MATHDISCOVERIES
Mathematical Preprint

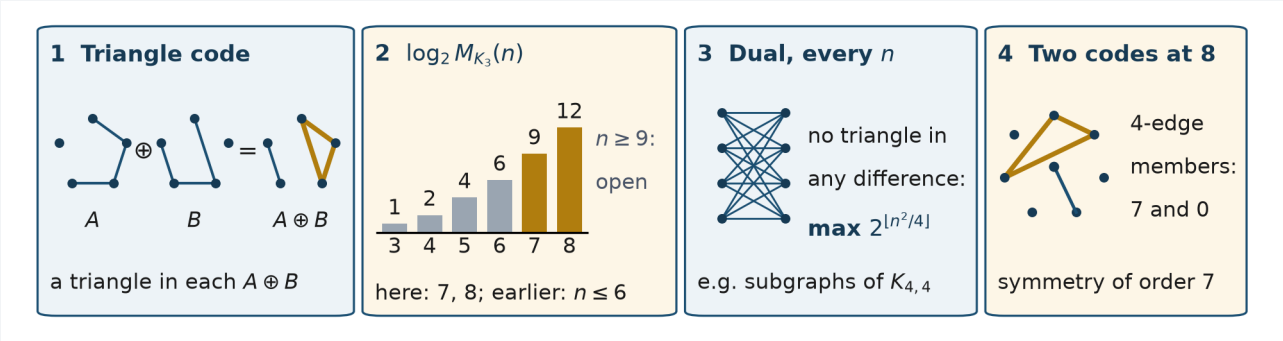
Working Draft

Triangle Codes of Graphs: Exact Values on Seven and Eight Vertices, and the Dual Problem on n Vertices

Tyler B. Hughes*,†

*AI research agents built and directed by the author wrote this draft. †Independent Researcher, United States

Disclosure: AI-assisted working draft directed by Tyler B. Hughes. Not for submission as is. No human mathematician has yet refereed the proof.



ABSTRACT: A family of graphs on $[n]$ is a triangle code if the symmetric difference of any two members contains a triangle; $M_{K_3}(n)$ is its largest size; $M_{C_{\mathrm{odd}}}(n)$ is defined with odd cycles. Alon et al. proved $M_{K_3}(n) \le M_{C_{\mathrm{odd}}}(n) \le 2^{\binom{n}{2}-\lfloor n^2/4 \rfloor}$, with equality for M_{K_3} at $3 \le n \le 6$ and for $M_{C_{\mathrm{odd}}}$ at $3 \le n \le 7$, and asked whether equality holds for all or infinitely many n. We show $M_{K_3}(7) = 512$ and $M_{K_3}(8) = M_{C_{\mathrm{odd}}}(8) = 4096$, so equality holds for $3 \le n \le 8$; $n \ge 9$ stays open. In the dual problem no difference may contain a triangle, or an odd cycle. Both maxima equal $2^{\lfloor n^2/4 \rfloor}$ for every $n \ge 0$; for large n this follows from a theorem of Balogh. For $n \ne 3$ the largest linear families are exactly the families of all subgraphs of a balanced complete bipartite graph. A linear triangle code meeting the upper bound is invariant under no transposition ($n \ge 5$), no 3-cycle ($n \ge 6$) and no double transposition (odd $n \ge 7$). On eight vertices two such codes with a common symmetry of order 7 are not isomorphic. Main results: 3 of 5 in Lean, 2 by written proof. AI research agents built and directed by Tyler B. Hughes produced and checked this work. No human mathematician has yet refereed the proof.

Verification Scope (G11): WARN: Human referee: pending
Main results: 3 of 5 in Lean Kernel (60%) · 2 written
Abstract claims: 3 of 5 in Lean Kernel (60%) · 2 written

Full Trust Boundary: Section 5.

MSC 2020: 05D05, 05C35, 94B05, 68V20 **Keywords:** graph codes, triangle codes, symmetric difference, triangle-free graphs, symmetrization, Lean 4

1. INTRODUCTION AND MAIN RESULTS

Draw several graphs on the same n labeled points. Any two of them differ in some set of edges. How many graphs can one draw so that this set of differing edges always contains a triangle? The question belongs to a theory of codes whose words are graphs, begun by Alon, Gujgiczer, Körner, Milojević and Simonyi [1]. The largest number is not known for general n.

Write $[n]$ for a set of n labeled points, $A \oplus B$ for the symmetric difference of two graphs on $[n]$ (the graph of the edges that lie in exactly one of them), and $m(n) = \lfloor n^2/4 \rfloor$, the number of edges of the balanced complete bipartite graph on n vertices. For a class $\mathcal{F}$ of graphs, $M_{\mathcal{F}}(n)$ is the largest size of a family of graphs on $[n]$ in which $A \oplus B$ contains a member of $\mathcal{F}$ for any two different members A, B, and $D_{\mathcal{F}}(n)$ is the largest size of a family in which no $A \oplus B$ with $A \ne B$ contains a member of $\mathcal{F}$. The second number is the *dual problem* of [1]. Here $\mathcal{F}$ is either the single graph K_3 or the class C_{odd} of all odd cycles of length at least 3. A family that is admissible for M_{K_3} is a *triangle code*; it is *linear* if it is a subspace of the $\mathbb{F}_2$-vector space of all graphs on $[n]$, with $\oplus$ as addition.

A bipartite graph contains no odd cycle. This gives, for all n,

$$M_{K_3}(n) \le M_{C_{\mathrm{odd}}}(n) \le 2^{\binom{n}{2}-m(n)}, \qquad D_{K_3}(n) \ge D_{C_{\mathrm{odd}}}(n) \ge 2^{m(n)}, \tag{1}$$

see Proposition 7, the text before Lemma 9, and [1, Proposition 9]. The authors of [1] showed that the first bound is attained by M_{K_3} at $n = 3,4,5,6$ (Propositions 14 to 16 there) and by $M_{C_{\mathrm{odd}}}$ also at $n = 7$ (Proposition 17 there), said that they could not decide $M_{K_3}(7)$, and asked whether $M_{K_3}(n) = 2^{\binom{n}{2}-m(n)}$ always or for infinitely many n, and the same for $M_{C_{\mathrm{odd}}}$ (Problem 4 there). We call a linear triangle code of dimension $\binom{n}{2}-m(n)$ *optimal*; by (1) no triangle code is larger.

Theorem 1 (Seven and eight vertices; Lean). *The values are* $M_{K_3}(7) = M_{C_{\mathrm{odd}}}(7) = 512$ *and* $M_{K_3}(8) = M_{C_{\mathrm{odd}}}(8) = 4096$.

There is a linear triangle code of dimension 9 *on* 7 *vertices and one of dimension* 12 *on* 8 *vertices.*

The value of $M_{C_{\mathrm{odd}}}(7)$ is Proposition 17 of [1]; the other three are not stated there. The exponents are $21-12=9$ and $28-16=12$, so equality holds in the first part of (1) for $3 \le n \le 8$. For $n \ge 9$ we do not know whether it does.

Theorem 2 (The dual problem; written proof). *The equations* $D_{K_3}(n) = D_{C_{\mathrm{odd}}}(n) = 2^{m(n)}$ *hold for every* $n \ge 0$.

Theorem 3 (Linear families in the dual problem; written proof). *Let* $n \ge 0$ *and let* S *be a linear family of graphs on* $[n]$ *none of whose members contains a triangle. Then* $\dim S \le m(n)$. *If* $n \ne 3$, *then* $\dim S = m(n)$ *if and only if* S *is the family of all subgraphs of a complete bipartite graph on* $[n]$ *with* $m(n)$ *edges.*

Theorem 4 (Symmetries that optimal codes do not have; Lean). *Let* V *be an optimal linear triangle code on* $[n]$.

(*i*) *If* $n \ge 5$, *then* V *is not invariant under any transposition of* $[n]$.
(*ii*) *If* $n \ge 6$, *then* V *is not invariant under any* 3*-cycle.*
(*iii*) *If* $n \ge 7$ *and* n *is odd, then* V *is not invariant under any product of two disjoint transpositions.*

Theorem 5 (Two codes on eight vertices; Lean). *There are a permutation* π *of order* 7 *of the* 8 *vertices and two optimal linear triangle codes* V *and* W *on* 8 *vertices, both invariant under* π *and both containing the complete graph* K_8, *such that* V *has exactly* 7 *members with* 4 *edges and* W *has none. No permutation of the vertices maps* V *to* W.

Remark 6 (What is not proved). Nothing here decides whether an optimal linear triangle code exists for any $n \ge 9$, nor what $M_{K_3}(n)$ or $M_{C_{\mathrm{odd}}}(n)$ is for any $n \ge 9$. For such n Theorem 4 is a condition that a code would have to satisfy; it may hold because there is no code. For $5 \le n \le 8$ optimal codes exist (Theorem 1 and Proposition 8). Theorem 3 describes the linear families only; whether a family that is not a translate of a linear one can attain $D_{K_3}(n)$ for $n \ge 4$ is open here. At $n = 3$ the description fails: the empty graph and the three paths with two edges form a linear family of dimension 2 (Section 3).

Earlier work. Several parts are known or close to known results; Section 6 gives details. The upper bound and the values for $n \le 6$ are from [1]. So are, by the remark after Lemma 1 there, the dual values $D_{K_3}(n) = 2^{m(n)}$ for $3 \le n \le 6$ and $D_{C_{\mathrm{odd}}}(n) = 2^{m(n)}$ for $3 \le n \le 7$: a family of $2^{\binom{n}{2}-m(n)}$ graphs of the first kind and one of $2^{m(n)}$ of the second are both largest. For general n the authors state the dual value asymptotically only ([1, Remark 6]). In Theorem 2 a compression step reduces the dual problem to counting edge sets that meet every triangle in at most one edge. For all sufficiently large n the bound on this count follows from a theorem of Balogh [4] on edge-colorings [to check]; what we add is the proof for every n. The inequality of Theorem 3 and its equality case are close to a result of Beers and Botnan [5] on Betti numbers of flag complexes; we give a direct proof.

An illustration. The box checks the first code of Table 1 directly. An edge *ab* is a two-digit number, a graph is a bit mask, the nine graphs are separated by semicolons, and they span 2^9 graphs under `xor`. The box tests each of the 511 nonempty ones for a triangle. The difference of two members is again a member, so any two members differ in a graph with a triangle.

Instant Witness Check: 512 graphs on 7 points

```
from itertools import combinations as C
B = """01 04 14;02 05 06 26;03 06 12 13 14 24;03 05 13 14 15 16 23
25 26 34;03 05 06 13 15 26 35;05 13 16 36;04 06 14 15 16 23 45;04
05 06 13 14 15 23 24 25 46;03 13 15 24 25 26 56"""
bit = {a + b: 1 << i for i, (a, b) in enumerate(C("0123456", 2))}
tri = [bit[a+b] + bit[a+c] + bit[b+c] for a, b, c in C("0123456", 3)]
span = {0}
for g in B.split(";"):
    span.update({x ^ sum(bit[e] for e in g.split()) for x in span})
print(len(span), all(any(x & t == t for t in tri) for x in span - {0}))
```

It prints `512 True`. Together with (1) this is the value $M_{K_3}(7) = 512$.

Roadmap. Section 2 fixes definitions and their Lean counterparts. Section 3 proves the results, in the order: the upper bound and the codes, the dual problem, its linear families, symmetries, and the codes on seven and eight vertices. Section 4 describes the computations, Section 5 the trust boundary, Section 6 prior work and open questions.

2. DEFINITIONS, NOTATION, AND SETUP

All graphs are finite and simple, and all families consist of labeled graphs on a fixed vertex set $[n]$. In Lean, $[n]$ is `Fin n`, whose elements are $0, \dots, n-1$; the table and the box use these labels. A pair of distinct vertices is an element of `UEdge` n, and a graph is a function from pairs to $\mathbb{F}_2$ (`Graph` n), where 1 means that the pair is an edge. The sum of two functions is the symmetric difference. The space of all graphs has dimension $\binom{n}{2}$ over $\mathbb{F}_2$.

Triangles and odd cycles. A graph G contains a triangle if there are vertices $a < b < c$ with the three pairs among them edges of G (`HasTriangle`). It contains an odd cycle if for some $k \ge 0$ there are $2k+3$ distinct vertices, in cyclic order, with consecutive ones adjacent (`HasOddCycle`); so a triangle is an odd cycle. A graph is bipartite if its vertices have a 2-coloring with no edge inside a color class (`IsBipartite`).

Codes. A finite family C of graphs on $[n]$ is a triangle code (`IsTriangleCode`) if $G \oplus H$ contains a triangle for all $G \ne H$ in C, and an odd-cycle code (`IsOddCycleCode`) if $G \oplus H$ contains an odd cycle for all such G, H. $M_{K_3}(n)$ and $M_{C_{\mathrm{odd}}}(n)$ (`M_K3`, `M_Codd`) are the largest sizes of such families; a family with one member is admissible, so both are at least 1, and for $n \le 2$ both equal 1. A linear triangle code is a subspace V in which every nonzero graph contains a triangle (`IsLinearTriangleCode`); it is optimal (`IsOptimalLinearTriangleCode`) if moreover $\dim V = \binom{n}{2} - m(n)$.

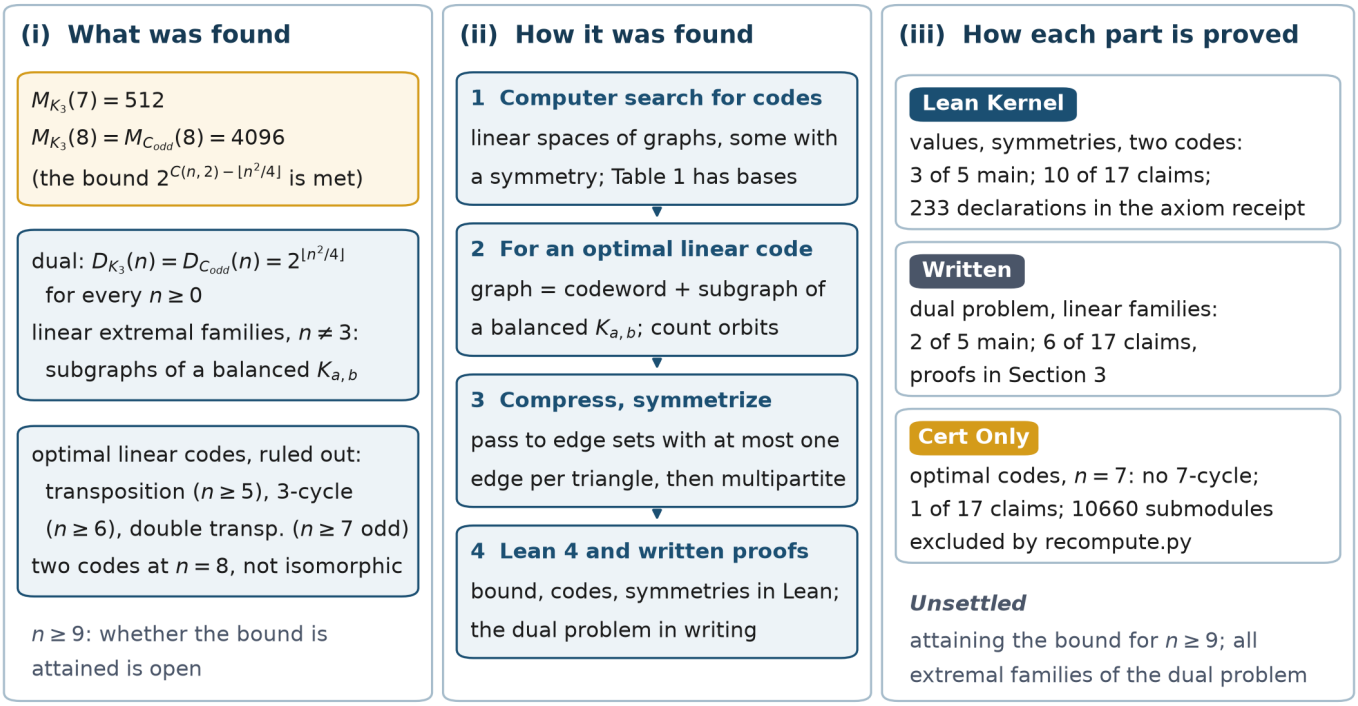


Figure 1. Discovery and proof architecture. Column (i) lists the results, column (ii) how they were found, column (iii) the proof tier of each result. The counts are those of the claim ledger of Section 5; the declaration count is read from the axiom receipt of the Lean library.

The dual problem. $D_{K_3}(n)$ and $D_{C_{\text{odd}}}(n)$ are the largest sizes of families in which no $G \oplus H$, for $G \neq H$, contains a triangle, respectively an odd cycle. A difference without an odd cycle has no triangle, so $D_{C_{\text{odd}}}(n) \leq D_{K_3}(n)$. These numbers and the statements about them are not formalized.

Bipartitions. A bipartition P of $[n]$ assigns one of two sides to each vertex. E_P is the set of pairs with ends on different sides, its *cross pairs* (`crossEdges`); the other pairs are the *inside pairs* (`insideEdges`). P is balanced if one side has $\lfloor n/2 \rfloor$ vertices (`IsBalancedPart`); then $|E_P| = m(n)$. For a set S of pairs, $\langle S \rangle$ is the space of all graphs whose edges lie in S (`supportSubspace`).

Symmetries. A permutation π of $[n]$ acts on pairs and hence on graphs (`permGraph`). V is invariant under π if $\pi(G) \in V$ for every $G \in V$ (`IsPermInvariant`), and under a group H of permutations if it is invariant under every element of H (`IsGroupInvariant`). A set S of pairs is H-invariant (`IsEdgeSetInvariant`) if the elements of H map S to S; $\text{orb}_H(S)$ is then the number of H-orbits in S (`orbitCount`), and $\text{Fix}(H)$ is the space of graphs fixed by every element of H (`fixedGraphs`). In Lean the intersection of two subspaces is their `min`. The weight of a graph is its number of edges (`graphWeight`), and `weightCount` V w is the number of members of V of weight w.

The count $\mathcal{I}$. For a graph G, let $\mathcal{I}(G)$ be the set of all $S \subseteq E(G)$ that contain at most one edge of every triangle of G, and $I(G) = |\mathcal{I}(G)|$. Such sets are called triangle-independent in [10]. The graph with no edges has $I = 1$, and $I(G) = 2^{|E(G)|}$ if G has no triangle.

3. PROOFS OF THE MAIN RESULTS

Statements marked Lean are proved in the library described in Section 5; for them this section gives the argument that the library formalizes. Statements marked written have their complete proof here. The proofs of Theorems 2 and 3 are the author agent's; the finding folder states these results without proofs (Section 7).

The upper bound and the codes.

Proposition 7 (Upper bound; Lean; the bound is known [1]). *For every $n \geq 0$, $M_{K_3}(n) \leq M_{C_{\text{odd}}}(n) \leq 2^{\binom{n}{2}-m(n)}$. If an optimal linear triangle code on $[n]$ exists, then $M_{K_3}(n) = M_{C_{\text{odd}}}(n) = 2^{\binom{n}{2}-m(n)}$.*

Proof. A triangle code is an odd-cycle code. Let C be an odd-cycle code and let P be a balanced bipartition. If $G \oplus X = H \oplus Y$ with $G, H \in C$ and $X, Y \in \langle E_P \rangle$, then $G \oplus H = X \oplus Y$ is bipartite, so $G = H$ and $X = Y$. Hence the $|C| \cdot 2^{m(n)}$ sums $G \oplus X$ are distinct graphs on $[n]$, and $|C| \cdot 2^{m(n)} \leq 2^{\binom{n}{2}}$. An optimal linear triangle code is a triangle code with $2^{\binom{n}{2}-m(n)}$ members, because the sum of two different members is a nonzero member. □

Proposition 8 (Small values; Lean; known [1]). *For $3 \leq n \leq 6$ the two numbers $M_{K_3}(n)$ and $M_{C_{\text{odd}}}(n)$ are equal, and the values are 2, 4, 16 and 64 for $n = 3, 4, 5, 6$.*

Table 1. Bases of three optimal linear triangle codes. Each word {ab cd ...} lists the edges of one basis graph; the vertices are $0,\dots,6$ for the first code and $0,\dots,7$ for the other two. The bases are those of the Lean file `Witnesses.lean` (`recompute.py`, block 7).

$n=7$, dimension 9	$n=8$, dimension 12: code V	$n=8$, dimension 12: code W
{01 04 14} {02 05 06 26} {03 06 12 13 14 24} {03 05 13 14 15 16 23 25 26 34} {03 05 06 13 15 26 35} {05 13 16 36} {04 06 14 15 16 23 45} {04 05 06 13 14 15 23 24 25 46} {03 13 15 24 25 26 56}	{01 05 15 37} {02 05 06 16 17 24 25 34 35 37} {03 04 07 15 17 37} {12 14 15 16 17 27} {04 07 13 14 26 27 34 36} {05 06 16 17 23 24 26 27 34 35} {07 24 25 45} {04 06 27 46} {14 15 17 26 27 47} {17 35 36 56} {04 07 15 17 25 26 27 57} {04 07 26 27 36 67}	{01 07 15 16 17 24 26 27 34 36} {02 04 16 24 26 36} {03 05 24 25 26 35} {04 05 12 14 16 17 24 25 26 27 34 35} {05 13 15 16 25 36} {15 16 23 25 26 27 34 35} {04 14 24 34 35 36 37 45} {05 14 16 24 25 26 36 46} {04 05 06 07 14 27 36 47} {04 06 07 14 15 16 26 27 34 36 37 56} {04 07 24 25 26 27 34 36 37 57} {04 06 07 15 17 25 26 27 34 35 36 67}

Proof of Theorem 1. We prove Proposition 8 at the same time. By Proposition 7 it is enough to give an optimal linear triangle code for each n with $3 \le n \le 8$. The exponents $\binom{n}{2} - m(n)$ are $1, 2, 4, 6, 9, 12$. Table 1 gives bases for $n = 7$ and $n = 8$; the bases for $3 \le n \le 6$ are in `Witnesses.lean`. In each basis the inside pairs of a balanced bipartition are pivots, one per basis graph, so the graphs are linearly independent. That every nonzero member of the span contains a triangle is a check of $2^d - 1$ graphs, with d the dimension. The Lean kernel evaluates it, and `recompute.py` (block 1) repeats it. □

The dual problem. The proof of Theorem 2 has four steps. For the lower bound let P be a balanced bipartition. The $2^{m(n)}$ graphs of $\langle E_P \rangle$ have bipartite differences, so $D_{K_3}(n) \ge D_{C_{\text{odd}}}(n) \ge 2^{m(n)}$. It remains to show $D_{K_3}(n) \le 2^{m(n)}$.

Lemma 9 (Compression; written proof). *The maximum of $I(G)$ over all graphs G on $[n]$ is $D_{K_3}(n)$, for every $n \ge 0$.*

Proof. If $S, S' \in \mathcal{J}(G)$, then $S \oplus S' \subseteq E(G)$ contains at most two edges of each triangle of G, hence contains no triangle. So $\mathcal{J}(G)$ is admissible and $D_{K_3}(n) \ge I(G)$.

Conversely let $\mathcal{F}$ be an admissible family with $D_{K_3}(n)$ members. For a pair e, replace every $A \in \mathcal{F}$ with $e \in A$ and $A - e \notin \mathcal{F}$ by $A - e$. The resulting family $\mathcal{F}'$ has as many members. It is admissible: consider two different members A', B' of $\mathcal{F}'$, which come from different members A, B of $\mathcal{F}$. If both were changed or both were not changed, then $A' \oplus B' = A \oplus B$. Otherwise let $A' = A - e$ and $B' = B$. If $e \notin B$, then $A' \oplus B' = (A \oplus B) - e$ is a subgraph of $A \oplus B$. If $e \in B$, then $B - e \in \mathcal{F}$, because B was not changed, and $A' \oplus B' = A \oplus (B - e)$; here $A \ne B - e$, since $e \in A$. In each case the difference contains no triangle. Repeat this for all pairs e as long as some member changes; each change lowers the total number of edges, so the process stops, at a family $\mathcal{F}$ closed under taking subgraphs. Let G be the union of its members. Suppose that $S \in \mathcal{F}$ contains two edges e, f of a triangle $\{e, f, g\}$ of G. Then $\{e, f\}$ and $\{g\}$ are subgraphs of members of $\mathcal{F}$, so they belong to $\mathcal{F}$, and their difference is the triangle, a contradiction. So $\mathcal{F} \subseteq \mathcal{J}(G)$ and $D_{K_3}(n) \le I(G)$. □

Lemma 10 (Symmetrization; written proof). *For every $n \ge 1$, some complete multipartite graph on $[n]$ attains $\max_G I(G)$.*

Proof. This is Zykov's symmetrization, which is standard for counts of this kind [11, 6]. Let u, v be two non-adjacent vertices of a graph G. For $T \in \mathcal{J}(G - u - v)$ let $f_u(T)$ be the number of sets X of edges at u with $T \cup X \in \mathcal{J}(G - v)$, and define $f_v(T)$ in the same way. No triangle contains both u and v, so $I(G) = \sum_T f_u(T) f_v(T)$. Let G_u be the graph obtained by giving v the neighbors of u, and G_v the reverse. Then $I(G_u) = \sum_T f_u(T)^2$ and $I(G_v) = \sum_T f_v(T)^2$, and $2 f_u f_v \le f_u^2 + f_v^2$ gives $2I(G) \le I(G_u) + I(G_v)$. So if G attains the maximum, then so does G_u.

Start with a graph that attains the maximum, choose a vertex u, and replace its non-neighbors one after another by copies of u. This does not change the neighbors of u. Afterwards u and its former non-neighbors form an independent set whose vertices are all joined to all other vertices. Choose a vertex outside this set and continue. All non-neighbors of the chosen vertex lie outside the sets formed so far, and a copy of it is joined to all of those sets, so they stay as they are. At the end the vertex set is divided into independent sets with all edges between different ones. □

Lemma 11 (Deletion; written proof). *Let $n \ge 1$, let G be the complete multipartite graph on $[n]$ with parts of sizes $a_1 \ge a_2 \ge \dots \ge a_r \ge 1$, and let v be a vertex of a part of size a_1. Then*

$$I(G) \le \Big(1 + \sum_{j=2}^{r} (2^{a_j} - 1)\Big)\, I(G - v). \tag{2}$$

Proof. Let $S \in \mathcal{J}(G)$. The edges of S that do not contain v form a set in $\mathcal{J}(G - v)$. If $vx, vy \in S$ with $x \ne y$, then $xy \notin E(G)$, as vxy would be a triangle with two edges in S. So the neighbors of v in S lie in one part other than the part of v, and the number of possibilities for them is at most the bracket in (2). □

Proof of Theorem 2. Because of Lemma 9, it is enough to show for every graph G on n vertices that $I(G) \le 2^{m(n)}$. We use induction on n; for $n = 0$ there is one graph and $I = 1$. Let $n \ge 1$ and $k = \lfloor n/2 \rfloor$, so that $m(n) - m(n-1) = k$. By Lemma 10 we may take G complete multipartite with parts $a_1 \ge \dots \ge a_r \ge 1$, and write $a = a_1$. If $r \le 2$, then G has no triangle and has at most $m(n)$ edges, so $I(G) \le 2^{m(n)}$. Let $r \ge 3$. By Lemma 11 and the induction hypothesis it suffices to show

$$L := 1 + \sum_{j=2}^{r} (2^{a_j} - 1) \le 2^k \tag{3}$$

unless we bound $I(G)$ directly. Note that $L \le 2^{n-a}$, since L counts some subsets of a set of $n - a$ vertices.

Case 1: $n - a \le k$. Then $L \le 2^{n-a} \le 2^k$.

From now on $a \le n - k - 1$. Put $g(x) = (2^x - 1)/x$ for integers $x \ge 1$. This is increasing, so $2^{a_j} - 1 \le a_j g(a)$ and

$$L \le 1 + (n - a)\, g(a). \tag{4}$$

We claim: if $A \ge 1$ is an integer, $n \ge 2A + 2$ and $1 \le a \le A$, then $(n - a)g(a) \le (n - A)g(A)$. Let $1 \le x \le A - 1$. Then

$n-x \geq A+3 \geq x+4$, so $(n-x-1)/(n-x) \geq (x+3)/(x+4)$. Further $g(x+1)/g(x) \geq (x+4)/(x+3)$: for $x=1$ this is $3/2 \geq 5/4$, and for $x \geq 2$ it follows from $g(x+1)/g(x) \geq 2x/(x+1)$ and $2x(x+3)-(x+1)(x+4) = x^2+x-4 > 0$. So $(n-x-1)g(x+1) \geq (n-x)g(x)$, which proves the claim.

Case 2: $n=2k$ is even. Here $a \leq k-1$, so $k \geq 2$. The claim with $A = k-1$ and (4) give

$$L \leq 1 + \frac{k+1}{k-1}(2^{k-1}-1).$$

For $k=2$ the right side is $4 = 2^2$. For $k \geq 3$ we have $(k+1)/(k-1) \leq 2$, and the right side is at most $2^k - 1$.

Case 3: $n = 2k+1$ is odd and $a \leq k-1$. So $k \geq 2$. The claim with $A = k-1$ and (4) give

$$L \leq 1 + \frac{k+2}{k-1}(2^{k-1}-1).$$

For $k \geq 4$ we have $(k+2)/(k-1) \leq 2$ and the right side is at most $2^k - 1$. For $k=3$ it is $17/2$, so the integer L is at most $8 = 2^3$. For $k=2$ we have $n=5$ and $a=1$, so $G = K_5$. Two edges of K_5 with a common end lie in a triangle, and two disjoint edges do not; hence $\mathcal{I}(K_5)$ is the set of matchings, and $I(K_5) = 1+10+15 = 26 \leq 64 = 2^{m(5)}$.

Case 4: $n = 2k+1$ is odd and $a = k$. As $n \geq 3$ here, $k \geq 1$. The parts after the first have sizes $a_2 \geq \ldots \geq a_r$ with sum $k+1$, and $r-1 \geq 2$.

If $a_2 = k$, then $r=3$, $a_3 = 1$ and $G = K_{1,k,k}$; let w be the vertex of the part of size one and X, Y the other two parts. The triangles of G are the sets wxy with $x \in X$, $y \in Y$. For $S \in \mathcal{I}(G)$ the neighbors of w in S lie in X or lie in Y. If they form a set $X' \subseteq X$, then S contains no pair xy with $x \in X'$, and the other $k(k-|X'|)$ pairs between X and Y are free. The same holds for Y, and the empty set is counted on both sides, so

$$I(G) = 2\sum_{i=0}^{k} \binom{k}{i} 2^{k(k-i)} - 2^{k^2} = 2(2^k+1)^k - 2^{k^2}.$$

Thus $I(G) \leq 2^{m(n)} = 2^{k^2+k}$ is equivalent to $2(1+2^{-k})^k - 1 \leq 2^k$. For $k=1$ both sides are 2. For $k \geq 2$, $(1+2^{-k})^k \leq e^{k/2^k} \leq e^{1/2} < 2$, so the left side is less than 3 and the right side is at least 4.

If $a_2 \leq k-1$ and $a_2 \geq 2$, then $k \geq 3$. The function $x \mapsto 2^x - 1$ satisfies $(2^x-1)+(2^y-1) \leq 2^{x+y}-1$, so $L \leq 1 + (2^{a_2}-1) + (2^{k+1-a_2}-1)$, where $2 \leq a_2 \leq k-1$. This is convex in a_2 and symmetric about $(k+1)/2$, so it is largest at $a_2 = 2$ and $a_2 = k-1$, and $L \leq 1+3+(2^{k-1}-1) = 2^{k-1}+3 \leq 2^k$.

If $a_2 = 1$ and $k \geq 2$, then the parts after the first are $k+1$ single vertices and $L = k+2 \leq 2^k$. For $k=1$ and $a_2 = 1$ we have $G = K_{1,1,1}$, which was treated above as $K_{1,k,k}$.

Since $a \leq n-k-1$ means $a \leq k-1$ for $n = 2k$ and $a \leq k$ for $n = 2k+1$, the cases cover all graphs with $r \geq 3$. □

Corollary 12 (Lemma 1 of [1] gives nothing more; written proof). *For every $n \geq 0$ and for $\mathcal{F} = K_3$ and $\mathcal{F} = C_{\text{odd}}$, the inequality $M_{\mathcal{F}}(n)\, D_{\mathcal{F}}(n) \leq 2^{\binom{n}{2}}$ is equivalent to $M_{\mathcal{F}}(n) \leq 2^{\binom{n}{2}-m(n)}$, and for $3 \leq n \leq 8$ it is an equality.*

Proof. Insert $D_{\mathcal{F}}(n) = 2^{m(n)}$; for $3 \leq n \leq 8$ use Theorem 1 and Proposition 8. □

Linear families in the dual problem.

Proof of Theorem 3. For a pair e let $x_e \colon S \to \mathbb{F}_2$ be the coordinate at e, and let H be the set of pairs e with $x_e \neq 0$, that is, the union of the members of S. We claim that $x_e + x_f + x_g = 0$ on S for every triangle $\{e,f,g\} \subseteq H$. The image of S under (x_e, x_f, x_g) is a subspace U of $\mathbb{F}_2^3$ that does not contain $(1,1,1)$, and no coordinate vanishes on U. If $\dim U \leq 1$, then a nonzero vector of U would have three nonzero coordinates, which is excluded, and $U = 0$ is excluded as well. So $\dim U = 2$, and U is the kernel of a nonzero linear form ℓ with $\ell(1,1,1) = 1$. Such a form is a coordinate or the sum of the three coordinates. It is not a coordinate, as no coordinate vanishes on U. This is the claim.

Let $K \subseteq H$ be a smallest set such that the projection $S \to \mathbb{F}_2^K$ is injective. If K contained a triangle $\{e,f,g\}$, then $x_e = x_f + x_g$ on S and the projection to $K \setminus \{e\}$ would be injective. So K has no triangle, and by Mantel's theorem [9], $\dim S \leq |K| \leq m(n)$.

The family of all subgraphs of a complete bipartite graph with $m(n)$ edges is linear of dimension $m(n)$ and its members have no triangle. Conversely let $\dim S = m(n)$. Then $|K| = m(n)$, the projection to $\mathbb{F}_2^K$ is bijective, and by the equality case of Mantel's theorem K is the set of pairs between the two sides of a balanced bipartition. If $K = H$, then S is the family of all subgraphs of K. This holds for $n \leq 2$, since for $n \leq 1$ there is no pair and for $n = 2$ the only pair is in K. Let $n \geq 4$, and suppose that there is a pair $ab \in H \setminus K$. Its ends are on the same side, and the other side has at least two vertices y, z. The triangles aby and abz lie in H, so $x_{ay} + x_{by} = x_{ab} = x_{az} + x_{bz}$ on S. This is a nontrivial linear relation between four different coordinates in K, which contradicts the surjectivity of $S \to \mathbb{F}_2^K$. So $H = K$. □

For $n = 3$ there are four linear families of dimension $m(3) = 2$: the subgraphs of each of the three paths with two edges, and the family of all graphs with an even number of edges. `recompute.py` (block 4) lists all linear families of largest dimension for $n = 3, 4, 5$; there are 4, 3 and 10, and the last two numbers are those of the balanced complete bipartite graphs.

Symmetries.

Lemma 13 (Complement; Lean). *Let V be an optimal linear triangle code on $[n]$ and let P be a balanced bipartition. The space of all graphs on $[n]$ is the direct sum of V and $\langle E_P \rangle$, and every prescribed set of inside pairs is the set of inside pairs of some member of V.*

Proof. A graph in $V \cap \langle E_P \rangle$ is bipartite and belongs to V, so it is empty; the dimensions add up to $\binom{n}{2}$. For the second part, write a graph with the prescribed inside pairs as the sum of a member of V and a graph in $\langle E_P \rangle$, which has no inside pairs. □

Lemma 14 (Orbits; Lean). *Let H be a group of permutations of $[n]$, let V be an H-invariant optimal linear triangle code on $[n]$, and let P be a balanced bipartition such that E_P is H-invariant. Then*

$$\dim(V \cap \mathrm{Fix}(H)) + \mathrm{orb}_H(E_P) = \mathrm{orb}_H(\textit{all pairs}).$$

If S is an H-invariant set of pairs that forms a graph without a triangle, then $\mathrm{orb}_H(S) \le \mathrm{orb}_H(E_P)$.

Proof. By Lemma 13 every graph is in a unique way a sum of a member of V and a member of $\langle E_P\rangle$. Both spaces are H-invariant, so if the graph is fixed by H then so are its two parts. Hence $\mathrm{Fix}(H)$ is the direct sum of $V \cap \mathrm{Fix}(H)$ and $\langle E_P\rangle \cap \mathrm{Fix}(H)$. For an H-invariant set S of pairs, the graphs of $\langle S\rangle$ fixed by H are the unions of H-orbits in S, a space of dimension $\mathrm{orb}_H(S)$. This gives the equation. If S has no triangle, then $\langle S\rangle \cap V = 0$, so $\mathrm{orb}_H(S) + \dim(V \cap \mathrm{Fix}(H)) \le \mathrm{orb}_H(\text{all pairs})$, and the inequality follows. □

Proof of Theorem 4. (i) Let $\tau = (a\,b)$ and suppose that V is τ-invariant. For $G \in V$ the graph $G \oplus \tau(G)$ belongs to V. It contains neither ab nor a pair disjoint from $\{a,b\}$, as these are fixed by τ. So all its edges join $\{a,b\}$ to the rest, it is bipartite, and hence it is empty. So every $G \in V$ is fixed by τ. As $n \ge 5$, some balanced bipartition has a, b and a third vertex x on one side. By Lemma 13 some $G \in V$ has ax and not bx as an edge, a contradiction.

(ii) Let H be generated by $(a\,b\,c)$. If a bipartition has a,b,c on a side of size s and the other side has size $t = n - s$, then E_P is H-invariant and $\mathrm{orb}_H(E_P) = (s-3)t + t = (s-2)t$. Let $n = 2k$, $k \ge 3$. A balanced bipartition with $s = t = k$ has $k(k-2)$ orbits, and the bipartition with $s = k+1$, $t = k-1$ has $(k-1)^2 = k(k-2)+1$. Let $n = 2k+1$, $k \ge 3$. The balanced bipartition with $s = k$ has $(k-2)(k+1) = k^2-k-2$ orbits, and the one with $s = k+1$ has $(k-1)k = k^2-k$. Both contradict Lemma 14, with S the cross pairs of the second bipartition.

(iii) Let $n = 2k+1$, $k \ge 3$, and let H be generated by $(a\,b)(c\,d)$. Take first a balanced bipartition with a,b,c,d on the side of size $k+1$. Of the $k(k+1)$ cross pairs, $(k-3)k$ are fixed and the $4k$ others form orbits of size two, so there are $(k-1)k = k^2-k$ orbits. Now take a balanced bipartition with a,b on one side and c,d on the other. The fixed cross pairs are the $(k-1)(k-2)$ pairs that meet none of a,b,c,d, and the others form orbits of size two, so there are $\frac{1}{2}(k(k+1) + (k-1)(k-2)) = k^2-k+1$ orbits. This contradicts Lemma 14. □

Proposition 15 (Powers at $n = 7$; Lean). *Let V be an optimal linear triangle code on 7 vertices and π a permutation of the vertices. If some power of π is a transposition, a 3-cycle, or a product of two disjoint transpositions, then V is not invariant under π.*

Proof. If V is invariant under π, it is invariant under its powers. Apply Theorem 4 with $n = 7$. □

Codes on seven and eight vertices.

Proposition 16 (Symmetric codes on seven vertices; Lean). *For each $r \in \{3,5,6\}$ there are a permutation of order r of the 7 vertices and an optimal linear triangle code on 7 vertices that is invariant under it.*

Proof. The bases for $r = 3,5,6$ are `basis7_z3`, `basis7_c5` and `basis7_c6` of the file `Witnesses.lean`; the permutations are $(0\,1\,2)(4\,5\,6)$, $(0\,1\,2\,3\,4)$, and $(0\,1\,2\,3\,4\,5)$. The check is as for Theorem 1, together with the check that each permutation maps the basis graphs into the span. □

Proposition 17 (Cycle types on seven vertices; certificate). *A permutation of 7 vertices leaves some optimal linear triangle code on 7 vertices invariant if and only if its cycle type is 1^7, $2^3\,1$, $3^2\,1$, $5\,1^2$ or $6\,1$.*

Proof. There are 15 cycle types. Proposition 15 excludes all but 1^7, $2^3\,1$, $3^2\,1$, $5\,1^2$, $6\,1$ and 7. The first five occur: 1^7 by Theorem 1, and $3^2\,1$, $5\,1^2$ and $6\,1$ by the permutations in the proof of Proposition 16; the cube of the 6-cycle has type $2^3\,1$. Conjugate permutations leave isomorphic codes invariant, so the choice of the permutation within a type does not matter.

It remains to exclude a 7-cycle π. A π-invariant code V is a module over $\mathbb{F}_2[x]/(x^7-1)$, where x acts as π. Over $\mathbb{F}_2$, $x^7-1 = (x-1)(x^3+x+1)(x^3+x^2+1)$, and the 21 pairs form three orbits of size 7. So the space of graphs is the direct sum of the space of fixed graphs, of dimension 3, and two spaces of dimension 9 on which a cubic factor acts as zero and which are 3-dimensional over the field with 8 elements. A graph fixed by π is a union of orbits. Each single orbit is a 7-cycle and has no triangle. If V has dimension 9, then $9 = t+3a+3b$, where $t \le 3$ is the dimension of the fixed graphs in V and $0 \le a,b \le 3$ are the dimensions over the field with 8 elements; so $t \in \{0,3\}$. If $t = 3$, then V contains the 7-cycles, which is excluded. If $t = 0$, then $a+b = 3$, and there are $1+73\cdot73+73\cdot73+1 = 10660$ submodules, because the space of dimension 3 over the field with 8 elements has 73 lines and 73 planes. `recompute.py` (block 6) goes through all of them and finds in each a nonzero graph without a triangle. A submodule is the sum of a part in the first space of dimension 9 and a part in the second; when the first part already contains a nonzero graph without a triangle, all submodules with that first part are excluded together (4089 of them), and each of the other 6571 is tested graph by graph. □

Proposition 18 (A weight distribution; Lean). *There is an optimal linear triangle code on 8 vertices all of whose members have an even number of edges, with the following numbers of members of each weight: one each of weights 0 and 28, none of weights 2 and 26, 7 each of weights 4 and 24, 14 each of weights 6 and 22, 105 each of weights 8 and 20, 336 each of weights 10 and 18, 847 each of weights 12 and 16, and 1476 of weight 14.*

Proof of Theorem 5 and Proposition 18. V and W are the codes of Table 1, and $\pi = (0\,1\,2\,3\,4\,5\,6)$. Dimension, triangles, invariance, the members with 4 edges, and $K_8 \in V \cap W$ are finite checks as before. A permutation σ of the vertices preserves the number of edges, so it maps the members of V with 4 edges to those of $\sigma(V)$; as $7 \ne 0$, $\sigma(V) \ne W$. The weight distribution in Proposition 18 is that of V. □

Remark 19 (The members with four edges). The 7 members of V with 4 edges are, for $i = 0,\dots,6$, the triangle $\{i, i+1, i+5\}$ together with the edge $\{i+3, 7\}$, with $i+1$, $i+3$ and $i+5$ taken modulo 7. The seven triangles are the lines of a projective plane of order two on the points $0,\dots,6$: any two of them share exactly one vertex. The weights of W are symmetric as well: W has 1, 7, 98, 420 and 861 members of weights 0, 6, 8, 10 and 12, 1322 of weight 14, and as many of weight $28-w$ as of weight w. These facts are read from `recompute.out` (block 2).

4. COMPUTATIONAL VERIFICATION

The Lean build. The library `TriangleFreeCodeCap` of the finding was built with Lean 4 version `4.35.0` and Mathlib `v4.35.0`, with no `native_decide` and no `sorry`; the finite checks of the codes are evaluated by the kernel. The module `TriangleFreeCodeCap.PaperDraft1` gathers the statements cited above as conjunctions of theorems of the library and adds four sanity equalities. The file `VerifyDraft1.lean` prints the axioms of every theorem and lemma of the four modules of the library (`Basic`, `SymmetryNoGo`, `Witnesses` and `PaperDraft1`), 233 declarations; the theorems of Mathlib that they use are not listed. Every one depends on a subset of `propext`, `Classical.choice` and `Quot.sound`. The two commands, run in the folder `lean/` of the finding, are

```
lake build TriangleFreeCodeCap.PaperDraft1
lake env lean TriangleFreeCodeCap/VerifyDraft1.lean
```

The printout is a committed receipt from the author's machine; no independent replay has been run.

Exact replay. `recompute.py` uses the Python standard library and exact integer and rational arithmetic; one run took 1.4 seconds on an Apple M4 Max with CPython 3.11. Its blocks are these. (1) The codes for $3 \le n \le 8$, read from `Witnesses.lean`: dimension, and a triangle in every nonzero member. (2) Invariance, weights, and the members with four edges. (3) The count I: its maximum over all labeled graphs for $n \le 6$, and over the complete multipartite graphs for $n = 7, 8$ up to isomorphism (one for each of the 15 and 22 partitions of n; I does not change under relabeling), is $2^{m(n)}$; for $4 \le n \le 6$ it is attained exactly by the balanced complete bipartite graphs (3, 10 and 10 of them); the formula for $I(K_{1,k,k})$ is compared with a count for $k \le 4$, where the values are 4, 34, 946 and 101506, and inequality (3) with all partitions of $n \le 30$ into at least three parts other than the two exceptions 1^5 and $k, k, 1$. (4) The linear families of Theorem 3 for $n = 3, 4, 5$. (5) The orbit counts in the proof of Theorem 4 for $n \le 20$ and $n \le 21$. (6) The cycle types, and the exclusion of all 10660 submodules in the proof of Proposition 17. (7) The printed bases and the Python box. The blocks (3) to (5) illustrate written or Lean-checked proofs and are not needed for them; block (6) is part of the proof of Proposition 17.

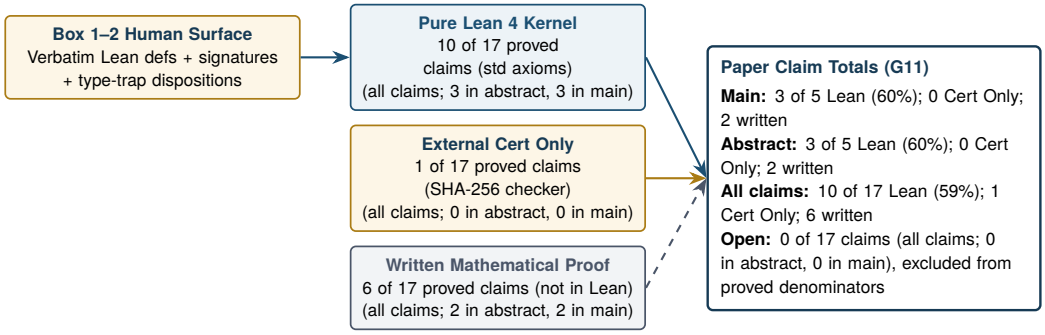
Reading Section 5. The next section is generated by the project's checking tool from the Lean sources and the manifest of this draft. Its tables restate every numbered result without its context and give its tier: Lean kernel, certificate only, or written proof. A written proof that cites a computation over a finite range as an illustration keeps the tier *written*.

5. FORMAL VERIFICATION & READER'S TRUST BOUNDARY

What You Must Trust to Accept This Paper:

1. **Lean 4 Kernel & Toolchain:** leanprover/lean4:v4.35.0; committed axiom receipt anchored to git commit aaf43e0528f3 (independent gate-box kernel replay pending); standard axioms only (propext, Classical.choice, Quot.sound).
2. **Core Definitions & Fidelity (Boxes 1–2):** Verify that the verbatim Lean definitions in Box 1 match Section 2 and that each Lean signature in Box 2 matches its stated theorem.
3. **External Certificates & Written Proofs:** 1 SHA-256-pinned external artifact(s) plus any claims marked Written or Unsettled in Table 3 below.

Reader's Trust-Flow Architecture (Distinct Evidence Paths to Claim Totals)



Anti-Shortcut Audit Strip — Source-Level Approx (Gate Replay Pending)

WARN: Committed receipt (gate replay pending) | PASS: sorry/admit: 0 (per committed receipt) | PASS: native_decide: 0 (per committed receipt) | PASS: Custom axioms & meta: 0 (per committed receipt) | PASS: Coverage: 233/233 public theorems (source-level approx) | WARN: Fidelity: 10/10 exact (draft; auditor/formalizer relationship unknown) | WARN: Witnesses: pending gate replay | WARN: Human referee: pending

Box 1 — Verbatim Lean 4 Custom Definitions (Human Audit Surface)

TriangleFreeCodeCap.UEdge (TriangleFreeCodeCap/Basic.lean:63-63) — Section 2, first paragraph

```
abbrev UEdge (n : ℕ) := { e : Sym2 (Fin n) // ¬ e.IsDiag }
```

TriangleFreeCodeCap.Graph (TriangleFreeCodeCap/Basic.lean:159-159) — Section 2, first paragraph

```
abbrev Graph (n : ℕ) := UEdge n → ZMod 2
```

TriangleFreeCodeCap.mkEdge (TriangleFreeCodeCap/Basic.lean:69-70) — Section 2, first paragraph

```
def mkEdge {n : ℕ} {u v : Fin n} (h : u ≠ v) : UEdge n :=
  ⟨s(u, v), Sym2.mk_isDiag_iff.not.mpr h⟩
```

TriangleFreeCodeCap.edgeVal (TriangleFreeCodeCap/Basic.lean:162-163) — Section 2, first paragraph

```
def edgeVal {n : ℕ} (G : Graph n) (u v : Fin n) : ZMod 2 :=
  if h : u = v then 0 else G (mkEdge h)
```

TriangleFreeCodeCap.adj (TriangleFreeCodeCap/Basic.lean:166-167) — Section 2, first paragraph

```
def adj {n : ℕ} (G : Graph n) (u v : Fin n) : Prop :=
  edgeVal G u v = 1
```

TriangleFreeCodeCap.HasTriangle (TriangleFreeCodeCap/Basic.lean:301-302) — Triangles and odd cycles (Section 2)

```
def HasTriangle {n : ℕ} (G : Graph n) : Prop :=
  ∃ a b c : Fin n, a < b ∧ b < c ∧ adj G a b ∧ adj G b c ∧ adj G a c
```

TriangleFreeCodeCap.HasOddCycle (TriangleFreeCodeCap/Basic.lean:337-339) — Triangles and odd cycles (Section 2)

```
def HasOddCycle {n : ℕ} (G : Graph n) : Prop :=
  ∃ (k : ℕ) (v : Fin (2 * k + 3) → Fin n),
    Function.Injective v ∧ ∀ i : Fin (2 * k + 3), adj G (v i) (v (i + 1))
```

TriangleFreeCodeCap.IsTriangleCode (TriangleFreeCodeCap/Basic.lean:642-643) — Codes (Section 2)

```
def IsTriangleCode {n : ℕ} (C : Finset (Graph n)) : Prop :=
  ∀ G ∈ C, ∀ H ∈ C, G ≠ H → HasTriangle (G + H)
```

TriangleFreeCodeCap.IsOddCycleCode (TriangleFreeCodeCap/Basic.lean:647-648) — Codes (Section 2)

```
def IsOddCycleCode {n : ℕ} (C : Finset (Graph n)) : Prop :=
  ∀ G ∈ C, ∀ H ∈ C, G ≠ H → HasOddCycle (G + H)
```

TriangleFreeCodeCap.M_K3 (TriangleFreeCodeCap/Basic.lean:767-769) — Codes (Section 2)

```
noncomputable def M_K3 (n : ℕ) : ℕ :=
  letI := Classical.decPred (fun C : Finset (Graph n) => IsTriangleCode C)
  (Finset.univ.filter (fun C : Finset (Graph n) => IsTriangleCode C)).sup Finset.card
```

TriangleFreeCodeCap.M_Codd (TriangleFreeCodeCap/Basic.lean:772-774) — Codes (Section 2)

```
noncomputable def M_Codd (n : ℕ) : ℕ :=
  letI := Classical.decPred (fun C : Finset (Graph n) => IsOddCycleCode C)
  (Finset.univ.filter (fun C : Finset (Graph n) => IsOddCycleCode C)).sup Finset.card
```

TriangleFreeCodeCap.IsLinearTriangleCode (TriangleFreeCodeCap/Basic.lean:656-657) — Codes (Section 2)

```
def IsLinearTriangleCode {n : ℕ} (V : Submodule (ZMod 2) (Graph n)) : Prop :=
  ∀ G ∈ V, G ≠ 0 → HasTriangle G
```

TriangleFreeCodeCap.IsOptimalLinearTriangleCode (TriangleFreeCodeCap/Basic.lean:661-662) — Codes (Section 2)

```
def IsOptimalLinearTriangleCode {n : ℕ} (V : Submodule (ZMod 2) (Graph n)) : Prop :=
  IsLinearTriangleCode V ∧ Module.finrank (ZMod 2) V = n.choose 2 - n ^ 2 / 4
```

TriangleFreeCodeCap.isCrossEdge (TriangleFreeCodeCap/Basic.lean:402-403) — Bipartitions (Section 2)

```
def isCrossEdge {n : ℕ} (P : Fin n → Bool) (e : UEdge n) : Bool :=
  Sym2.lift ⟨fun u v => P u != P v, fun u v => by dsimp only; cases P u <;> cases P v
    ↪ <;> rfl⟩ e.1
```

TriangleFreeCodeCap.crossEdges (TriangleFreeCodeCap/Basic.lean:409-410) — Bipartitions (Section 2)

```
def crossEdges {n : ℕ} (P : Fin n → Bool) : Finset (UEdge n) :=
  Finset.univ.filter (fun e => isCrossEdge P e = true)
```

TriangleFreeCodeCap.insideEdges (TriangleFreeCodeCap/Basic.lean:413-414) — Bipartitions (Section 2)

```
def insideEdges {n : ℕ} (P : Fin n → Bool) : Finset (UEdge n) :=
  Finset.univ.filter (fun e => isCrossEdge P e = false)
```

TriangleFreeCodeCap.IsBalancedPart (TriangleFreeCodeCap/Basic.lean:533-535) — Bipartitions (Section 2)

```
def IsBalancedPart {n : ℕ} (P : Fin n → Bool) : Prop :=
  (Finset.univ.filter (fun v : Fin n => P v = true)).card = n / 2 ∨
  (Finset.univ.filter (fun v : Fin n => P v = false)).card = n / 2
```

TriangleFreeCodeCap.supportSubspace (TriangleFreeCodeCap/Basic.lean:417-421) — Bipartitions (Section 2)

```
def supportSubspace {n : ℕ} (S : Finset (UEdge n)) : Submodule (ZMod 2) (Graph n)
  ↪ where
  carrier := { G | ∀ e : UEdge n, e ∉ S → G e = 0 }
  add_mem' hG hH e he := by simp [hG e he, hH e he]
  zero_mem' _ _ := rfl
  smul_mem' c _ hG e he := by simp [hG e he]
```

TriangleFreeCodeCap.permUEdge (TriangleFreeCodeCap/SymmetryNoGo.lean:56-66) — Symmetries (Section 2)

```
def permUEdge {n : ℕ} (π : Equiv.Perm (Fin n)) (e : UEdge n) : UEdge n :=
  ⟨Sym2.map π e.1, by
    rcases e with ⟨e, he⟩
    induction e using Sym2.ind with
    | _ u v =>
      intro h
      apply he
      change s(π u, π v).IsDiag at h
      rw [Sym2.mk_isDiag_iff] at h
      rw [Sym2.mk_isDiag_iff]
      exact π.injective h⟩
```

TriangleFreeCodeCap.permGraph (TriangleFreeCodeCap/SymmetryNoGo.lean:102-103) — Symmetries (Section 2)

```
def permGraph {n : ℕ} (π : Equiv.Perm (Fin n)) (G : Graph n) : Graph n :=
  fun e => G (permUEdge π⁻¹ e)
```

TriangleFreeCodeCap.permGraphLin (TriangleFreeCodeCap/SymmetryNoGo.lean:132-138) — Symmetries (Section 2)

```
def permGraphLin {n : ℕ} (π : Equiv.Perm (Fin n)) : Graph n ≃ₗ[ZMod 2] Graph n where
  toFun := permGraph π
  map_add' := permGraph_add π
  map_smul' := permGraph_smul π
  invFun := permGraph π⁻¹
  left_inv := permGraph_cancel_inv π
  right_inv := permGraph_inv_cancel π
```

TriangleFreeCodeCap.permSubmodule (TriangleFreeCodeCap/SymmetryNoGo.lean:184-186) — Symmetries (Section 2)

```
def permSubmodule {n : ℕ} (π : Equiv.Perm (Fin n)) (V : Submodule (ZMod 2) (Graph n))
  ↪ :
    Submodule (ZMod 2) (Graph n) :=
  V.map (permGraphLin π).toLinearMap
```

TriangleFreeCodeCap.IsPermInvariant (TriangleFreeCodeCap/SymmetryNoGo.lean:231-232) — Symmetries (Section 2)

```
def IsPermInvariant {n : ℕ} (π : Equiv.Perm (Fin n)) (V : Submodule (ZMod 2) (Graph
  ↪ n)) : Prop :=
  ∀ G ∈ V, permGraph π G ∈ V
```

TriangleFreeCodeCap.IsGroupInvariant (TriangleFreeCodeCap/SymmetryNoGo.lean:235-237) — Symmetries (Section 2)

```
def IsGroupInvariant {n : ℕ} (H : Subgroup (Equiv.Perm (Fin n)))
    (V : Submodule (ZMod 2) (Graph n)) : Prop :=
  ∀ π ∈ H, IsPermInvariant π V
```

TriangleFreeCodeCap.IsEdgeSetInvariant (TriangleFreeCodeCap/SymmetryNoGo.lean:402-403) — Symmetries (Section 2)

```
def IsEdgeSetInvariant {n : ℕ} (H : Subgroup (Equiv.Perm (Fin n))) (S : Finset (UEdge
  ↪ n)) : Prop :=
  ∀ π ∈ H, ∀ e ∈ S, permUEdge π e ∈ S
```

TriangleFreeCodeCap.fixedGraphs (TriangleFreeCodeCap/SymmetryNoGo.lean:392-396) — Symmetries (Section 2)

```
def fixedGraphs {n : ℕ} (H : Subgroup (Equiv.Perm (Fin n))) : Submodule (ZMod 2)
  ↪ (Graph n) where
  carrier := { G | ∀ π ∈ H, permGraph π G = G }
  add_mem' hG hH π hπ := by rw [permGraph_add, hG π hπ, hH π hπ]
  zero_mem' _ _ := rfl
  smul_mem' c _ hG π hπ := by rw [permGraph_smul, hG π hπ]
```

TriangleFreeCodeCap.edgeOrbit (TriangleFreeCodeCap/SymmetryNoGo.lean:421-424) — Symmetries (Section 2)

```
noncomputable def edgeOrbit {n : ℕ} (H : Subgroup (Equiv.Perm (Fin n))) (e : UEdge n)
  ↪ :
    Finset (UEdge n) :=
  letI := Classical.decPred (fun π : Equiv.Perm (Fin n) => π ∈ H)
  (Finset.univ.filter (fun π : Equiv.Perm (Fin n) => π ∈ H)).image (fun π =>
    ↪ permUEdge π e)
```

TriangleFreeCodeCap.orbitSet (TriangleFreeCodeCap/SymmetryNoGo.lean:427-429) — Symmetries (Section 2)

```
noncomputable def orbitSet {n : ℕ} (H : Subgroup (Equiv.Perm (Fin n))) (S : Finset
  ↪ (UEdge n)) :
    Finset (Finset (UEdge n)) :=
  S.image (edgeOrbit H)
```

TriangleFreeCodeCap.orbitCount (TriangleFreeCodeCap/SymmetryNoGo.lean:432-434) — Symmetries (Section 2)

```
noncomputable def orbitCount {n : ℕ} (H : Subgroup (Equiv.Perm (Fin n))) (S : Finset
  ↪ (UEdge n)) :
    ℕ :=
  (orbitSet H S).card
```

TriangleFreeCodeCap.threeCycle (TriangleFreeCodeCap/SymmetryNoGo.lean:712-713) — Theorem on symmetries

```
def threeCycle {n : ℕ} (a b c : Fin n) : Equiv.Perm (Fin n) :=
  Equiv.swap a b * Equiv.swap b c
```

TriangleFreeCodeCap.doubleTransposition (TriangleFreeCodeCap/SymmetryNoGo.lean:994-995) — Theorem on symmetries

```
def doubleTransposition {n : ℕ} (a b c d : Fin n) : Equiv.Perm (Fin n) :=
  Equiv.swap a b * Equiv.swap c d
```

TriangleFreeCodeCap.completeGraph (TriangleFreeCodeCap/Basic.lean:289-289) — Theorem on two codes

```
def completeGraph (n : ℕ) : Graph n := fun _ => 1
```

TriangleFreeCodeCap.graphWeight (TriangleFreeCodeCap/Basic.lean:292-293) — Symmetries (Section 2)

```
def graphWeight {n : ℕ} (G : Graph n) : ℕ :=
  (Finset.univ.filter (fun e : UEdge n => G e = 1)).card
```

TriangleFreeCodeCap.submoduleFinset (TriangleFreeCodeCap/Basic.lean:665-667) — Symmetries (Section 2)

```
noncomputable def submoduleFinset {n : ℕ} (V : Submodule (ZMod 2) (Graph n)) : Finset
  ↪ (Graph n) :=
  letI := Classical.decPred (fun G : Graph n => G ∈ V)
  Finset.univ.filter (fun G => G ∈ V)
```

TriangleFreeCodeCap.weightCount (TriangleFreeCodeCap/SymmetryNoGo.lean:200-201) — Symmetries (Section 2)

```
noncomputable def weightCount {n : ℕ} (V : Submodule (ZMod 2) (Graph n)) (w : ℕ) : ℕ
  ↪ :=
  ((submoduleFinset V).filter (fun G => graphWeight G = w)).card
```

Selected Concrete Sanity Examples (checked in Lean; examples only, not global identity proofs): TriangleFreeCodeCap.sanity_card_UEdge_four (Fintype.card (UEdge 4) = 6); TriangleFreeCodeCap.sanity_exponent_seven (Nat.choose 7 2 - 7 ^ 2 / 4 = 9); TriangleFreeCodeCap.sanity_exponent_eight (Nat.choose 8 2 - 8 ^ 2 / 4 = 12); TriangleFreeCodeCap.sanity_M_K3_three (M_K3 3 = 2).

Box 2 — Verbatim Main Theorem Signatures, Context Binders & Applicability

TriangleFreeCodeCap.paper_values (TriangleFreeCodeCap/PaperDraft1.lean:35-43) src:3f2cfc6065cd52

```
theorem paper_values :
    (M_K3 7 = 512 ∧ M_Codd 7 = 512) ∧ (M_K3 8 = 4096 ∧ M_Codd 8 = 4096) ∧
    (∃ V : Submodule (ZMod 2) (Graph 7),
      IsLinearTriangleCode V ∧ Module.finrank (ZMod 2) V = 9) ∧
    (∃ V : Submodule (ZMod 2) (Graph 8),
      IsLinearTriangleCode V ∧ Module.finrank (ZMod 2) V = 12)
```

Assumptions: No binders. M_K3 and M_Codd are suprema of cardinalities over all finite families with the property. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

TriangleFreeCodeCap.paper_symmetry (TriangleFreeCodeCap/PaperDraft1.lean:70-81) src:89e2441f14afae

```
theorem paper_symmetry {n : ℕ} {V : Submodule (ZMod 2) (Graph n)}
    (hopt : IsOptimalLinearTriangleCode V) :
    (5 ≤ n → ∀ a b : Fin n, a ≠ b → ¬ IsPermInvariant (Equiv.swap a b) V) ∧
    (6 ≤ n → ∀ a b c : Fin n, a ≠ b → b ≠ c → a ≠ c →
      ¬ IsPermInvariant (threeCycle a b c) V) ∧
    (7 ≤ n → n % 2 = 1 → ∀ a b c d : Fin n,
      a ≠ b → a ≠ c → a ≠ d → b ≠ c → b ≠ d → c ≠ d →
      ¬ IsPermInvariant (doubleTransposition a b c d) V)
```

Assumptions: n is a natural number, V a subspace of Graph n that is optimal. Part (i) assumes n >= 5 and quantifies over distinct a, b. Part (ii) assumes n >= 6 and quantifies over pairwise distinct a, b, c. Part (iii) assumes odd n >= 7 and quantifies over pairwise distinct a, b, c, d. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

TriangleFreeCodeCap.paper_two_codes (TriangleFreeCodeCap/PaperDraft1.lean:97-110) src:c278bd055c666c

```
theorem paper_two_codes :
    ∃ (π : Equiv.Perm (Fin 8)) (V W : Submodule (ZMod 2) (Graph 8)),
      orderOf π = 7 ∧
      (IsOptimalLinearTriangleCode V ∧ IsPermInvariant π V ∧ completeGraph 8 ∈ V ∧
        weightCount V 4 = 7) ∧
      (IsOptimalLinearTriangleCode W ∧ IsPermInvariant π W ∧ completeGraph 8 ∈ W ∧
        weightCount W 4 = 0) ∧
      ∀ σ : Equiv.Perm (Fin 8), permSubmodule σ V ≠ W
```

Assumptions: No binders: the permutation and the two codes are given by the existential statement; in the last clause sigma is any permutation of the 8 vertices. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

TriangleFreeCodeCap.paper_upper (TriangleFreeCodeCap/PaperDraft1.lean:20-25) src:f33c2fa76576d1

```
theorem paper_upper (n : ℕ) :
    M_K3 n ≤ M_Codd n ∧ M_Codd n ≤ 2 ^ (n.choose 2 - n ^ 2 / 4) ∧
    ∀ V : Submodule (ZMod 2) (Graph n), IsOptimalLinearTriangleCode V →
      M_K3 n = 2 ^ (n.choose 2 - n ^ 2 / 4) ∧ M_Codd n = 2 ^ (n.choose 2 - n ^ 2 / 4)
```

Assumptions: n is any natural number; in the second part V is any optimal linear triangle code on n vertices. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

TriangleFreeCodeCap.paper_small (TriangleFreeCodeCap/PaperDraft1.lean:28-32) src:00cb95fedddb53

```
theorem paper_small :
    (M_K3 3 = 2 ∧ M_Codd 3 = 2) ∧ (M_K3 4 = 4 ∧ M_Codd 4 = 4) ∧
    (M_K3 5 = 16 ∧ M_Codd 5 = 16) ∧ (M_K3 6 = 64 ∧ M_Codd 6 = 64)
```

Assumptions: No binders. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

TriangleFreeCodeCap.paper_complement (TriangleFreeCodeCap/PaperDraft1.lean:47-54) src:0232f3d2338768

```
theorem paper_complement {n : ℕ} {V : Submodule (ZMod 2) (Graph n)}
    (hopt : IsOptimalLinearTriangleCode V) {P : Fin n → Bool} (hbal : IsBalancedPart
      ↪ P) :
    IsCompl V (supportSubspace (crossEdges P)) ∧
    ∀ f : { e : UEdge n // e ∈ insideEdges P } → ZMod 2,
      ∃ G ∈ V, ∀ e : { e : UEdge n // e ∈ insideEdges P }, G e.1 = f e
```

Assumptions: n is a natural number, V an optimal linear triangle code, P a balanced map to Bool; f is any function on the inside pairs with values in ZMod 2. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

TriangleFreeCodeCap.paper_orbit (TriangleFreeCodeCap/PaperDraft1.lean:57-67) src:ee2ecc29c619ce

```
theorem paper_orbit {n : ℕ} {H : Subgroup (Equiv.Perm (Fin n))}
    {V : Submodule (ZMod 2) (Graph n)} (hopt : IsOptimalLinearTriangleCode V)
    (hV : IsGroupInvariant H V) {P : Fin n → Bool} (hbal : IsBalancedPart P)
    (hP : IsEdgeSetInvariant H (crossEdges P)) :
    Module.finrank (ZMod 2) (min V (fixedGraphs H) : Submodule (ZMod 2) (Graph n)) +
      orbitCount H (crossEdges P) = orbitCount H Finset.univ ∧
    ∀ S : Finset (UEdge n), IsEdgeSetInvariant H S →
      ¬ HasTriangle (fun e => if e ∈ S then 1 else 0) →
      orbitCount H S ≤ orbitCount H (crossEdges P)
```

Assumptions: n is a natural number, H any subgroup of the permutations of Fin n, V an optimal linear triangle code invariant under H, P a balanced map to Bool whose set of cross pairs is H-invariant; min of two subspaces is their intersection; S is any H-invariant finite set of pairs whose indicator graph has no triangle. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

TriangleFreeCodeCap.no_perm_with_forbidden_pow_n7 (TriangleFreeCodeCap/SymmetryNoGo.lean:1508-1529) src:29ba7131958e97

```
theorem no_perm_with_forbidden_pow_n7
    {π : Equiv.Perm (Fin 7)} (V : Submodule (ZMod 2) (Graph 7))
    (hopt : IsOptimalLinearTriangleCode V)
    (hforb :
      (∃ (k : ℕ) (a b : Fin 7), a ≠ b ∧ π ^ k = Equiv.swap a b) ∨
      (∃ (k : ℕ) (a b c : Fin 7), a ≠ b ∧ b ≠ c ∧ a ≠ c ∧ π ^ k = threeCycle a b c) ∨
      (∃ (k : ℕ) (a b c d : Fin 7),
        a ≠ b ∧ a ≠ c ∧ a ≠ d ∧ b ≠ c ∧ b ≠ d ∧ c ≠ d ∧
        π ^ k = doubleTransposition a b c d)) :
    ¬ IsPermInvariant π V
```

Assumptions: pi is any permutation of Fin 7 and V any optimal linear triangle code on 7 vertices; the hypothesis is that for some natural number k the power pi^k is a transposition, a 3-cycle, or a product of two disjoint transpositions, of distinct vertices. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

TriangleFreeCodeCap.paper_symmetric_seven (TriangleFreeCodeCap/PaperDraft1.lean:84-93) src:d5af4e8769e6c4

```
theorem paper_symmetric_seven :
    (∃ (π : Equiv.Perm (Fin 7)) (V : Submodule (ZMod 2) (Graph 7)),
      orderOf π = 3 ∧ IsOptimalLinearTriangleCode V ∧ IsPermInvariant π V) ∧
    (∃ (π : Equiv.Perm (Fin 7)) (V : Submodule (ZMod 2) (Graph 7)),
      orderOf π = 5 ∧ IsOptimalLinearTriangleCode V ∧ IsPermInvariant π V) ∧
    (∃ (π : Equiv.Perm (Fin 7)) (V : Submodule (ZMod 2) (Graph 7)),
      orderOf π = 6 ∧ IsOptimalLinearTriangleCode V ∧ IsPermInvariant π V)
```

Assumptions: No binders: three existential statements. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

TriangleFreeCodeCap.paper_weights (TriangleFreeCodeCap/PaperDraft1.lean:113-123) src:2df174ba9863ae

```
theorem paper_weights :
    ∃ V : Submodule (ZMod 2) (Graph 8), IsOptimalLinearTriangleCode V ∧
      (∀ G ∈ V, Even (graphWeight G)) ∧
      weightCount V 0 = 1 ∧ weightCount V 2 = 0 ∧ weightCount V 4 = 7 ∧
      weightCount V 6 = 14 ∧ weightCount V 8 = 105 ∧ weightCount V 10 = 336 ∧
      weightCount V 12 = 847 ∧ weightCount V 14 = 1476 ∧ weightCount V 16 = 847 ∧
      weightCount V 18 = 336 ∧ weightCount V 20 = 105 ∧ weightCount V 22 = 14 ∧
      weightCount V 24 = 7 ∧ weightCount V 26 = 0 ∧ weightCount V 28 = 1
```

Assumptions: No binders: one existential statement. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

Table 2. Lean 4 Type-Trap Audit Checklist — Source-Level Approximation (Gate Replay Pending).[a]

Declaration	Hazard	Operand	Category	Audited Disposition & Guard
`IsBalancedPart`	`nat_div`	`/ 2`	intended_convention	n / 2 is natural-number division, the floor of n/2, the size of the smaller side of a balanced bipartition, as Section 2 states.
`IsOptimalLinearTriangleCode`	`classical_choose`	`Classical.choose`	intended_convention	The match is the binomial coefficient n. choose 2 (Nat.choose), the number of pairs; no choice operator occurs.
`IsOptimalLinearTriangleCode`	`nat_sub`	`- n`	proved_guard (sq_div_four_le_choose_two)	The subtrahend n ^ 2 / 4 is at most n. choose 2 for every natural n (sq_div_four_le_choose_two), so the truncated difference is the ordinary one.
`IsOptimalLinearTriangleCode`	`nat_div`	`/ 4`	intended_convention	n^2 / 4 is natural-number division, the floor of n^2/4, written m(n) in the paper.
`UEdge`	`nat_div`	`/ /`	intended_convention	The match is the subtype separator // of { e : Sym2 (Fin n) // not e.IsDiag }; no division occurs.

Declaration	Hazard	Operand	Category	Audited Disposition & Guard
UEdge	nat_div	/ ¬	intended_convention	The match is the subtype separator // of { e : Sym2 (Fin n) // not e.IsDiag }; no division occurs.
no_perm_with_forbidden_pow_n7	real_log_rpow	^ k	intended_convention	pi ^ k is the k-th power in the group of permutations, k a natural number; no real power occurs.
paper_complement	nat_div	/ /	intended_convention	The match is the subtype separator // of the type of inside pairs; no division occurs.
paper_complement	nat_div	/ e	intended_convention	The match is the subtype separator // of the type of inside pairs; no division occurs.
paper_symmetry	nat_mod	% 2	intended_convention	n % 2 = 1 says that n is odd; the modulus 2 is not zero.
paper_upper	real_log_rpow	^ (n.choose 2 - n ^ 2 / 4)	intended_convention	A natural-number power of 2 with a natural exponent; no real power occurs.
paper_upper	classical_choose	Classical.choose	intended_convention	The match is the binomial coefficient n. choose 2 (Nat.choose); no choice operator occurs.
paper_upper	nat_sub	- n	proved_guard (sq_div_four_le_choose_two)	The subtrahend n ^ 2 / 4 is at most n. choose 2 for every natural n (sq_div_four_le_choose_two), so the truncated difference is the ordinary one.
paper_upper	nat_div	/ 4	intended_convention	n^2 / 4 is natural-number division, the floor of n^2/4, written m(n) in the paper.
instDecidableAdj	local_instance_or_notation	instDecidableAdj	intended_convention	Decidability instance for adjacency, by unfolding to equality in ZMod 2; a decidability instance is a subsingleton, so no statement depends on which instance is used.

[a] Covers the specified Lean 4 hazard set (Nat subtraction/division/mod, totalized real/rational division, Real.log/rpow, Finset.range, Classical.choose, local instances); no finite checklist certifies all semantics.

Table 3. Per-Claim Reader's Trust Boundary Ledger (G11 Denominators Derived Mechanically).[a]

Paper Claim	Proof Tier	Lean Decl / Artifact	Actual Axioms & Fidelity	Assumptions & What You Must Trust
Thm. 1: M_{K_3}(7) = M_{C_odd}(7) = 512 and M_{K_3}(8) = M_{C_odd}(8) = 4096. There is a linear triangle code of dimension 9 on 7 vertices and one of dimension 12 on 8 vertices	Lean Kernel	TriangleFreeCodeCap.paper_values	Axioms: Classical.choice, Quot.sound, propext; fidelity: exact	Lean 4 kernel; axioms propext, Classical.choice, Quot.sound; definitions of Section 2; hypotheses as stated.
Thm. 2: For every n >= 0, D_{K_3}(n) = D_{C_odd}(n) = 2^floor(n^2/4)	Written	Written proof	Lean: none; fidelity: n/a	Written proof by the author of this draft (compression, symmetrization, induction on n without a computed base case); the finding folder states the result without a proof. Block 3 of recompute.out illustrates it for n <= 8 and is not needed. Not in Lean: the library has no definition of D_K3, D_Codd or I(G), and the compression, symmetrization and counting steps are not formalized

Paper Claim	Proof Tier	Lean Decl / Artifact	Actual Axioms & Fidelity	Assumptions & What You Must Trust
Thm. 3: Let n >= 0 and let S be a linear family of graphs on n vertices none of whose members contains a triangle. Then dim S <= floor(n^2/4). If n != 3, equality holds if and only if S is the family of all subgraphs of a complete bipartite graph on the n vertices with floor(n^2/4) edges	Written	Written proof	Lean: none; fidelity: n/a	Written proof by the author of this draft; uses Mantel's theorem with its equality case. Block 4 of recompute.out illustrates it for n = 3, 4, 5. Not in Lean: the coordinate argument and the equality case of Mantel's theorem for this graph type are not formalized
Thm. 4: Let V be an optimal linear triangle code on n vertices. (i) If n >= 5, V is not invariant under any transposition. (ii) If n >= 6, V is not invariant under any 3-cycle. (iii) If n >= 7 and n is odd, V is not invariant under any product of two disjoint transpositions	Lean Kernel	`TriangleFreeCodeCap.paper_symmetry`	Axioms: Classical.choice, Quot.sound, propext; fidelity: exact	Lean 4 kernel; axioms propext, Classical.choice, Quot.sound; definitions of Section 2; hypotheses as stated.
Thm. 5: There are a permutation pi of order 7 of the 8 vertices and two optimal linear triangle codes V and W on 8 vertices, both invariant under pi and both containing the complete graph, such that V has exactly 7 members with 4 edges and W has none. No permutation of the vertices maps V to W	Lean Kernel	`TriangleFreeCodeCap.paper_two_codes`	Axioms: Classical.choice, Quot.sound, propext; fidelity: exact	Lean 4 kernel; axioms propext, Classical.choice, Quot.sound; definitions of Section 2; hypotheses as stated.
Prop. 7: For every n >= 0, M_{K_3}(n) <= M_{C_odd}(n) <= 2^(C(n,2) - floor(n^2/4)). If an optimal linear triangle code on n vertices exists, both numbers equal 2^(C(n,2) - floor(n^2/4))	Lean Kernel	`TriangleFreeCodeCap.paper_upper`	Axioms: Classical.choice, Quot.sound, propext; fidelity: exact	Lean 4 kernel; axioms propext, Classical.choice, Quot.sound; definitions of Section 2; hypotheses as stated.
Prop. 8: M_{K_3}(n) = M_{C_odd}(n) for 3 <= n <= 6, with values 2, 4, 16, 64 at n = 3, 4, 5, 6	Lean Kernel	`TriangleFreeCodeCap.paper_small`	Axioms: Classical.choice, Quot.sound, propext; fidelity: exact	Lean 4 kernel; axioms propext, Classical.choice, Quot.sound; definitions of Section 2; hypotheses as stated.
Lem. 9: For every n >= 0, D_{K_3}(n) is the maximum of I(G) over all graphs G on n vertices, where I(G) counts the edge sets of G that contain at most one edge of every triangle of G	Written	Written proof	Lean: none; fidelity: n/a	Written proof by the author of this draft. Not in Lean: the library has no definition of D_K3 or I(G)
Lem. 10: For every n >= 1 some complete multipartite graph on n vertices attains the maximum of I(G)	Written	Written proof	Lean: none; fidelity: n/a	Written proof (standard symmetrization). Not in Lean: the library has no definition of I(G) or of complete multipartite graphs
Lem. 11: Let n >= 1, let G be complete multipartite with part sizes a_1 >= ... >= a_r >= 1, and let v lie in a part of size a_1. Then I(G) <= (1 + sum_{j>=2} (2^(a_j) - 1)) I(G - v)	Written	Written proof	Lean: none; fidelity: n/a	Written proof by the author of this draft. Not in Lean: the library has no definition of I(G)

Paper Claim	Proof Tier	Lean Decl / Artifact	Actual Axioms & Fidelity	Assumptions & What You Must Trust
Cor. 12: For every n >= 0 and for F = K_3 and F = C_ odd, M_F(n) D_F(n) <= 2^C(n,2) is equivalent to M_F(n) <= 2^(C(n,2) - floor(n^2/4)), and for 3 <= n <= 8 it is an equality	Written	Written proof	Lean: none; fidelity: n/a	Written; from the theorem on the dual problem, and for 3 <= n <= 8 from the Lean claims thm_ values and prop_small. Not in Lean: it depends on the written theorem on the dual problem
Lem. 13: Let V be an optimal linear triangle code on n vertices and P a balanced bipartition. The space of all graphs is the direct sum of V and the space of subgraphs of the complete bipartite graph of P, and every prescribed set of inside pairs of P is the set of inside pairs of some member of V	Lean Kernel	`TriangleFreeCodeCap.paper_ complement`	Axioms: Classical.choice, Quot.sound, propext; fidelity: exact	Lean 4 kernel; axioms propext, Classical.choice, Quot.sound; definitions of Section 2; hypotheses as stated.
Lem. 14: Let H be a group of permutations of the n vertices, V an H-invariant optimal linear triangle code, and P a balanced bipartition whose set E_P of cross pairs is H-invariant. Then dim(V cap Fix(H)) + orb_H(E_P) = orb_H(all pairs). If S is an H-invariant set of pairs that forms a graph without a triangle, then orb_H(S) <= orb_H(E_P)	Lean Kernel	`TriangleFreeCodeCap.paper_ orbit`	Axioms: Classical.choice, Quot.sound, propext; fidelity: exact	Lean 4 kernel; axioms propext, Classical.choice, Quot.sound; definitions of Section 2; hypotheses as stated.
Prop. 15: Let V be an optimal linear triangle code on 7 vertices and pi a permutation of the vertices. If some power of pi is a transposition, a 3-cycle or a product of two disjoint transpositions, then V is not invariant under pi	Lean Kernel	`TriangleFreeCodeCap.no_perm_ with_forbidden_pow_n7`	Axioms: Classical.choice, Quot.sound, propext; fidelity: exact	Lean 4 kernel; axioms propext, Classical.choice, Quot.sound; definitions of Section 2; hypotheses as stated.
Prop. 16: For each r in {3, 5, 6} there are a permutation of order r of the 7 vertices and an optimal linear triangle code on 7 vertices that is invariant under it	Lean Kernel	`TriangleFreeCodeCap.paper_ symmetric_seven`	Axioms: Classical.choice, Quot.sound, propext; fidelity: exact	Lean 4 kernel; axioms propext, Classical.choice, Quot.sound; definitions of Section 2; hypotheses as stated.
Prop. 17: A permutation of 7 vertices leaves some optimal linear triangle code on 7 vertices invariant if and only if its cycle type is 1^7, 2^3 1, 3^2 1, 5 1^2 or 6 1	Cert Only	recompute.py (361a86aa5391) [regenerable; replay pending]	External cert only; fidelity: n/a	Written reduction by the author of this draft to the Lean claims prop_forbidden, prop_sevensym and thm_values, and the exclusion of all 10660 submodules by recompute.py, block 6. Not in Lean: the kernel checks neither the list of submodules nor its completeness

Paper Claim	Proof Tier	Lean Decl / Artifact	Actual Axioms & Fidelity	Assumptions & What You Must Trust
Prop. 18: There is an optimal linear triangle code on 8 vertices all of whose members have an even number of edges, with 1 member each of weights 0 and 28, none of weights 2 and 26, 7 each of weights 4 and 24, 14 each of weights 6 and 22, 105 each of weights 8 and 20, 336 each of weights 10 and 18, 847 each of weights 12 and 16, and 1476 of weight 14	Lean Kernel	`TriangleFreeCodeCap.paper_weights`	Axioms: Classical.choice, Quot.sound, propext; fidelity: exact	Lean 4 kernel; axioms propext, Classical.choice, Quot.sound; definitions of Section 2; hypotheses as stated.

[a] Allowed pure-kernel axioms: `propext`, `Classical.choice`, `Quot.sound`. Actual axioms printed above are read directly from verified outputs. Provenance nonce: 9523a87095af | Lean mode: committed_receipt | Closure SHA-256: 16cf59f57e45 | Git anchor: aaf43e0528f3 | Axioms SHA-256: e923235c7cf5 | Fidelity SHA-256: 988b7d838fb4.

6. PRIOR WORK AND OPEN QUESTIONS

The source. Alon, Gujgiczer, Körner, Milojević and Simonyi [1] define $M_{\mathcal{F}}(n)$ and $D_{\mathcal{F}}(n)$ and prove $M_{\mathcal{F}}(n)D_{\mathcal{F}}(n) \le 2^{\binom{n}{2}}$ (Lemma 1). The proof of Proposition 7 is the argument of their Remark 1, specialized to bipartite graphs, and gives their Proposition 9 for K_3 and C_{odd}. They found the optimal codes for $3 \le n \le 6$ and the odd-cycle family at $n = 7$. The numbering is that of version 3 of the arXiv paper, which we read; we have not compared it with the journal version. Asymptotic results in the same direction are due to Bai, Gao, Ma and Wu [3]. Alon's lectures on graph codes [2] say about the largest families in which every difference contains a copy of a fixed graph L that determining their size precisely "is open, even for relatively simple graphs L like a triangle". Versteegen [12] compares linear and general codes in which no difference is equal to a given graph.

The count of Theorem 2. Balogh [4] fixed a 2-coloring of the edges of a complete graph K_{k+1} and counted 2-colorings of the edges of an n-vertex graph without a copy colored in exactly this way. His abstract determines the maximum for $n > n_0(k)$ when both colors are used in the pattern. The formula is missing from the copy of the abstract that we saw; according to [6] the maximum is the trivial lower bound, that is $2^{t_k(n)}$, where $t_k(n)$ is the largest number of edges of an n-vertex graph without K_{k+1}. We have not read the article [to check]. Take $k = 2$ and the pattern with two red edges and one blue edge. A set in $\mathcal{I}(G)$, colored red with the rest of $E(G)$ blue, gives a coloring without that pattern, so $I(G) \le 2^{m(n)}$ for all sufficiently large n, and with Lemma 9 the value of $D_{K_3}(n)$ for such n follows. The proof in Section 3 holds for every n and uses no regularity lemma. Gupta, Pehova, Powierski and Staden [6] survey such counting problems; Pikhurko, Staden and Yilma [11] use symmetrization in this setting. Hoppen, Lefmann and Schmidt [7] treat two-colored triangles without fixed colors, for large n. Norin and Sun [10] study the largest set in $\mathcal{I}(G)$, not the number of such sets.

Linear families. Beers and Botnan [5] show that the first Betti number of the flag complex of a graph on n vertices is at most that of the balanced complete bipartite graph, with a statement on the case of equality [to check]; their proof also removes an edge of a triangle. The coordinates of a linear family satisfy the cocycle relation $x_e + x_f + x_g = 0$ of Section 3 on the union of its members, so the dimension bound of Theorem 3 can also be obtained from their inequality. They do not discuss families of graphs. They say that their bound on Betti numbers of flag complexes also follows from a theorem of Kozlov [8] [to check].

What was searched. The prior-work search is reported in `prior-work-search/search-report.md`. The checks of the search tools read the same log under the name `searchlog.md`, which differs only in one column heading that the tools require; both checks pass, and their output is in `prior-work-search/check.out`. It covers Google Scholar through a keyed provider, CORE, OEIS, arXiv, zbMATH Open, Semantic Scholar, MathOverflow and Mathematics Stack Exchange, Zenodo, DataCite, GitHub, and the issues and pull requests of the Formal Conjectures repository. Not searched: OpenAlex, which refused requests; the papers citing [5] on Semantic Scholar; the full text of Balogh's article and of Kozlov's; MathSciNet; books; other paywalled sources. The values of $M_{K_3}(7)$, $M_{K_3}(8)$ and $M_{C_{\text{odd}}}(8)$, the exact dual value for every n, the description of the linear extremal families as families of graphs, and Theorems 4 and 5 were not found in the searches logged.

Open questions. (i) Is $M_{K_3}(9) = 2^{36-20}$, that is, is there a triangle code with 2^{16} members on 9 vertices? (ii) Problem 4 of [1] for all n. (iii) Is every family that attains $D_{K_3}(n)$, for $n \ge 4$, a translate of the family of all subgraphs of a balanced complete bipartite graph? (iv) Are the balanced complete bipartite graphs the only graphs with $I(G) = 2^{m(n)}$ for every

$n \geq 4$? Section 4 confirms this for $4 \leq n \leq 6$. (v) Can the written proofs be moved into Lean? (vi) Review by a human mathematician.

7. DATA AVAILABILITY, REPRODUCIBILITY, AND USE OF AI

In the `mathdisc` repository the draft folder `papers/alon-korner-triangle-free-code-capacity/draft1/` has the manuscript, `recompute.py` with its output, the figure scripts, the search log and the ledgers; its `README.md` gives the commands that rerun everything. The finding folder `findings/alon-korner-triangle-free-code-capacity/` has the Lean library in `lean/`, with `PaperDraft1.lean`, `VerifyDraft1.lean`, the axiom printout and the receipt.

What differs from the record of the finding. The finding folder (`README.md`, `finding.json`) states the results on the dual problem without proofs or programs and refers to a dossier that is not in the folder. The following are supplied by this draft. (1) The proof of Theorem 2, with Lemmas 9, 10 and 11; it needs no computation. (2) The proof of Theorem 3. (3) The proof of Proposition 17, with the enumeration of block 6 of `recompute.py`; the README of the Lean library of the finding (`lean/TriangleFreeCodeCap/README.md`, line 50) lists the exclusion of a 7-cycle as one of three exclusions at $n = 7$ that were established outside Lean, by SAT refutations checked with `drat-trim` and by enumerations of modules, without saying which method gave which; neither is in the folder. (4) Corollary 12. The finding lists $M_{C_{\mathrm{odd}}}(7)$ among its results without naming Proposition 17 of [1], and does not name Balogh [4]; it names Beers and Botnan. It reports Lean coverage by declarations; of the 17 claims of this draft listed in Section 5, 10 are in Lean, the 6 on the dual problem have written proofs, and Proposition 17 rests on a computation.

Use of AI. AI research agents built and directed by Tyler B. Hughes found the results, wrote the Lean library, the proofs, the programs and this text, and checked them. The author agent of this draft is a model of one company (Anthropic); the statement auditor and the blind referee are two separate sessions of one model of another company (OpenAI). Their names are in `README.md`. No human mathematician has yet refereed the proof. The Lean kernel checks the statements marked Lean against the definitions printed in Section 5. It does not check that these definitions say what the text says; the statement audit of that section compares them.

REFERENCES

(1) Alon, N.; Gujgiczer, A.; Körner, J.; Milojević, A.; Simonyi, G. Structured codes of graphs. *SIAM J. Discrete Math.* **2023**, *37*, 379–403. arXiv:2202.06810.

(2) Alon, N. Graph-codes: questions, results and methods. *Jpn. J. Math.* 2026. doi:10.1007/s11537-026-2536-1. [to check]

(3) Bai, B.; Gao, Y.; Ma, J.; Wu, Y. Phase transitions of structured codes of graphs. arXiv:2307.08266, 2023. [to check]

(4) Balogh, J. A remark on the number of edge colorings of graphs. *European J. Combin.* **2006**, *27*, 565–573. doi:10.1016/j.ejc.2004.12.004. [to check]

(5) Beers, L.; Botnan, M. B. Extremal Betti numbers and persistence in flag complexes. arXiv:2502.21294, 2025. [to check]

(6) Gupta, P.; Pehova, Y.; Powierski, E.; Staden, K. A framework for the generalised Erdős–Rothschild problem and a resolution of the dichromatic triangle case. arXiv:2502.12291, 2025. [to check]

(7) Hoppen, C.; Lefmann, H.; Schmidt, D. R. Edge-colorings avoiding patterns in a triangle. arXiv:2209.06991, 2022. [to check]

(8) Kozlov, D. N. Convex hulls of f- and β-vectors. *Discrete Comput. Geom.* **1997**, *18*, 421–431. doi:10.1007/PL00009326. [to check]

(9) Mantel, W. Vraagstuk XXVIII. *Wiskundige Opgaven* **1907**, *10*, 60–61. Record: https://zbmath.org/2643703 (JFM 38.0270.01). [to check]

(10) Norin, S.; Sun, Y. R. Triangle-independent sets vs. cuts. arXiv:1602.04370, 2016. [to check]

(11) Pikhurko, O.; Staden, K.; Yilma, Z. B. The Erdős–Rothschild problem on edge-colourings with forbidden monochromatic cliques. *Math. Proc. Cambridge Philos. Soc.* **2017**, *163*, 341–356. arXiv:1605.05074. [to check]

(12) Versteegen, L. Upper bounds for linear graph codes. arXiv:2310.19891, 2023. [to check]