Supplementary information for "Classification in biological networks with hypergraphlet kernels"

Jose Lugo-Martinez
Computational Biology Department, Carnegie Mellon University
Pittsburgh, Pennsylvania, U.S.A.
jlugomar@cs.cmu.edu

Daniel Zeiberg
Khoury College of Computer Sciences, Northeastern University
Boston, Massachusetts, U.S.A.
zeiberg.d@northeastern.edu

Thomas Gaudelet
Department of Computer Science, University College London
London WC1E 6BT, U.K.
thomas.gaudelet.16@cs.ucl.ac.uk

Noël Malod-Dognin
Department of Life Sciences, Barcelona Supercomputing Center
Barcelona, 08034 Spain
noel.malod@bsc.es

Nataša Pržulj
Department of Life Sciences, Barcelona Supercomputing Center
Barcelona, 08034 Spain
ICREA, Pg. Lluis Companys 23
08010 Barcelona, Spain
natasa@cs.ucl.ac.uk

Predrag Radivojac*
Khoury College of Computer Sciences, Northeastern University
Boston, Massachusetts, U.S.A.
predrag@northeastern.edu

*To whom correspondence should be addressed.

1 Enumeration of labeled hypergraphlets

Here we characterize the feature space of fully labeled hypergraphlets by describing the dimensionality of count vectors $\phi_{(n,\tau)}(v)$. We are interested in the order of growth of $\kappa(n, \Sigma, \Xi)$ as a function of n, Σ and Ξ, as introduced in the main text.

Suppose that G and H are base hypergraphlets with n vertices and m hyperedges. We say that G and H belong to the same equivalence class if and only if the total number of (non-isomorphic) fully labeled hypergraphlets corresponding to the base cases G and H are equal for any Σ and Ξ. The total counts of labeled hypergraphlets over all alphabet sizes induce a partition of base hypergraphlets into equivalence classes. We

Table S1: **Equivalence classes over vertex- and hyperedge-labeled hypergraphlets.** List of equivalence classes and their cardinality produced by partitioning the set of undirected base hypergraphlets for $n \in \{1, 2, 3, 4\}$ over vertex-labels alphabet Σ and hyperedge-labels alphabet Ξ. We also list the number of vertex-labeled n-hypergraphlets over an alphabet Σ denoted as $m_i(n, \Sigma, 1)$, as well as fully labeled n-hypergraphlets over an alphabet Σ and Ξ denoted as $m_i(n, \Sigma, \Xi)$.

Vertex-labeled hypergraphlets		
$S_i(n)$	$\lvert S_i(n)\rvert$	$m_i(n, \Sigma, 1)$
$S_1(1)$	1	$\lvert\Sigma\rvert$
$S_1(2)$	1	$\lvert\Sigma\rvert^2$
$S_1(3)$	3	$\lvert\Sigma\rvert^3$
$S_2(3)$	6	$1/2(\lvert\Sigma\rvert^3 + \lvert\Sigma\rvert^2)$
$S_1(4)$	221	$\lvert\Sigma\rvert^4$
$S_2(4)$	212	$1/2(\lvert\Sigma\rvert^4 + \lvert\Sigma\rvert^3)$
$S_3(4)$	28	$1/6(\lvert\Sigma\rvert^4 + 3 \cdot \lvert\Sigma\rvert^3 + 2 \cdot \lvert\Sigma\rvert^2)$
Fully-labeled hypergraphlets		
$S_i(n)$	$\lvert S_i(n)\rvert$	$m_i(n, \Sigma, \Xi)$
$S_1(1)$	1	$\lvert\Sigma\rvert$
$S_1(2)$	1	$\lvert\Sigma\rvert^2 \cdot \lvert\Xi\rvert$
$S_1(3)$	1	$\lvert\Sigma\rvert^3 \cdot \lvert\Xi\rvert^3$
$S_2(3)$	2	$\lvert\Sigma\rvert^3 \cdot \lvert\Xi\rvert^2$
$S_3(3)$	1	$1/2(\lvert\Sigma\rvert^3 \cdot \lvert\Xi\rvert^4 + \lvert\Sigma\rvert^2 \cdot \lvert\Xi\rvert^3)$
$S_4(3)$	2	$1/2(\lvert\Sigma\rvert^3 \cdot \lvert\Xi\rvert^3 + \lvert\Sigma\rvert^2 \cdot \lvert\Xi\rvert^2)$
$S_5(3)$	1	$1/2(\lvert\Sigma\rvert^3 \cdot \lvert\Xi\rvert^2 + \lvert\Sigma\rvert^2 \cdot \lvert\Xi\rvert^2)$
$S_6(3)$	1	$1/2(\lvert\Sigma\rvert^3 \cdot \lvert\Xi\rvert^2 + \lvert\Sigma\rvert^2 \cdot \lvert\Xi\rvert)$
$S_7(3)$	1	$1/2(\lvert\Sigma\rvert^3 \cdot \lvert\Xi\rvert + \lvert\Sigma\rvert^2 \cdot \lvert\Xi\rvert)$

denote the set of all equivalence classes over the hypergraphlets of order n as $S(n) = \{S_1(n), S_2(n), \ldots\}$. For example, the set of vertex- and hyperedge-labeled 3-hypergraphlets can be partitioned into either: two symmetry classes when $|\Xi| = 1$: $S_1(3) = \{3_2, 3_4, 3_7\}$ and $S_2(3) = \{3_1, 3_3, 3_5, 3_6, 3_8, 3_9\}$, or seven symmetry classes when $|\Xi| > 1$: $S_1(3) = \{3_2\}$, $S_2(3) = \{3_4, 3_7\}$, $S_3(3) = \{3_9\}$, $S_4(3) = \{3_6, 3_8\}$, $S_5(3) = \{3_3\}$, $S_6(3) = \{3_5\}$ and $S_7(3) = \{3_1\}$. Table S1 summarizes equivalence classes induced by partitioning base hypergraphlets up to the order of 4 along with the cardinality of each set. Overall, observe that the cardinality of $S(n)$ can be significantly larger than those reported for graphlets [3] because the possible number of hyperedges in a hypergraphlet is generally much larger than the possible number of edges in a graphlet. Additionally, hyperedge-labels require base hypergraphlets G and H to have an equal number of hyperedges.

This approach can be generalized to hypergraphlets labeled by any alphabet Σ and Ξ, such that

$$\kappa(n, \Sigma, \Xi) = \sum_{i=1}^{|S(n)|} m_i(n, \Sigma, \Xi) \cdot |S_i(n)|,$$

where $m_i(n, \Sigma, \Xi)$ is the number of (non-isomorphic) fully labeled hypergraphlets corresponding to any base hypergraphlet from the equivalence class $S_i(n)$. We use this decomposition to compute the total dimensionality of the count vectors by first finding the equivalence classes corresponding to the base hypergraphlets and then counting the number of labeled hypergraphlets for any one member of the group.

In the case of undirected fully labeled hypergraphlets, $m_i(n, \Sigma, \Xi)$ can also be computed by applying the theory of enumeration developed by Pólya [5]. In order to get the derivation of the complete generating function for each equivalence class $S_i(n)$, we first define the automorphism group $\mathcal{A}$ of a given vertex- and hyperedge-labeled hypergraph $G = (V, E)$. That is, in the case of fully-labeled hypergraphs, set $\mathcal{A}$ is a collection of permutations (automorphisms) of V and E. Therefore, the counting problem can be re-formulated as follows: Let G be a base hypergraphlet of n vertices and m hyperedges, and $\mathcal{A}$ be the automorphism group of G over V and E. Then, each permutation $\alpha \in \mathcal{A}$ can be written uniquely as the

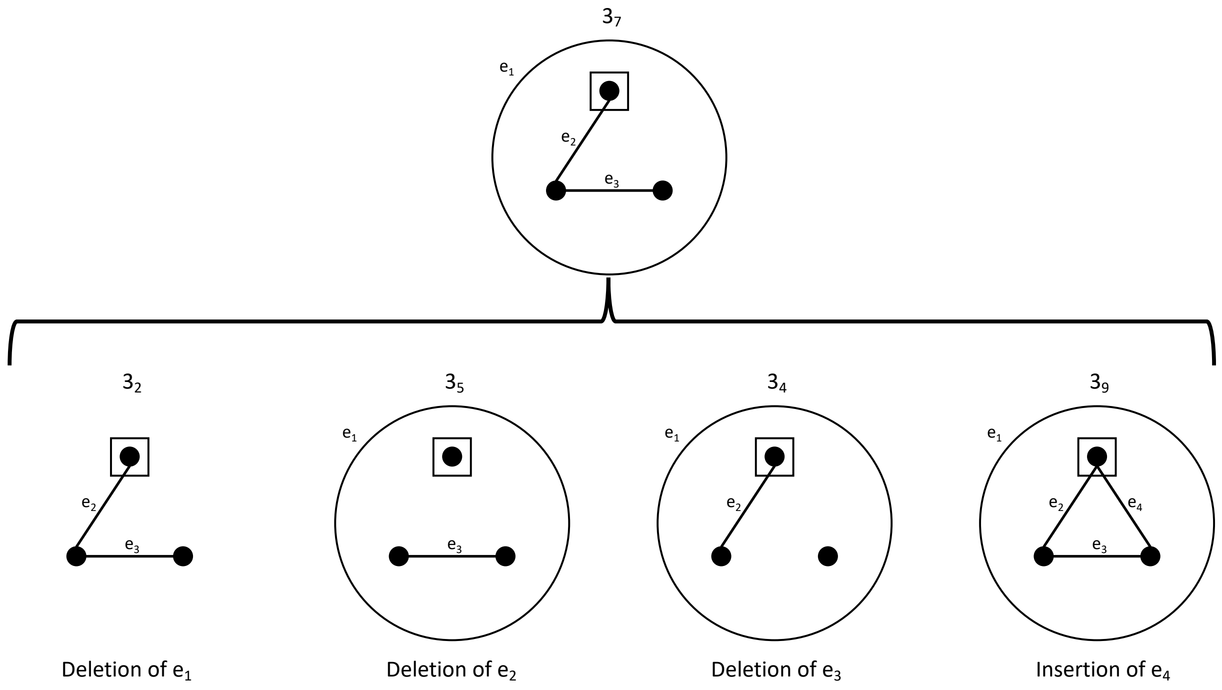


Figure S1: Example of precomputed level-1 insertions/deletions (indels) of hyperedges for base hypergraphlet 3_7. The precomputed level-1 indels include three hyperedge deletions and one hyperedge insertion.

product of disjoint cycles such that for each integer $k \in \{1, \ldots, n\}$ ($k' \in \{1, \ldots, m\}$), we define $j_k(\alpha)$ ($j_{k'}(\alpha)$) as the number of cycles of length k (k') in the disjoint cycle expansion of α. Interestingly, the generalized formula for the cycle index of $\mathcal{A}$, denoted as $Z(\mathcal{A})$, is a polynomial in $s_1, \ldots, s_n; s'_1, \ldots, s'_m$ given by

$$Z(\mathcal{A}; s_1, \ldots, s_n; s'_1, \ldots, s'_m) = \frac{1}{|\mathcal{A}|} \sum_{\alpha \in \mathcal{A}} \prod_{k=1}^{n} \prod_{k'=1}^{m} s_k^{j_k(\alpha)} \cdot s_{k'}^{j_{k'}(\alpha)}.$$

By applying Pólya's theorem in the context of enumerating vertex- and hyperedge-labeled hypergraphlets corresponding to any base hypergraphlet in $S_i(n)$, we get that $m_i(n, \Sigma, \Xi)$ is determined by substituting $|\Sigma|$ for each variable s_k and $|\Xi|$ for each variable $s'_{k'}$ in $Z(\mathcal{A})$. Hence,

$$m_i(n, \Sigma, \Xi) = Z(\mathcal{A}; |\Sigma|, |\Sigma|, \ldots, |\Sigma|; |\Xi|, |\Xi|, \ldots, |\Xi|),$$

where $\mathcal{A}$ is the automorphism group of a base hypergraphlet from $S_i(n)$. As an example, consider the equivalence class $S_3(3) = \{3_9\}$ with $\Sigma = \{A, B, C\}$ and $\Xi = \{X, Y\}$ (Figure 2 in the main paper illustrates an unlabeled version of hypergraphlet 3_9). The automorphism group is given by

$$\mathcal{A} = \{(v_1)(v_2)(v_3)(e_1)(e_2)(e_3)(e_4), (v_1)(v_2 v_3)(e_1 e_2)(e_3)(e_4)\},$$

thus, $Z(\mathcal{A}; s_1, s_2, s_3; s'_1, s'_2) = \frac{1}{2}(s_1^3 \cdot s'^4_1 + s_1 \cdot s_2 \cdot s'^2_1 \cdot s'_2)$. Therefore, it follows that, $m_3(3, \Sigma, \Xi) = Z(\mathcal{A}; 3, 3, 3; 2, 2) = \frac{1}{2}(3^3 \cdot 2^4 + 3 \cdot 3 \cdot 2^2 \cdot 2) = 252$.

2 Computational complexity

The implementation and the analysis of hypergraphlet kernels is an extension of the available solutions for string kernels [6]. Let $G_l(r) = (V_r, E_r)$ be a *neighborhood hypergraph* of a rooted hypergraph $G = (V, r, E)$ such that there exits a walk w of length at most l between any node $u \in V$ and root vertex r, denoted as $|w(u, r)| \leq l$. Formally, $E_r = \{e \,|\, e \in E \wedge e \subseteq V_r \wedge |w(u, r)| \leq l, \forall u \in V_r, l \geq 0, r \in V_r\}$. In this work, we consider the set of undirected base hypergraphlets for $n \in \{1, 2, 3, 4\}$, thus, neighborhood hypergraph $G_{n-1}(r)$ contains all possible n-hypergraphlets rooted at vertex r. Suppose $G_{n-1}(r)$ is significantly smaller than G. An upper bound on the complexity of the counting algorithm for all n-hypergraphlets rooted at vertex r is $\mathcal{O}(|V_r||E_r| + (\delta_{\max} d_{\max})^{n-1})$, where $d_{\max}$ is the maximum degree of a vertex in V_r and $\delta_{\max}$ is the maximum degree of a hyperedge in E_r. The first term is related to computing the neighbors for each node $v \in V_r$, whereas the second term reflects the cost of counting over all base hypergraphlets. Similarly, an upper bound on the complexity of generating the minimum cost edit path is $\mathcal{O}((n|\Sigma| + 2^n|\Xi|) + (2^n|\Xi|))$ per single hypergraphlet edit operation. In this case, the first term reflects the cost of generating all vertex and hyperedge label substitutions, whereas the second term corresponds to the cost of performing hyperedge insertions and deletions (indels); see Figure S1 for an example of level-1 indels on a base hypergraphlet. Therefore, for each vertex v an order of

$$\mathcal{O}(\min\{|V_r|^n, \kappa(n, \Sigma, \Xi)\}\,((n|\Sigma| + 2^n|\Xi|) + (2^n|\Xi|))^\tau)$$

operations are necessary, where the $|V_r|^n$ term enumerates possible n-hypergraphlets in $G_{n-1}(r)$. Note that the possible number of hyperedges in a hypergraph $|E_r|$ can be significantly larger than the possible number of edges in a standard graph. Hence, in a practical setting, the edit-distance hypergraphlet kernels could greatly benefit from effective sampling techniques or exploitation of special types of hypergraphlets. The proposed implementation for computing hypergraphlet kernel functions is computed in time linear in the number of non-zero elements.

2.1 Converting hypergraphs to graphs

A commonly used approach for learning from hypergraph data is to construct a standard graph representation from the input hypergraph and use well-established graph approaches. Among these hypergraph-to-graph transformations, the two most popular are *clique expansion* and *star expansion* [7]. A clique expansion of a hypergraph $G = (V, E)$ is defined as graph $G^x = (V, E^x \subseteq V^2)$ such that each hyperedge $e = \{v_1, \cdots, v_{\delta(e)}\} \in E$ is replaced with an edge for each pair of vertices in hyperedge e, thus, forming a clique in G^x; mathematically, $E^x = \{(u, v)\,|\,u, v \in e, e \in E\}$. A star expansion of a hypergraph $G = (V, E)$ is defined as a graph $G^* = (V^*, E^*)$ such that for each hyperedge $e = \{v_1, \cdots, v_{\delta(e)}\} \in E$, a new vertex e is introduced in G^* connecting e to each $v \in e$, hence, $V^* = V \cup E$; mathematically, $E^* = \{(v, e)\,|\,v \in e, e \in E\}$.

We used the clique expansion to evaluate the benefits of using hypregraphs in the second stage of our approach. Note that the first stage consists of converting graphs into dual extended hypergraphs and formulating edge classification and link prediction as a vertex classification approach on hypergraphs. A combination of the dual hypergraph construction, subsequent clique expansion, with graphlet kernel-based prediction is referred to here as *dual graphlet kernel* to distinguish this method from standard graphlet kernels used on the vertices of the original network (note that edge classification on standard graphs corresponds here to the vertex classification on line graphs). The dual graphlet kernels that here integrate vertex and edge alphabets into edit-distance similarity functions were not available in prior work and were implemented within our proposed methodology.

References

[1] A.L. Barabási et al. Evolution of the social network of scientific collaborations. *Physica A*, 311(3):590–614, 2002.

[2] I.A. Kovács et al. Network-based prediction of protein interactions. *Nat Commun*, 10:1240–1247, 2019.

[3] J. Lugo-Martinez and P. Radivojac. Generalized graphlet kernels for probabilistic inference in sparse graphs. *Network Science*, 2(2):254–276, 2014.

[4] Y. Park and E. M. Marcotte. Flaws in evaluation schemes for pair-input computational predictions. *Nat Methods*, 9(12):1134–1136, 2012.

[5] G. Pólya. Kombinatorische anzahlbestimmungen für gruppen, graphen und chemische verbindungen. *Acta Math*, 68:145–254, 1937.

[6] K. Rieck and P. Laskov. Linear-time computation of similarity measures for sequential data. *J Mach Learn Res*, 9:23–48, 2008.

[7] J. Y. Zien, M. D. F. Schlag, and P. K. Chan. Multilevel spectral hypergraph partitioning with arbitrary vertex sizes. *IEEE Trans Comput Aid Design*, 18(9):1389–1399, 1999.

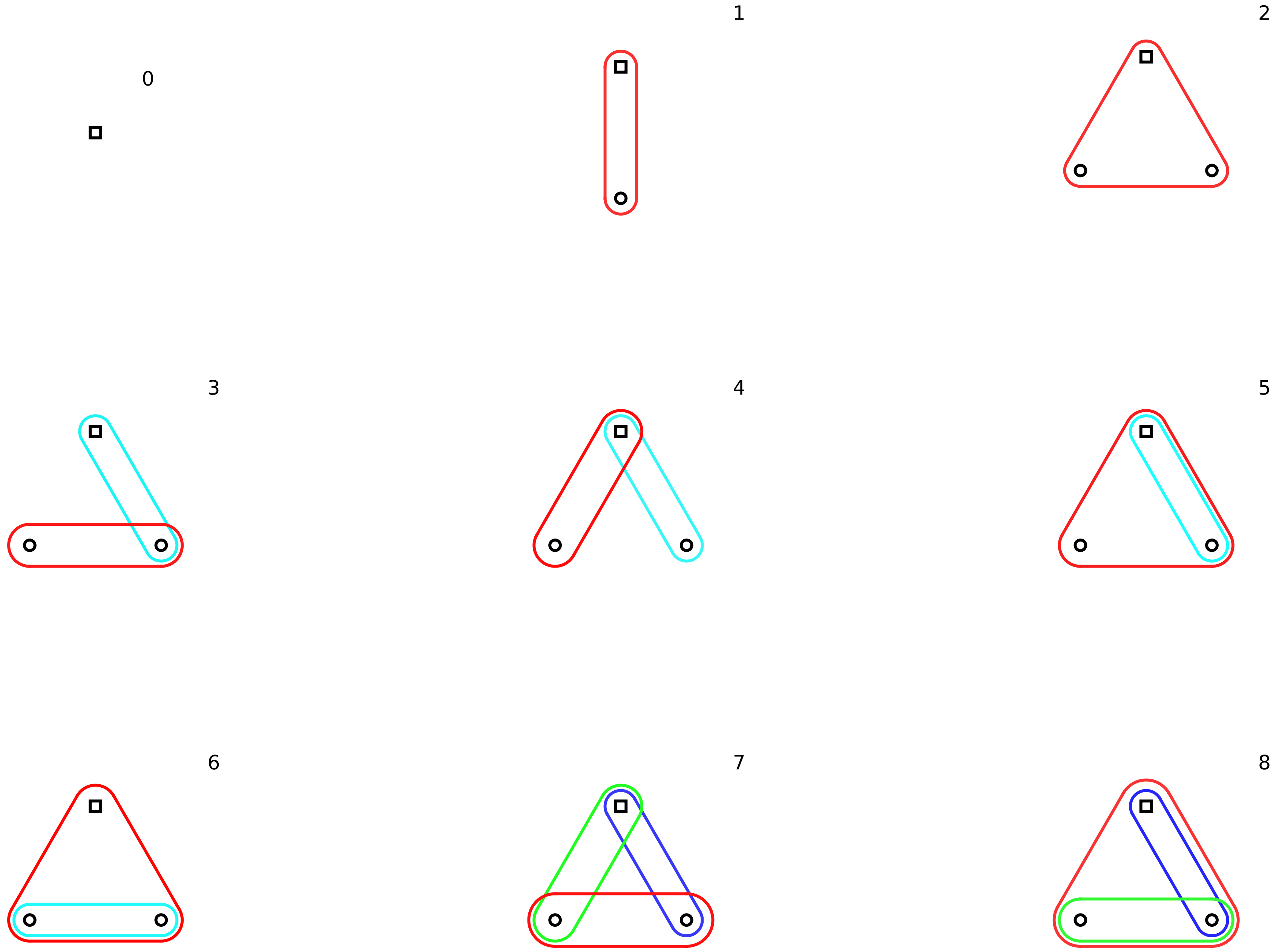
Table S2: **Area under the ROC curve estimates for each method on the PPI data sets using 10-fold cross-validation based on the three categories C_1, C_2, C_3 of difficulty identified by Park and Marcotte [4].** The highest performance for each class and data set pair is shown in boldface. Since L3 framework relies on paths of a specific length, it cannot predict on all possible protein pairs. Therefore, the last two rows compare the AUC between L3 and the best hypergraphlet kernel on the same set of predicted protein pairs.

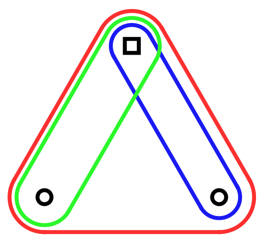
Method/Dataset	**EC**			**SP**			**RN**			**MM**			**CE**			**AT**		
	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3
Without domain information, $\lvert\Sigma\rvert = 1$																		
Dual graphlet kernel ($\tau = 0$)	0.715	0.705	0.674	0.719	0.718	**0.712**	0.693	0.691	0.651	0.637	0.635	0.625	**0.873**	**0.872**	**0.872**	0.681	0.683	0.681
Section hypergraphlet kernel ($\tau = 0$)	0.724	0.713	0.719	0.639	0.646	0.628	0.671	0.672	0.664	0.628	0.630	0.633	0.762	0.755	0.774	0.578	0.563	0.588
Sub-hypergraphlet kernel ($\tau = 0$)	0.692	0.687	0.657	0.691	0.691	0.686	0.651	0.647	0.633	0.612	0.610	0.605	**0.873**	**0.872**	**0.872**	0.681	0.683	0.681
Dual graphlet kernel ($\tau = 1$)	0.678	0.669	0.649	0.698	0.699	0.691	0.671	0.670	0.648	0.623	0.620	0.614	0.869	0.869	0.869	0.672	0.671	0.670
Section hypergraphlet kernel ($\tau = 1$)	0.708	0.695	0.700	0.641	0.643	0.623	0.678	0.666	0.668	0.628	0.628	0.633	0.772	0.781	0.767	0.581	0.563	0.567
Sub-hypergraphlet kernel ($\tau = 1$)	0.681	0.682	0.660	0.686	0.687	0.682	0.649	0.649	0.634	0.611	0.610	0.606	0.871	0.870	0.870	0.683	0.684	0.682
L3 framework [2]	0.717	0.597	0.556	0.577	0.529	0.497	0.634	0.611	0.510	0.573	0.550	0.506	0.511	0.504	0.504	0.501	0.501	0.502
Preferential attachment [1]	0.814	0.675	0.639	0.492	0.456	0.453	0.552	0.504	0.511	0.498	0.485	0.463	0.660	0.523	0.549	0.449	0.441	0.463
With domain information, $\lvert\Sigma\rvert = \{4, 8, 16\}$; $\Sigma = \Sigma_{GO}$																		
Random walk	0.762	0.755	0.739	0.636	0.617	0.574	0.537	0.521	0.422	0.585	0.563	0.497	0.805	0.803	0.802	0.613	0.605	0.585
Random hyperwalk	0.815	0.817	0.786	0.610	0.599	0.550	0.599	0.587	0.484	0.588	0.567	0.531	0.597	0.595	0.594	0.608	0.601	0.580
Cumulative random walk	0.799	0.791	0.789	0.706	0.701	0.680	0.587	0.585	0.466	0.641	0.641	0.613	0.716	0.701	0.708	0.647	0.643	0.629
Cumulative random hyperwalk	0.848	0.841	0.820	0.681	0.666	0.641	0.647	0.616	0.524	0.670	0.669	0.646	0.569	0.552	0.561	0.637	0.614	0.624
Pairwise spectrum kernel ($k = \{3, 4, 5\}$)	0.816	0.754	0.618	0.584	0.505	0.432	0.716	0.648	0.580	0.689	0.632	0.548	0.754	0.659	0.650	0.629	0.616	0.541
Dual graphlet kernel ($\tau = 0$)	0.802	0.763	0.733	0.734	0.704	0.609	0.673	0.665	0.528	0.630	0.663	0.595	0.846	0.848	0.843	0.678	0.671	0.619
Section hypergraphlet kernel ($\tau = 0$)	0.861	0.840	0.815	0.677	0.664	0.634	0.698	0.670	0.637	0.655	0.647	0.641	0.804	0.799	0.798	0.559	0.567	0.535
Sub-hypergraphlet kernel ($\tau = 0$)	0.796	0.771	0.747	0.740	0.720	0.665	0.676	0.637	0.500	0.708	0.692	0.641	0.846	0.848	0.843	0.678	0.671	0.619
Dual graphlet kernel ($\tau = 1$)	0.816	0.776	0.742	0.745	0.719	0.630	0.682	0.691	0.559	0.702	0.673	0.611	0.853	0.854	0.851	0.699	0.687	0.633
Section hypergraphlet kernel ($\tau = 1$)	**0.875**	**0.843**	**0.830**	0.697	0.684	0.649	0.706	0.695	**0.686**	0.667	0.648	0.639	0.786	0.784	0.783	0.574	0.576	0.574
Sub-hypergraphlet kernel ($\tau = 1$)	0.812	0.776	0.772	**0.750**	**0.738**	**0.712**	**0.719**	**0.698**	0.563	**0.736**	**0.726**	**0.695**	0.860	0.860	0.861	**0.728**	**0.726**	**0.711**
L3 framework [2]	0.588	0.449	0.691	0.638	0.635	0.565	0.751	0.755	0.613	0.622	0.616	0.531	0.130	0.100	-	0.677	0.549	0.659
Hypergraphlet kernel (highest)	0.896	0.687	0.833	0.759	0.728	0.712	0.683	0.696	0.632	0.738	0.726	0.696	0.966	0.714	-	0.652	0.843	0.846

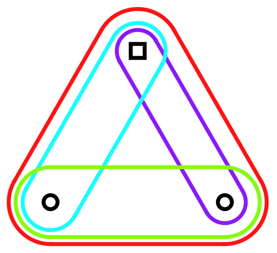
Table S3: **Area under the ROC curve estimates for each method on the DTI data sets using 10-fold cross-validation based on the three categories C_1, C_2, C_3 of difficulty identified by Park and Marcotte [4].** The highest performance for each class and data set pair is shown in boldface. Since L3 framework relies on paths of a specific length, it cannot predict on all possible drug-target pairs. Therefore, the last two rows compare the AUC between L3 and the best hypergraphlet kernel on the same set of predicted drug-target pairs.

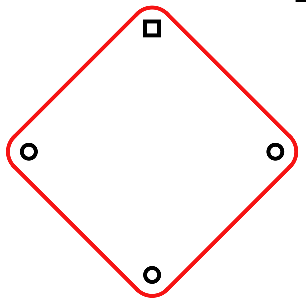
Method/Dataset	**EZ**			**IC**			**GR**			**NR**		
	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3
Without domain information, $\|\Sigma\| = 1$												
Dual graphlet kernel ($\tau = 0$)	0.871	0.865	0.838	0.759	0.729	0.714	0.624	0.606	0.525	0.714	0.676	0.712
Section hypergraphlet kernel ($\tau = 0$)	0.537	0.541	0.514	0.542	0.506	0.509	0.552	0.542	0.493	0.783	**0.772**	0.679
Sub-hypergraphlet kernel ($\tau = 0$)	0.788	0.786	0.778	0.711	0.697	0.680	0.579	0.566	0.484	0.732	0.719	**0.735**
Dual graphlet kernel ($\tau = 1$)	0.851	0.836	0.792	0.713	0.689	0.651	0.599	0.580	0.519	0.734	0.725	0.728
Section hypergraphlet kernel ($\tau = 1$)	0.535	0.541	0.502	0.550	0.510	0.524	0.542	0.504	0.513	**0.791**	0.736	0.704
Sub-hypergraphlet kernel ($\tau = 1$)	0.784	0.780	0.770	0.702	0.684	0.671	0.580	0.566	0.482	0.736	0.733	0.731
L3 framework [2]	0.629	0.626	0.500	0.642	0.534	0.500	0.596	0.584	0.500	0.505	0.420	0.500
Preferential attachment [1]	0.534	0.502	0.500	0.561	0.457	0.500	0.545	0.530	0.500	0.497	0.431	0.500
With domain information, $\|\Sigma\| = \{4, 8, 16\}$; $\Sigma = \Sigma_{GO} \bigcup \Sigma_{SS}$												
Dual graphlet kernel ($\tau = 0$)	0.952	0.914	0.830	0.883	0.830	0.659	0.756	0.679	0.534	0.771	0.709	0.694
Section hypergraphlet kernel ($\tau = 0$)	0.633	0.630	0.596	0.583	0.546	0.527	0.625	0.581	0.518	0.775	0.605	0.638
Sub-hypergraphlet kernel ($\tau = 0$)	0.936	0.885	0.801	0.879	0.834	0.678	0.724	0.641	0.531	0.757	0.717	0.709
Dual graphlet kernel ($\tau = 1$)	**0.953**	**0.923**	0.832	**0.900**	0.845	0.686	**0.758**	0.684	0.533	0.778	0.719	0.719
Section hypergraphlet kernel ($\tau = 1$)	0.637	0.605	0.605	0.585	0.543	0.523	0.619	0.591	**0.550**	0.777	0.707	0.713
Sub-hypergraphlet kernel ($\tau = 1$)	0.944	0.913	**0.845**	0.884	**0.852**	**0.715**	0.747	**0.693**	0.526	0.764	0.713	0.730
L3 framework [2]	0.658	0.668	-	0.655	0.534	-	0.624	0.609	-	0.577	0.501	-
Hypergraphlet kernel (highest)	0.953	0.921	-	0.900	0.858	-	0.760	0.687	-	0.802	0.818	-

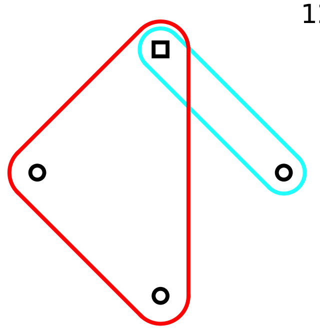
Base hypergraphlets up to 4 vertices

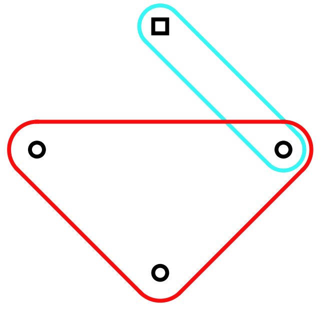


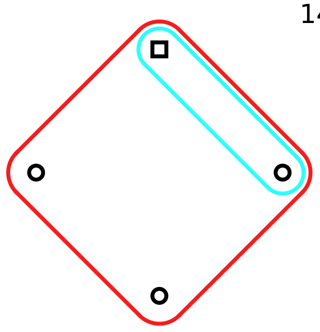

9

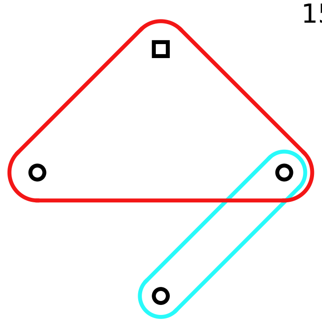

10

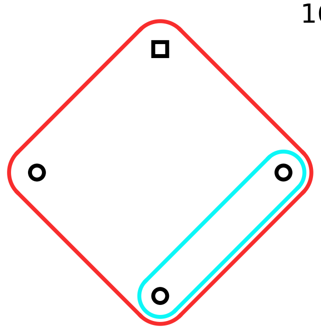

11

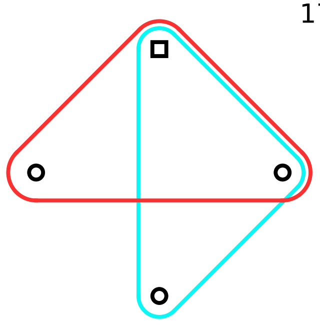

12

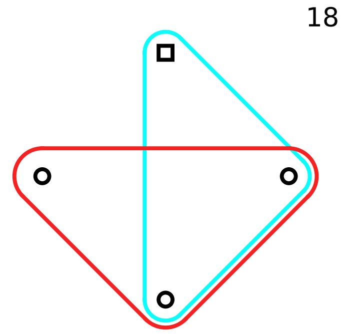

13

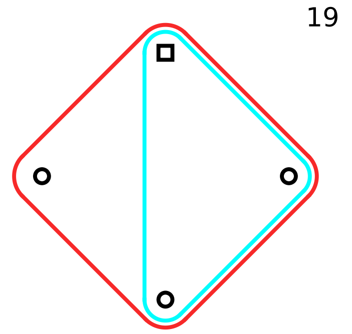

14

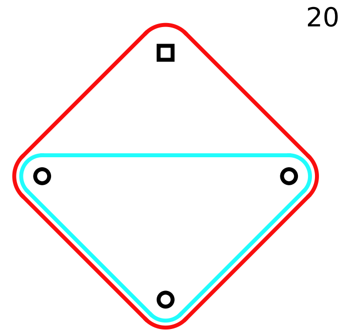

15

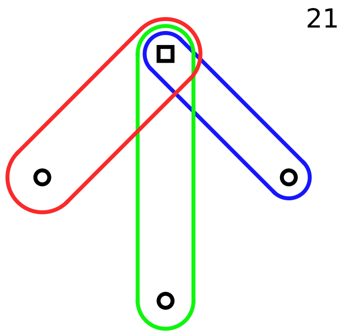

16

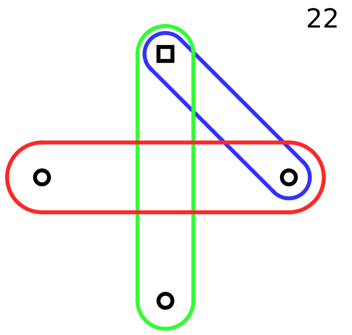

17

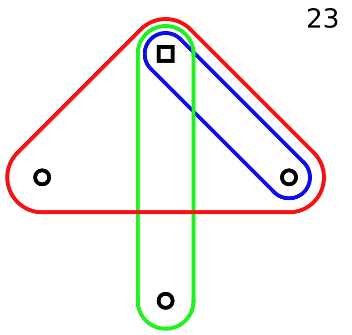

18

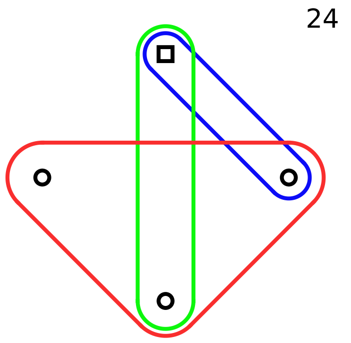

19

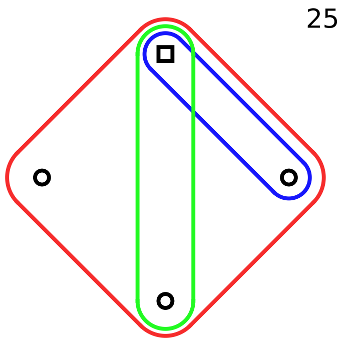

20

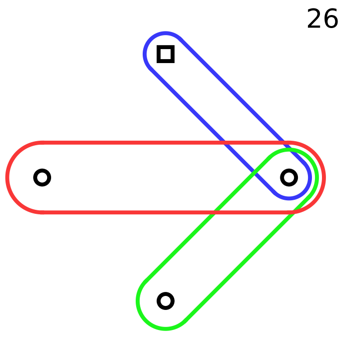

21

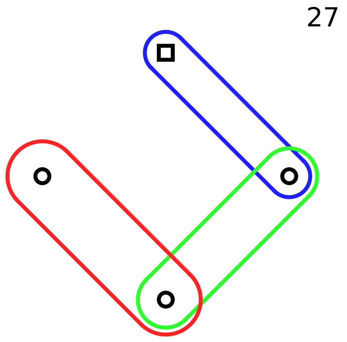

22

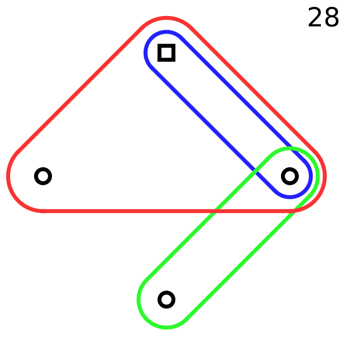

23

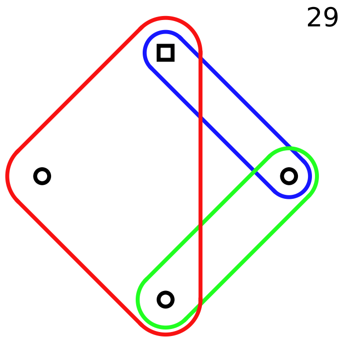

24

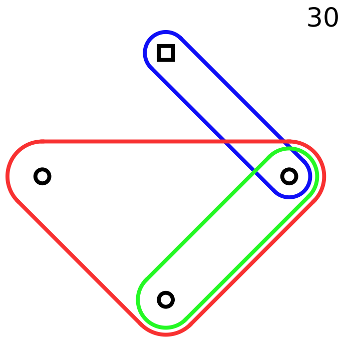

25

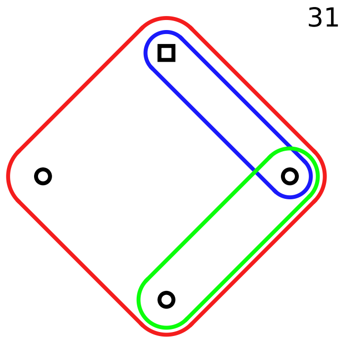

26

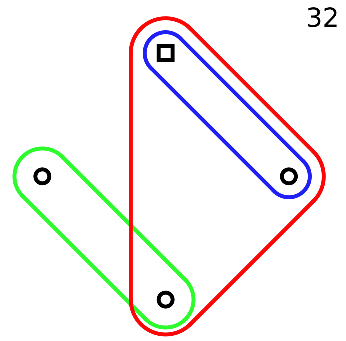

27

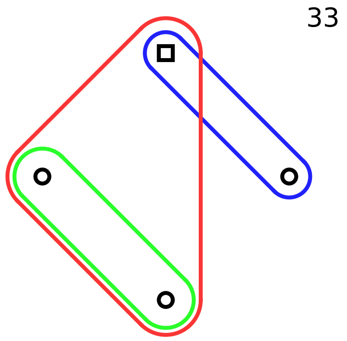

28

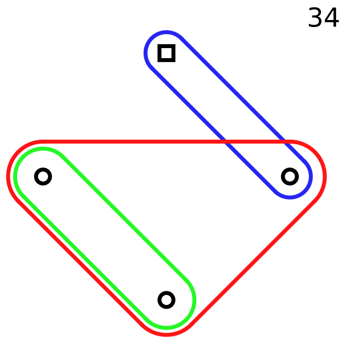

29

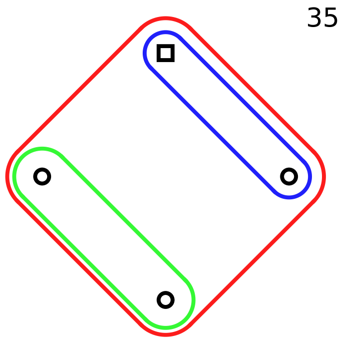

30


31

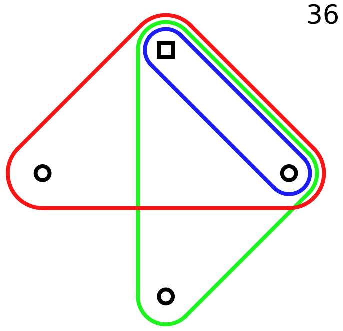

32


33

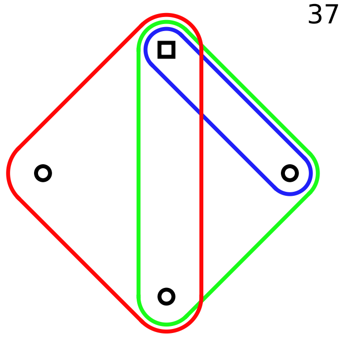

34


35

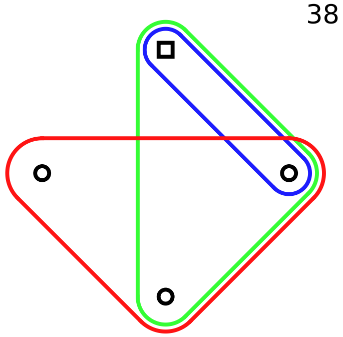
36



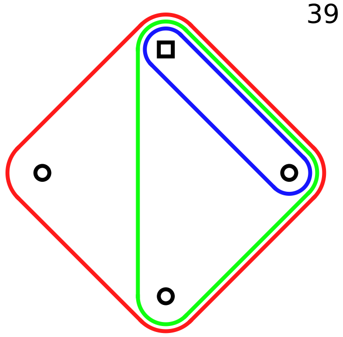
37



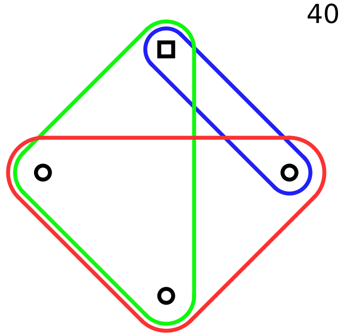
38



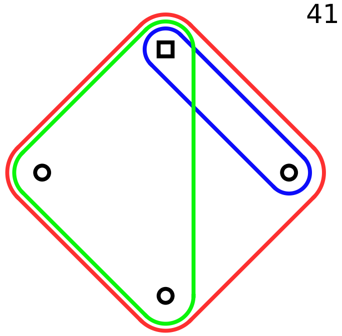
39



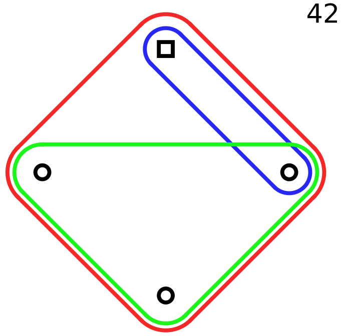
40



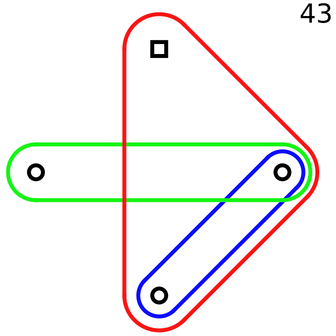
41



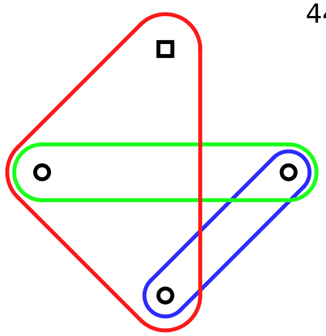
42



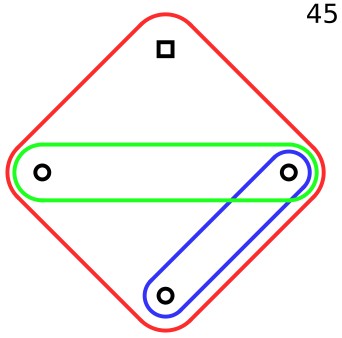
43



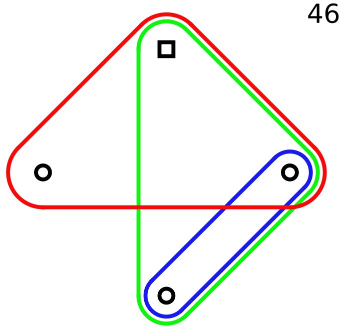
44



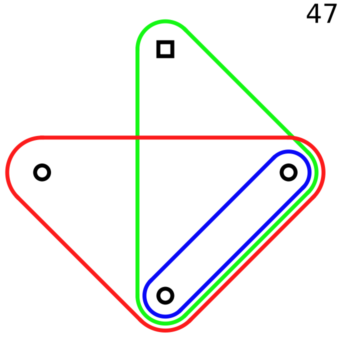
45



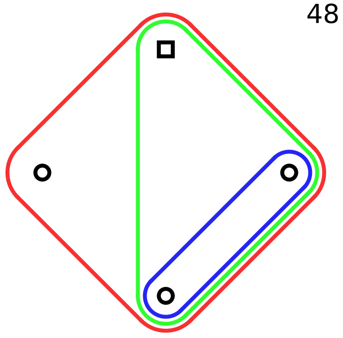
46



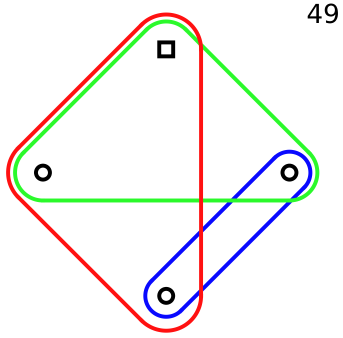
47



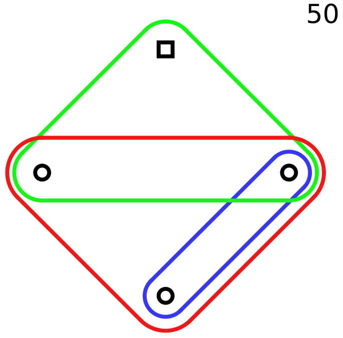
48



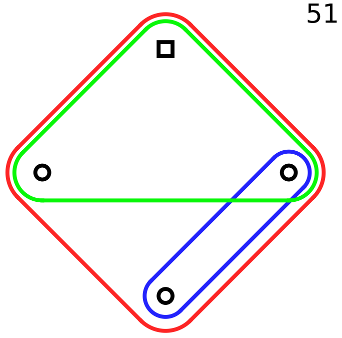
49



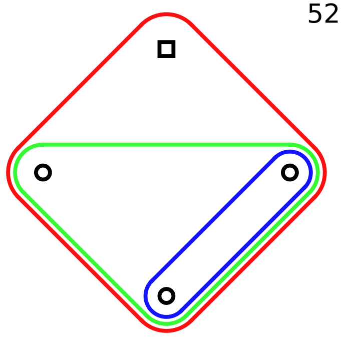
50



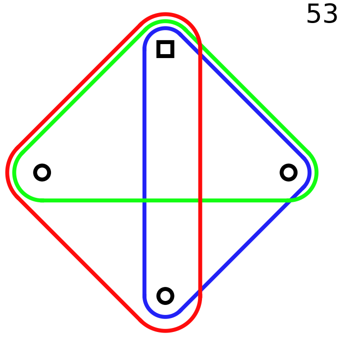
51

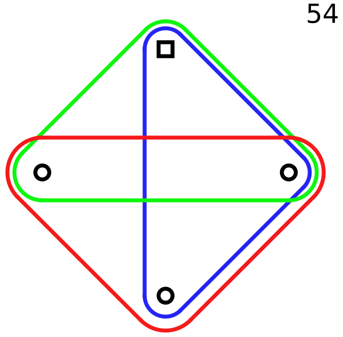


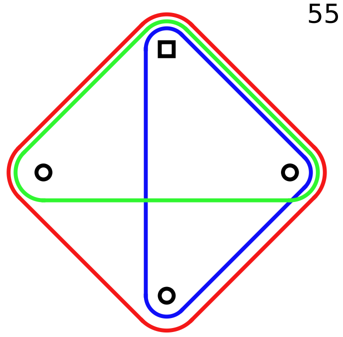
52

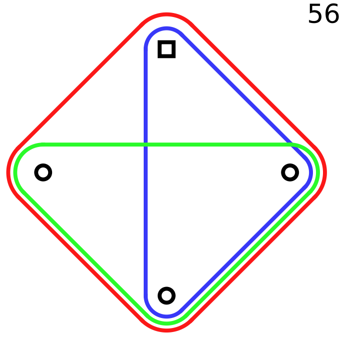


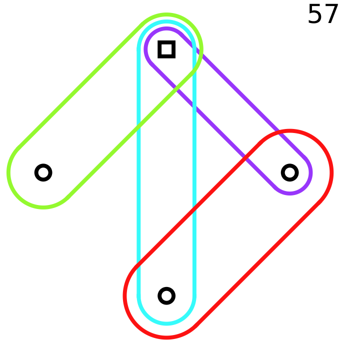
53

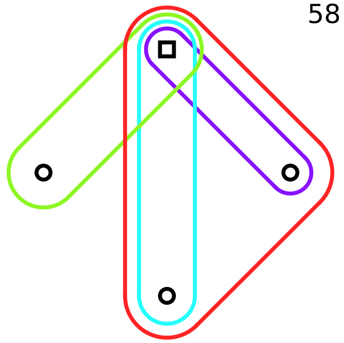


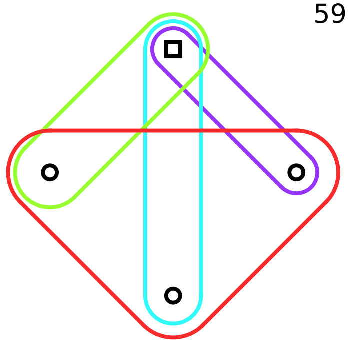

54

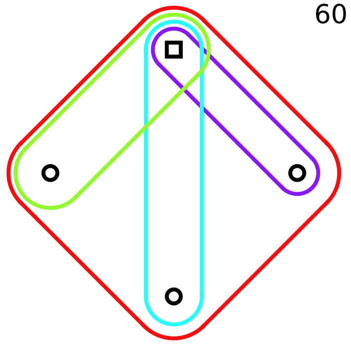

55

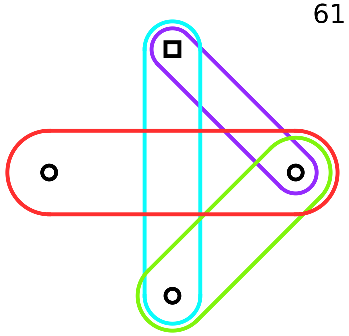

56

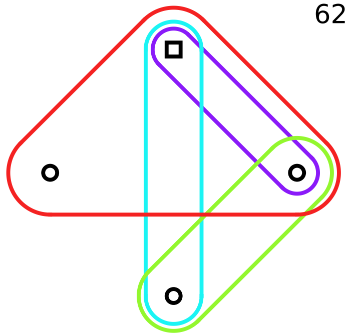

57

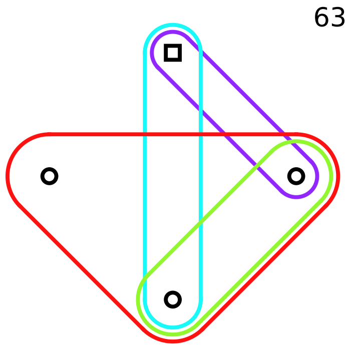

58

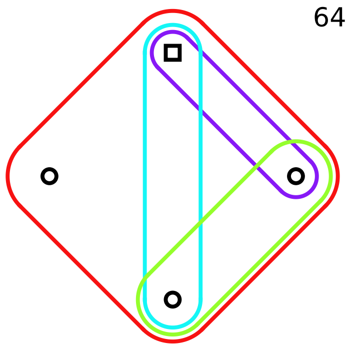

59

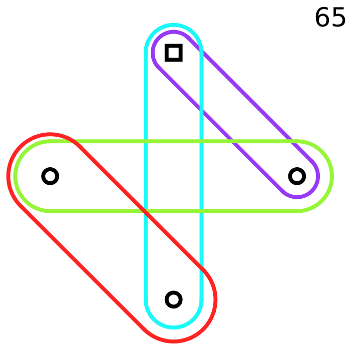

60

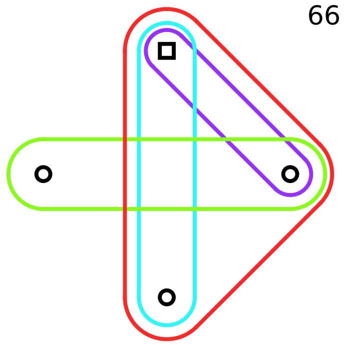

61

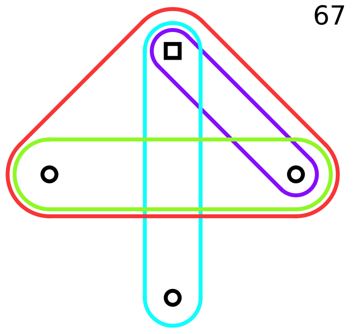

62

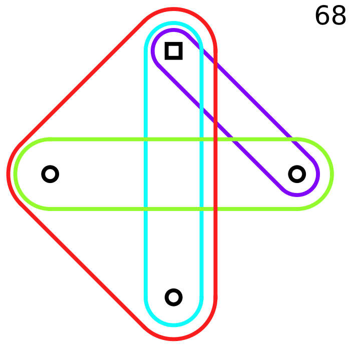

63

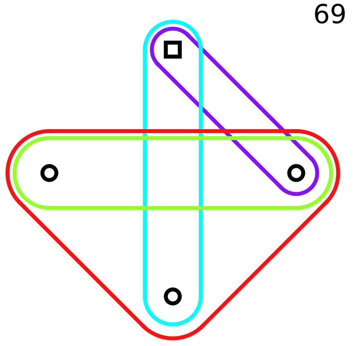

64

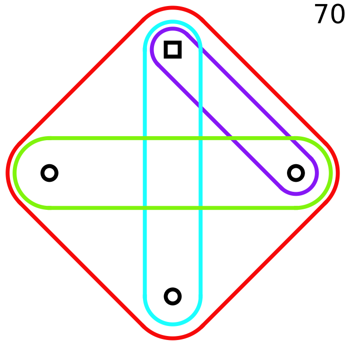

65

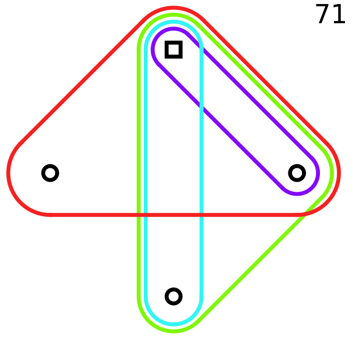

66


67

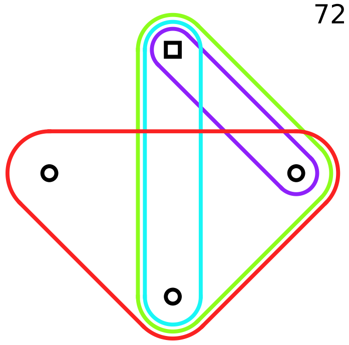

68


69

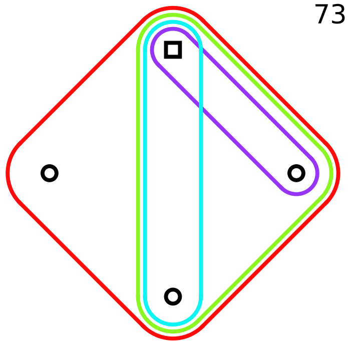

70


71

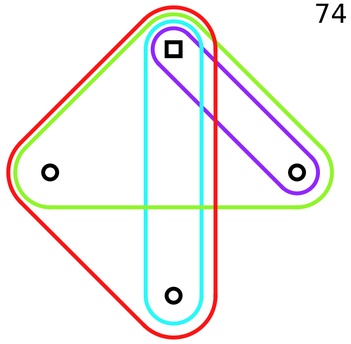
72



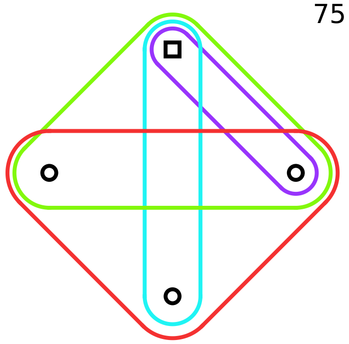
73



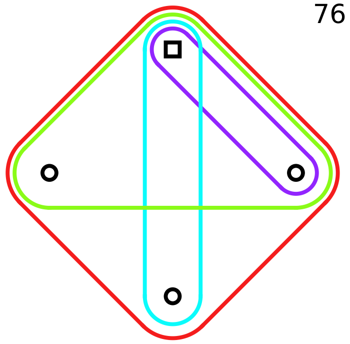
74



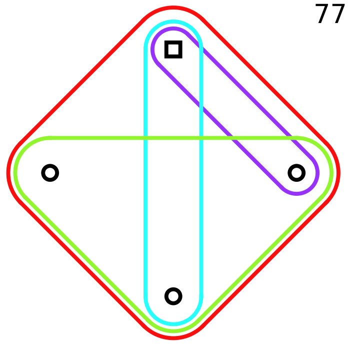
75



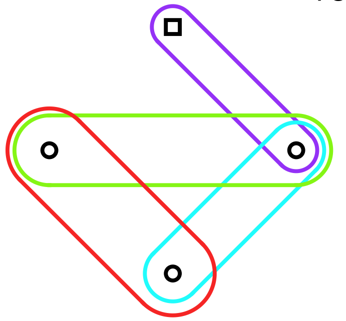
76



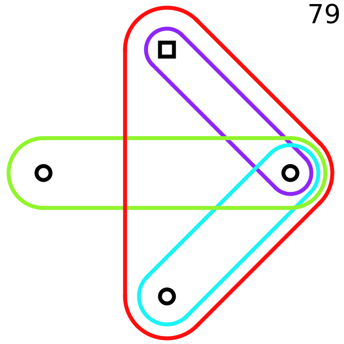
77



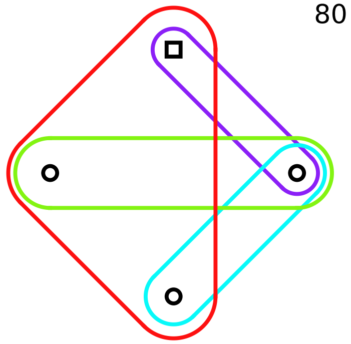
78

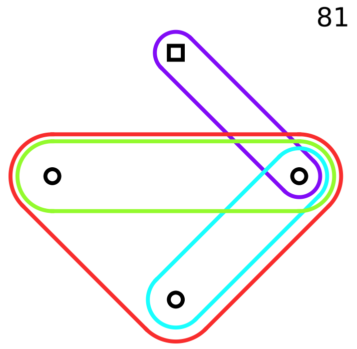
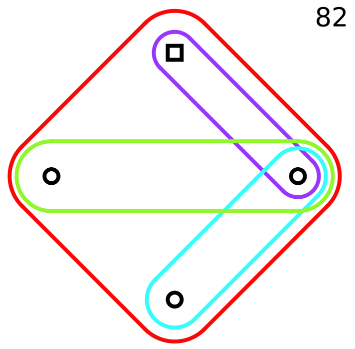
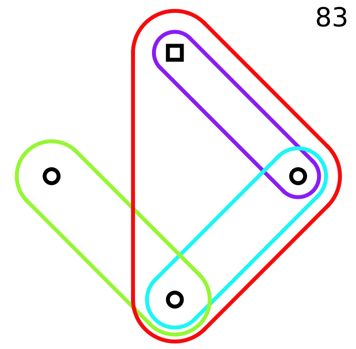
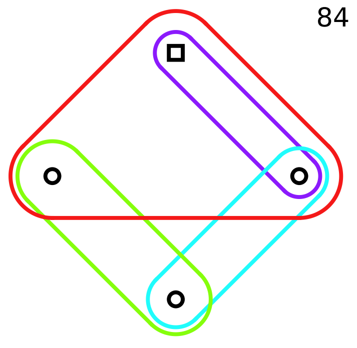
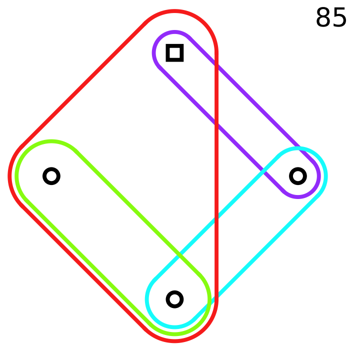
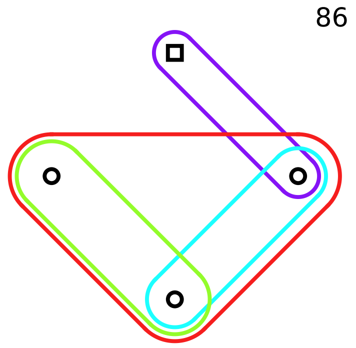
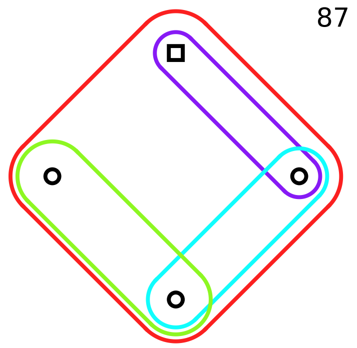
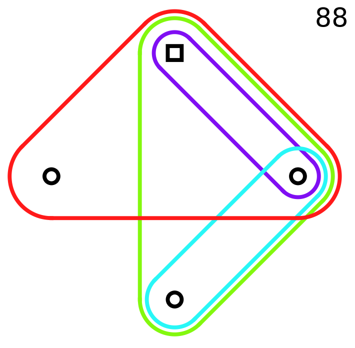
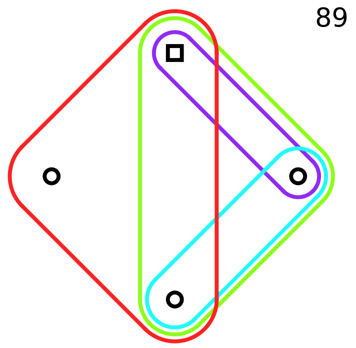


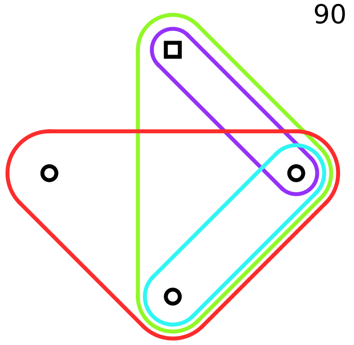
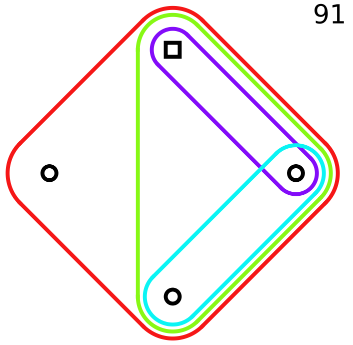
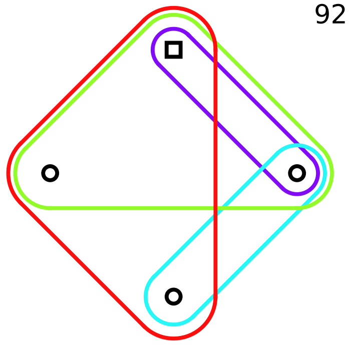
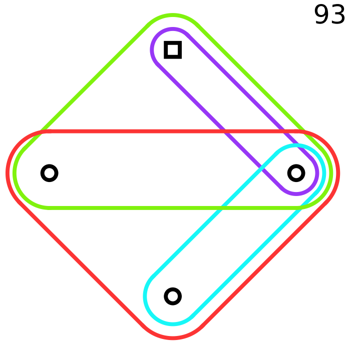
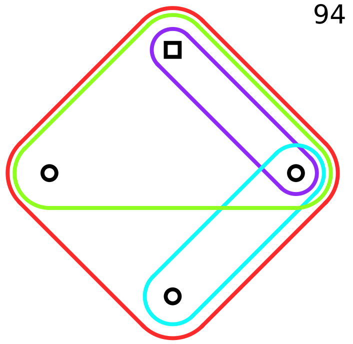
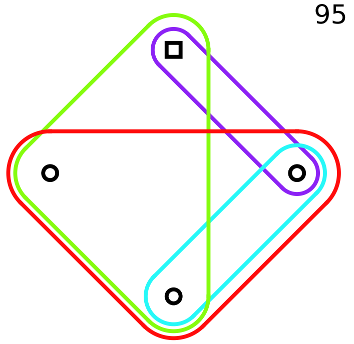
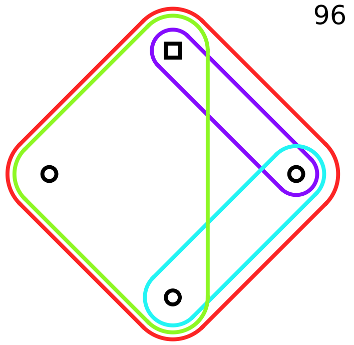
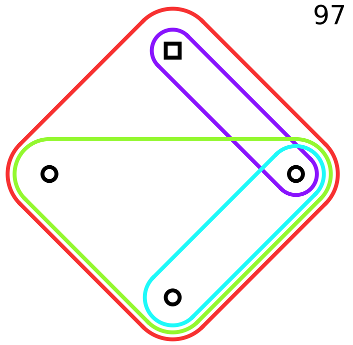
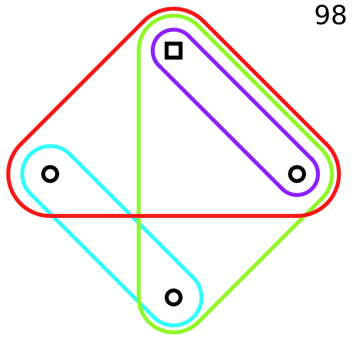
79

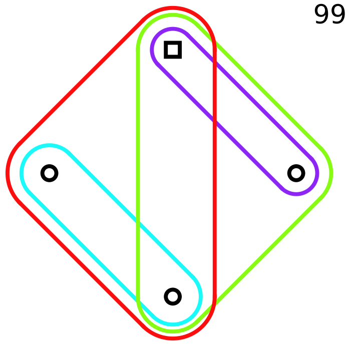


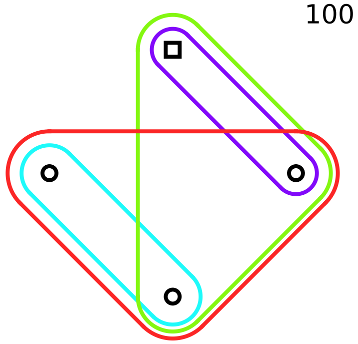
80

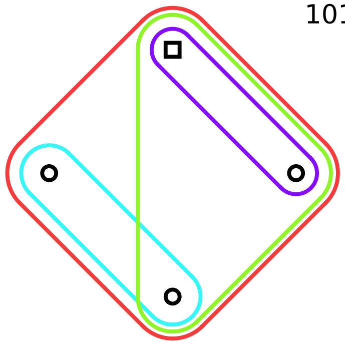


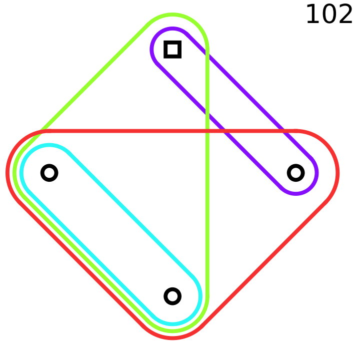









81
82
83
84
85
86
87
88
89

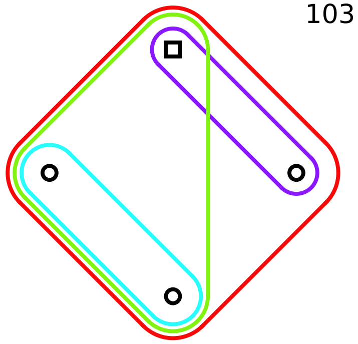

90

91

92

93

94

95

96

97

98

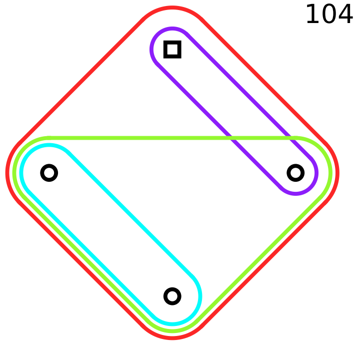

99

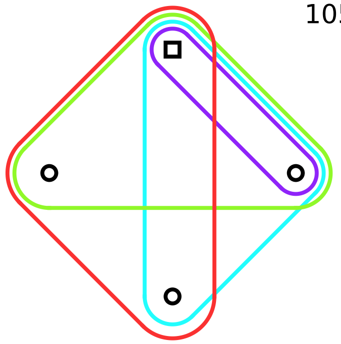

100

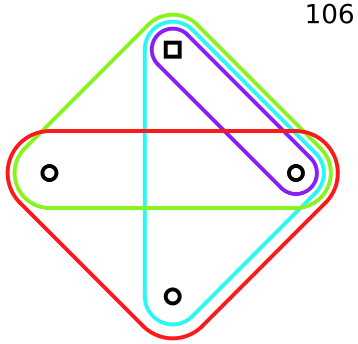

101

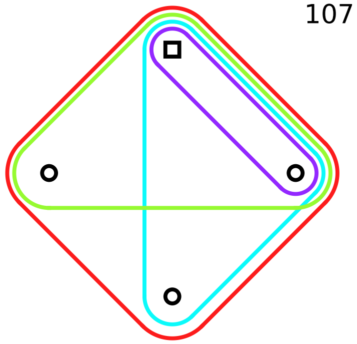

102

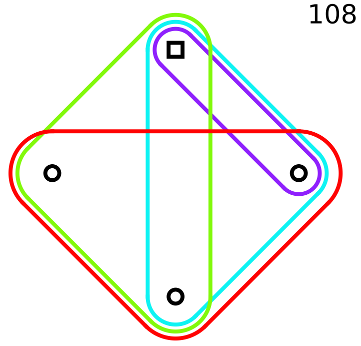

103

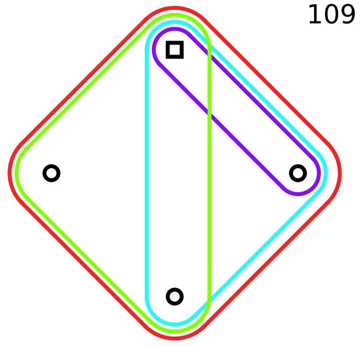

104

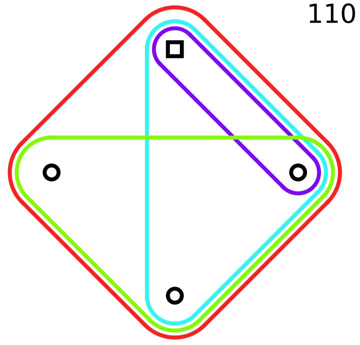

105

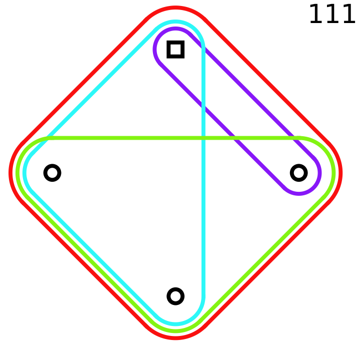

106

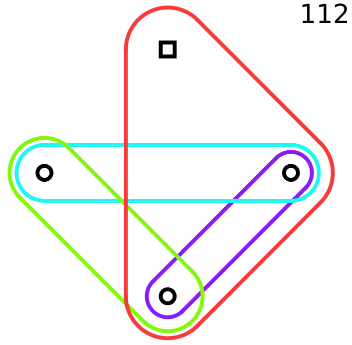

107

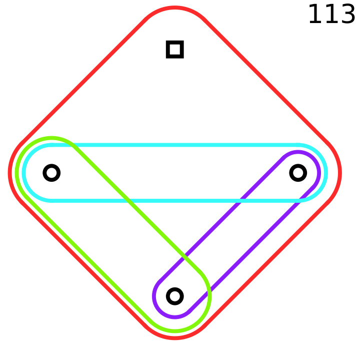

108

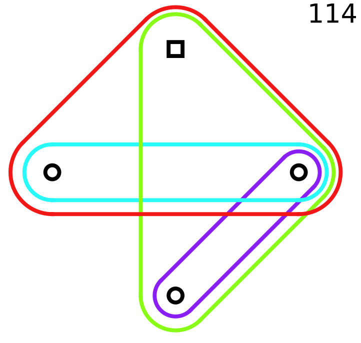

109

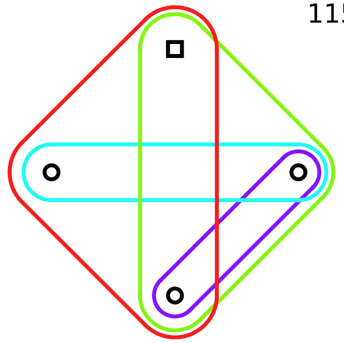

110

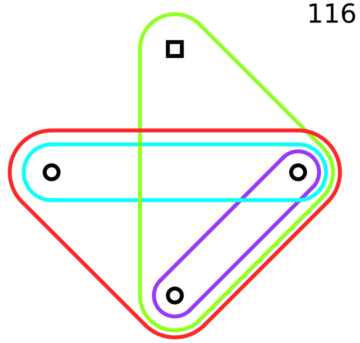

111

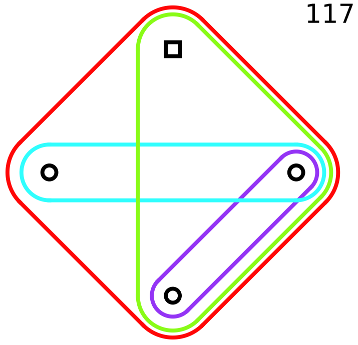

112

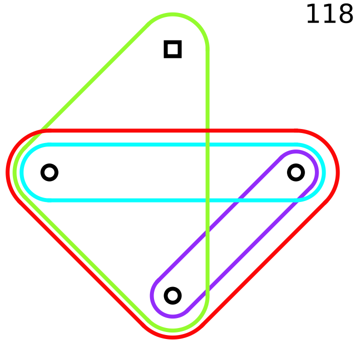

113

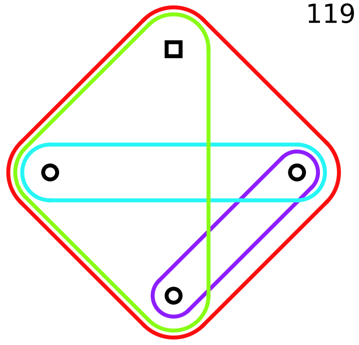

114

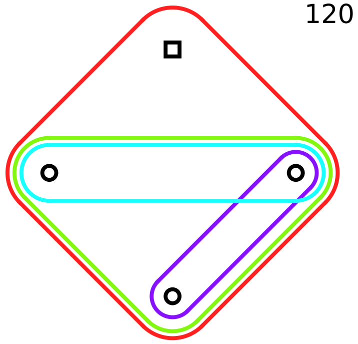

115

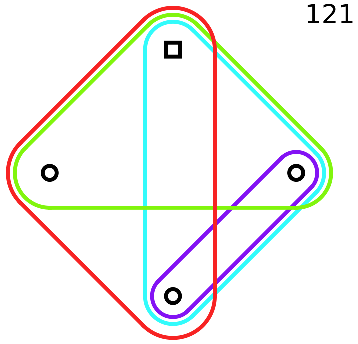

116

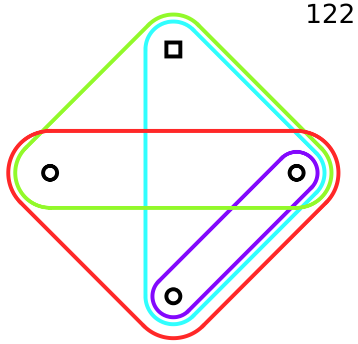

117

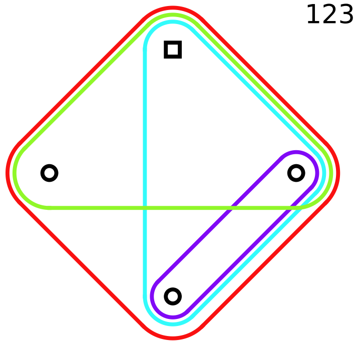

118

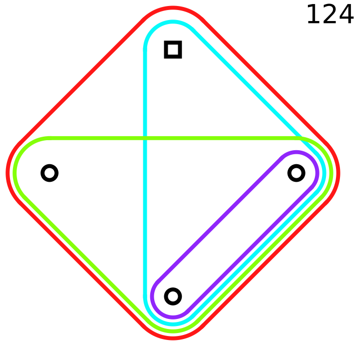

119

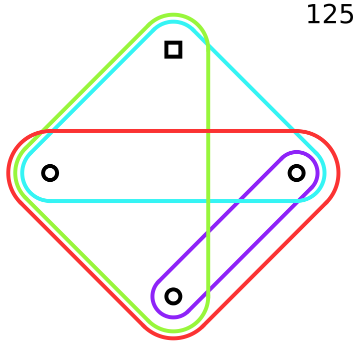

120

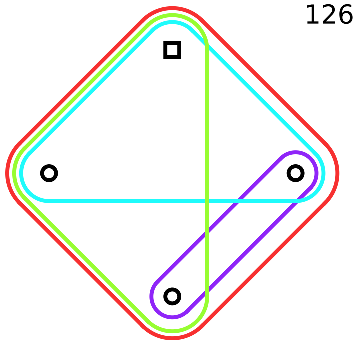

121

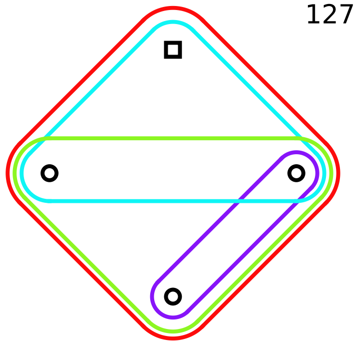

122

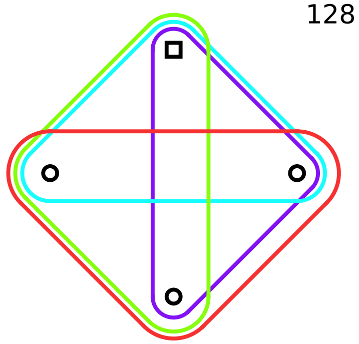

123

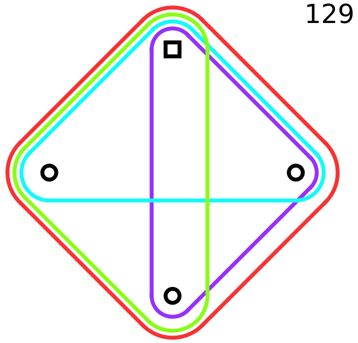

124

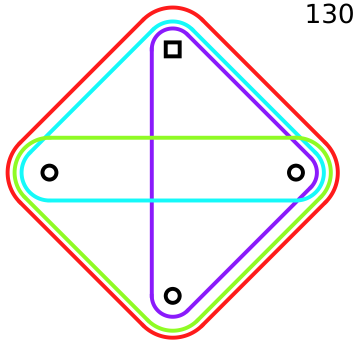

125

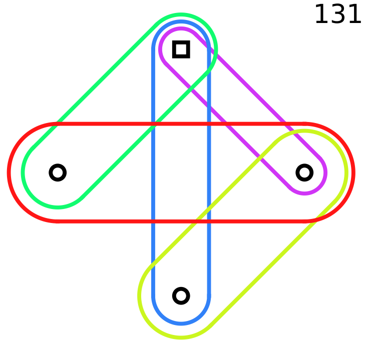

126

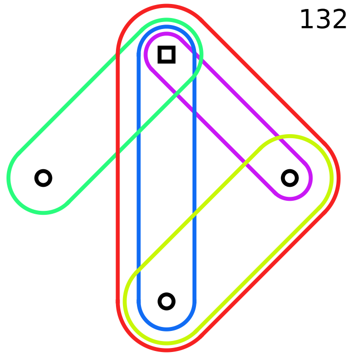

127

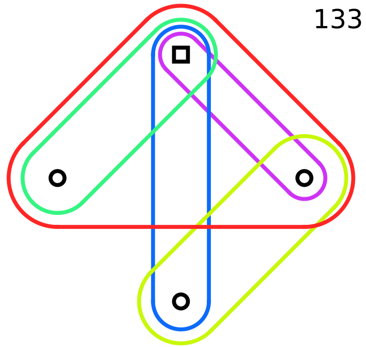

128

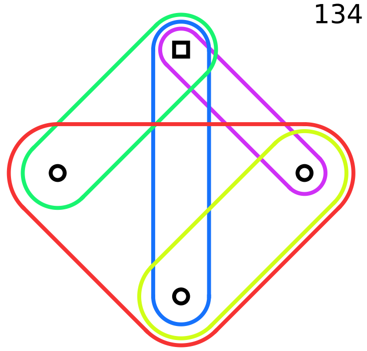

129

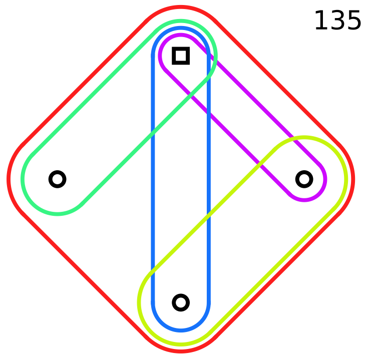

130

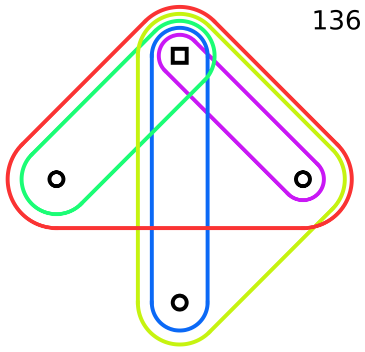

131

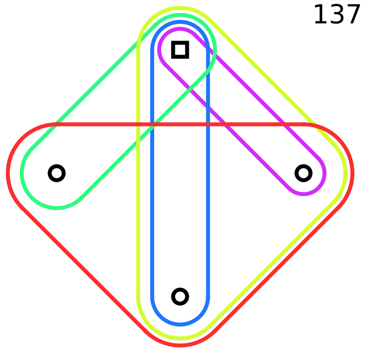

132

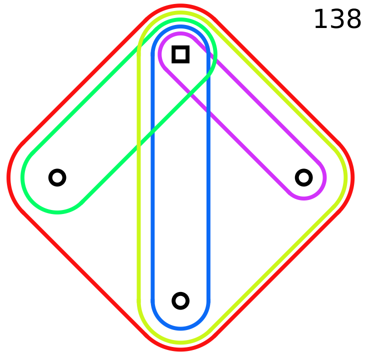

133

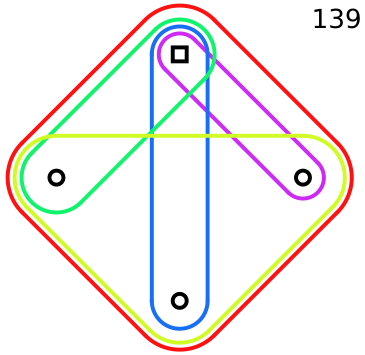

134

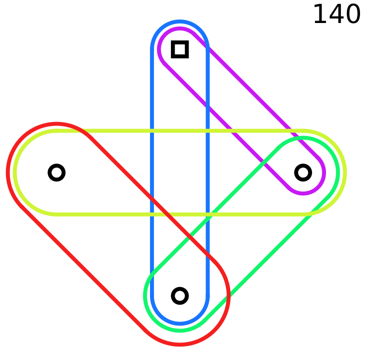

135

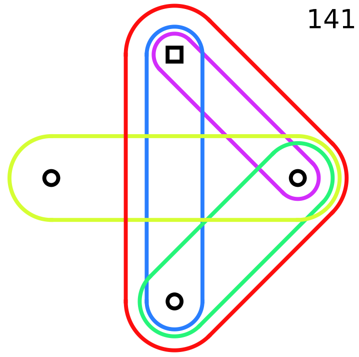

136

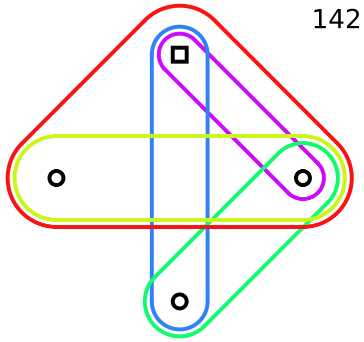

137

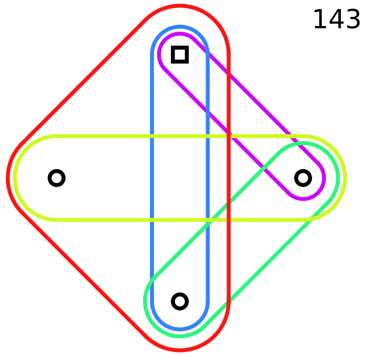

138

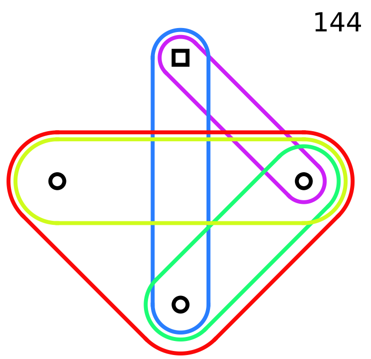

139

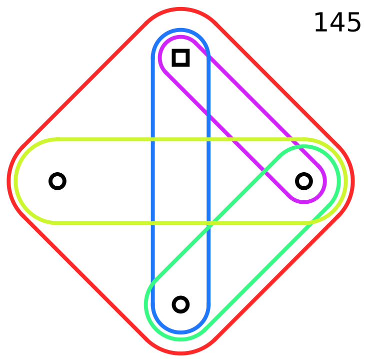

140

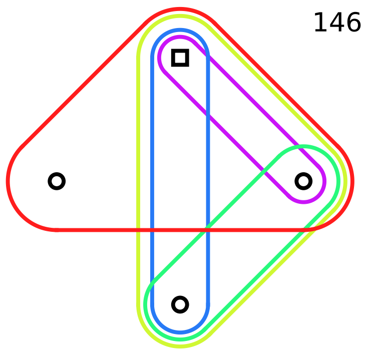

141

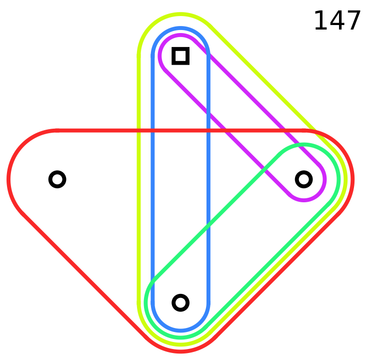

142

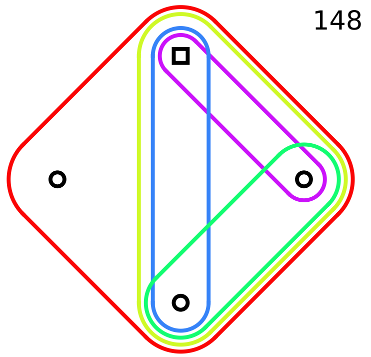

143

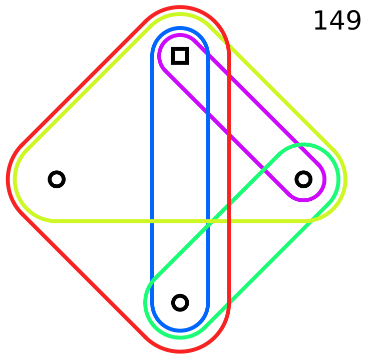

144

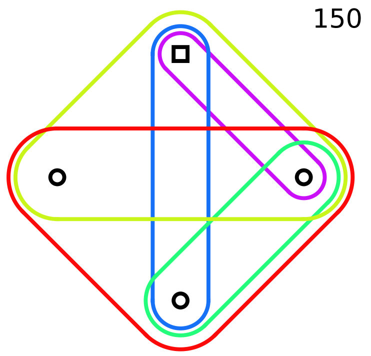

145

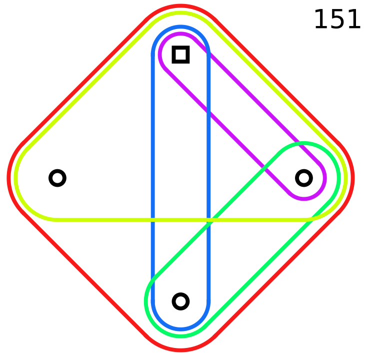

146

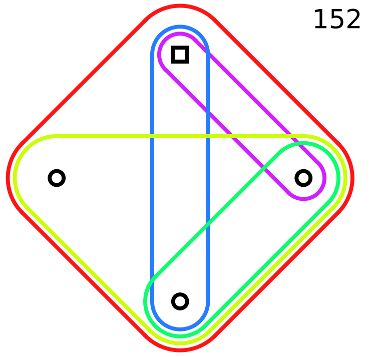

147

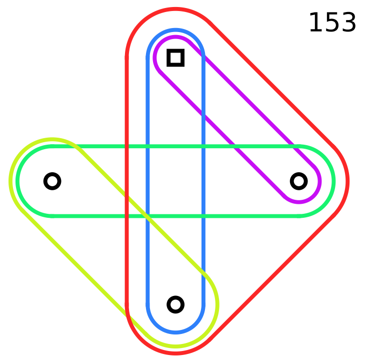

148

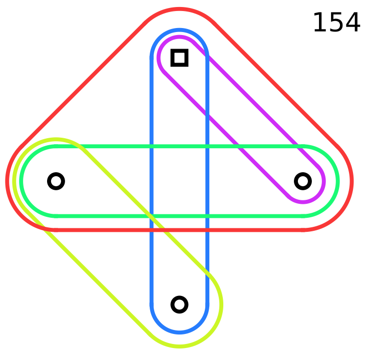

149

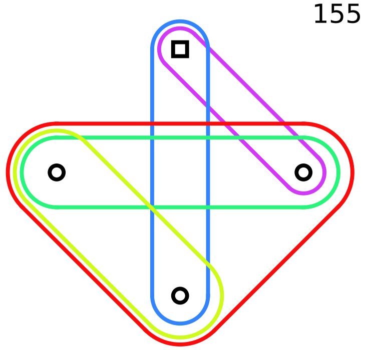

150

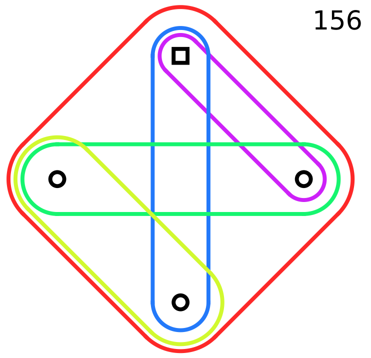

151

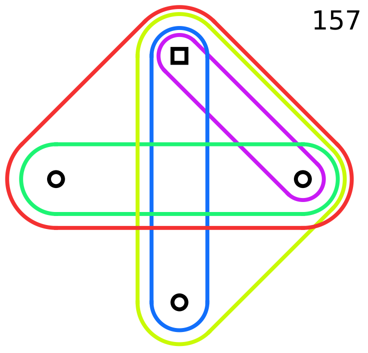

152

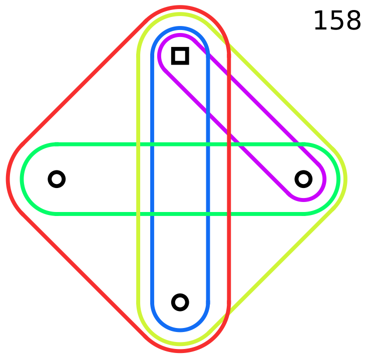

153

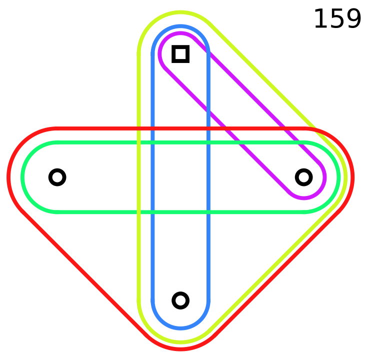

154

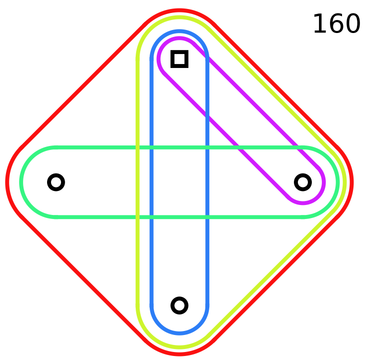

155

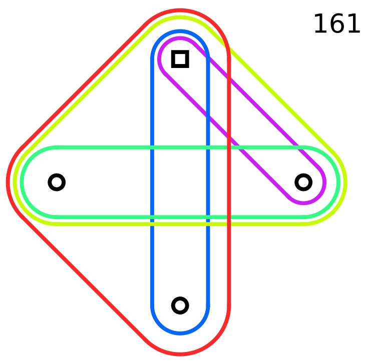

156

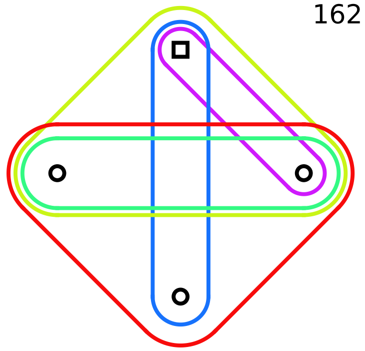

157

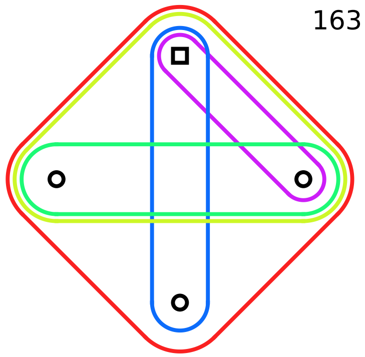

158

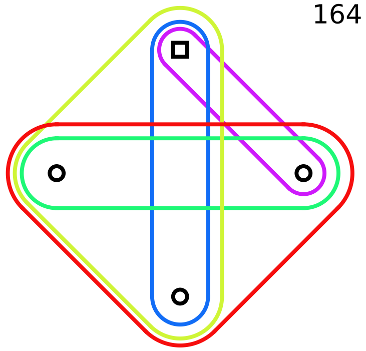

159

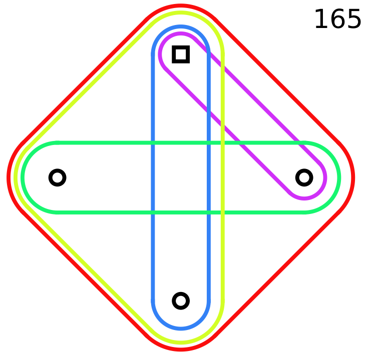

160

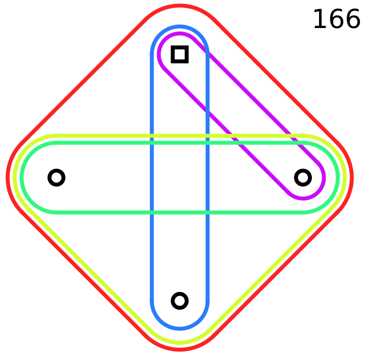

161

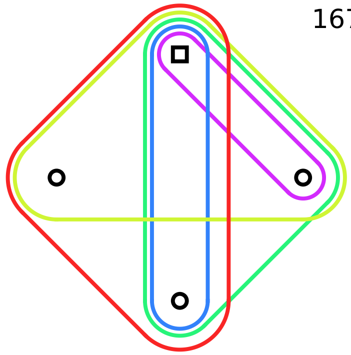

162

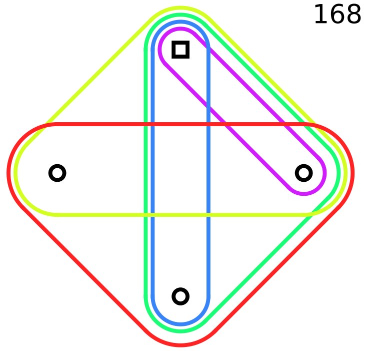

163

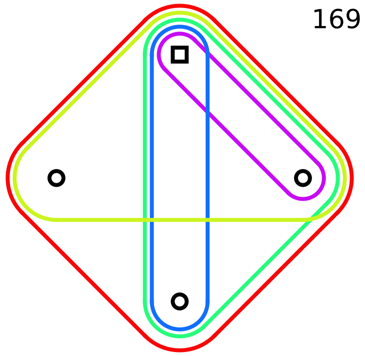

164

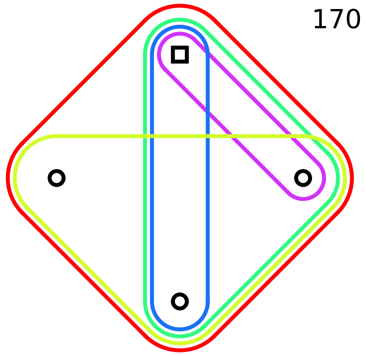

165

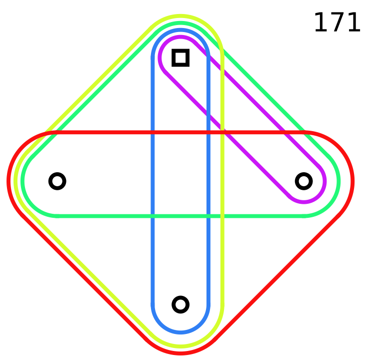

166

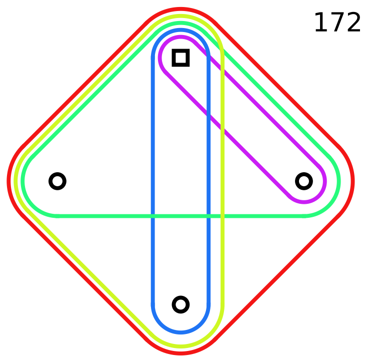

167

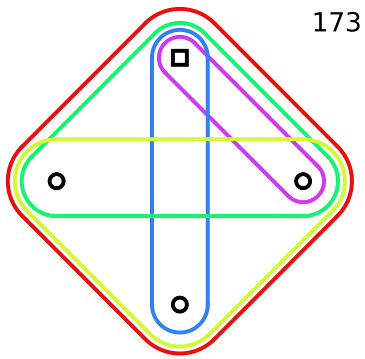

168

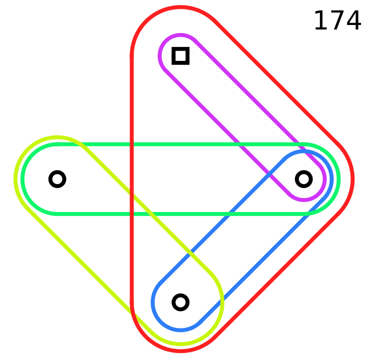

169

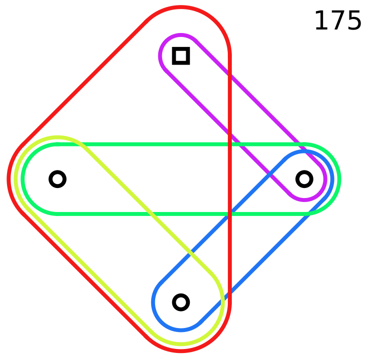

170

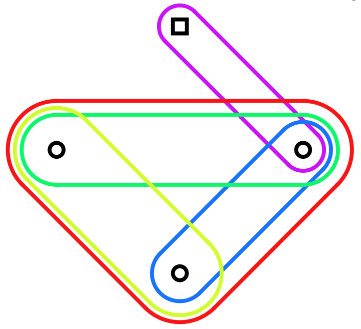

171

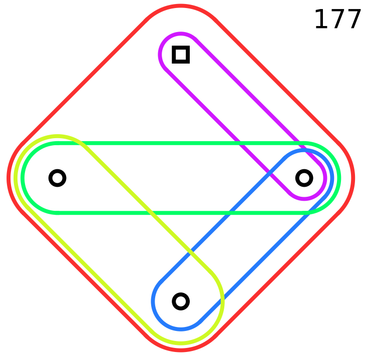

172

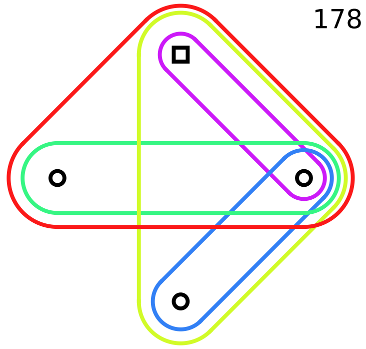

173

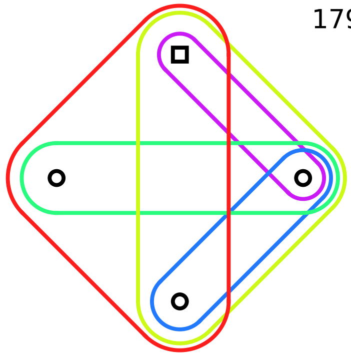

174

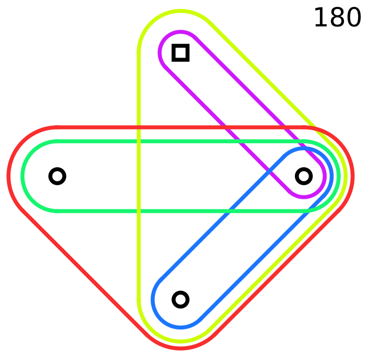

175

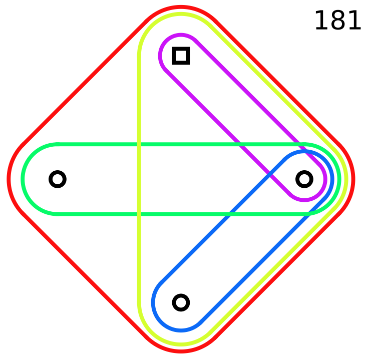

176

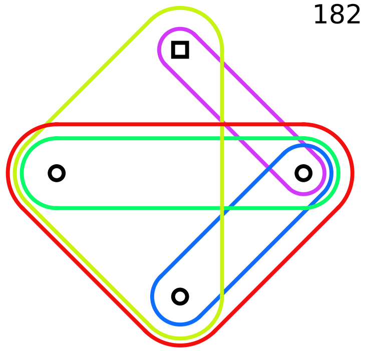

177

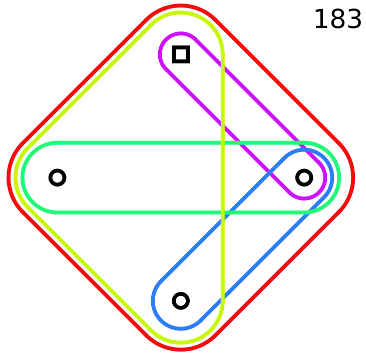

178

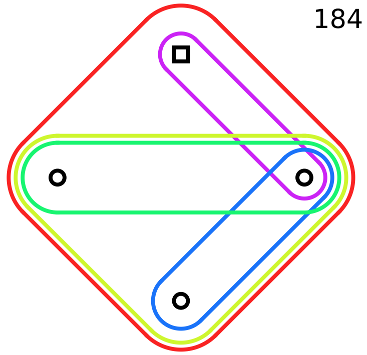

179

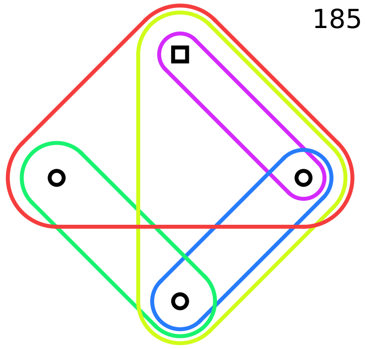

180

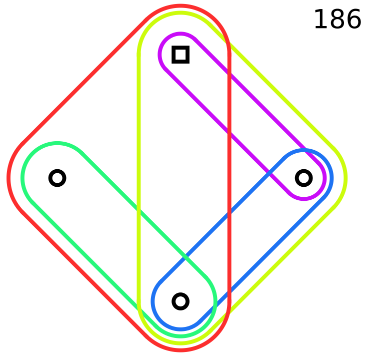

181

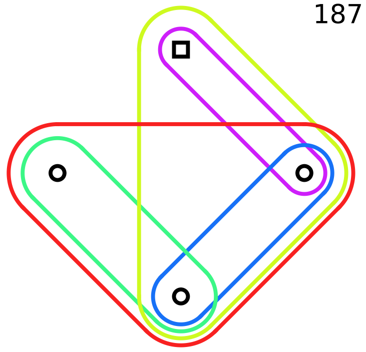

182

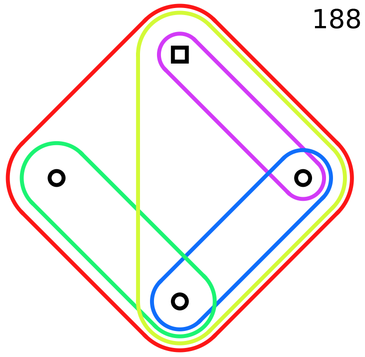

183

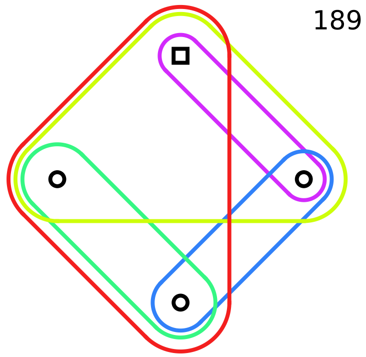

184

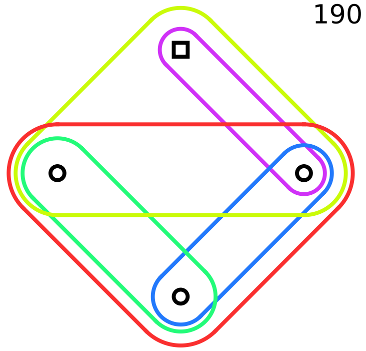

185

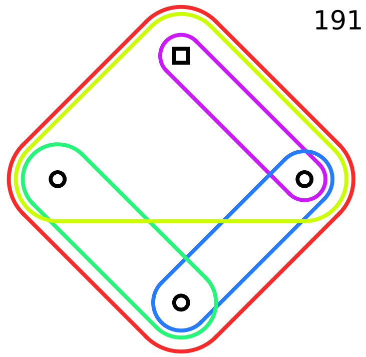

186

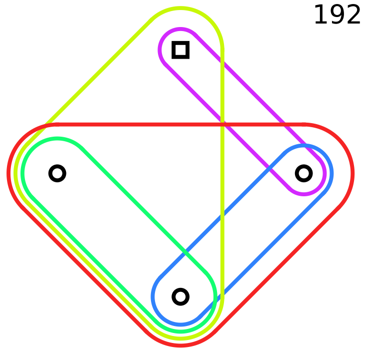

187

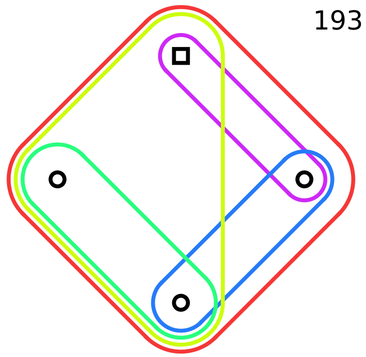

188

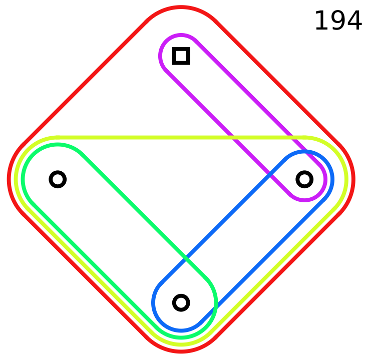

189

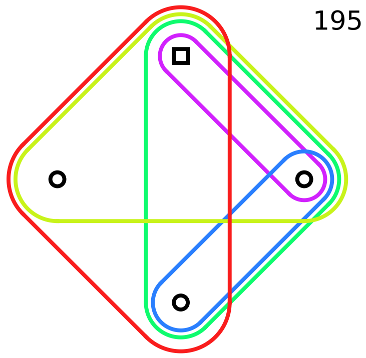

190

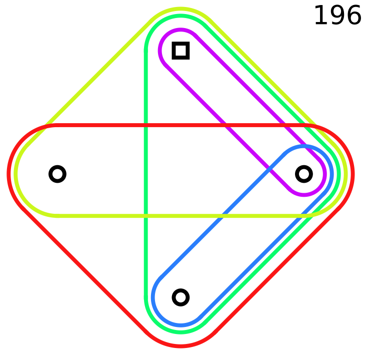

191

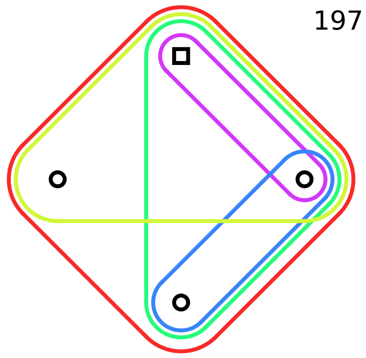

192

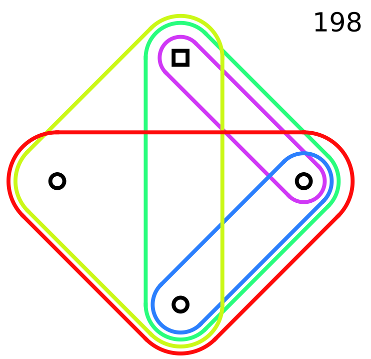

193

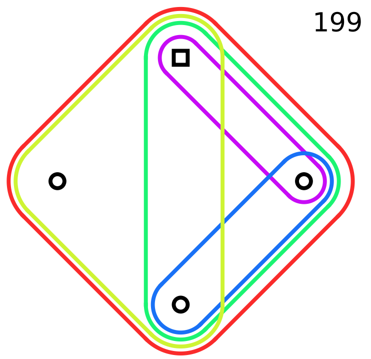

194

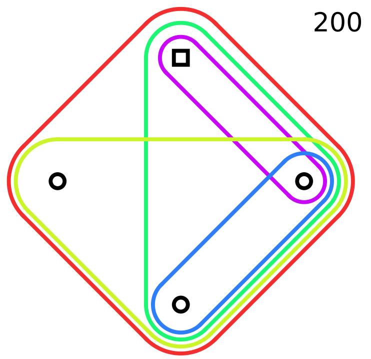

195

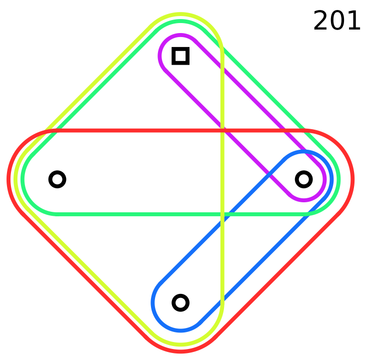

196

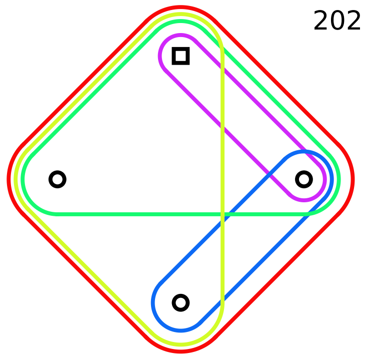

197

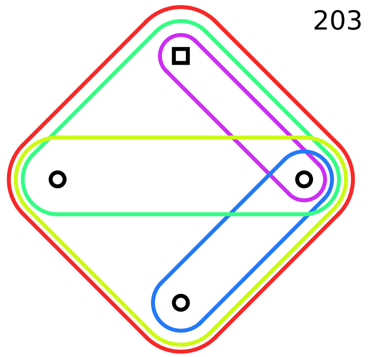

198

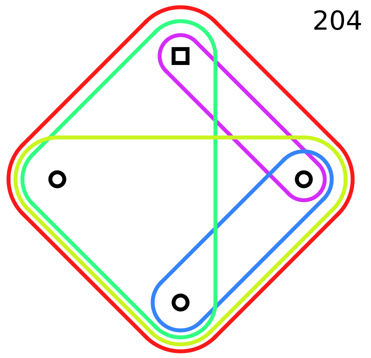

199

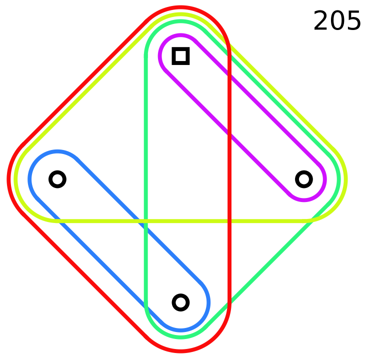

200

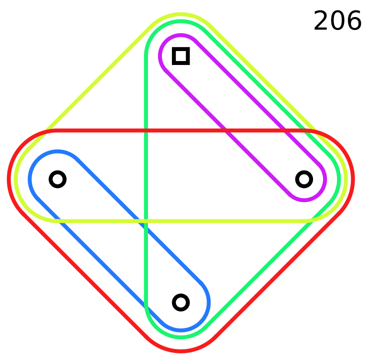

201

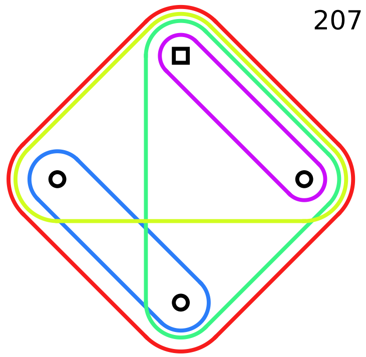

202

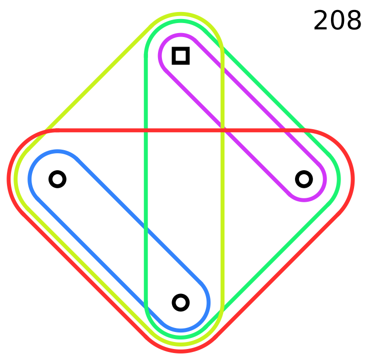

203

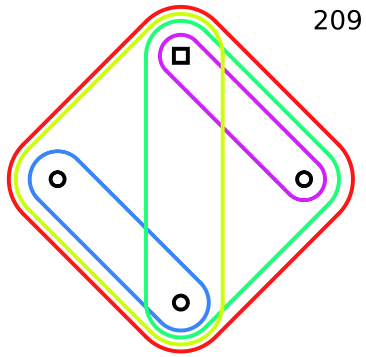

204

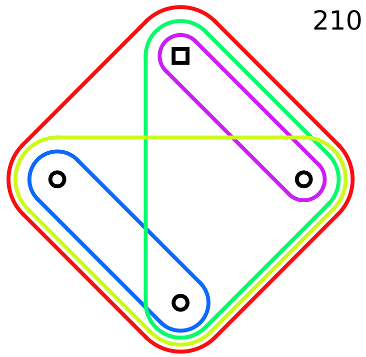

205

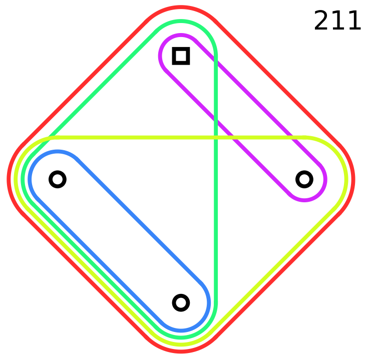

206

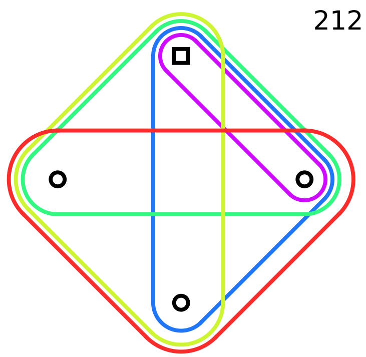

207

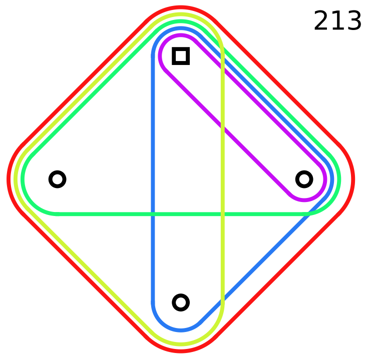

208

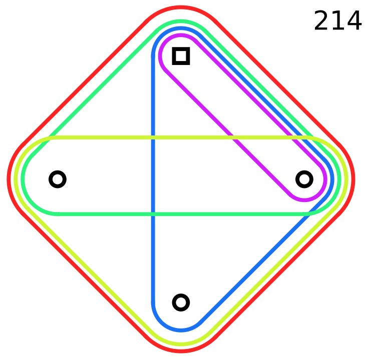

209

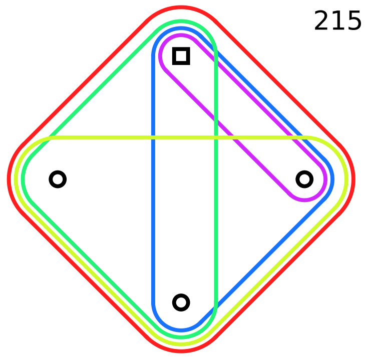

210

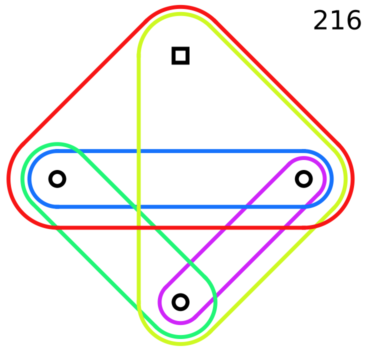

211

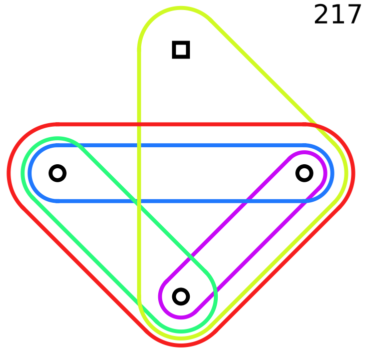

212

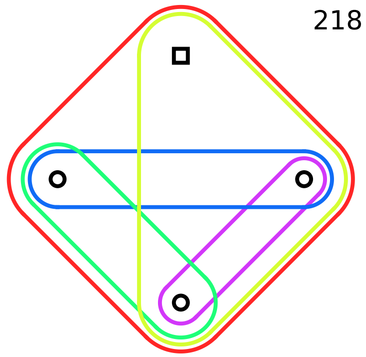

213

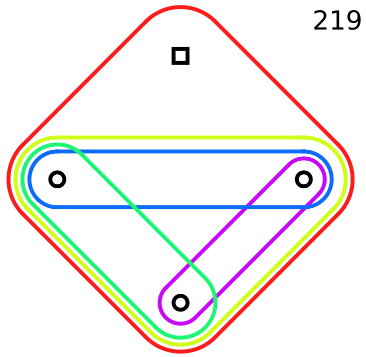

214

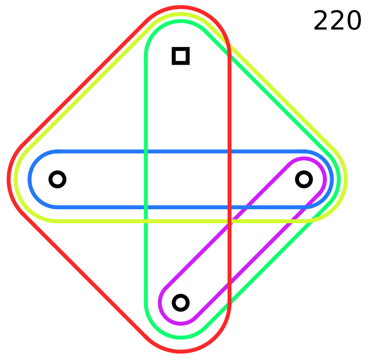

215

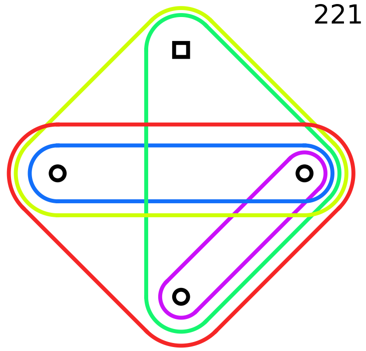

216

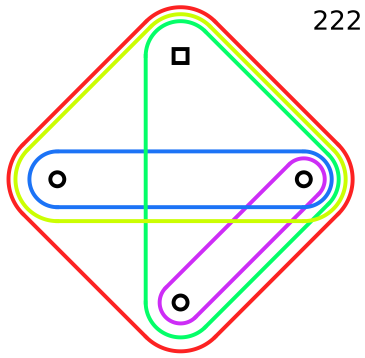

217

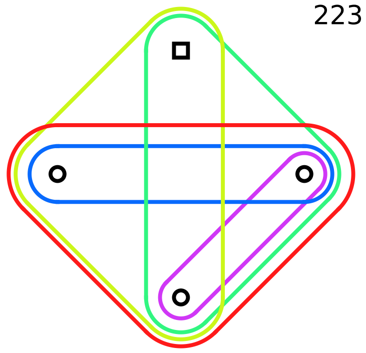

218

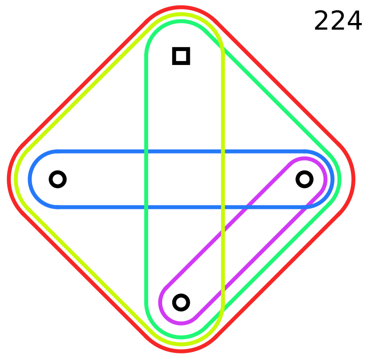

219

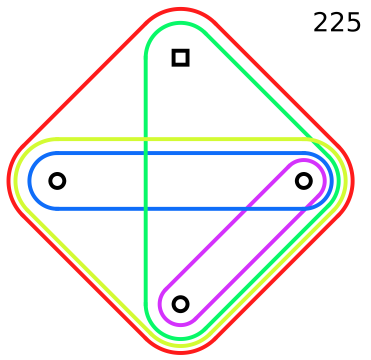
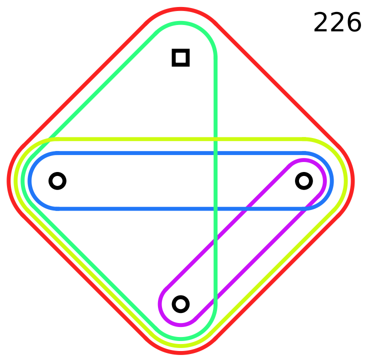
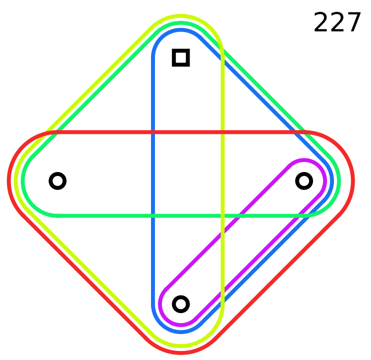
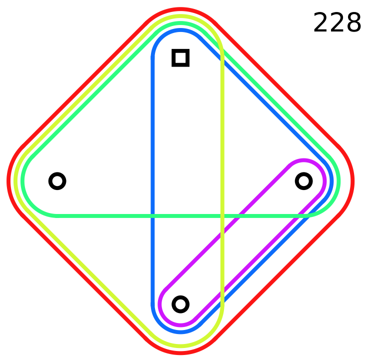
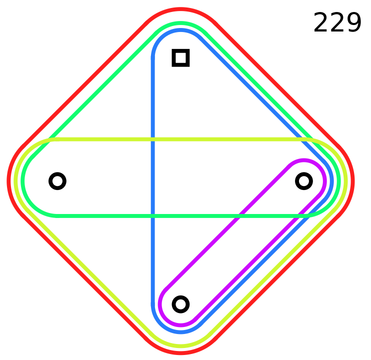
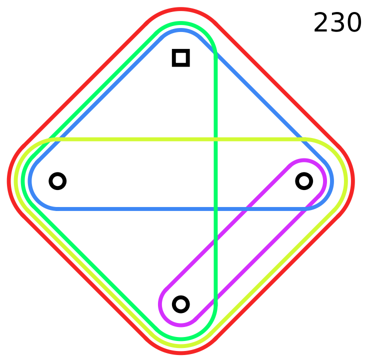
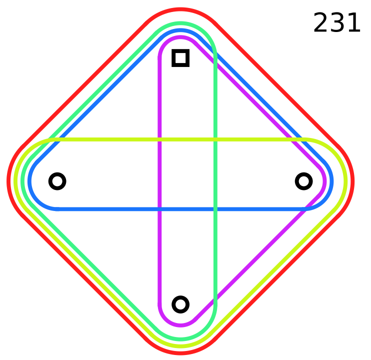
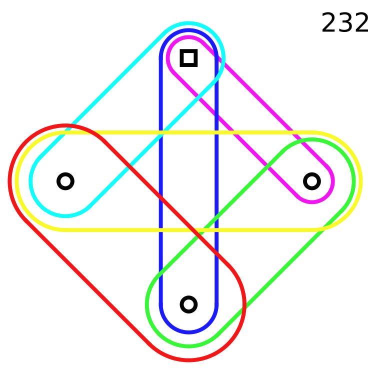
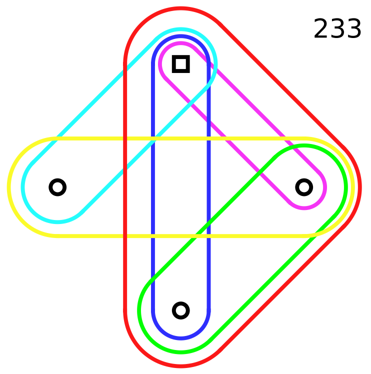

220

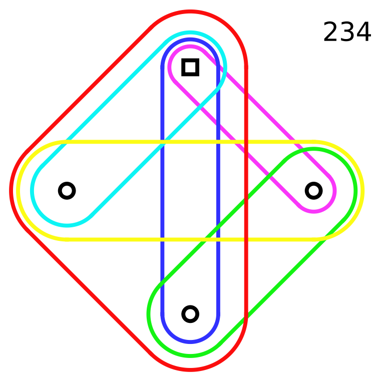

221

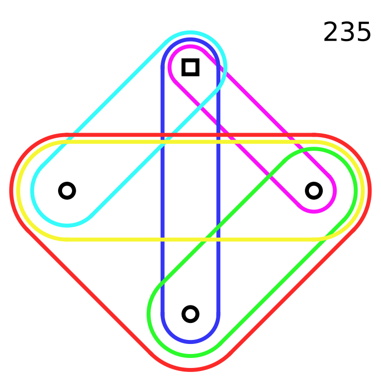

222

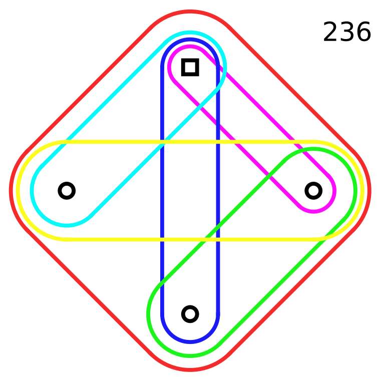

223

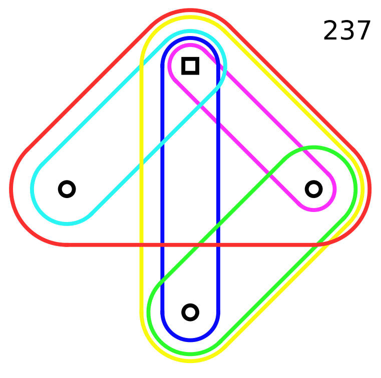

224

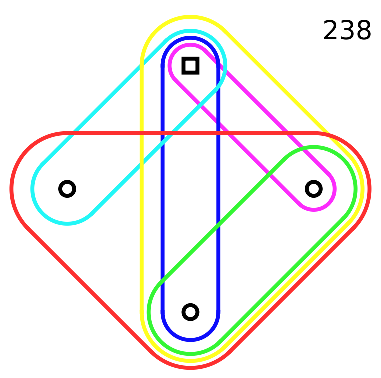

225

226

227

228

229

230

231

232

233

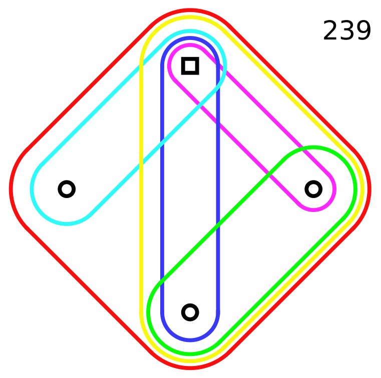

234

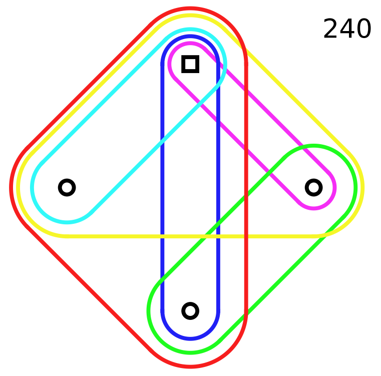

235

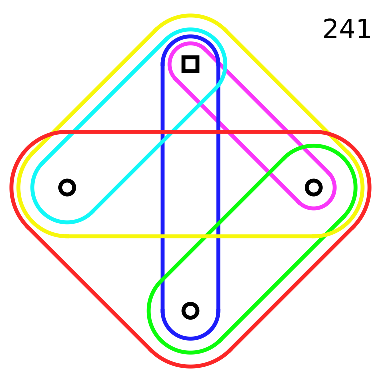

236

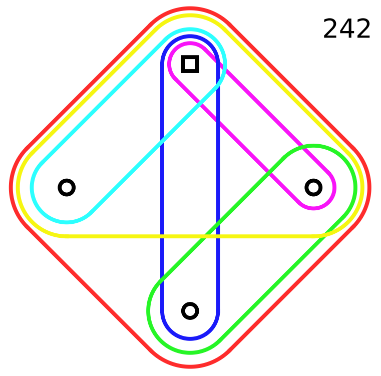

237

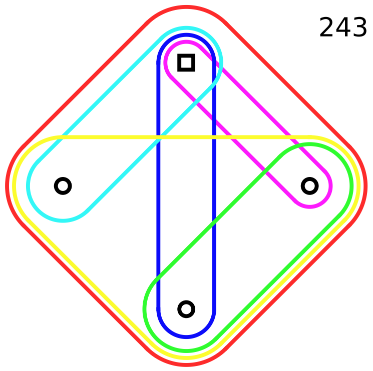

238

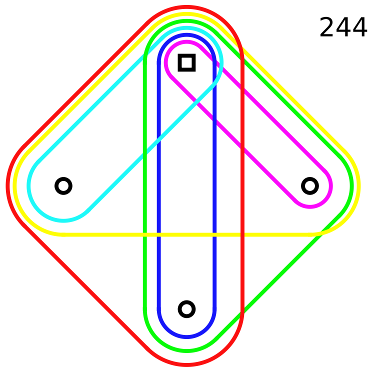

239

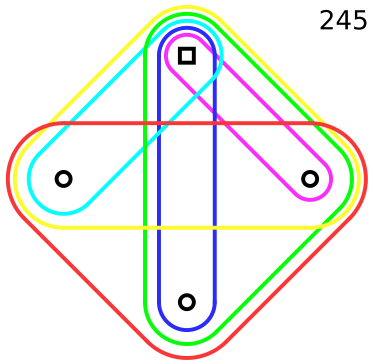

240

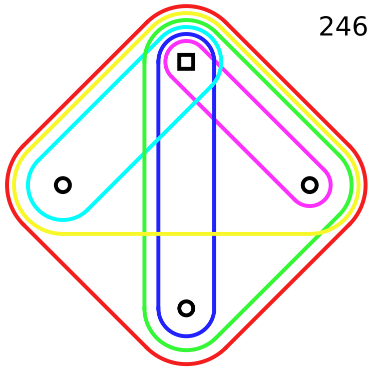

241

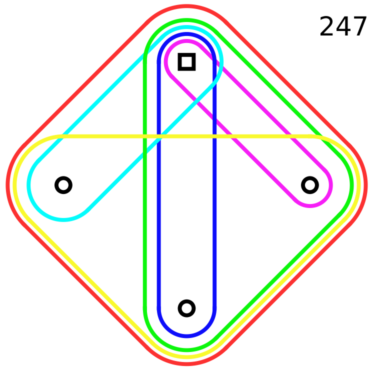

242

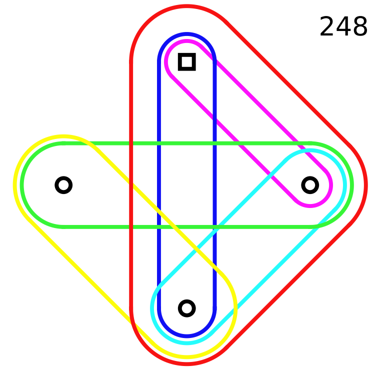
243


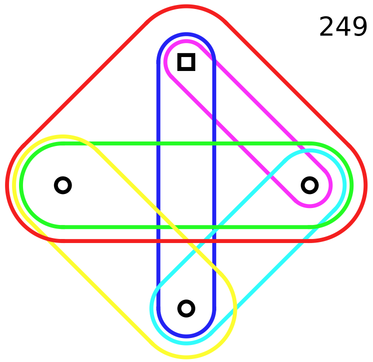
244


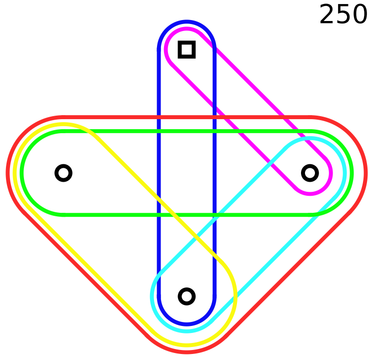
245


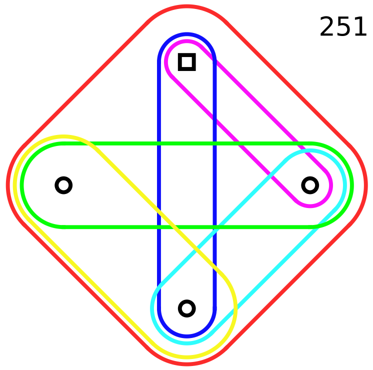
246


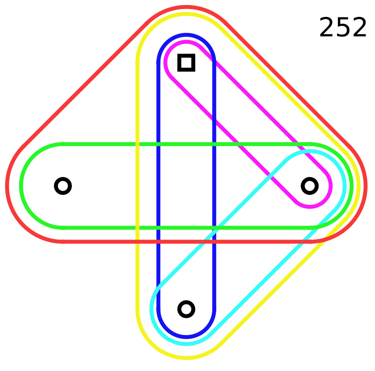
247


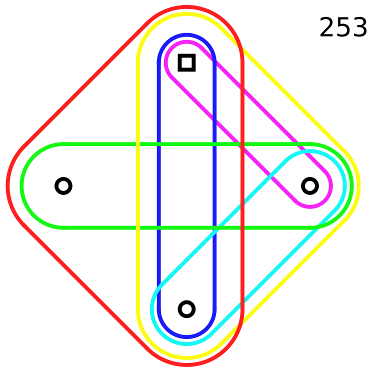
248


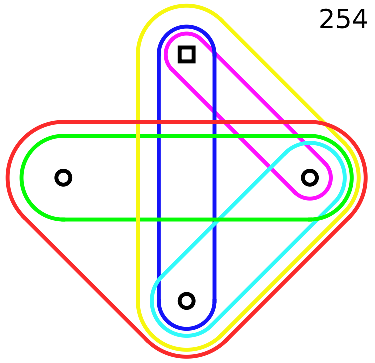
249


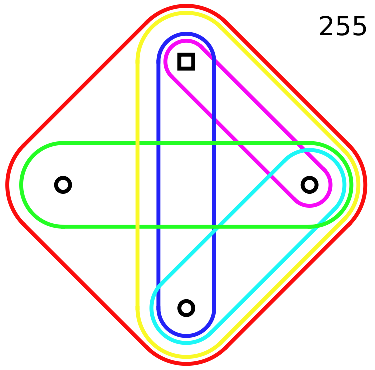
250


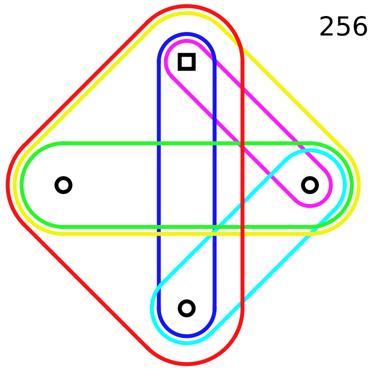
251


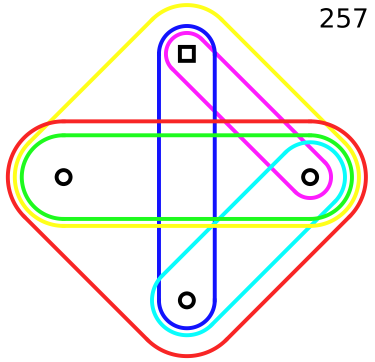

252

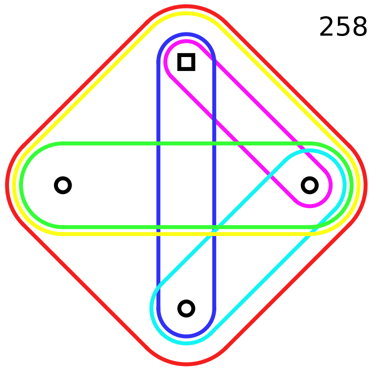

253

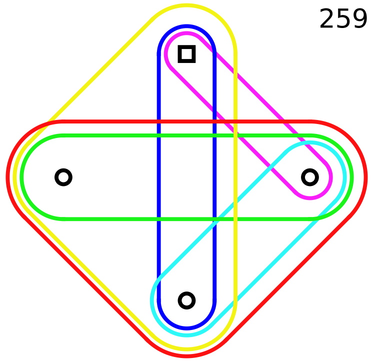

254

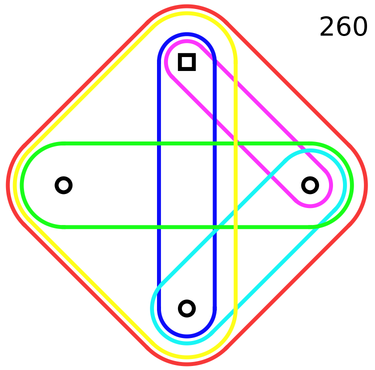

255

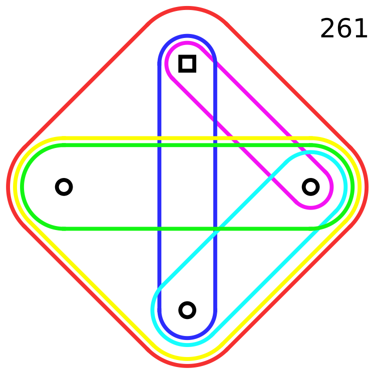

256

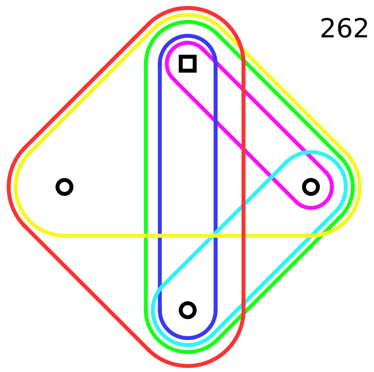

257

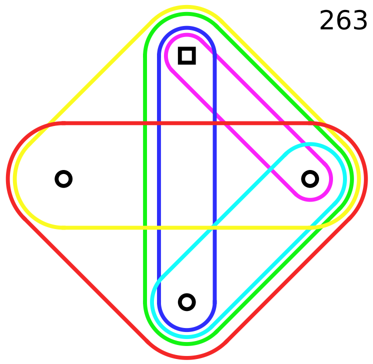

258

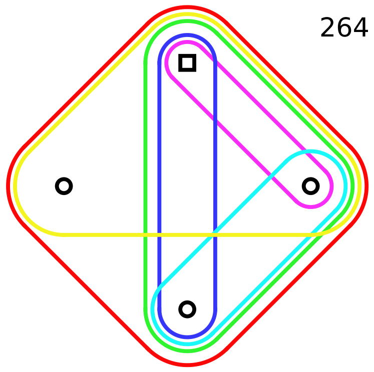

259

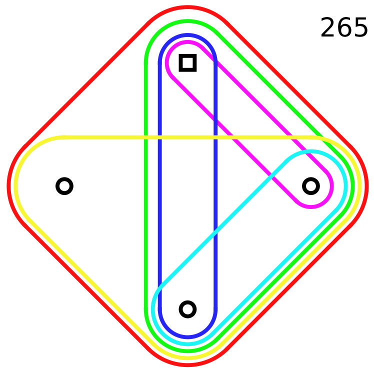

260

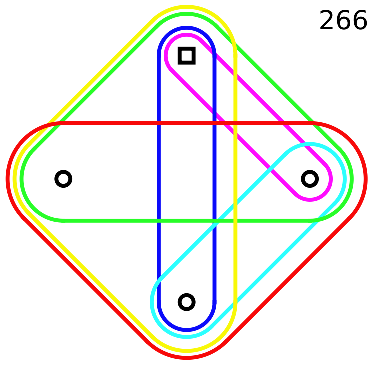

261

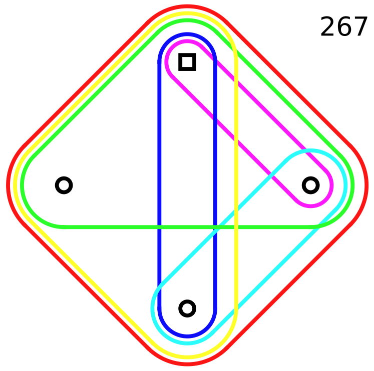

262

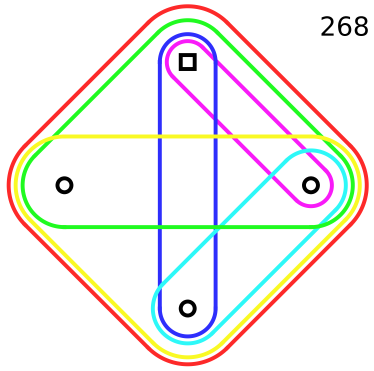

263

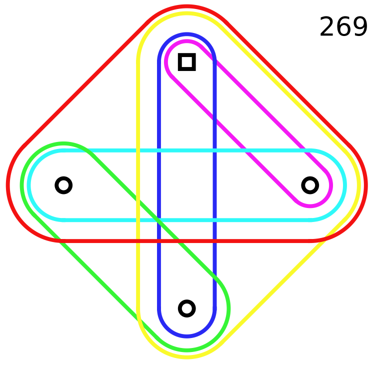

264

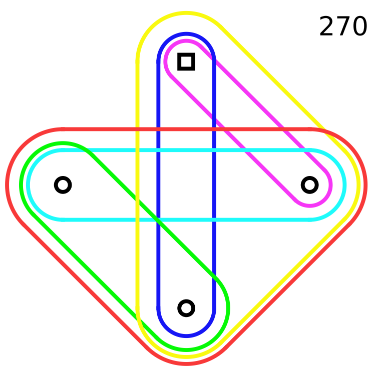

265

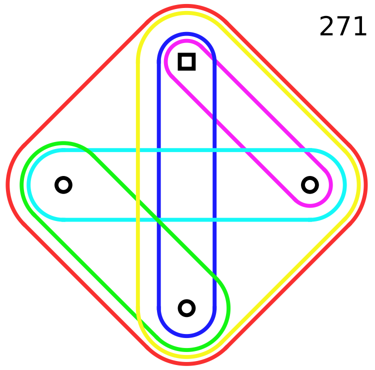

266

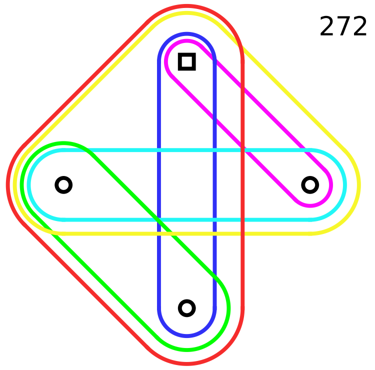

267

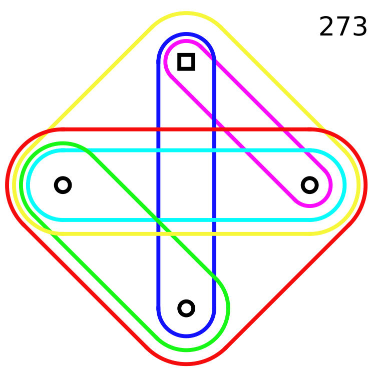

268

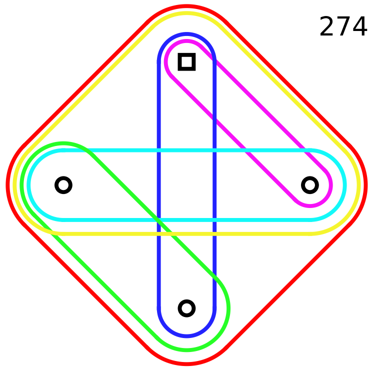

269

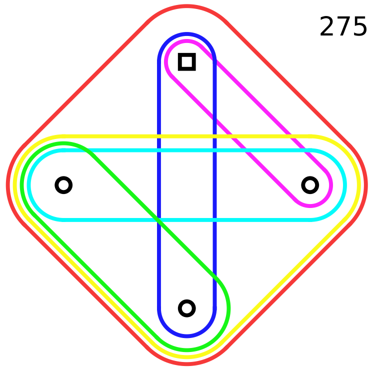

270

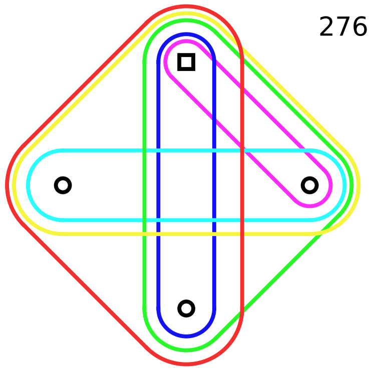

271

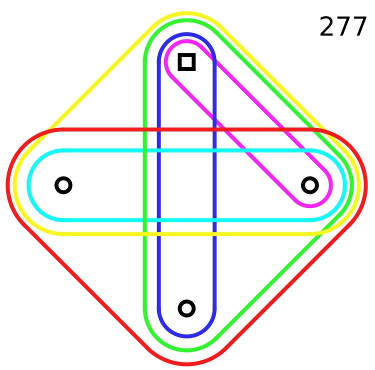

272

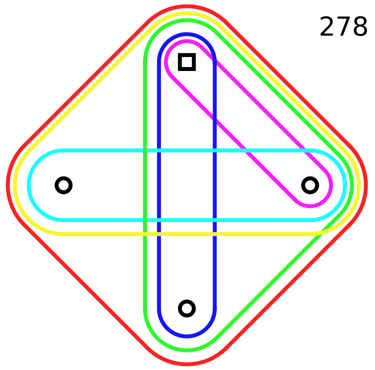

273

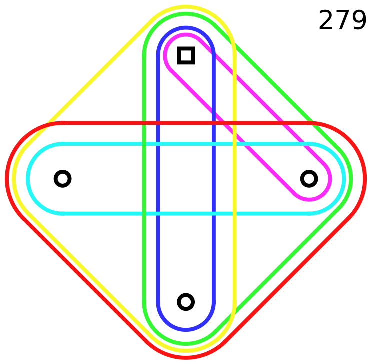

274

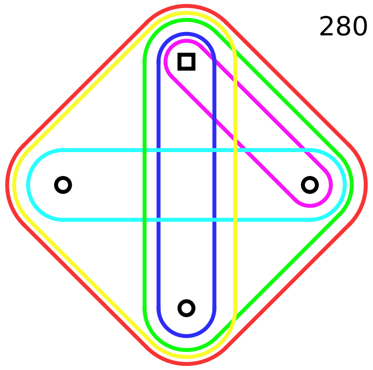

275

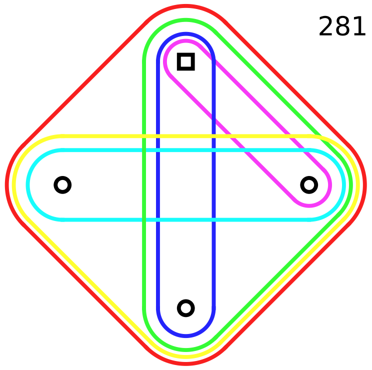

276

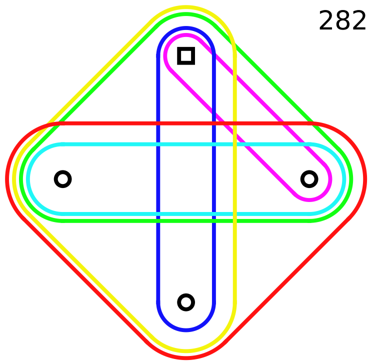

277

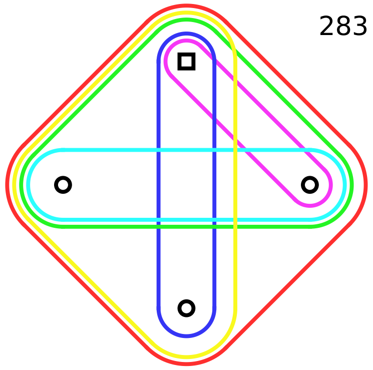

278

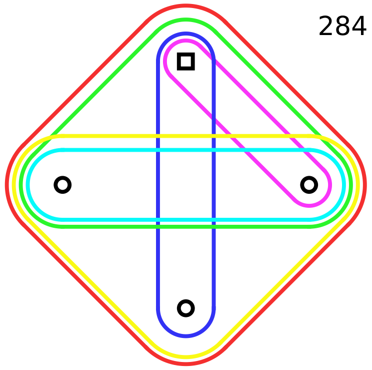

279

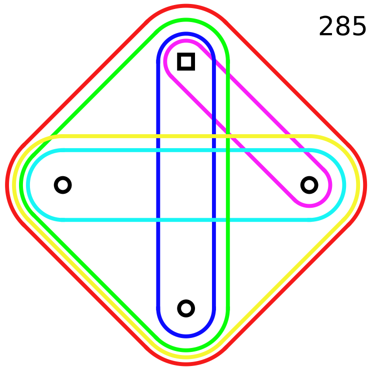

280

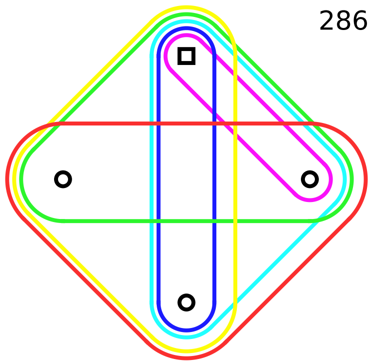

281

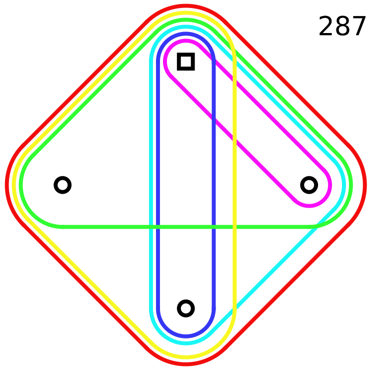

282

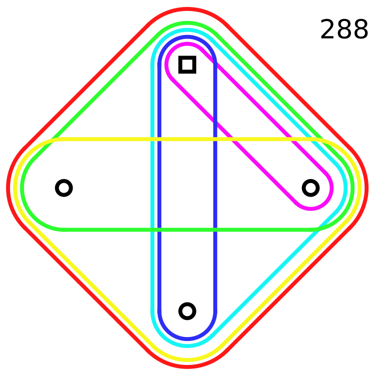

283

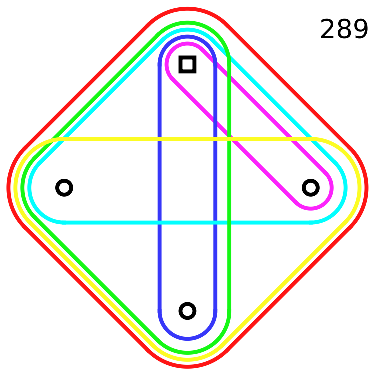

284

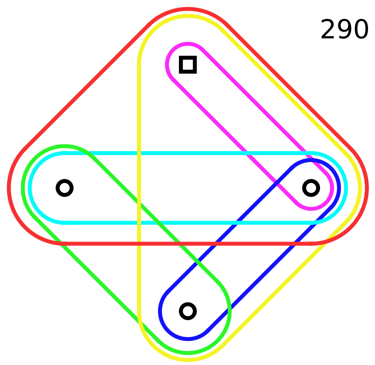

285

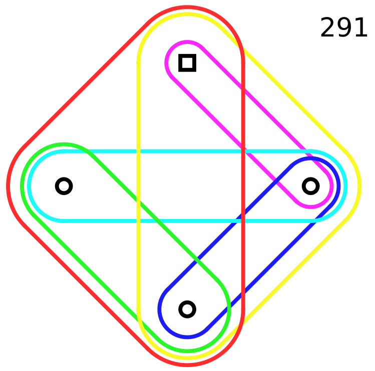

286

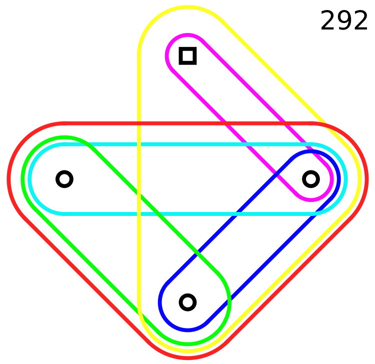

287

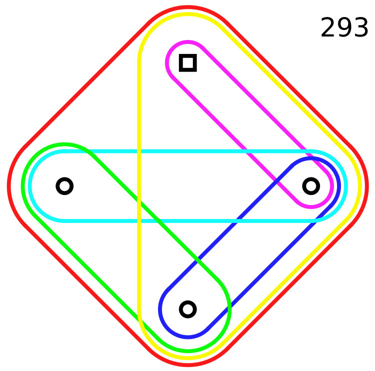

288

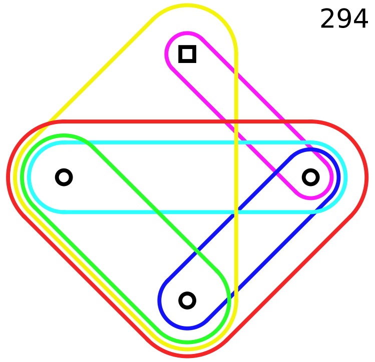

289

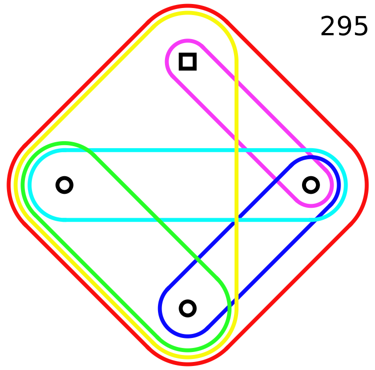

290

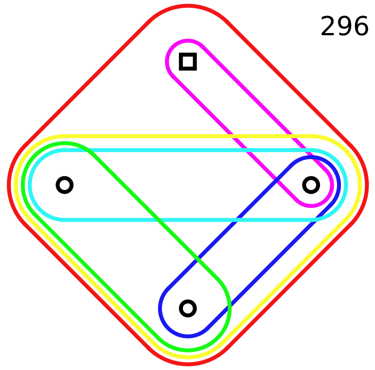

291

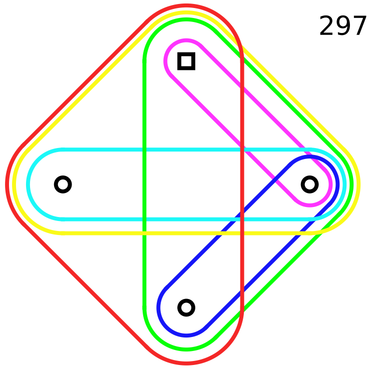

292

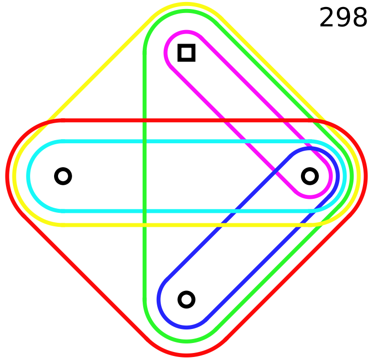

293

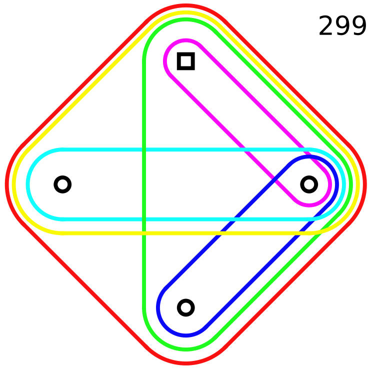

294

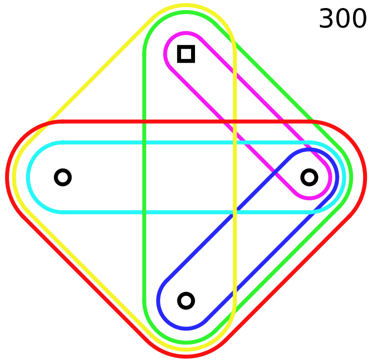

295

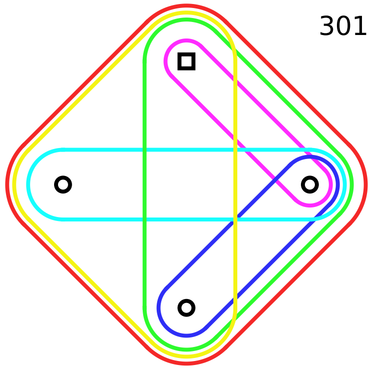

296

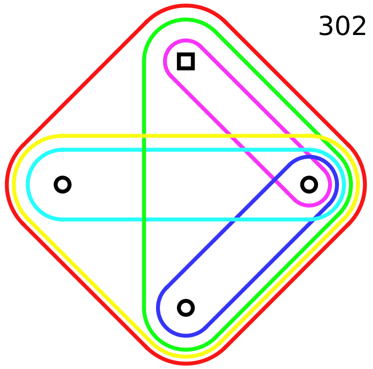

297

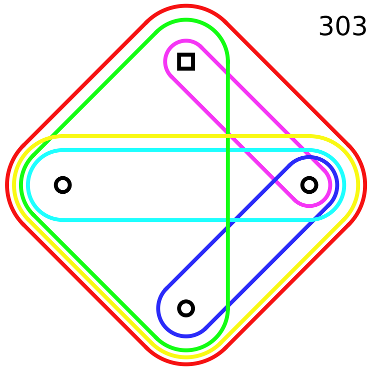

298

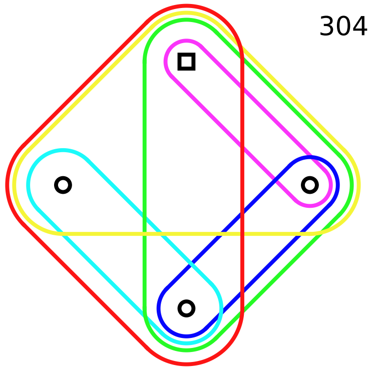

299

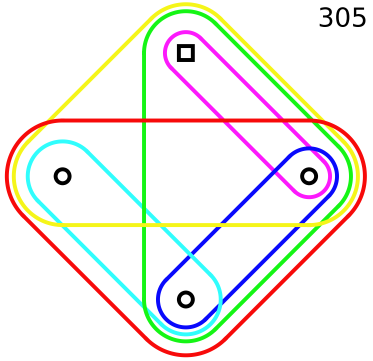

300

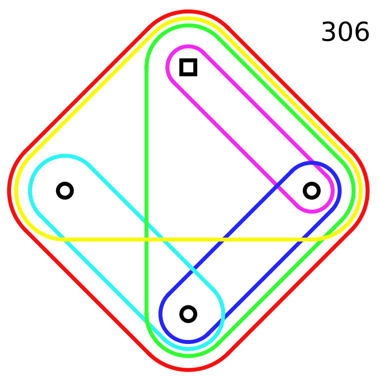

301

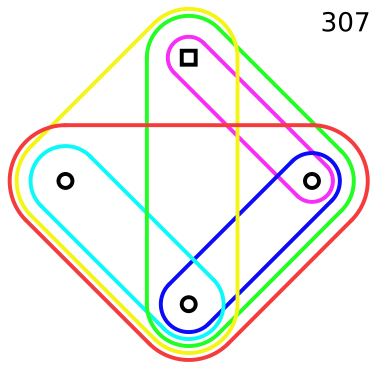

302

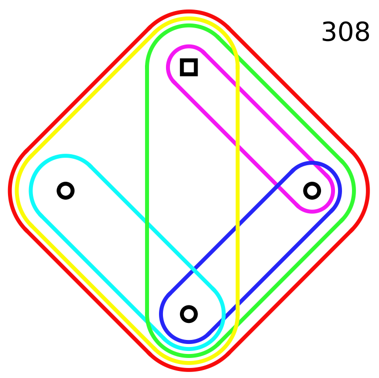

303

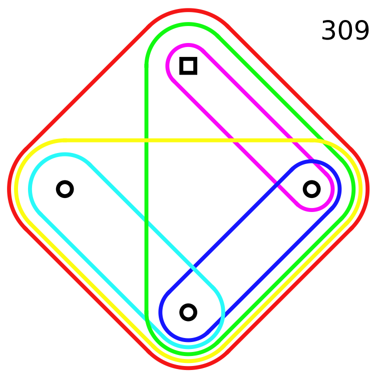

304

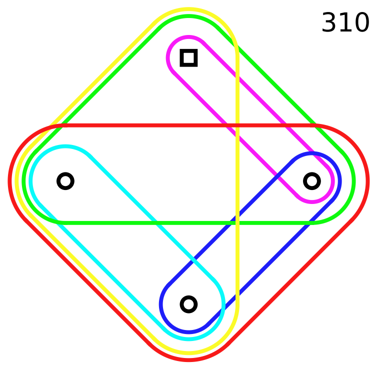

305

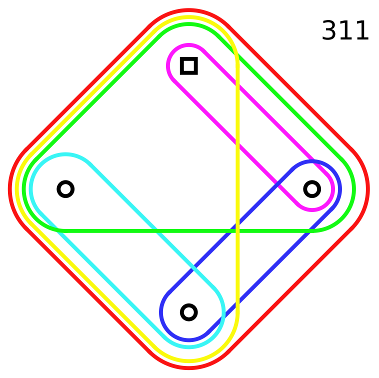

306

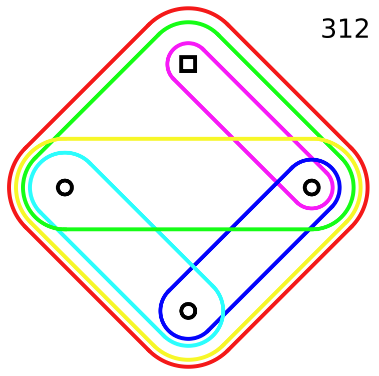

307

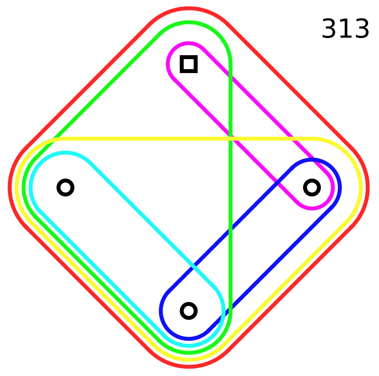

308

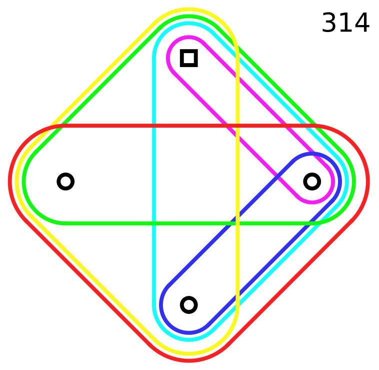

309

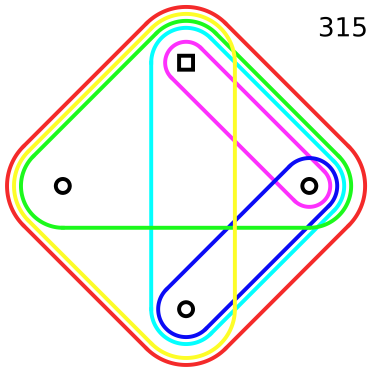

310

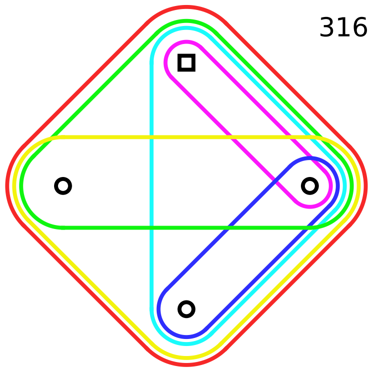

311

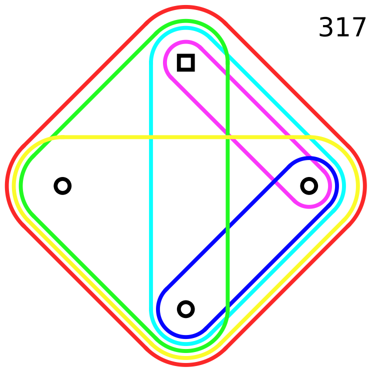

312

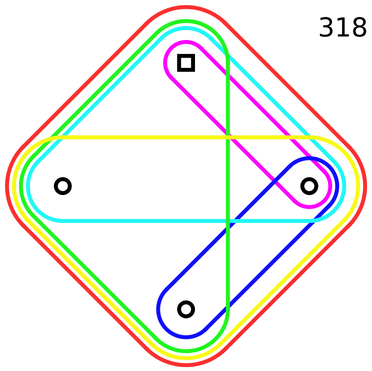

313

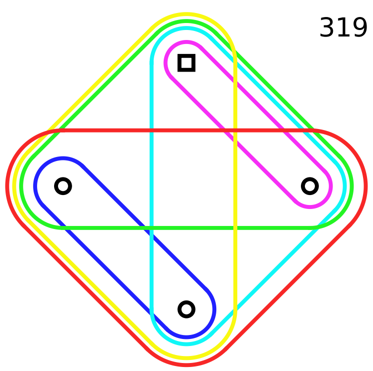

314

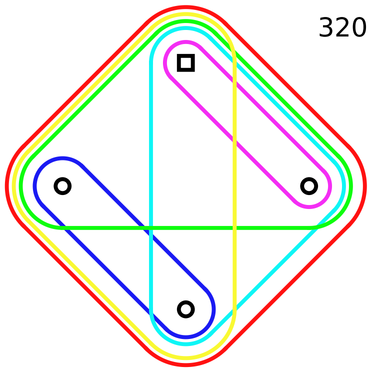

315

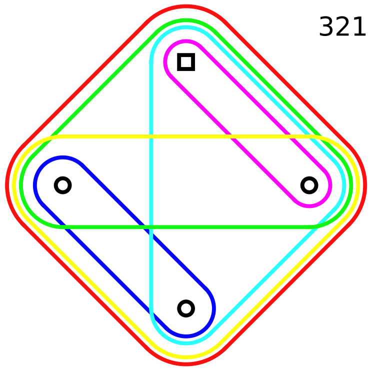

316

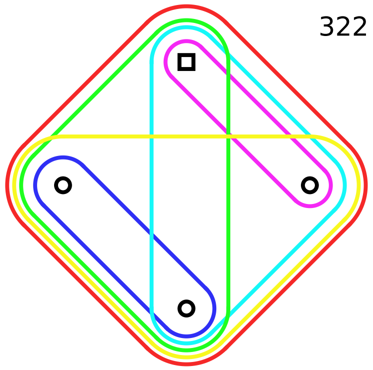

317

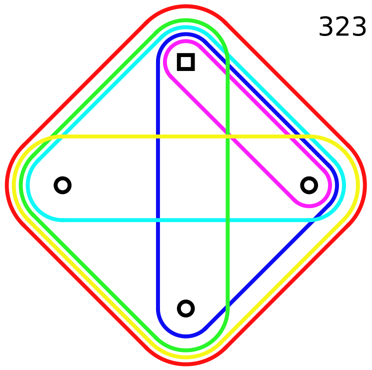

318

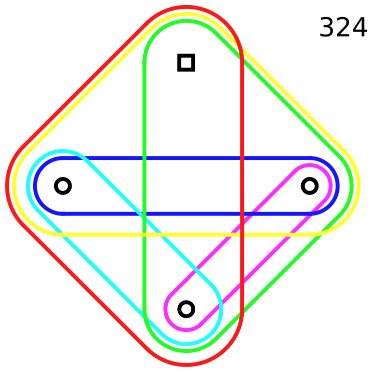

319

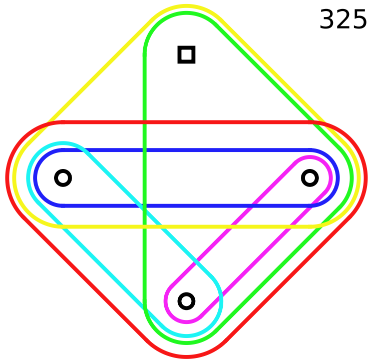

320

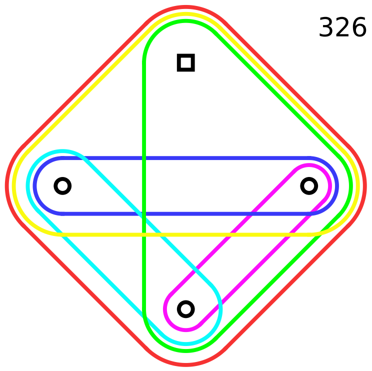

321

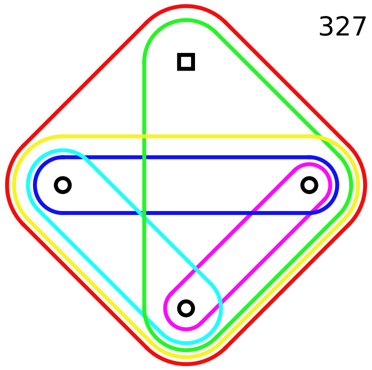

322

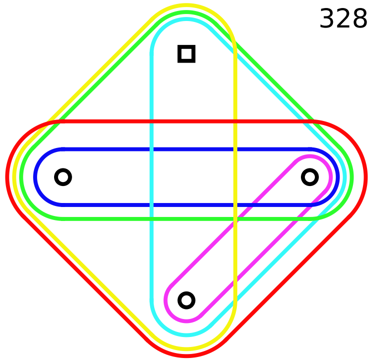

323

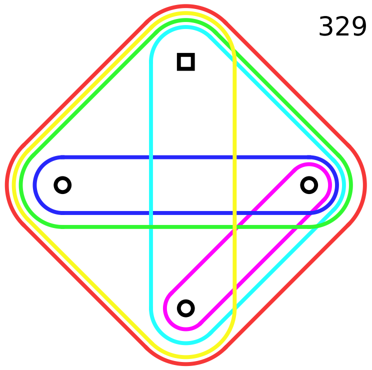

324

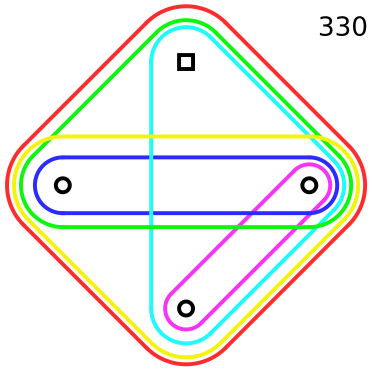

325

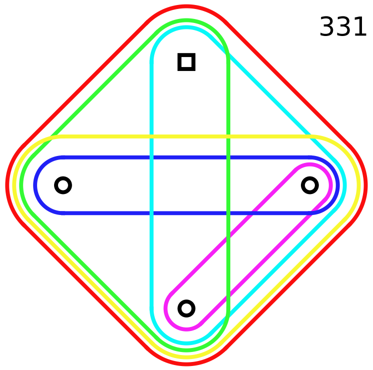

326

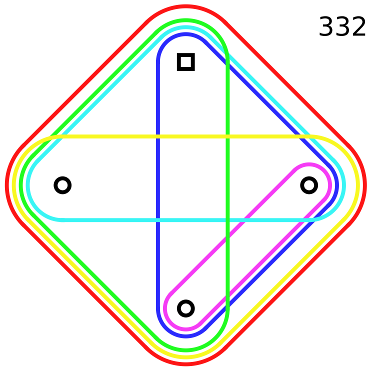

327

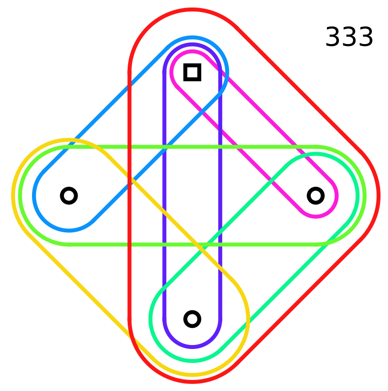

328

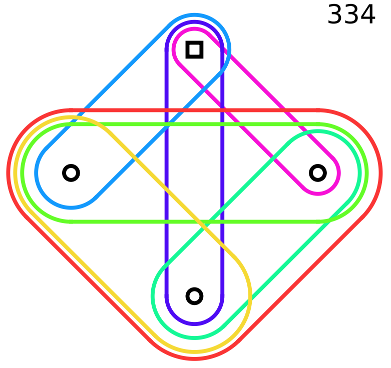

329

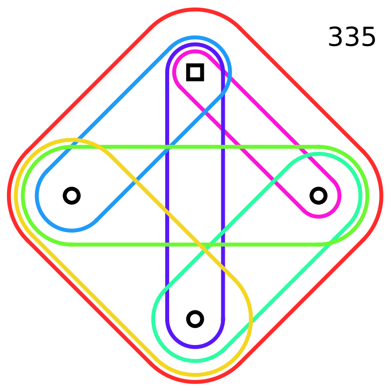

330

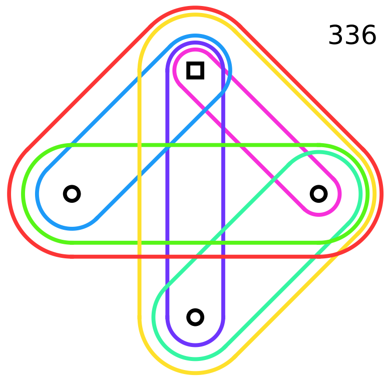

331

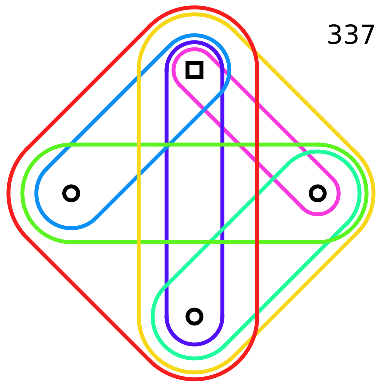

332

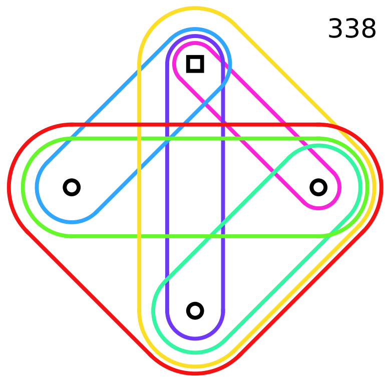

333

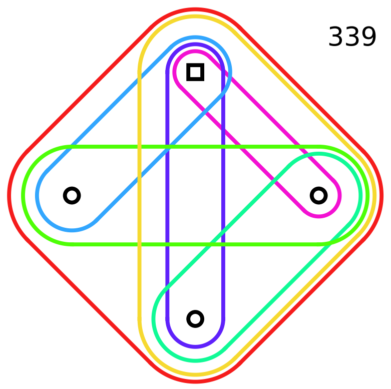

334

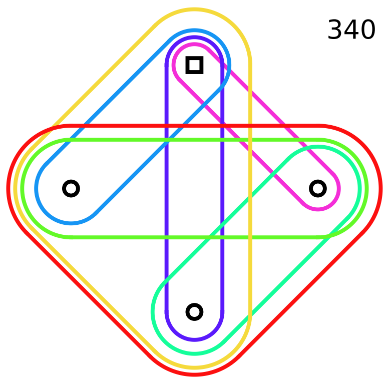

335

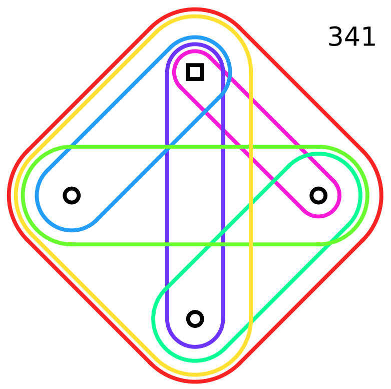

336

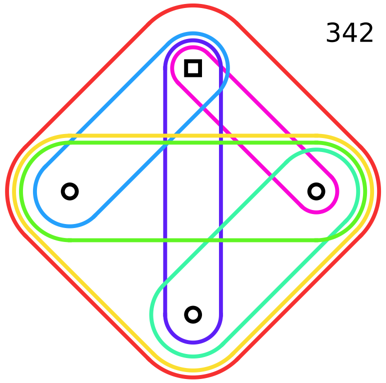

337

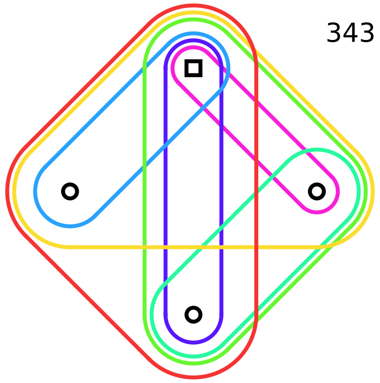

338

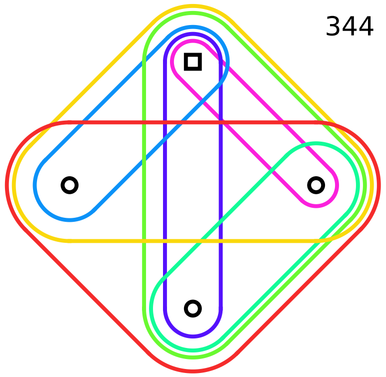

339

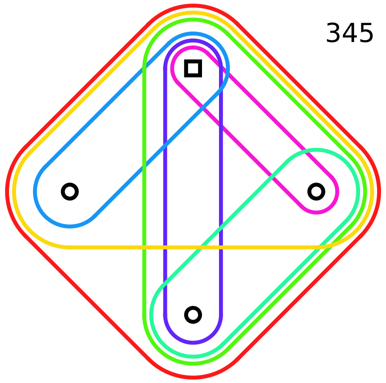

340

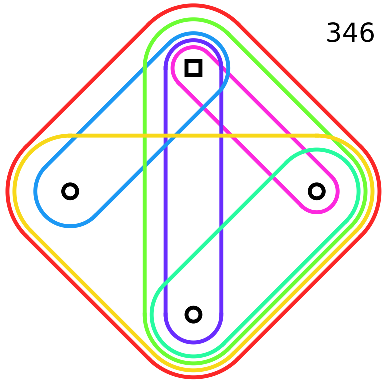

341

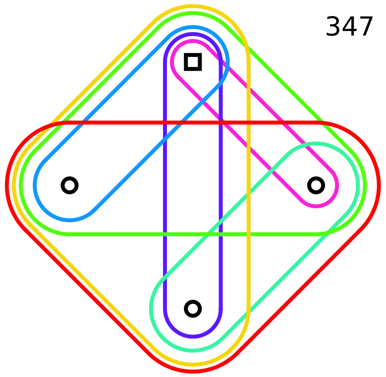

342

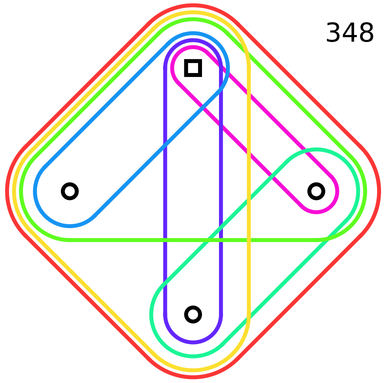

343

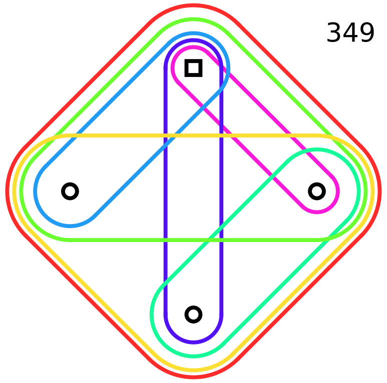

344

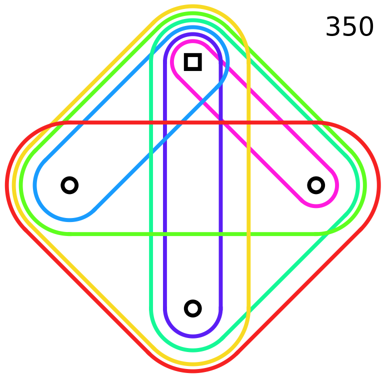

345

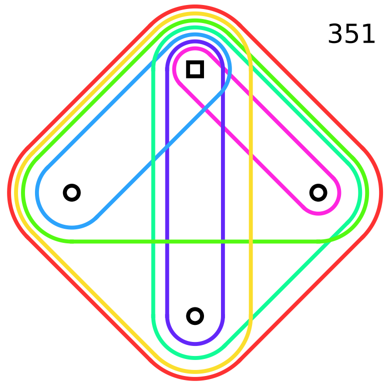

346

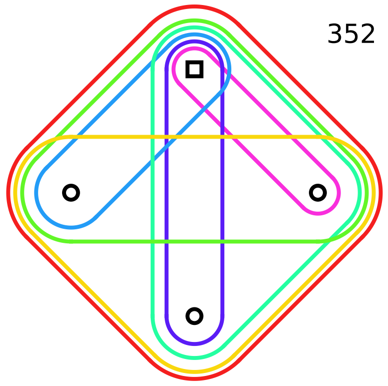

347

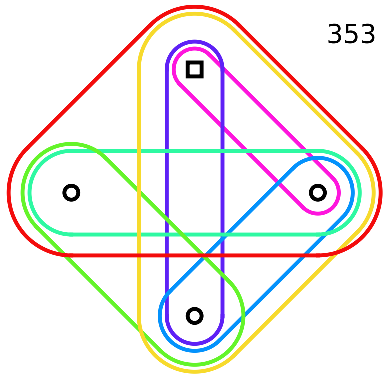

348

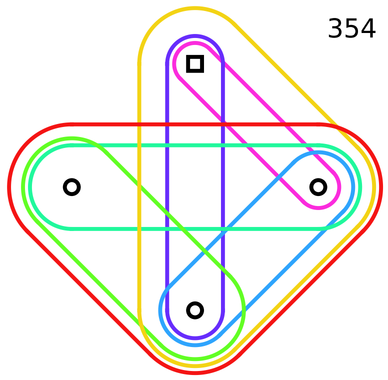

349

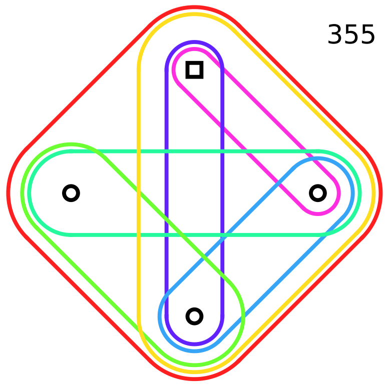

350

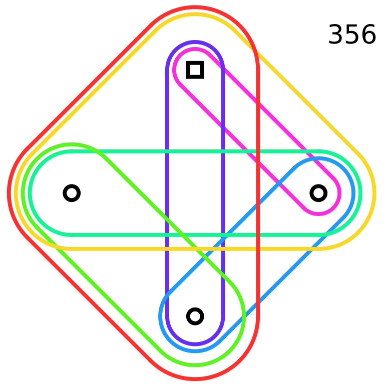

351

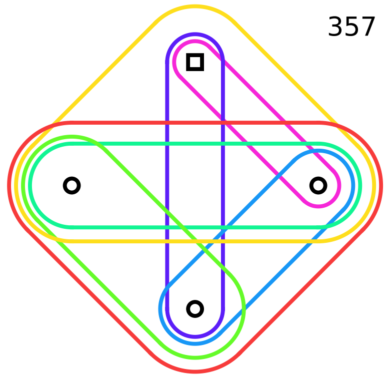

352

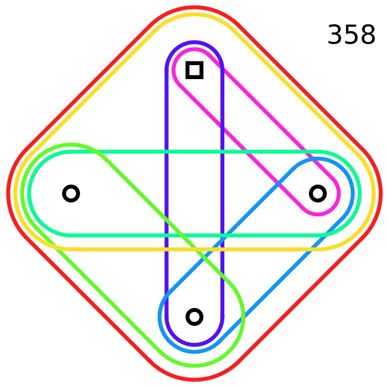

353

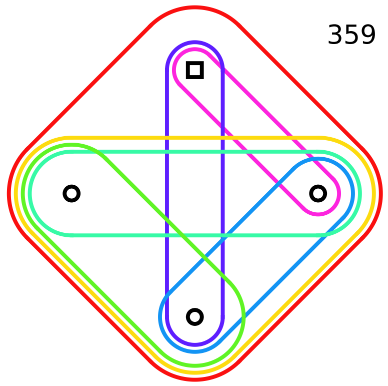

354

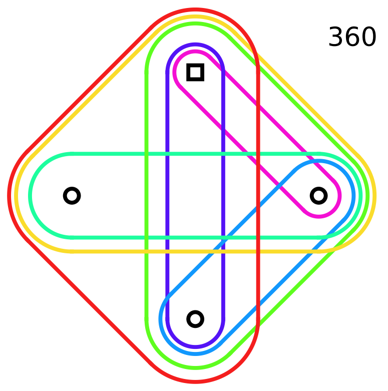

355

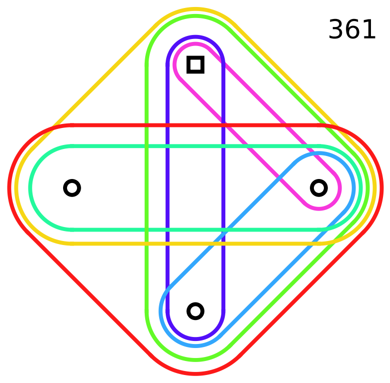

356

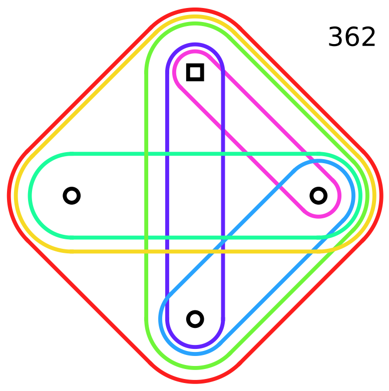

357

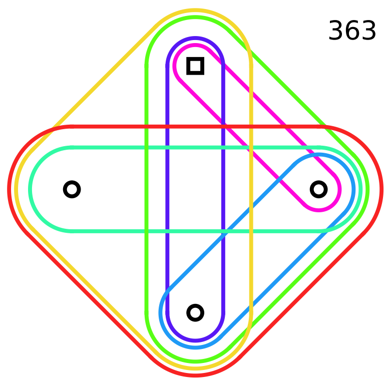

358

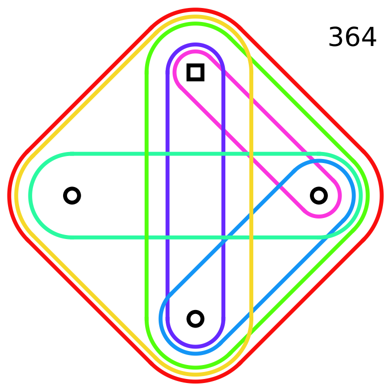

359

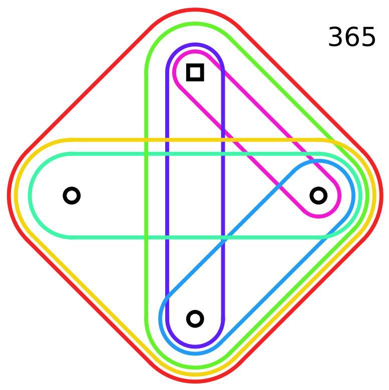

360

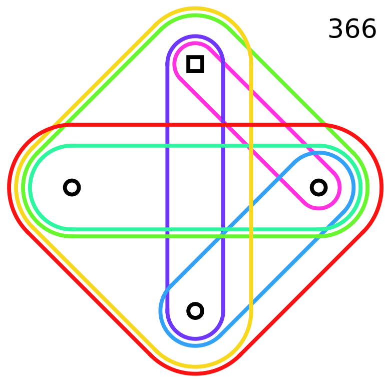

361

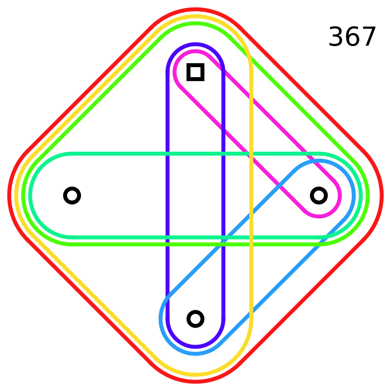

362

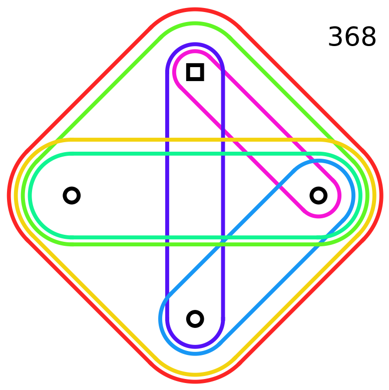

363

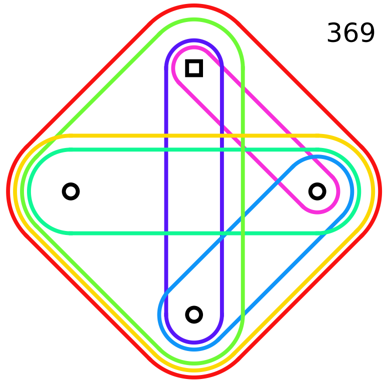

364

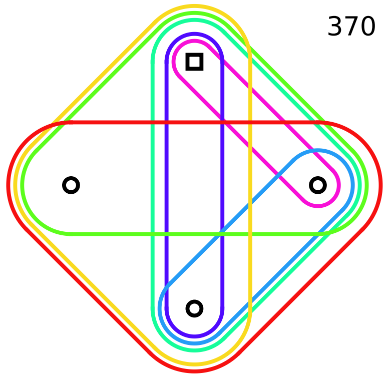

365

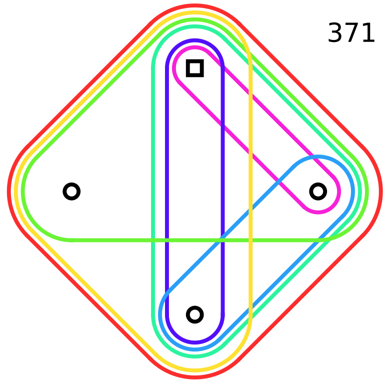

366

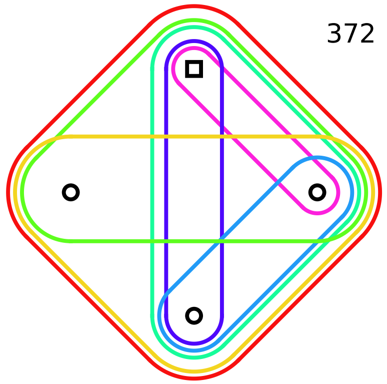

367

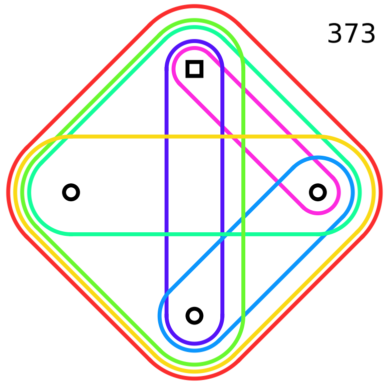

368

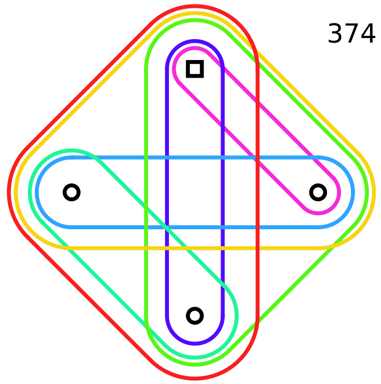

369

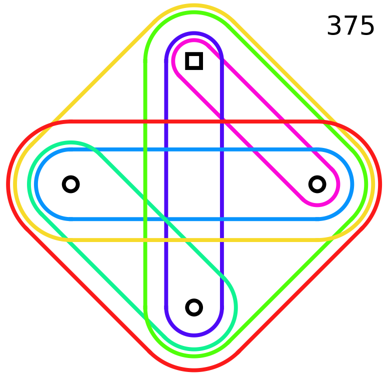

370

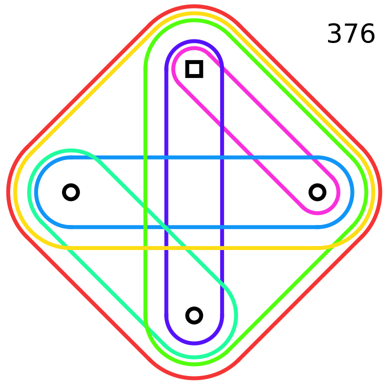

371

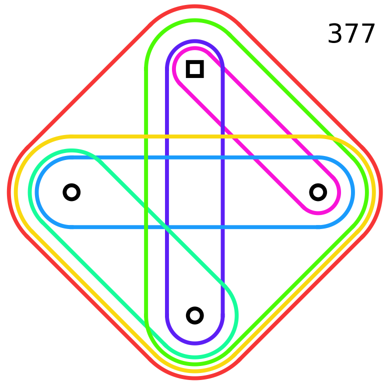

372

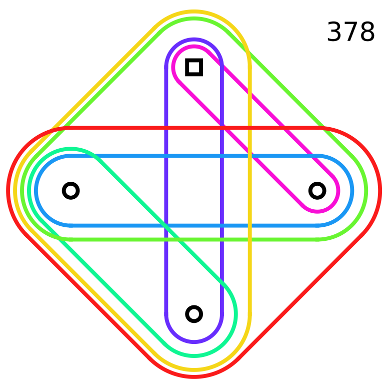

373

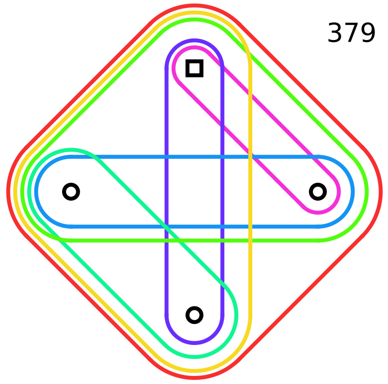

374

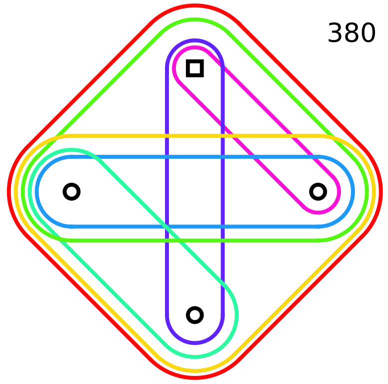

375

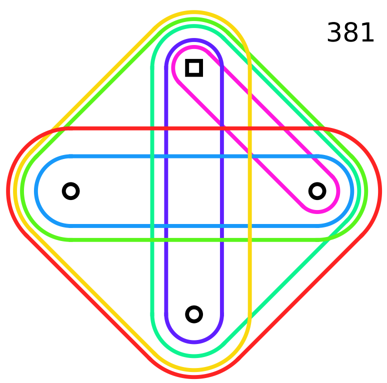

376

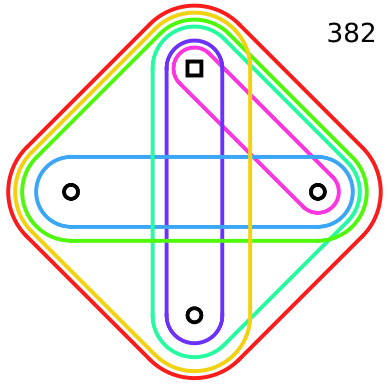

377

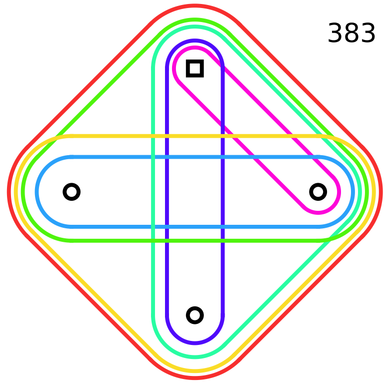

378

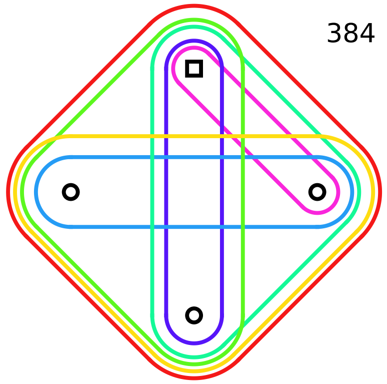

379

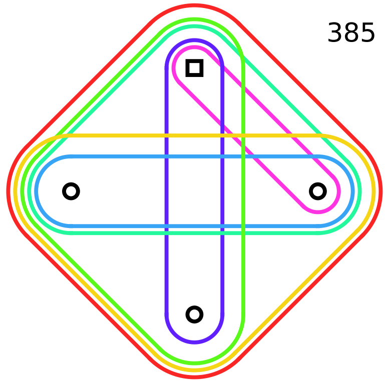

380

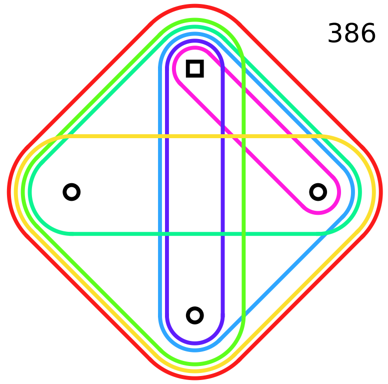

381

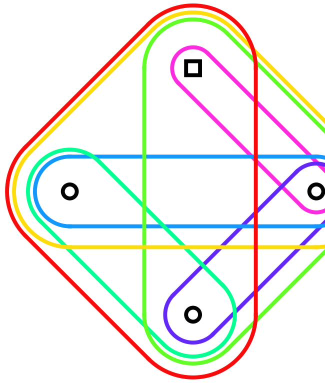

382

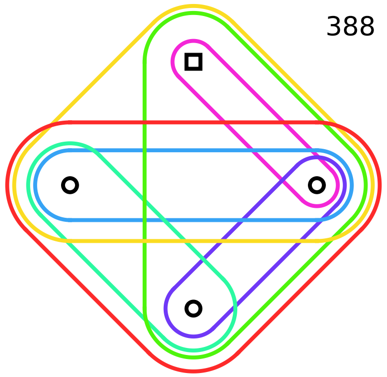

383

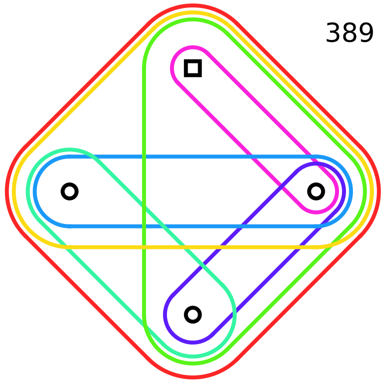

384

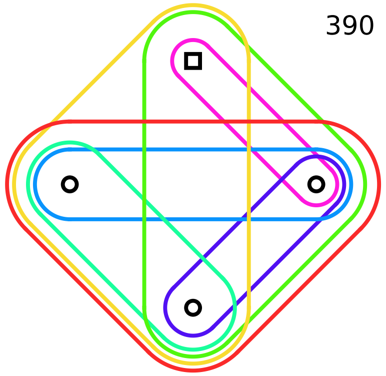

385

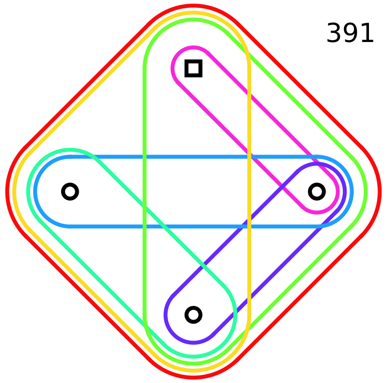

386

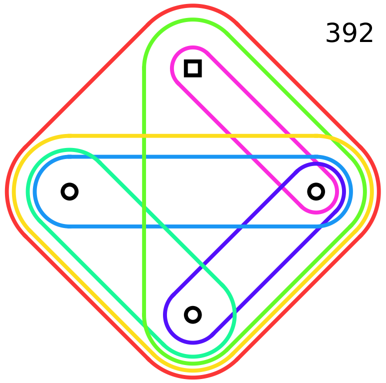

387

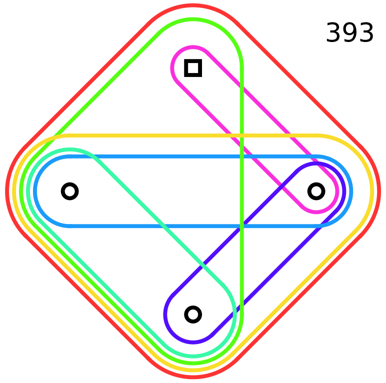

388

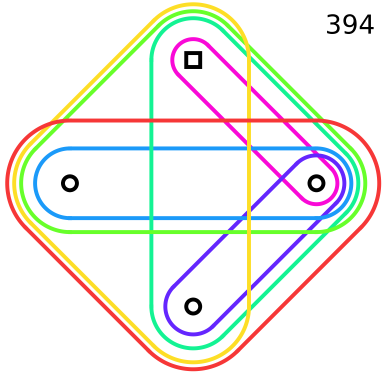

389

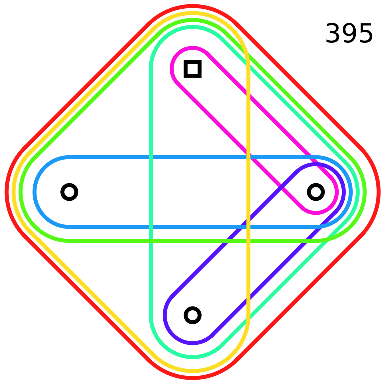

390

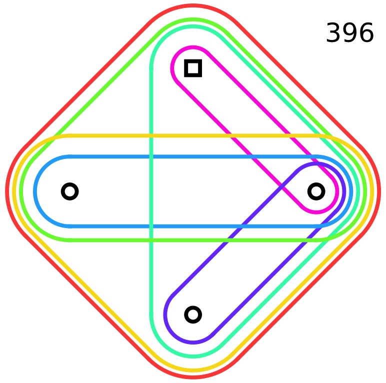

391

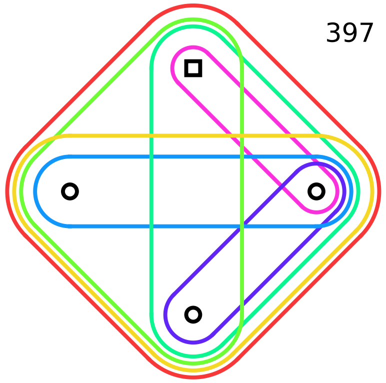

392

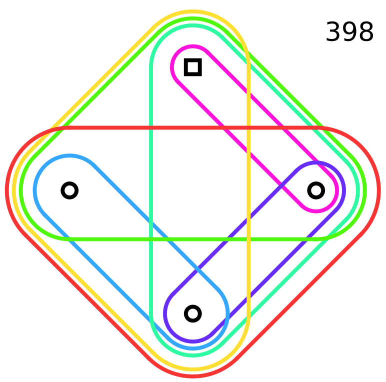

393

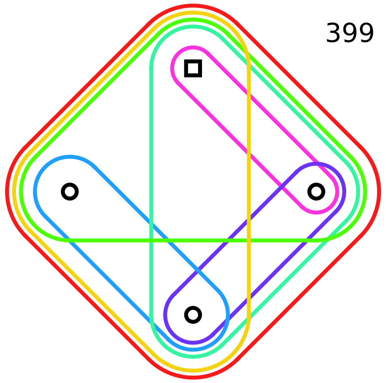

394

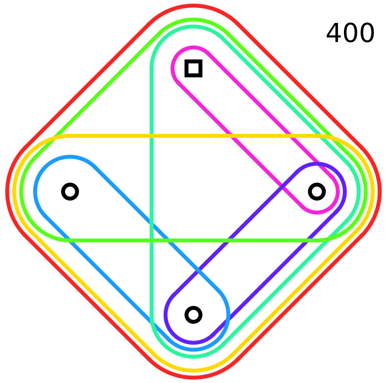

395

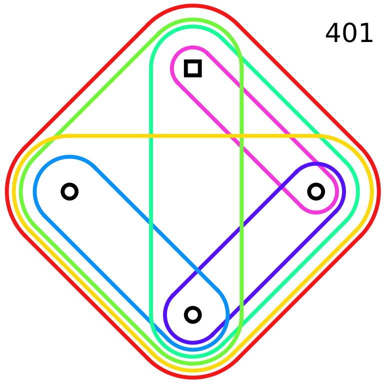

396

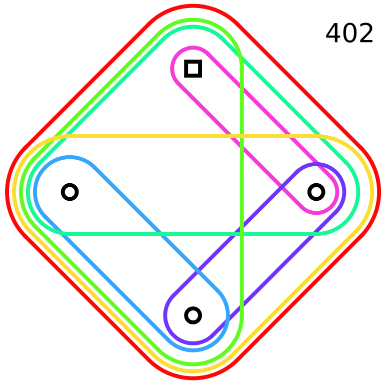

397

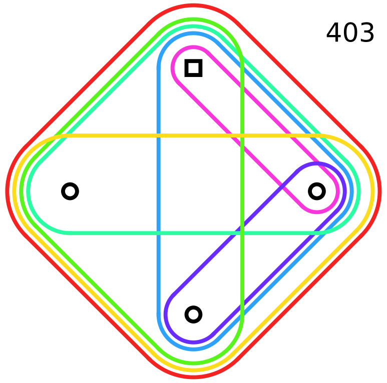

398

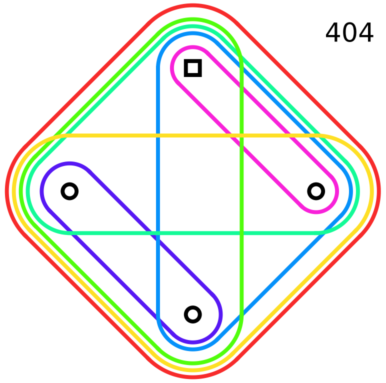

399

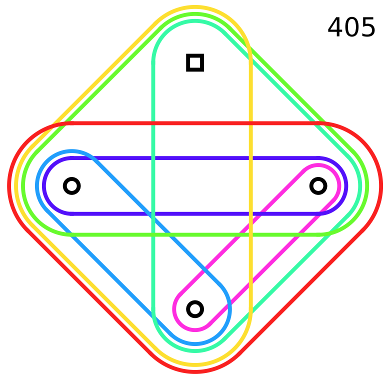

400

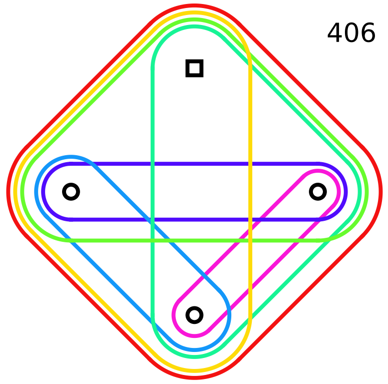

401

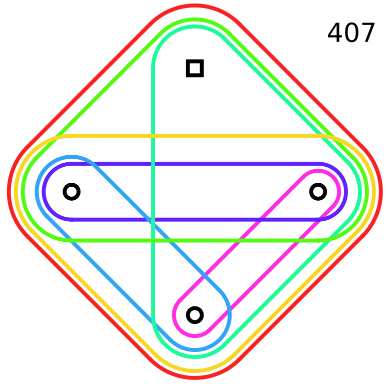

402

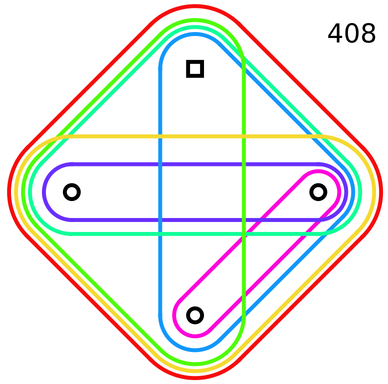

403

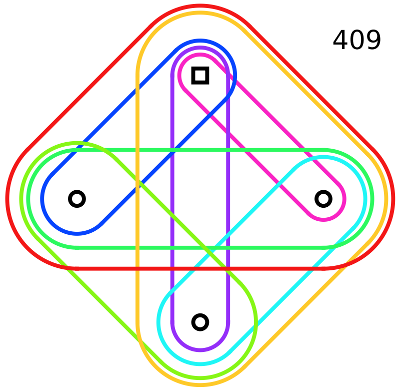

404

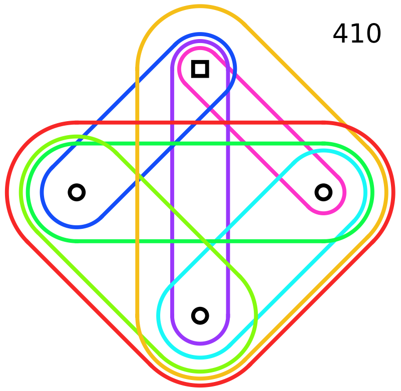

405

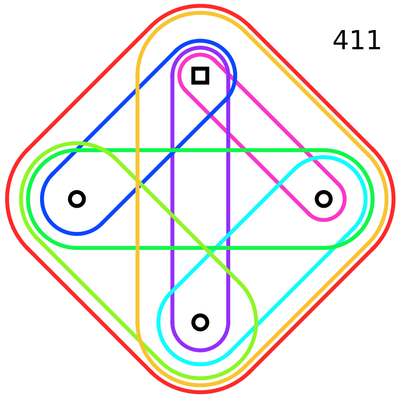

406

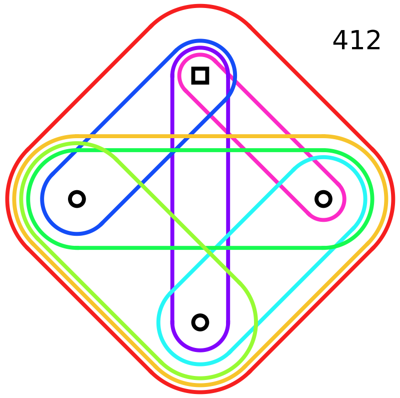

407

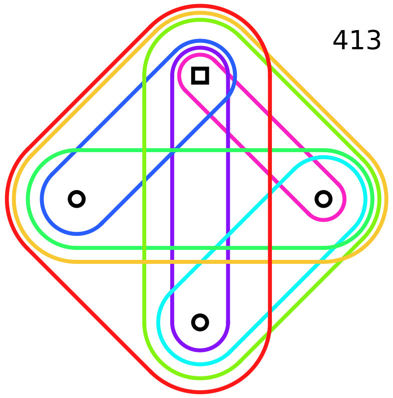

408

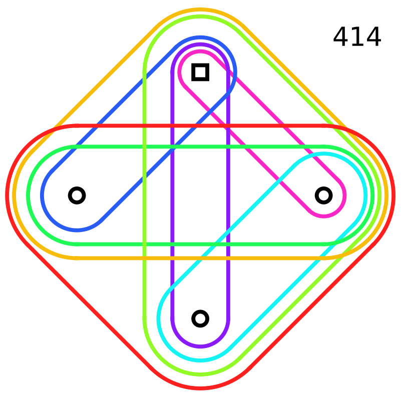

409

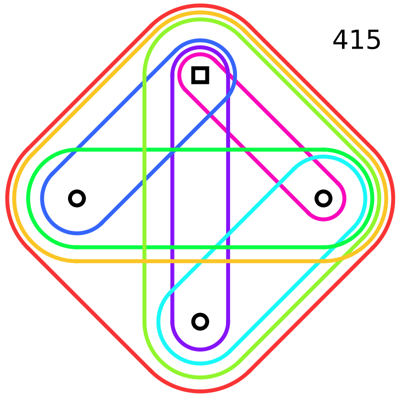

410

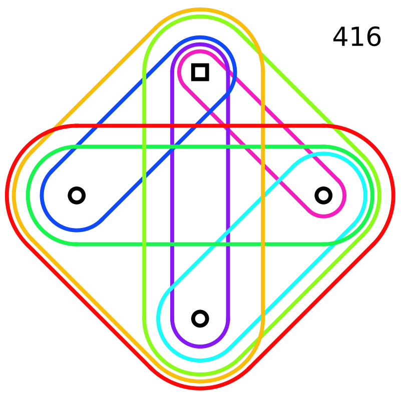

411

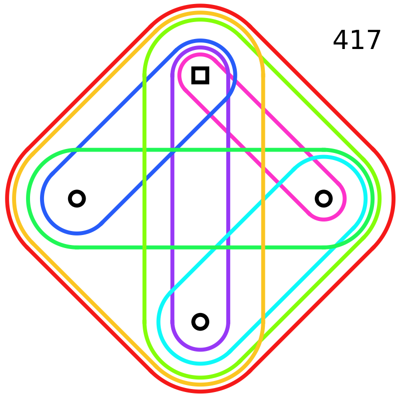

412

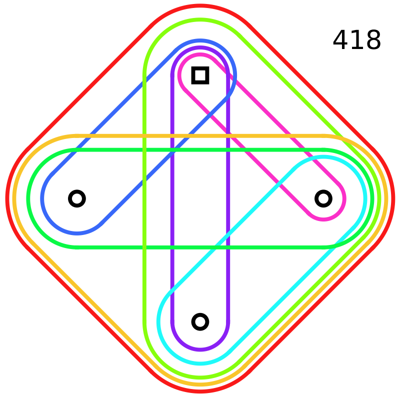

413

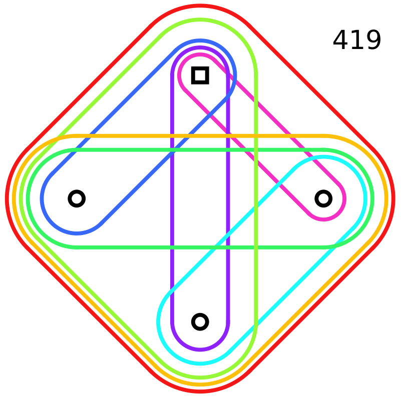

414

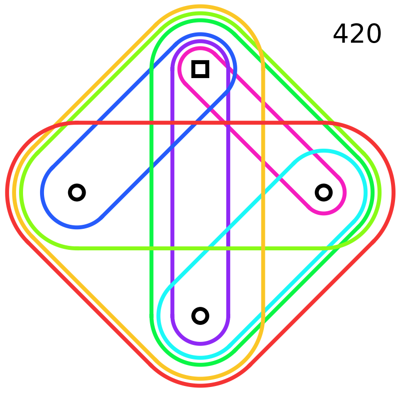

415

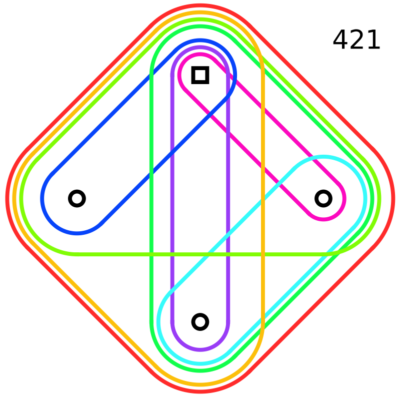

416

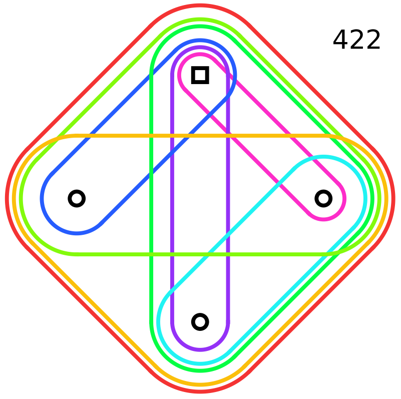

417

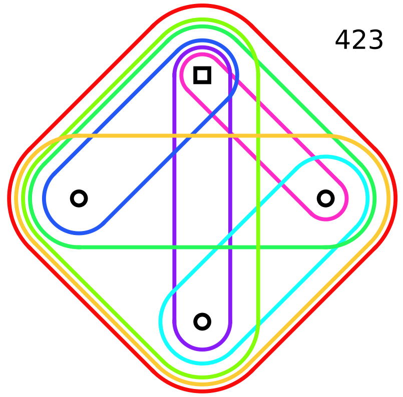

418

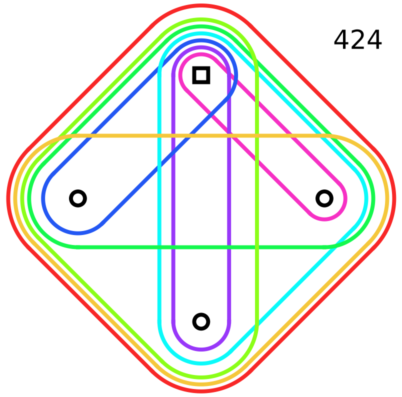

419

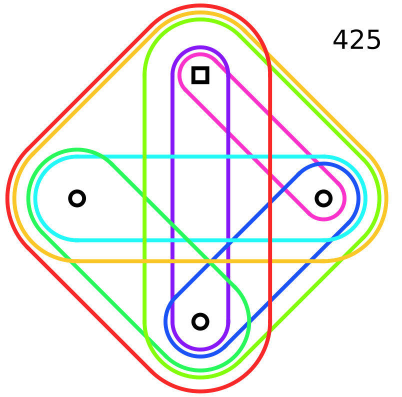

420

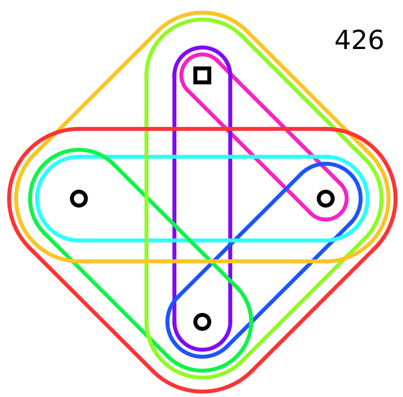

421

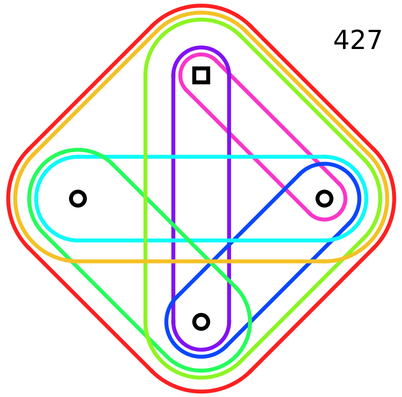

422

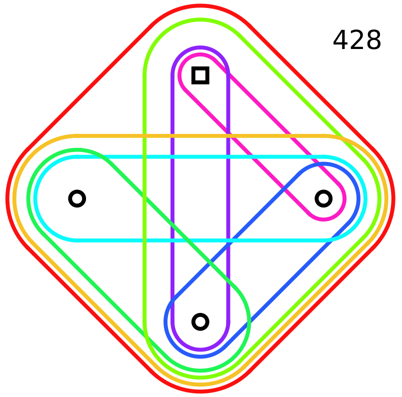

423

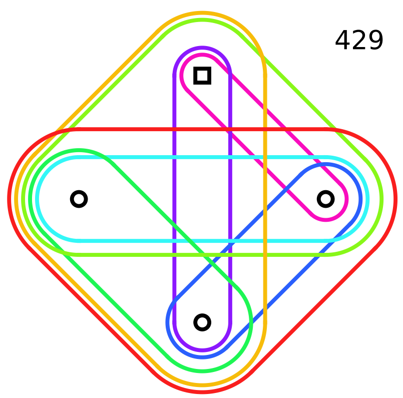

424

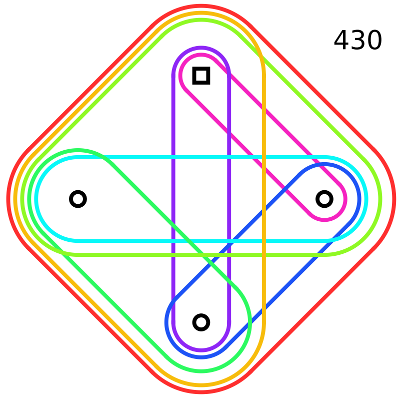

425

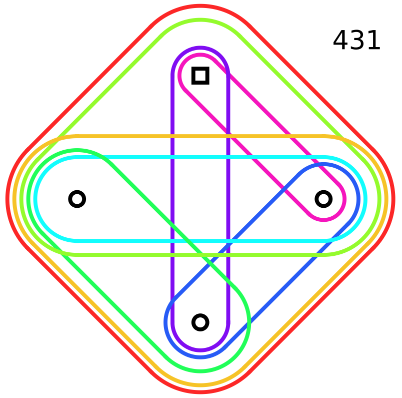

426

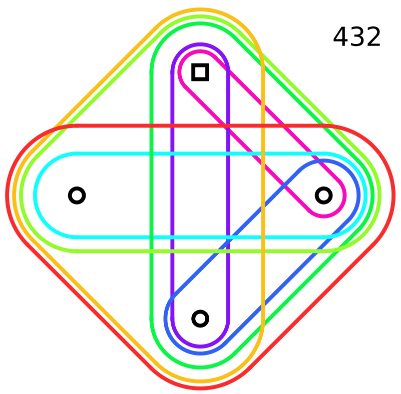

427

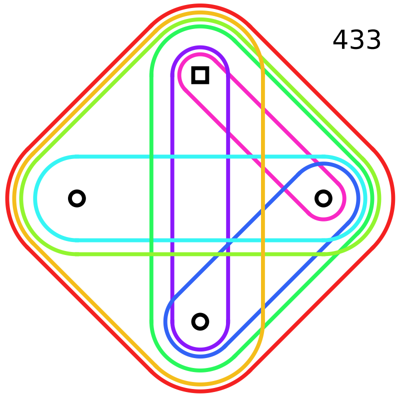

428

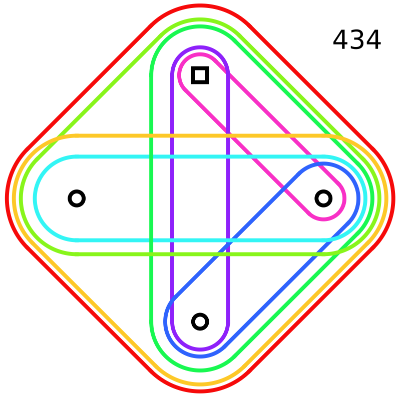

429

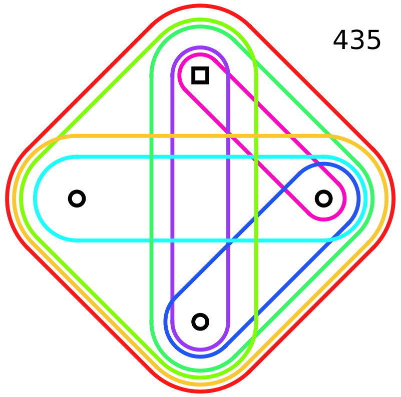

430

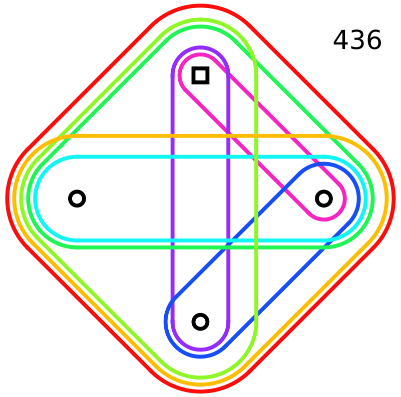

431

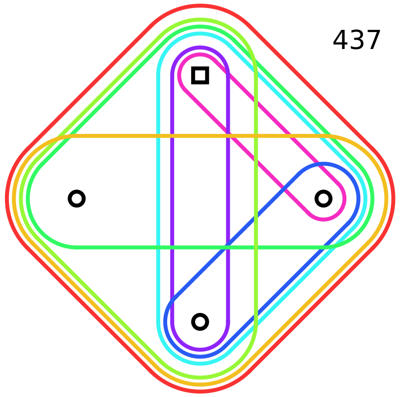

432

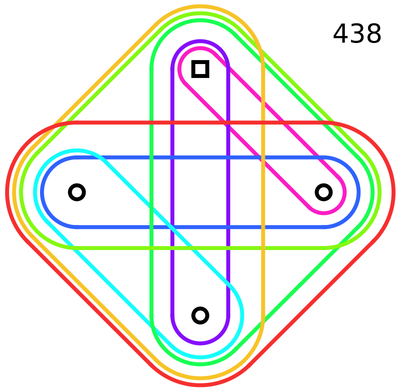

433

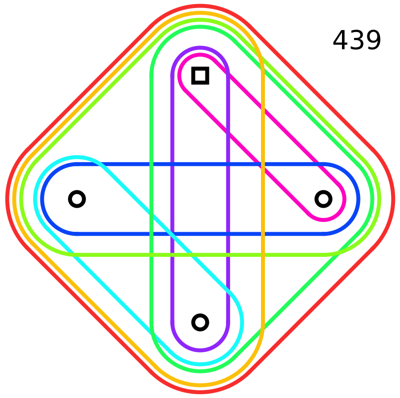

434

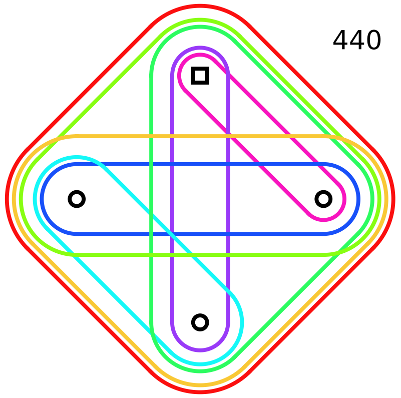

435

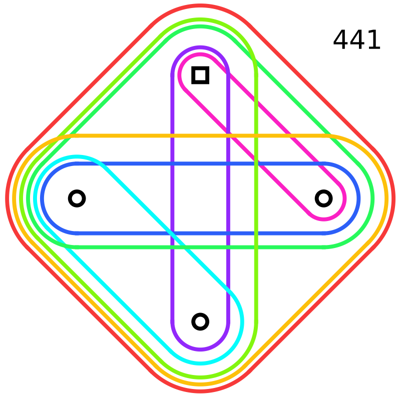

436

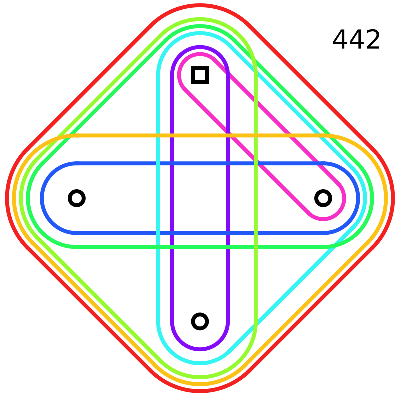

437

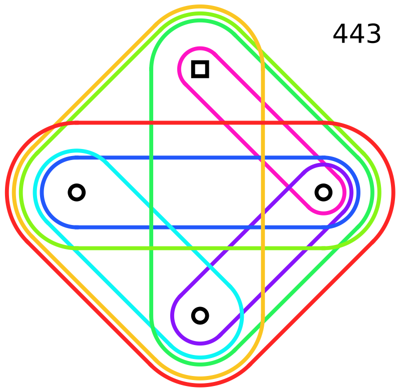

438

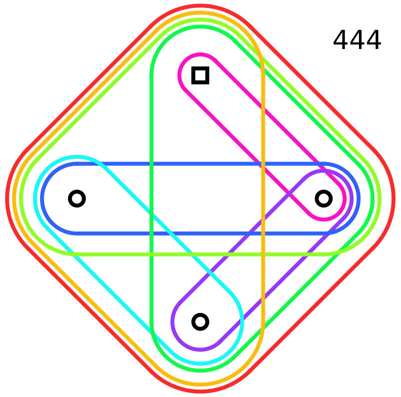

439

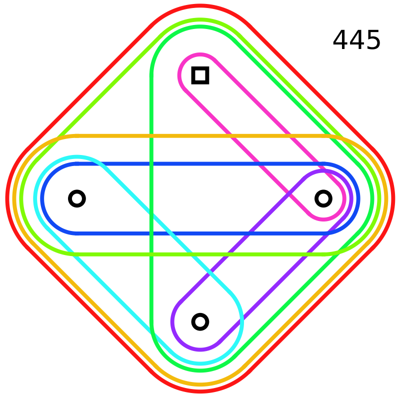

440

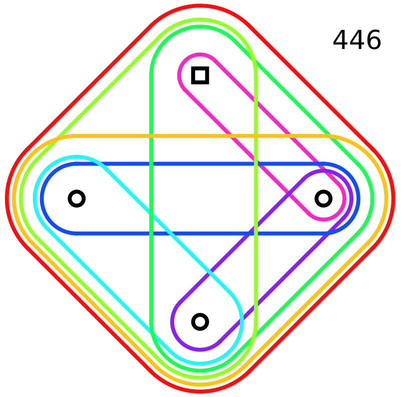

441

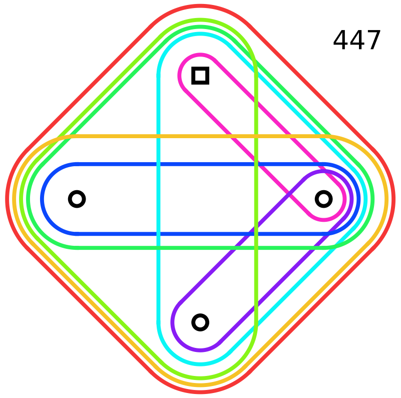

442

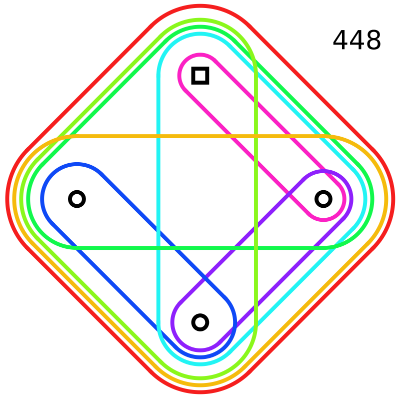

443

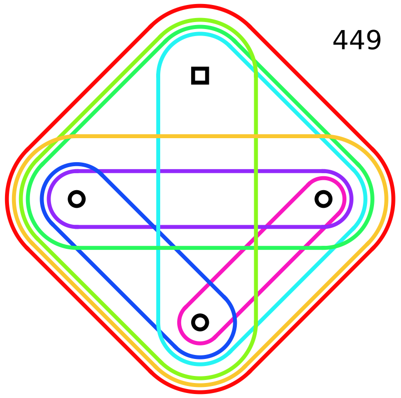

444

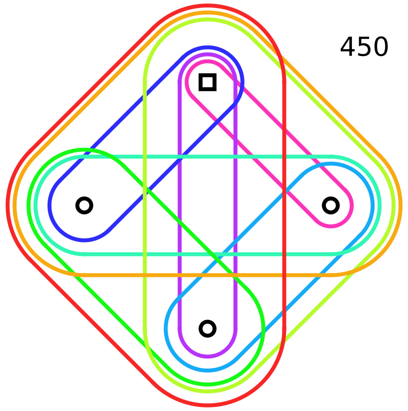

445

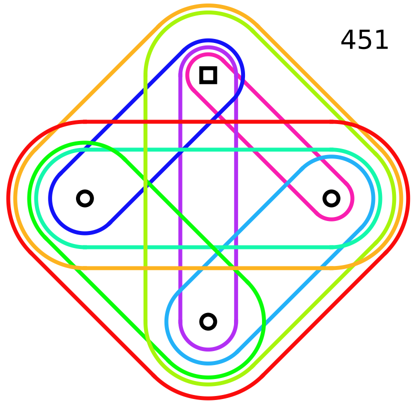

446

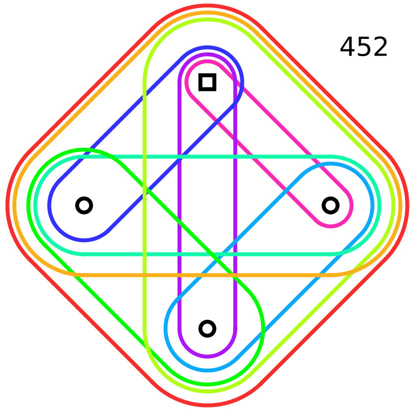

447

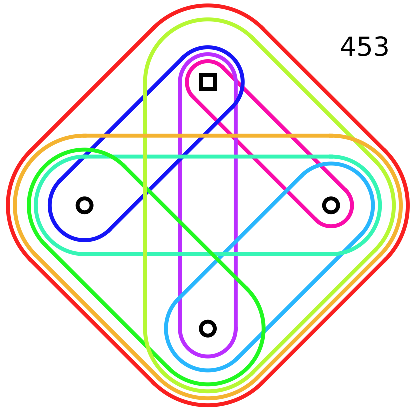

448

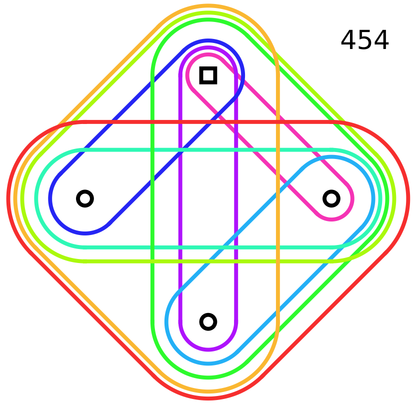

449

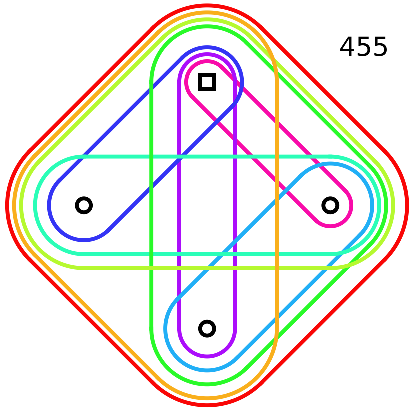

450

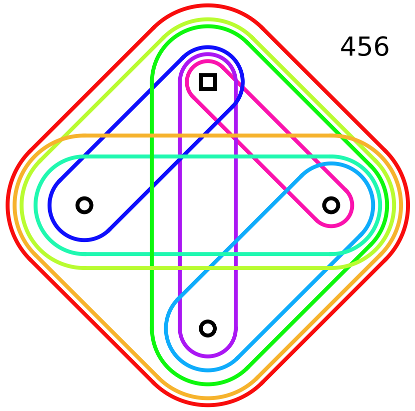

451

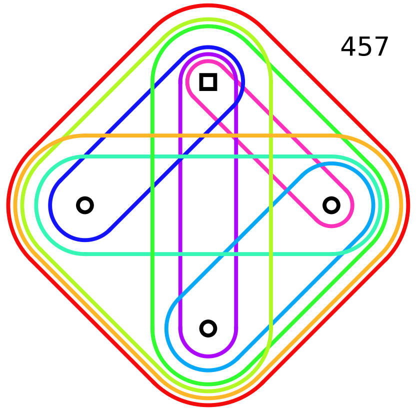

452

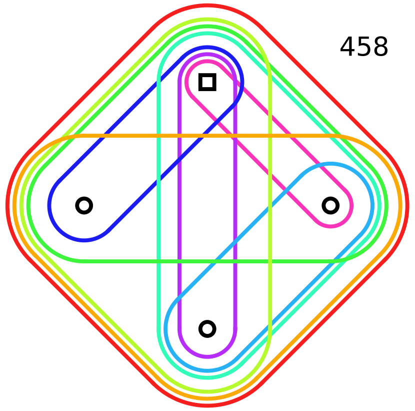

453

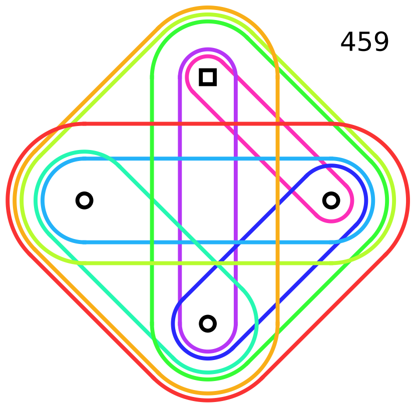

454

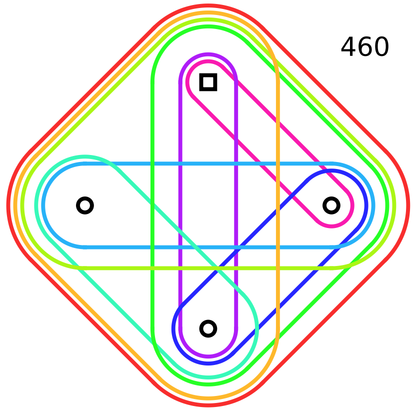

455

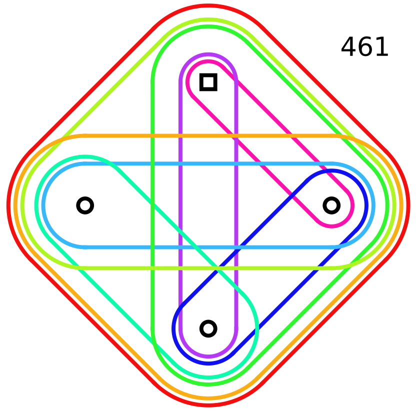

456

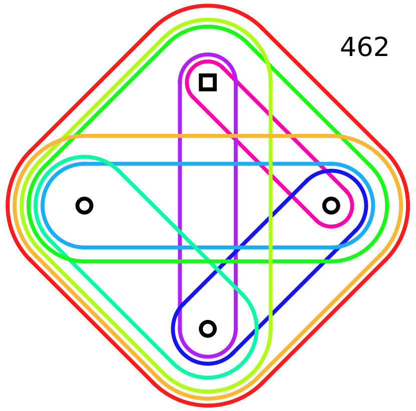

457

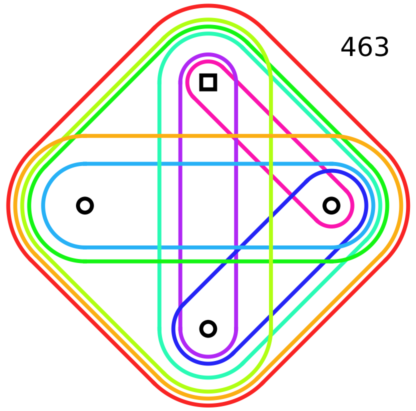

458

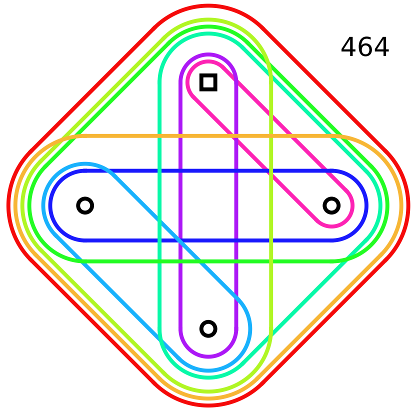

459

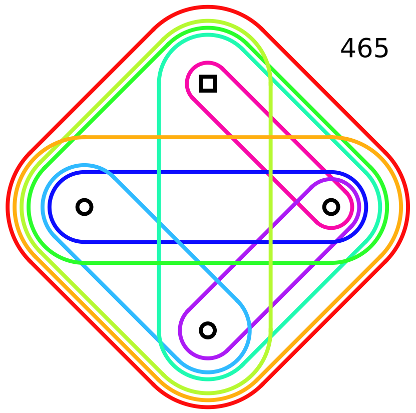

460

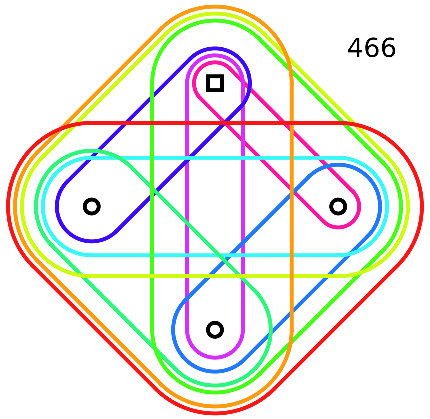

461

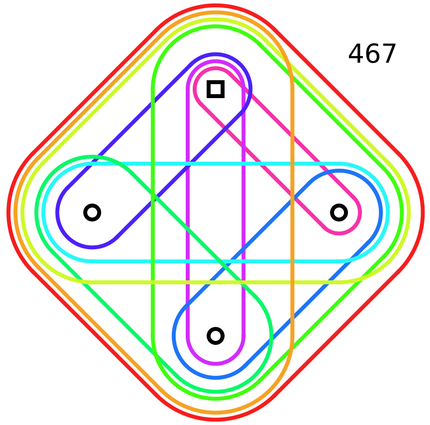

462

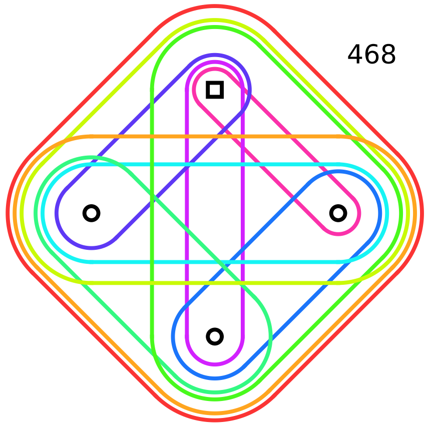

463

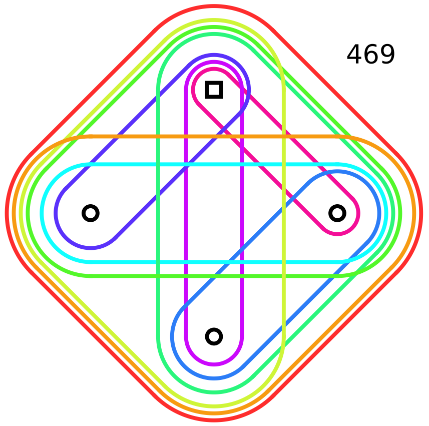

464

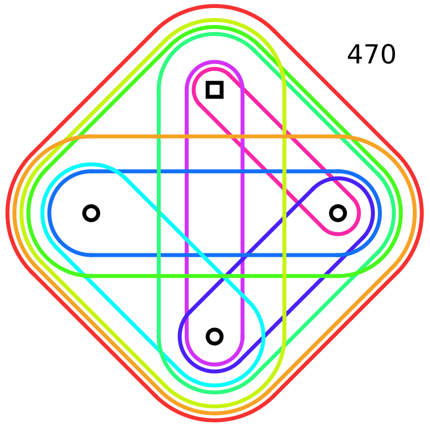

465

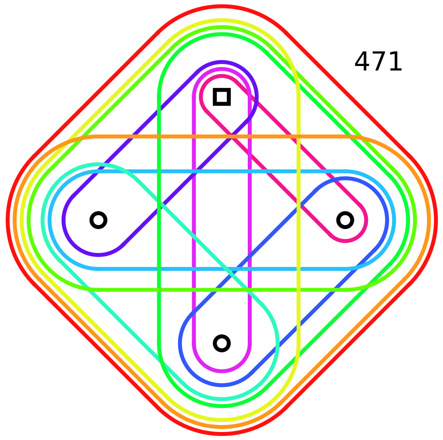

466


467


468


469


470


471