

Divisorial Gonality of Banana Trees in Quadratic Time

Alper Ferudun*

19 October 2026

Abstract

A banana tree is a connected loopless multigraph whose underlying simple graph is a tree. We give an exact algorithm that computes its divisorial gonality and an effective divisor attaining it in $O(n^2)$ word operations and $O(n^2)$ space, where n is the number of vertices. Edge multiplicities may be supplied in binary. A rooted residue normal form reduces positive rank to positivity of one integer pile at every possible root. The rerooting rule makes different branches independent conditional on their parent pile. The resulting dynamic program is evaluated by sliding-window minima, with only $n + 1$ states at each vertex. This extends computation from banana paths to arbitrary branching and improves the cubic path bound of Beougher et al. The theorem concerns exact optimization; the gonality conjecture for banana trees was already proved in that work.

1 Statement and relation to earlier work

Divisor theory on graphs was developed by Baker and Norine [1]. Let T be a tree on $n \geq 1$ vertices. For each edge $e \in E(T)$ choose a positive integer a_e , and replace e by a_e parallel edges. The resulting multigraph G is a *banana tree*. A banana path is the special case in which T is a path. Multiplicities equal to one are allowed.

Beougher, Cibu, Ding, DiSilvio, Heyse, Kononova, Lee, Morrison and Singal [2, Theorem 1.1] give a cubic-time algorithm for banana paths. Their leaf recurrence does not immediately iterate at branching vertices, as explained before the proof of that theorem on page 8. We use a different state: a bounded pile in a rooted residue normal form. This state permits branching.

Theorem 1. *Let G be a banana tree on n vertices, given by its underlying tree and its positive integral edge multiplicities. After the input is read, its divisorial gonality and a positive-rank effective divisor of minimum degree can be computed in $O(n^2)$ word operations and $O(n^2)$ words of storage. All arithmetic after input truncation uses $O(\log n)$ -bit integers, with unit-cost arithmetic including division on such words.*

The cost does not depend on the sum of the multiplicities. For an expanded list of the m parallel edges, forming the multiplicity representation adds $O(m + n)$ time. For binary multiplicities, values greater than n can be truncated while reading; we explain this precisely in Section 5.

The genus is $g = \sum_e a_e - n + 1$. The inequality $\text{gon}(G) \leq \lfloor (g + 3)/2 \rfloor$ for banana trees is already [2, Theorem 1.3]. We neither claim that inequality as a new result nor settle its general connected-multigraph version. Classical chip-firing and normal-form principles are ingredients; the result here is the bounded-state exact optimization for arbitrary branching.

*Mercury Software GmbH. alper@mercurycodelab.com; <https://github.com/AlperTheKing>.

2 Divisors and a rooted normal form

A divisor is an integer vector $D \in \mathbb{Z}^{V(T)}$. Its degree is $\deg D = \sum_v D(v)$, and it is effective when $D(v) \geq 0$ for every v . Put

$$(Lf)(v) = \sum_{u: uv \in E(T)} a_{uv}(f(v) - f(u)).$$

Two divisors are equivalent, written $D \sim E$, if $D - E = Lf$ for some $f \in \mathbb{Z}^{V(T)}$. Write $[q]$ for the divisor with one chip at q and zero elsewhere. A divisor has rank at least one when $D - [q]$ is equivalent to an effective divisor for every vertex q . Its minimum possible degree is $\text{gon}(G)$.

The all-ones divisor has rank at least one, and hence

$$1 \leq \text{gon}(G) \leq n. \quad (1)$$

A minimum-degree positive-rank divisor may be assumed effective: for any q , adding $[q]$ to an effective representative of $D - [q]$ gives an effective representative of D .

Fix a root r . For $v \neq r$ write $p(v)$ for its parent, $a_v = a_{vp(v)}$, and T_v for its descendant subtree. An r -normal divisor is a divisor d satisfying

$$0 \leq d(v) < a_v \quad (v \neq r). \quad (2)$$

There is no restriction on $d(r)$ in this definition.

Lemma 2. *Every divisor is equivalent to a unique r -normal divisor. It can be found by a leaf-to-root sequence of integer divisions. A divisor is equivalent to an effective divisor if and only if the root coefficient of its r -normal form is nonnegative.*

Proof. Process the nonroot vertices in reverse tree order. If the current coefficient at v is $a_v k + b$ with $0 \leq b < a_v$, leave b at v and add $a_v k$ at $p(v)$. The change is $-kL\mathbf{1}_{T_v}$, because the only edges crossing the cut $T_v, V(T) \setminus T_v$ are the a_v copies of $vp(v)$. Thus the operation preserves equivalence and degree. Previously processed strict descendants are unchanged. This constructs a normal form, even when some coefficients or carries are negative.

For uniqueness, let $d - d' = Lf$ be the difference of two normal forms. For every $v \neq r$,

$$\sum_{u \in T_v} (d - d')(u) = a_v(f(v) - f(p(v))). \quad (3)$$

At a leaf, the left side is a difference of two residues in $[0, a_v - 1]$ divisible by a_v , so it is zero. Inducting upward, once all strict descendants agree, (3) gives the same conclusion at v . The root coefficients agree because the degrees agree.

If the initial divisor is effective, all carries in the construction are nonnegative, and the root stays nonnegative. Uniqueness therefore proves necessity in the final assertion. Sufficiency follows because a normal form with nonnegative root is itself effective. $\square$

For a divisor D , let $R_q(D)$ denote the root coefficient of its q -normal form.

Corollary 3. *A divisor D has rank at least one if and only if $R_q(D) \geq 1$ for every vertex q .*

Proof. Subtracting $[q]$ from the q -normal form changes only the root coefficient and leaves all nonroot residue conditions intact. Apply Lemma 2 to $D - [q]$. $\square$

3 Rerooting and positive rank

Let d be an effective r -normal divisor. The following rule computes all root coefficients using only one pass down the tree.

Lemma 4. *Set $R_r = d(r)$. For each nonroot vertex v , set*

$$R_v = d(v) + a_v \left\lfloor \frac{R_{p(v)}}{a_v} \right\rfloor. \quad (4)$$

Then $R_v = R_v(d)$ for every vertex v . If $\deg d \leq B$, all these piles lie in $\{0, \dots, B\}$.

Proof. Suppose the root has been moved from r along the unique path to $u = p(v)$. Its current coefficient is R_u , and the coefficient at v is still the original residue $d(v)$. Move $a_v \lfloor R_u/a_v \rfloor$ chips from u to v by firing the component on the u side of the cut the appropriate number of times. The coefficient left at u is $R_u \bmod a_v$; the new root coefficient is the right side of (4). The only edge whose parent orientation changes is uv . All other nonroot residue bounds therefore persist. By uniqueness, the new divisor is the v -normal form.

Starting at r and repeating along every root-to-vertex path proves the formula. Each move preserves nonnegativity and degree. Thus each root coefficient is at most the total degree. $\square$

Consequently, a choice of $d(r) \geq 0$ and the residues in (2) has positive rank precisely when (4) gives a positive pile at every vertex. The test uses a different equivalent representative for each target vertex. It does not require simultaneous coverage by one configuration, which is why the same parent pile may be used in the tests for different branches.

4 The dynamic program

Fix $B = n$. For each nonroot vertex v and $0 \leq s \leq B$, let $F_v(s)$ be the minimum sum of residues in T_v , conditional on s chips arriving at v from the parent, such that every pile in that subtree computed by (4) lies in $\{1, \dots, B\}$. The incoming chips are not charged to the subtree: they are already accounted for by the root pile and ancestor residues. The value is infinity when no such residue assignment exists.

If $\mathcal{C}(v)$ is the set of children of v , put

$$H_v(t) = \sum_{w \in \mathcal{C}(v)} F_w(a_w \lfloor t/a_w \rfloor), \quad 0 \leq t \leq B, \quad (5)$$

with the empty sum equal to zero.

Proposition 5. *For a nonroot vertex v ,*

$$F_v(s) = \min_{\substack{0 \leq d < a_v \\ 1 \leq s+d \leq B}} (d + H_v(s + d)). \quad (6)$$

Moreover,

$$\text{gon}(G) = \min_{1 \leq t \leq B} (t + H_r(t)). \quad (7)$$

A minimizing choice in each entry reconstructs an effective divisor attaining the minimum.

Proof. Choose the residue $d = d(v)$. The pile at v is $t = s + d$, which must lie between 1 and B . By (4), the incoming pile for a child w is fixed as $a_w \lfloor t/a_w \rfloor$. Conditional on t , the choices in different child subtrees involve disjoint residues and no further common variable. Their costs therefore add. Induction on the subtree size proves (6), in both directions: each feasible assignment is at least the indicated minimum, and minimizing child assignments can be combined to attain it.

At the root the charged coefficient is t itself, giving (7). Every finite choice reconstructs an effective normal divisor whose root piles are positive; by Corollary 3 it has positive rank. Conversely, take an optimal positive-rank divisor, make it effective and normalize it at r . Its degree is at most B by (1), and Lemma 4 puts all of its piles within the allowed range. It is therefore among the assignments optimized by (7).

For reconstruction, choose a minimizing root pile, and at each child look up the minimizing residue for the incoming pile specified by (4). Proceed downward. The sum of the chosen residues and root coefficient equals the computed minimum. $\square$

5 Sliding windows and the complexity bound

Directly evaluating (6) would use $O(B^2)$ operations per vertex. Put $K_v(t) = t + H_v(t)$. The same recurrence is

$$F_v(s) = -s + \min_{\max(1,s) \leq t \leq \min(B, s+a_v-1)} K_v(t). \quad (8)$$

As s increases, both endpoints of this interval are nondecreasing. All $B+1$ entries can therefore be evaluated in $O(B)$ operations using a deque of candidate indices. Here is explicit pseudocode; an empty minimum gives infinity.

```

 $Q \leftarrow$  empty deque;  $j \leftarrow 0$ .
For  $s = 0, \dots, B$ :
   $\ell \leftarrow \max(1, s)$ ;  $h \leftarrow \min(B, s + a_v - 1)$ .
  While  $j < h$ :
     $j \leftarrow j + 1$ .
    While  $Q \neq \emptyset$  and  $K_v(Q.\text{back}) > K_v(j)$ , remove the back.
    Append  $j$  to  $Q$ .
  While  $Q \neq \emptyset$  and  $Q.\text{front} < \ell$ , remove the front.
  If  $Q = \emptyset$ , set  $F_v(s) \leftarrow \infty$ .
  Otherwise set  $t \leftarrow Q.\text{front}$ ,  $F_v(s) \leftarrow K_v(t) - s$ ,
  and record  $t - s$  as the minimizing residue.
```

Each index enters the deque once and leaves at most once. Deleting a trailing index is safe: the newer index has no larger value and expires no sooner. Deleting a front index removes one that is outside the current window. The remaining front attains the minimum. Keeping the smaller index on ties also gives deterministic reconstruction.

Proof of Theorem 1. Root the tree and process it in reverse tree order. For each parent-child edge, forming (5) uses $O(B)$ array lookups and additions. Since there are $n-1$ edges, the total is $O(nB)$. The deque computation costs $O(B)$ per nonroot vertex, again $O(nB)$. Root minimization and reconstruction add $O(B+n)$. All cost and choice arrays occupy $O(nB)$ words.

It remains to bound the integer sizes and justify truncation. Represent costs greater than B by $B+1$, saturating each sum in (5). Nonnegative costs cannot turn a discarded value into a root answer of size at most B . In (8), however, form $K_v(t) = t + H_v(t)$ *before* subtracting s , and saturate only afterward. An infinite child cost still gives $K_v(t) - s \geq B+1$ because $t \geq s$. Thus saturation introduces no false feasible answer. Every intermediate cost is at most $2B+1$.

If $a_v > B$, then $a_v \lfloor t/a_v \rfloor = 0$ for all $t \leq B$, while $d < a_v$ imposes no additional restriction on $d \leq B$. Replacing every such multiplicity by $B+1$ therefore leaves all relevant recurrences unchanged. A witness for the truncated instance obeys the original residue intervals and rerooting rule, so it is valid for the original graph. Binary input can be truncated during reading without constructing a list of parallel edges. Subsequent integers have $O(\log n)$ bits. Taking $B = n$ proves the stated time and space bounds. $\square$

Remark 6. The case $n = 1$ has no nonroot states and gives gonality one. An edge of multiplicity one gives residue zero and transmits the entire pile. Thus bridges need no special treatment.

An edge of multiplicity greater than n transmits no chips in any optimal divisor's rerooting calculation.

6 Exact regression checks

The accompanying Python implementation uses integer arithmetic. A separate oracle enumerates effective divisors of each degree and groups them using the inverse of a reduced Laplacian, computed with rational arithmetic. For fixed degree, two divisors have the same inverse-Laplacian coordinates modulo $\mathbb{Z}^{n-1}$ if and only if they are equivalent: solving the reduced system gives an integral firing function precisely in that case, and the omitted coordinate follows from the zero sum. An equivalence class has positive rank if and only if the union of supports of its effective representatives is the whole vertex set. This oracle does not use the dynamic recurrence.

The tests cover every rooted unlabelled tree shape through five vertices with each edge multiplicity in $\{1, 2, 3, 4\}$, together with 60 seeded trees on six to eight vertices. There are 2,657 test instances and 445,497 enumerated effective divisors. The quadratic algorithm agrees with both the lattice oracle and a direct cubic implementation. Rerooting all reconstructed witnesses gives 65,005 positive-root checks. Additional tests cover signed normalization, known star formulas, heavy-edge splitting, binary multiplicities as large as 10^{1000} , invalid inputs, and larger tree shapes up to 600 vertices. The normal and optimized Python runs agree on all mathematical outputs. These are finite regression checks, not a substitute for the all-size proof or a proof-assistant verification.

The proof and computational checks were developed with AI assistance. The author is responsible for the claims and source attribution. This manuscript is unrefereed and makes no claim of independent review or absolute priority.

References

- [1] M. Baker and S. Norine, Riemann–Roch and Abel–Jacobi theory on a finite graph, *Advances in Mathematics* **215** (2007), 766–788. arXiv:math/0608360.
- [2] M. Beougher, N. Cibu, K. Ding, S. DiSilvio, K. Heyse, S. Kononova, C. Lee, R. Morrison and K. Singal, Chip-Firing Games on Banana Trees, arXiv:2510.03499v1 (2025).