

Monotone Bounded Depth Formula Complexity of Graph Homomorphism Polynomials

Balagopal Komarath 

Indian Institute of Technology Gandhinagar, India

Rohit Narayanan 

Indian Institute of Technology Gandhinagar, India

Abstract

We introduce *baggy elimination trees*, a novel graph decomposition that generalises the classical elimination trees underlying treedepth, and use them to give a complete characterisation of the monotone bounded-depth formula complexity of graph homomorphism and coloured isomorphism polynomials. Specifically, we prove that the Δ -product depth monotone formula complexity of these polynomials is $\Theta(n^{\lambda_\Delta(H)})$, where $\lambda_\Delta(H)$ is the minimum cost of a baggy elimination tree for H at BET-depth Δ .

This result closes the last open case in the programme initiated by Komarath, Pandey and Rahul [8] and continued by Bhargav, Chen, Curticapean and Dwivedi [2]: tight size characterisations of monotone circuit complexity (via treewidth / bounded-depth treewidth), monotone ABP complexity (via pathwidth / bounded-depth pathwidth), and monotone formula complexity (via treedepth) were already known; our theorem supplies the missing bounded-depth formula characterisation via the new notion of bounded-depth baggy-elimination-tree cost λ_Δ , completing the picture for all three models in algebraic complexity and their fixed depth variants.

As applications, for constant-degree polynomial families we derive an almost-optimal separation between monotone circuits and monotone formulas at every fixed product depth: there exists a family computable by $O(N)$ -size monotone circuits of product depth Δ that requires $\Omega(N^{\Delta/2})$ -size monotone formulas of the same depth (and this exponent is optimal up to a constant factor). We also prove a strict depth hierarchy: for every $\Delta \geq 1$ and every constant $k \geq 2$, there is a constant-degree family with $O(s(N))$ -size monotone formulas of product depth Δ that requires $\Omega(s(N)^k)$ -size monotone formulas of product depth $\Delta - 1$.

2012 ACM Subject Classification Theory of computation $\rightarrow$ Circuit complexity; Theory of computation $\rightarrow$ Algebraic complexity theory

Keywords and phrases Monotone complexity, bounded depth, formula complexity, graph homomorphism, algebraic complexity

Digital Object Identifier 10.4230/LIPIcs.MFCS.2026.58

Related Version *arXiv submission*: <https://arxiv.org/abs/2511.03388v2>

Funding *Balagopal Komarath*: Supported by ANRF grant ANRF/ARGM/2025/001725/MTR.

1 Introduction

The study of graph homomorphism polynomials has emerged as a surprisingly powerful lens for understanding fundamental questions in both algorithm design and algebraic complexity. As an algorithmic tool, efficient constructions for graph homomorphisms yield optimal algorithms for various pattern counting and detection problems [4]. In algebraic complexity, these same polynomials provide a unified framework for defining natural families that are complete for major complexity classes like VP and VNP [5].

Komarath, Pandey, and Rahul [8] showed that the *monotone* complexity, where subtractions are disallowed in computation, of graph homomorphism polynomials is characterized by various structural parameters of the pattern graph H . Specifically, in the unbounded-depth



© Balagopal Komarath and Rohit Narayanan;

licensed under Creative Commons License CC-BY 4.0

51st International Symposium on Mathematical Foundations of Computer Science (MFCS 2026).

Editors: Michal Koucký and Daniela Petrişan; Article No. 58; pp. 58:1–58:13

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

setting, the monotone circuit complexity is characterized by H 's treewidth, the monotone ABP complexity by pathwidth, and the monotone formula complexity treedepth. Therefore, we can use known separations between these graph parameters to separate corresponding algebraic computational models.

The above characterizations naturally raise the question: how does this correspondence translate to *bounded-depth* computation? Recently, Bhargav, Chen, Curticapean and Dwivedi [2] provided a characterization for monotone bounded-depth circuits, linking their size to a depth-restricted variant of treewidth, and for monotone bounded-depth ABPs, linking their size to a depth-restricted variant of pathwidth. However, a significant model remained uncharacterized: monotone bounded-depth formulas. This model is computationally weaker than circuits but fundamental in algebraic complexity theory. What graph-theoretic parameter, if any, governs the trade-off between depth and size for a formula computing $\text{Hom}_{H,n}$?

In this paper, we resolve this question and thereby close the last open case in the characterization programme of [8, 2]. We introduce a graph decomposition, which we call the *Baggy elimination tree*, that are generalizations of elimination trees used to define treedepth. We define two measures for a baggy elimination tree: *BET-depth or the Baggy Elimination Tree depth* (Δ), and *cost* at BET-depth Δ (λ_Δ). We define $\lambda_\Delta(H)$ as the minimum possible cost that can be achieved for a given BET-depth Δ . Our main theorem proves that this graph-theoretic parameter characterizes bounded product depth monotone formula complexity for graph homomorphism polynomials.

In the Boolean setting, monotone circuits exhibit exponential gaps in depth compared to non-monotone circuits [12]; superpolynomial separations between monotone circuits and formulas are also well-established [6], and a strict depth hierarchy is known for constant-depth monotone formulas [1, 7, 11]. In the non-monotone setting, depth hierarchies have been established for *homogeneous* arithmetic circuits [9], which is relevant in the context of homomorphism polynomials studied in this paper. Monotone formula lower bounds for computing the iterated matrix multiplication polynomial $\text{IMM}_{n,d}$ (the coloured homomorphism polynomial of a path) were first proved by Shamir and Snir [13].

For general (non-monotone) arithmetic circuits, superpolynomial lower bounds against low-depth circuits have also been established [10]. In the monotone setting, robust separations for the monotone depth hierarchy were recently established [3]. In contrast to previous existence results relying on polynomials specifically constructed only to achieve the separation, we achieve separation via a precise structural characterization for a natural family of polynomials: we show that the separation between depth Δ and $\Delta + 1$ is strictly determined by the *baggy elimination tree* cost of the underlying graph patterns.

The paper is organized as follows. We introduce definitions for the model and the polynomials in Section 2. We define baggy elimination trees, the parameter that characterizes bounded product depth monotone formula complexity, in Section 3. We prove the characterization in Section 4. This result provides a precise description of the complexity of a well-studied model for a large class of important polynomials in terms of a structural graph parameter, and constitutes the main contribution of our work. In fact, for any fixed graph H , our upper and lower bounds only differ by constant factors. For constant-degree polynomial families, we prove an almost optimal separation between monotone circuits and monotone formulas in Section 5 and a depth hierarchy theorem for monotone formulas in Section 6. Although better lower bounds and separations are known for monotone computation (see [3, 7]), our main contribution here is to show that these separations can be derived from easily provable separations between graph parameters, continuing the line of work in [2] and [8].

2 Preliminaries

► **Definition 1.** A polynomial over $\mathbb{Q}$ is called *monotone* if all its coefficients are non-negative. An arithmetic formula computing a polynomial in $\mathbb{Q}[x_1, \dots, x_n]$ is either a variable, a field constant, or $F_1 + \dots + F_k$, or $F_1 \cdots F_k$ where F_i for $i \in [k]$ are arithmetic formulas computing polynomials in $\mathbb{Q}[x_1, \dots, x_n]$. The formula is called *monotone* if all constants in it are non-negative.

An arithmetic formula can be naturally represented as a rooted tree where the internal nodes (called *gates*) are labelled $+$ or $\times$ and the leaves (called *input gates*) are labelled by variables or constant. The size of a formula is then the number of edges in the tree. The product depth of the formula is the maximum number of gates labelled $\times$ over a path from the root to an input gate.

The families of polynomials that we look at in this paper enumerate graph homomorphisms or colored isomorphisms. The following definitions are from [8]:

► **Definition 2.** For graphs H and G , a homomorphism from H to G is a function $\phi : V(H) \mapsto V(G)$ such that $\{i, j\} \in E(H)$ implies $\{\phi(i), \phi(j)\} \in E(G)$. For an edge $e = \{i, j\}$ in H , we use $\phi(e)$ to denote $\{\phi(i), \phi(j)\}$.

► **Definition 3.** Let H be a k -vertex graph where its vertices are labeled by $[k]$ and let G be a graph where each vertex has a color in $[k]$. Then, a *colored isomorphism* of H in G is a subgraph of G isomorphic to H such that all vertices in the subgraph have different colors and for each edge $\{i, j\}$ in H , there is an edge in the subgraph between vertices colored i and j .

► **Definition 4.** For a pattern graph H on k vertices, the n -th homomorphism polynomial for H is a polynomial on $\binom{n}{2}$ variables x_e where $e = \{u, v\}$ for $u, v \in [n]$.

$$\text{Hom}_{H,n} = \sum_{\phi} \prod_{e \in E(H)} x_{\phi(e)}$$

where ϕ ranges over all homomorphisms from H to K_n .

Computing the homomorphism polynomial is an important intermediate step in many algorithms related to finding and counting graph patterns. Instead of the homomorphism polynomial, we consider an equivalent polynomial (See Lemma 8 in [8]) called the *colored isomorphism polynomial* which enumerates all colored isomorphisms from a pattern to a host graph where there are n vertices of each color.

► **Definition 5.** For a pattern graph H on k vertices, the n -th colored isomorphism polynomial for H is a polynomial on $|E(H)|n^2$ variables x_e where $e = \{(i, u), (j, v)\}$ for $u, v \in [n]$ and $\{i, j\} \in E(H)$.

$$\text{Collso}_{H,n} = \sum_{u_1, \dots, u_k \in [n]} \prod_{\{i, j\} \in E(H)} x_{\{(i, u_i), (j, u_j)\}}$$

We notice that the labeling of H does not affect the complexity of $\text{Collso}_{H,n}$. Given the polynomial $\text{Collso}_{H,n}$ for some labeling of H and if ψ is a relabeling of H , then the polynomial $\text{Collso}_{H,n}$ for the new labeling can be obtained by the substitution $x_{\{(i, u), (j, v)\}} \mapsto x_{\{(\psi(i), u), (\psi(j), v)\}}$.

Monomials of the polynomial are computed using parse trees in formulas:

► **Definition 6.** Let g be a gate in a formula F . A parse tree rooted at g is any rooted tree which can be obtained by the following procedure:

1. The gate g is the root of the parse tree.
2. If there is a multiplication gate g in the tree, include all its children in the formula as its children in the parse tree.
3. If there is an addition gate g in the tree, pick an arbitrary child of g in the formula and include it in the parse tree.

Any gate can occur at most once in any parse tree in F as F is a tree. Given a parse tree T that contains a gate g , we use T_g to denote the subtree of T rooted at g . Note that we can replace T_g in T with any parse tree rooted at g to obtain another parse tree. Similarly, if we have two parse trees T and T' that both contain the same multiplication gate g from the formula, then we can replace any subtree of T_g with the corresponding subtree of T'_g to obtain another parse tree. This is because all children of g in both parse trees are the same and therefore we can apply the aforementioned replacement. We call the tree obtained by removing T_g from T as the *tree outside T_g (or g) in T* .

Let $H = (V, E)$ be a graph. A vertex $v \in V(H)$ is called a *pendant vertex* if its degree is one. Throughout this paper, we assume that that pattern graph H has more than one edge and is connected.

3 Baggy Elimination Trees

In this section, we introduce the most important definition in this paper and work through some examples to understand the intuition behind this definition.

► **Definition 7 (Baggy Elimination Tree).** For a graph H , a baggy elimination tree T is a rooted tree where each node in $V(T)$ is labeled with a non-empty “bag” of vertices of H such that every vertex of H appears in exactly one bag, and the tree satisfies: if $\{u, v\} \in E(H)$, then u and v are either in the same bag or in bags that are in an ancestor-descendant relationship in T . A leaf node t_m in T is a *core leaf* if it contains some vertex that is not pendant in H . Otherwise, we call the leaf *non-core*.

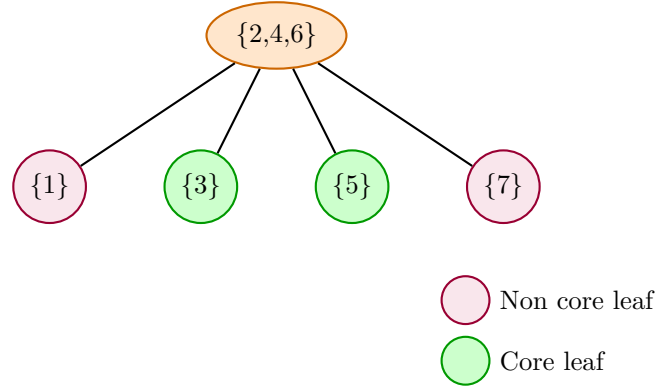
The BET-depth or the Baggy Elimination Tree depth of T is the maximum number of nodes on any root-to-leaf path $P = (t_1, \dots, t_m)$, excluding the leaf node t_m if t_m is a non-core leaf. The cost of a single path P is the sum of the cardinalities of the bags of all nodes on that path. The cost of the tree T is the maximum cost over all root-to-leaf paths in T . The Δ -BET-depth baggy elimination tree cost, denoted by $\lambda_\Delta(H)$, is the cost of the minimum cost baggy elimination tree of BET-depth at most Δ .

To motivate the above definition, we consider monotone formulas for $\text{Collso}_{P_7, n}$, where P_7 is the path on 7 vertices. This polynomial has n^7 monomials. So it has an n^7 -size monotone formula of product depth one. Komarath, Pandey, and Rahul [8] describe how to construct $O(n^3)$ -size monotone formulas for this polynomial. Their construction has a product depth of three.

► **Example 8.** Consider the following formula for $\text{Collso}_{P_7, n}$:

$$\sum_{i_2, i_4, i_6 \in [n]} \left(\sum_{i_1 \in [n]} x_{\{(1, i_1), (2, i_2)\}} \sum_{i_3 \in [n]} x_{\{(2, i_2), (3, i_3)\}} x_{\{(3, i_3), (4, i_4)\}} \sum_{i_5 \in [n]} x_{\{(4, i_4), (5, i_5)\}} x_{\{(5, i_5), (6, i_6)\}} \sum_{i_7 \in [n]} x_{\{(6, i_6), (7, i_7)\}} \right)$$

This formula has size $O(n^4)$ and has product depth two. It corresponds to the baggy elimination tree of BET-depth two for P_7 shown in Figure 1. The size of the formula is determined by the cost of the tree and the nesting of product gates increasing the product



■ **Figure 1** Baggy elimination tree of BET-depth two for P_7 .

depth to two is contributed only by the core leaves in the baggy elimination tree. This is the reason for distinguishing between core and non-core leaves in the definition of baggy elimination trees.

Baggy elimination trees are a simple generalization of elimination trees used to define treedepth. In an elimination tree, each bag has to contain exactly one vertex. Observe that any baggy elimination tree of cost c can be converted into an elimination tree of cost c by replacing bags with more than one vertex with a path containing those vertices. Therefore, for any Δ , we have treedepth of H is at most the $\lambda_\Delta(H)$ for all H .

► **Remark 9.** The graph that is just an edge has a formula of product depth zero that has linear size and this is optimal. Disconnected graphs can be characterized by defining *baggy elimination forests* instead of trees and by defining the product depth of such forests as one plus the max of product depths of trees in the forest if there is more than one tree in the forest. In the interest of simplicity, we omit this generalization from this paper.

4 Characterizing Bounded Product Depth Monotone Formulas

Our main theorem is as follows:

► **Theorem 10.** *For any connected graph H on more than two vertices, the polynomial family Collso_H has Δ -product depth monotone formula complexity of $\Theta(n^{\lambda_\Delta(H)})$.*

We prove the upper bound first.

Proof. Let H be the pattern graph. For any $\Delta \geq 1$, we construct a monotone formula $\mathcal{F}$ for $\text{Collso}_{H,n}$ with product depth Δ and size $O(n^{\lambda_\Delta(H)})$. Let T be a baggy elimination tree T for H with BET-depth Δ and cost $k = \lambda_\Delta(H)$.

We build a recursive formula $\mathcal{F}$ whose structure mirrors the structure of the tree T . For any node t in T , we use X_t to denote all vertices in H in the bag t and let $A(t)$ denote the set of t 's proper ancestors. Let $X_{A(t)} = \bigcup_{a \in A(t)} X_a$ be the set of all vertices in its ancestors. Let $\phi_{A(t)}$ be an assignment $\phi_{A(t)} : X_{A(t)} \rightarrow [n]$. We define a formula $\mathcal{F}(t \mid \phi_{A(t)})$ that computes the polynomial for the sub-problem induced by t and its descendants, given the fixed assignment $\phi_{A(t)}$ to all its ancestors.

First, we define short names for two classes of monomials for convenience. Let $\text{EM}(\phi_t, X_t)$ (Edge Monomial) be the product of variables for edges *within* the bag X_t :

$$\text{EM}(\phi_t, X_t) = \prod_{\substack{\{u,v\} \in E(H) \\ u,v \in X_t}} x_{\{\phi_t(u), \phi_t(v)\}}$$

EM is 1 if X_t is an independent set. Now, let $\text{ALM}(\phi_{A(t)}, \phi_t)$ (Ancestor Link Monomial) be the product of variables for all edges *between* the current bag and *an ancestor bag*:

$$\text{ALM}(\phi_{A(t)}, \phi_t) = \prod_{\substack{\{u,v\} \in E(H) \\ u \in X_{A(t)}, v \in X_t}} x_{\{\phi_{A(t)}(u), \phi_t(v)\}}$$

If no vertex in $X - t$ is adjacent to a vertex in an ancestor bag of t , then ALM is 1. The required formula is $\mathcal{F}(r \mid \emptyset)$, where r is the root node (it has no ancestors). If there are no edges contributing to the product defining EM or ALM , the corresponding monomial is equal to 1. We construct this formula inductively starting from the leaves of t . If t is a leaf node in T with ancestors $A(t)$, the formula is:

$$\mathcal{F}(t \mid \phi_{A(t)}) = \sum_{\phi: X_t \rightarrow [n]} (\text{EM}(\phi, X_t) \cdot \text{ALM}(\phi_{A(t)}, \phi))$$

If t is an internal node with children $u_1, \dots, u_m$, its formula is:

$$\mathcal{F}(t \mid \phi_{A(t)}) = \sum_{\phi_t: X_t \rightarrow [n]} \left(\text{EM}(\phi_t, X_t) \cdot \text{ALM}(\phi_{A(t)}, \phi_t) \cdot \prod_{i=1}^m \mathcal{F}(u_i \mid \phi_{A(t)} \cup \phi_t) \right)$$

It is easy to see that the formula is correct using an induction. We now prove that it has product depth Δ and size $O(n^k)$. Observe that each node in the tree along any root to leaf path contributes at most one to the product depth. It now suffices to show that non-core leaves of T do not add to the product depth of the formula.

Consider a non-core leaf t in T . We claim that without loss of generality we may assume t contains exactly one pendant vertex. Suppose t contains two pendant vertices u and v . Since H is connected and has more than one edge, there cannot be an edge between u and v . Therefore, u and v are each adjacent only to some vertex in a proper ancestor bag. We may replace t in T with two non-core leaves t_1 and t_2 containing u and v respectively, without increasing the cost or the BET-depth of T (non-core leaves do not contribute to product depth by definition). Repeating this argument, we may assume without loss of generality that $t = \{u\}$ for some pendant u in H . Therefore, in the formula $\mathcal{F}(t \mid \phi_{A(t)})$, we have $\text{EM}(\phi_t, X_t) = 1$ and that $\text{ALM}(\phi_{A(t)}, \phi_t)$ is a single variable. So there are no multiplication gates in this formula. By definition, the longest core path in T has length Δ . Therefore, the constructed formula $\mathcal{F}$ has a product depth of at most Δ .

We now prove the size upper-bound. Observe that the fan-in of each $+$ gate at the top-level in $\mathcal{F}(t \mid \cdot)$ is n^k where $k = |X_t|$. The fan-ins of all $\times$ gates are constant (independent¹ of n). Therefore, the total size of the formula is $O(n^{|X_{t_1}| + \dots + |X_{t_k}| = \lambda_{\Delta}(H)})$ where $t_1, \dots, t_k$ is the maximum cost path in T . ◀

Now, we prove the lower bound.

¹ These constants depend on T and therefore H . But since we regard H as a fixed pattern graph, we can absorb this cost into the $O(\cdot)$ notation.

Proof. The proof is similar to the lower bound proof of Theorem 3 in [8]. Given a parse tree computing a monomial of $\text{Collso}_{H,n}$, we construct a baggy elimination tree for H from the parse tree. We then show that only a few monomials can be computed using a gate that corresponds to the leaf bag in the baggy elimination tree that achieves the maximum cost. This implies a lower bound on the total number of gates.

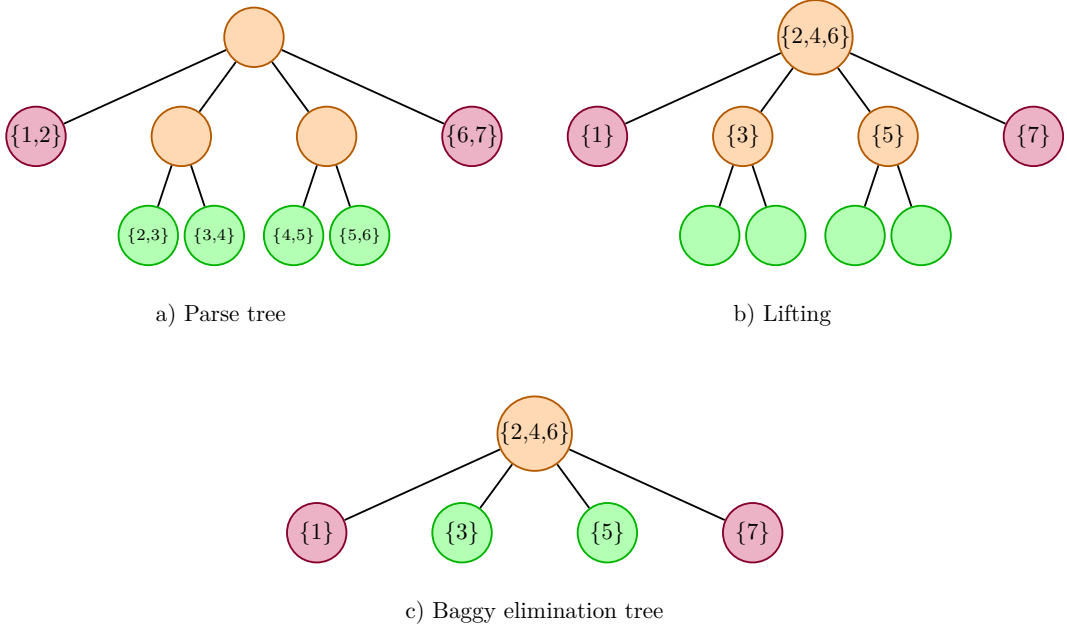
Let m be a monomial in $\text{Collso}_{H,n}$ and let T be a parse tree of $\mathcal{F}$ computing m after removing all $+$ gates from the parse tree and attaching the child of $+$ gate to the $+$ gate's parent. We construct a baggy elimination tree B from T as follows: For each vertex $i \in V(H)$, we find all leaves in T corresponding to variables that involve i (e.g., $x_{\{(i,f(i)),(j,f(j))\}}$ for all neighbors j of i in H). We put i into the bag that is the Least Common Ancestor (LCA) of these leaves in T , denoted by $t_i = \text{LCA}(i)$. After this process, if a non-root node is empty, we remove it from B , attaching any children to the removed node's parent. We will see later that the root bag is either non-empty or has exactly one child after this process. If the root bag is empty and has one child, we can make the child the root. The nodes of our baggy tree B then correspond to gates $g \in T$ that are an LCA for at least one vertex. We denote the bag for a node g as $B(g) = \{i \in V(H) \mid \text{LCA}(i) = g\}$. The tree structure of B is thus inherited from T .

We claim that B is a valid baggy elimination tree of BET-depth Δ for H . For any edge $\{i, j\} \in E(H)$, in the initial tree (Refer Figure 2, Part (a)), the corresponding leaf in T is a descendant of both t_i and t_j , implying t_i and t_j must also be in an ancestor-descendant relationship. We will now prove our earlier claim about the root bag. Suppose the root bag is empty and has more than one child. Let u and v be two vertices of H in two distinct children of the root. Observe that all vertices adjacent to u or v in H are in the respective subtrees. So there is not path between u and v in H , which contradicts the assumption that H is connected.

We claim the BET-depth of T is at most Δ . A non-pendant vertex i of H appears in multiple leaves in T , forcing its t_i to be the LCA of distinct leaves in B , which must correspond to a multiplication gate. Therefore, any leaf in B that corresponds to an input gate can only contain pendant vertices in H . So the BET-depth of B is bounded by the product depth of $\mathcal{F}$.

Consider a monomial m and a parse tree T computing it in $\mathcal{F}$. After the construction of the baggy elimination tree from T as detailed above, let g be the gate in T that corresponds to the leaf in T that maximizes the cost. Let $d = \lambda_\Delta(H)$. Assume without loss of generality that $1, \dots, d$ are the vertices of H appearing in g and its ancestors in the baggy elimination tree. The monomial m fixes an assignment for all vertices in H to $[n]^d$. Let $\phi_m(i) = u_i$ for $i \in [d]$ in this assignment. We claim that for any monomial m' for which g appears in a parse tree computing m' , we must have $\phi_{m'}(i) = u_i$ for $i \in [d]$ as well. Suppose for contradiction that $\phi_{m'}(i) = v_i \neq u_i$ for some $i \in [d]$. Let T' be the parse tree for m' . Let g' be the gate in T such that i is in the bag corresponding to g' . Since there is a unique path from g to g' in $\mathcal{F}$, the gate g' must appear in T' as well. We say that a vertex i in H is contained in a subtree T of some parse tree if there is some input gate labelled by $x_{\{(i, \cdot), (j, \cdot)\}}$ in the subtree T . It is worth noting that the ordering is irrelevant since the variables are indexed by unordered pairs; only the incidence of the edge to i is used in the argument. We now split the proof into cases:

- If g' is an input gate, then it is labelled $x_{\{(i, w_i), (j, w_j)\}}$ for some $w_i, w_j \in [n]$ and $j \in V(H)$. Since g' is in the parse tree for both m and m' , it must be that $w_i = u_i$ and $w_i = v_i$, a contradiction.



■ **Figure 2** Parse tree to baggy elimination tree for P_7 .

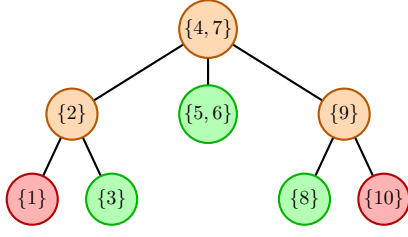
- The gate g' is a multiplication gate. The vertex i occurs in at least two subtrees of g' as we lifted i to g' . We split into two cases:
 - If $T'_{g'}$ contains i , then we use one such subtree to replace the corresponding subtree in T . The resulting parse tree computes a monomial that contains variables indexed by (i, u_i) and (i, v_i) , a contradiction.
 - If $T'_{g'}$ does not contain i , then i must occur in the tree outside $T'_{g'}$ in T' . We replace $T'_{g'}$ with $T_{g'}$ in T' . The resulting parse tree again computes a monomial that contains variables indexed by (i, u_i) and (i, v_i) , a contradiction.

Now, we prove the lower bound. The polynomial $\text{Collso}_{H,n}$ has n^k monomials where $k = |V(H)|$. Any monomial can be mapped to a gate as above such that a gate in the image has at most $n^{k-\lambda_\Delta(H)}$ pre-images. Therefore, there must be at least $n^{\lambda_\Delta(H)}$ such gates. ◀

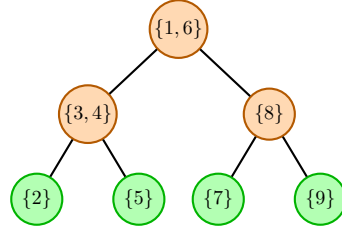
5 Separating Circuits from Formulas in Bounded Depth

The full b -ary tree of depth Δ denoted $F_{b,\Delta}$ is defined as follows: $F_{b,1}$ is the single node tree. $F_{b,\Delta+1}$ is a root node with b subtrees each isomorphic to $F_{b,\Delta}$. We note that the treedepth of $F_{b,\Delta}$ is Δ for $b \geq 2$. It is at most Δ because the tree itself is an elimination tree. For $\Delta = 1$, the lower bound is true. For $\Delta > 1$, since $b \geq 2$, at least one $F_{b,\Delta-1}$ subtree is untouched in the root. So the depth of any elimination tree is at least $1 + (\Delta - 1) = \Delta$.

Bhargav et. al. [2] showed that the Δ -product depth monotone circuit complexity of $\text{Collso}_{F_{b,\Delta+1}}$ is $\Theta(n^2)$. Theorem 10 and the above lower bound on treedepth shows that the Δ -product depth monotone formula complexity of this polynomial family is $\Theta(n^\Delta)$. Therefore, we get the following theorem separating the power of circuits and formulas of bounded product depth.



■ **Figure 3** A cost-4 BET-depth-3 baggy elimination tree for P_{10} .



■ **Figure 4** A cost-5 BET-depth-3 baggy elimination tree for C_9 .

► **Theorem 11.** *For any $\Delta \geq 1$, there is a constant-degree (does not depend on N) polynomial family that is computable by $O(N)$ -size monotone circuits of product depth Δ but requires $\Omega(N^{\Delta/2})$ monotone formulas of product depth Δ , where N is the number of variables in the formula.*

► **Remark 12.** Observe that a product depth Δ monotone circuit of size s can be converted into a product depth Δ monotone formula of size $O(s^{2\Delta})$ by simply duplicating gates as needed. Therefore, this separation is optimal up to constant factors independent of Δ in the exponent of N .

► **Remark 13.** Consider $\text{CollIso}_{P_{10},n}$ and $\text{CollIso}_{C_9,n}$, where P_{10} is the path on 10 vertices and C_9 is the cycle on 9 vertices. Both polynomials have degree 9. By definition of pruned- Δ -treewidth ($ptw_\Delta(H)$) for a graph H [2], we have $ptw_3(P_{10})$ and $ptw_3(C_9)$ to be 2, and both families admit $O(n^3)$ -size monotone circuits at product depth 3.

On the other hand, Figure 3 shows a baggy elimination tree of BET-depth 3 for P_{10} with $\lambda_3(P_{10}) = 4$. By Theorem 10, $\text{CollIso}_{P_{10},n}$ monotone circuits of size $\Omega(n^4)$ at product depth 3.

We claim $\lambda_3(C_9) \geq 5$. Let T be any baggy elimination tree for C_9 and let r denote its root bag. Note that since there are no pendant vertices in C_9 , no T will have a non-core leaf. We now case-split on $|r|$.

- $|r| \geq 3$: Removing the vertices of r from C_9 leaves at least one edge component, whose contribution to the cost of T is 2 (since any path on at least one edge has treedepth ≥ 2). Hence the maximum-cost root-to-leaf path in T has cost at least $3 + 2 = 5$.
 - $|r| = 2$: The optimal choice of two vertices (see Figure 4) leaves at least a P_4 , and $\lambda_2(P_4) \geq 3$. Now the maximum-cost is at least $2 + 3 = 5$.
 - $|r| = 1$: Removal of one vertex leaves a P_8 with $\lambda_2(P_8) \geq 4$, giving total cost $\geq 1 + 4 = 5$.
- In every case $\lambda_3(C_9) \geq 5$, so by Theorem 10 the monotone formula complexity of $\text{CollIso}_{C_9,n}$ at product depth 3 is $\Omega(n^5)$.

Thus, $\text{CollIso}_{C_9,n}$, a degree-9 polynomial family admitting $O(n^3)$ -size monotone circuits, requires $\Omega(n^5)$ -size monotone formulas of product depth 3, strictly improving on the $O(n^3)$ vs. $\Omega(n^4)$ separation obtained from P_{10} at the same degree and depth. Since our primary objective and focus is characterization of model rather than separation, maximize separation, we present these examples to show the potential of the framework and leave an exhaustive search for optimal separations to future work.

6 Product Depth Hierarchy for Monotone Formulas

We show that $F_{b,\Delta+2}$ has high cost for BET-depth Δ baggy elimination trees.

► **Theorem 14.** *For any $b > \Delta \geq 1$, we have $\lambda_\Delta(F_{b,\Delta+2}) \geq b + \Delta$.*

Proof. We prove by induction on Δ . For $\Delta = 1$, note that the root bag must contain all non-leaf vertices of $F_{b,3}$ and there are $b + 1$ such vertices.

For $\Delta > 1$, we split the proof into two cases:

- If the root bag does not contain any vertex from at least one $F_{b,\Delta+1}$ subtree, then by the induction hypothesis, the $\Delta - 1$ BET-depth baggy elimination tree for this subtree contributes at least $b + \Delta - 1$ to the cost. The root bag contains at least one vertex. So the total cost is $b + \Delta$.
- Otherwise, the root bag contains at least b vertices. We claim that there are no vertices in the root bag from at least one of the b^2 subtrees isomorphic to $F_{b,\Delta}$. Otherwise, root bag itself will contain $b^2 \geq 2b \geq b + \Delta$ vertices. But the subtree $F_{b,\Delta}$ has cost at least Δ for any product depth. Therefore, the total cost is at least $b + \Delta$. ◀

We can now prove a depth hierarchy theorem for monotone formulas.

► **Theorem 15.** *For any $\Delta \geq 1$, for any constant $k \geq 2$, there is a constant-degree (does not depend on N) polynomial family that has $O(s(N))$ -size monotone formulas of product depth Δ but requires $\Omega(s(N)^k)$ -size monotone formulas of product depth $\Delta - 1$, where N is the number of variables in the formula.*

Proof. The family is $\text{Collso}_{F_{b,\Delta+1}}$ with $b = (k - 1)\Delta + 1$. The corresponding colored isomorphism polynomial has $O(n^\Delta)$ size monotone formulas of product depth Δ but need $\Omega(n^{b+\Delta})$ for product depth $\Delta - 1$. ◀

► **Remark 16.** Observe that any polynomial family where degree is bounded by constant d has product depth one monotone formulas of size $O(n^d)$. Therefore, polynomial separations as above are the best one could hope for. However, it is possible that the above separations can be achieved using polynomials with lower (constant) degree.

We note that the degree of $\text{Collso}_{F_{b,\Delta+1}}$ equals $|E(F_{b,\Delta+1})|$, which grows with b and Δ . Since $b = (k - 1)\Delta + 1$ for given constants k and Δ , this degree is a constant with respect to the input size N (it does not grow with N), but it does depend on k and Δ . Concretely, the degree is $b \cdot \frac{b^{\Delta+1}-1}{b-1} \leq b^{\Delta+2}$, which is a constant for fixed k and Δ . Thus, for every fixed pair (k, Δ) , Theorem 15 exhibits a polynomial family of constant degree (with respect to N) achieving the stated separation.

References

- 1 Miklos Ajtai and Yuri Gurevich. Monotone versus positive. *Journal of the ACM*, 34(4):1004–1015, 1987. doi:10.1145/31846.31852.
- 2 C. S. Bhargav, Shiteng Chen, Radu Curticapean, and Prateek Dwivedi. Monotone bounded-depth complexity of homomorphism polynomials. *CoRR*, abs/2505.22894, 2025. doi:10.48550/arXiv.2505.22894.
- 3 Arkadev Chattopadhyay, Utsab Ghosal, and Partha Mukhopadhyay. Robustly Separating the Arithmetic Monotone Hierarchy via Graph Inner-Product. In *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 250 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.FSTTCS.2022.12.
- 4 Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms are a good basis for counting small subgraphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 210–223. ACM, 2017. doi:10.1145/3055399.3055502.

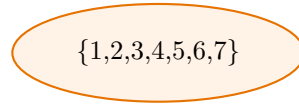
- 5 Arnaud Durand, Meena Mahajan, Guillaume Malod, Nicolas de Rugy-Altherre, and Nitin Saurabh. Homomorphism Polynomials Complete for VP. In Venkatesh Raman and S. P. Suresh, editors, *34th International Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS 2014)*, volume 29 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 493–504, Dagstuhl, Germany, 2014. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.FSTTCS.2014.493.
- 6 Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM Journal on Computing*, 19(1):1–11, 1990.
- 7 Maria Klawe, Wolfgang J. Paul, Nicholas Pippenger, and Mihalis Yannakakis. On monotone formulae with restricted depth. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing (STOC)*, pages 480–487. ACM, 1984. doi:10.1145/800057.808717.
- 8 Balagopal Komarath, Anurag Pandey, and C. S. Rahul. Monotone arithmetic complexity of graph homomorphism polynomials. *Algorithmica*, 85(9):2554–2579, 2023. doi:10.1007/s00453-023-01108-0.
- 9 Mrinal Kumar and Shubhangi Saraf. On the power of homogeneous depth 4 arithmetic circuits. *SIAM Journal on Computing*, 46(1):336–387, 2017. doi:10.1137/140999335.
- 10 Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. Superpolynomial lower bounds against low-depth algebraic circuits. In *62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 804–814. IEEE Computer Society, 2021. doi:10.1109/FOCS52979.2021.00083.
- 11 Elizaveta A. Okol’nishnikova. On the complexity of monotone formulas for some class of Boolean functions. *Metody Diskretnogo Analiza*, 38:66–80, 1982.
- 12 Ran Raz and Avi Wigderson. Monotone circuits for matching require linear depth. *J. ACM*, 39(3):736–744, July 1992. doi:10.1145/146637.146684.
- 13 Eli Shamir and Marc Snir. On the depth complexity of formulas. *Mathematical Systems Theory*, 13(1):301–322, 1980. doi:10.1007/BF01744302.

A Examples: Formula Construction for $H = P_7$

Let H be the path graph $v_1 - v_2 - v_3 - v_4 - v_5 - v_6 - v_7$. The pendant vertices are $\{v_1\}$ and $\{v_7\}$. We use i_j to denote the assignment $\phi(v_j)$. We now construct a formula from a baggy elimination tree (Refer Section 4).

A.1 Example 1: Baggy elimination tree with BET-depth = 1

Consider the trivial tree T_1 from Figure 5 with BET-depth = 1 and $\lambda = 7$. The tree T_1 has a single root node R , whose bag is $X_R = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$. There are no leaves apart from the root itself.



■ **Figure 5** Baggy elimination tree for P_7 with BET-depth 1 and $\lambda = 7$.

The corresponding formula is $\mathcal{F} = \mathcal{F}(R \mid \emptyset)$. Since the ancestor set $A(R)$ is empty, we apply the Base Case rule to the root node R . The root R is a *core leaf* because its bag is not a single pendant vertex. Therefore,

$$\mathcal{F} = \mathcal{F}(R \mid \emptyset) = \sum_{\phi_R: X_R \rightarrow [n]} (\text{EM}(\phi_R, X_R) \cdot \text{ALM}(\emptyset, \phi_R)).$$

Here, $\text{ALM}(\emptyset, \phi_R)$ corresponds to the ancestor link monomial. Since there are no ancestors, it contributes a factor of 1. The term $\text{EM}(\phi_R, X_R)$ represents the edge monomial for the bag X_R , which contains all seven vertices of P_7 . Hence, the monomial is the product of variables corresponding to all six edges in the path:

$$x_{i_1, i_2} \cdot x_{i_2, i_3} \cdot x_{i_3, i_4} \cdot x_{i_4, i_5} \cdot x_{i_5, i_6} \cdot x_{i_6, i_7}.$$

Substituting this in gives the original $\Sigma_1 \rightarrow \Pi_1$ formula:

$$\mathcal{F} = \sum_{\underbrace{i_1, \dots, i_7 \in [n]}_{\Sigma_1}} \underbrace{(x_{i_1, i_2} \cdot x_{i_2, i_3} \cdot x_{i_3, i_4} \cdot x_{i_4, i_5} \cdot x_{i_5, i_6} \cdot x_{i_6, i_7})}_{\Pi_1}.$$

Thus, the formula has product depth 1. The total size of the formula is $O(n^\lambda) = O(n^7)$, which also agrees with the cost $\lambda = 7$.

A.2 Example 2: Baggy elimination tree with BET-depth = 2

We use the optimal tree T_2 from Figure 1 with BET-depth = 2 and $\lambda = 4$. The root R is $X_R = \{v_2, v_4, v_6\}$, and the leaves are $L_1 = \{v_1\}$, $L_3 = \{v_3\}$, $L_5 = \{v_5\}$, $L_7 = \{v_7\}$.

The formula is $\mathcal{F} = \mathcal{F}(R \mid \emptyset)$. Let ϕ_R be the assignment $\phi_R(v_j) = i_j$ for $j \in \{2, 4, 6\}$.

$$\mathcal{F}(R) = \sum_{\phi_R: X_R \rightarrow [n]} \left(\text{EM}(\phi_R, X_R) \cdot \text{ALM}(\emptyset, \phi_R) \cdot \prod_{j \in \{1, 3, 5, 7\}} \mathcal{F}(L_j \mid \phi_R) \right)$$

The term $\text{EM}(\phi_R, X_R)$ corresponds to the bag $X_R = \{v_2, v_4, v_6\}$, which is an independent set, so the monomial equals 1. Similarly, $\text{ALM}(\emptyset, \phi_R)$ has no ancestors, and its monomial also equals 1. This gives the $\Sigma_1 \rightarrow \Pi_1$ layers:

$$\mathcal{F} = \sum_{\underbrace{i_2, i_4, i_6 \in [n]}_{\Sigma_1}} \underbrace{(\mathcal{F}(L_1 \mid \phi_R) \cdot \mathcal{F}(L_3 \mid \phi_R) \cdot \mathcal{F}(L_5 \mid \phi_R) \cdot \mathcal{F}(L_7 \mid \phi_R))}_{\Pi_1}$$

We now apply the Base Case rule to each of the four leaf formulas. The ancestor set for all leaves is $A(L_j) = \{R\}$, so $\phi_{A(L_j)} = \phi_R$.

For $L_3 = \{v_3\}$ (Core Leaf),

$$\mathcal{F}(L_3 \mid \phi_R) = \sum_{\phi_3: X_3 \rightarrow [n]} (\text{EM}(\phi_3, X_3) \cdot \text{ALM}(\phi_R, \phi_3))$$

Here, EM for $X_3 = \{v_3\}$ is 1. The ALM term links $X_3 = \{v_3\}$ to $X_{A(L_3)} = X_R = \{v_2, v_4, v_6\}$. The edges are (v_2, v_3) and (v_3, v_4) , giving the monomial $x_{\phi_R(v_2), \phi_3(v_3)} \cdot x_{\phi_3(v_3), \phi_R(v_4)} = x_{i_2, i_3} \cdot x_{i_3, i_4}$. Thus,

$$\mathcal{F}(L_3 \mid \phi_R) = \sum_{\underbrace{i_3 \in [n]}_{\Sigma_2}} \underbrace{(x_{i_2, i_3} \cdot x_{i_3, i_4})}_{\Pi_2}.$$

For $L_1 = \{v_1\}$ (Pendant Leaf),

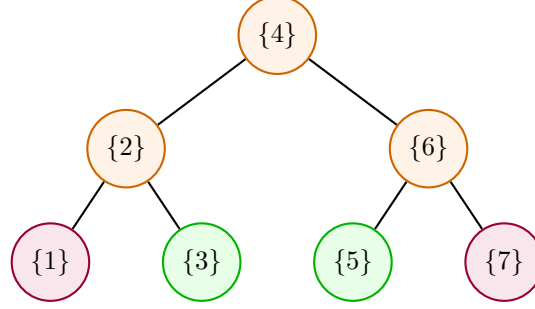
$$\mathcal{F}(L_1 \mid \phi_R) = \sum_{\phi_1: X_1 \rightarrow [n]} (\text{EM}(\phi_1, X_1) \cdot \text{ALM}(\phi_R, \phi_1))$$

The ALM term links $X_1 = \{v_1\}$ to $X_R = \{v_2, v_4, v_6\}$, and the only edge is (v_1, v_2) , giving monomial $x_{\phi_1(v_1), \phi_R(v_2)} = x_{i_1, i_2}$. Thus, $\mathcal{F}(L_1 \mid \phi_R) = \sum_{i_1 \in [n]} x_{i_1, i_2}$. Similarly, $\mathcal{F}(L_5) = \sum_{i_5} x_{i_4, i_5} x_{i_5, i_6}$ and $\mathcal{F}(L_7) = \sum_{i_7} x_{i_6, i_7}$.

The product depth is 2. The size is $O(n^\lambda) = O(n^4)$.

A.3 Example 3: Baggy elimination tree with BET-depth = 3

Consider the optimal tree T_3 from Figure 6 with BET-depth 3 and $\lambda = 3$. The root R is $X_R = \{v_4\}$. The internal nodes are $t_1 = \{v_2\}$ (child of R) and $t_2 = \{v_6\}$ (child of R). The leaves are $L_1 = \{v_1\}$, $L_3 = \{v_3\}$ (children of t_1) and $L_5 = \{v_5\}$, $L_7 = \{v_7\}$ (children of t_2).



■ **Figure 6** Baggy elimination tree for P_7 with BET-depth 3 and $\lambda = 3$.

The root formula is $\mathcal{F} = \mathcal{F}(R \mid \emptyset)$. Let $\phi_R(v_4) = i_4$.

$$\mathcal{F}(R) = \sum_{\phi_R: X_R \rightarrow [n]} (\text{EM}(\phi_R, X_R) \cdot 1 \cdot \mathcal{F}(t_1 \mid \phi_R) \cdot \mathcal{F}(t_2 \mid \phi_R))$$

$$\text{Thus, } \mathcal{F} = \sum_{\substack{i_4 \in [n] \\ \Sigma_1}} \underbrace{(\mathcal{F}(t_1 \mid \phi_R) \cdot \mathcal{F}(t_2 \mid \phi_R))}_{\Pi_1}.$$

For the internal node formula, consider $\mathcal{F}(t_1 \mid \phi_R)$. The ancestor set is $A(t_1) = \{R\}$, so $\phi_{A(t_1)} = \phi_R$. Let $\phi_1(v_2) = i_2$. Then,

$$\mathcal{F}(t_1 \mid \phi_R) = \sum_{\phi_1: X_1 \rightarrow [n]} (\text{EM}(\phi_1, X_1) \cdot \text{ALM}(\phi_R, \phi_1) \cdot \mathcal{F}(L_1 \mid \phi_R \cup \phi_1) \cdot \mathcal{F}(L_3 \mid \phi_R \cup \phi_1))$$

For $X_1 = \{v_2\}$, $\text{EM} = 1$, and ALM links $X_1 = \{v_2\}$ to $X_R = \{v_4\}$, but since there is no edge between them, the monomial is 1. Thus,

$$\mathcal{F}(t_1 \mid \phi_R) = \sum_{\substack{i_2 \in [n] \\ \Sigma_2}} \underbrace{(\mathcal{F}(L_1 \mid \phi_R \cup \phi_1) \cdot \mathcal{F}(L_3 \mid \phi_R \cup \phi_1))}_{\Pi_2}.$$

Now, consider the leaf formula for L_3 . The ancestor set is $A(L_3) = \{R, t_1\}$, and the ancestor assignment is $\phi_{A(L_3)}$ which maps $v_4 \rightarrow i_4$ and $v_2 \rightarrow i_2$.

$$\mathcal{F}(L_3 \mid \phi_{A(L_3)}) = \sum_{\phi_3: X_3 \rightarrow [n]} (\text{EM}(\phi_3, X_3) \cdot \text{ALM}(\phi_{A(L_3)}, \phi_3))$$

Here, $\text{EM} = 1$ for $X_3 = \{v_3\}$. The ALM term links $X_3 = \{v_3\}$ to $X_{A(L_3)} = \{v_2, v_4\}$. The edges are (v_2, v_3) and (v_3, v_4) , giving monomial $x_{\phi_{A(L_3)}(v_2), \phi_3(v_3)} \cdot x_{\phi_3(v_3), \phi_{A(L_3)}(v_4)} = x_{i_2, i_3} \cdot x_{i_3, i_4}$. Thus,

$$\mathcal{F}(L_3 \mid \phi_{A(L_3)}) = \sum_{\substack{i_3 \in [n] \\ \Sigma_3}} \underbrace{(x_{i_2, i_3} \cdot x_{i_3, i_4})}_{\Pi_3}.$$

The size is $O(n^\lambda) = O(n^3)$, matching the cost $\lambda = 3$.