MATHDISCOVERIES
Mathematical Preprint

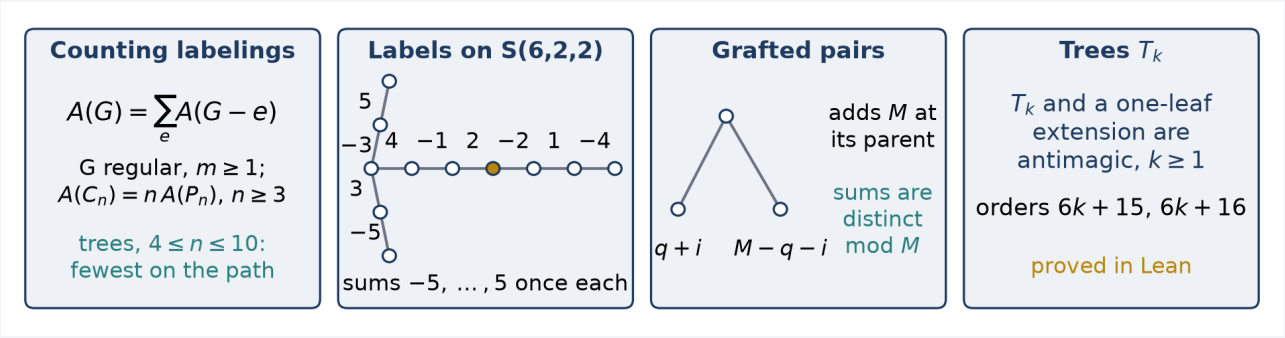
Working Draft

# A Deletion Identity for Antimagic Labelings of Regular Graphs, Counts for Small Trees, and a Lean Check of Trees Grafted on $S(6k,2,2)$

Tyler B. Hughes*,†

†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 graph with $m$ edges is antimagic if its edges can be labeled $1,\dots,m$ with distinct vertex sums. Hartsfield and Ringel conjectured in 1990 that every connected graph other than $K_2$ is antimagic; it is open for trees. For every regular graph $G$ with at least one edge, the number $A(G)$ of antimagic labelings satisfies $A(G)=\sum_e A(G-e)$; in particular $A(C_n)=nA(P_n)$ for $n \geq 3$. Among trees on 4 to 10 vertices, only the path has the fewest antimagic labelings and only the star has nothing but antimagic labelings. By results of Mitchem and Simoson and of Keene and Simoson on super edge-graceful labelings (labels $\pm1,\dots,\pm q$, vertex sums $-q,\dots,q$), the trees $T_k$ grafted on $S(6k,2,2)$ are edge-graceful, hence antimagic. We give an explicit one on $S(6k,2,2)$ and prove in Lean that $T_k$, of order $6k+15$, is antimagic and that adding one leaf at a suitable vertex of $T_k$ gives an antimagic tree of order $6k+16$, for every $k \geq 1$. Neither $T_k$ nor $T_k$ with a leaf at the middle of its long leg satisfies the hypotheses of the classes of antimagic trees we list. AI research agents built and directed by Tyler B. Hughes generated and checked this work. No human mathematician has yet refereed the proof.

**Verification Scope (G11):** WARN: Human referee: pending
Main results: 2 of 3 in Lean Kernel (67%) · 1 Cert Only
Abstract claims: 4 of 6 in Lean Kernel (67%) · 1 Cert Only · 1 written

**Full Trust Boundary:** Section 5.

**MSC 2020:** 05C78, 05C05, 05A15, 68V20 **Keywords:** antimagic labeling, super edge-graceful labeling, trees, edge-graceful, Combinatorial Nullstellensatz, Lean 4

## 1. INTRODUCTION AND MAIN RESULTS

Label the $m$ edges of a graph with the numbers $1,\dots,m$, each used once, and give every vertex the sum of the labels on its edges. The labeling is *antimagic* if no two vertices get the same sum. Hartsfield and Ringel conjectured in 1990 that every connected graph other than the single edge $K_2$ has such a labeling, and asked the same for trees [1] [to check]. Gallian's survey [2] [to check] records both questions as open and lists the classes of trees known to be antimagic. Computer searches have confirmed the statement for every tree on 3 to 31 vertices [3] [to check].

**Counting antimagic labelings.** Write $A(G)$ for the number of antimagic labelings of $G$. The conjecture says $A(G) \geq 1$; the next statement relates the counts of a regular graph and of its subgraphs with one edge fewer.

**Theorem 1** (Deletion identity for regular graphs). *Let $G$ be a finite $r$-regular graph with at least one edge. Then $A(G)=\sum_{e\in E(G)} A(G-e)$, where $G-e$ has the vertex set of $G$.*

For edge-transitive regular graphs the terms are equal (Corollary 20); for the cycle $C_n$ this gives $A(C_n)=nA(P_n)$ (Corollary 21), an equality that the repository of [28] [to check] reports as measured for small $n$.

**Proposition 2** (Extremal trees on at most ten vertices). *(a) For $4 \leq n \leq 10$, the path $P_n$ is the only tree on $n$ vertices with the fewest antimagic labelings. (b) For $4 \leq n \leq 10$, the star $K_{1,n-1}$ is the only tree on $n$ vertices all of whose $(n-1)!$ bijective labelings are antimagic.*

**Proposition 3** (Counts for paths and small graphs). *(a) $A(P_n)=2,6,16,92,352,1910,10892$ for $n=4,\dots,10$. (b) Each of the 141 connected graphs on 3 to 6 vertices is antimagic, and $A(G) \geq 2$ for each of the 89 among them with at most 5 vertices, or with 6 vertices and at most 8 edges.*

Part (b) only confirms, with counts, what exhaustive searches show for far larger graphs [29] [to check].

**Super edge-graceful trees.** Let $T_1$ be a tree with $2q$ edges, and so with $2q+1$ vertices. A *super edge-graceful labeling* of $T_1$ labels its edges bijectively with $\pm 1, \dots, \pm q$ so that the vertex sums are exactly $-q, \dots, q$ (Mitchem and Simoson [5]). The Lean library of this paper calls the same object a *balanced signed antimagic* (BSA) labeling, and we keep its notation in Section 2. Mitchem and Simoson show that a super edge-graceful tree of odd order is edge-graceful: adding $2q+1$ to each negative label gives a bijection onto $\{1, \dots, 2q\}$ whose vertex sums are distinct modulo $2q+1$ ([5, Theorem 1, Corollary 2]). They also show that appending two edges at one vertex of a super edge-graceful tree, labeled $\pm(q+1)$, gives a super edge-graceful tree ([5, Proposition 3]). Kaplan, Lev and Roditty [15] [to check], Liang, Wong and Zhu [16] [to check], and Lozano, Mora, Seara and Tey [17] label antimagic trees with the same device: sibling edges get labels that add to the modulus. Theorem 4 states the two steps of Mitchem and Simoson together, in the form that the Lean library proves. For a tree grown from $T_1$ by attaching the pairs one at a time to vertices already present, which covers the trees $T_k$, it follows from Proposition 3 and Theorem 1 of [5]. Definition 11 also allows pairs attached to each other in a cycle (three pairs, each attached to a child of the next, give a triangle with three pendant vertices, apart from the core); these graphs are not trees, and for them Theorem 4 extends the same argument.

**Theorem 4** (Grafting on a super edge-graceful core; Mitchem and Simoson for trees). *Let $\psi$ be a BSA labeling with parameter $q$ of a graph $T_1$, and let $T$ be an even-children grafting of $T_1$ with $r$ pairs (Definition 11). Put $m = 2q + 2r$ and $M = m + 1$. Then the lifted labeling $\varphi$ of Definition 11 is a $\mathbb{Z}_M$-antimagic labeling of $T$.*

A $\mathbb{Z}_M$-antimagic labeling is antimagic (Corollary 14). For a tree of odd order it is an edge-graceful labeling [2], so every edge-graceful tree is antimagic.

**Spiders with three legs.** Keene and Simoson prove that every spider of odd order with three legs is edge-graceful, and their proof builds super edge-graceful labelings [8, Theorem 11 and p. 58]; this includes the spiders $S(6k, 2, 2)$, with one leg of length $6k$ and two of length 2. We use the explicit labeling below, whose sum vanishes at the middle vertex $x_{3k}$ of the long leg.

**Proposition 5** (An explicit labeling of $S(6k, 2, 2)$). *For every $k \geq 1$ the labeling $\psi_k$ of Definition 12 is a BSA labeling of $S(6k, 2, 2)$ with parameter $3k + 2$.*

**The trees $T_k$.** The tree $T_k$ of Definition 12 is $S(6k, 2, 2)$ with five pairs of edges grafted onto it: a binary tree of depth two at the end of the long leg and a pair of leaves at the end of each short leg. By Proposition 5 and Theorem 4, or by the results of Mitchem and Simoson and of Keene and Simoson, $T_k$ is edge-graceful. The next theorem adds the leaf of Kaplan, Lev and Roditty and of Liang, Wong and Zhu, and records the Lean proof.

**Theorem 6** (The trees $T_k$ and their leaf extensions). *For every $k \geq 1$, $T_k$ is a tree with $6k + 15$ vertices and is antimagic, and there is a vertex $v_0$ of $T_k$ such that the graph obtained from $T_k$ by attaching one pendant leaf to $v_0$ is a tree with $6k + 16$ vertices and is antimagic.*

Proposition 19, which has a written proof, identifies $v_0$: it is $x_{3k}$, and $T'_k$ denotes $T_k$ with a leaf at $x_{3k}$. The classes of antimagic trees recorded in Gallian's survey that we checked (Section 6) do not contain these trees; one of the following invariants excludes each.

**Proposition 7** (Structural invariants of $T_k$ and $T'_k$). *For every $k \geq 1$ the trees $T_k$ and $T'_k$ have maximum degree 3. $T_k$ has $6k + 1$ vertices of degree 2, six of degree 3 and eight leaves; $T'_k$ has $6k$ vertices of degree 2, seven of degree 3 and nine leaves. In both trees two vertices of degree 2 are adjacent, two vertices of degree 3 are adjacent, the vertices of even degree induce a forest with at least three components, the branch vertex $c = x_0$ of $S(6k, 2, 2)$ has degree 3 and no leaf neighbor, no path contains all the vertices of degree 3, and the diameter is $6k + 5$.*

**Corollary 8** (Outside the listed classes). *For every $k \geq 1$, neither $T_k$ nor $T'_k$ satisfies the hypotheses of any class in Table 5.*

The orders $6k + 15$ and $6k + 16$ are at most 31 only for $k = 1, 2$, so for $k \geq 3$ these trees are beyond the exhaustive searches as well.

**How the results are checked.** Main results: 2 of 3 in Lean (67%) · 1 by certificate. Section 5 lists every numbered statement with its tier and Lean declaration. Propositions 16 and 17, and Theorem 4 for trees grown by attaching pairs one at a time, are known results; the Lean library proves them, Theorem 4 in the generality of Definition 11, which extends the same argument. None is counted as a main result. Proposition 7 has a written proof, which `recompute.py` checks for $k \leq 25$; Propositions 2 and 3 rest on the exhaustive enumeration of `recompute.py`. Section 3.4 adds Lean-checked facts on the Combinatorial Nullstellensatz for this problem. Figure 1 shows how the parts fit together.

**Concept in Action (verbatim in `recompute.py`)**

```
q, M = 5, 21            # T_1: core S(6,2,2), r = 5 pairs
E = [(i, i + 1, x) for i, x in enumerate([4, -1, 2, -2, 1, -4])]
E += [(0, 7, -3), (7, 8, 5), (0, 9, 3), (9, 10, -5)]
for i, p in enumerate([6, 11, 12, 8, 10], 1):
    E += [(p, 9 + 2 * i, q + i), (p, 10 + 2 * i, M - q - i)]
f = [(u, v, x % M) for u, v, x in E]   # the lifted labels
s = [sum(x for u, v, x in f if w in (u, v)) for w in range(M)]
assert sorted(x for u, v, x in f) == list(range(1, M))
assert sorted(x % M for x in s) == list(range(M))
assert s[3] % M == 0
```

The program builds $T_1$: vertices $0, \dots, 6$ are the long leg $c = x_0, \dots, x_6$ of $S(6, 2, 2)$, vertices 7 to 10 the short legs, and 11 to 20 the grafted vertices. The signed labels of the core are those of $\psi_1$; a negative label $x$ becomes $M + x$. The checks say that the labels are $1, \dots, 20$ and that the 21 vertex sums leave all 21 residues modulo 21, so the sums are distinct; the sum at $x_3$ is divisible by 21, which is where the leaf of $T'_1$ goes.

| (i) What was found | (ii) How it was found | (iii) Basis |
|---|---|---|
| A(G) = sum of A(G - e), G regular with an edge | shift every label by one | Lean |
| fewest labelings: the path, n = 4 to 10 | all labelings of all 199 trees | Certificate |
| $T_k$ and a one-leaf extension of it are antimagic | grafting, then a leaf at residue zero | Lean |
| the leaf goes at $x_{3k}$, so $T'_k$ is antimagic | the only zero residue | Written |
| explicit labeling of S(6k, 2, 2) | alternating labels, one gap | Lean |
| grafting on a super edge-graceful core | for trees: Mitchem, Simoson | Lean |
| $T_k$, $T'_k$ outside the listed antimagic classes | degree and path invariants | Written |
| no box monomial for trees ($n \geq 2$); $c_{\mathrm{box}}$ table | degrees; Vandermonde sums | Lean |
| is every tree of odd order super edge-graceful? | open question (1) | Unsettled |

**Figure 1.** Discovery and proof architecture. Left: what was found. Middle: the way each statement was found and checked. Right: the basis of each statement. The grafting row restates the result of Mitchem and Simoson for trees. The unsettled row is a question, not a claim of this paper.

**Roadmap.** Section 2 fixes the definitions in the form of the Lean library. Section 3 proves the results on trees and on counts, and gives the facts on the Combinatorial Nullstellensatz in Section 3.4. Section 4 records the computations, Section 5 lists what the Lean library proves, and Section 6 compares the results with the literature.

## 2. DEFINITIONS, NOTATION, AND SETUP

Graphs are simple, with finitely many edges and, except in Corollary 15, finitely many vertices. For a graph $G$ we write $V$ for its vertex set, $E$ for its edge set and $m = |E|$.

**Definition 9** (Labelings). An *edge labeling* of $G$ is a map $f : E \to \mathbb{N}$. It is *bijective* if it maps $E$ bijectively onto $\{1, \dots, m\}$. The *vertex sum* of $f$ at $v$ is $s_f(v) = \sum_{e \ni v} f(e)$. An *antimagic labeling* is a bijective labeling whose vertex sums are pairwise distinct, and $G$ is *antimagic* if it has one. A $\mathbb{Z}_{m+1}$*-antimagic labeling* is a bijective labeling such that $v \mapsto s_f(v) \bmod (m+1)$ is a bijection from $V$ onto $\mathbb{Z}_{m+1}$; in particular $|V| = m+1$. The number of antimagic labelings of $G$ is $A(G)$.

**Definition 10** (Signed labelings and BSA cores). For $q \geq 1$ let $S_q = \{-q, \dots, -1\} \cup \{1, \dots, q\}$ and $[-q, q] = \{-q, \dots, q\}$. A *signed labeling* of $G$ is a map $\psi \colon E \to \mathbb{Z}$, with *signed vertex sums* $\sigma_\psi(v) = \sum_{e \ni v} \psi(e)$. A *BSA labeling with parameter* $q$ of $G$ is a signed labeling such that $q \geq 1$, $G$ is a tree with $2q$ edges and $2q+1$ vertices, $\psi$ maps $E$ bijectively onto $S_q$, and $\sigma_\psi$ maps $V$ bijectively onto $[-q, q]$. A tree is a *BSA core with parameter* $q$ if it has such a labeling. For a tree of odd order this is a super edge-graceful labeling [6].

**Definition 11** (Even-children grafting and the lifted labeling). Let $T_1$ be a graph on $V_1$, let $r \geq 0$, and let $V_2$ be a set with a bijection $(i, b) \mapsto c(i, b)$ from $\{1, \dots, r\} \times \{0, 1\}$ onto $V_2$, and a map $p$ from $\{1, \dots, r\}$ to $V_1 \sqcup V_2$. A graph $T$ on $V_1 \sqcup V_2$ is an *even-children grafting of* $T_1$ *with* $r$ *pairs* if (i) $p(i) \neq c(i, b)$ for all $i, b$; (ii) there are no two indices $(i, b)$, $(j, b')$ with $p(i) = c(j, b')$ and $p(j) = c(i, b)$; and (iii) two vertices are adjacent in $T$ if and only if both lie in $V_1$ and are adjacent in $T_1$, or they are $p(i)$ and $c(i, b)$ for some $(i, b)$. The edges $p(i)c(i, 0)$ and $p(i)c(i, 1)$ form the *i-th pair*. If $\psi$ is a signed labeling of $T_1$ and $M = 2q + 2r + 1$, the *lifted labeling* $\varphi$ of $T$ is $\varphi(e) = \psi(e)$ if $e \in E(T_1)$ and $\psi(e) > 0$; $\varphi(e) = \max(M + \psi(e), 0)$ if $e \in E(T_1)$ and $\psi(e) \leq 0$ (for a BSA labeling this is $M + \psi(e)$); $\varphi(p(i)c(i, 0)) = q + i$; and $\varphi(p(i)c(i, 1)) = M - q - i$. The *leaf extension* $G^{+v_0}$ of a graph $G$ at a vertex $v_0$ is $G$ with one added vertex $\ell$ and the edge $v_0\ell$.

Every vertex of $V_2$ is the endpoint of exactly one pair edge from its parent, and its children come in pairs, which explains the name.

**Definition 12** (Named graphs and labelings). (a) $P_n$ has vertices $0, \dots, n-1$ and edges $\{i, i+1\}$. $C_n$ ($n \geq 3$) has vertices $0, \dots, n-1$ and edges $\{i, i+1 \bmod n\}$. $K_n$ is the complete graph on $\{0, \dots, n-1\}$, and $K_{1,m}$ is the star with $m$ leaves. (b) $S(2^{2s+1})$ has a center $c$ and legs $c\,u_j\,w_j$ for $j = 0, \dots, 2s$. (c) $S(6k, 2, 2)$ has the long leg $c = x_0, x_1, \dots, x_{6k} = w_0$ and the short legs $c\,u_1\,w_1$ and $c\,u_2\,w_2$. Its labeling $\psi_k$ is as follows. Let $\mu_j = j$ for $1 \leq j \leq 2k$ and $\mu_j = j+1$ for $2k < j \leq 3k$, so that $\{\mu_1, \dots, \mu_{3k}\} = \{1, \dots, 3k+1\} \setminus \{2k+1\}$. For $1 \leq j \leq 3k$ let $z_j = \mu_{3k-(j-1)/2}$ if $j$ is odd and $z_j = -\mu_{j/2}$ if $j$ is even. Then $\psi_k(x_{j-1}x_j) = z_j$ and $\psi_k(x_{6k-j}x_{6k-j+1}) = -z_j$ for $1 \leq j \leq 3k$; $\psi_k(cu_1) = -(2k+1)$, $\psi_k(u_1w_1) = 3k+2$, $\psi_k(cu_2) = 2k+1$ and $\psi_k(u_2w_2) = -(3k+2)$. (d) $T_k$ is the even-children grafting of $S(6k, 2, 2)$ with $r = 5$ pairs and parents $p(1) = w_0$,

$p(2) = c(1,0), p(3) = c(1,1), p(4) = w_1, p(5) = w_2$. $T'_k$ is the leaf extension of $T_k$ at $x_{3k}$. (e) A graph is *r-regular* if every vertex has degree $r$, and *edge-transitive* if its automorphism group is transitive on edges.

For $k = 1$ the labels of the long leg, read from $c$, are $4, -1, 2, -2, 1, -4$, and the short legs carry $-3, 5$ and $3, -5$; this is the core of the Python box. Definitions for Section 3.4 follow.

**Definition 13** (Polynomials of the Combinatorial Nullstellensatz). Let $G$ have vertices $0, \dots, n-1$ and edges indexed $e_1, \dots, e_m$, with variables $x_1, \dots, x_m$. Put $s_v(x) = \sum_{j:\, v \in e_j} x_j$, $P_G(x) = \prod_{0 \le u < v < n} (s_v(x) - s_u(x))$ and $\Delta_m(x) = \prod_{1 \le i < j \le m} (x_j - x_i)$. When $m = n$, the *box coefficient* $c_{\mathrm{box}}(G)$ is the coefficient of $x_1^{n-1} \cdots x_n^{n-1}$ in $\Delta_n P_G$; for a polynomial $F$ with integer coefficients in $n$ variables, $c_{\mathrm{box}}(F)$ is the same coefficient of $\Delta_n F$. The *degree gap* is $\Gamma(n,m) = \binom{n}{2} - \binom{m}{2}$.

The box coefficient depends on the indexing of vertices and edges only through its sign. We use the edge orders of Table 1, which are those of the Lean library.

## 3. PROOF OF THE MAIN RESULTS

### 3.1 Grafting.

*Proof of Theorem 4.* The positive values $1, \dots, q$ of $\psi$ stay, the negative values $-q, \dots, -1$ become $M-q, \dots, M-1 = m$, and the pairs take $q+1, \dots, q+r$ and $M-q-r, \dots, M-q-1$. Since $M - q - r = q + r + 1$, these four blocks partition $\{1, \dots, m\}$, so $\varphi$ is bijective. Modulo $M$, $\varphi$ agrees with $\psi$ on $E(T_1)$, and the two edges of a pair add to $M \equiv 0$. A vertex $v \in V_1$ is the parent of some pairs and lies on no other pair edge, so $s_\varphi(v) \equiv \sigma_\psi(v)$. A vertex $c(i,b)$ lies on one edge from its parent, with label $q+i$ or $M-q-i$, and on whole pairs, so $s_\varphi(c(i,b)) \equiv \pm(q+i)$. The residues are therefore $[-q, q]$ from $V_1$ and $\pm(q+1), \dots, \pm(q+r)$ from $V_2$, each once: this is $[-(q+r), q+r]$, a complete set of residues modulo $M = 2(q+r)+1$, and $|V_1 \sqcup V_2| = M$. □

**Corollary 14** (Grafted cores are antimagic). *If $T_1$ is a BSA core with parameter $q$ and $T$ is an even-children grafting of $T_1$, then $T$ is antimagic.*

*Proof.* Sums that differ modulo $M$ differ. □

**Corollary 15** (Leaf extension at the zero residue). *Let $G$ be a connected graph with $m \ge 2$ edges, on a vertex set not assumed finite, and $f$ a $\mathbb{Z}_{m+1}$-antimagic labeling of $G$. Then there is a vertex $v_0$ with $s_f(v_0) \equiv 0 \pmod{m+1}$ such that $G^{+v_0}$ is antimagic.*

*Proof.* Let $v_0$ be the vertex whose sum is 0 modulo $M = m+1$, and give the edge $v_0\ell$ the label $M$. The sums become $s_f(v)$ for $v \ne v_0$, which are distinct and not divisible by $M$; $s_f(v_0) + M$, which is divisible by $M$; and $M$ at $\ell$. The last two differ because $s_f(v_0) \ge 1$: $G$ is connected with at least two edges, so $v_0$ lies on an edge. □

This is the even-order step of [15, 16] [to check], as described in [17].

**Proposition 16** (Paths; Mitchem and Simoson). *For every $t \ge 1$, $P_{2t+1}$ is a BSA core with parameter $t$.*

*Proof.* Label the edge $\{j, j+1\}$, $0 \le j < t$, with $t - j/2$ if $j$ is even and $-(j+1)/2$ if $j$ is odd, and the edge $\{2t-1-j, 2t-j\}$ with the negative of the label of $\{j, j+1\}$. The labels are $S_t$, and the sum at vertex $v$ is $t - v$. The statement is Corollary 10 of [5]; it is also contained in [7] [to check]. □

**Proposition 17** (Subdivided stars; Mitchem and Simoson). *For every $s \ge 1$, $S(2^{2s+1})$ is a BSA core with parameter $2s+1$.*

*Proof.* Put $q = 2s+1$. Give the two edges $cu_j$ and $u_j w_j$ of the leg $j$ the labels $q$ and $-q$ if $j = 0$; $-(q-2i)$ and $2i$ if $j = 2i-1$; and $q-2i$ and $-2i$ if $j = 2i$, where $1 \le i \le s$. The labels are $\pm q$, the odd numbers $\pm 1, \dots, \pm(q-2)$ and the even numbers $\pm 2, \dots, \pm 2s$, each once. The leaves get $-q$ and $\pm 2, \dots, \pm 2s$; the midpoints get 0, the numbers $4i - q$ and the numbers $q - 4i$, which are the odd numbers of absolute value at most $2s-1$; and the center gets $q$. The statement is the case of Theorem 5 of [5] with $2s+1$ legs of length 2; it is also contained in Lemmas 4.9 and 4.10 of [9] and in Theorem 3.5 of [6]. □

### 3.2 The spiders $S(6k, 2, 2)$ and the trees $T_k$.

**Lemma 18** (Sums along the long leg). *For $k \ge 1$ and $0 \le j \le 6k$, the signed vertex sum of $\psi_k$ at $x_j$ is $3k+1$ if $j = 0$; $3k+1-j$ if $1 \le j < 2k$; $3k - j$ if $2k \le j < 3k$; $0$ if $j = 3k$; $-(j-3k)$ if $3k < j \le 4k$; $-(j+1-3k)$ if $4k < j < 6k$; and $-(3k+1)$ if $j = 6k$.*

*Proof.* At $c$ the three labels are $z_1 = \mu_{3k} = 3k+1$, $-(2k+1)$ and $2k+1$. At $x_{6k}$ the only label is $-z_1$. For $1 \le j < 3k$ the sum at $x_j$ is $z_j + z_{j+1} = \mu_a - \mu_b$, where $a + b = 3k+1$ and $b = (j+1)/2$ if $j$ is odd, and $a + b = 3k$ and $b = j/2$ if $j$ is even; in both cases $a - b = 3k - j$. If $j < 2k$ then $a > 2k \ge b$, so $\mu_a - \mu_b = a + 1 - b = 3k + 1 - j$; if $2k \le j < 3k$ then $a, b \le 2k$ and $\mu_a - \mu_b = 3k - j$. At $x_{3k}$ the two labels are $z_{3k}$ and $-z_{3k}$. The labels beyond $x_{3k}$ are the negatives of the labels before it in reverse order, so the sum at $x_{6k-j}$ is minus the sum at $x_j$. □

*Proof of Proposition 5.* Each $\mu_r$ is $\pm z_j$ for exactly one $j \le 3k$ (with $+$ for odd $j$ and $-$ for even $j$), so the long leg carries $\pm\mu_1, \dots, \pm\mu_{3k}$, that is $\pm(\{1, \dots, 3k+1\} \setminus \{2k+1\})$. The short legs carry $\pm(2k+1)$ and $\pm(3k+2)$, so the labels are $S_{3k+2}$. By Lemma 18 the long leg has the sums $\pm(3k+1)$ at its ends, 0, and $\pm 1, \dots, \pm k$ and $\pm(k+2), \dots, \pm 3k$ inside. The short legs have $\pm(k+1)$ at $u_1, u_2$ and $\pm(3k+2)$ at $w_1, w_2$. These are the elements of $[-(3k+2), 3k+2]$, each once. □

*Proof of Theorem 6.* The grafting of Definition 12(d) attaches each pair to a vertex that is already present, so $T_k$ is a tree with $(6k+5) + 10 = 6k + 15$ vertices. By Proposition 5 and Theorem 4 the lifted labeling is $\mathbb{Z}_{6k+15}$-antimagic, so $T_k$ is antimagic, and Corollary 15 gives $v_0$. Attaching a leaf to a tree gives a tree. □

**Proposition 19** (The leaf goes at $x_{3k}$). *For every $k \ge 1$, $x_{3k}$ is the only vertex of $T_k$ at which the sum of the lifted labeling of $\psi_k$ is divisible by $6k+15$, and $T'_k$ is an antimagic tree with $6k+16$ vertices.*

*Proof.* In the proof of Theorem 4 the residue at a core vertex is its signed sum, and the residues at the grafted vertices are $\pm(q+i) \neq 0$. By Lemma 18 and the proof of Proposition 5, where the sums at $u_1, u_2, w_1, w_2$ are $\pm(k+1)$ and $\pm(3k+2)$, the only core vertex with signed sum 0 is $x_{3k}$. The residues are distinct, so the vertex $v_0$ of Corollary 15 applied to this labeling is $x_{3k}$. □

*Proof of Proposition 7.* In $T_k$ the vertices of degree 3 are $c$, $w_0$, $w_1$, $w_2$ and the two children $a = c(1,0)$, $b = c(1,1)$ of $w_0$; those of degree 2 are $x_1, \dots, x_{6k-1}$, $u_1$ and $u_2$; the eight leaves are the children of $a$, $b$, $w_1$ and $w_2$. In $T'_k$ the vertex $x_{3k}$ has degree 3 and the added vertex is a ninth leaf. The vertices $x_1, x_2$ have degree 2 in both trees (for $k = 1$ in $T'_1$ too, since $x_3$ is the only changed vertex), and $w_0, a$ have degree 3. The even-degree vertices induce the paths $x_1 \cdots x_{6k-1}$, $u_1$, $u_2$ in $T_k$, and $x_1 \cdots x_{3k-1}, x_{3k+1} \cdots x_{6k-1}, u_1, u_2$ in $T'_k$. The neighbors of $c$ are $x_1$, $u_1$, $u_2$, none a leaf. The vertices $w_0$, $w_1$, $w_2$ lie in the three components of $T_k - c$, so a path through all three would pass through $c$ twice. The path from a leaf below $a$ through $a$, $w_0$, the long leg, $c$, $u_1$ and $w_1$ to a leaf below $w_1$ has $2+6k+2+1 = 6k+5$ edges. No path is longer: removing $x_{3k}$ (and the added leaf in $T'_k$) leaves a component containing $c$, whose vertices are within distance $3k+3$ of $x_{3k}$, and one containing $w_0$, whose vertices are within distance $3k+2$ of it. A path between the two components passes through $x_{3k}$ and has at most $6k+5$ edges; a path inside the component of $c$ has at most $3k+3$ edges, since that component is the path $x_0 \cdots x_{3k-1}$ with the short legs and their four leaves at $x_0$, and one inside the component of $w_0$ has at most $3k+2$ edges, by the same count. □

*Proof of Corollary 8.* The second column of Table 5 names, for each class, an invariant of Proposition 7 that its hypothesis contradicts. □

### 3.3 Counting.

*Proof of Theorem 1.* Let $f$ be an antimagic labeling of $G$ and $e$ its edge with label 1. On $G - e$, $g = f - 1$ is a bijection onto $\{1, \dots, m-1\}$, and $s_f(v) = s_g(v) + r$ at every vertex: a vertex off $e$ loses 1 on each of its $r$ edges, and an endpoint of $e$ loses 1 on $r-1$ edges and the label 1 itself. So $g$ is antimagic. Conversely, an edge $e$ and an antimagic labeling $g$ of $G - e$ give the antimagic labeling $f = g + 1, f(e) = 1$, of $G$. These maps are inverse. □

**Corollary 20** (Edge-transitive graphs). *If $G$ is $r$-regular and edge-transitive, then $A(G) = m\,A(G - e_0)$ for every edge $e_0$.*

**Corollary 21** (Cycles). *For every $n \geq 3$, $A(C_n) = n\,A(P_n)$.*

**Corollary 22** (Complete graphs). *For every $n$ and every edge $e_0$ of $K_n$, $A(K_n) = \binom{n}{2} A(K_n - e_0)$.*

*Proof of Corollaries 20 to 22.* An automorphism carrying $e$ to $e_0$ carries $G - e$ to $G - e_0$, and $A$ is invariant under isomorphism. $C_n$ is 2-regular and edge-transitive and $C_n$ minus an edge is $P_n$; $K_n$ is $(n-1)$-regular and edge-transitive. □

**Proposition 23** (Stars). *For every $m \geq 2$, $A(K_{1,m}) = m!$.*

*Proof.* The leaves get the distinct labels $1, \dots, m$, and the center gets $m(m+1)/2 > m$. □

*Proof of Propositions 2 and 3.* The enumeration of Section 4 generates the 199 trees on 3 to 10 vertices and the 141 connected graphs on 3 to 6 vertices, one per isomorphism class, and counts the bijective labelings with distinct sums of every tree and of the 89 graphs of Proposition 3(b); for each of the other 52 graphs, on 6 vertices with more than 8 edges, it searches for one antimagic labeling and finds one. □

**3.4 The Combinatorial Nullstellensatz.** Alon's Combinatorial Nullstellensatz [30] [to check] says that if the coefficient of $x_1^{t_1} \cdots x_n^{t_n}$ in a polynomial $F$ of degree $t_1 + \cdots + t_n$ is not zero, then $F$ does not vanish on any product $S_1 \times \cdots \times S_n$ with $|S_i| > t_i$. Hefetz applied it to antimagic labelings [31] [to check]: a point of $\{1, \dots, m\}^m$ where $\Delta_m P_G \neq 0$ is an antimagic labeling. The polynomial $\Delta_m P_G$ is homogeneous of degree $\binom{m}{2} + \binom{n}{2}$, and the sets have size $m$, so the only usable monomials have every exponent at most $m-1$.

**Proposition 24** (Degree gap). *For $n \geq 2$ and every $m$, $\Gamma(n,m) \leq 0$ if and only if $n \leq m$.*

**Proposition 25** (No admissible monomial). *If $\Gamma(n,m) > 0$, then no exponent vector $(d_1, \dots, d_m)$ with every $d_i \leq m-1$ has $d_1 + \cdots + d_m = \binom{m}{2} + \binom{n}{2}$.*

**Corollary 26** (Trees). *For a tree on $n \geq 2$ vertices, $\Gamma(n, n-1) = n - 1 > 0$.*

*Proofs.* $\binom{n}{2} - \binom{m}{2}$ has the sign of $n - m$ when $n \geq 2$. The exponents add to at most $m(m-1) = 2\binom{m}{2}$. For a tree, $\binom{n}{2} - \binom{n-1}{2} = n - 1$. □

So for a tree $T$ on $n \geq 2$ vertices the Nullstellensatz cannot be applied directly to $\Delta_{n-1} P_T$ on $\{1, \dots, n-1\}^{n-1}$: every monomial of $\Delta_{n-1} P_T$ has an exponent at least $n-1$. For unicyclic graphs ($m = n$) the only usable monomial is the box monomial.

**Proposition 27** (Box criterion). *Let $G$ have $n$ vertices and $n$ indexed edges. If $c_{\mathrm{box}}(G) \neq 0$, then $G$ is antimagic.*

*Proof.* By the Nullstellensatz with $S_i = \{1, \dots, n\}$ there is $x \in \{1, \dots, n\}^n$ with $\Delta_n(x) P_G(x) \neq 0$: the $x_i$ are distinct, so $x$ is a bijective labeling, and its vertex sums are distinct. □

**Proposition 28** (Sign-reversing automorphisms). *Let $G$ have $n$ vertices and $n$ indexed edges, and let $\phi$ be an automorphism of $G$. If the sign of $\phi$ as a permutation of the vertices times the sign of the permutation it induces on the edge indices is $-1$, then $c_{\mathrm{box}}(G) = 0$.*

*Proof.* Substituting the edge permutation into $\Delta_n P_G$ multiplies $\Delta_n$ by the sign of the edge permutation and $P_G$ by the sign of the vertex permutation, so it multiplies the polynomial by $-1$; the box monomial is fixed, so its coefficient is its own negative. □

**Corollary 29** (Even cycles). *For every $k \geq 2$ and every indexing of its edges, $c_{\mathrm{box}}(C_{2k}) = 0$.*

*Proof.* The reflection of $C_{2k}$ through two opposite vertices is a product of $k-1$ transpositions on the vertices and of $k$ transpositions on the edges, so its two signs differ. □

**Table 1.** Box coefficients and counts of the unicyclic graphs on 3 to 5 vertices, with the edges in the order listed (pairs of vertices $0, \dots, n-1$). The coefficients are theorems of the Lean library; the counts $A(G)$ are recomputed by `recompute.py`.

| graph | $n$ | edges | $c_{\mathrm{box}}(G)$ | $A(G)$ |
|---|---|---|---|---|
| $C_3$ | 3 | 01 02 12 | $-6$ | 6 |
| $C_4$ | 4 | 01 03 12 23 | 0 | 8 |
| paw | 4 | 01 02 12 23 | 0 | 8 |
| $C_5$ | 5 | 01 04 12 23 34 | $-980$ | 30 |
| banner | 5 | 01 03 12 23 34 | 0 | 48 |
| tadpole $(3,2)$ | 5 | 01 02 12 23 34 | $-400$ | 30 |
| bull | 5 | 01 02 12 13 24 | 180 | 48 |
| cricket | 5 | 01 02 12 23 24 | 40 | 64 |

Table 1 shows that the criterion of Proposition 27 decides five of the eight unicyclic graphs on at most 5 vertices and is silent on $C_4$, the paw and the banner, which are antimagic. Proposition 28 explains the zeros of $C_4$ and of the banner: in the banner, exchanging the vertices 0 and 2 induces $(01\,12)(03\,23)$ on the edges and fixes 34, so the two signs are $-1$ and $+1$; this is the library's proof for the banner. Every automorphism of the paw has sign product $+1$ (a theorem of the library), so its zero coefficient is not explained by Proposition 28.

**Example 30** (The Vandermonde factor is needed). Let $F = (x_1 + x_2 + x_3)^3 - 36x_1x_2x_3$. For each permutation $\pi$ of $\{0,1,2\}$ the coefficient of $x_1^{\pi(0)}x_2^{\pi(1)}x_3^{\pi(2)}$ in $F$ is 3, yet $F$ vanishes at every permutation of $(1,2,3)$, since $6^3 = 36 \cdot 6$; and $c_{\mathrm{box}}(F) = 0$. A nonzero coefficient of a staircase monomial of $F$ alone therefore does not give a point of $\{1,2,3\}^3$ with distinct coordinates where $F$ is nonzero (elsewhere it can be: $F(1,2,4) = 7^3 - 36 \cdot 8 = 55$), while the coefficient of the box monomial in $\Delta_3 F$, which is what the Nullstellensatz needs, is zero here. All three facts are theorems of the Lean library.

## 4. COMPUTATIONAL VERIFICATION AND EMPIRICAL PICTURE

All computations of this section are in `recompute.py`, which uses exact integer arithmetic (Python integers and `numpy` arrays of 64-bit integers, with no floating point); its output is `recompute.out`. It is independent of the programs of the finding bundle: it builds the trees, graphs and labelings itself.

**Cores and the family.** The labelings of Proposition 16 for $t \le 40$, of Proposition 17 for $s \le 25$ and of Proposition 5 for $k \le 25$ are BSA labelings. For $k \le 25$ the lifted labeling of $T_k$ is a bijection onto $\{1, \dots, 6k+14\}$ whose sums are distinct modulo $6k+15$, the only sum divisible by $6k+15$ is the one at $x_{3k}$, the leaf extension at $x_{3k}$ is antimagic, and the invariants of Proposition 7 hold for $T_k$ and $T'_k$. It also finds a super edge-graceful labeling of each of the $1+3+11+47 = 62$ trees with 2, 4, 6 and 8 edges, in agreement with Theorem 3.2 of [9].

**Trees and graphs.** Trees are generated by adding a leaf in every way to the trees of the previous order and keeping one per canonical form, and connected graphs by reducing every edge set to its least relabeling; the numbers of classes are $1,2,3,6,11,23,47,106$ trees for $n = 3, \dots, 10$ and $2,6,21,112$ connected graphs for $n = 3, \dots, 6$, the values of OEIS A000055 and A001349 [36]. For each tree, and each graph counted exactly, the program forms all $m!$ bijective labelings and counts those with distinct sums. Table 2 gives the extremes over trees. All 141 connected graphs get a search for one antimagic labeling, which finds one for each. The 89 graphs with at most 5 vertices, or with 6 vertices and at most 8 edges, are counted exactly, and so are $K_{3,3}$ and the prism $C_3 \times K_2$, which have 9 edges; the other 50 get only the search. The counts include $A(K_4) = 240$, $A(K_5) = 2292480$, $A(K_{3,3}) = 111744$ and $A(C_3 \times K_2) = 108072$.

**Table 2.** Antimagic labelings of trees on $n$ vertices, $3 \le n \le 10$: the minimum, which only $P_n$ attains for $4 \le n \le 10$; the next value; and the maximum $(n-1)!$, which only the star attains for $4 \le n \le 10$.

| $n$ | trees | $A(P_n)$ | next smallest | $(n-1)!$ |
|---|---|---|---|---|
| 3 | 1 | 2 | – | 2 |
| 4 | 2 | 2 | 6 | 6 |
| 5 | 3 | 6 | 12 | 24 |
| 6 | 6 | 16 | 30 | 120 |
| 7 | 11 | 92 | 142 | 720 |
| 8 | 23 | 352 | 596 | 5040 |
| 9 | 47 | 1910 | 3226 | 40320 |
| 10 | 106 | 10892 | 19464 | 362880 |

**Deletion identity.** The program checks $A(G) = \sum_e A(G-e)$ on the eight regular graphs among those it counts: $C_3$, $C_4$, $K_4$, $C_5$, $C_6$, $K_{3,3}$, the prism $C_3 \times K_2$ and $K_5$. It also checks $A(C_n) = nA(P_n)$ for $n = 3, \dots, 10$, with $A(C_n) = 6, 8, 30, 96, 644, 2816, 17190, 108920$.

**Nullstellensatz coefficients.** The program expands $\Delta_n P_G$ for the graphs of Table 1, dropping every monomial with an exponent above $n-1$ (exponents never decrease, so this is exact), and obtains the coefficients of the table; it also checks the three facts of Example 30: the six staircase coefficients, the six values and the box coefficient of $\Delta_3 F$.

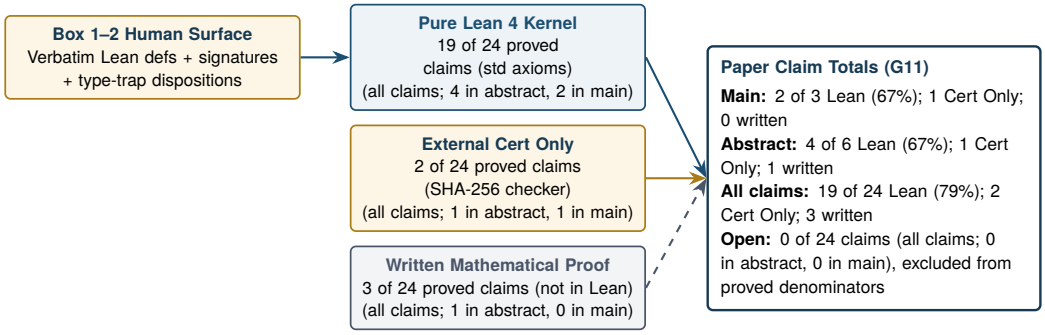
**Runs and receipts.** `recompute.py` ran in 7.35 seconds on an Apple M4 Max (arm64, Python 3.11.15, NumPy 2.4.6); `run_recompute.py` records the command, the versions, the elapsed time, the exit status and whether the output equals `recompute.out` byte for byte, in `recompute-run.json`. The Lean library's receipt in the finding bundle records its build with Lean `v4.35.0` and the SHA-256 digest of each source file and of the axiom report; the check of this draft compares these digests with the files. The finding's own programs and their receipts are in the bundle; this draft does not rely on them.

## 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 849f300f5d44 (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 4 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: 177/177 public theorems (source-level approx) | WARN: Fidelity: 19/19 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)**

**HartsfieldAntimagic.EdgeLabeling (HartsfieldAntimagic/Basic.lean:35-35) — Definition "Labelings" (Section 2)**

*Context binders in scope:* `variable {V : Type*}`

```
abbrev EdgeLabeling (G : SimpleGraph V) : Type _ := G.edgeSet → ℕ
```

**HartsfieldAntimagic.IsBijectiveEdgeLabeling (HartsfieldAntimagic/Basic.lean:39-41) — Definition "Labelings" (Section 2)**

*Context binders in scope:* `variable {V : Type*}`

```
def IsBijectiveEdgeLabeling (G : SimpleGraph V) [Fintype G.edgeSet]
    (f : EdgeLabeling G) : Prop :=
  Set.BijOn f Set.univ (Finset.Icc 1 G.edgeFinset.card : Set ℕ)
```

**HartsfieldAntimagic.vertexSum (HartsfieldAntimagic/Basic.lean:44-46) — Definition "Labelings" (Section 2)**

*Context binders in scope:* `variable {V : Type*}`

```
def vertexSum [DecidableEq V] (G : SimpleGraph V) [Fintype G.edgeSet]
    (f : EdgeLabeling G) (v : V) : ℕ :=
  ∑ e : G.edgeSet, if v ∈ (e : Sym2 V) then f e else 0
```

**HartsfieldAntimagic.IsAntimagicLabeling (HartsfieldAntimagic/Basic.lean:50-52) — Definition "Labelings" (Section 2)**

*Context binders in scope:* `variable {V : Type*}`

```
def IsAntimagicLabeling [DecidableEq V] (G : SimpleGraph V) [Fintype G.edgeSet]
    (f : EdgeLabeling G) : Prop :=
  IsBijectiveEdgeLabeling G f ∧ Function.Injective (vertexSum G f)
```

**HartsfieldAntimagic.IsAntimagic (HartsfieldAntimagic/Basic.lean:55-56) — Definition "Labelings" (Section 2)**

*Context binders in scope:* `variable {V : Type*}`

```
def IsAntimagic [DecidableEq V] (G : SimpleGraph V) [Fintype G.edgeSet] : Prop :=
  ∃ f : EdgeLabeling G, IsAntimagicLabeling G f
```

**HartsfieldAntimagic.IsModAntimagicLabeling (HartsfieldAntimagic/Basic.lean:61-64) — Definition "Labelings" (Section 2)**

*Context binders in scope:* `variable {V : Type*}`

```
def IsModAntimagicLabeling [DecidableEq V] (G : SimpleGraph V) [Fintype G.edgeSet]
    (f : EdgeLabeling G) : Prop :=
  IsBijectiveEdgeLabeling G f ∧
    Function.Bijective (fun v : V => (vertexSum G f v : ZMod (G.edgeFinset.card + 1)))
```

**HartsfieldAntimagic.antimagicCount (HartsfieldAntimagic/Basic.lean:67-68) — Definition "Labelings" (Section 2)**

*Context binders in scope:* `variable {V : Type*}`

```
noncomputable def antimagicCount [DecidableEq V] (G : SimpleGraph V) [Fintype
  ↪ G.edgeSet] : ℕ :=
  Nat.card { f : EdgeLabeling G // IsAntimagicLabeling G f }
```

**HartsfieldAntimagic.signedShell (HartsfieldAntimagic/Basic.lean:376-377) — Definition "Signed labelings and BSA cores" (Section 2)**

*Context binders in scope:* `variable {V : Type*}`

```
def signedShell (q : ℕ) : Finset ℤ :=
  (Finset.Icc (-(q : ℤ)) (q : ℤ)).erase 0
```

**HartsfieldAntimagic.signedInterval (HartsfieldAntimagic/Basic.lean:380-381) — Definition "Signed labelings and BSA cores" (Section 2)**

*Context binders in scope:* `variable {V : Type*}`

```
def signedInterval (q : ℕ) : Finset ℤ :=
  Finset.Icc (-(q : ℤ)) (q : ℤ)
```

**HartsfieldAntimagic.SignedEdgeLabeling (HartsfieldAntimagic/Basic.lean:411-411) — Definition "Signed labelings and BSA cores" (Section 2)**

*Context binders in scope:* `variable {V : Type*}`

```
abbrev SignedEdgeLabeling (G : SimpleGraph V) : Type _ := G.edgeSet → ℤ
```

**HartsfieldAntimagic.signedVertexSum (HartsfieldAntimagic/Basic.lean:414-416) — Definition "Signed labelings and BSA cores" (Section 2)**

*Context binders in scope:* `variable {V : Type*}`

```
def signedVertexSum [DecidableEq V] (G : SimpleGraph V) [Fintype G.edgeSet]
    (ψ : SignedEdgeLabeling G) (v : V) : ℤ :=
  ∑ e : G.edgeSet, if v ∈ (e : Sym2 V) then ψ e else 0
```

**HartsfieldAntimagic.IsBSALabeling (HartsfieldAntimagic/Basic.lean:421-428) — Definition "Signed labelings and BSA cores" (Section 2)**

*Context binders in scope:* `variable {V : Type*}`

```
structure IsBSALabeling [Fintype V] [DecidableEq V] (G : SimpleGraph V) [Fintype
  ↪ G.edgeSet]
    (q : ℕ) (ψ : SignedEdgeLabeling G) : Prop where
  pos_q : 1 ≤ q
  isTree : G.IsTree
  card_edges : G.edgeFinset.card = 2 * q
  card_verts : Fintype.card V = 2 * q + 1
  edge_bij : Set.BijOn ψ Set.univ (signedShell q : Set ℤ)
  sum_bij : Set.BijOn (signedVertexSum G ψ) Set.univ (signedInterval q : Set ℤ)
```

**HartsfieldAntimagic.IsBSACore (HartsfieldAntimagic/Basic.lean:432-434) — Definition "Signed labelings and BSA cores" (Section 2)**

*Context binders in scope:* `variable {V : Type*}`

```
def IsBSACore [Fintype V] [DecidableEq V] (G : SimpleGraph V) [Fintype G.edgeSet]
    (q : ℕ) : Prop :=
  ∃ ψ : SignedEdgeLabeling G, IsBSALabeling G q ψ
```

**HartsfieldAntimagic.IsEvenChildrenGrafting (HartsfieldAntimagic/BSAShellLifting.lean:53-63) — Definition "Even-children grafting and the lifted labeling" (Section 2)**

*Context binders in scope:* `variable {V1 V2 : Type*}`

```
structure IsEvenChildrenGrafting (T1 : SimpleGraph V1) (T : SimpleGraph (V1 ⊕ V2)) (r
  ↪ : ℕ)
    (parent : Fin r → V1 ⊕ V2) (child : Fin r × Bool ≃ V2) : Prop where
  loopless_graft : ∀ (i : Fin r) (b : Bool), parent i ≠ Sum.inr (child (i, b))
  no_reverse_edge : ∀ p₁ p₂ : Fin r × Bool,
    ¬(parent p₁.1 = Sum.inr (child p₂) ∧ parent p₂.1 = Sum.inr (child p₁))
  adj_iff : ∀ x y : V1 ⊕ V2,
    T.Adj x y ↔
      (∃ u v : V1, x = Sum.inl u ∧ y = Sum.inl v ∧ T1.Adj u v) ∨
      (∃ p : Fin r × Bool,
        (x = parent p.1 ∧ y = Sum.inr (child p)) ∨
        (y = parent p.1 ∧ x = Sum.inr (child p)))
```

**HartsfieldAntimagic.IsEvenChildrenGrafting.edgeEquiv (HartsfieldAntimagic/BSAShellLifting.lean:101-164) — Definition "Even-children grafting and the lifted labeling" (Section 2)**

*Context binders in scope:* `variable {V1 V2 : Type*}`

```
noncomputable def IsEvenChildrenGrafting.edgeEquiv {T1 : SimpleGraph V1}
    {T : SimpleGraph (V1 ⊕ V2)} {r : ℕ} {parent : Fin r → V1 ⊕ V2} {child : Fin r ×
```

```
    ↪ Bool ≃ V2}
  (hg : IsEvenChildrenGrafting T1 T r parent child) :
  T1.edgeSet ⊕ (Fin r × Bool) ≃ T.edgeSet :=
Equiv.ofBijective
  (fun
    | Sum.inl e => ⟨Sym2.map Sum.inl e.1, by
        rcases e with ⟨e, he⟩
        induction e using Sym2.ind with
        | _ u v =>
          exact T.mem_edgeSet.mpr ((hg.adj_iff _ _).mpr
            (Or.inl ⟨u, v, rfl, rfl, T1.mem_edgeSet.mp he⟩))⟩
    | Sum.inr p => ⟨s(parent p.1, Sum.inr (child p)),
        T.mem_edgeSet.mpr ((hg.adj_iff _ _).mpr
          (Or.inr ⟨p, Or.inl ⟨rfl, rfl⟩⟩))⟩)
  (by
    constructor
    · intro a b hab
      cases a with
      | inl e₁ =>
        cases b with
        | inl e₂ =>
          have hval : Sym2.map Sum.inl e₁.1 = Sym2.map Sum.inl e₂.1 :=
            congrArg Subtype.val hab
          congr 1
          apply Subtype.ext
          exact Sym2.map.injective Sum.inl_injective hval
        | inr p₂ =>
          exfalso
          have hval : Sym2.map Sum.inl e₁.1 = s(parent p₂.1, Sum.inr (child p₂)) :=
            congrArg Subtype.val hab
          have hmem : Sum.inr (child p₂) ∈ Sym2.map Sum.inl e₁.1 := by
            rw [hval]
            exact Sym2.mem_mk_right _ _
          rcases Sym2.mem_map.mp hmem with ⟨_, _, hcontra⟩
          cases hcontra
      | inr p₁ =>
        cases b with
        | inl e₂ =>
          exfalso
          have hval : s(parent p₁.1, Sum.inr (child p₁)) = Sym2.map Sum.inl e₂.1 :=
            congrArg Subtype.val hab
          have hmem : Sum.inr (child p₁) ∈ Sym2.map Sum.inl e₂.1 := by
            rw [← hval]
            exact Sym2.mem_mk_right _ _
          rcases Sym2.mem_map.mp hmem with ⟨_, _, hcontra⟩
          cases hcontra
        | inr p₂ =>
          have hval : s(parent p₁.1, Sum.inr (child p₁)) =
              s(parent p₂.1, Sum.inr (child p₂)) := congrArg Subtype.val hab
          congr 1
          rcases Sym2.eq_iff.mp hval with ⟨_, h2⟩ | ⟨h1, h2⟩
          · exact child.injective (Sum.inr.inj h2)
          · exfalso
            exact hg.no_reverse_edge p₁ p₂ ⟨h1, h2.symm⟩
    · rintro ⟨e, he⟩
      induction e using Sym2.ind with
      | _ x y =>
        have hadj : T.Adj x y := T.mem_edgeSet.mp he
        rcases (hg.adj_iff x y).mp hadj with
          ⟨u, v, rfl, rfl, huv⟩ | ⟨p, ⟨rfl, rfl⟩ | ⟨rfl, rfl⟩⟩
        · exact ⟨Sum.inl ⟨s(u, v), T1.mem_edgeSet.mpr huv⟩, Subtype.ext rfl⟩
        · exact ⟨Sum.inr p, Subtype.ext rfl⟩
        · exact ⟨Sum.inr p, Subtype.ext (Sym2.eq_swap)⟩)
```

**HartsfieldAntimagic.bsaShellLiftLabeling (HartsfieldAntimagic/BSAShellLifting.lean:229-238) — Definition "Even-children grafting and the lifted labeling" (Section 2)**

*Context binders in scope:* `variable {V1 V2 : Type*} ; variable [Fintype V1] [DecidableEq V1] [Fintype V2] [DecidableEq V2]`

```
noncomputable def bsaShellLiftLabeling {T1 : SimpleGraph V1} {T : SimpleGraph (V1 ⊕
  ↪ V2)}
    {q r : ℕ} {parent : Fin r → V1 ⊕ V2} {child : Fin r × Bool ≃ V2}
    (hg : IsEvenChildrenGrafting T1 T r parent child)
    (ψ : SignedEdgeLabeling T1) : EdgeLabeling T := fun e =>
  let M : ℕ := 2 * q + 2 * r + 1
  match hg.edgeEquiv.symm e with
  | Sum.inl e1 =>
      if 0 < ψ e1 then (ψ e1).toNat else ((M : ℤ) + ψ e1).toNat
  | Sum.inr (i, false) => q + (i.1 + 1)
  | Sum.inr (i, true) => M - q - (i.1 + 1)
```

**HartsfieldAntimagic.leafExtendGraph (HartsfieldAntimagic/BSAShellLifting.lean:670-680) — Definition "Even-children grafting and the lifted labeling" (Section 2)**

*Context binders in scope:* `variable {V : Type*}`

```
def leafExtendGraph (G : SimpleGraph V) (v0 : V) : SimpleGraph (Option V) where
  Adj x y :=
    match x, y with
    | some u, some v => G.Adj u v
    | some u, none => u = v0
    | none, some v => v = v0
    | none, none => False
  symm := ⟨fun x y h => by
    cases x <;> cases y <;> simp_all [SimpleGraph.adj_comm]⟩
  loopless := ⟨fun x h => by
    cases x <;> simp_all⟩
```

**HartsfieldAntimagic.pathGraph (HartsfieldAntimagic/BSAShellLifting.lean:928-931) — Definition "Named graphs and labelings" (Section 2)**

```
def pathGraph (n : ℕ) : SimpleGraph (Fin n) where
  Adj i j := i.1 + 1 = j.1 ∨ j.1 + 1 = i.1
  symm := ⟨fun _ _ h => h.symm⟩
  loopless := ⟨fun i h => by omega⟩
```

**HartsfieldAntimagic.cycleGraph (HartsfieldAntimagic/NullstellensatzObstructions.lean:1074-1077) — Definition "Named graphs and labelings" (Section 2)**

```
def cycleGraph (n : ℕ) : SimpleGraph (Fin n) where
  Adj i j := ((i.1 + 1) % n = j.1 ∨ (j.1 + 1) % n = i.1) ∧ i ≠ j
  symm := ⟨fun _ _ ⟨h1, h2⟩ => ⟨h1.symm, h2.symm⟩⟩
  loopless := ⟨fun _ ⟨_, h2⟩ => h2 rfl⟩
```

**HartsfieldAntimagic.starGraph (HartsfieldAntimagic/NullstellensatzObstructions.lean:351-354) — Definition "Named graphs and labelings" (Section 2)**

```
def starGraph (m : ℕ) : SimpleGraph (Fin (m + 1)) where
  Adj u v := (u.1 = 0 ∧ v.1 ≠ 0) ∨ (v.1 = 0 ∧ u.1 ≠ 0)
  symm := ⟨fun _ _ h => h.symm⟩
  loopless := ⟨fun u h => by rcases h with ⟨h1, h2⟩ | ⟨h1, h2⟩ <;> exact h2 h1⟩
```

**HartsfieldAntimagic.subdivStar (HartsfieldAntimagic/BSAShellLifting.lean:1159-1183) — Definition "Named graphs and labelings" (Section 2)**

```
def subdivStar (s : ℕ) : SimpleGraph (SubdivStarVert s) where
  Adj x y :=
    match x, y with
    | none, some (_, false) => True
    | some (_, false), none => True
    | some (j₁, false), some (j₂, true) => j₁ = j₂
    | some (j₁, true), some (j₂, false) => j₁ = j₂
    | _, _ => False
  symm := ⟨fun x y h => by
    cases x with
    | none =>
      cases y with
      | none => exact h
      | some p => rcases p with ⟨j, b⟩; cases b <;> exact h
    | some p =>
      rcases p with ⟨j₁, b₁⟩
      cases y with
      | none => cases b₁ <;> exact h
      | some q =>
```

```
      rcases q with ⟨j₂, b₂⟩
      cases b₁ <;> cases b₂ <;> simp_all⟩
  loopless := ⟨fun x h => by
    cases x with
    | none => exact h
    | some p => rcases p with ⟨j, b⟩; cases b <;> exact h⟩
```

**HartsfieldAntimagic.holeJumpStar (HartsfieldAntimagic/BSAShellLifting.lean:1511-1534) — Definition "Named graphs and labelings" (Section 2)**

```
def holeJumpStar (k : ℕ) : SimpleGraph (HoleJumpVert k) where
  Adj x y :=
    match x, y with
    | Sum.inl i, Sum.inl j => i.1 + 1 = j.1 ∨ j.1 + 1 = i.1
    | Sum.inl i, Sum.inr (_, false) => i.1 = 0
    | Sum.inr (_, false), Sum.inl j => j.1 = 0
    | Sum.inr (b₁, false), Sum.inr (b₂, true) => b₁ = b₂
    | Sum.inr (b₁, true), Sum.inr (b₂, false) => b₁ = b₂
    | _, _ => False
  symm := ⟨fun x y h => by
    cases x with
    | inl i =>
      cases y with
      | inl j => exact h.symm
      | inr q => rcases q with ⟨b, f⟩; cases f <;> exact h
    | inr p =>
      rcases p with ⟨b₁, f₁⟩
      cases y with
      | inl j => cases f₁ <;> exact h
      | inr q => rcases q with ⟨b₂, f₂⟩; cases f₁ <;> cases f₂ <;> simp_all⟩
  loopless := ⟨fun x h => by
    cases x with
    | inl i => omega
    | inr p => rcases p with ⟨b, f⟩; cases f <;> exact h⟩
```

**HartsfieldAntimagic.holeJumpM (HartsfieldAntimagic/BSAShellLifting.lean:1694-1695) — Definition "Named graphs and labelings" (Section 2)**

```
def holeJumpM (k r : ℕ) : ℕ :=
  if r ≤ 2 * k then r else r + 1
```

**HartsfieldAntimagic.holeJumpZ (HartsfieldAntimagic/BSAShellLifting.lean:1698-1700) — Definition "Named graphs and labelings" (Section 2)**

```
def holeJumpZ (k j : ℕ) : ℤ :=
  if j % 2 = 0 then (holeJumpM k (3 * k - j / 2) : ℤ)
  else -(holeJumpM k (j / 2 + 1) : ℤ)
```

**HartsfieldAntimagic.holeJumpLeg0Val (HartsfieldAntimagic/BSAShellLifting.lean:1703-1704) — Definition "Named graphs and labelings" (Section 2)**

```
def holeJumpLeg0Val (k j : ℕ) : ℤ :=
  if j < 3 * k then holeJumpZ k j else -holeJumpZ k (6 * k - 1 - j)
```

**HartsfieldAntimagic.holeJumpEdgeVal (HartsfieldAntimagic/BSAShellLifting.lean:1708-1713) — Definition "Named graphs and labelings" (Section 2)**

```
def holeJumpEdgeVal (k : ℕ) : Fin (6 * k) ⊕ (Fin 2 × Bool) → ℤ
  | Sum.inl j => holeJumpLeg0Val k j.1
  | Sum.inr (⟨0, _⟩, false) => -((2 * k + 1 : ℕ) : ℤ)
  | Sum.inr (⟨0, _⟩, true) => ((3 * k + 2 : ℕ) : ℤ)
  | Sum.inr (⟨1, _⟩, false) => ((2 * k + 1 : ℕ) : ℤ)
  | Sum.inr (⟨1, _⟩, true) => -((3 * k + 2 : ℕ) : ℤ)
```

**HartsfieldAntimagic.holeJumpEdgeEquiv (HartsfieldAntimagic/BSAShellLifting.lean:1566-1649) — Definition "Named graphs and labelings" (Section 2)**

```
noncomputable def holeJumpEdgeEquiv (k : ℕ) :
    Fin (6 * k) ⊕ (Fin 2 × Bool) ≃ (holeJumpStar k).edgeSet :=
  Equiv.ofBijective
    (fun
      | Sum.inl j =>
          ⟨s(Sum.inl ⟨j.1, by omega⟩, Sum.inl ⟨j.1 + 1, by omega⟩),
            (holeJumpStar k).mem_edgeSet.mpr (Or.inl rfl)⟩
      | Sum.inr (b, false) =>
          ⟨s(Sum.inl ⟨0, by omega⟩, Sum.inr (b, false)),
            (holeJumpStar k).mem_edgeSet.mpr rfl⟩
      | Sum.inr (b, true) =>
          ⟨s(Sum.inr (b, false), Sum.inr (b, true)),
            (holeJumpStar k).mem_edgeSet.mpr rfl⟩)
    (by
      constructor
```

```
    · intro a b hab
      have hval := congrArg Subtype.val hab
      cases a with
      | inl j₁ =>
        cases b with
        | inl j₂ =>
          dsimp at hval
          rcases Sym2.eq_iff.mp hval with ⟨h1, _⟩ | ⟨h1, h2⟩
          · have h1' : (⟨j₁.1, by omega⟩ : Fin (6 * k + 1)) = ⟨j₂.1, by omega⟩ :=
              Sum.inl.inj h1
            exact congrArg Sum.inl (Fin.ext (Fin.mk.inj h1'))
          · exfalso
            have h1' : (⟨j₁.1, by omega⟩ : Fin (6 * k + 1)) = ⟨j₂.1 + 1, by omega⟩
          ↪ :=
              Sum.inl.inj h1
            have h2' : (⟨j₁.1 + 1, by omega⟩ : Fin (6 * k + 1)) = ⟨j₂.1, by omega⟩
          ↪ :=
              Sum.inl.inj h2
            have hv1 := congrArg Fin.val h1'
            have hv2 := congrArg Fin.val h2'
            dsimp at hv1 hv2; omega
        | inr p₂ =>
          rcases p₂ with ⟨b₂, f₂⟩
          cases f₂ <;> simp_all
      | inr p₁ =>
        rcases p₁ with ⟨b₁, f₁⟩
        cases b with
        | inl j₂ => cases f₁ <;> simp_all
        | inr p₂ =>
          rcases p₂ with ⟨b₂, f₂⟩
          cases f₁ <;> cases f₂ <;> simp_all
    · rintro ⟨e, he⟩
      induction e using Sym2.ind with
      | _ x y =>
        have hadj : (holeJumpStar k).Adj x y := (holeJumpStar k).mem_edgeSet.mp he
        cases x with
        | inl i =>
          cases y with
          | inl j =>
            rcases hadj with h | h
            · refine ⟨Sum.inl ⟨i.1, by omega⟩, Subtype.ext ?_⟩
              have hi : (⟨i.1, by omega⟩ : Fin (6 * k + 1)) = i := Fin.ext rfl
              have hj : (⟨i.1 + 1, by omega⟩ : Fin (6 * k + 1)) = j := Fin.ext h
              simp [hi, hj]
            · refine ⟨Sum.inl ⟨j.1, by omega⟩, Subtype.ext ?_⟩
              have hj : (⟨j.1, by omega⟩ : Fin (6 * k + 1)) = j := Fin.ext rfl
              have hi : (⟨j.1 + 1, by omega⟩ : Fin (6 * k + 1)) = i := Fin.ext h
              simp [hi, hj, Sym2.eq_swap]
          | inr q =>
            rcases q with ⟨b, f⟩
            cases f
            · have hi0 : i = ⟨0, by omega⟩ := Fin.ext hadj
              exact ⟨Sum.inr (b, false), Subtype.ext (by simp [hi0])⟩
            · exact False.elim hadj
        | inr p =>
          rcases p with ⟨b₁, f₁⟩
          cases y with
          | inl j =>
            cases f₁
            · have hj0 : j = ⟨0, by omega⟩ := Fin.ext hadj
              exact ⟨Sum.inr (b₁, false), Subtype.ext (by simp [hj0, Sym2.eq_swap])⟩
            · exact False.elim hadj
          | inr q =>
            rcases q with ⟨b₂, f₂⟩
            cases f₁ <;> cases f₂
            · exact False.elim hadj
            · cases hadj
```

```
              exact ⟨Sum.inr (b₁, true), Subtype.ext rfl⟩
            · cases hadj
              exact ⟨Sum.inr (b₁, true), Subtype.ext Sym2.eq_swap⟩
            · exact False.elim hadj)
```

**HartsfieldAntimagic.holeJumpBSALabeling (HartsfieldAntimagic/BSAShellLifting.lean:1716-1717) — Definition "Named graphs and labelings" (Section 2)**

```
noncomputable def holeJumpBSALabeling (k : ℕ) : SignedEdgeLabeling (holeJumpStar k) :=
  fun e => holeJumpEdgeVal k ((holeJumpEdgeEquiv k).symm e)
```

**HartsfieldAntimagic.graftedHoleJumpParent (HartsfieldAntimagic/BSAShellLifting.lean:1993-1998) — Definition "Named graphs and labelings" (Section 2)**

```
def graftedHoleJumpParent (k : ℕ) : Fin 5 → HoleJumpVert k ⊕ (Fin 5 × Bool)
  | ⟨0, _⟩ => Sum.inl (Sum.inl ⟨6 * k, by omega⟩)
  | ⟨1, _⟩ => Sum.inr (⟨0, by omega⟩, false)
  | ⟨2, _⟩ => Sum.inr (⟨0, by omega⟩, true)
  | ⟨3, _⟩ => Sum.inl (Sum.inr (⟨0, by omega⟩, true))
  | ⟨4, _⟩ => Sum.inl (Sum.inr (⟨1, by omega⟩, true))
```

**HartsfieldAntimagic.graftedHoleJumpDepth (HartsfieldAntimagic/BSAShellLifting.lean:2001-2007) — Definition "Named graphs and labelings" (Section 2)**

```
def graftedHoleJumpDepth (k : ℕ) : HoleJumpVert k ⊕ (Fin 5 × Bool) → ℕ
  | Sum.inl _ => 0
  | Sum.inr (⟨0, _⟩, _) => 1
  | Sum.inr (⟨1, _⟩, _) => 2
  | Sum.inr (⟨2, _⟩, _) => 2
  | Sum.inr (⟨3, _⟩, _) => 1
  | Sum.inr (⟨4, _⟩, _) => 1
```

**HartsfieldAntimagic.graftedGraph (HartsfieldAntimagic/BSAShellLifting.lean:66-85) — Definition "Even-children grafting and the lifted labeling" (Section 2)**

*Context binders in scope:* `variable {V1 V2 : Type*}`

```
def graftedGraph (T1 : SimpleGraph V1) {r : ℕ}
    (parent : Fin r → V1 ⊕ V2) (child : Fin r × Bool ≃ V2)
    (hloop : ∀ (i : Fin r) (b : Bool), parent i ≠ Sum.inr (child (i, b))) :
    SimpleGraph (V1 ⊕ V2) where
  Adj x y :=
    (∃ u v : V1, x = Sum.inl u ∧ y = Sum.inl v ∧ T1.Adj u v) ∨
    (∃ p : Fin r × Bool,
      (x = parent p.1 ∧ y = Sum.inr (child p)) ∨
      (y = parent p.1 ∧ x = Sum.inr (child p)))
  symm := ⟨fun x y h => by
    rcases h with ⟨u, v, rfl, rfl, huv⟩ | ⟨p, ⟨h1, h2⟩ | ⟨h1, h2⟩⟩
    · exact Or.inl ⟨v, u, rfl, rfl, (T1.adj_comm _ _).mp huv⟩
    · exact Or.inr ⟨p, Or.inr ⟨h1, h2⟩⟩
    · exact Or.inr ⟨p, Or.inl ⟨h1, h2⟩⟩⟩
  loopless := ⟨fun x h => by
    rcases h with ⟨u, v, hx1, hx2, huv⟩ | ⟨p, ⟨h1, h2⟩ | ⟨h1, h2⟩⟩
    · cases hx1.symm.trans hx2
      exact T1.loopless.irrefl u huv
    · exact hloop p.1 p.2 (h1.symm.trans h2)
    · exact hloop p.1 p.2 (h1.symm.trans h2)⟩
```

**HartsfieldAntimagic.graftedHoleJumpTree (HartsfieldAntimagic/BSAShellLifting.lean:2017-2021) — Definition "Named graphs and labelings" (Section 2)**

```
def graftedHoleJumpTree (k : ℕ) : SimpleGraph (HoleJumpVert k ⊕ (Fin 5 × Bool)) :=
  graftedGraph (holeJumpStar k) (graftedHoleJumpParent k) (Equiv.refl _) (fun i b h
    ↪ => by
    have hlt := graftedHoleJumpDepth_lt k i b
    rw [h] at hlt
    exact Nat.lt_irrefl _ hlt)
```

**HartsfieldAntimagic.HoleJumpVert (HartsfieldAntimagic/BSAShellLifting.lean:1508-1508) — Definition "Named graphs and labelings" (Section 2)**

```
abbrev HoleJumpVert (k : ℕ) := Fin (6 * k + 1) ⊕ (Fin 2 × Bool)
```

**HartsfieldAntimagic.SubdivStarVert (HartsfieldAntimagic/BSAShellLifting.lean:1156-1156) — Definition "Named graphs and labelings" (Section 2)**

```
abbrev SubdivStarVert (s : ℕ) := Option (Fin (2 * s + 1) × Bool)
```

**HartsfieldAntimagic.IsEdgeTransitive (HartsfieldAntimagic/RegularDeletion.lean:305-306) — Definition "Named graphs and labelings" (Section 2)**

*Context binders in scope:* `variable {V : Type*}`

```
def IsEdgeTransitive (G : SimpleGraph V) : Prop :=
  ∀ e₁ e₂ : G.edgeSet, ∃ φ : G ≃g G, Sym2.map φ e₁.1 = e₂.1
```

**HartsfieldAntimagic.vertexSumPoly (HartsfieldAntimagic/NullstellensatzObstructions.lean:57-59) — Definition "Polynomials of the Combinatorial Nullstellensatz" (Section 2)**

```
noncomputable def vertexSumPoly {n m : ℕ} (G : SimpleGraph (Fin n)) (e : Fin m ≃
  ↪ G.edgeSet)
    (v : Fin n) : MvPolynomial (Fin m) ℤ :=
  ∑ i : Fin m, if v ∈ ((e i : G.edgeSet) : Sym2 (Fin n)) then X i else 0
```

**HartsfieldAntimagic.vertexDiffPoly (HartsfieldAntimagic/NullstellensatzObstructions.lean:62-64) — Definition "Polynomials of the Combinatorial Nullstellensatz" (Section 2)**

```
noncomputable def vertexDiffPoly {n m : ℕ} (G : SimpleGraph (Fin n)) (e : Fin m ≃
  ↪ G.edgeSet) :
    MvPolynomial (Fin m) ℤ :=
  ∏ u : Fin n, ∏ v ∈ Finset.Ioi u, (vertexSumPoly G e v - vertexSumPoly G e u)
```

**HartsfieldAntimagic.edgeVandermondePoly (HartsfieldAntimagic/NullstellensatzObstructions.lean:67-68) — Definition "Polynomials of the Combinatorial Nullstellensatz" (Section 2)**

```
noncomputable def edgeVandermondePoly (m : ℕ) : MvPolynomial (Fin m) ℤ :=
  ∏ i : Fin m, ∏ j ∈ Finset.Ioi i, (X j - X i)
```

**HartsfieldAntimagic.boxMonomial (HartsfieldAntimagic/NullstellensatzObstructions.lean:76-77) — Definition "Polynomials of the Combinatorial Nullstellensatz" (Section 2)**

```
noncomputable def boxMonomial (n : ℕ) : Fin n →₀ ℕ :=
  Finsupp.equivFunOnFinite.symm (fun _ => n - 1)
```

**HartsfieldAntimagic.hefetzBoxCoeff (HartsfieldAntimagic/NullstellensatzObstructions.lean:89-90) — Definition "Polynomials of the Combinatorial Nullstellensatz" (Section 2)**

```
noncomputable def hefetzBoxCoeff (n : ℕ) (P : MvPolynomial (Fin n) ℤ) : ℤ :=
  (edgeVandermondePoly n * P).coeff (boxMonomial n)
```

**HartsfieldAntimagic.hefetzDegreeGap (HartsfieldAntimagic/NullstellensatzObstructions.lean:251-252) — Definition "Polynomials of the Combinatorial Nullstellensatz" (Section 2)**

```
def hefetzDegreeGap (n m : ℕ) : ℤ :=
  (Nat.choose n 2 : ℤ) - (Nat.choose m 2 : ℤ)
```

**HartsfieldAntimagic.Iso.edgeEquiv (HartsfieldAntimagic/Basic.lean:293-312) — Definition "Polynomials of the Combinatorial Nullstellensatz" (Section 2)**

*Context binders in scope:* `variable {V : Type*}`

```
def Iso.edgeEquiv {W : Type*} {G : SimpleGraph V} {H : SimpleGraph W} (φ : G ≃g H) :
    G.edgeSet ≃ H.edgeSet where
  toFun := fun e => ⟨Sym2.map φ e.1, by
    rcases e with ⟨e, he⟩
    induction e using Sym2.ind with
    | _ u v => exact H.mem_edgeSet.mpr ((φ.map_rel_iff).mpr (G.mem_edgeSet.mp he))⟩
  invFun := fun e => ⟨Sym2.map φ.symm e.1, by
    rcases e with ⟨e, he⟩
    induction e using Sym2.ind with
    | _ u v => exact G.mem_edgeSet.mpr ((φ.symm.map_rel_iff).mpr (H.mem_edgeSet.mp
      ↪ he))⟩
  left_inv := by
    rintro ⟨e, he⟩
    apply Subtype.ext
    induction e using Sym2.ind with
    | _ u v => simp
  right_inv := by
    rintro ⟨e, he⟩
    apply Subtype.ext
    induction e using Sym2.ind with
    | _ u v => simp
```

**Selected Concrete Sanity Examples (checked in Lean; examples only, not global identity proofs):** HartsfieldAntimagic.antimagicCount_pawGraph (antimagicCount pawGraph = 8); HartsfieldAntimagic.antimagicCount_K4 (antimagicCount (⊤ : SimpleGraph (Fin 4)) = 240); HartsfieldAntimagic.c3_hefetzBoxCoeff (hefetzBoxCoeff 3 (vertexDiffPoly c3Graph c3EdgeEquiv) = -6).

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

**HartsfieldAntimagic.regular_zero_shift_deletion (HartsfieldAntimagic/RegularDeletion.lean:282-292)** src:ee964c6d855ba4

*Context binders in scope:* `variable {V : Type*}`

```
theorem regular_zero_shift_deletion [Fintype V] [DecidableEq V] (G : SimpleGraph V)
    [DecidableRel G.Adj] {r : ℕ} (hreg : G.IsRegularOfDegree r) (hm : 1 ≤
      ↪ G.edgeFinset.card) :
    antimagicCount G = ∑ e : G.edgeSet, antimagicCount (G.deleteEdges {e.1})
```

**Assumptions:** G is a simple graph on a finite type with decidable equality and decidable adjacency; G is r-regular for some natural r; G has at least one edge. G - e is deleteEdges {e}, on the same vertex type. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

**HartsfieldAntimagic.bsa_modular_shell_lifting (HartsfieldAntimagic/BSAShellLifting.lean:248-567)** src:f5320517d9c873

*Context binders in scope:* `variable {V1 V2 : Type*} ; variable [Fintype V1] [DecidableEq V1] [Fintype V2] [DecidableEq V2]`

```
theorem bsa_modular_shell_lifting {T1 : SimpleGraph V1} [Fintype T1.edgeSet]
    {T : SimpleGraph (V1 ⊕ V2)} [Fintype T.edgeSet]
    {q r : ℕ} {parent : Fin r → V1 ⊕ V2} {child : Fin r × Bool ≃ V2}
    (hg : IsEvenChildrenGrafting T1 T r parent child)
    {ψ : SignedEdgeLabeling T1} (hψ : IsBSALabeling T1 q ψ) :
    IsModAntimagicLabeling T (bsaShellLiftLabeling (q := q) hg ψ)
```

**Assumptions:** T1 a simple graph on V1 and T on V1 + V2 with finite edge sets; q, r natural; parent : Fin r -> V1 + V2 and child : Fin r x Bool ~ V2 satisfy IsEvenChildrenGrafting; psi is a BSA labeling of T1 with parameter q (so T1 is a tree with 2q edges on 2q + 1 vertices, q >= 1). Finiteness and decidable equality of V1 and V2 are section variables. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

**HartsfieldAntimagic.holeJumpStar_isBSALabeling (HartsfieldAntimagic/BSAShellLifting.lean:1875-1974)** src:34d8adcfa7a1de

```
theorem holeJumpStar_isBSALabeling (k : ℕ) (hk : 1 ≤ k) :
    IsBSALabeling (holeJumpStar k) (3 * k + 2) (holeJumpBSALabeling k)
```

**Assumptions:** k a natural number with 1 <= k. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

**HartsfieldAntimagic.graftedHoleJump_odd_and_even_antimagic (HartsfieldAntimagic/BSAShellLifting.lean:2027-2072)** src:178d6d2dbd12de

```
theorem graftedHoleJump_odd_and_even_antimagic (k : ℕ) (hk : 1 ≤ k)
    [Fintype (graftedHoleJumpTree k).edgeSet] :
    (graftedHoleJumpTree k).IsTree ∧
      Fintype.card (HoleJumpVert k ⊕ (Fin 5 × Bool)) = 6 * k + 15 ∧
      IsAntimagic (graftedHoleJumpTree k) ∧
      ∃ (v0 : HoleJumpVert k ⊕ (Fin 5 × Bool)),
        ∀ [Fintype (leafExtendGraph (graftedHoleJumpTree k) v0).edgeSet],
          (leafExtendGraph (graftedHoleJumpTree k) v0).IsTree ∧
            Fintype.card (Option (HoleJumpVert k ⊕ (Fin 5 × Bool))) = 6 * k + 16 ∧
            IsAntimagic (leafExtendGraph (graftedHoleJumpTree k) v0)
```

**Assumptions:** k a natural number with 1 <= k; any Fintype instance on the edge sets. The leaf extension is leafExtendGraph on Option of the vertex type. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

**HartsfieldAntimagic.bsa_grafted_isAntimagic (HartsfieldAntimagic/BSAShellLifting.lean:571-578)** src:9375b95e98598c

*Context binders in scope:* `variable {V1 V2 : Type*} ; variable [Fintype V1] [DecidableEq V1] [Fintype V2] [DecidableEq V2]`

```
theorem bsa_grafted_isAntimagic {T1 : SimpleGraph V1} [Fintype T1.edgeSet]
    {T : SimpleGraph (V1 ⊕ V2)} [Fintype T.edgeSet]
    {q r : ℕ} {parent : Fin r → V1 ⊕ V2} {child : Fin r × Bool ≃ V2}
    (hg : IsEvenChildrenGrafting T1 T r parent child)
    (hcore : IsBSACore T1 q) :
    IsAntimagic T
```

**Assumptions:** As in Theorem lift, with IsBSACore T1 q in place of a given labeling. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

**HartsfieldAntimagic.even_order_zero_mod_leaf_extension (HartsfieldAntimagic/BSAShellLifting.lean:773-919)** src:129fc21f9bfae8

*Context binders in scope:* `variable {V : Type*} ; variable [Fintype V] [DecidableEq V]`

```
theorem even_order_zero_mod_leaf_extension (G : SimpleGraph V) [Fintype G.edgeSet]
    (hconn : G.Connected) (hm : 2 ≤ G.edgeFinset.card)
    {f : EdgeLabeling G} (hf : IsModAntimagicLabeling G f) :
    ∃ (v0 : V), (vertexSum G f v0 : ZMod (G.edgeFinset.card + 1)) = 0 ∧
      ∀ [Fintype (leafExtendGraph G v0).edgeSet], IsAntimagic (leafExtendGraph G v0)
```

**Assumptions:** G a simple graph on a type with decidable equality (the type is not assumed finite), finite edge set, connected, with at least 2 edges; f a ZMod (m + 1)-antimagic labeling. The conclusion holds for every Fintype instance on the edge set of the extension. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

**HartsfieldAntimagic.path_isBSACore (HartsfieldAntimagic/BSAShellLifting.lean:1139-1140)** src:a760efa1e7dc98

```
theorem path_isBSACore (t : ℕ) (ht : 1 ≤ t) : IsBSACore (pathGraph (2 * t + 1)) t
```

**Assumptions:** t a natural number with 1 <= t. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

**HartsfieldAntimagic.subdivStar_isBSACore (HartsfieldAntimagic/BSAShellLifting.lean:1489-1491)** src:5469ab69751f6a

```
theorem subdivStar_isBSACore (s : ℕ) (hs : 1 ≤ s) :
    IsBSACore (subdivStar s) (2 * s + 1)
```

**Assumptions:** s a natural number with 1 <= s. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

**HartsfieldAntimagic.holeJump_sum_leg0 (HartsfieldAntimagic/BSAShellLifting.lean:1735-1846)** src:949c4a6296cabe

```
theorem holeJump_sum_leg0 (k : ℕ) (hk : 1 ≤ k) (v : Fin (6 * k + 1)) :
    signedVertexSum (holeJumpStar k) (holeJumpBSALabeling k) (Sum.inl v) =
      if v.1 = 0 then ((3 * k + 1 : ℕ) : ℤ)
      else if v.1 < 2 * k then ((3 * k + 1 - v.1 : ℕ) : ℤ)
      else if v.1 < 3 * k then ((3 * k - v.1 : ℕ) : ℤ)
      else if v.1 = 3 * k then 0
      else if v.1 ≤ 4 * k then -((v.1 - 3 * k : ℕ) : ℤ)
      else if v.1 < 6 * k then -((v.1 + 1 - 3 * k : ℕ) : ℤ)
      else -((3 * k + 1 : ℕ) : ℤ)
```

**Assumptions:** k a natural number with 1 <= k; v in Fin (6k + 1) is x_v. The subtractions are of natural numbers and are taken only where they do not truncate. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

**HartsfieldAntimagic.regular_edgeTransitive_deletion (HartsfieldAntimagic/RegularDeletion.lean:325-340)** src:8593beffad4fa4

*Context binders in scope:* `variable {V : Type*}`

```
theorem regular_edgeTransitive_deletion [Fintype V] [DecidableEq V] (G : SimpleGraph
  ↪ V)
    [DecidableRel G.Adj] {r : ℕ} (hreg : G.IsRegularOfDegree r) (het :
      ↪ IsEdgeTransitive G)
    (e0 : G.edgeSet) :
    antimagicCount G = G.edgeFinset.card * antimagicCount (G.deleteEdges {e0.1})
```

**Assumptions:** G is a simple graph on a finite type with decidable equality and decidable adjacency, r-regular for some natural r, and edge-transitive; e0 any edge. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

**HartsfieldAntimagic.antimagicCount_cycleGraph (HartsfieldAntimagic/RegularDeletion.lean:583-591)** src:0afc323fae7d8c

```
theorem antimagicCount_cycleGraph (n : ℕ) (hn : 3 ≤ n) :
    antimagicCount (cycleGraph n) = n * antimagicCount (pathGraph n)
```

**Assumptions:** n a natural number with 3 <= n. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

**HartsfieldAntimagic.antimagicCount_completeGraph (HartsfieldAntimagic/RegularDeletion.lean:664-668)** src:7a8887e4db58e5

```
theorem antimagicCount_completeGraph (n : ℕ) (e0 : (⊤ : SimpleGraph (Fin n)).edgeSet)
  ↪ :
    antimagicCount (⊤ : SimpleGraph (Fin n)) =
      Nat.choose n 2 * antimagicCount ((⊤ : SimpleGraph (Fin n)).deleteEdges {e0.1})
```

**Assumptions:** n any natural number; e0 any edge of the complete graph on Fin n (so n >= 2 in effect). Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

**HartsfieldAntimagic.antimagicCount_starGraph (HartsfieldAntimagic/RegularDeletion.lean:680-768)** src:74bff51a0b0227

```
theorem antimagicCount_starGraph (m : ℕ) (hm : 2 ≤ m) :
    antimagicCount (starGraph m) = Nat.factorial m
```

**Assumptions:** m a natural number with 2 <= m. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

**HartsfieldAntimagic.hefetzDegreeGap_le_zero_iff (HartsfieldAntimagic/NullstellensatzObstructions.lean:261-278)** src:51339093fad914

```
theorem hefetzDegreeGap_le_zero_iff {n m : ℕ} (hn : 2 ≤ n) :
    hefetzDegreeGap n m ≤ 0 ↔ n ≤ m
```

**Assumptions:** n, m natural numbers with 2 <= n. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

**HartsfieldAntimagic.no_box_monomial_of_degreeGap_pos (HartsfieldAntimagic/NullstellensatzObstructions.lean:343-348)** src:5c592b9b23f1e6

```
theorem no_box_monomial_of_degreeGap_pos {n m : ℕ} (hgap : 0 < hefetzDegreeGap n m)
    (d : Fin m →₀ ℕ) (hd : ∀ i : Fin m, d i ≤ m - 1) :
    Finsupp.degree d ≠ Nat.choose m 2 + Nat.choose n 2
```

**Assumptions:** n, m natural numbers with Gamma(n, m) > 0; d any exponent vector on Fin m with every entry at most m - 1 (natural subtraction; for m = 0 there is no entry). Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

**HartsfieldAntimagic.hefetzDegreeGap_isTree (HartsfieldAntimagic/NullstellensatzObstructions.lean:320-329)** src:c382ecb4c4cbe6

```
theorem hefetzDegreeGap_isTree {n : ℕ} (hn : 2 ≤ n) (T : SimpleGraph (Fin n))
    [Fintype T.edgeSet] (hT : T.IsTree) :
    hefetzDegreeGap n T.edgeFinset.card = (n : ℤ) - 1 ∧
      0 < hefetzDegreeGap n T.edgeFinset.card
```

**Assumptions:** n a natural number with 2 <= n; T a simple graph on Fin n with finite edge set that is a tree. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

**HartsfieldAntimagic.isAntimagic_of_hefetzBoxCoeff_ne_zero (HartsfieldAntimagic/NullstellensatzObstructions.lean:748-841)** src:aa3f16f7cb7744

```
theorem isAntimagic_of_hefetzBoxCoeff_ne_zero {n : ℕ} (G : SimpleGraph (Fin n))
    [Fintype G.edgeSet] (e : Fin n ≃ G.edgeSet)
    (hnonzero : hefetzBoxCoeff n (vertexDiffPoly G e) ≠ 0) :
    IsAntimagic G
```

**Assumptions:** G a simple graph on Fin n with finite edge set; e : Fin n ~ E(G) an indexing of its edges, so m = n; c_box(G) for this indexing is nonzero. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

**HartsfieldAntimagic.hefetzBoxCoeff_eq_zero_of_signReversingAut (HartsfieldAntimagic/NullstellensatzObstructions.lean:1049-1070)**

src:86dced62772abe

```
theorem hefetzBoxCoeff_eq_zero_of_signReversingAut {n : ℕ} (G : SimpleGraph (Fin n))
    (e : Fin n ≃ G.edgeSet) (φ : G ≃g G)
    (hrev : (Equiv.Perm.sign φ.toEquiv : ℤ) *
      (Equiv.Perm.sign (e.trans ((Iso.edgeEquiv φ).trans e.symm)) : ℤ) = -1) :
    hefetzBoxCoeff n (vertexDiffPoly G e) = 0
```

**Assumptions:** G a simple graph on Fin n; e : Fin n ~ E(G); phi : G ~ G an automorphism with the sign product equal to -1. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

**HartsfieldAntimagic.evenCycle_hefetzBoxCoeff_eq_zero (HartsfieldAntimagic/NullstellensatzObstructions.lean:1248-1372)** src:21ac0a15128057

```
theorem evenCycle_hefetzBoxCoeff_eq_zero (k : ℕ) (hk : 2 ≤ k)
    (e : Fin (2 * k) ≃ (cycleGraph (2 * k)).edgeSet) :
    hefetzBoxCoeff (2 * k) (vertexDiffPoly (cycleGraph (2 * k)) e) = 0
```

**Assumptions:** k a natural number with 2 <= k; e : Fin (2k) ~ E(C_{2k}) any indexing. Applicability: Instantiable — closed instantiation witness PENDING gate-box replay.

**Table 3.** Lean 4 Type-Trap Audit Checklist — Source-Level Approximation (Gate Replay Pending).[a]

| Declaration | Hazard | Operand | Category | Audited Disposition & Guard |
|---|---|---|---|---|
| antimagicCount | nat_div | / / | intended_convention | The characters are the subtype bar { f // .. . } of Nat.card, not a division. |
| antimagicCount | nat_div | / IsAntimagicLabeling | intended_convention | The subtype bar { f // IsAntimagic Labeling G f } again. |
| antimagicCount_complete Graph | classical_ choose | Classical.choose | intended_convention | Not a choice function: Nat.choose is the binomial coefficient, binom(n, 2) of Definition cn and Corollary complete; the tool flags every name ending in .choose. |
| hefetzDegreeGap | classical_ choose | Classical.choose | intended_convention | Not a choice function: Nat.choose is the binomial coefficient, binom(n, 2) of Definition cn and Corollary complete; the tool flags every name ending in .choose. |
| no_box_monomial_of_degree Gap_pos | classical_ choose | Classical.choose | intended_convention | Not a choice function: Nat.choose is the binomial coefficient, binom(n, 2) of Definition cn and Corollary complete; the tool flags every name ending in .choose. |
| boxMonomial | nat_sub | - 1 | intended_convention | The exponent n - 1; Proposition cn has n >= 1 whenever there is a variable, and for n = 0 the monomial is empty. |
| no_box_monomial_of_degree Gap_pos | nat_sub | - 1 | intended_convention | The bound d_i <= m - 1 is quantified over i in Fin m, so m >= 1 whenever it is used. |
| bsaShellLiftLabeling | nat_sub | - q | intended_convention | M - q - (i + 1) with M = 2q + 2r + 1 and i + 1 <= r is at least q + 1 > 0, so neither subtraction truncates; this is M - q - i of Definition graft with i from 1. |
| bsaShellLiftLabeling | nat_sub | - (i.1 + 1) | intended_convention | As for - q: M - q >= q + 2r + 1 > i + 1. |
| bsaShellLiftLabeling | int_to_nat | .toNat | intended_convention | On the branch psi <= 0 the label is Int.to Nat (M + psi), that is max(M + psi, 0) of Definition graft; for a signed labeling with psi < -M it would be 0, which the theorems never use, since they assume a BSA labeling. |
| cycleGraph | nat_mod | % n | intended_convention | The edge {i, i + 1 mod n} of Definition named; modulo n is the intended reduction. |
| edgeVandermondePoly | nat_sub | - X | intended_convention | x_j - x_i in MvPolynomial (Fin m) Z, a ring; no truncation. |
| vertexDiffPoly | nat_sub | - vertexSumPoly | intended_convention | s_v(x) - s_u(x) in MvPolynomial (Fin m) Z, a ring. |
| holeJumpLeg0Val | nat_sub | - holeJumpZ | intended_convention | The label -z of Definition named, an integer. |
| holeJumpLeg0Val | nat_sub | - 1 | intended_convention | 6k - 1 - j with 3k <= j < 6k is 6k - 1 - j >= 0, the 0-indexed j of z_{6k-j}. |
| holeJumpLeg0Val | nat_sub | - j | intended_convention | As for - 1: j <= 6k - 1 on the long leg. |
| holeJumpZ | nat_mod | % 2 | intended_convention | The parity test of Definition named. |
| holeJumpZ | nat_div | / 2 | intended_convention | j / 2 with natural division: for even j it is j/2, for odd j it is (j - 1)/2, which gives the indices 3k - (i - 1)/2 and i/2 of Definition named for i = j + 1. |

| Declaration | Hazard | Operand | Category | Audited Disposition & Guard |
|---|---|---|---|---|
| holeJumpZ | nat_sub | - j | intended_convention | 3k - j / 2 with j < 3k is at least 1. |
| holeJump_sum_leg0 | nat_sub | - v.1 | intended_convention | 3k + 1 - v and 3k - v occur under v < 2k and v < 3k, where they do not truncate. |
| holeJump_sum_leg0 | nat_sub | - ((v.1 - 3 * k : ℕ) : ℤ) | intended_convention | v - 3k occurs under 3k < v, where it does not truncate; the minus in front is integer negation. |
| holeJump_sum_leg0 | nat_sub | - 3 | intended_convention | v - 3k and v + 1 - 3k occur under v > 3k. |
| holeJump_sum_leg0 | nat_sub | - ((v.1 + 1 - 3 * k : ℕ) : ℤ) | intended_convention | v + 1 - 3k occurs under 4k < v. |
| instDecidableRelPathGraph | local_instance_or_notation | instDecidableRelPathGraph | intended_convention | Decidability instance for the adjacency relation of P_n, found by instance search; it decides the relation of the definition and no statement depends on which instance is used. |
| instDecidableRelSubdivStar | local_instance_or_notation | instDecidableRelSubdivStar | intended_convention | Decidability instance for the adjacency relation of S(2^{2s+1}), found by instance search; it decides the relation of the definition and no statement depends on which instance is used. |
| instDecidableRelHoleJumpStar | local_instance_or_notation | instDecidableRelHoleJumpStar | intended_convention | Decidability instance for the adjacency relation of S(6k,2,2), found by instance search; it decides the relation of the definition and no statement depends on which instance is used. |
| instDecidableRelStarGraph | local_instance_or_notation | instDecidableRelStarGraph | intended_convention | Decidability instance for the adjacency relation of K_{1,m}, found by instance search; it decides the relation of the definition and no statement depends on which instance is used. |
| instDecidableRelCycleGraph | local_instance_or_notation | instDecidableRelCycleGraph | intended_convention | Decidability instance for the adjacency relation of C_n, found by instance search; it decides the relation of the definition and no statement depends on which instance is used. |
| instDecidableRelGraphOfEdges | local_instance_or_notation | instDecidableRelGraphOfEdges | intended_convention | Decidability instance for the adjacency relation of a graph given by an edge list, found by instance search; it decides the relation of the definition and no statement depends on which instance is used. |
| decidableRel_deleteEdges_singleton | local_instance_or_notation | decidableRel_deleteEdges_singleton | intended_convention | Decidability instance for the adjacency relation of G - e, found by instance search; it decides the relation of the definition and no statement depends on which instance is used. |
| finite_bijectiveEdgeLabeling | local_instance_or_notation | finite_bijectiveEdgeLabeling | intended_convention | Finiteness instance (bijective labelings): the labelings in question are maps into {1, ..., m} on a finite edge set, so the type is finite; Nat.card counts its elements and no statement depends on which instance is used. |
| finite_antimagicLabeling | local_instance_or_notation | finite_antimagicLabeling | intended_convention | Finiteness instance (antimagic labelings): the labelings in question are maps into {1, ..., m} on a finite edge set, so the type is finite; Nat.card counts its elements and 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 4.** 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: Let G be a finite r-regular graph with at least one edge. Then A(G) equals the sum over the edges e of G of A(G - e), where G - e has the vertex set of G. | Lean Kernel | `HartsfieldAntimagic.regular_zero_shift_deletion` | Axioms: Classical.choice, Quot.sound, propext; fidelity: exact | Lean 4 kernel with the axioms propext, Classical.choice and Quot.sound; the definitions of Section 2; the hypotheses as stated. |
| Prop. 2: (a) For 4 <= n <= 10 the path P_n is the only tree on n vertices with the fewest antimagic labelings. (b) For 4 <= n <= 10 the star K_{1,n-1} is the only tree on n vertices all of whose (n - 1)! bijective labelings are antimagic. | Cert Only | recompute.py (7c243c36139b) [regenerable; replay pending] | External cert only; fidelity: n/a | Exhaustive enumeration of recompute.py: every tree on 4 to 10 vertices up to isomorphism, with all bijective labelings counted. |
| Prop. 3: (a) A(P_n) = 2, 6, 16, 92, 352, 1910, 10892 for n = 4, ..., 10. (b) Each of the 141 connected graphs on 3 to 6 vertices is antimagic, and A(G) >= 2 for each of the 89 among them with at most 5 vertices, or with 6 vertices and at most 8 edges. | Cert Only | recompute.py (7c243c36139b) [regenerable; replay pending] | External cert only; fidelity: n/a | Exhaustive enumeration of recompute.py: every path P_n for 4 <= n <= 10 counted exactly; every connected graph on 3 to 6 vertices up to isomorphism (141) searched for an antimagic labeling; the 89 with at most 5 vertices, or 6 vertices and at most 8 edges, counted exactly, and also K_{3,3} and the prism (9 edges); the other 50 only searched. |
| Thm. 4: Let psi be a BSA labeling with parameter q of a graph T1 and T an even-children grafting of T1 with r pairs. With m = 2q + 2r and M = m + 1, the lifted labeling phi is a Z_M-antimagic labeling of T. | Lean Kernel | `HartsfieldAntimagic.bsa_modular_shell_lifting` | Axioms: Classical.choice, Quot.sound, propext; fidelity: exact | Lean 4 kernel with the axioms propext, Classical.choice and Quot.sound; the definitions of Section 2; the hypotheses as stated. |
| Prop. 5: For every k >= 1 the labeling psi_k is a BSA labeling of S(6k,2,2) with parameter 3k + 2. | Lean Kernel | `HartsfieldAntimagic.holeJumpStar_isBSALabeling` | Axioms: Classical.choice, Quot.sound, propext; fidelity: exact | Lean 4 kernel with the axioms propext, Classical.choice and Quot.sound; the definitions of Section 2; the hypotheses as stated. |
| Thm. 6: For every k >= 1, T_k is a tree with 6k + 15 vertices and is antimagic, and there is a vertex v0 of T_k such that T_k with one pendant leaf attached to v0 is a tree with 6k + 16 vertices and is antimagic. | Lean Kernel | `HartsfieldAntimagic.graftedHoleJump_odd_and_even_antimagic` | Axioms: Classical.choice, Quot.sound, propext; fidelity: exact | Lean 4 kernel with the axioms propext, Classical.choice and Quot.sound; the definitions of Section 2; the hypotheses as stated. |
| Prop. 7: For every k >= 1, T_k and T_k' have maximum degree 3; T_k has 6k + 1 vertices of degree 2, six of degree 3 and eight leaves; T_k' has 6k vertices of degree 2, seven of degree 3 and nine leaves; in both, two vertices of degree 2 are adjacent, two vertices of degree 3 are adjacent, the even-degree vertices induce a forest with at least three components, the branch vertex c = x_0 of S(6k,2,2) has degree 3 and no leaf neighbor, no path contains all vertices of degree 3, and the diameter is 6k + 5. | Written | Written proof | Lean: none; fidelity: n/a | Written proof in this paper; not checked in Lean. |

| Paper Claim | Proof Tier | Lean Decl / Artifact | Actual Axioms & Fidelity | Assumptions & What You Must Trust |
|---|---|---|---|---|
| Cor. 8: For every k >= 1, neither T_k nor T_k' satisfies the hypotheses of any class in Table classes. | Written | Written proof | Lean: none; fidelity: n/a | Written proof in this paper; not checked in Lean. |
| Cor. 14: If T1 is a BSA core with parameter q and T is an even-children grafting of T1, then T is antimagic. | Lean Kernel | `HartsfieldAntimagic.bsa_grafted_isAntimagic` | Axioms: Classical.choice, Quot.sound, propext; fidelity: exact | Lean 4 kernel with the axioms propext, Classical.choice and Quot.sound; the definitions of Section 2; the hypotheses as stated. |
| Cor. 15: Let G be a connected graph with m >= 2 edges, on a vertex set not assumed finite, and f a Z_{m+1}-antimagic labeling of G. Then there is a vertex v0 with s_f(v0) = 0 mod (m + 1) such that G with a leaf attached at v0 is antimagic. | Lean Kernel | `HartsfieldAntimagic.even_order_zero_mod_leaf_extension` | Axioms: Classical.choice, Quot.sound, propext; fidelity: exact | Lean 4 kernel with the axioms propext, Classical.choice and Quot.sound; the definitions of Section 2; the hypotheses as stated. |
| Prop. 16: For every t >= 1, P_{2t+1} is a BSA core with parameter t. | Lean Kernel | `HartsfieldAntimagic.path_isBSACore` | Axioms: Classical.choice, Quot.sound, propext; fidelity: exact | Lean 4 kernel with the axioms propext, Classical.choice and Quot.sound; the definitions of Section 2; the hypotheses as stated. |
| Prop. 17: For every s >= 1, S(2^{2s+1}) is a BSA core with parameter 2s + 1. | Lean Kernel | `HartsfieldAntimagic.subdivStar_isBSACore` | Axioms: Classical.choice, Quot.sound, propext; fidelity: exact | Lean 4 kernel with the axioms propext, Classical.choice and Quot.sound; the definitions of Section 2; the hypotheses as stated. |
| Lem. 18: For k >= 1 and 0 <= j <= 6k, the signed vertex sum of psi_k at x_j is 3k + 1 if j = 0; 3k + 1 - j if 1 <= j < 2k; 3k - j if 2k <= j < 3k; 0 if j = 3k; -(j - 3k) if 3k < j <= 4k; -(j + 1 - 3k) if 4k < j < 6k; and -(3k + 1) if j = 6k. | Lean Kernel | `HartsfieldAntimagic.holeJump_sum_leg0` | Axioms: Classical.choice, Quot.sound, propext; fidelity: exact | Lean 4 kernel with the axioms propext, Classical.choice and Quot.sound; the definitions of Section 2; the hypotheses as stated. |
| Prop. 19: For every k >= 1, x_{3k} is the only vertex of T_k at which the sum of the lifted labeling of psi_k is divisible by 6k + 15, and T_k' is an antimagic tree with 6k + 16 vertices. | Written | Written proof | Lean: none; fidelity: n/a | Written proof in this paper; not checked in Lean. |
| Cor. 20: If G is r-regular and edge-transitive, then A(G) = m A(G - e0) for every edge e0. | Lean Kernel | `HartsfieldAntimagic.regular_edgeTransitive_deletion` | Axioms: Classical.choice, Quot.sound, propext; fidelity: exact | Lean 4 kernel with the axioms propext, Classical.choice and Quot.sound; the definitions of Section 2; the hypotheses as stated. |
| Cor. 21: For every n >= 3, A(C_n) = n A(P_n). | Lean Kernel | `HartsfieldAntimagic.antimagicCount_cycleGraph` | Axioms: Classical.choice, Quot.sound, propext; fidelity: exact | Lean 4 kernel with the axioms propext, Classical.choice and Quot.sound; the definitions of Section 2; the hypotheses as stated. |
| Cor. 22: For every n and every edge e0 of K_n, A(K_n) = binom(n, 2) A(K_n - e0). | Lean Kernel | `HartsfieldAntimagic.antimagicCount_completeGraph` | Axioms: Classical.choice, Quot.sound, propext; fidelity: exact | Lean 4 kernel with the axioms propext, Classical.choice and Quot.sound; the definitions of Section 2; the hypotheses as stated. |
| Prop. 23: For every m >= 2, A(K_{1,m}) = m!. | Lean Kernel | `HartsfieldAntimagic.antimagicCount_starGraph` | Axioms: Classical.choice, Quot.sound, propext; fidelity: exact | Lean 4 kernel with the axioms propext, Classical.choice and Quot.sound; the definitions of Section 2; the hypotheses as stated. |
| Prop. 24: For n >= 2 and every m, Gamma(n, m) <= 0 if and only if n <= m. | Lean Kernel | `HartsfieldAntimagic.hefetzDegreeGap_le_zero_iff` | Axioms: Classical.choice, Quot.sound, propext; fidelity: exact | Lean 4 kernel with the axioms propext, Classical.choice and Quot.sound; the definitions of Section 2; the hypotheses as stated. |

| Paper Claim | Proof Tier | Lean Decl / Artifact | Actual Axioms & Fidelity | Assumptions & What You Must Trust |
|---|---|---|---|---|
| Prop. 25: If Gamma(n, m) > 0, then no exponent vector (d_1, ..., d_m) with every d_i <= m - 1 has d_1 + ... + d_m = binom(m, 2) + binom(n, 2). | Lean Kernel | `HartsfieldAntimagic.no_box_monomial_of_degreeGap_pos` | Axioms: Classical.choice, Quot.sound, propext; fidelity: exact | Lean 4 kernel with the axioms propext, Classical.choice and Quot.sound; the definitions of Section 2; the hypotheses as stated. |
| Cor. 26: For a tree on n >= 2 vertices, Gamma(n, n - 1) = n - 1 > 0. | Lean Kernel | `HartsfieldAntimagic.hefetzDegreeGap_isTree` | Axioms: Classical.choice, Quot.sound, propext; fidelity: exact | Lean 4 kernel with the axioms propext, Classical.choice and Quot.sound; the definitions of Section 2; the hypotheses as stated. |
| Prop. 27: Let G have n vertices and n indexed edges. If c_box(G) is not 0, then G is antimagic. | Lean Kernel | `HartsfieldAntimagic.isAntimagic_of_hefetzBoxCoeff_ne_zero` | Axioms: Classical.choice, Quot.sound, propext; fidelity: exact | Lean 4 kernel with the axioms propext, Classical.choice and Quot.sound; the definitions of Section 2; the hypotheses as stated. |
| Prop. 28: Let G have n vertices and n indexed edges and phi an automorphism of G. If the sign of phi on the vertices times the sign of the induced permutation of the edge indices is -1, then c_box(G) = 0. | Lean Kernel | `HartsfieldAntimagic.hefetzBoxCoeff_eq_zero_of_signReversingAut` | Axioms: Classical.choice, Quot.sound, propext; fidelity: exact | Lean 4 kernel with the axioms propext, Classical.choice and Quot.sound; the definitions of Section 2; the hypotheses as stated. |
| Cor. 29: For every k >= 2 and every indexing of its edges, c_box(C_{2k}) = 0. | Lean Kernel | `HartsfieldAntimagic.evenCycle_hefetzBoxCoeff_eq_zero` | Axioms: Classical.choice, Quot.sound, propext; fidelity: exact | Lean 4 kernel with the axioms propext, Classical.choice and Quot.sound; the definitions of Section 2; the 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: 8bbfd6d3df04 | Lean mode: committed_receipt | Closure SHA-256: cd5f60747c8b | Git anchor: 849f300f5d44 | Axioms SHA-256: ca44a56ebf15 | Fidelity SHA-256: 817058003aca.

## 6. PRIOR WORK AND OPEN QUESTIONS

**Antimagic trees.** Table 5 lists the classes of antimagic trees that we compared with $T_k$ and $T'_k$, each restricted to trees on at least 3 vertices, with the descriptions as printed in Gallian's survey [2] or in the source when we read it, and the invariant of Proposition 7 that excludes both trees. The list is not all of the literature: Gallian's survey has other classes, of graphs that are not trees or of trees with other constraints, which we did not compare one by one.

The descriptions of [15, 16, 22, 18, 21, 25, 26, 27] are taken from Gallian's survey or from abstracts; we did not read these papers. Li and Wang state their result for $k$-shifted antimagic labelings, which include antimagic labelings at $k = 0$, and Chang, Chin, Li and Pan prove that double spiders are strongly antimagic, which implies antimagic [2]. The finding that this draft starts from also names subdivided caterpillars (Wu, Deng and Zhao) and a class of Chang, Chang and Pan; we did not read these sources, and they are not in the table. Computations have confirmed antimagic labelings for every tree on 3 to 25 vertices [4] [to check] and on 3 to 31 vertices [3] [to check]; this covers $T_1$, $T'_1$, $T_2$ and $T'_2$.

**Super edge-graceful and edge-graceful trees.** The tree results of Section 3 are close to the literature on edge-graceful labelings, and most of them are in it. Mitchem and Simoson [5] define super edge-graceful labelings and prove the two steps of Theorem 4 for trees grown by attaching pairs one at a time (Theorem 1 and Proposition 3 there), that regular spiders with an even number of edges are super edge-graceful (Theorem 5, which contains Proposition 17), and that $P_{2n+1}$ is (Corollary 10, which is Proposition 16); their Corollary 7 is the case of Theorem 4 for regular spiders grown by attaching pairs one at a time. Keene and Simoson [8] prove that every spider of odd order with three legs is edge-graceful by super edge-graceful labelings, which includes $S(6k, 2, 2)$; Proposition 5 gives the explicit labeling used here. Lee, Wang, Nowak and Wei [9] conjecture that every tree of odd order is super edge-graceful and prove it for trees with at most 17 vertices, by computer; `recompute.py` repeats the check for the 62 trees with 3, 5, 7 or 9 vertices. Krop, Mutiso and Raridan [6] treat trees of diameter 4, Lee and Ho [10] [to check] trees of odd order with three vertices of even degree, Lee, Sun, Wei, Wen and Yiu [12] [to check] and Locke and Wei [13] [to check] paths with pendant paths of length 2, Lee, Su and Wei [14] [to check] a further family, and Chung and Lee [11] [to check] spiders of even order. Lee's conjecture that every tree of odd order is edge-graceful is open [2]; Meng [20] proves it for trees with at most two vertices of degree 2. So $T_k$ is antimagic by a combination of published results (Proposition 5 replaces only the citation of [8]), and Theorem 6 adds the leaf extension and a Lean proof of the whole chain. We did not find the trees $T_k$ or $T'_k$ named as antimagic in the sources we read.

**Table 5.** Published classes of antimagic trees, restricted to trees on at least 3 vertices, and the invariant that excludes $T_k$ and $T'_k$ (all $k \geq 1$).

| class | excluded because |
|---|---|
| rooted trees in which every non-leaf vertex has at least two children (Kaplan, Lev, Roditty [15]); trees with at most one vertex of degree 2, a case of the forests of Sierra, Liu and Toy [19] [to check] | such a tree has at most one vertex of degree 2; ours have at least $6k \geq 6$ |
| $V_2$ and $V \setminus V_2$ both independent, or $V_2$ induces a path and all other vertices have odd degree (Liang, Wong, Zhu [16]) | two vertices of degree 2 are adjacent, so $V_2$ is not independent; $V_2$ induces at least three components, so not a path |
| even-degree vertices induce a path (Lozano, Mora, Seara, Tey [17]) | they induce at least three components |
| odd order and at most two vertices of degree 2 (Meng [20] [to check]) | at least $6k$ vertices of degree 2 |
| uniform subdivisions of trees with at most one vertex of degree 2 (Kuo, Li, Liu [21]) | in a subdivision of a tree with at least one subdivision vertex per edge, no two vertices of degree 3 are adjacent; without subdivision, at least $6k$ vertices of degree 2 |
| spiders (Shang [22]), double spiders (strongly antimagic; Chang, Chin, Li, Pan [23] [to check]), caterpillars [18] | the vertices of degree 3 do not lie on one path, while in each of these classes they do |
| connected graphs with at least 3 edges in which every vertex of degree at least 3 has a leaf neighbor (Mora, Tey [24] [to check]) | $c = x_0$ has degree 3 and no leaf neighbor |
| two copies of an antimagic tree joined by an edge at vertices of maximum degree (Muthukkumar, Rajendran [25]) | $T_k$ has odd order; in $T'_k$ the joined vertices would have degree 3, so each copy would be a path and $T'_k$ would have two vertices of degree 3, not seven |
| trees of diameter 4 or 5, except two cases (Li, Wang [26]) | diameter $6k + 5 \geq 11$ |
| complete $m$-ary trees (Chawathe, Krishna [27]) | at most one vertex of degree 2, or a path |

**Counting and the Nullstellensatz.** Beck and Farahmand [32] count weak antimagic labelings, with labels in $\{1, \dots, k\}$ that may repeat, and show that their number is a quasi-polynomial in $k$; they also count partially magic labelings. They note that, by the inside-out polytopes of Beck and Zaslavsky [33] [to check], the number of antimagic labelings of $G$ with distinct labels in $\{1, \dots, k\}$ is a quasi-polynomial in $k$; its value at $k = m$ is $A(G)$. The shift of all labels by a constant, which keeps the differences of vertex sums of a regular graph, is the observation behind Theorem 1; Cranston, Liang and Zhu use regularity in a different way [34]. Chalise and Low [35] apply the Nullstellensatz to $\mathbb{Z}_k$-antimagic labelings with labels in $\mathbb{Z}_k \setminus \{0\}$, a different problem from ours.

**Search log.** The search for prior work, with the records it read, is logged in the file `sources/searchlog.md`; its gate, `lit-search.py check`, exits with status 0 (output in `sources/litsearch-check.out`). Not searched: Hartsfield and Ringel's book; the papers of Hefetz, of Lee and Ho, of Lee, Su and Wei, of Lee, Sun, Wei, Wen and Yiu, of Locke and Wei and of Chung and Lee (only their reviews or survey entries were read); and the journal versions of the other papers marked [to check]. Within these limits, Theorem 1 and Proposition 2 were not found in the searches logged.

**Open questions.** (1) Is every tree of odd order super edge-graceful? This is the conjecture of Lee, Wang, Nowak and Wei [9]; by Theorem 4 it would make every even-children grafting of every tree of odd order at least 3 antimagic. (2) Keene and Simoson record as open a general procedure to edge-gracefully label all asymmetric spiders of odd order, even with four legs [8]. The signed question is separate: which spiders of odd order with more than three legs are super edge-graceful? (3) Is the path $P_n$ the tree with the fewest antimagic labelings for every $n \geq 4$? (4) Is there an identity like Theorem 1 for graphs that are not regular?

## 7. DATA AVAILABILITY, REPRODUCIBILITY, AND USE OF AI

The Lean 4 library `HartsfieldAntimagic` (Lean `v4.35.0` with Mathlib) with its axiom report and build receipt is in the bundle `findings/hartsfield-ringel-antimagic` of the `mathdisc` repository. The paper directory contains `recompute.py` and its output `recompute.out`, the figure scripts in `figures/`, the ledgers `claims.md` and `references-checked.md`, the search log in `sources/`, and the manifests behind Section 5. Running `python3 recompute.py` executes the Python box of Section 1 from this source and performs every computation of Section 4. Section 5 is written by `ourotrix paper trust -p . -g`, and `ourotrix paper check -p . -b -x -B` rebuilds the PDF and runs the checks recorded in `receipt.json`.

The finding record that this draft starts from calls Example 30 a counterexample to "CN Prop 2.4". In the finding's mathematical report, Proposition 2.4 is the counterexample itself, offered against an inference, by "sequential avoidance", from a nonzero staircase coefficient without the factor $\Delta_m$ to a nonvanishing point with distinct coordinates. For a general polynomial, a nonzero staircase coefficient does not guarantee nonvanishing at a permutation of $(1, 2, 3)$; a proof for the polynomials $P_G$ would need additional structure. We found no published proposition with that number in the version of the paper of

Cranston, Liang and Zhu [34] that we read (the arXiv version; the journal version is unread). The draft presents $F$ only as an example. `README.md` lists this and the other changes.

AI research agents built and directed by Tyler B. Hughes generated and checked this work: they found the statements, wrote and audited the Lean library, wrote the programs, and drafted this paper. No human mathematician has yet refereed the proof.

**Drafted:** November 2026 **Updated:** November 25, 2026

## REFERENCES


(1) N. Hartsfield and G. Ringel, *Pearls in Graph Theory*, Academic Press, San Diego, 1990. [to check]

(2) J. A. Gallian, *A dynamic survey of graph labeling*, Electron. J. Combin. (2026), #DS6, 29th edition (6 November 2026). [to check]

(3) M. Elisov, *Edge-graceful and antimagic labelings of all trees with at most 31 vertices: code and data*, Zenodo (2026), doi:10.5281/zenodo.23405211; repository `https://github.com/Hardos220/lee-edge-graceful-trees`. [to check]

(4) L. Branson, A. Sutton and X. Yan, *Finding antimagic labelings of trees by evolutionary search*, Proceedings of the 17th ACM/SIGEVO Conference on Foundations of Genetic Algorithms (FOGA '23) (2023), 27–37. [to check]

(5) J. Mitchem and A. Simoson, *On edge-graceful and super-edge-graceful graphs*, Ars Combin. 37 (1994), 97–111.

(6) E. Krop, F. Mutiso and C. Raridan, *On super edge-graceful trees of diameter four*, Congr. Numer. 210 (2011), 151–167; read as arXiv:1106.3063v1.

(7) S. Cichacz, D. Fronček and W. Xu, *Super edge-graceful paths*, arXiv:0804.3640v1 (2008). [to check]

(8) J. Keene and A. Simoson, *Balanced strands for asymmetric, edge-graceful spiders*, Ars Combin. 42 (1996), 49–64.

(9) S.-M. Lee, L. Wang, K. Nowak and W. Wei, *On the edge-graceful trees conjecture*, J. Combin. Math. Combin. Comput. 54 (2005), 83–98.

(10) S.-M. Lee and Y.-S. Ho, *All trees of odd order with three even vertices are super edge-graceful*, J. Combin. Math. Combin. Comput. 62 (2007), 53–64. [to check]

(11) P.-T. Chung and S.-M. Lee, *On the super edge-graceful spiders of even order*, J. Combin. Math. Combin. Comput. 64 (2008), 3–17. [to check]

(12) S.-M. Lee, H. Sun, W. Wei, Y. Wen and P. Yiu, *The super-edge-gracefulness of two infinite families of trees*, Congr. Numer. 190 (2008), 109–128. [to check]

(13) S. C. Locke and W. Wei, *Infinite families of super edge-graceful trees*, Congr. Numer. 195 (2009), 125–145. [to check]

(14) S.-M. Lee, H.-H. Su and W. Wei, *On a family of the super edge-graceful trees*, J. Combin. Math. Combin. Comput. 106 (2018), 23–36. [to check]

(15) G. Kaplan, A. Lev and Y. Roditty, *On zero-sum partitions and anti-magic trees*, Discrete Math. 309 (2009), 2010–2014. [to check]

(16) Y.-C. Liang, T.-L. Wong and X. Zhu, *Anti-magic labeling of trees*, Discrete Math. 331 (2014), 9–14. [to check]

(17) A. Lozano, M. Mora, C. Seara and J. Tey, *Trees whose even-degree vertices induce a path are antimagic*, Discuss. Math. Graph Theory 42 (2022), 959–966; read as arXiv:1905.06595.

(18) A. Lozano, M. Mora, C. Seara and J. Tey, *Caterpillars are antimagic*, Mediterr. J. Math. 18 (2021), 39. [to check]

(19) J. Sierra, D. D.-F. Liu and J. Toy, *Antimagic labelings of forests*, arXiv:2307.16836 (2023). [to check]

(20) L. Meng, *Trees of odd order with at most two vertices of degree two are edge-graceful*, arXiv:2608.23881 (2026). [to check]

(21) K.-N. Kuo, W.-T. Li and Y.-T. Liu, *A generalization of the subdivision of antimagic trees with at most one vertex of degree-2*, Involve 18 (2025), no. 5, doi:10.2140/involve.2025.18.821. [to check]

(22) J.-L. Shang, *Spiders are antimagic*, Ars Combin. 118 (2015), 367–372. [to check]

(23) F.-H. Chang, P. Chin, W.-T. Li and Z. Pan, *The strongly antimagic labelings of double spiders*, arXiv:1712.09477 (2017). [to check]

(24) M. Mora and J. Tey, *Antimagic and product antimagic graphs with pendant edges*, arXiv:2405.05375 (2024); Mediterr. J. Math. (2025). [to check]

(25) S. Muthukkumar and K. Rajendran, *Generation of anti-magic graphs*, International Journal of Analysis and Applications 23 (2025), 30. [to check]

(26) W.-T. Li and Y.-S. Wang, *Labeling trees of small diameters with consecutive integers*, Taiwanese J. Math. (2022). [to check]

(27) P. D. Chawathe and V. Krishna, *Antimagic labeling of complete m-ary trees*, Number Theory and Discrete Mathematics, Trends Math., Birkhäuser, Basel, 2002, 77–80. [to check]

(28) condeDeveloper, repository `antimagica`, `https://github.com/condeDeveloper/antimagica` (2026). [to check]

(29) Patrick-9936, repository `antimagic-exhaustive`, `https://github.com/Patrick-9936/antimagic-exhaustive` (2026). [to check]

(30) N. Alon, *Combinatorial Nullstellensatz*, Combin. Probab. Comput. 8 (1999), 7–29. [to check]

(31) D. Hefetz, *Anti-magic graphs via the combinatorial Nullstellensatz*, J. Graph Theory 50 (2005), 263–272. [to check]

(32) M. Beck and M. Farahmand, *Partially magic labelings and the Antimagic Graph Conjecture*, arXiv:1511.04154 (2015).

(33) M. Beck and T. Zaslavsky, *Inside-out polytopes*, Adv. Math. 205 (2006), 134–162, doi:10.1016/j.aim.2005.07.006. [to check]

(34) D. W. Cranston, Y.-C. Liang and X. Zhu, *Regular graphs of odd degree are antimagic*, J. Graph Theory 80 (2015), 28–33 [to check]; read as arXiv:1303.4850v1, posted by D. W. Cranston alone.

(35) P. Chalise and R. M. Low, *Application of the Combinatorial Nullstellensatz to magic-type graph labelings*, arXiv:2607.20724v1 (2026).

(36) OEIS Foundation, *The On-Line Encyclopedia of Integer Sequences*, sequences A000055 and A001349, `https://oeis.org` (accessed 25 November 2026).