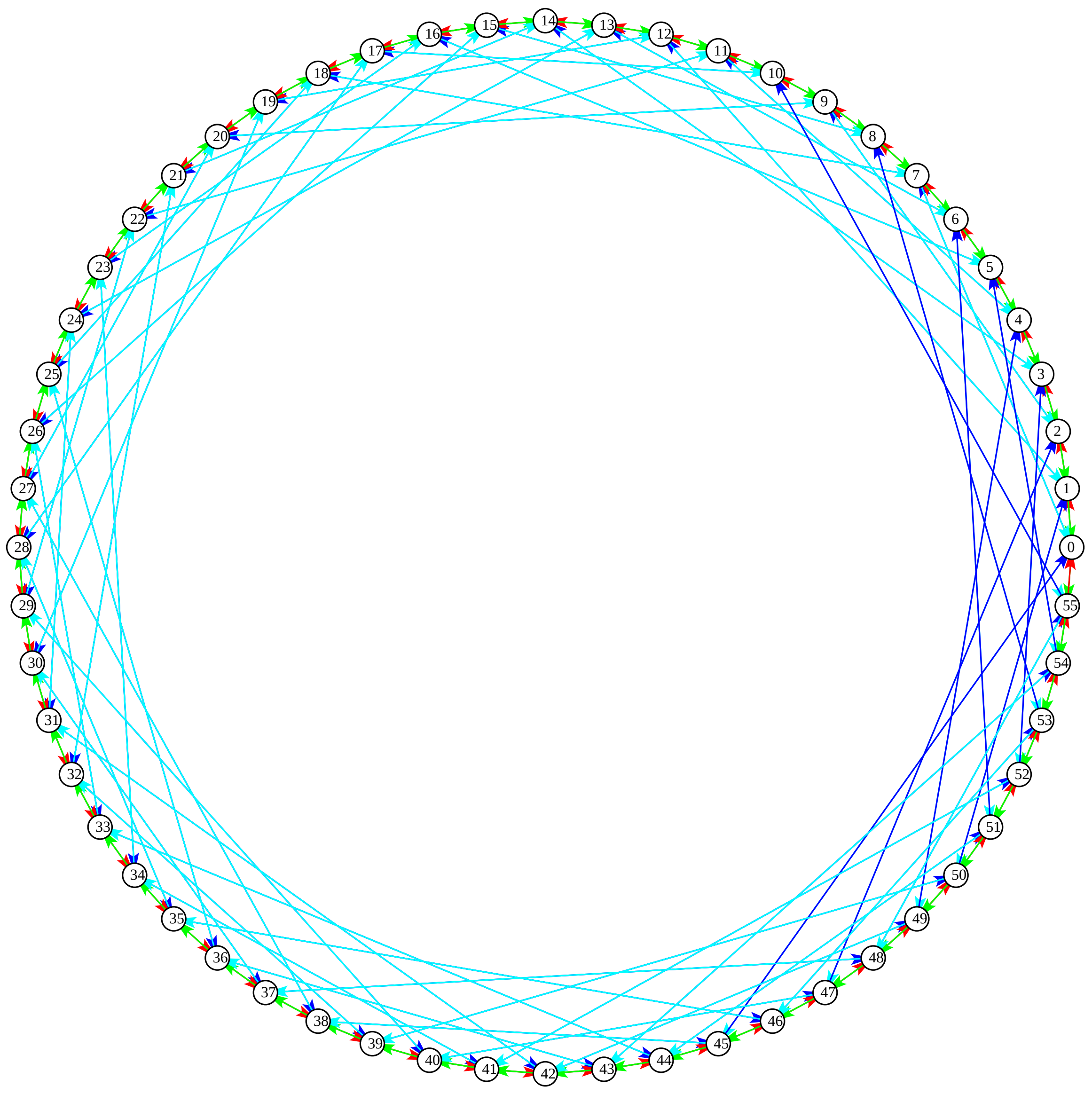
B Drawings of the Graphs in Appendix A

B.1 Radial drawings

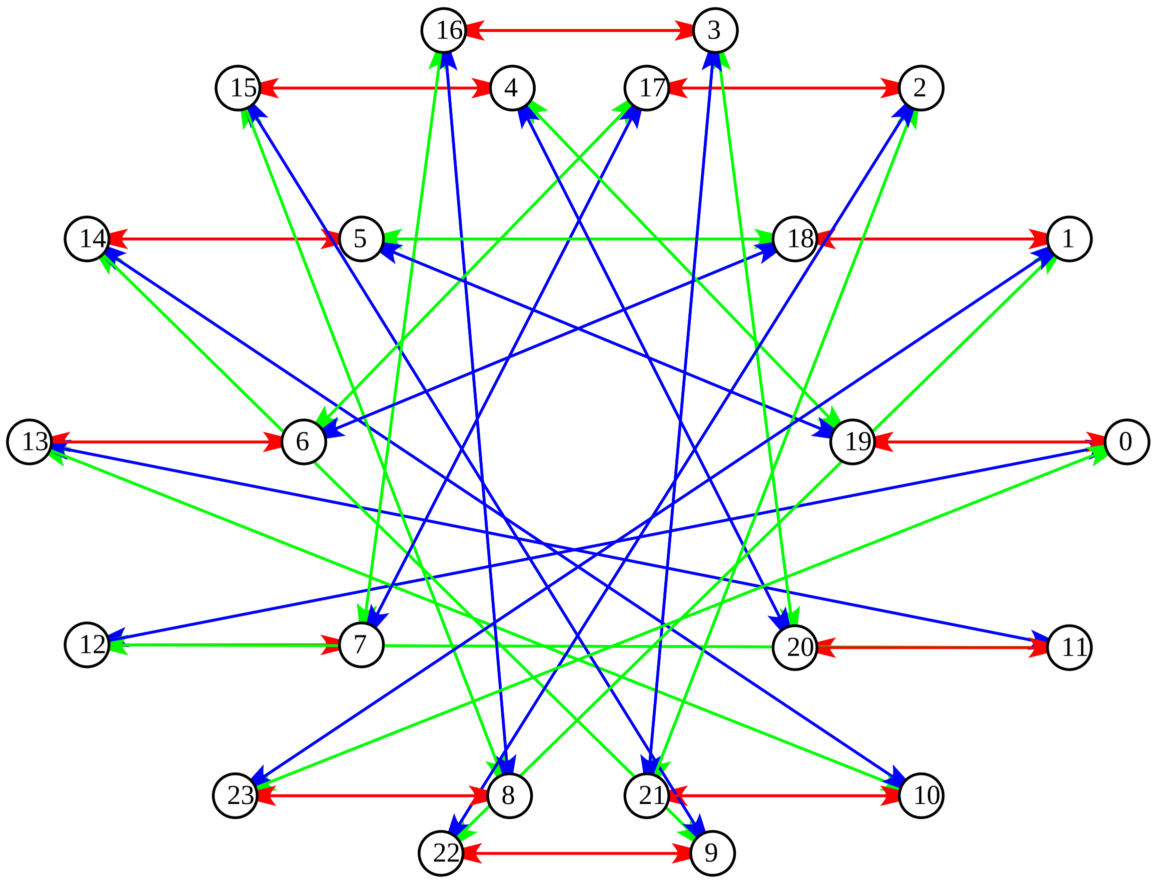
$Z_4 \times_\sigma Z_{29}$:



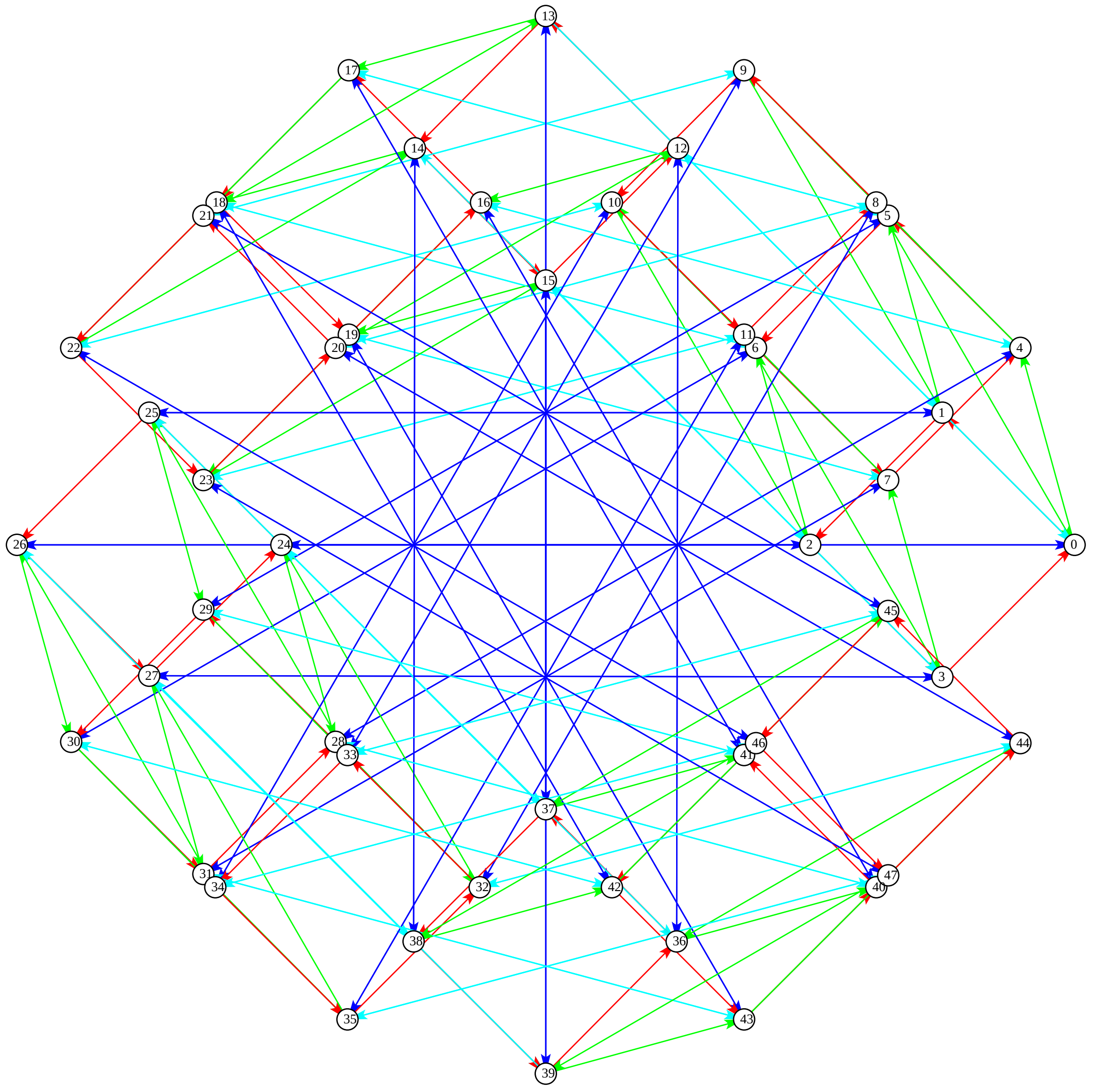
D_{28}:


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55

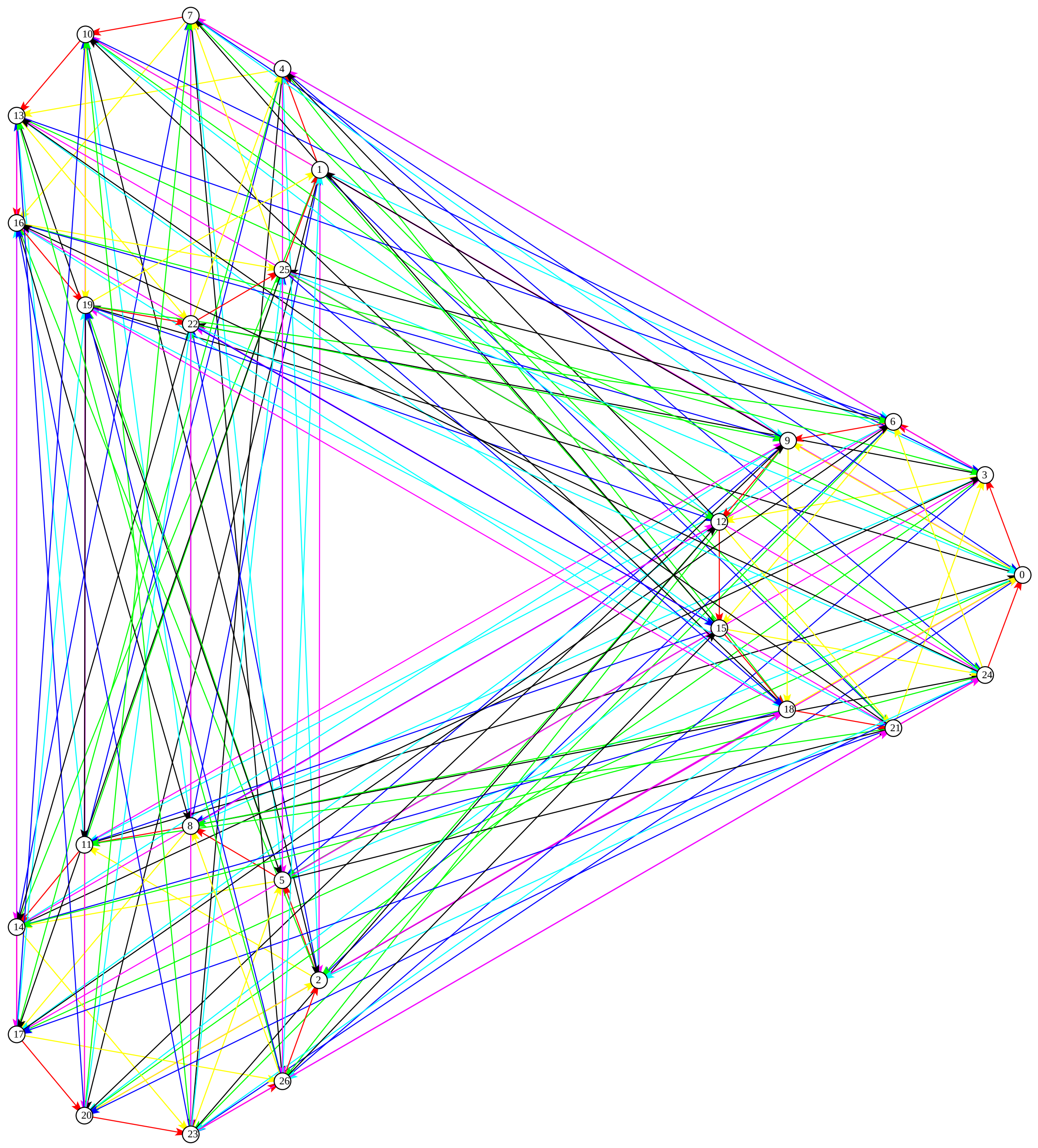
$Z_2 \times_\sigma Z_{12}$:



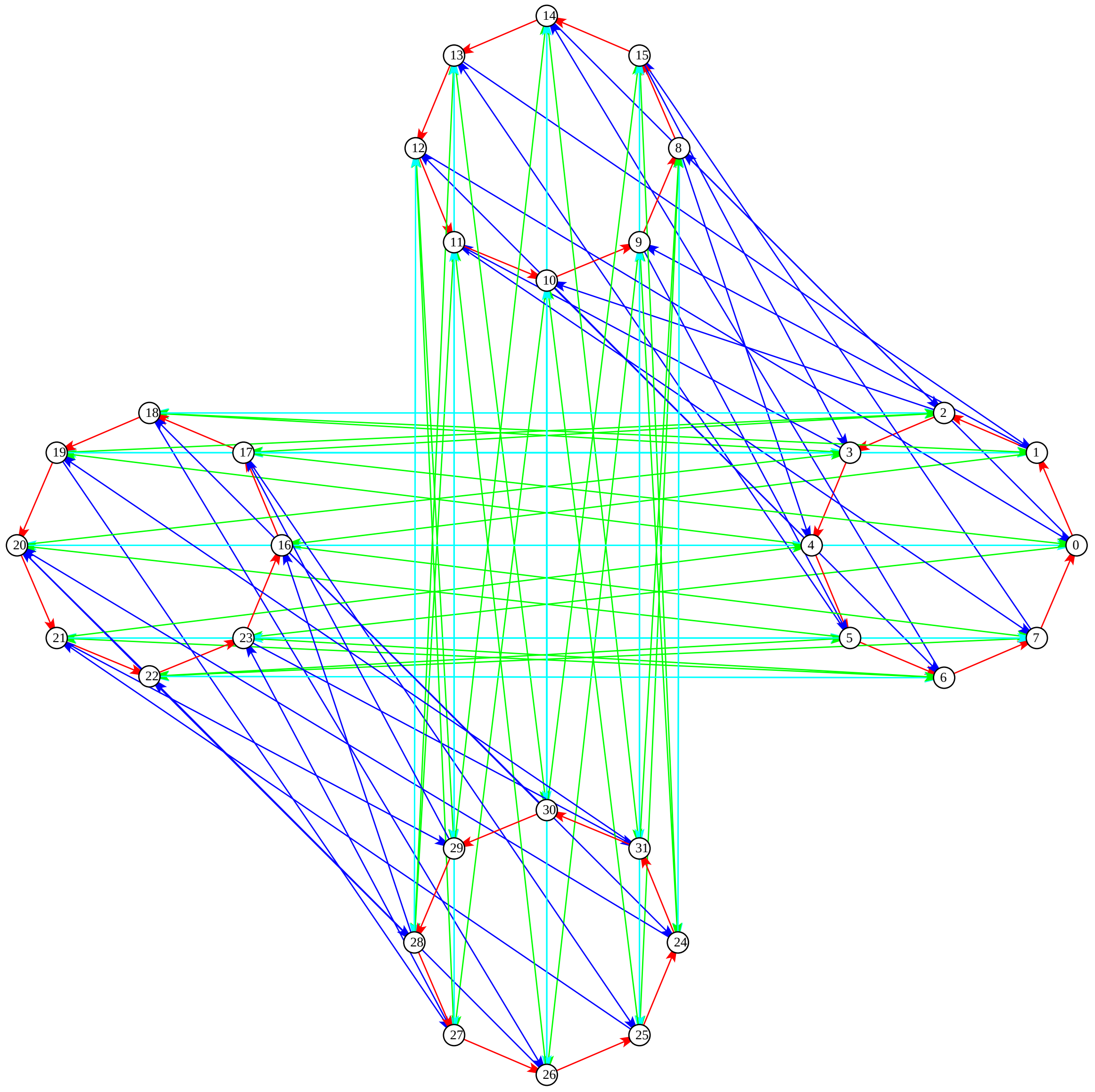
$Z_2 \times Z_2 \times Z_3 \times Z_4$:



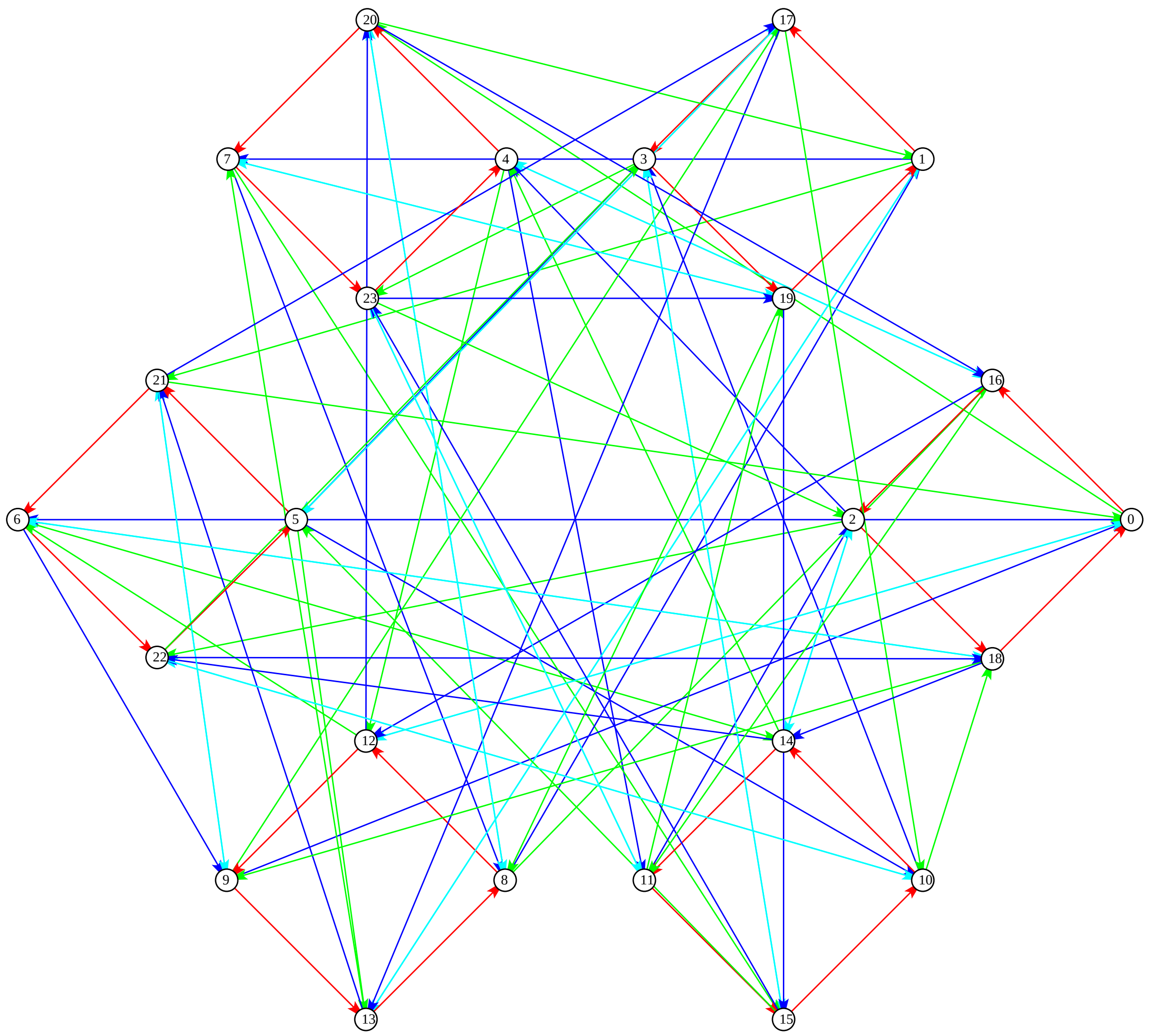
$Z_9 \times Z_3$:



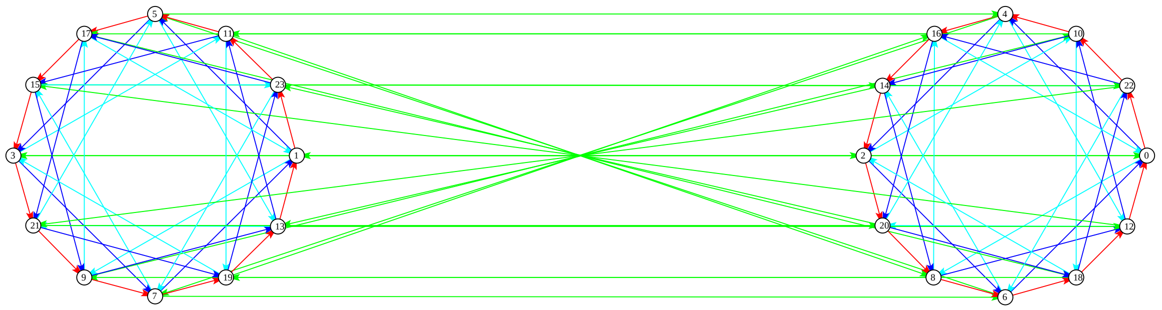
$Q_8 \times Z_2$:


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

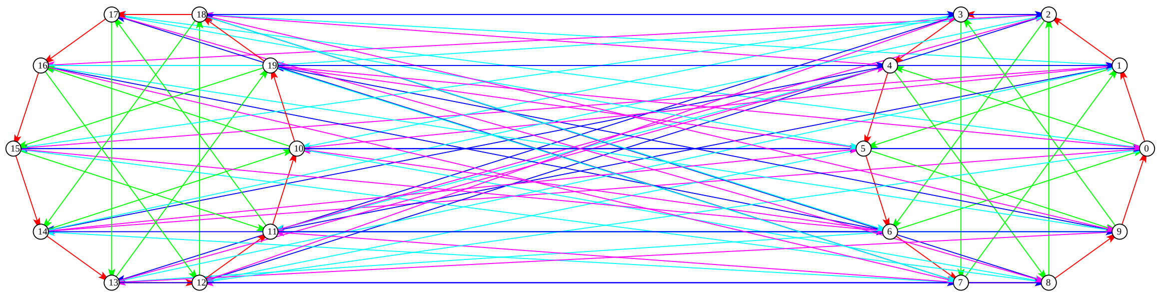
S_4:



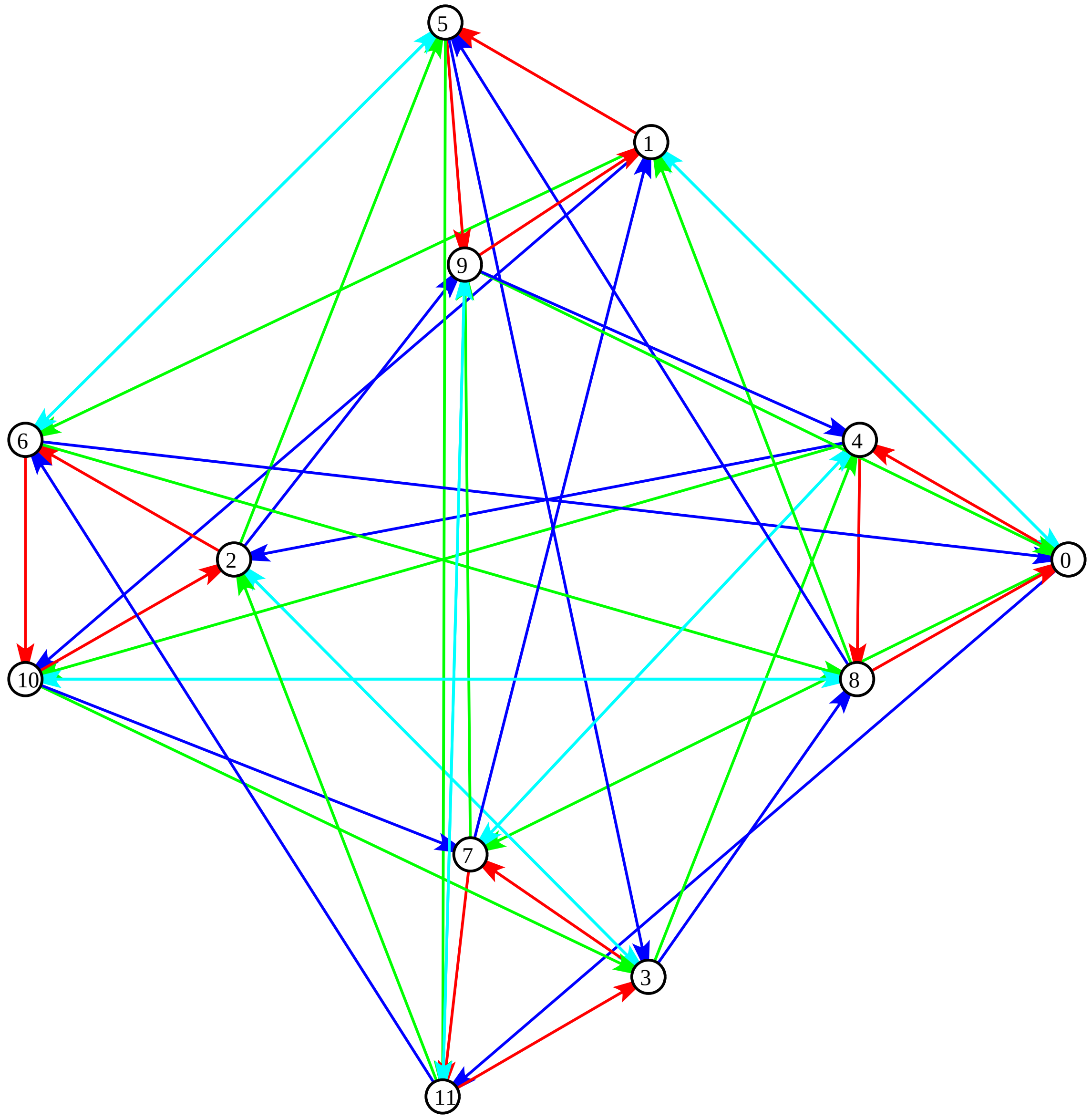
$Q \times Z_3$:



Q_{10}:



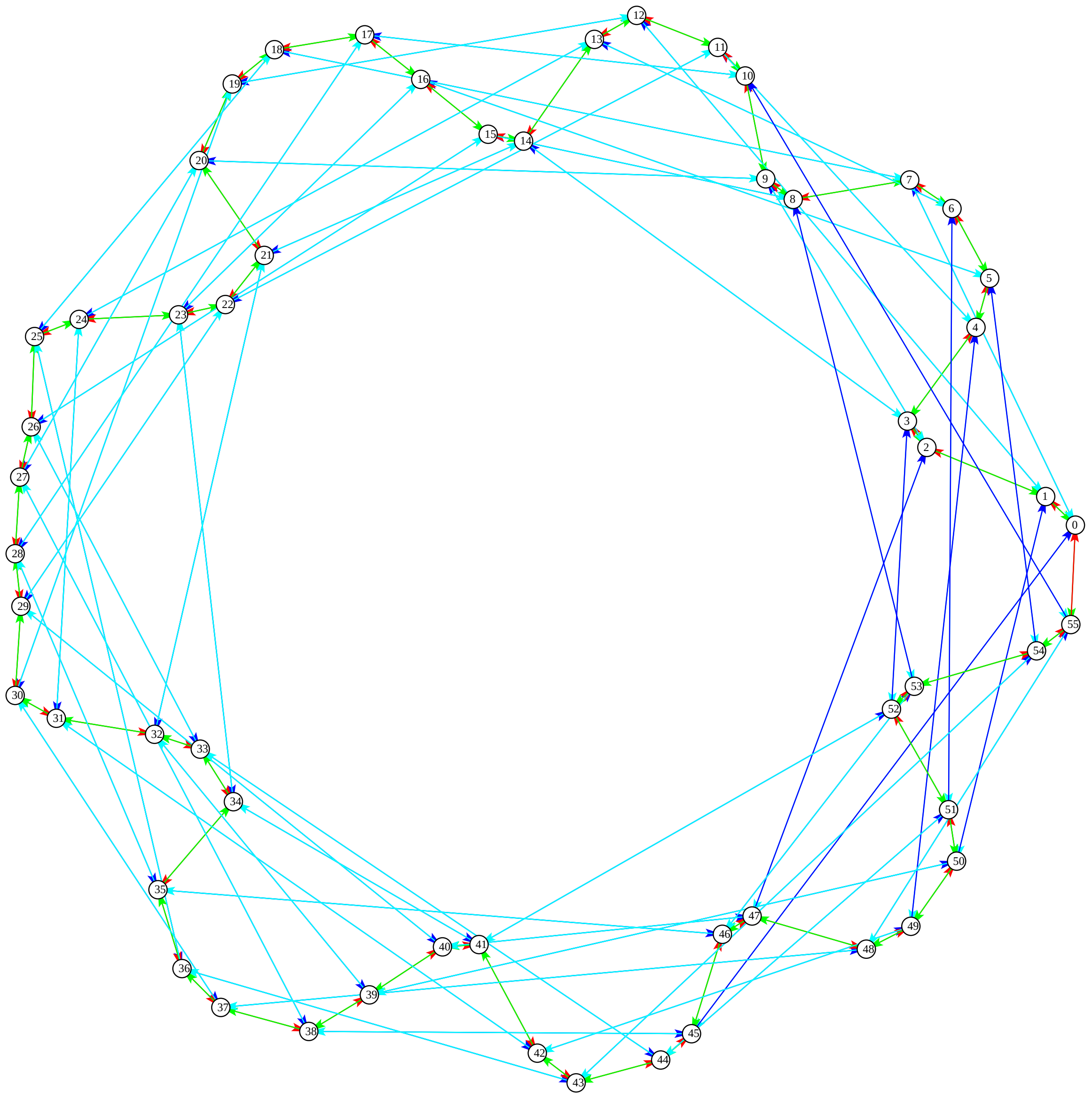
A_4:



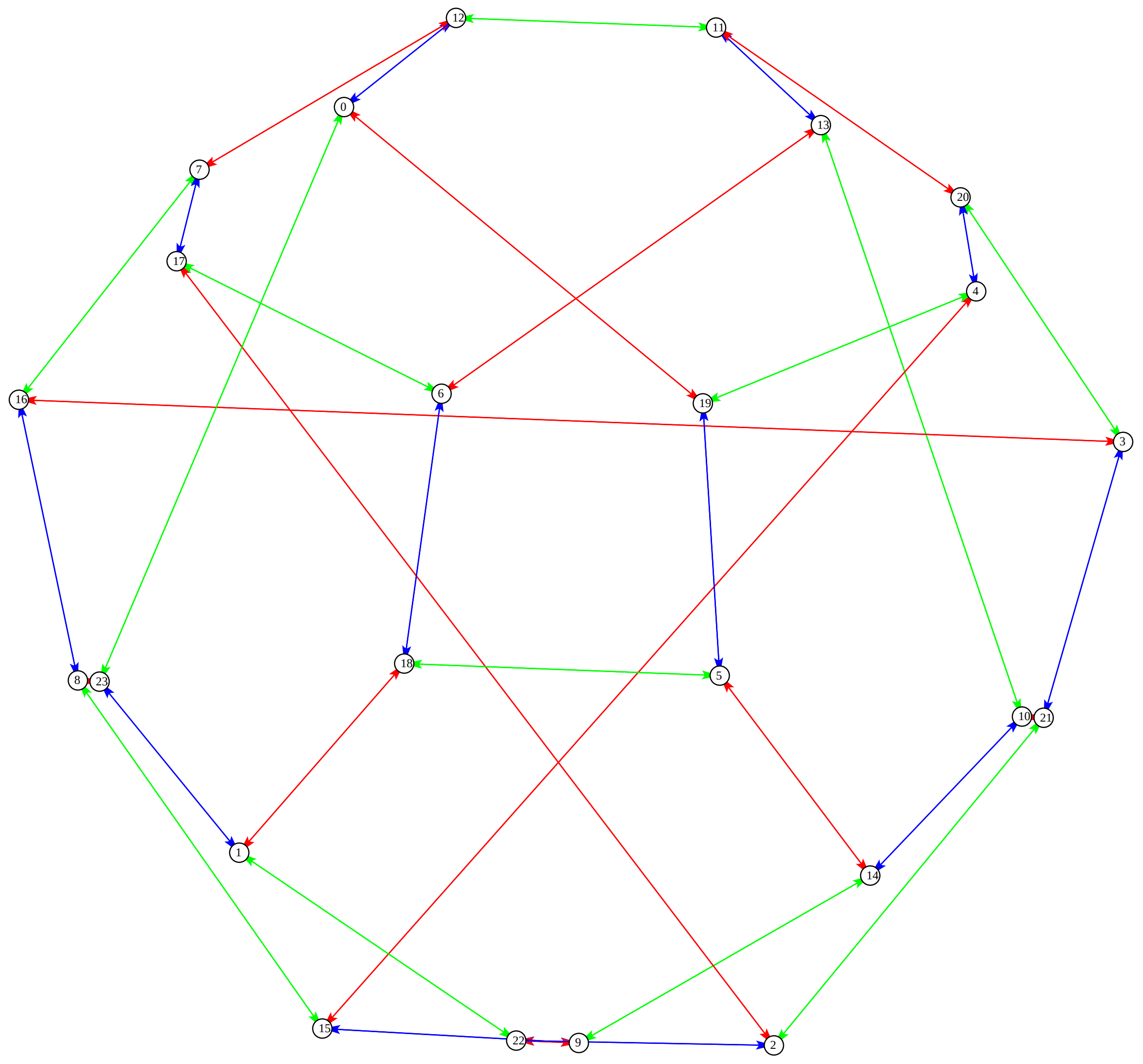
B.2 Force directed with radial base

$Z_4 \times_\sigma Z_{29}$: (too cluttered to display)

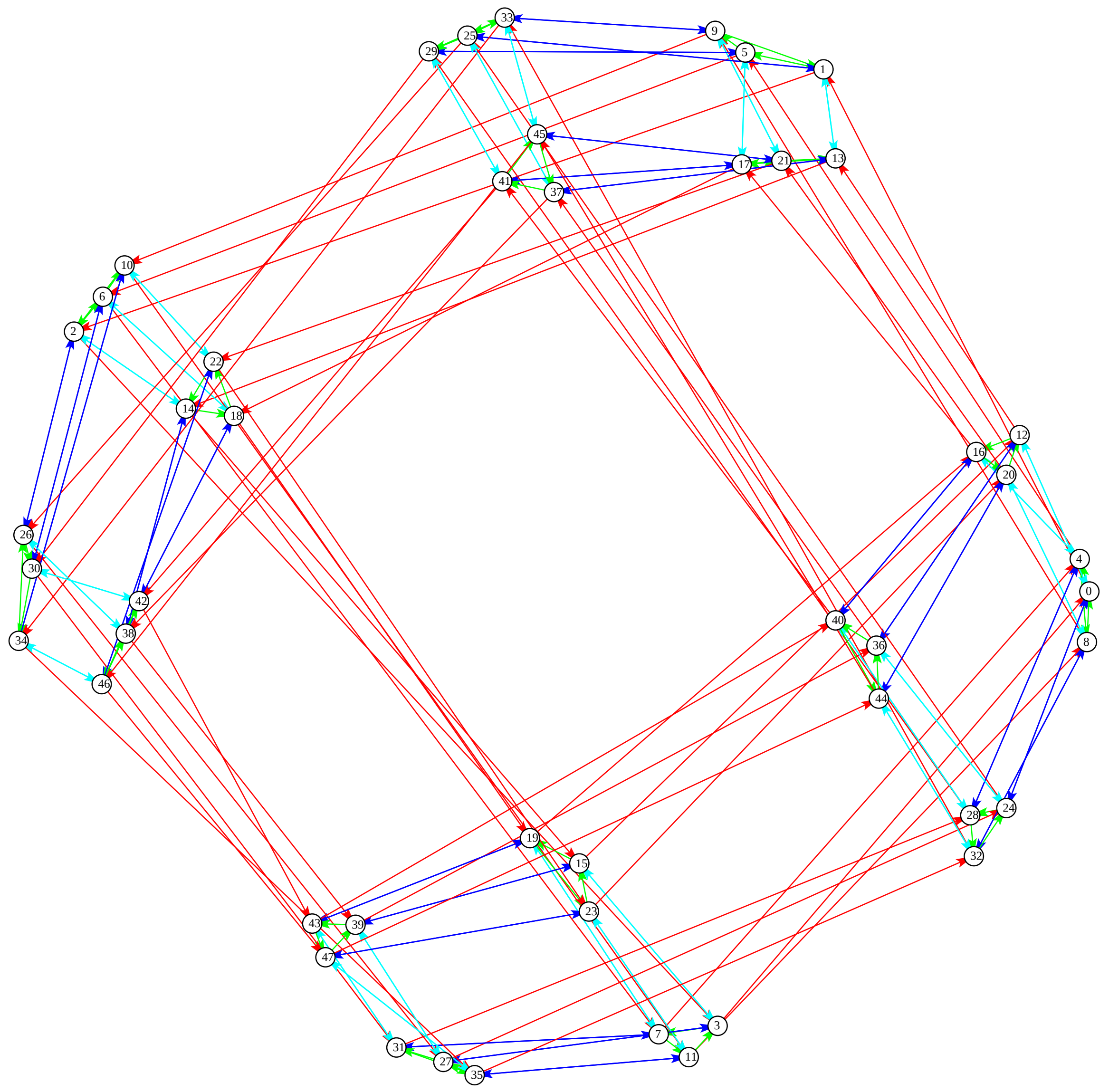
D_{28}:



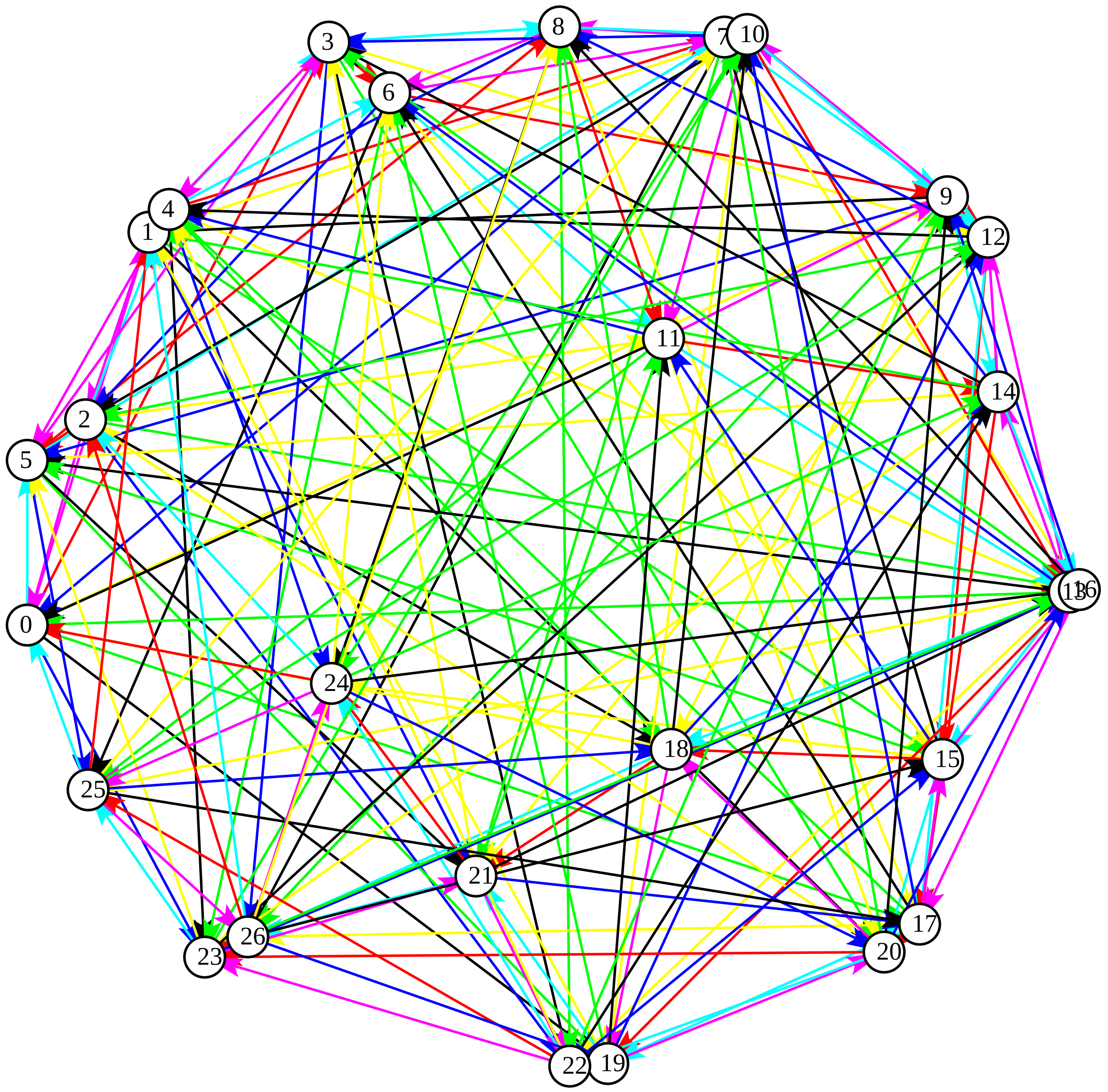
$Z_2 \times_\sigma Z_{12}$:



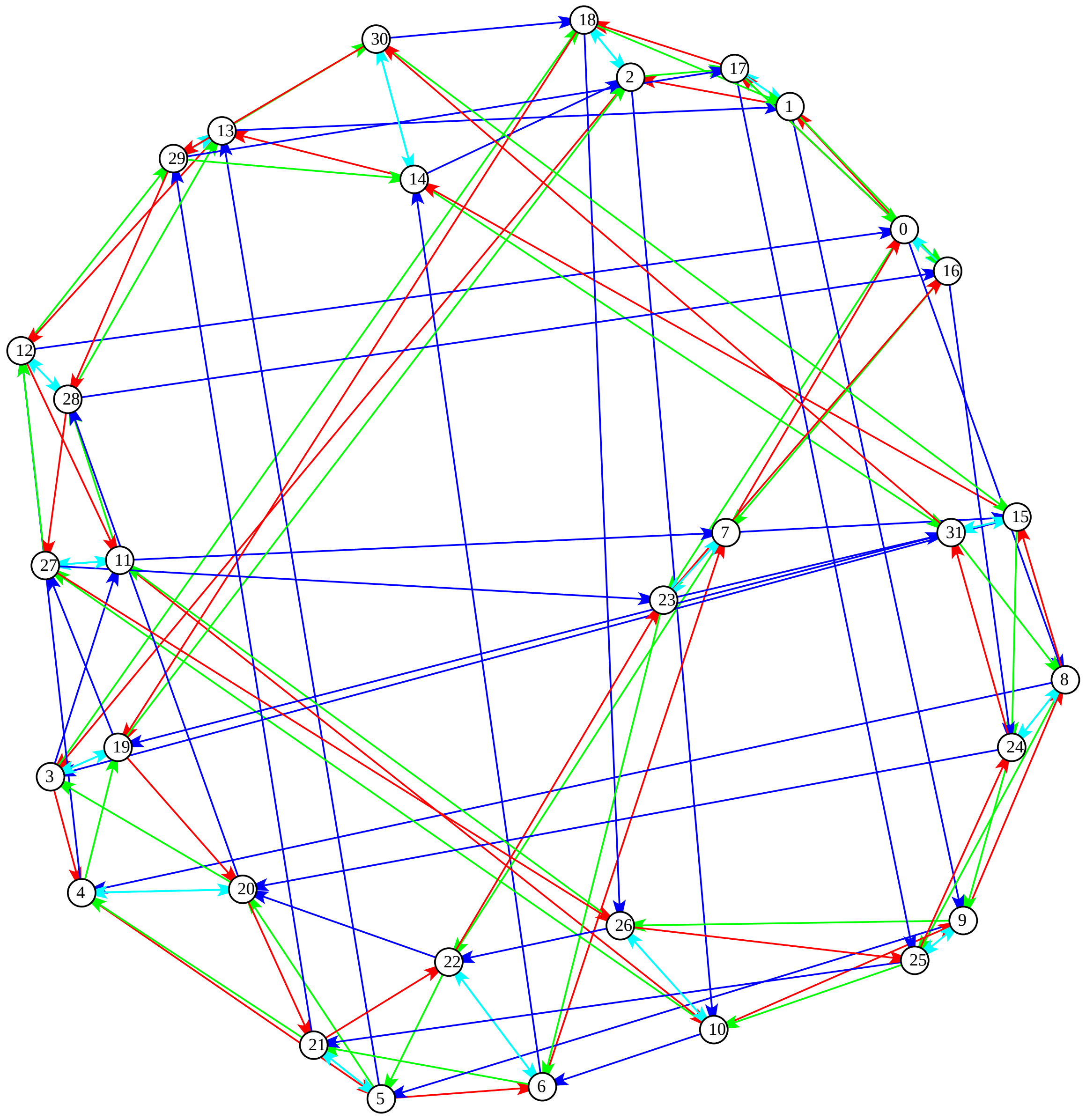
$Z_2 \times Z_2 \times Z_3 \times Z_4$:



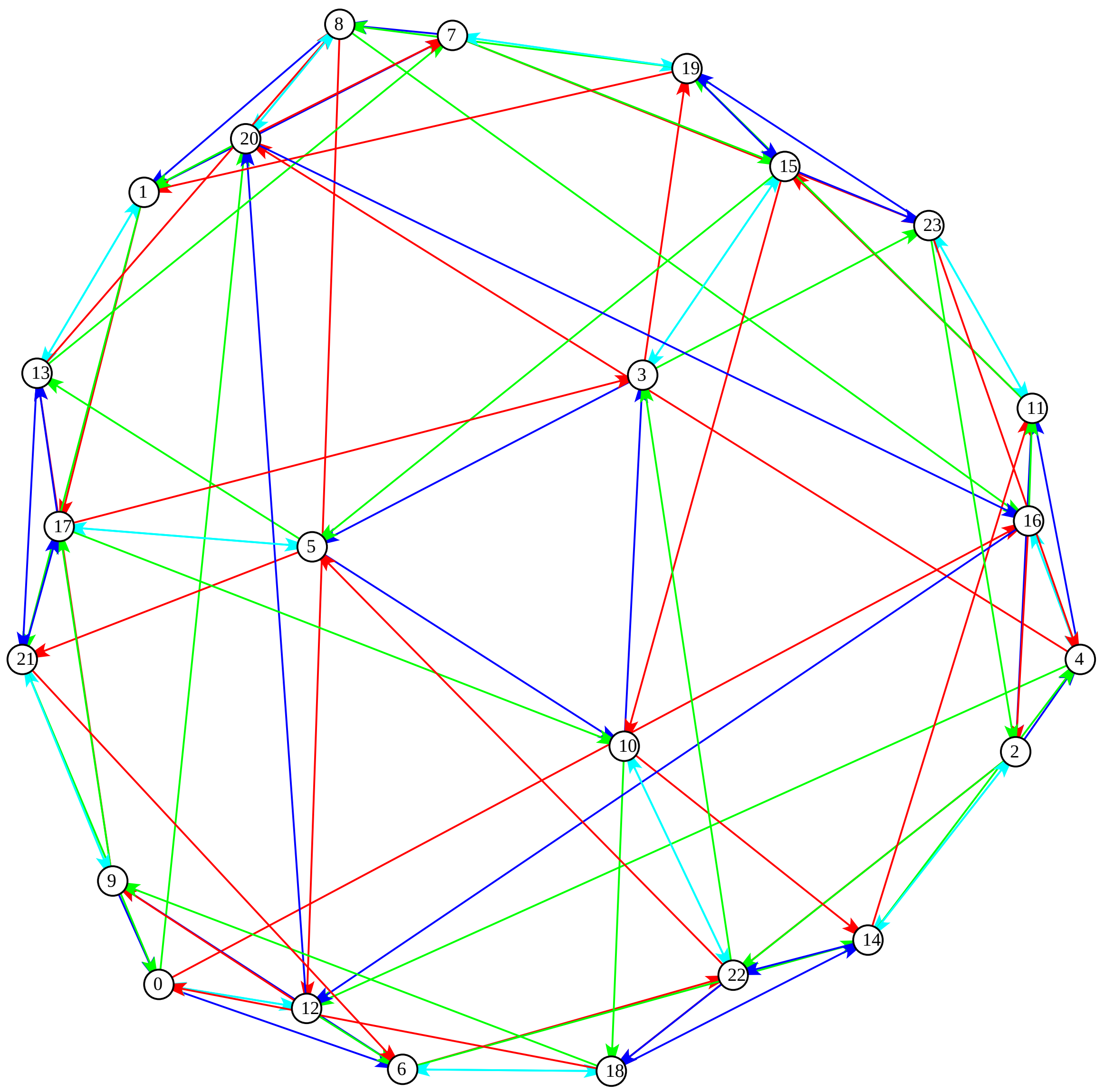
$Z_9 \times Z_3$:



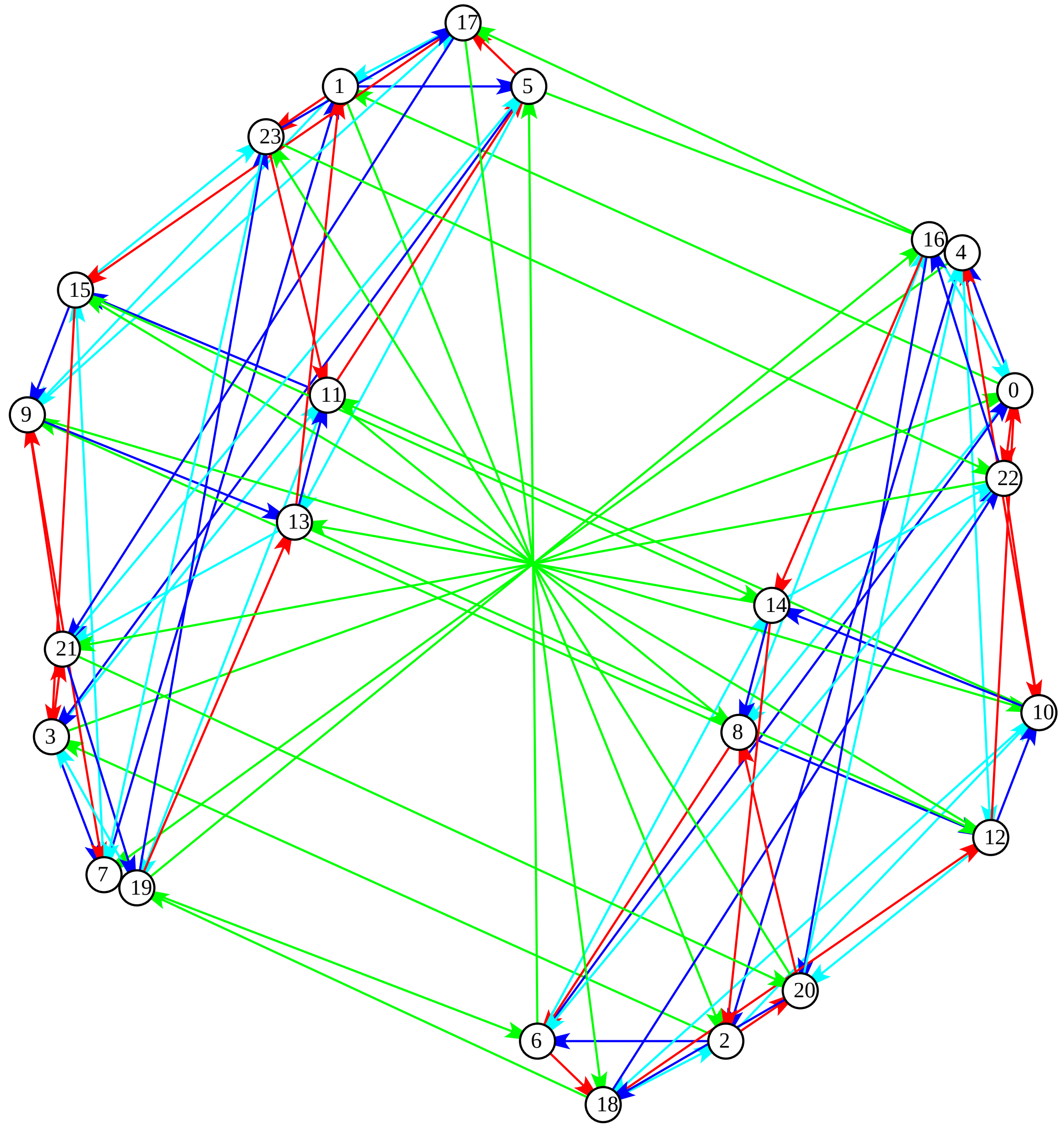
$Q_8 \times Z_2$:


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

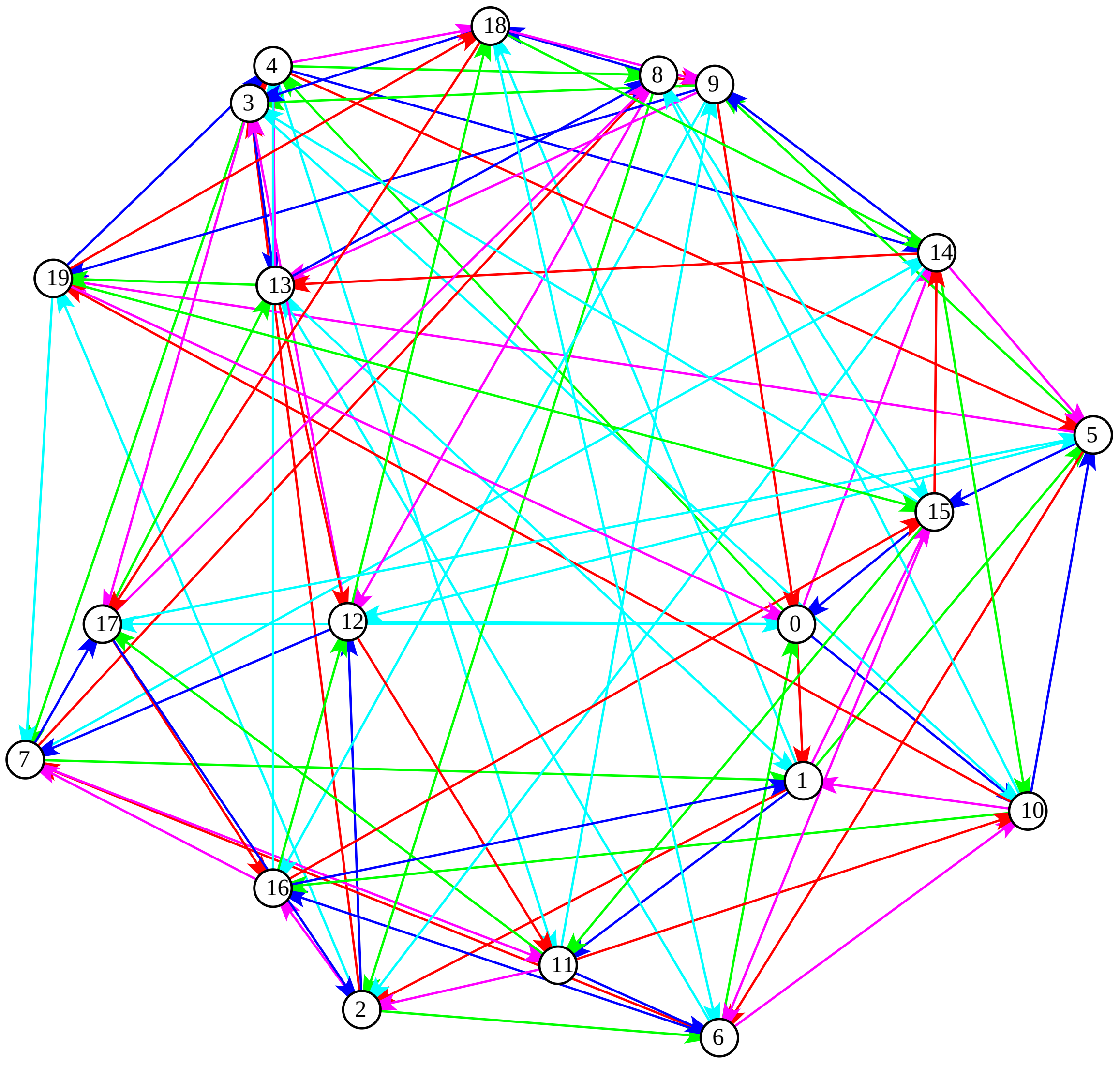
S_4:



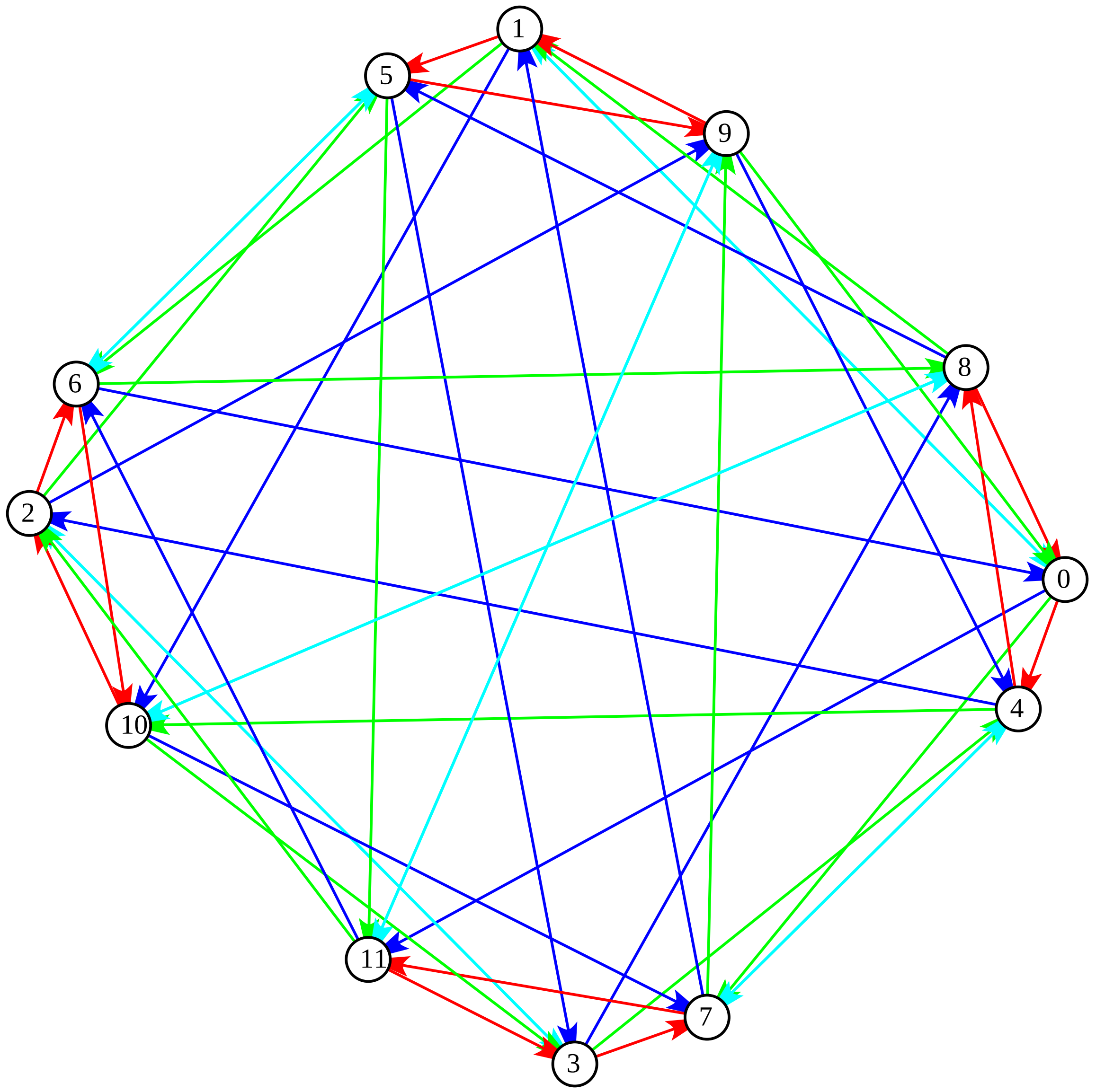
$Q \times Z_3$:



Q_{10}:

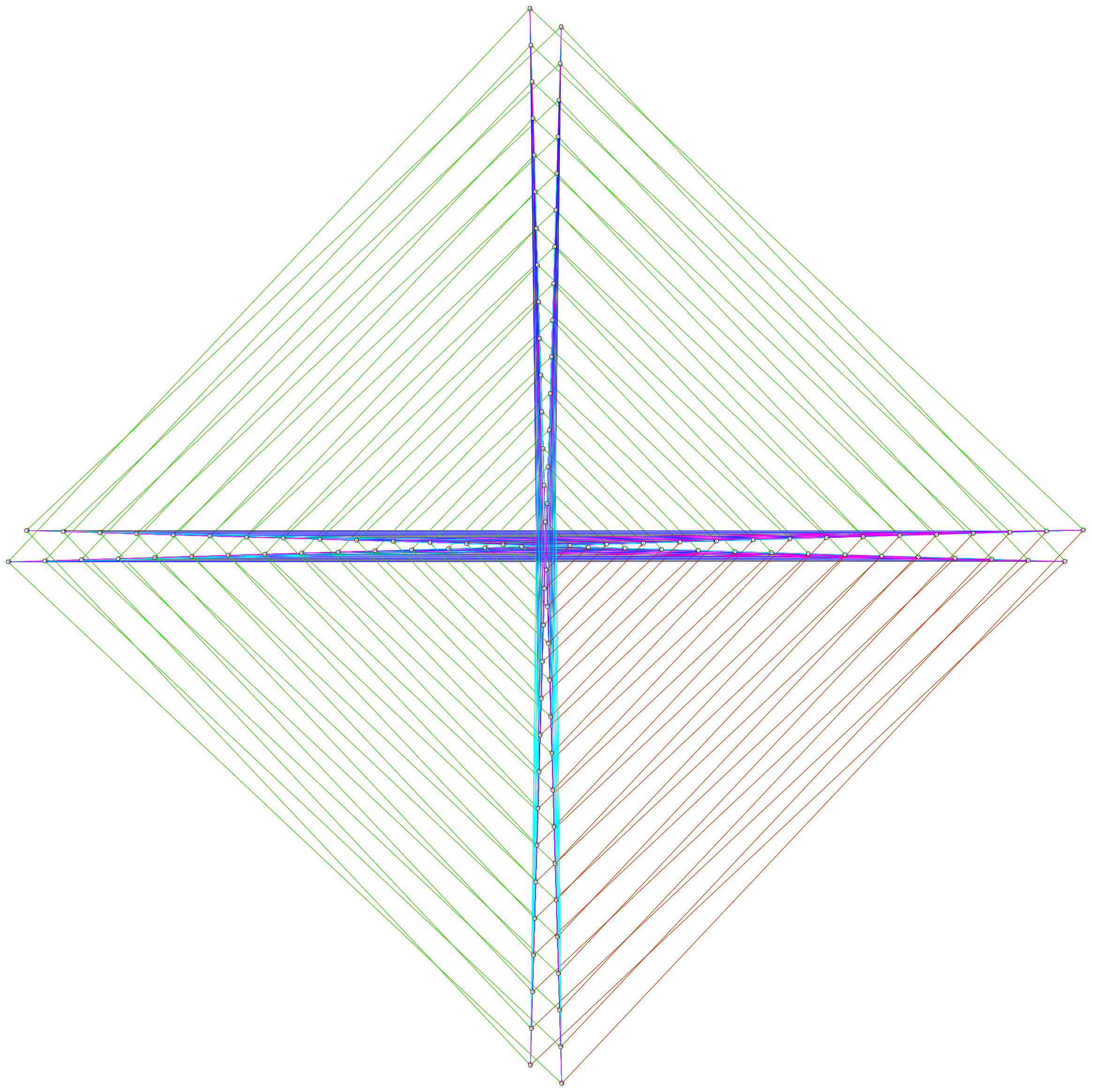


A_4:

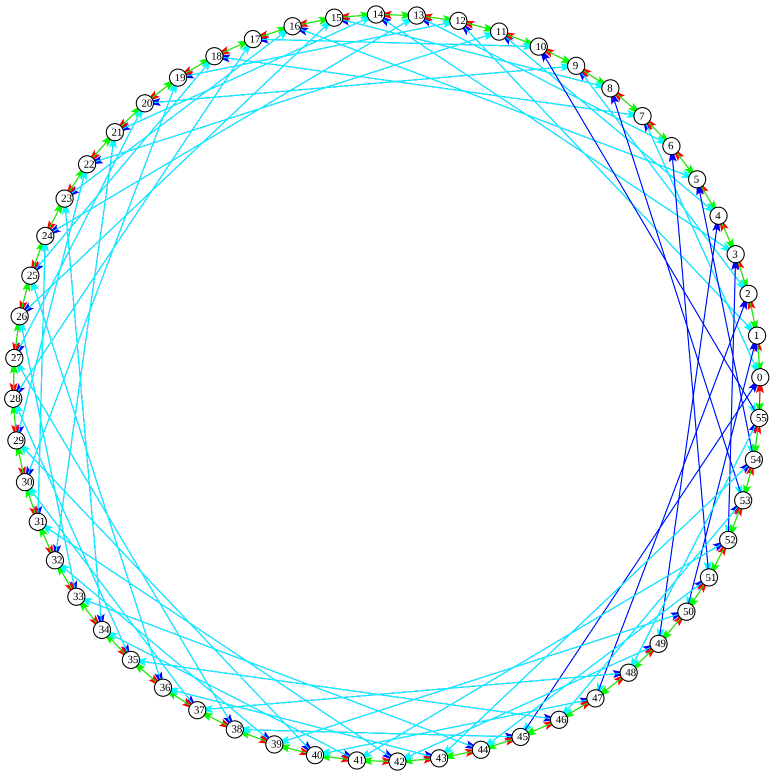


B.3 Nested drawings

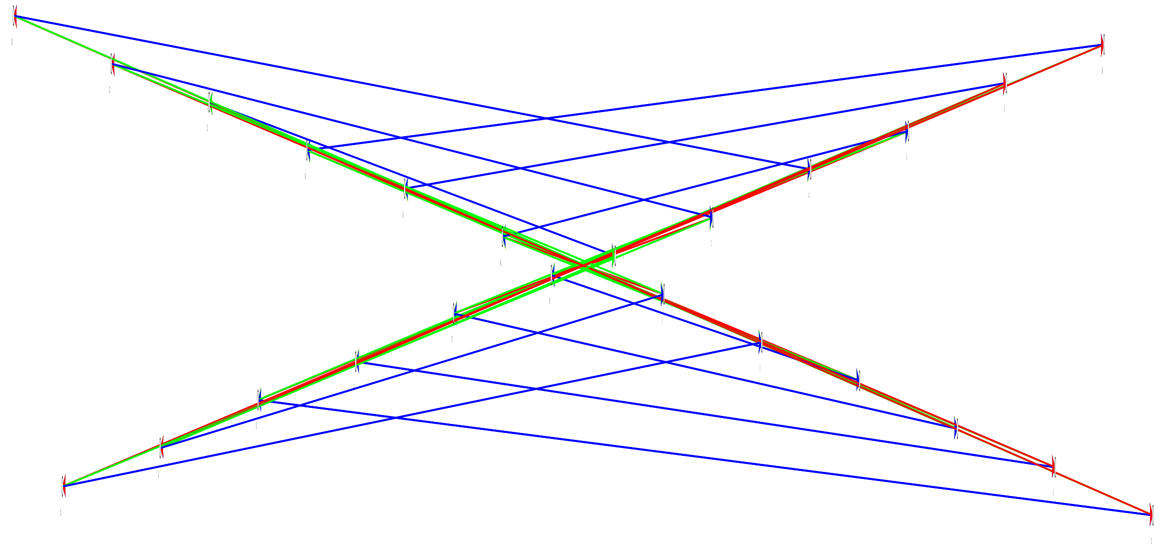
$Z_4 \times_\sigma Z_{29}$:



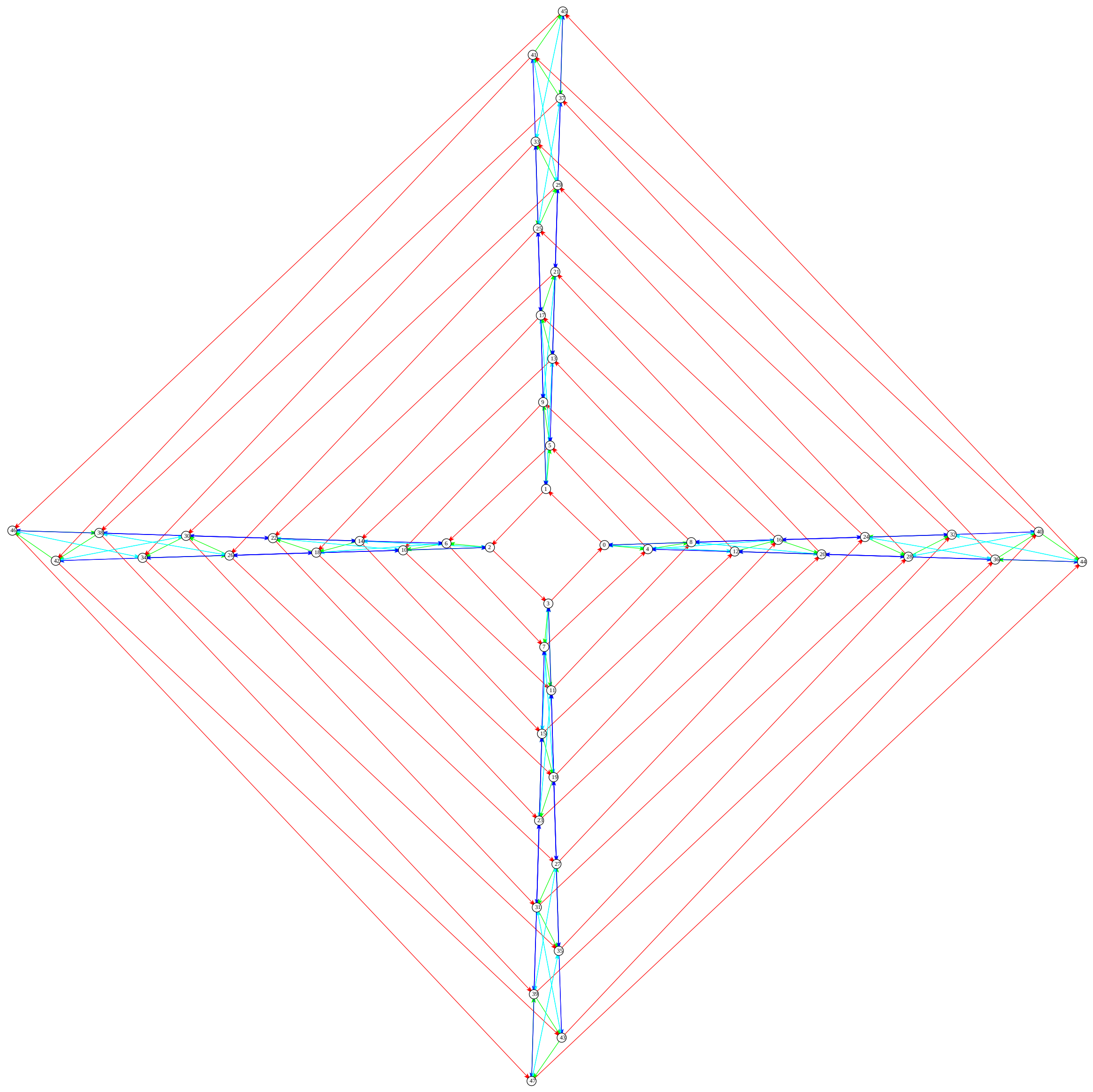
D_{28}:



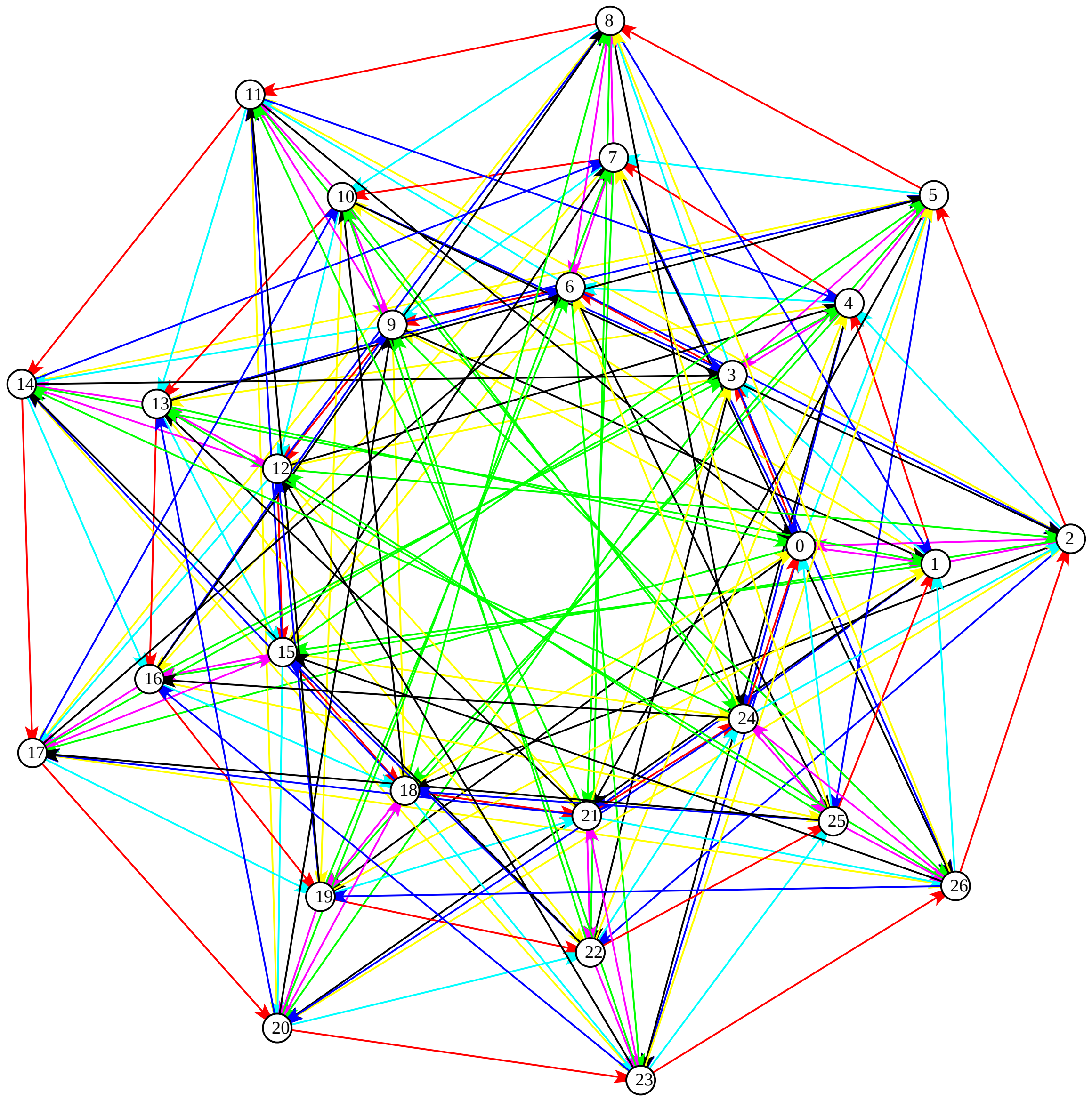
$Z_2 \times_\sigma Z_{12}$:



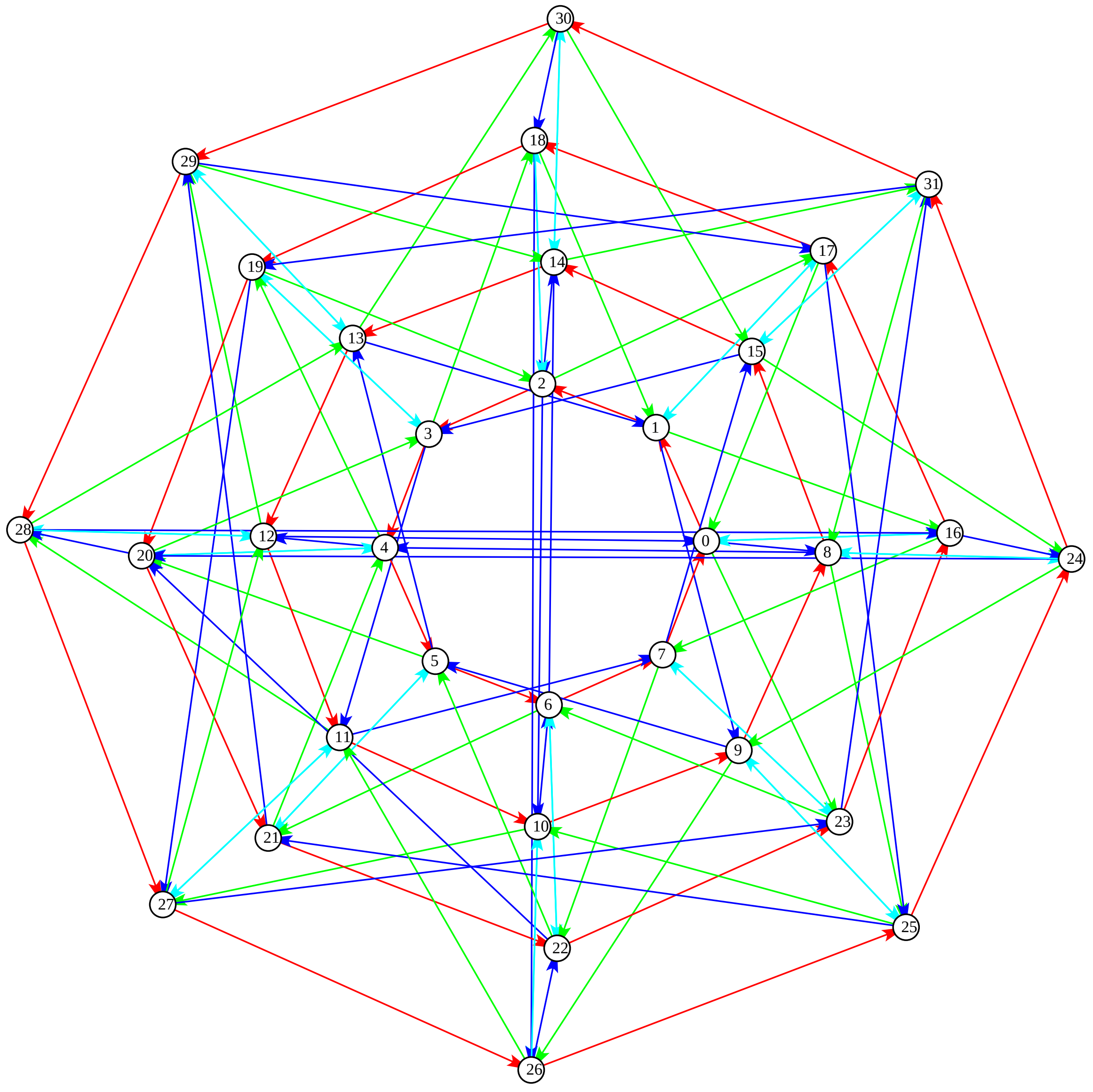
$Z_2 \times Z_2 \times Z_3 \times Z_4$:



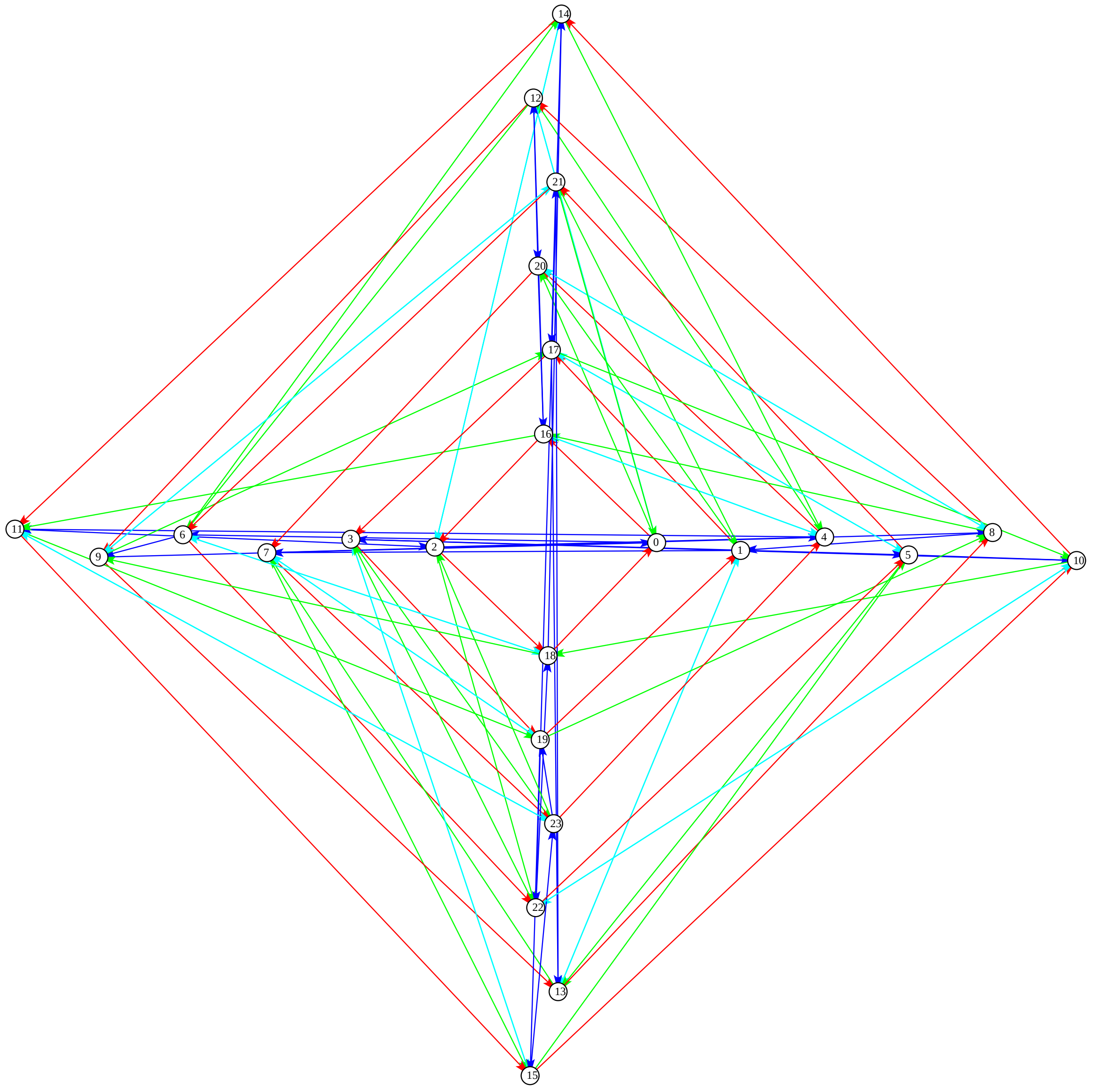
$Z_9 \times Z_3$:



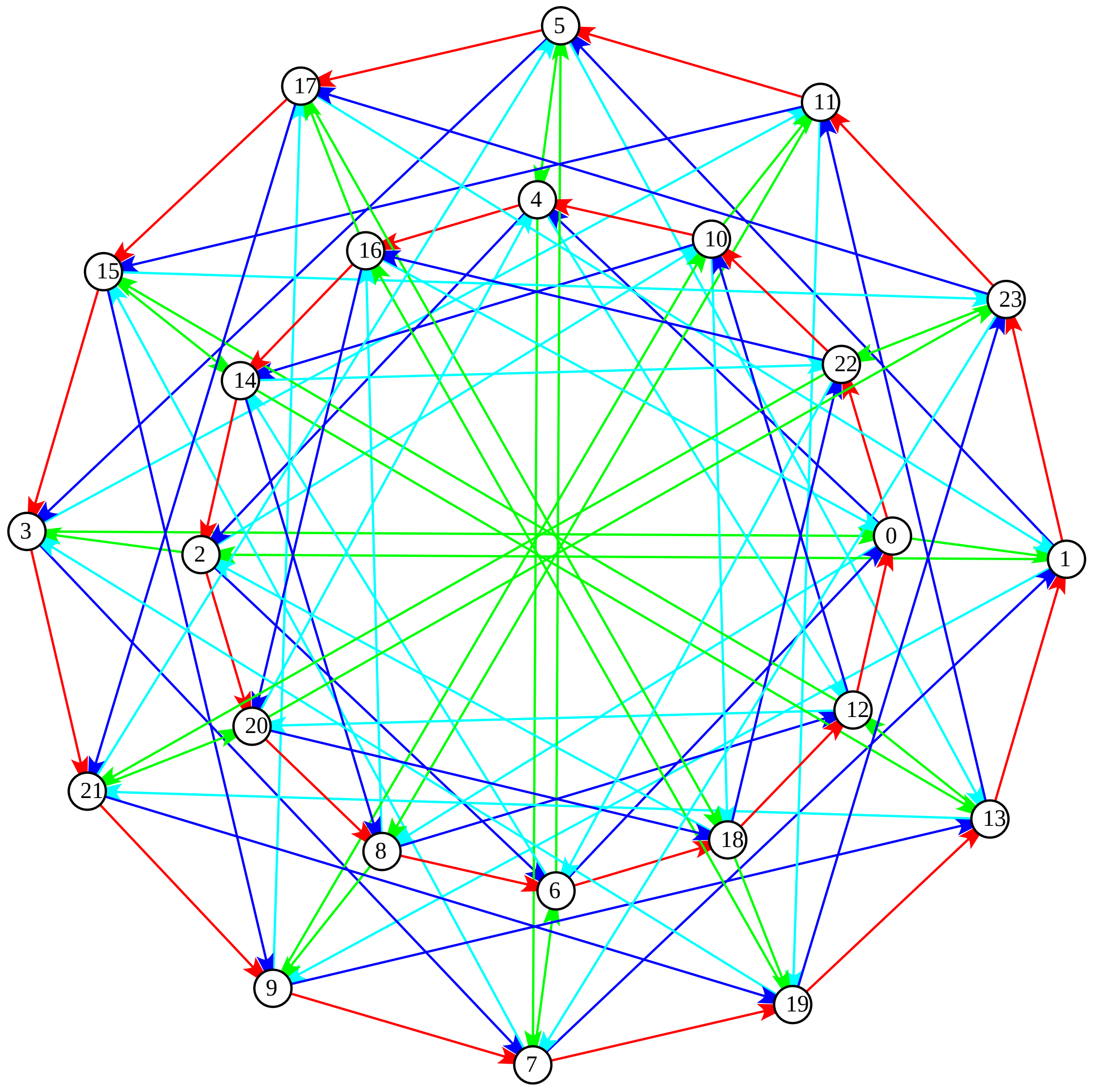
$Q_8 \times Z_2$:


30
18
29
31
17
19
14
13
15
2
3
1
28
12
4
0
16
8
24
20
5
7
6
11
9
10
23
21
27
22
25
26

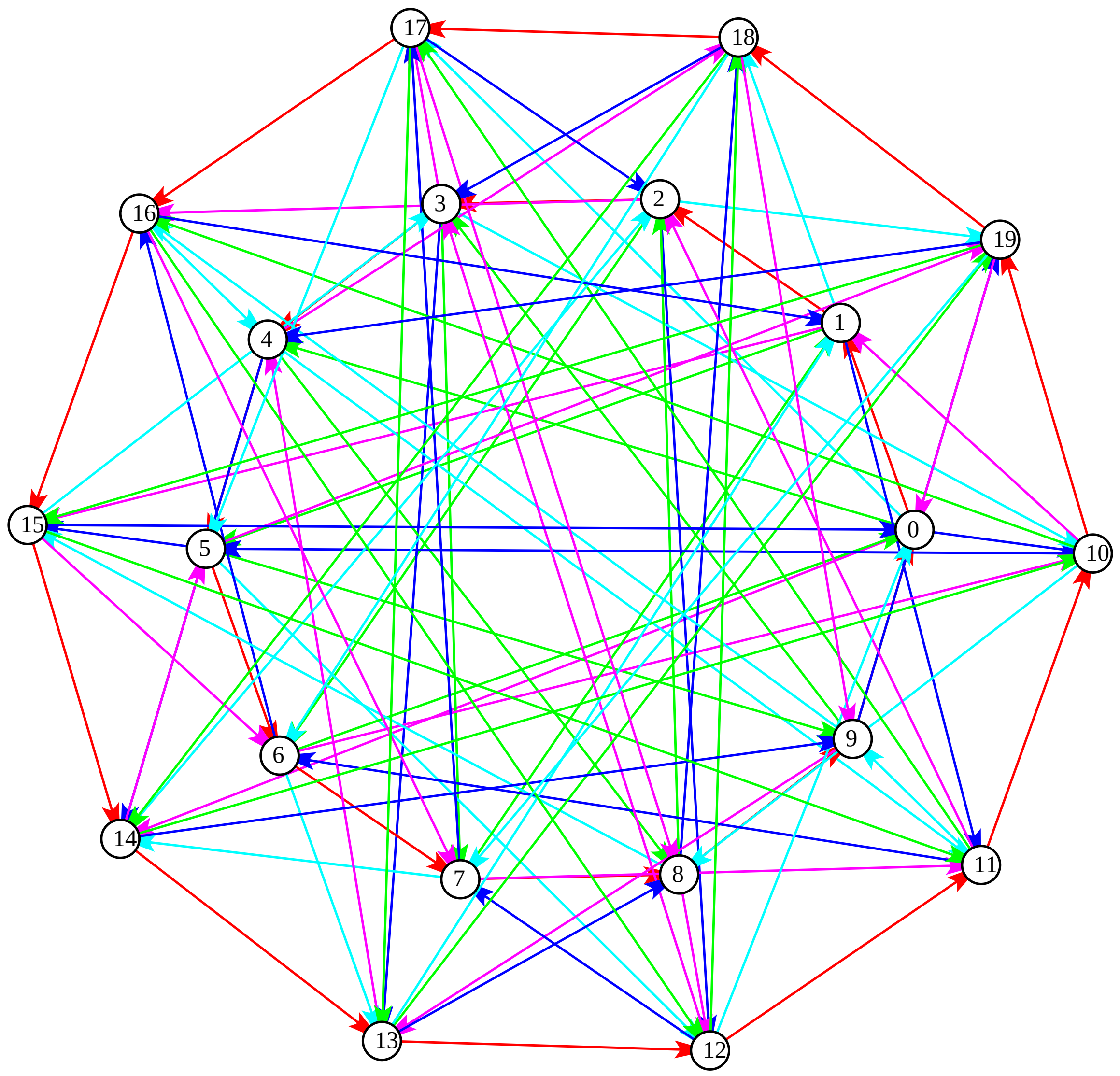
S_4:



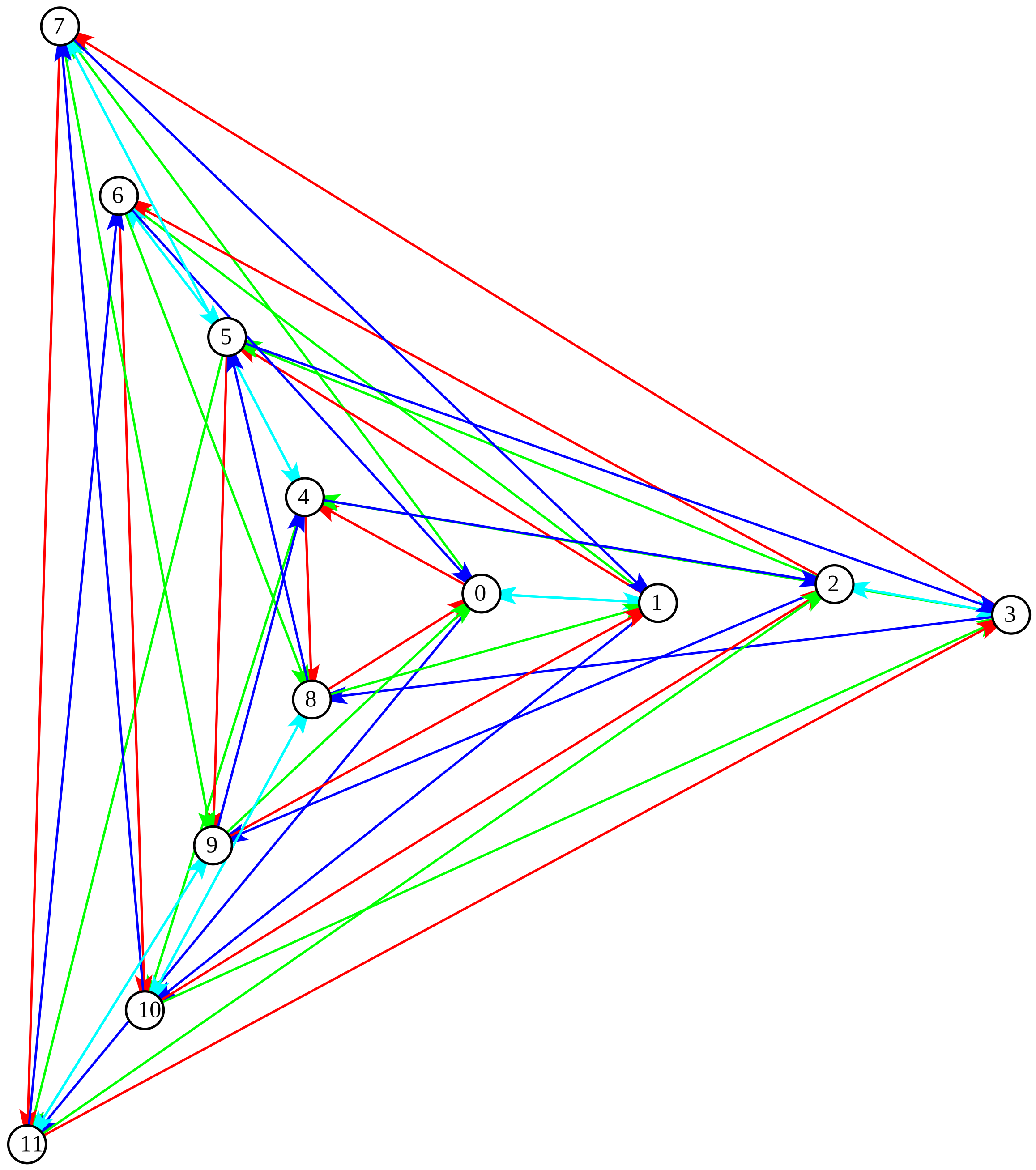
$Q \times Z_3$:



Q_{10}:



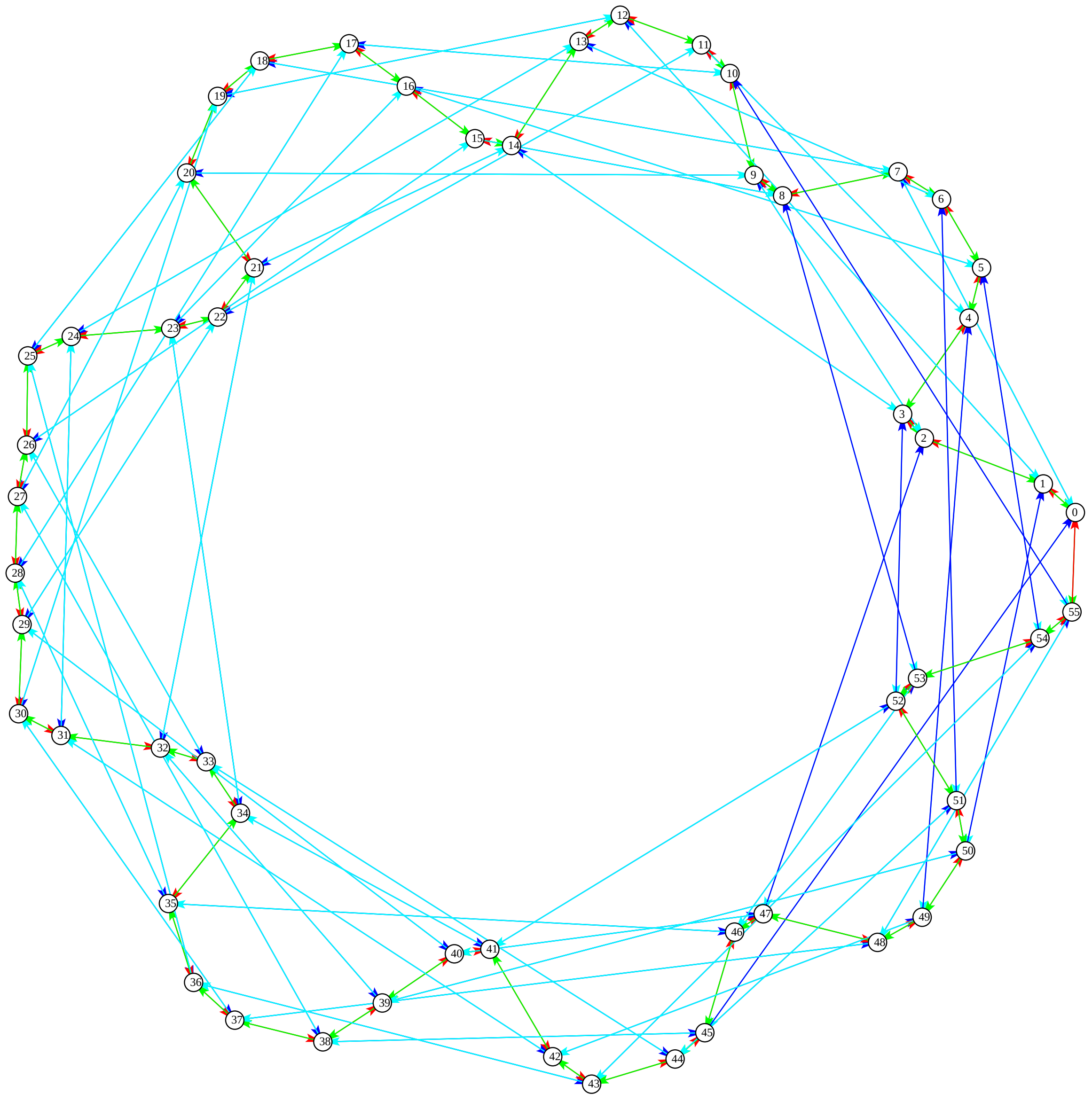
A_4:



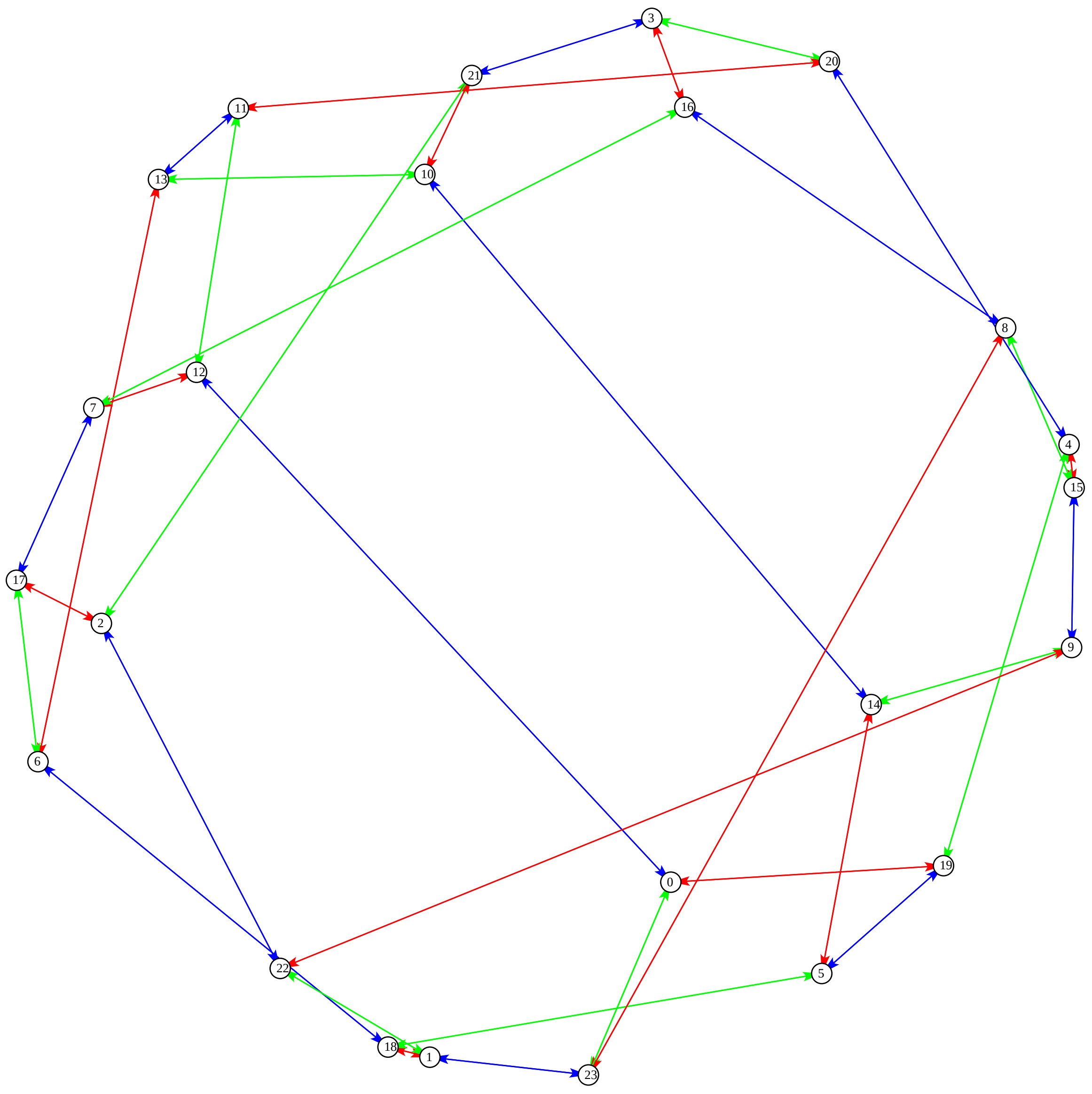
B.4 Force directed with nested base

$Z_4 \times_\sigma Z_{29}$: (too cluttered to display)

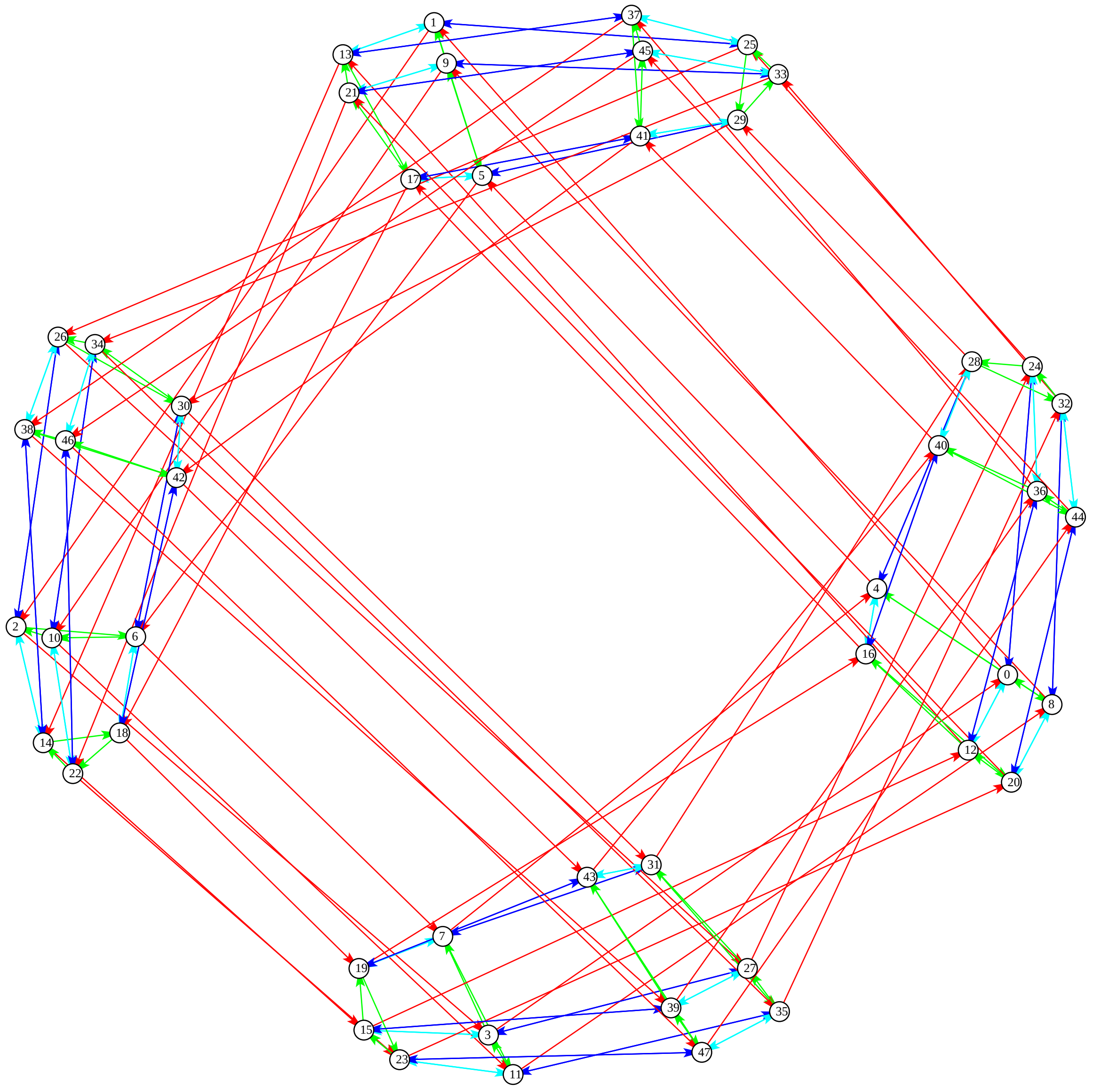
D_{28}:



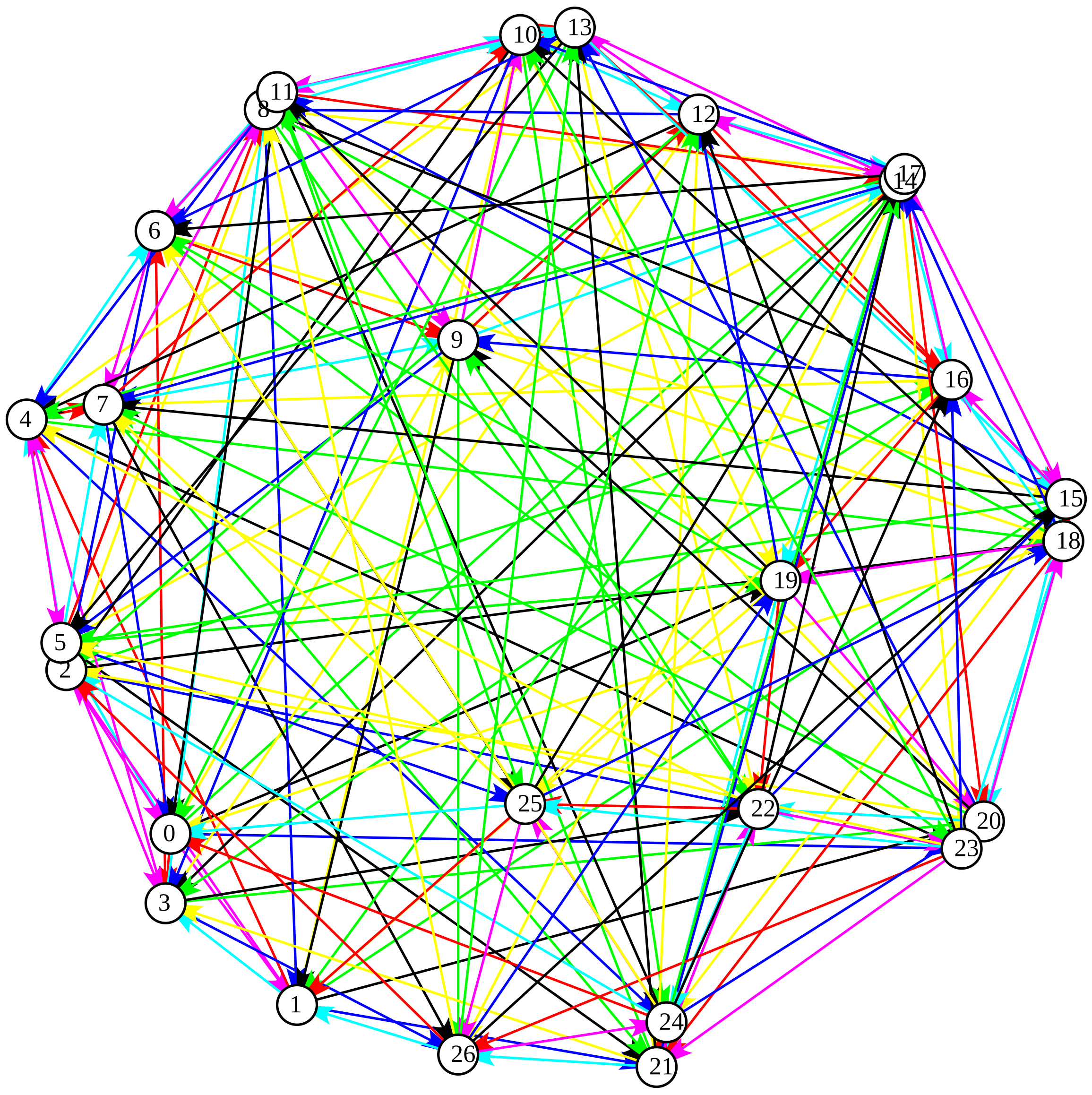
$Z_2 \times_\sigma Z_{12}$:



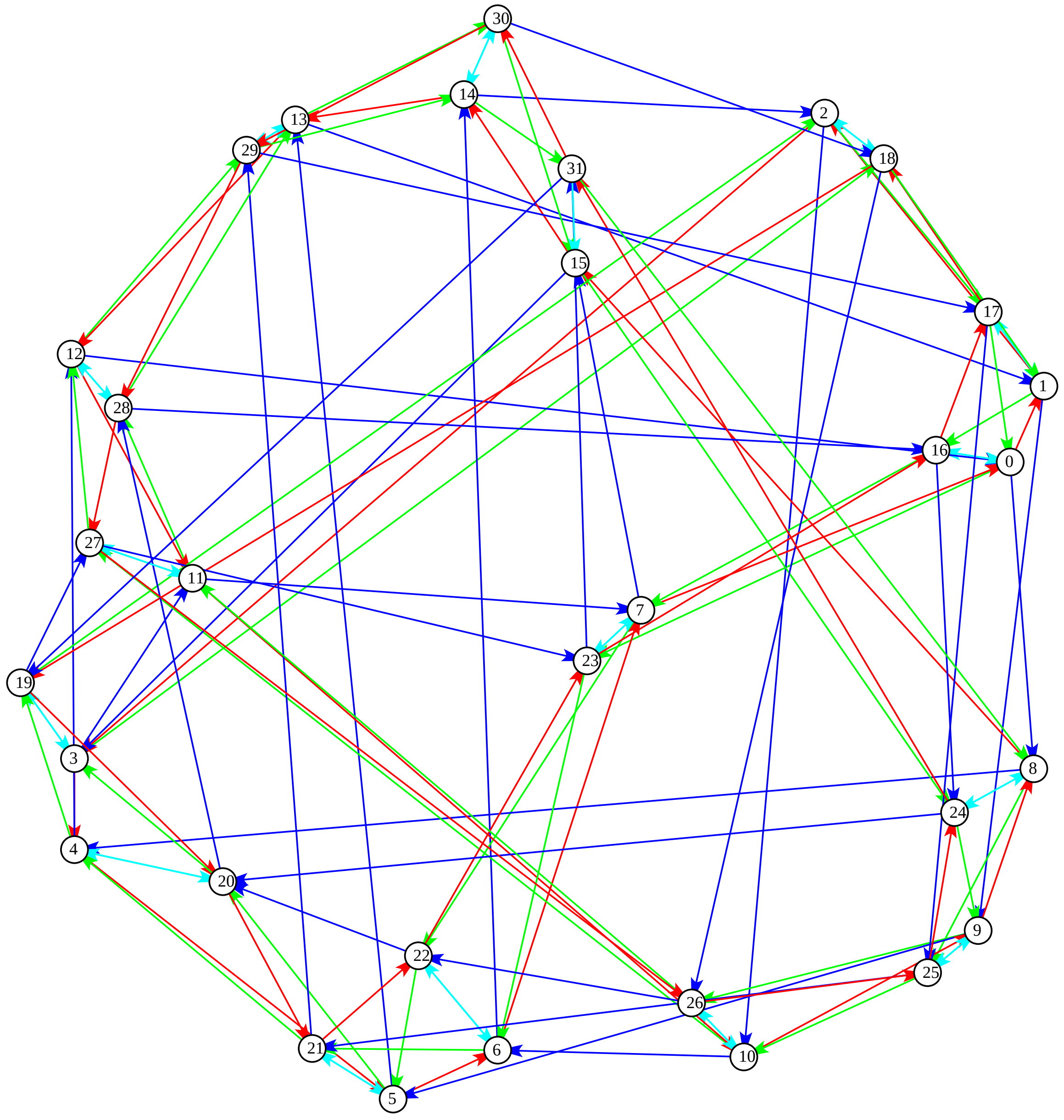
$Z_2 \times Z_2 \times Z_3 \times Z_4$:



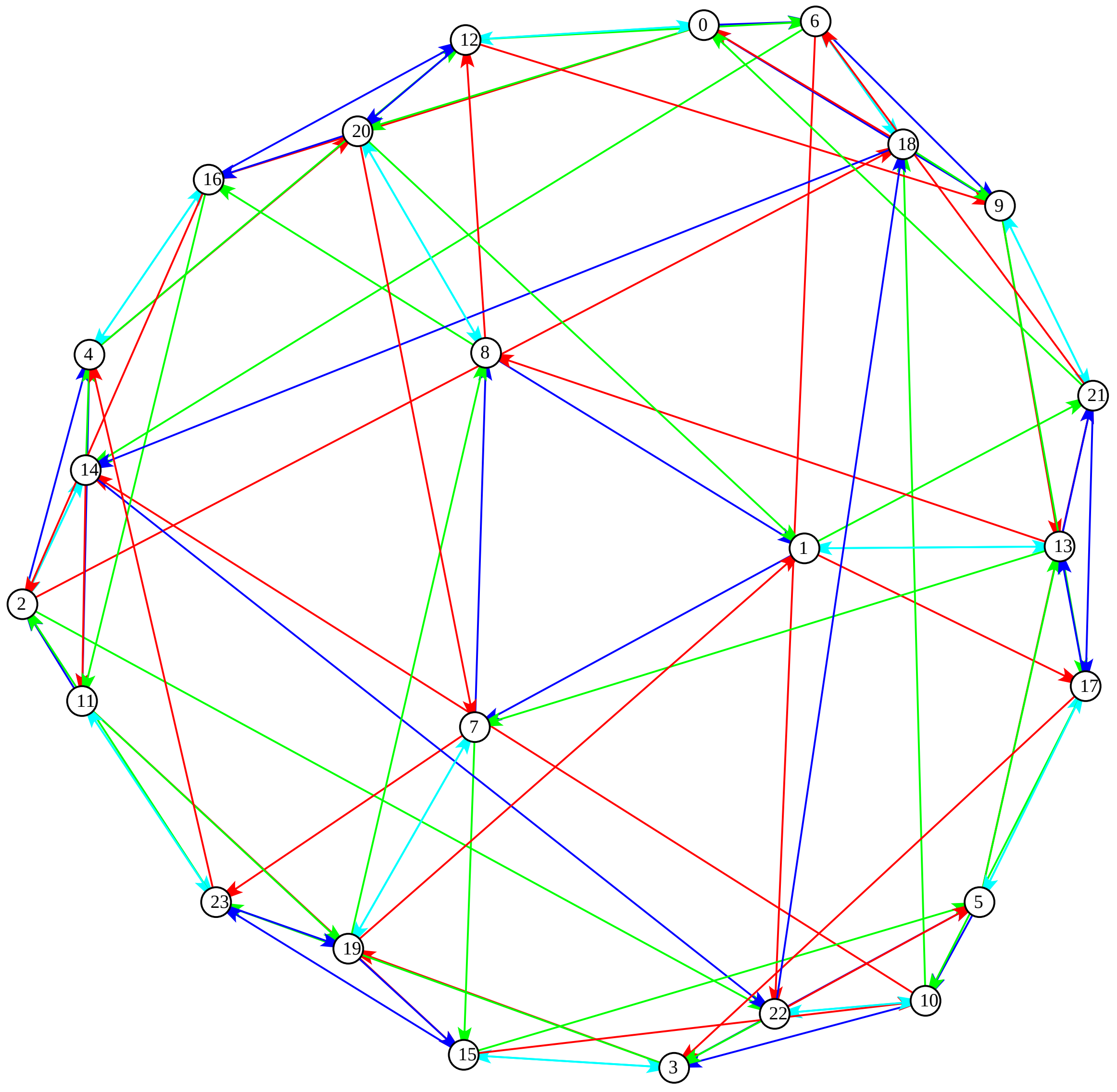
$Z_9 \times Z_3$:



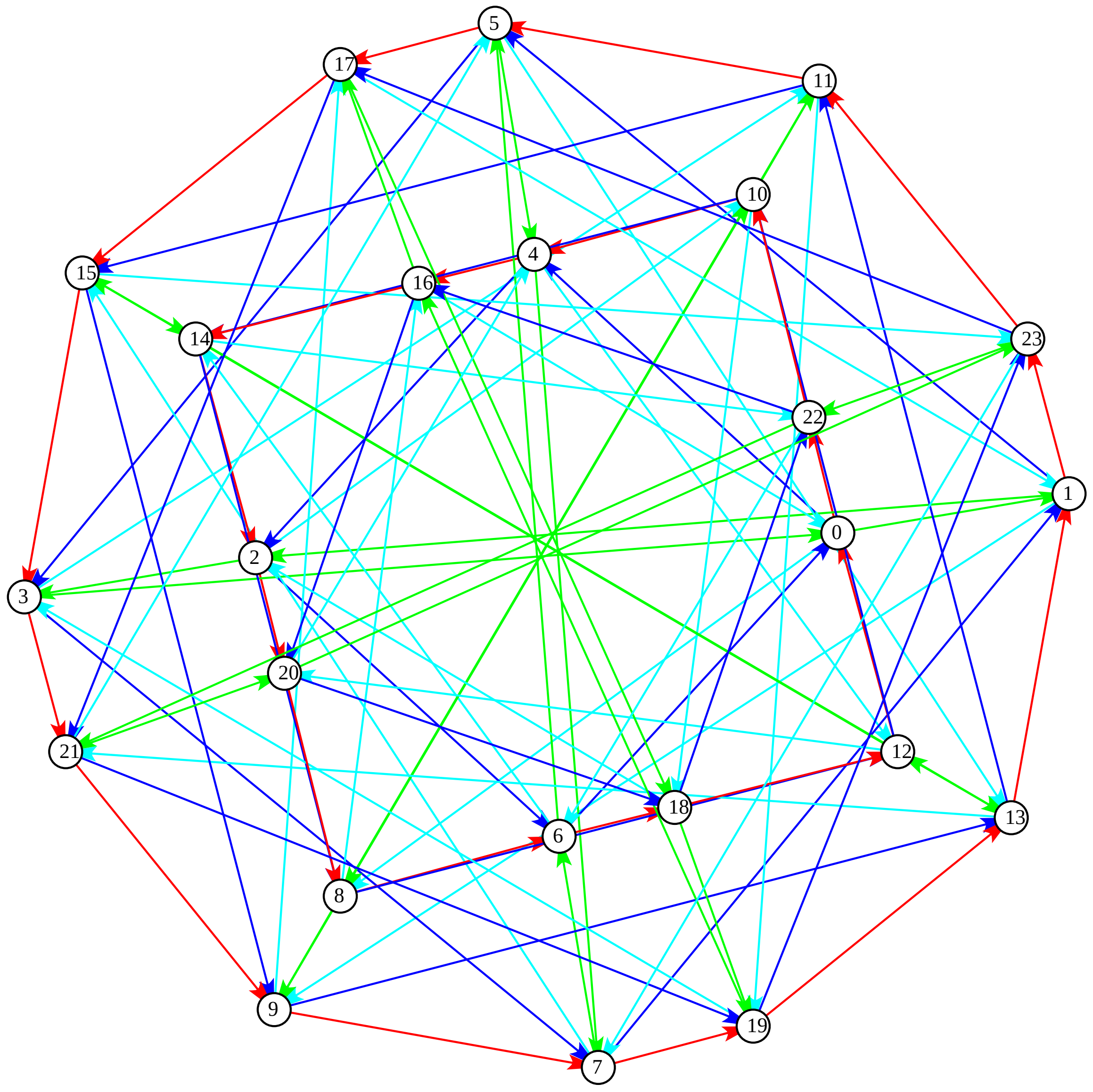
$Q_8 \times Z_2$:


30
14
2
13
29
18
31
15
17
12
1
28
16
0
27
11
7
23
19
3
8
24
4
20
9
22
25
26
10
21
6
5

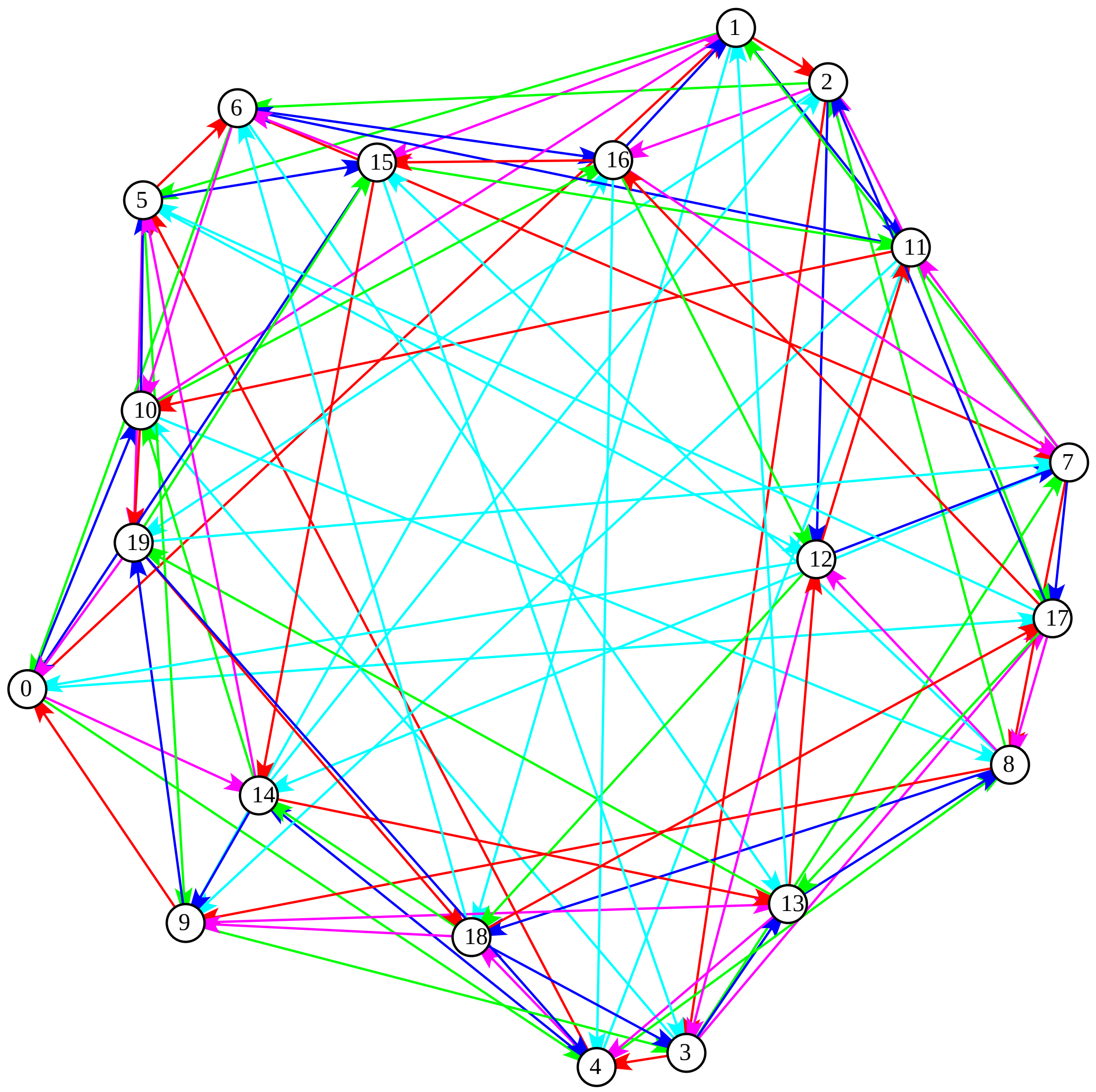
S_4:



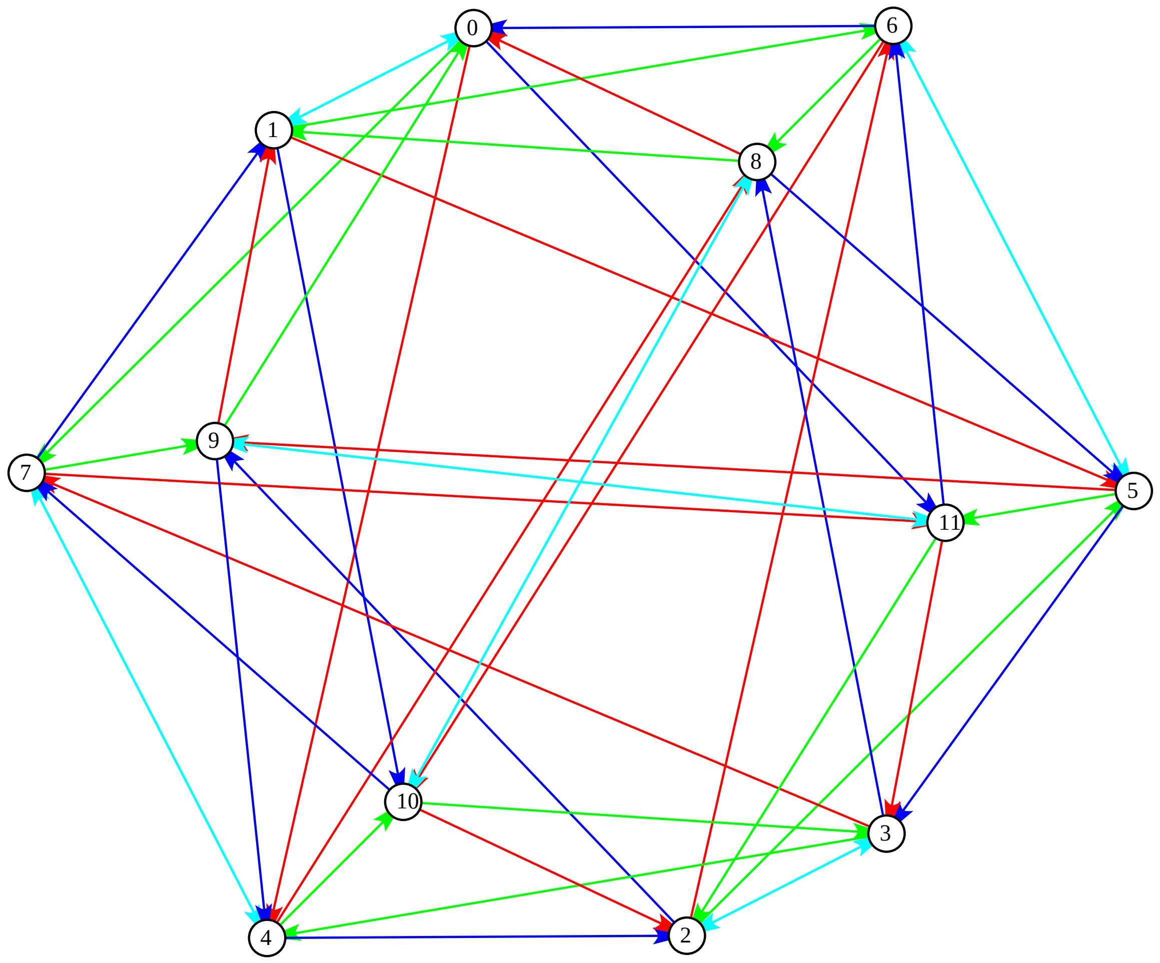
$Q \times Z_3$:



Q_{10}:



A_4:



References

[1] R. Fleischer and C. Hirsch, “Graph Drawing and its Applications,” in *Drawing Graphs – Methods and Models*, vol. 2025, *Lecture Notes in Computer Science*, M. Kaufmann and D. Wagner, Eds. New York: Springer, pp. 1-22.

[2] M. Jünger and P. Mutzel, *Graph Drawing Software.* Berlin: Springer-Verlag, 2004.

[3] P. Eades and X. Lin, “Spring algorithm and symmetry,” *Theoretical Computer Science*, vol. 240(2), pp. 379-405, 2000.

[4] J.A. Gallian, *Contemporary Abstract Algebra, Fifth Edition.* Boston: Houghton Mifflin Company, 2002.

[5] M.-C Heydemann, “Cayley graphs and interconnection networks,” in *Graph Symmetry – Algebraic Methods and Applications,* vol 497, *Mathematical and Physical Sciences*, G.Hahn and G. Sabidussi, Eds. Dordrecht: Kluwer Academic Publishers, 1996, pp. 167-224.

[6] W.H. Campbell, “Indexing Permutations,” *Journal of Computer Sciences in Colleges*, vol. 19(3), pp. 296-300, January 2004.

[7] D. Renault, “Enumerating Planar Locally Finite Cayley Graphs,” *GeometriaeDedicata*, vol. 112(1), pp. 25-49, April 2005.

[8] T. Kamada and S. Kawai, “Automatic display of network structure for human understanding,” *Information Processing Letters*, vol. 31, pp. 7-15, 1989.

[9] M. J. Dinneen and P. R. Hafner, “New results for the degree/diameter problem,” *Networks*, vol. 24, pp. 359-367, October 1994.

[10] A. T. White, *Graphs of Groups on Surfaces – Interactions and Models.* Amsterdam: Elsevier Science B.V, 2001.

[11] A. D. Thomas and G. V. Wood, *Group Tables.* Kent: Shiv A Publishing Limited, 1980.