

Comparing Stack Layouts and Queue Layouts

Bachelor's Thesis of

Fabian Tolksdorf

At the Department of Informatics
Institute of Theoretical Informatics (ITI)

Reviewer:	Dr. Torsten Ueckerdt
Second reviewer:	Prof. Dr. Thomas Bläsius
Advisor:	Laura Merker

25.05.2026 – 11.09.2026

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself. I have not used any other than the aids that I have mentioned. I have marked all parts of the thesis that I have included from referenced literature, either in their original wording or paraphrasing their contents. I have followed the by-laws to implement scientific integrity at KIT.

Karlsruhe, 11.09.2026

.....
(Fabian Tolksdorf)

Abstract

A *stack layout* (*queue layout*) of a graph is an ordering of its vertices together with a partition of its edges such that no part contains two edges with alternating (nested) endpoints. The *stack number* $\text{sn}(G)$ (*queue number* $\text{qn}(G)$) minimizes the number of sets in a stack layout (queue layout) of a graph G .

It is a long-standing question whether there is a function f such that it holds for every graph G that its queue number $\text{qn}(G)$ is at most $f(\text{sn}(G))$. Recently, the reverse direction, whether there is a function f such that it holds for every graph G that $\text{sn}(G) \leq f(\text{qn}(G))$, has been answered in the negative [Dujmović et al. 2022].

We focus on graphs that are interesting regarding the first question. That is, we want to find graphs with small stack number and relatively large queue number. While we unfortunately are unable to answer the question, we present candidates for which our best upper bounds on the queue number are asymptotically worse than our best upper bounds on the stack number. We further consider versions of the queue number in which we only allow a restricted set of vertex orderings. So, we ask whether these modified versions are bounded by a function of the queue number aiming to thereby ease proving lower bounds on the queue number. We prove that the queue number restricted to layouts with vertex orderings, in which vertices are sorted by their distance to a designated vertex, is at most $3 \cdot \text{qn}(G)$ for every graph G . We show that, when restricting the vertex orderings even more to so-called “Lex-BFS-orderings”, for every $q \in \mathbb{N}_+$ with $q \geq 3$, there is a graph G with queue number at most q and queue number in the restricted version at least 2^{q-2} .

Zusammenfassung

Ein *Stack Layout* (*Queue Layout*) eines Graphen ist eine Ordnung der Knoten zusammen mit einer Partitionierung der Kanten, sodass in jeder Menge keine zwei Kanten mit alternierenden (umschlossenen) Endpunkten sind. Die *Stack Number* $\text{sn}(G)$ (*Queue Number* $\text{qn}(G)$) minimiert die Anzahl Mengen in einem Stack Layout (Queue Layout) für einen Graphen G .

Es ist eine seit langem offene Frage, ob es eine Funktion f gibt, sodass für jeden Graphen G dessen Queue Number $\text{qn}(G)$ höchstens $f(\text{sn}(G))$ ist. Erst kürzlich wurde gezeigt, dass es in der umgekehrten Richtung keine Funktion f gibt, sodass für jeden Graphen G die Abschätzung $\text{sn}(G) \leq f(\text{qn}(G))$ gilt [Dujmović et al. 2022].

Wir fokussieren uns auf Graphen, die interessant bezüglich der ersten Frage erscheinen. Das heißt, wir wollen Graphen mit kleiner Stack Number und vergleichsweise großer Queue Number finden. Es gelingt uns zwar leider nicht, die Frage zu beantworten, jedoch präsentieren wir einige Kandidaten für die unsere besten oberen Schranken für die Queue Number asymptotisch schlechter sind als unsere besten oberen Schranken für die Stack Number. Des Weiteren betrachten wir Varianten der Queue Number, in denen nicht alle Knotenordnungen erlaubt sind. Wir fragen uns, ob diese Varianten jeweils von einer Funktion der Queue Number beschränkt sind. Dadurch wollen wir es vereinfachen, untere Schranken für die Queue Number zu finden. Wir beweisen, dass die Variante der Queue Number, in der wir nur Layouts mit Knotenordnungen, in denen Knoten nach Abstand zu einem ausgewählten Knoten sortiert sind, für jeden Graphen G höchstens $3 \cdot \text{qn}(G)$ ist. Wir zeigen weiterhin, dass es, wenn wir die Knotenordnungen noch weiter auf sogenannte “Lex-BFS-orderings” einschränken, für jedes $q \in \mathbb{N}_+$ mit $q \geq 3$, einen Graphen G gibt, der höchstens Queue Number q und in der eingeschränkten Version mindestens Queue Number 2^{q-2} hat.

Contents

1	Introduction	1
1.1	Outline	1
1.2	Motivation and Related Work	2
1.3	Contributions	3
2	Preliminaries	9
2.1	General Graph Theory	9
2.1.1	Graph Classes	9
2.1.2	Graph Operations	10
2.1.3	Graph Parameters	10
2.1.4	Trees	11
2.2	Vertex Orderings	11
2.3	Stack Number and Queue Number	13
2.4	Treewidth and Related Concepts	15
2.5	Known Results	15
2.5.1	Results on Treewidth and Related Concepts	15
2.5.2	Results on Stack Layouts and Queue Layouts	16
2.5.3	Results on Binary Trees	17
3	Subdivisions of Complete Graphs	19
4	Some Interesting Graphs	25
4.1	Tree-Based Graphs	25
4.1.1	A Bound on the Stack Number of Graphs with Bounded Treewidth	32
4.2	HUDG-inspired graphs	33
4.2.1	Clique Number of the Graphs H_r	34
4.2.2	Stack Number of the Graphs H_r	42
4.2.3	The Intermediate Graphs H_r^P	44
4.2.4	The Family $\mathcal{H}$	46
4.2.5	Queue Number of the Graphs H_r	48
4.2.6	Modifying the Graphs H_r	49
4.2.7	Queue Number of the Graphs H_r^P and $H_s'^P$	52
5	Queue Numbers with Restricted Vertex Orderings	55
5.1	BFS-orderings	55
5.2	Queue-BFS-orderings and Lex-BFS-orderings	60
5.3	Comparing Queue Number and Lex-BFS-Queue Number	66
5.4	Cherries	70
6	Conclusion	73
	Bibliography	75

1 Introduction

A *linear layout* is a vertex ordering of a graph together with a partition of its edges such that each set meets some given criteria. The most common representatives are *stack layouts* and *queue layouts* in which each set is processable by a stack or a queue, respectively. That is, edges are inserted into the data structure at the first endpoint in the order and removed at the second endpoint. The *stack number* $sn(G)$ and *queue number* $qn(G)$ minimize the number of sets in a stack layout and queue layout of a graph G , respectively. The stack number was introduced by Bernhart and Kainen [3] in 1979 and the queue number thirteen years later by Heath and Rosenberg [29] as the converse concept. Hence, comparing stack layouts and queue layouts can be considered as asking whether one of the data structures is “more powerful”. Given that the stack number and queue number minimize the number of sets in the respective layouts over all vertex orderings, it is apparent that establishing lower bounds is harder than upper bounds for which it suffices to provide one “good” layout. This can also be seen in an overview by Pupyrev [32] that contains more upper bounds than lower bounds for stack number and queue number of various graph classes.

In this thesis, we compare queue number and stack number of some graphs aiming to find graphs with small stack number and large queue number. Furthermore, since determining nontrivial bounds for the queue number of some of those graphs turns out to be rather difficult, we define restricted versions of the queue number. This is an attempt to limit considered vertex orderings while still serving as a lower bound for the regular queue number. If those restricted versions of the queue number are upper-bounded by a function of the queue number, there is a function of the restricted version serving as a lower bound for the regular queue number. For example take *separated queue layouts*, in which the vertices are partitioned into fixed sets A and B and one of the sets is completely before the other in the vertex ordering. Pemmaraju [31, Theorem 3.13] showed that the *separated queue number* is at most three times the regular queue number. Hence, the queue number is lower bounded by a third of the separated queue number.

1.1 Outline

We begin by providing an overview of previous work on the topics in Section 1.2. Afterwards, in Section 1.3, we present a preview of our results. We then define our notations in Chapter 2 and provide definitions to essential concepts of the thesis. Furthermore, we cite some classical and well-known results which we use in proofs in later chapters. In Chapter 3, we discuss the stack number and queue number of certain subdivisions of complete graphs. In Chapter 4, we introduce the concept of tree-based graphs in an attempt to find a family of graphs with bounded stack number but unbounded queue number. In Chapter 5, we define variations of the queue number in which we restrict the considered vertex orderings, aiming to ease determining bounds on the queue number. Finally, in Chapter 6, we discuss our results and some open questions they leave.

1.2 Motivation and Related Work

Stack layouts and queue layouts have a variety of applications, some of which are presented by Dujmović and Wood [19]. Even more applications arise when considering linear layouts as optimization problems with cost functions other than stack number or queue number [11].

Shortly after introducing the queue number [29], Heath and Rosenberg proved, now together with Leighton, that the graphs with queue number 1 have stack number at most 2 ([28, Theorem 4.2]), and conversely, that the graphs with stack number 1 have queue number at most 2 ([28, Theorem 4.3]). Naturally, they asked whether such a behavior extends to graphs of arbitrary queue number and stack number. Therefore, they conjectured [28, Section 5] that queue number and stack number of a graph are bounded by a function of one another, here formulated as questions.

Open Problem 1.1: *Is the stack number bounded by a function of the queue number?*

Open Problem 1.2: *Is the queue number bounded by a function of the stack number?*

Open Problem 1.1 has since been answered in the negative by Dujmović, Eppstein, Hickingbotham, Morin, and Wood [15]. Eppstein, Hickingbotham, Merker, Norin, Seweryn, and Wood [23] further present a family of graphs with bounded degree, bounded queue number and bounded stack number. So there is no function bounding the stack number in terms of queue number and maximum degree either. The converse direction (Open Problem 1.2) is still open, but there are equivalent statements. Katheder, Kaufmann, Pupyrev, and Ueckerdt [30] present and prove some of them. For example, they prove that Open Problem 1.2 is equivalent to the question whether the queue number is bounded by a function of the *mixed number*, where the mixed number minimizes over the number of sets in a mixed layout in which each set is a stack or a queue.

We have a different situation for bipartite graphs. Pemmaraju [31, Corollary 3.11] proved that the stack number of a bipartite graph is at most double its queue number. In contrast, Dujmović and Wood [20, Theorem 8] proved that Open Problem 1.2 is equivalent to 3-stack bipartite graphs having bounded queue number. Therefore, a bound reversely to [31, Corollary 3.11] would solve Open Problem 1.2 in the positive.

A longstanding candidate with bounded stack number and super-constant best known upper bounds for the queue number were planar graphs. Buss and Shor [8] proved that every planar graph has stack number at most 9, a result which Yannakakis [38] improved to 4. Finally, Bekos, Kaufmann, Klute, Pupyrev, Raftopoulou, and Ueckerdt [1], and independently Yannakakis [39], showed that this bound is tight. Heath, Leighton, and Rosenberg [28] conjectured in 1992 that planar graphs have bounded queue number, but it took until 2020 for Dujmović, Joret, Micek, Morin, Ueckerdt, and Wood [16] to prove it. Before they proved an upper bound of 49, only super-constant bounds were known.

This led Pemmaraju [31] to conjecture that a certain family of planar 3-trees, the so-called *stellated triangles*, has unbounded queue number which would resolve Open Problem 1.2 in the negative. The conjecture was debunked by Dujmović, Morin, and Wood [17] when they provided a first upper bound on the queue number in terms of the treewidth of a graph. The treewidth is a parameter that measures how similar a graph is to a tree. Often, treewidth is used as a parameter to allow algorithms which are efficient in terms of the number of vertices

and edges of a graph but have a super-polynomial term in its treewidth, e.g., with a linear term in the number of vertices for calculating a maximum independent set [10, Theorem 7.5] when given a tree-decomposition¹ of the graph.

Wiechert [37] provided the currently best known bounds for the queue number in terms of the treewidth k by providing some k -trees² with queue number at least $k + 1$ for $k \geq 2$ and a proof that all graphs with treewidth k have queue number at most $2^k - 1$. This is a rather large gap especially given that the converse problem is solved for the stack number. Ganley and Heath [27] proved that every graph with treewidth k has stack number at most $k + 1$. Dujmović and Wood [18] gave an example of a family of graphs with treewidth k and stack number exactly $k + 1$ for $k \geq 3$, thus showing that the bound by Ganley and Heath is tight for $k \geq 3$.

Our research is mainly motivated by Open Problem 1.2 and the intention of working towards answering it in the negative. So, we consider graphs that are possibly good candidates. We further try to simplify the tedious work of determining nontrivial lower bounds on the queue number of a graph.

1.3 Contributions

Building on previous work of Blankenship and Oporowski [4], Dujmović and Wood [20] and Eppstein [21], we summarize results on the 1-subdivisions of complete graphs.

Theorem 1.3 (Theorem 3.6): *Let $n \in \mathbb{N}_+$. The queue number of $K_n^{(1)}$ is in $\Theta(\sqrt{n})$. The stack number of $K_n^{(1)}$ is in $\omega(1) \cap O(\sqrt{n})$.*

We emphasize that the 1-subdivisions of complete graphs thus serve as an example of graphs whose stack number is asymptotically at most their queue number. Since the stack number of each graph is super-constant in terms of n , the number of vertices each graph is bounded by a function of the stack number. With the queue number being bounded by the number of vertices, the 1-subdivisions of complete graphs are no example which decides Open Problem 1.2 in the negative. Since we need large queue number and relatively small stack number, it still is an example in the right direction when assuming that Open Problem 1.2 can be answered negatively.

As Open Problem 1.1 was resolved in the negative [23] by providing a family of graphs with bounded queue number and unbounded stack number, we would like to similarly find a family of graphs with bounded stack number and unbounded queue number. This immediately requires it to have unbounded treewidth since there is a bound on the queue number in terms of treewidth [37]. So we focus on graphs with small stack number. In the proof that planar graphs have bounded queue number [16], the authors use a tool called *product structure*³ and show that graph families admitting product structure have bounded queue number. Bläsius, Dohse, Haun, and Merker [7] describe a family of hyperbolic uniform disk graphs that do not admit product structure. Since it is necessary for a family with bounded queue number to not admit graph structure, we iteratively define more promisingly looking families ending up at a variation of the family of hyperbolic uniform disk graphs from [7]. We determine their stack number in various forms (see Corollaries 4.18, 4.22, 4.33 and 4.37) and provide some

¹For a formal definition of a tree-decomposition, see [10 | 12].

²A k -tree is a complete graph on $k + 1$ vertices or a graph which can be obtained by adding a new vertex to a k -tree such that its neighbor is a clique on k vertices.

³For a definition and some applications of product structure, see [14].

bounds for their queue number (see Lemma 4.29 and Corollaries 4.34, 4.41 and 4.42). All in all, we arrive at a family of graphs with bounded clique number, bounded stack number and unbounded treewidth. We do not know a constant upper bound for its queue number, so it is an interesting candidate regarding the question of the existence of a function bounding the queue number in terms of the stack number (Open Problem 1.2).

To determine these constant-bounded stack numbers, we introduce the concept of ancestor-preserving tree-based graphs which allows for some nice vertex orderings of chordal⁴ graphs with treewidth k while admitting stack layouts using $k + 1$ stacks. This meets the tight [18] upper bound given by Ganley and Heath [27]. What is nice about these vertex orderings, is that for our purposes they allow adding the edges that cause unbounded treewidth in our variation of the hyperbolic uniform disk graphs while using only a constant number of additional stacks.

Theorem 1.4 (Ganley and Heath, 2001 [27], see also Corollary 4.7): *Let G be a graph with treewidth k for some $k \in \mathbb{N}_0$. We have $\text{sn}(G) \leq k + 1$.*

Given that it is hard to determine lower bounds on the queue number, we aim to lighten the effort by restricting the considered vertex orderings in hope to derive a lower bound for the regular queue number. To find such a lower bound, we instead focus on upper bounding our restricted version by a function of the queue number. For example take the separated queue number from the beginning. It asymptotically serves as a lower bound for the queue number while for an n -vertex graph with adequately chosen parts, “only” $2 \cdot \lfloor \frac{n}{2} \rfloor! \cdot \lceil \frac{n}{2} \rceil!$ vertex orderings have to be considered compared to $n!$ for the regular queue number.

For some graph G , some vertex ordering σ and some vertex $r \in V(G)$, in this section, we consider $\sigma'/\sigma''/\sigma'''$ to be the vertex ordering obtained by performing a BFS⁵/queue-BFS⁶/Lex-BFS⁷ from start vertex r while using σ as a tiebreaker, respectively. We refer to Chapters 2 and 5 for the definitions of the algorithms and how the tiebreakers are used precisely.

We prove that we can reorder vertices based on their distance to a specific vertex, i.e., obtain σ' from σ , while increasing queue number at most by a constant factor.

Lemma 1.5 (Lemma 5.1): *Let G be a graph with a vertex ordering σ and vertex $r \in V(G)$. We have $\text{qn}_{\sigma'}(G) \leq 3 \cdot \text{qn}_{\sigma}(G)$.*

Strengthening Lemma 5.1, we show in Lemma 5.2 that this result is tight, i.e., that this bound cannot be improved, by providing graphs meeting the given upper bound.

In contrast to Lemma 5.1, we prove that such modifications of vertex orderings with queue number being within a constant factor are generally not possible with the versions using queue-BFS and Lex-BFS, i.e., that the queue number with respect to σ'' and σ''' may be unbounded even if the queue number with respect to σ is bounded.

Lemma 1.6 (Lemma 5.9): *For every $\ell \in \mathbb{N}_+$, there is a graph G with a vertex ordering σ and a vertex $r \in V(G)$ such that $\text{qn}_{\sigma}(G) \leq 3$, but we have $\text{qn}_{\sigma''}(G) \geq \ell$ and $\text{qn}_{\sigma'''}(G) \geq \ell$.*

⁴A graph is chordal if every induced cycle has length 3.

⁵Note that, in this thesis, we consider a BFS to be an algorithm that visits vertices ordered by their distance to a distinguished start vertex. This could be implemented by maintaining sets of discovered vertices by their distance to the start vertex and choosing an unvisited vertex with the least distance as the vertex to visit next. A vertex is discovered when one of its neighbors is visited.

⁶The queue-BFS is a version of the BFS in which we use a queue to pick the next vertex.

⁷The Lex-BFS may reorder vertices in the queue which were first discovered through the same neighbor.

This provides the following two corollaries which record this result from a different point of view: We do not have a function that bounds the queue number with respect to σ'' in terms of the queue number with respect to σ for every graph, vertex ordering, and start vertex.

Corollary 1.7 (Corollary 5.10): *There is no function f such that for every graph G , every vertex ordering σ of G and every start vertex $r \in V(G)$, we have $\text{qn}_{\sigma''}(G) \leq f(\text{qn}_{\sigma}(G))$.*

We have the same statement but with the vertex ordering σ''' instead of σ'' . Recall that the vertex orderings σ'' and σ''' depend on the graph G , the vertex ordering σ and the vertex r .

Corollary 1.8 (Corollary 5.11): *There is no function f such that for every graph G , every vertex ordering σ of G and every start vertex $r \in V(G)$, we have $\text{qn}_{\sigma'''}(G) \leq f(\text{qn}_{\sigma}(G))$.*

As mentioned above, we construct versions of the queue number with restricted sets of vertex orderings. In particular, we restrict ourselves, for versions of the queue number, to vertex orderings that are obtainable by performing a BFS, queue-BFS or Lex-BFS, respectively, while optionally additionally restricting to a fixed start vertex r . We denote them for a graph G by $\text{BFSqn}(G)$, $\text{qBFSqn}(G)$ and $\text{LexBFSqn}(G)$ ($\text{BFSqn}(r, G)$, $\text{qBFSqn}(r, G)$ and $\text{LexBFSqn}(r, G)$), respectively.

Deducing from Lemma 5.1 yields that restricting considered vertex orderings to r -BFS-orderings for some fixed vertex r does not asymptotically change the queue number, i.e., for graphs G we have $\Theta(\text{qn}(G)) = \Theta(\text{BFSqn}(r, G))$.

Corollary 1.9 (Corollary 5.5): *For every graph G and vertex $r \in V(G)$, we have*

$$\frac{1}{3} \cdot \text{BFSqn}(r, G) \leq \text{qn}(G) \leq \text{BFSqn}(r, G).$$

We know that the queue number with respect to σ''' is not bounded by a function of the queue number with respect to σ , but the (r) -Lex-BFS-queue number might still be bounded by a function of the queue number. While the r -BFS-queue number is pretty small – compared to the queue number – for every graph and start vertex r , we do however find graphs that have high – compared to their queue number – r -Lex-BFS-queue number for every start vertex r . Hence, they have high Lex-BFS-queue number compared to their queue number.

Theorem 1.10 (Theorem 5.20): *For each integer $\kappa \geq 2$, there is a chordal graph with treewidth κ , queue number at most $\kappa + 1$ and Lex-BFS-queue number at least $2^{\kappa-1}$.*

Therefore, we derive the following corollary by providing an exemplary family of chordal graphs.

Corollary 1.11 (Corollary 5.21): *If the Lex-BFS-queue number is bounded by a function of the queue number, i.e., if there is a function f such that it holds for every graph G that $\text{LexBFSqn}(G) \leq f(\text{qn}(G))$, then we know $f(q) \in \Omega(2^q)$.*

See Table 1.1 for a tabular overview of results that we find on the different versions of the queue number and some unknown – potentially not existing – related bounds. Further, see Figure 1.1 for an overview which of the results imply which other results in case they hold.

	$\exists f$	$f(q) \in \Omega(\cdot)^\diamond$	$f^*(q) \in O(\cdot)$
$\text{qn}_{\sigma'}(G) \leq f(\text{qn}_\sigma(G))$	✓	q (5.2)	q (5.1)
$\text{BFSqn}(r, G) \leq f(\text{qn}(G))$	✓	q (trivial)	q (5.5)
$\text{BFSqn}(G) \leq f(\text{qn}(G))$	✓	q (trivial)	q (5.5)
$\text{qn}_{\sigma''}(G) \leq f(\text{qn}_\sigma(G))$	✗ (5.10)	✗	✗
$\text{qBFSqn}(r, G) \leq f(\text{qn}(G))$	?	q (trivial)	?
$\text{qBFSqn}(G) \leq f(\text{qn}(G))$	?	q (trivial)	?
$\text{qn}_{\sigma'''}(G) \leq f(\text{qn}_\sigma(G))$	✗ (5.11)	✗	✗
$\text{LexBFSqn}(r, G) \leq f(\text{qn}(G))$	?	2^q (5.14,5.15)	?
$\text{LexBFSqn}(G) \leq f(\text{qn}(G))$	?	2^q (5.21)	?

Table 1.1: This table contains an overview of the results on our variants of the queue number. The column “ $\exists f$ ” states whether there is a function such that the statement in the leftmost column hold for every graph G , vertex ordering σ and vertex r . Note that, with Theorem 5.22, the existence of such a function is equivalent for a modified queue number and its version with a fixed start vertex. The rightmost column provides the asymptotic best known function f^* that fulfills the statement for every graph G , vertex ordering σ and vertex r . And the second rightmost column states an asymptotic lower bound for f assuming that such a function exists. Figure 1.1 illustrates relationships between the rows. The vertex orderings $\sigma', \sigma'', \sigma'''$ are defined as in Section 1.3 and are dependent on G, σ and r .

✓: Such a function f exists.

?: Open whether such a function f exists.

✗: No such function f exists.

$\diamond$: If there is a function f that fulfills the statement, then we know that $f \in \Omega(\cdot)$.

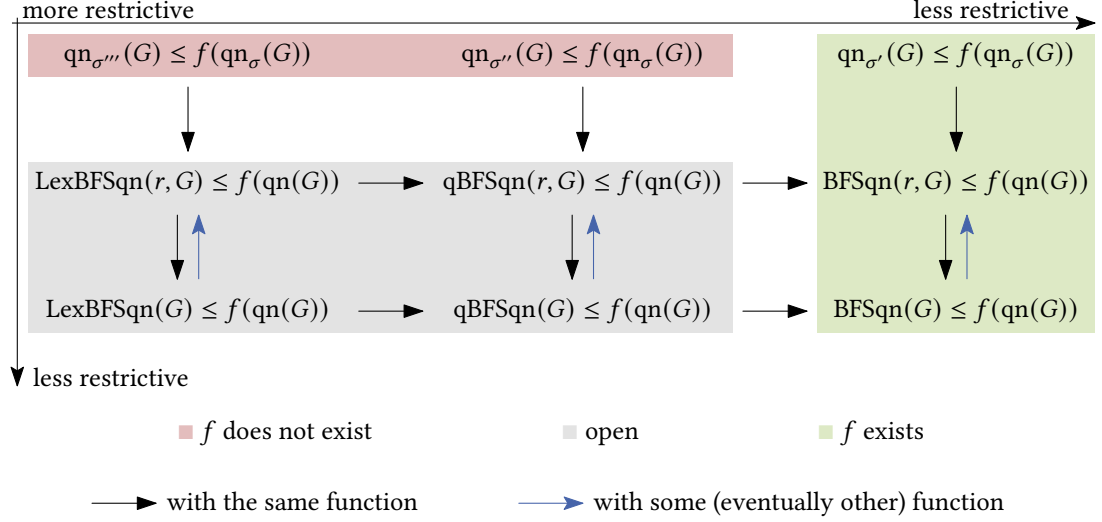


Figure 1.1: Displays for an increasing function f that if a stated inequality holds for every graph G , vertex ordering σ and start vertex r , which other inequalities are implied to hold as well, given by the direction of the black arrows. Hence, lower bounds for f – if there are functions such that the statements hold in both cases – follow the black arrows in reverse direction. The vertex orderings $\sigma', \sigma'', \sigma'''$ are defined as in Section 1.3 and are dependent on G, σ and r . The implications follow immediately by definition since the definitions provide a certain hierarchy in the lower two rows and every column as the considered vertex orderings get increasingly less restrictive going down and to the right. We have:

■ in the rows

- $\text{BFSqn}(r, G) \leq \text{qBFSqn}(r, G) \leq \text{LexBFSqn}(r, G)$,
- $\text{BFSqn}(G) \leq \text{qBFSqn}(G) \leq \text{LexBFSqn}(G)$,

■ and in the columns

- $\text{BFSqn}(G) \leq \text{BFSqn}(r, G) \leq \text{qn}_{\sigma'}(G)$,
- $\text{qBFSqn}(G) \leq \text{qBFSqn}(r, G) \leq \text{qn}_{\sigma''}(G)$,
- $\text{LexBFSqn}(G) \leq \text{LexBFSqn}(r, G) \leq \text{qn}_{\sigma'''}(G)$.

We see that $\text{LexBFSqn}(r, G) \leq f(\text{qn}(G))$ is the strongest statement which is not yet proven to hold for no function. It comprises the smallest number of vertex orderings and implies every other open question in the figure. The blue arrows signalize that the implication of one estimate implying another estimate is not necessarily given for the same function f , but proven in Theorem 5.22 to imply existence of some other function for which the estimate holds for every graph G and start vertex r .

2 Preliminaries

When we use the natural numbers in this thesis, we write $\mathbb{N}_0$ if we want to include 0 and $\mathbb{N}_+$ if we do not. We annotate both cases to avoid confusion caused by personal preferences. As we often use the set of numbers 1 to n for some $n \in \mathbb{N}_0$, we define $[n] = \{i \in \mathbb{N}_+ \mid i \leq n\}$. For a set X and $k \in \mathbb{N}_0$, the set of subsets of size k is denoted by $\binom{X}{k} = \{S \subseteq X \mid |S| = k\}$. For a set X , its power set is denoted by $2^X = \{S \mid S \subseteq X\}$. We sometimes write $A \dot{\cup} B$ instead of $A \cup B$ for disjoint sets A, B to emphasize that they are disjoint. If not explicitly mentioned otherwise, we consider graphs which are finite, simple, and undirected. Furthermore, every graph has a nonempty set of vertices.

2.1 General Graph Theory

Let $G = (V, E)$ be a *graph* on *vertex* set V with *edge* set $E \subseteq \binom{V}{2}$. The vertex set of G is also called $V(G)$ and similarly its edge set may be referred to as $E(G)$. We typically use n to refer to the number of vertices $|V(G)|$ and m to the number of edges $|E(G)|$ of a graph.

For an edge $e = \{u, v\} \in E$, we write $e = uv$ and consider u and v to be the *endpoints* of the edge e . We call two vertices $u, v \in V$ *adjacent* if they are connected by an edge, i.e., $uv \in E$. Likewise, two edges e, e' are *adjacent* if they share a vertex. An edge e is *incident* to a vertex v if v is an endpoint of e .

The *neighborhood* $N_G(v)$ of a vertex $v \in V$ is the set of adjacent vertices. We denote it as $N_G(v) = \{u \in V \mid uv \in E\}$, although we may write $N(v) = N_G(v)$ if the respective graph is clear from the context. The *degree* of a vertex $v \in V$ is the number of vertices in its neighborhood. We denote it as $\deg_G(v) = |N_G(v)|$, while we may write $\deg(v) = \deg_G(v)$ if the graph G is clear from the context. A vertex with degree 0 is called *isolated* and a vertex with degree $n - 1$ is called *universal*.

A graph H is a *subgraph* of a graph G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$, and we say that G is a *supergraph* of H . Note that the set $E(H)$ still needs to be a subset of $\binom{V(H)}{2}$ as H is a graph. A graph H is an *induced subgraph* of a graph G if $S = V(H) \subseteq V(G)$ and $E(H) = E(G) \cap \binom{S}{2}$. We denote $H = G[S]$ and say that H is *induced* by S . If H is a subgraph of G , we write $H \subseteq G$. Analogously, if H is an induced subgraph of G , we write $H \subseteq_{\text{ind}} G$.

Two graphs G and H are *isomorphic* if there is a bijection $\varphi : V(G) \rightarrow V(H)$ such that $\forall \{u, v\} \in \binom{V}{2} : uv \in E(G) \Leftrightarrow \varphi(u)\varphi(v) \in E(H)$. If for some graphs H and G , there is a subgraph $I \subseteq G$ and I is isomorphic to H , we say that G contains a *copy* of H .

Let G and H be graphs. The graphs G and H are *vertex-disjoint* if they have no shared vertices, i.e., $V(G) \cap V(H) = \emptyset$.

2.1.1 Graph Classes

A *path* of length $k \in \mathbb{N}_0$ is a graph P on $k + 1$ vertices $v_1, \dots, v_{k+1}$ with edge set $E = \{v_i v_{i+1} \mid i \in [k]\}$. It is denoted by P_{k+1} or its vertices $v_1 v_2 \dots v_{k+1}$. The vertices v_1 and v_{k+1} are the *endpoints* of the path. Thus, we might refer to P as a v_1 - v_{k+1} -path. A *path* in a graph G is a subgraph of G that is a path. The distance of two vertices $u, v \in V$ is the length of a shortest u -

v -path. If u and v are not in the same component, their distance is defined to be ∞ . We denote the distance between u and v by $\text{dist}(u, v) = \text{dist}_G(u, v)$. A graph G is *connected* if for each two vertices $u, v \in V(G)$, there is a u - v -path in G . We say that a path P in a graph G is an A - B -path if P is an a - b -path for some $a \in A \subseteq V(G)$ and $b \in B \subseteq V(G)$. A cycle of length $k \in \mathbb{N}_+$, $k \geq 3$, is a graph C on k vertices $v_1, \dots, v_k$ with edge set $E = \{v_i v_{i+1} \mid i \in [k-1]\} \cup \{v_k v_1\}$. It is denoted by C_k or its vertices $v_1 v_2 \dots v_k v_1$. A cycle in a graph G is a subgraph of G that is a cycle. A graph G is *acyclic* if it contains no cycle, i.e., there is no subgraph of G that is a cycle. A *tree* is a graph that is acyclic and connected. A *star* is a tree with a universal vertex. The universal vertex of a star is called the *center* vertex. All other vertices in a star have degree 1. The *complete* graph K_n is a graph on n vertices and edge set $E = \binom{V}{2}$. A *clique* in a graph G is a subgraph of G that is a complete graph. A complete graph on three vertices is a *triangle*. The *empty* graph E_n is a graph on n vertices that contains no edges. An *independent set* in a graph G is a subset of vertices $I \subseteq V(G)$ such that I induces no edges in G .

A *planar* graph is a graph that has a drawing on the plane such that no edges cross in it. This informal definition is sufficient in our case. For a formal definition, we refer to Diestel [12, Chapter 4]. An *outerplanar* graph is a graph that has a drawing on the plane such that no edges cross and every vertex is on the boundary of the outer face.

2.1.2 Graph Operations

For a graph G , we define its k -subdivision to be the graph $G^{(k)}$ where every edge $uv \in E(G)$ is replaced by a path of length $(k+1)$ and the endpoints of the path that replace uv are u and v . The vertices of the original graph G in $G^{(k)}$ are called *branch vertices*. We call the newly added vertices of $G^{(k)}$ *subdivision vertices*. Note that per edge in the graph G , the graph $G^{(k)}$ contains precisely k subdivision vertices. See Figure 2.1 for a 1-subdivision of a K_4 . For a graph G and some integer $k \in \mathbb{N}_+$, a subdivision G' with at most k subdivision vertices per edge is a graph in which every edge uv of G is replaced by a path with at most k subdivision vertices.

Let $G = (V, E)$ be a graph. For $e \in \binom{V}{2}$, $G - e$ defines the graph obtained by removing the edge e , or the same graph if e is not part of G . Formally, $G - e$ is defined as $G - e = (V, E \setminus \{e\})$. For $v \in V$, $G - v$ is the graph that is obtained by removing v and its incident edges from G , i.e., $G - v = G[V \setminus \{v\}]$. Moreover, we define analogously for a set of edges $F \subseteq \binom{V}{2}$ and for a set of vertices $U \subseteq V$ the removal of those sets by $G - F = (V, E \setminus F)$ and $G - U = G[V \setminus U]$. We may also add some edges $F \subseteq \binom{V}{2}$ to the graph, and we denote it by $G + F = (V, E \cup F)$. If we want to add a single edge $e \in \binom{V}{2}$ to the graph, we denote the operation by $G + e = G + \{e\}$. For an edge $e = uv \in E(G)$, *contracting* e yields the graph $G \circ e$ where $V(G \circ e) = (V(G) \setminus \{u, v\}) \cup \{w\}$ for some vertex $w \notin V(G)$ and $E(G \circ e) = \{e' \in E(G) \mid e' \cap e = \emptyset\} \cup \{wx \mid x \in (N_G(u) \cup N_G(v)) \setminus \{u, v\}\}$.

Let $H = (V', E')$ be another graph. We say that H is a *minor* of G if H (or a copy of H) can be obtained from G by a sequence of edge deletions, vertex deletions and edge contractions. In this case, we also say that G *contains* an H -minor.

2.1.3 Graph Parameters

The *diameter* of a graph G is the maximum distance of two vertices in G . We denote the diameter of G by $\text{diam}(G) = \max_{u, v \in V(G)} \text{dist}_G(u, v)$. The *minimum degree* of a graph is denoted as $\delta(G) = \min_{v \in V(G)} \deg_G(v)$ and the *maximum degree* of a graph is denoted as $\Delta(G) = \max_{v \in V(G)} \deg_G(v)$. The *clique number* $\omega(G)$ of a graph G is the order of the largest

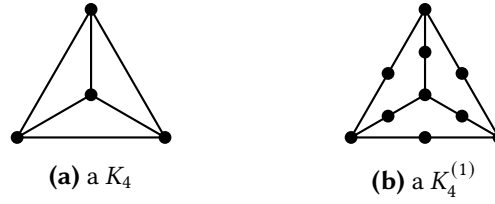


Figure 2.1: Example of a subdivision of a K_4 using paths of length 2.

clique in G , i.e., $\omega(G) = \max \left\{ k \in \mathbb{N}_+ \mid \exists S \in \binom{V(G)}{k} : G[S] \text{ is a clique in } G \right\}$. A *vertex cover* of a graph G is a set X such that every edge of G is incident to a vertex of X . The *vertex cover number* $\tau(G)$ of a graph G is the size of a smallest vertex cover of G .

Let $k \in \mathbb{N}_+$ be an integer. We say that a graph G is *k-colorable* (or *k-vertex-colorable*) if there is a *proper k-coloring* of G , that is a map $c: V(G) \rightarrow [k]$ such that the endpoints of each edge $uv \in E(G)$ have different colors $c(u) \neq c(v)$. This implies that the set of vertices sharing a color in a proper coloring is an independent set. The *chromatic number* $\chi(G)$ of G is the smallest $k \in \mathbb{N}_+$ such that G is *k-colorable*. We say that a graph G is *bipartite* if its vertices can be partitioned into two independent sets. In particular, the bipartite graphs are the graphs with chromatic number at most 2.

2.1.4 Trees

A *rooted tree* is a tree T together with a vertex $r \in V(T)$ which we call its *root* vertex. We also say the tree T *rooted at* r . We may omit explicitly providing the root of a rooted tree and still use the properties obtained by rooting the tree. The main benefit of a rooted tree is that it allows for additional structure which we introduce in the following. Let $T = (V, E)$ be a tree rooted at r . We can partition the vertices of a rooted tree T by their distance to the root r and call the set of vertices with distance i *level* i . Levels of this kind are also called *BFS-levels*. The *height* of the tree T rooted at r is the maximum distance of another vertex to the root vertex r . The *parent* of a vertex, that is not the root, is the unique next vertex on the shortest path to the root. The *children* of a vertex are its neighbors except the parent. A vertex u is an *ancestor* of a vertex v , if there are some vertices $u = v_1, \dots, v_k = v$ such that v_i is a parent of v_{i+1} for all $i \in [k-1]$. Vice versa, this means that the vertex v is a *descendant* of the vertex u . A *leaf* is a vertex with degree 1. The *subtree* of a vertex v in a rooted tree T is the graph that is induced by the vertex v and its descendants. The *transitive closure* of a rooted tree T is a graph on the same set of vertices as T , where the neighborhood of each vertex consists of its ancestors and descendants in the tree. See Figure 2.2 for illustrations of most of the concepts around trees.

A *binary tree* is a rooted tree where every vertex has at most two children. A *complete binary tree* of height h is a binary tree of height h such that every vertex in a level less than h has exactly two children and every vertex in level h has no children.

2.2 Vertex Orderings

An *ordering* σ of a set S is a total order. We compare elements of S using the operator $<_\sigma$. We write $a \leq_\sigma b$ if $a = b$ or $a <_\sigma b$ for elements $a, b \in S$. When we talk about the ordering itself, we however usually refer to it by σ . To refer to or define the ordering σ of the set $S = \{s_1, \dots, s_n\}$, we may write $\sigma = (s_1, \dots, s_n)$ or $\sigma = s_1, \dots, s_n$ instead of writing $s_1 <_\sigma \dots <_\sigma s_n$. A *vertex ordering* σ of a graph G is an ordering of its vertices.

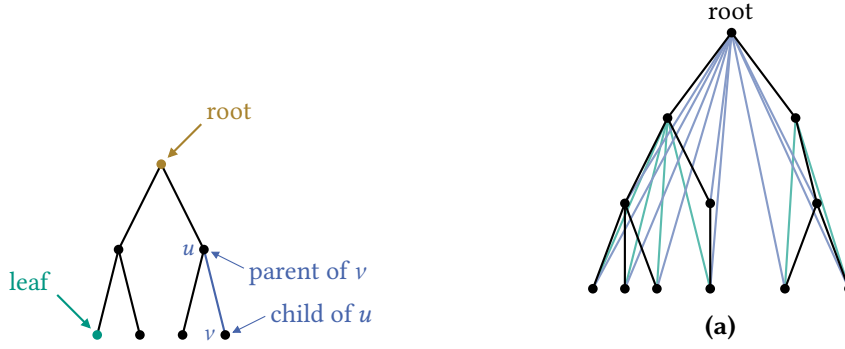


Figure 2.2: Left: A tree rooted at r , annotated with important terms. Right: A rooted tree with its transitive closure. The edges of the tree are black. The transitive closure consists of all depicted edges in the graph.

Let $G = (V, E)$ be a graph and σ be a vertex ordering of G . If $u, v \in V$ are two vertices with $u <_{\sigma} v$, we say that u is to the *left* of v and that v is to the *right* of u . We implicitly use σ as a partial ordering of subsets of vertices of G . That means for sets $A, B \subseteq V$, we denote $A <_{\sigma} B$ if $a <_{\sigma} b$ for every pair of vertices $a \in A, b \in B$. Similarly, we denote $A \leq_{\sigma} B$ if $a \leq_{\sigma} b$ for every pair of vertices $a \in A, b \in B$ and use the notions of *left* and *right*. If one of the sets in the comparison consists of a single vertex v , we abbreviate v for $\{v\}$. The *leftmost* (*rightmost*) vertex in vertex ordering σ of graph G is the unique vertex $v \in V(G)$ such that there is no vertex that is to the left (right) of vertex v in σ .

A *breadth-first search* or *BFS* is an algorithm that explores the vertices of a graph G by starting at a vertex $r \in V(G)$ and exploring all vertices of distance $d \in \mathbb{N}_0$ to r before exploring a vertex that has a distance greater than d to vertex r . Contrary to some definitions, we do not require a BFS to use a queue and vertices of the same distance may be processed in an arbitrary order. An *r-BFS-ordering* σ of a graph G , for $r \in V(G)$ is a vertex ordering that can be obtained by performing a BFS starting at vertex r , i.e., it holds $\text{dist}(r, a) \leq \text{dist}(r, b)$ for each two vertices $a, b \in V(G)$ with $a \leq_{\sigma} b$. A *BFS-ordering* σ of a graph G is a vertex ordering that is an *r-BFS-ordering* of G for some vertex $r \in V(G)$. A *queue-BFS* is a breadth-first search in which the data structure for managing the vertices that have not yet been explored but have been discovered is a queue. The *empty word* is represented by the symbol ε . For two words $s = s_1 s_2 \dots s_l$ and $t = t_1 t_2 \dots t_m$ over the alphabet $[n]$ for some $n \in \mathbb{N}_+$, the word s is lexicographically greater than t if t is a prefix of s or there is some $i \in [l]$ such that $s_i > t_i$ and $s_j = t_j$ for every $j < i$. The *lexicographic breadth-first search* or *Lex-BFS* was introduced by Rose, Tarjan and Lueker in 1976 [35]. It is a specialization of the BFS by prioritizing vertices using a descending lexicographic order of their explored neighbors who receive decreasing labels. We use the definition given by Corneil [9, p. 4], see Algorithm 2.1. When picking the next vertex, the Lex-BFS prefers vertices with neighbors further to the left. More precisely, the next vertex to explore is chosen to be one with the leftmost neighbor like in queue-BFS. If there are however multiple vertices with the same leftmost neighbor, we pick one out of those one with the second leftmost and so on until we have a set of vertices with equal neighborhood in the vertices already processed. If there are still multiple vertices, we pick one arbitrarily. So, we partition the vertices and iteratively refine this partition and then pick a vertex out of the “best” part.

Algorithm 2.1: Lexicographic Breadth-First Search [35 | 9].**Input:** A graph $G = (V, E)$ and a start vertex r of G .**Output:** A vertex ordering σ of G .

```

1  $\sigma \leftarrow$  empty list
2 remaining  $\leftarrow V$ 
3 label( $r$ )  $\leftarrow |V|$ 
4 for each vertex  $v$  in  $V \setminus \{r\}$  do
5   | label( $v$ )  $\leftarrow \varepsilon$ 
6 forall  $i \leftarrow |V|$  down to 1 do
7   | pick a remaining vertex  $v$  with lexicographically largest label
8   | append  $v$  to  $\sigma$ 
9   | remove  $v$  from remaining
10  | for each remaining vertex  $u \in N(v)$  do
11  |   | append  $i$  to label( $u$ )
12 return  $\sigma$ 

```

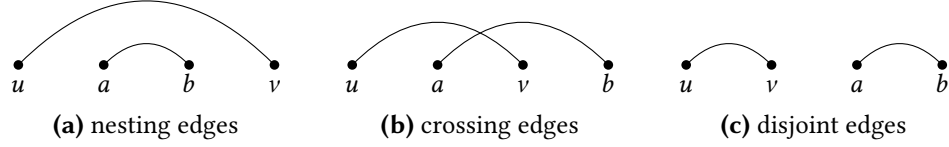


Figure 2.3: Illustrates the possibilities of the relative behavior of two edges ab and uv in a vertex ordering.

The behavior of a Lex-BFS adheres to being a queue-BFS assuming that the order of inserting vertices into the queue is given by an oracle. Similarly to a general BFS, we define the notions of an r -queue-BFS-ordering, a queue-BFS-ordering, an r -Lex-BFS-ordering and a Lex-BFS-ordering. We observe that the set of queue-BFS-orderings is a subset of the set of BFS-orderings and that the set of Lex-BFS-orderings is a subset of the set of queue-BFS-orderings. The same holds for the sets of orderings with a fixed start vertex r .

2.3 Stack Number and Queue Number

Let $G = (V, E)$ be a graph with a vertex ordering σ . If $e = uv, e' = ab \in E$ are two different edges of E such that e and e' have no common endpoints, we want to consider how they are relatively positioned in σ . Without loss of generality, we assume $u <_\sigma v$ and $a <_\sigma b$ and for uniqueness of the situations $u <_\sigma a$ as well. Then, there are three situations. We say that the edges e and e' *nest* if $u <_\sigma a <_\sigma b <_\sigma v$. In this case, the edge e *nests above* e' and the edge e' *nests below* e . We say that the edges e and e' *cross* if $u <_\sigma a <_\sigma v <_\sigma b$, and the edges e and e' are *disjoint* if $u <_\sigma v <_\sigma a <_\sigma b$. See Figure 2.3 for a visualization of the different relative behaviors. Note that if two edges have a common endpoint, none of these three situations applies. If two edges cross (nest) in a vertex ordering, they do so as well visually if the vertices are put in a line according to the vertex ordering and edges are drawn in the half-plane above.

A k -stack layout (k -queue layout) of G is a pair of a vertex ordering σ and a partition of the edges $E = E_1 \cup \dots \cup E_k$ for some $k \in \mathbb{N}_0$ such that for all $i \in [k]$ there are no two edges in E_i that cross (nest). We may omit k and just call it a *stack layout* (*queue layout*). It is called a

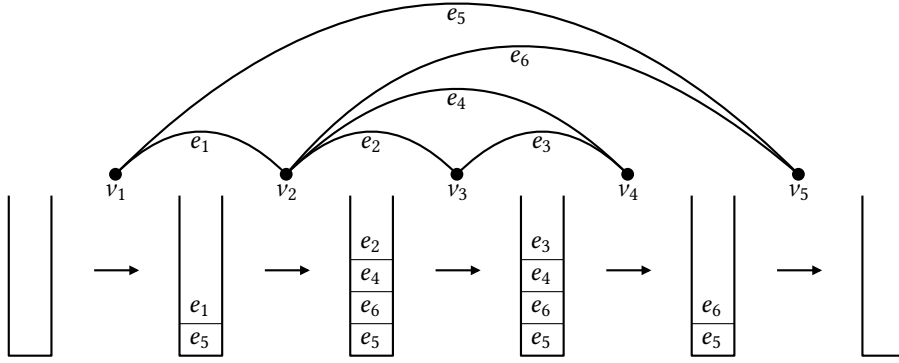


Figure 2.4: An example graph with a 1-stack layout to show how to insert and remove edges from a stack for a stack layout of a graph.

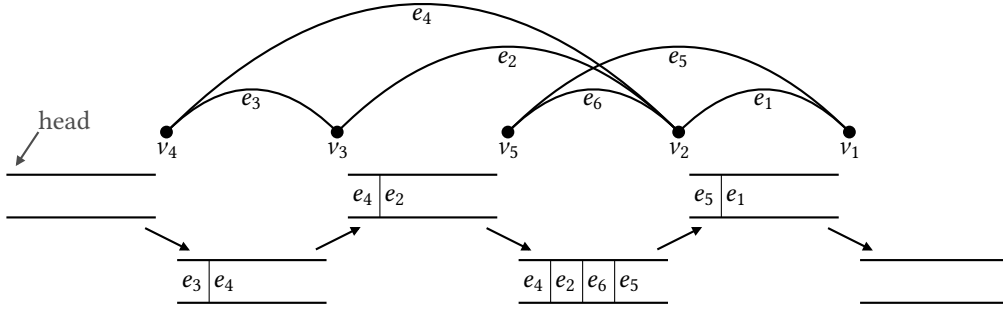


Figure 2.5: An example graph with a 1-queue layout to show how to insert and remove edges from a queue for a queue layout of a graph. Note that this is the same graph as in Figure 2.4.

stack layout (queue layout) because the edges from each edge set E_i may be processed using a stack (queue) by inserting them when the first vertex of an edge appears with respect to σ , and removing them when the second vertex appears. See Figure 2.4 (Figure 2.5) for an example of using a stack (queue) for managing the edges.

The *stack number* (*queue number*) of G with respect to σ is the smallest $k \in \mathbb{N}_+$ such that G allows a k -stack layout (k -queue layout) with vertex ordering σ . We denote it by $\text{sn}_\sigma(G)$ ($\text{qn}_\sigma(G)$). The *stack number* $\text{sn}(G)$ (*queue number* $\text{qn}(G)$) of G is the smallest $k \in \mathbb{N}_+$ such that G has a k -stack layout (k -queue layout) for some vertex ordering σ . This means that the stack number (queue number) minimizes $\text{sn}_\sigma(G)$ ($\text{qn}_\sigma(G)$) over all vertex orderings of G .

A k -twist (k -rainbow) in a vertex ordering σ of G is a set of edges $e_1 = u_1v_1, \dots, e_k = u_kv_k$ such that they pairwise cross (nest), i.e., they appear in the ordering as $u_1 <_\sigma u_2 <_\sigma \dots <_\sigma u_k <_\sigma v_1 <_\sigma v_2 <_\sigma \dots <_\sigma v_k$ ($u_1 <_\sigma u_2 <_\sigma \dots <_\sigma u_k <_\sigma v_k <_\sigma \dots <_\sigma v_2 <_\sigma v_1$). See Figure 2.6 for an illustration of the two patterns. We say that there is a *twist* (*rainbow*) in a vertex ordering σ if there is a k -twist (k -rainbow) in σ for some $k \geq 2$. If there is a k -twist (k -rainbow) in a vertex ordering σ of G , then the stack number (queue number) with respect to σ trivially is at least k .

A *leveled-planar graph* is a graph G which has a *leveled-planar embedding*. That is a partition of $V(G)$ into levels $V_1, \dots, V_m$ such that exactly the vertices of level i are on y -coordinate $-i$ with pairwise distinct x -coordinates and each edge is a straight line segment between vertices of consecutive levels. We further require the straight line segments to be internally disjoint, i.e., they may only share points in the plane at their ends which are vertices. A leveled-planar



Figure 2.6: A 4-twist (left) and a 4-rainbow (right).

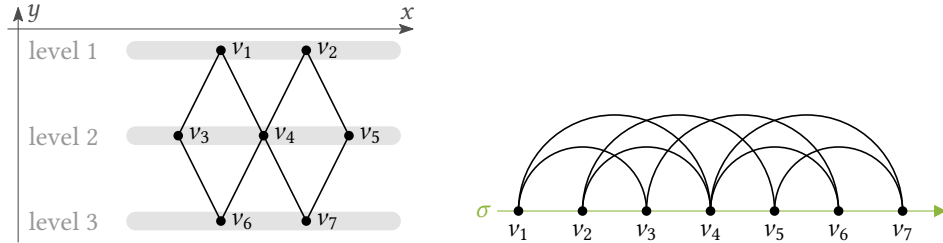


Figure 2.7: A leveled-planar graph with a leveled-planar embedding (left) and the induced vertex ordering admitting a 1-queue layout (right).

embedding *induces* a vertex ordering by ordering levels $V_1 < \dots < V_m$ and internally sorting vertices ascending by their x -coordinates. See Figure 2.7 for an exemplary leveled-planar graph.

2.4 Treewidth and Related Concepts

A *chord* in a cycle is an edge that connects that two vertices of the cycle but is itself not part of the cycle. A *chordal* graph is a graph in which every cycle on four or more vertices has a chord. This means that every induced cycle is a triangle. The *treewidth* $\text{tw}(G)$ is a parameter that measures how close a graph is to being a tree in some way. It was introduced by Robertson and Seymour [33] and is classically defined via so-called tree-decompositions. We primarily use the equivalent definition of the treewidth as being one less than the minimal clique number of a chordal supergraph [33]. A *k-tree* is a K_{k+1} or a graph G such that there is a vertex $v \in V(G)$ for which $G - v$ is a k -tree and the graph induced by the neighborhood of v is a clique on k vertices. A *perfect elimination scheme* for a graph G is a vertex ordering σ such that for every vertex $v \in V(G)$ the set of neighbors of v that are to the left of v is an empty set or forms a clique in G . More formally, this means that the set $X = \{u \in N_G(v) \mid u <_\sigma v\}$, if it is not empty, induces a clique in G . Note that in some publications, perfect elimination schemes are defined such that the neighbors to the right of each vertex induce a clique. A *simplicial* vertex is a vertex which is isolated or whose neighborhood induces a clique.

2.5 Known Results

We present some previously known results which we use in this thesis. We further note some corollaries that shorten later arguments.

2.5.1 Results on Treewidth and Related Concepts

Dujmović, Morin, and Wood [17] mention some statements which are equivalent to the classical definition of treewidth. Most importantly, we have that the treewidth of a chordal graph is approximately equal to its clique number.

Lemma 2.1 ([17, Lemma 2.1]): *Let G be a graph and $k \in \mathbb{N}_+$. The following are equivalent:*

- G has treewidth $\text{tw}(G) \leq k$,
- G is a subgraph of a k -tree,
- G is a subgraph of a chordal graph with clique number at most $k + 1$.

The equivalence between the first and the third statement of Lemma 2.1 is shown using the following lemma.

Lemma 2.2 ([33]): *Let G be a chordal graph. Then the treewidth of G is $\text{tw}(G) = \omega(G) - 1$.*

Given this lemma, it is clearly useful to be able to show that a graph is chordal. One way to do this is via perfect elimination schemes.

Lemma 2.3 (Rose, 1970 [34, Theorem 1]): *Let G be a graph. The graph G is chordal if and only if there is a perfect elimination scheme for G .*

One key insight allowing to prove Lemma 2.3 is given by the following lemma.

Lemma 2.4 (Dirac, 1961 [13]): *Every chordal graph has a simplicial vertex.*

A perfect elimination scheme for a chordal graph can be found using a lexicographic breadth-first search.

Lemma 2.5 (Rose, Tarjan and Lueker, 1976 [35, Section 4.2]): *Every Lex-BFS-ordering of a chordal graph is a perfect elimination scheme.*

Berge [2] observed that induced subgraphs of chordal graphs are chordal which is useful if we want to consider certain induced subgraphs of chordal graphs.

Observation 2.6 ([2, p. 161]): *Every induced subgraph of a chordal graph is chordal.*

This means that the class of chordal graphs is closed under taking induced subgraphs, we say it is *hereditary*.

2.5.2 Results on Stack Layouts and Queue Layouts

As we determine stack number and queue number of several graph classes, we mention some earlier results that we frequently use throughout this thesis. Firstly, we mention the result that leveled-planar graphs have a 1-queue layout.

Lemma 2.7 (Heath and Rosenberg, 1992 [29, Lemma 3.3]): *Every leveled-planar graph has a 1-queue layout.*

This result allows for simple visual proofs of some graphs to have queue number 1. We continue with the stack number of complete graphs.

Lemma 2.8 (Bernhart and Kainen, 1979 [3, Theorem 3.4]): *Let $n \in \mathbb{N}_+$, $n \geq 4$, and let K_n be the complete graph on n vertices. Then $\text{sn}(K_n) = \lceil \frac{n}{2} \rceil$.*

As we regularly want to use the clique number as a lower bound for the stack number, we derive the following corollary.

Corollary 2.9: *Let G be a graph. Then $\text{sn}(G) \in \Omega(\omega(G))$.*

Proof. We use Lemma 2.8, that the stack number of K_n is $\lceil \frac{n}{2} \rceil$ for $n \geq 4$. Therefore, the stack number of K_n is in $\Theta(n)$. Let C be a clique of order $\omega(G)$ in G . Then we have for the stack number of G that $\text{sn}(G) \geq \text{sn}(C) \in \Theta(\omega(G))$. Thus, $\text{sn}(G) \in \Omega(\omega(G))$. $\square$

We have similar results for the queue number of complete graphs and a lower bound for the queue number in terms of the clique number for arbitrary graphs.

Lemma 2.10 (Heath and Rosenberg, 1992 [29, Proposition 4.7]): *Let $n \in \mathbb{N}_+$ and K_n be the complete graph on n vertices. Then $\text{qn}(K_n) = \lfloor \frac{n}{2} \rfloor$.*

As we regularly want to use the clique number as a lower bound for the queue number, we derive the following corollary. The proof is analog to the proof of Corollary 2.9.

Corollary 2.11: *Let G be a graph. Then $\text{qn}(G) \in \Omega(\omega(G))$.*

We have a pretty straight forward upper bound on both the queue number and the stack number by the vertex cover number of a graph.

Lemma 2.12: *Let G be a graph and $\tau(G)$ its vertex cover number. Then $\text{sn}(G) \leq \text{sn}_\sigma(G) \leq \tau(G)$ and $\text{qn}(G) \leq \text{qn}_\sigma(G) \leq \tau(G)$ for every vertex ordering σ of G .*

This result has been proven for the stack number by Bernhart and Kainen [3, Lemma 3.2] as well. We provide the proof for the result on the queue number. The proof for the result on the stack number can be obtained by replacing “queue” with “stack” and “nest” with “cross” in the following proof.

Proof. Let X be a minimum vertex cover of G and σ be a vertex ordering of G . We aim to show that there is a $\tau(G)$ -queue layout with vertex ordering σ . We create a queue Q_x for every vertex $x \in X$ and partition the edges into stars with centers in X where the edges in the star with center $x \in X$ are assigned to queue Q_x . This is possible since X is a vertex cover. Consider the queue Q_x for $x \in X$. Since the edges in Q_x are a star, every edge in Q_x is incident to the vertex x , and therefore, the set Q_x does not contain any nesting edges.

Hence, we have a $\tau(G)$ -queue layout with vertex ordering σ , thereby completing the proof. $\square$

2.5.3 Results on Binary Trees

Let T_h be a complete binary tree of height $h \in \mathbb{N}_0$. The *in-order* vertex ordering σ_h^{in} is obtained by performing a depth-first-search starting at the root of T_h and processing vertices in order, i.e., we process the left subtree first, then the parent, and then the right subtree. See Figure 2.8 for an illustration of T_3 with σ_h^{in} .

We make some observations that are useful for some results later in the thesis. As the vertex ordering σ_h^{in} has been obtained by a depth-first-search, we have the following observation.

Observation 2.13: *Let T_h be a complete binary tree of height $h \in \mathbb{N}_0$. For each vertex $v \in V(T_h)$, the following holds: The descendants of v together with v itself appear as a contiguous sequence, a block, in σ_h^{in} .*

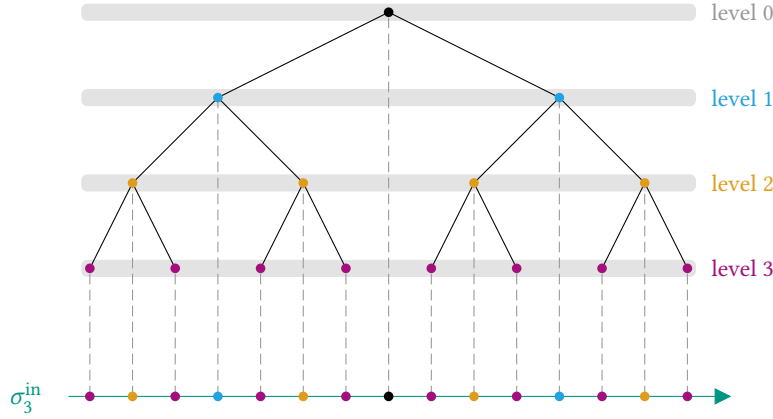


Figure 2.8: Shows the graph T_3 with its in-order vertex ordering σ_3^{in} . The vertices are colored according to their level in the tree. Illustrates Observation 2.15, i.e., that for two consecutive leaves, there is exactly one other vertex between them in the vertex ordering σ_3^{in} .

Let v be a vertex of T_h and let T_v denote the rooted subtree induced by v and its descendants using vertex v as its root. The statement of Observation 2.13 means that if a vertex x is not in $V(T_v)$, there are two cases. If x is to the left of v in σ , then x is to the left of all vertices in $V(T_v)$, or similarly, if x is to the right of v , then x is to the right of all vertices in $V(T_v)$.

We observe that, starting from σ_{h+1}^{in} , deleting the leaves of T_{h+1} yields the vertex ordering σ_h^{in} .

Observation 2.14: The vertex ordering σ_h^{in} can be obtained from σ_{h+1}^{in} by restricting the vertices to $V(T_h)$.

This is true, because up to descending to the leaves, the in-order traversal still processes the other vertices in the same order in the larger graph. We further make an observation how the leaves are placed in the in-order.

Observation 2.15: Let T_h be a complete binary tree of height $h \in \mathbb{N}_0$. Counting from the left in σ_h^{in} , there is exactly one (non-leaf) vertex in T_h that is in σ_h^{in} between the i -th leaf and the $(i + 1)$ -th leaf for every $i \in [2^h - 1]$.

For a complete binary tree of height 0, this observation is trivial, as there are no leaves in the graph. If we consider two leaves with the same parent, they get their parent as the vertex between them, otherwise they inherit their “between-vertex” to be the “between-vertex” of their respective parents in σ_{r-1}^{in} . See also Figure 2.8 as a proof by picture in the case $h = 3$.

3 Subdivisions of Complete Graphs

We now look at a specific family of graphs: the 1-subdivisions of complete graphs. We show that, although they are sparse, their queue number grows super-linearly. Recall that, for $n \in \mathbb{N}_+$, the graph $K_n^{(1)}$ denotes the graph obtained by subdivision of K_n with paths of length 2.

Therefore, the number of vertices n' of $K_n^{(1)}$ is $n' = n + m$ where m is the number of edges of the complete graph K_n . As every edge of $K_n^{(1)}$ is incident to exactly one subdivision vertex, the number of edges of $K_n^{(1)}$ is $m' = 2m$. We start by showing that the queue number of $K_n^{(1)}$ is in $O(\sqrt{n'})$ and in $\Omega(\sqrt[4]{n'})$. These bounds are obtained mostly by applying lemmas from other papers. We use the result that we know the queue number of complete graphs by Heath and Rosenberg [29]. Furthermore, we use two lemmas from a paper by Dujmović and Wood [20]. Afterwards, we construct a vertex ordering that allows for a queue layout using $O(\sqrt[4]{n'})$ queues and consider the stack number of $K_n^{(1)}$.

Lemma 3.1 (Dujmović and Wood, 2005 [20, Lemma 9]): *Let G be a graph and $G^{(1)}$ its 1-subdivision. If G has queue number q , then the queue number of $G^{(1)}$ is at most $(q + 1)$.*

Lemma 3.2 (Dujmović and Wood, 2005 [20, Lemma 10]): *Let G be a graph and $G^{(1)}$ its 1-subdivision. If $G^{(1)}$ has queue number q , then the queue number of G is at most $q(2q + 1)$.*

As mentioned, we begin with the proof obtained by applying results by Dujmović and Wood [20].

Lemma 3.3: *Consider the 1-subdivision $K_n^{(1)}$ with n' vertices. Then*

$$\text{qn}(K_n^{(1)}) \in \Omega(\sqrt[4]{n'}) \cap O(\sqrt{n'}).$$

Proof. Let $m = \binom{n}{2}$ be the number of edges of K_n . Note that $n' = n + m = n + \binom{n}{2}$, so $n \in \Theta(\sqrt{n'})$ and $n' \in \Theta(n^2)$.

For the upper bound we use that $K_n^{(1)}$ is a subdivision of K_n which allows us to apply Lemma 3.1 and bound the queue number of $K_n^{(1)}$ using the queue number of K_n (Lemma 2.10), arriving at the desired asymptotic behavior.

$$\begin{aligned} \text{qn}(K_n^{(1)}) &\stackrel{3.1}{\leq} 1 + \text{qn}(K_n) \\ &\stackrel{2.10}{=} 1 + \left\lfloor \frac{n}{2} \right\rfloor \in O(n) = O(\sqrt{n'}) \end{aligned}$$

For the lower bound we use that $K_n^{(1)}$ is a subdivision of K_n which allows us to apply Lemma 3.2 and bound the queue number of K_n using the queue number of $K_n^{(1)}$.

$$\begin{aligned} \text{qn}(K_n) &\stackrel{3.2}{\leq} \text{qn}(K_n^{(1)}) \cdot (2 \text{qn}(K_n^{(1)}) + 1) \\ &\leq \text{qn}(K_n^{(1)}) \cdot (3 \text{qn}(K_n^{(1)})) \in O(\text{qn}(K_n^{(1)})^2) \end{aligned}$$

Hence, we conclude $\text{qn}(K_n^{(1)}) \in \Omega(\sqrt{\text{qn}(K_n)}) \stackrel{\text{Lemma 2.10}}{=} \Omega(\sqrt{n}) = \Omega(\sqrt[4]{n'})$. $\square$

This result is interesting because the converse result for the stack number is not obtained as easily. In the proof of Lemma 3.3, we use the result by Dujmović and Wood that the queue number of a graph is bounded by the queue number of the 1-subdivision of the same graph [20]. This result is implied by the result that the queue number of a graph is bounded by the queue number of every subdivision with at most one subdivision vertex per edge [20, Lemma 11]. Conversely, Blankenship and Oporowski conjectured that the stack number of a graph is bounded by the stack number of every subdivision with at most one subdivision vertex per edge [4]. The conjecture can be viewed in a paper by Dujmović and Wood [20]. Dujmović and Wood thoroughly discuss subdivisions of graphs and the relation of their stack number and queue number to the stack number and queue number of the original graph, respectively. They prove that the conjecture being answered in the positive implies that the stack number is bounded by the queue number [20, Theorem 10]. As this bound has been proven to not exist [15], we know that the conjecture does not hold either.

However, it is known that the stack number of $K_n^{(1)}$ is not constant [4, Theorem 3.1 | 22, Proof of Theorem 1 | 5, Proposition 6.35]. At the same time, we are able to observe that n is an upper bound for the stack number again, as the branch vertices are a vertex cover of $K_n^{(1)}$. The bound is then obtained by Lemma 2.12.

Observation 3.4: Let $n \in \mathbb{N}_+$ and let n' be the number of vertices of $K_n^{(1)}$. The stack number of $K_n^{(1)}$ is in $O(n) = O(\sqrt{n'})$.

We are able to achieve an improved upper bound by applying a strategy briefly described by Eppstein [21, Discussion]. We assign the branch vertices into blocks of $\sqrt{n}$ vertices each which have their pool of adjacent subdivision vertices next to the block in the order, and then, assign edges blockwise to stacks. Simultaneously, we slightly modify the strategy to obtain a queue layout for $K_n^{(1)}$ using an equal number of queues.

Lemma 3.5: Let $n \in \mathbb{N}_+$ and let n' be the number of vertices of $K_n^{(1)}$. Both the stack number of $K_n^{(1)}$ and the queue number of $K_n^{(1)}$ are in $O(\sqrt{n}) = O(\sqrt[4]{n'})$.

Proof. Without loss of generality, assume $n = k^2$ for some $k \in \mathbb{N}_+$, otherwise consider the supergraph $K_{k^2}^{(1)}$ for $k = \lceil \sqrt{n} \rceil$. Assume that the set of branch vertices of $K_n^{(1)}$ is the set $[n]$. We group the branch vertices of $K_n^{(1)}$ in k groups of k vertices each. We define for every $i \in [k]$ the vertex set $V_i = \{(i-1) \cdot k + j \mid j \in [k]\}$. The subdivision vertices are collected in k blocks according to the group of their first neighbor. That is, the subdivision vertex with neighbors $u, v \in [n]$ is in the block B_i for $i = \lceil \frac{\min(u,v)}{k} \rceil \in [k]$.

We create a vertex ordering σ_s for the stack layout and a vertex ordering σ_q for the queue layout. We use $\sigma_\square$ for statements that hold for both orderings σ_s and σ_q . We primarily order vertices by their groups and blocks in both vertex orderings. More precisely, we order them alternately $V_1, B_1, V_2, B_2, \dots, V_k, B_k$, i.e., it holds that $V_1 <_{\sigma_\square} B_1 <_{\sigma_\square} V_2 <_{\sigma_\square} B_2 <_{\sigma_\square} \dots <_{\sigma_\square} V_k <_{\sigma_\square} B_k$. Within each group V_i , the vertices are ordered by their given order of the natural numbers, i.e., restricted to the branch vertices, both σ_s and σ_q are just the ascending order of the set $[n]$ with the usual greater relation $<$ of the natural numbers. Within each block B_i , we order the subdivision vertices by the maximum of the numerical values of their neighbors, descending for σ_s and ascending for σ_q . E.g., within the block B_1 , the subdivision vertex with neighbors $\{1, 7\}$ would be to the left of the subdivision vertex with neighbors $\{2, 5\}$ in the vertex ordering σ_s . See also Figure 3.1.

We use $2k$ sets $S_1, \dots, S_{2k}$. Each of the sets represents a stack or a queue in the respective layout with their respective vertex orderings σ_s and σ_q . The assignment of edges to stacks or queues is, however, the same. We assign edges to sets blockwise. Let $i \in [k]$ and consider the block B_i . We partition the edges with an endpoint in V_i into k stars with center vertices in V_i . The edges of the star with center vertex $(i-1) \cdot k + j$ are assigned to the set S_j , for each $j \in [k]$. All edges with an endpoint in B_i but no endpoint in V_i are assigned to the set S_{k+i} . See Figures 3.2a and 3.2b for an impression of the chosen assignment, and to visually confirm that it is indeed a stack layout or queue layout, respectively.

It remains to show that each set contains no crossing (nesting) edges in the vertex ordering σ_s (σ_q). We start by considering a set S_{k+i} for each $i \in [k]$. If there is at most one edge in the set S_{k+i} , we are done, otherwise let e and e' be two different edges in the set S_{k+i} . By assignment, the edges $e = uv, e' = u'v'$ have incident vertices $v, v' \notin V_i \cup B_i$ and incident vertices $u, u' \in B_i$, respectively. By assignment of the subdivision vertices to the block corresponding to the group of their minimum neighbor, the subdivision vertices u and u' each have no neighbor $x \in [n]$ such that $x <_{\sigma_\square} V_i$. Thus, together with $v, v' \notin V_i$, we have $B_i <_{\sigma_\square} \{v, v'\}$. If $u = u'$ or $v = v'$, the edges e and e' trivially do not cross (nest). We assume without loss of generality that $u \neq u'$ and $v <_{\sigma_\square} v'$.

We consider the case $\sigma_\square = \sigma_s$. As the subdivision vertices within the block B_i are ordered reversely to their maximum neighbor in σ_s and the maximum neighbors of u, u' are v, v' , respectively, ordered $v <_{\sigma_s} v'$, we have $u' <_{\sigma_s} u$. Combined with the fact that the vertex v is to the right of vertices in block B_i , it follows that the endpoints of the edges e and e' behave like the following: $u' <_{\sigma_s} u <_{\sigma_s} v <_{\sigma_s} v'$, i.e., the edges e and e' nest and do not cross.

We consider the case $\sigma_\square = \sigma_q$. As the subdivision vertices within the block B_i are ordered by their maximum neighbor in σ_q and the maximum neighbors of u, u' are v, v' , respectively, ordered $v <_{\sigma_q} v'$, we have $u <_{\sigma_q} u'$. Combined with the fact that the vertex v is to the right of vertices in block B_i , it follows that the endpoints of the edges e and e' behave like the following: $u <_{\sigma_q} u' <_{\sigma_q} v <_{\sigma_q} v'$, i.e., the edges e and e' cross and do not nest.

We now consider the set S_i for each $i \in [k]$. By assignment, for each edge $e \in S_i$, there is a $j \in [k]$ such that both endpoints are in $V_j \cup B_j$. As observed when creating the edge partition, the edges in $S_i \cap \binom{V_j \cup B_j}{2}$ are a star, and hence, they neither nest nor cross. Since the groups and blocks are ordered $V_1 \cup B_1 <_{\sigma_\square} \dots <_{\sigma_\square} V_k \cup B_k$, edges of different stars are disjoint in $\sigma_\square$, and thus, they neither cross nor nest. Therefore, the set S_i contains no crossing (nesting) edges.

Hence, we have a $2k$ -stack layout for $K_n^{(1)}$ with the vertex ordering σ_s , and we conclude $\text{sn}(K_n^{(1)}) \in O(\sqrt{n})$. Similarly, we have a $2k$ -queue layout for $K_n^{(1)}$ with the vertex ordering σ_q , and we conclude $\text{qn}(K_n^{(1)}) \in O(\sqrt{n})$. Recall that $n' = n + \binom{n}{2} \in \Theta(n^2)$, and conversely, $n \in \Theta(\sqrt{n'})$ for the equality $O(\sqrt{n}) = O(\sqrt[4]{n'})$. $\square$

Together with Lemma 3.3 and the unboundedness of the stack number of $K_n^{(1)}$ [22, Proof of Theorem 1 | 5, Proposition 6.35], we can summarize our results in the following theorem.

Theorem 3.6: *Let $n \in \mathbb{N}_+$. The queue number of $K_n^{(1)}$ is in $\Theta(\sqrt{n})$. The stack number of $K_n^{(1)}$ is in $\omega(1) \cap O(\sqrt{n})$.*

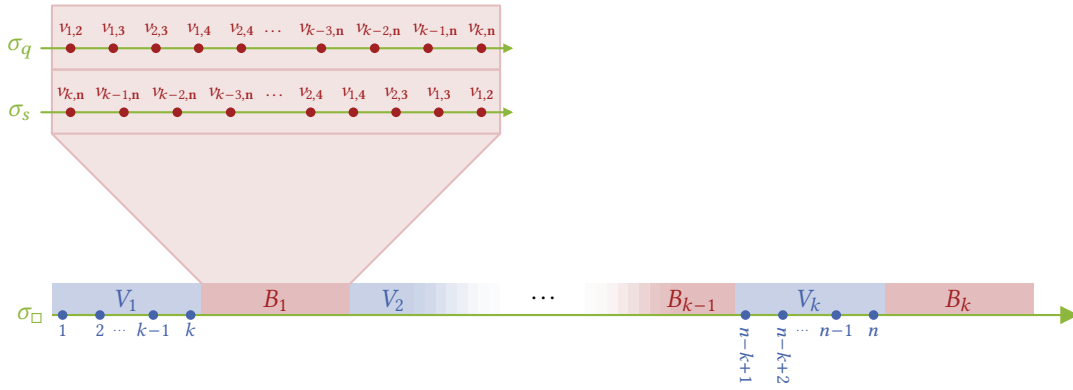


Figure 3.1: Illustrates the vertex ordering $\sigma_{\square}$, for $\square \in \{s, q\}$, of $K_{k^2}^{(1)}$ considered in the proof of Lemma 3.5, for some $k \in \mathbb{N}_+$, $k \geq 4$. The groups V_i are colored blue and the blocks B_i are colored red. The subdivision vertex with neighbors i and j , with $i < j$, is labeled $v_{i,j}$. The number j is printed in bold to emphasize that the vertices are ordered by the neighbor j . The first and the last vertices of the order of the vertices in the block B_1 is depicted in a closer-up for each of the vertex orderings σ_s and σ_q .

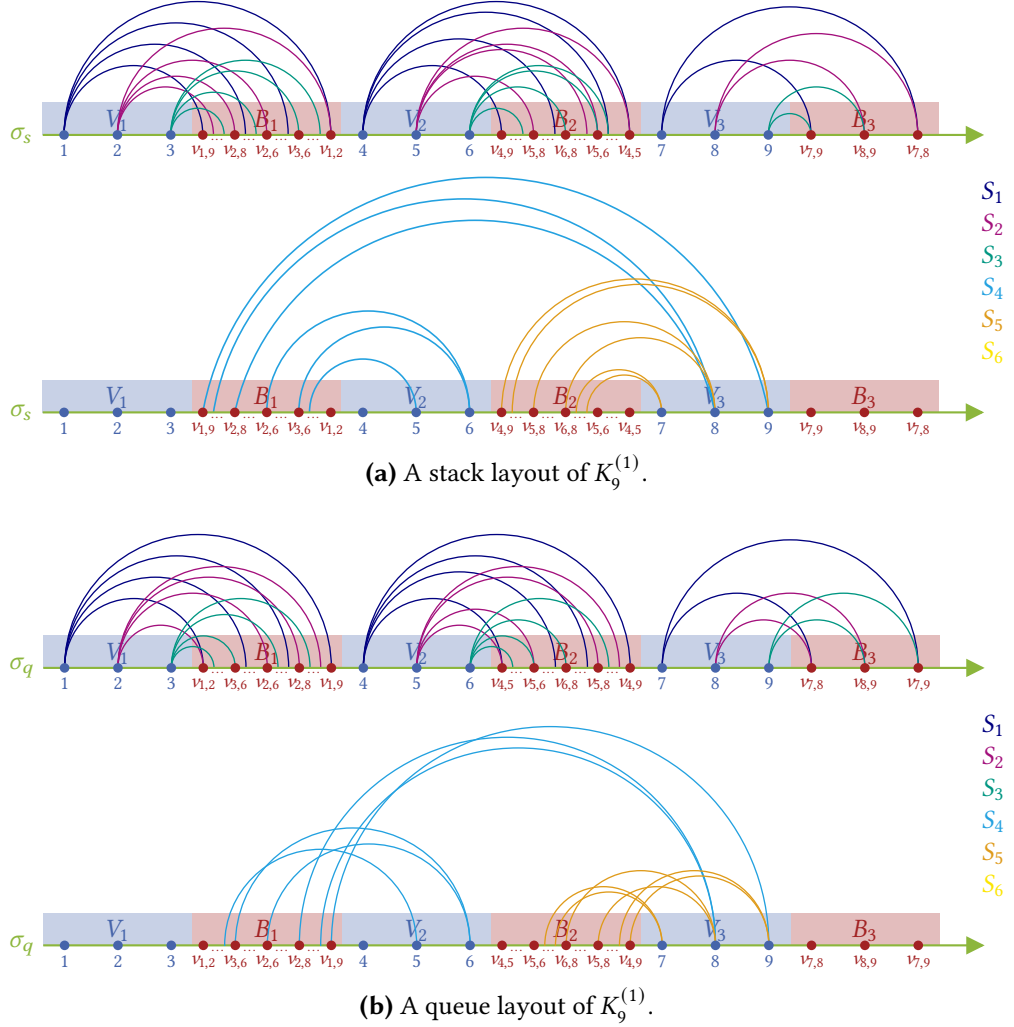


Figure 3.2: Illustrates an excerpt of the resulting stack layout (a) (queue layout (b)) as described in the proof of Lemma 3.5 for $K_9^{(1)}$. The sets that contain the stars are depicted in the upper part and the sets that contain the edges per block B_i are depicted in the lower part of the figure. On the right side, the sets indicate the color of their contained edges. Observe that the set S_6 does not contain any edges. The subdivision vertex with neighbors $i < j$ is labeled $v_{i,j}$. Note that the figure does not contain all edges and not all subdivision vertices. Edges that do not end at a vertex in the illustration represent that there is some edge between a vertex and some vertex in the block where the edge ends. This primarily serves the purpose of providing an improved impression without worsening the clarity of the figure even further.

4 Some Interesting Graphs

In this chapter, we work towards a family of graphs with unbounded treewidth and bounded stack number. It is interesting since it might have large queue number. There are subgraphs which contain most of the edges and have bounded treewidth. However, we may not use arbitrary vertex orderings that allow bounded stack number for these subgraphs since we want to add the remaining edges while maintaining bounded stack number. We begin by introducing new concepts that provide vertex orderings which allow this while using only a constant number of additional stacks.

4.1 Tree-Based Graphs

In this section, we introduce a rather broad class of graphs and determine their stack number when given some additional constraints. Some interesting considerations for the graphs in Section 4.2 do not require explicit graphs. They rather require a relation of levels and where neighbors of vertices are located. Therefore, we introduce a more general concept and prove certain aspects just using selected requirements, simplifying the scope of the proofs.

Let $T = (V_T, E_T)$ be a rooted tree with a root vertex $r \in V_T$ and $T^* = (V_T, E_T^*)$ the transitive closure of the rooted tree T with root r . A *tree-based* graph $G = (V, E)$ is a subgraph of T^* . We say that T is the *underlying rooted tree* of G . In this thesis, the underlying rooted tree T of the tree-based graph G is usually hidden behind the fact that we call it a tree-based graph, and we then access the properties that are inherited from the rooted tree T . Most importantly, this allows us to access the ancestor-descendant-relationship inherited from the underlying rooted tree. We consider the vertices of G as a partition in the BFS-levels of the rooted tree T starting at its root vertex r . We refer to the set of vertices in level i by $V_i(G)$. If $a < b$, we consider level a to be *smaller* than level b and level b to be *larger* than level a . We sometimes refer to the levels just by their integer indices, especially when comparing them or when giving constraints on the indices in relation to another. Often, it is relevant to consider neighboring vertices that are on smaller (larger) levels, and we define the *neighborhood in smaller levels* (*neighborhood in larger levels*) of a vertex to be the set of neighbors such that each vertex is in a smaller (larger) level than the level of the vertex itself. For a vertex $v \in V$ in level i , we denote its neighborhood in smaller (larger) levels by $N^-(v) = N_G^-(v) = N_G(v) \cap \bigcup_{j=1}^{i-1} V_j(G)$ ($N^+(v) = N_G^+(v) = N_G(v) \cap \bigcup_{j=i+1}^l V_j(G)$).

We adopt the notion of descendants from trees and define that a vertex v is a *descendant* of a vertex u in G if v is a descendant of u in the underlying rooted tree T . We define $D(v)$ to be the set of v together with its descendants in G . We refer to $D(v)$ as the *descendants including v* instead of the longer descendants of v together with vertex v . Note that, for a vertex $v \in V$, every vertex in the neighborhood in larger levels of v , is a descendant of v .

For an edge $e \in E$, its *parent* is the endpoint of e that is on the smaller level. See Figure 4.1 for an example of a tree-based graph with indications of the introduced concepts.

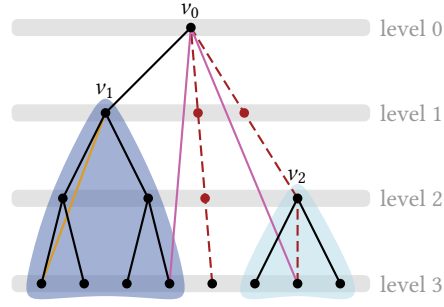


Figure 4.1: An example of a tree-based graph with four levels. The red vertices and the red dashed edges are not present in the graph but signalize that they are part of a tree that is used as underlying tree for this tree-based graph. The black edges are part of the underlying rooted tree. The orange and pink edges are part of the transitive closure of the underlying rooted tree. Their different colors are primarily intended to improve clarity. The light gray bars indicate the levels. The vertices covered by the descendants including v_1 and the descendants including v_2 are highlighted in blue and light blue respectively.

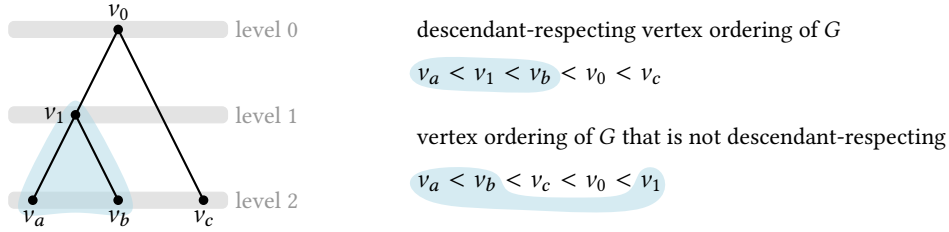


Figure 4.2: An example of a tree-based graph G with a vertex ordering that is descendant-respecting and a vertex ordering that is not. The descendants including v_1 is highlighted in light blue in the graph as well as in each of the vertex orderings.

We say that a vertex ordering σ of G is *descendant-respecting*, if, for every vertex $v \in V(G)$, the descendants including v appears as a contiguous subsequence of σ as a sequence of vertices. We say that the descendants including v appear as a *block* in σ . This means, that there is no vertex $x \in V(G) \setminus D(v)$ such that there are vertices $a, b \in D(v)$ that are ordered $a <_\sigma x <_\sigma b$. See Figure 4.2 for a visual intuition of what the concept means. In the common case that the underlying rooted tree T is a binary tree, the in-order of T is a good example of a descendant respecting vertex ordering of G . In fact, the concept is designed as a generalization of the in-order on a binary tree.

If a vertex ordering is descendant-respecting, we can make the following observation that is useful for constructing stack layouts of tree-based graphs.

Lemma 4.1: *Let G be a tree-based graph and σ be a descendant-respecting vertex ordering of G . Let $e, e' \in E(G)$ be two edges with parents that are on the same level. Then the edges e and e' neither cross nor nest in the vertex ordering σ .*

The situation of the lemma, in the case that the parents of the two edges are different, is depicted in Figure 4.3.

Proof. Let $e = ab$ and $e' = uv$ be two edges with parents on the same level. Without loss of generality, the vertices a and u are the parents, and they are on level i . In case that $a = u$, the statement trivial. If $a \neq u$, then the vertex b is a descendant of the vertex a , and the

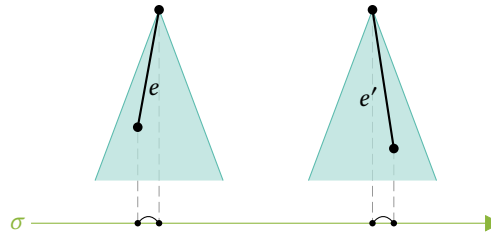


Figure 4.3: Illustrates the situation that the two edges e and e' have different parents, although their parents are on the same level. Observe that the edges e and e' do not cross.

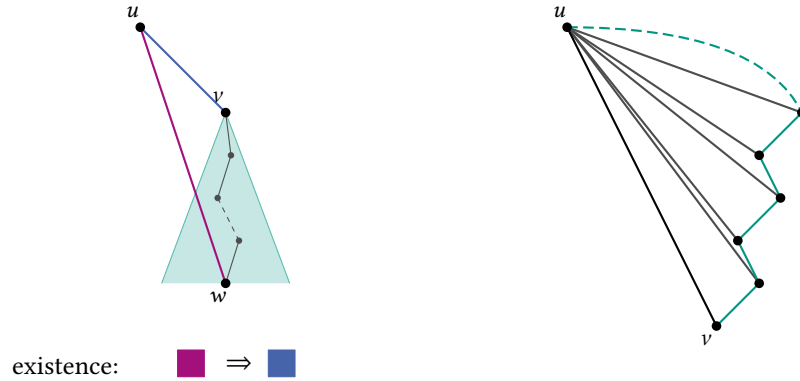


Figure 4.4: Illustrates in two ways what it means that a tree-based graph G is ancestor-preserving.

Left: The way, the property is defined. The green area represents the descendants of v .

Right: The case that $T \subseteq G$. The vertex v is a descendant of u with $uv \in E(G)$ and therefore, the vertex u is incident to every vertex in the u - v -path in T .

vertex v is a descendant of the vertex u . As the vertex ordering σ is descendant-respecting, the descendants of vertices a and u , respectively, appear in blocks in σ . By definition of a rooted tree being acyclic, vertices a and b being on the same level implies that they do not share any descendants. Therefore, the descendants of a are all to the left (or right) of the descendants of u . Hence, the edges ab and uv are disjoint in σ , and they neither cross nor nest. The situation looks like in Figure 4.3. $\square$

We say that a vertex w in a tree-based graph G is *ancestor-preserving* if for every ancestor u of w that is also a neighbor of w , we have that every vertex v that is both an ancestor of w and a descendant of u , is a neighbor of u . We say that the tree-based graph G is *ancestor-preserving* if all of its vertices are ancestor-preserving. Alternatively, we can see it from the point of view of the ancestors: for every vertex u in level i and vertex v in a level larger than i , we have the following condition: If there is some vertex w that is a descendant of v and w is a neighbor of vertex u , then $uv \in E(G)$. The Figure 4.4 helps to digest the property.

If the underlying rooted tree T is a subgraph of G , we can view the property in another way: If vertex u is an ancestor of vertex v and $uv \in E(G)$, then u is adjacent to all vertices on the unique u - v -path in T . One can also prove that in the case $T \subseteq G$, being ancestor-preserving is equivalent to the neighborhood in smaller levels inducing a clique for every vertex.

If we know that a tree-based graph is ancestor-preserving, we obtain an upper bound of the stack number which is one more than the maximum number of neighbors in smaller levels.

Lemma 4.2: *Let G be an ancestor-preserving tree-based graph and σ a descendant-respecting vertex ordering of G such that every vertex has at most k neighbors in smaller levels for $k \in \mathbb{N}_+$. Then the stack number of G with respect to σ is at most $k + 1$.*

Proof. We show that there is a way to use $k + 1$ stacks for a stack layout of G with vertex ordering σ . We use a family of auxiliary graphs. For $i \in \mathbb{N}_0$, we define the tree-based graph $H_{(i)}$ to be the subgraph of G which contains all vertices and the edges that have their parent in a level that is at most i . See Figure 4.5 for an illustrative example. We prove the result by induction on $i \in \mathbb{N}_0$ of the following claim: There is a $(k + 1)$ -stack layout of $H_{(i)}$ with vertex ordering σ such that all edges with the same parent are assigned to the same stack.

Note that σ being descendant-respecting allows us to apply Lemma 4.1. Therefore, we know that edges with parents on the same level do not cross.

For the induction base, we consider level $i = 0$. We assign all edges with parents in level 0 to the same arbitrary stack. The edges with parents in level 0 do not cross. Therefore, we have a valid stack layout and all edges with the same parent are assigned to the same stack.

For the induction step, consider level $i > 0$. By induction, there is a partition of the edges with parents in levels smaller than i with the given constraints. If there are no edges with parent in level i , we are done.

As edges with parents on the same level do not cross, we only need to consider whether edges with parent in level i cross an edge in $H_{(i-1)}$, and we may consider the vertices in level i independently. Let v be a vertex in level i . By the preconditions, the set $N^-(v)$ has size at most k .

The edges of v to the vertices $N^-(v)$ are assigned to at most k different stacks, and there is some stack that none of the edges with parent in $N^-(v)$ are assigned to, as all edges with the same parent are assigned to the same stack. Let S be a stack that none of the edges to $N^-(v)$ are assigned to, and assign all edges with parent v to stack S . It remains to prove that no edge with parent v crosses an edge of $H_{(i-1)}$ that has been assigned to stack S . The situation in the graph, as well as the assignment of the edges to stacks, can be viewed in Figure 4.7.

If the edges with parent v do not cross any edges in $H_{(i-1)}$ regardless of assigned stacks, we are good. If there is some edge e in $H_{(i-1)}$ that crosses an edge with parent v , we show that e is assigned to a stack other than S .

The other endpoint of an edge with parent v is a descendant of v . As the vertex ordering σ is descendant-respecting, the descendants including v are laid out contiguously. Therefore, every edge that crosses an edge with parent v , has an endpoint that is a descendant of v . The fact that edges with both endpoints outside the descendants of v do not have to be considered is illustrated by Figure 4.6. Let e be such an edge of $H_{(i-1)}$ with a parent other than v but an endpoint that is a descendant of v . However, as G is ancestor-preserving, the parent of e is in $N^-(v)$. By induction, the edge e is in the same stack as all edges with the same parent. By the choice of the stack S , this stack is not S , and thus, the assignment does not cause crossings in S , and we have a $(k + 1)$ -stack layout of $H_{(i)}$ with vertex ordering σ .

Hence, $\text{sn}_\sigma(G) \leq k + 1$. □

We show that, in an ancestor-preserving tree-based graph, the neighbors in smaller levels induce a clique (see Figure 4.8). This can then be used to bound the number of vertices in smaller levels and gives bounds on the stack number of ancestor-preserving tree-based graphs.

Lemma 4.3: *Let G be an ancestor-preserving tree-based graph and v be a vertex in level k . Then the neighborhood in smaller levels of v is empty or induces a clique.*

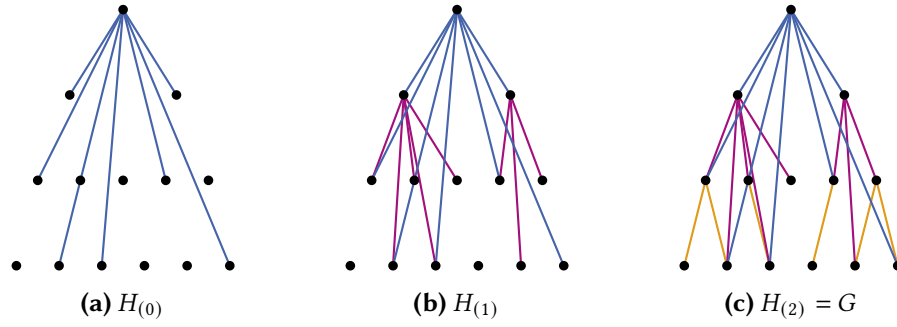


Figure 4.5: Illustrates how the graphs on which the induction, in the proof of Lemma 4.2, works, look like for an example tree-based graph G , which is the right figure. The different colors of edges represent the level of their parent.

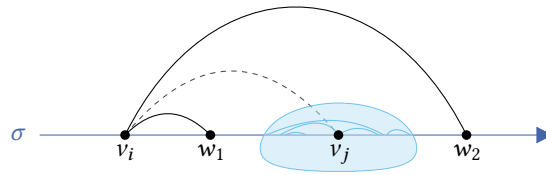


Figure 4.6: We consider an ancestor-preserving graph G with a descendant-respecting vertex ordering σ and vertices v_i and v_j where v_i is no neighbor of v_j and v_i is not a descendant of v_j . Then it holds that edges incident to v_i do not cross edges with both endpoints in the descendants including v_j . This is illustrated for some neighbors w_1 or w_2 but neither edge $v_i w_1$ nor $v_i w_2$ crosses any of the edges induced by the descendants including v_j . The descendants including v_j are marked in light blue as well as the edges induced by them.

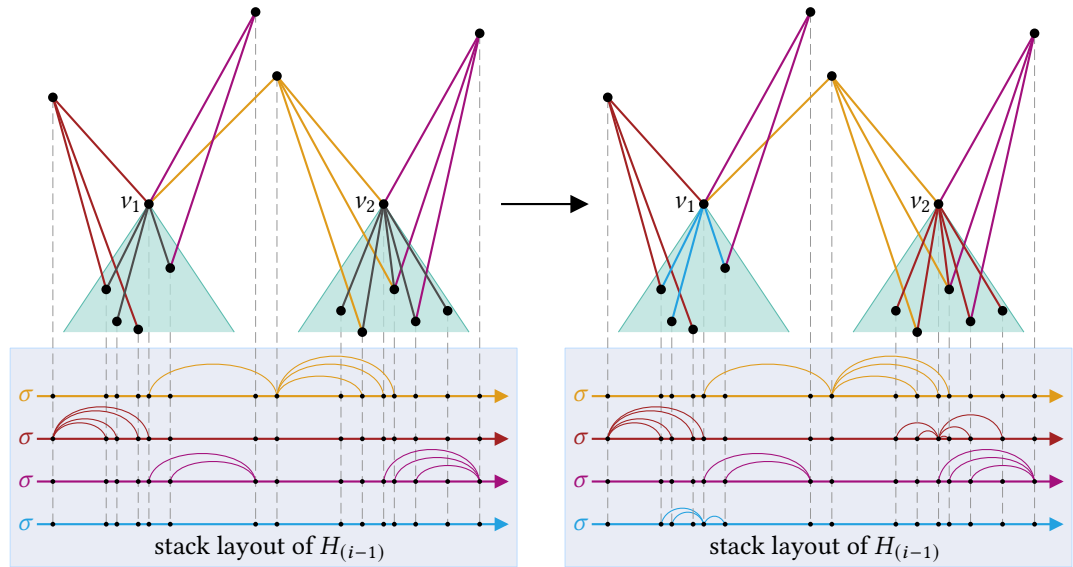


Figure 4.7: Illustration for the situation in the proof of Lemma 4.2. The vertices v_1 and v_2 are vertices in level i . The left part shows the situation before assigning the new edges in $H_{(i)}$ to some stacks. The gray edges are the new ones that are in $H_{(i)}$ but not in $H_{(i-1)}$. Each color represents a stack. The right part shows a possible assignment of the new edges to stacks as constructed in the proof.

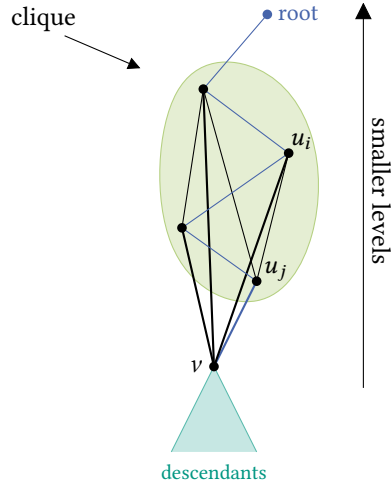


Figure 4.8: Illustrates that the neighborhood in smaller levels induces a clique in ancestor-preserving tree-based graphs. The green area represents the descendants of the vertex v and that v might have some neighbors there as well. It includes the vertices mentioned in the proof of Lemma 4.3 as well as the path to the root in the underlying tree in blue.

Proof. By definition of the transitive closure, no two vertices of $N^-(v)$ are on the same level. Assume without loss of generality that $N^-(v)$ contains at least two vertices. Let $u_i, u_j \in N^-(v)$ be two vertices in levels i and j , respectively. Assume without loss of generality that $i < j$. See Figure 4.8 for the relative positions of the vertices. Since u_i and u_j are on the unique path in the underlying tree between its root and v , we have that v is a descendant of u_j which in turn is a descendant of u_i . Together with the fact that G is ancestor-preserving and with $u_i v \in E(G)$, we get that the edge $u_i u_j$ is present in the graph G . As the vertices u_i and u_j were chosen arbitrarily, the set $N^-(v)$ induces a clique in G . $\square$

Now, we combine our results for ancestor-preserving tree-based graphs. We show that the stack number is linear in the clique number for those graphs.

Corollary 4.4: *Let G be an ancestor-preserving tree-based graph. Then the stack number of G is at most $\omega(G)$ and an $\omega(G)$ -stack layout may be obtained using an arbitrary descendant-respecting vertex ordering of G . That is, the stack number of G is in $\Theta(\omega(G))$.*

Proof. The lower bound is given by Corollary 2.9. Observe that there is some descendant-respecting vertex ordering σ of G , e.g., by performing a depth-first-search on the underlying rooted tree and considering the order of discovering the vertices. So, we use some arbitrary descendant-respecting vertex ordering σ of G . Let v be a vertex of G . As G is ancestor-preserving, by Lemma 4.3, there are no neighbors of v in smaller levels or the neighbors of v in smaller levels induce a clique in G . Therefore, the number of vertices in $N^-(v) \cup \{v\}$ is at most the clique number of G , i.e., $|N^-(v)| \leq \omega(G) - 1$ since $v \notin N^-(v)$.

We may, thus, apply Lemma 4.2 with $k = \omega(G) - 1$ and the vertex ordering σ . This gives that the stack number of G with respect to σ is at most $k + 1 = (\omega(G) - 1) + 1 = \omega(G)$. Hence, we have the upper bound of $\omega(G)$. $\square$

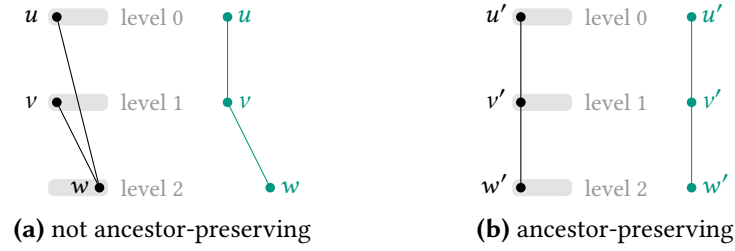


Figure 4.9: Illustrates that being ancestor-preserving is not a property of a graph per se. It depends on the underlying rooted tree. The graphs are black, their underlying rooted trees are green. Both graphs are paths of length 2, and thus, are clearly isomorphic (e.g. with the map $u \mapsto u', v \mapsto w', w \mapsto v'$). The tree-based graph on the left, however is not ancestor-preserving as vertex u is in level 0, vertex v is in level $1 > 0$ and vertex w is a descendant of v . Further, the graph contains the edge uw . If the tree-based graph was ancestor-preserving, the graph would have to contain the edge uv which it does not. Hence, the tree-based graph on the left is not ancestor-preserving with its underlying rooted tree. However, the tree-based and isomorphic graph with a different underlying rooted tree on the right is clearly ancestor-preserving as there are no transitive edges. Therefore, there are no edges that would satisfy the condition of having endpoints x and z where x is a vertex in level i , there is a vertex y in level $j > i$ such that z is a descendant of y .

Given that we determined the stack number of ancestor-preserving tree-based graphs in terms of the clique number, we would like to be able to apply the result as often as possible. Therefore, it is helpful to know that an induced subgraph of a tree-based ancestor-preserving graph is again a tree-based ancestor-preserving graph when considering the same underlying rooted tree.

Lemma 4.5: *Let G be an ancestor-preserving tree-based graph and T its underlying rooted tree. Then every induced subgraph $H \subseteq_{\text{ind}} G$ is an ancestor-preserving tree-based graph with respect to the underlying rooted tree T .*

This statement is similar to saying that the graph class of the ancestor-preserving tree-based graphs is hereditary, i.e., closed under taking induced subgraphs. However, the same graph can be a tree-based ancestor-preserving graph with respect to one underlying rooted tree and a tree-based graph that is not ancestor-preserving with respect to a different underlying rooted tree. See Figure 4.9 for an example of such a graph. This behavior is similar to the planarity of a drawing of a graph, as a graph can have non-planar drawings even if the graph itself is planar, i.e., admits planar drawings. If we however have a planar drawing of a graph G , we obtain a planar drawing of an induced subgraph $H \subseteq_{\text{ind}} G$ by restricting the previous planar drawing to the vertices and edges of H .

Similar to the planarity of a graph, one could obtain a hereditary graph class by defining ancestor-preserving for a general graph. We provide one possible definition, although we do not use it in this thesis any further. We say that a graph G is ancestor-preserving if there is a rooted tree T such that G is an ancestor-preserving tree-based graph with the underlying rooted tree T .

Proof. Recall that every subgraph of a tree-based graph is a tree-based graph. In particular, we know that the graph H is a tree-based graph with respect to the rooted tree T .

Recall, to be ancestor-preserving, every vertex w of the tree-based graph H has to be ancestor-preserving, i.e., for every two ancestors u, v with v a descendant of u , it has to hold that the presence of the edge uw in H implies the presence of the edge uv in H . If there are no such vertices u, v, w in H , the tree-based graph H is trivially ancestor-preserving. Therefore, we assume without loss of generality, that there are vertices $u, v, w \in V(H)$ as described above.

If uw is an edge of H , then uw is an edge of G as well. As H uses the same underlying tree as G , we have that w is a descendant of v which in turn is a descendant of u in G . Since G is an ancestor-preserving tree-based graph, this implies that uv is an edge of G . As H is an induced subgraph of G and contains the vertices u and v , the edge uv is in H . Hence, the graph H is ancestor-preserving. $\square$

4.1.1 A Bound on the Stack Number of Graphs with Bounded Treewidth

In this section, we show that using our approach of ancestor-preserving tree-based graphs, we are actually able to achieve the already known tight upper bound for the stack number of graphs with bounded treewidth. The bound is due to Ganley and Heath [27] and its tightness to Dujmović and Wood [18]. An advantage of our proof is that it allows a variety of different vertex orderings admitting $(k + 1)$ -stack layouts. We use this capability in Section 4.2.3 to augment a chordal graph with some additional edges which drastically increase the treewidth while increasing the stack number by only an additional constant term.

To do this, we show that every chordal graph can be embedded as an ancestor-preserving tree-based graph using some underlying rooted tree.

Lemma 4.6: *Let G be a chordal graph. Then there is a rooted tree T such that G with underlying rooted tree T is an ancestor-preserving tree-based graph.*

Proof. We prove the statement by induction on the number n of vertices in G . The base case of $n = 1$ is trivial, we use $T = G$ with the single vertex as the root. It is trivially ancestor-preserving since there are no ancestor-descendant relationships between distinct vertices. We consider the induction step with $n > 1$. Recall that an ancestor-preserving tree-based graph is not necessarily connected and does not necessarily contain all the vertices of its underlying rooted tree. Without loss of generality, we assume that G is connected, otherwise use induction on each connected component and introduce a new root vertex in the underlying tree, connecting it to the root of the underlying tree of each component (the vertex is *not* added to the graph G). Observe that this maintains being tree-based and ancestor-preserving since no new ancestor-descendant relationships are introduced between vertices of G .

Since G is chordal, by Lemma 2.4, G has a simplicial vertex v . Consider the graph $G' = G - v$. The graph G' is chordal since an induced subgraph of a chordal graph is chordal by Observation 2.6. We apply induction to the graph G' yielding a rooted tree T' with which G' is an ancestor-preserving tree-based graph as an underlying tree.

By definition of v being simplicial and with G being connected, we know that $N_G(v)$ induces a clique. Hence, since the graph G' is a tree-based graph, i.e., a subgraph of the transitive closure of the rooted tree T' , we know that every pair of vertices in $N_G(v)$ is in an ancestor-descendant relationship.

Let $u \in N_G(v)$ be the lowest neighbor of v in T' , i.e., the unique vertex such that every vertex of $N_G(v) \setminus \{u\}$ is an ancestor of u . We use the rooted tree T which is obtained by adding v as a child of u in T' .

Since G' is a tree-based graph with underlying rooted tree T' , we know that G' is a subgraph of the transitive closure of the rooted tree T' . Furthermore, we have that the parent of vertex v in T is a vertex $u \in N_G(v)$ which has all vertices of $N_G(v) \setminus \{u\}$ as ancestors, implying that all neighbors of v are ancestors of v by transitivity. This implies that the graph G is a subgraph of the transitive closure of the rooted tree T . Recall that for a tree-based graph to be ancestor-preserving, every vertex has to be ancestor-preserving. And for a vertex d to be ancestor-preserving, it has to hold that for every neighbor $a \in N_G(d)$ that is an ancestor of d , every vertex between a and d in the ancestor-descendant relationship, is also a neighbor of vertex a . Note that since the tree-based graph G' is ancestor-preserving and with G' being an induced subgraph of G obtained by removing v and v not having any descendants, every vertex of G' is ancestor-preserving in the tree-based graph G . Note that there are no vertices between u and v in the ancestor-descendant-relationship of G . Let $a \in N_G(v) \setminus \{u\}$ be an ancestor of v that is also a neighbor of v . Let further $b \in V(G)$ be an ancestor of v that is a descendant of vertex a in G . Since u is the parent of v in T , b is an ancestor of u or the vertex u itself. In the case $u = b$, we have the edge $ab \in E(G)$ since $N_G(v)$ induces a clique. If $u \neq b$, we have the edge $ab \in E(G)$ since u is ancestor-preserving in G , vertex a is an ancestor of u , and, with $N_G(v)$ inducing a clique, vertex a is a neighbor of u . Both cases imply, that the vertex v is ancestor-preserving in the tree-based graph G . Hence, the graph G with the underlying rooted tree T is an ancestor-preserving tree-based graph. $\square$

Next, we derive the aforementioned upper bound on the stack number of graphs with bounded treewidth.

Corollary 4.7: *Let G be a graph. We have $\text{sn}(G) \leq \text{tw}(G) + 1$.*

Proof. By Lemma 2.1, for every $k \in \mathbb{N}_0$, the fact that we have $\text{tw}(G) = k$ is equivalent to k being the minimal integer such that there exists of a chordal supergraph H with clique number $\omega(H) = k + 1$. Since H is chordal, we may apply Lemma 4.6 to obtain a rooted tree T such that H with the underlying rooted tree T is an ancestor-preserving tree-based graph. By Corollary 4.4, we have that the stack number of H is at most $\omega(H)$. Hence, we conclude $\text{sn}(G) \leq \text{sn}(H) \leq \omega(H) = k + 1 = \text{tw}(G) + 1$. $\square$

4.2 HUDG-inspired graphs

We look at a family of graphs that is inspired by a family of graphs that appeared in a paper by Bläsius, Dohse, Haun, and Merker [7, Chapter 3]. These are graphs that appear when connecting close vertices in the hyperbolic plane. Because we do not need the geometry, we just ignore it and define a similar family of graphs. We aim to modify this family such that its stack number is bounded while its queue number might not be. However, we do not know whether the queue number of the modified family is bounded or not. We iteratively define families of graphs building up on the previous versions.

For each $r \in \mathbb{N}_0$, we define a tree-based graph $H_r = (V, E)$ with $r + 1$ levels. The graph H_r contains a complete binary tree T_r of height r . We label the vertices in level i using 0-1-bit-strings of length i . For a vertex with label l , its left child is the vertex with label $l0$ and its right child is the vertex with label $l1$. If we want to refer to the vertex with label l , we refer to it by level and index within the level. We take $v_{i,x}$, where i is the length of the label l and the index x is the number represented by the binary string in the label l . E.g., the label 01100 corresponds to $v_{5,12}$. Let v be a vertex of H_r with label l in level i . For a larger level $j > i$, the *left central descendant* of v in level j is the vertex with label $l01^{j-i-1}$, and the *right central*

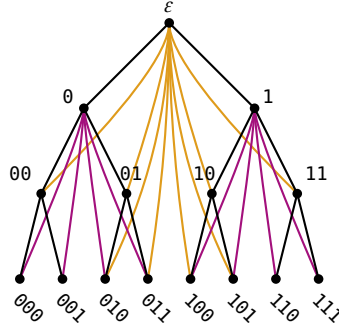


Figure 4.10: Illustration of the constructed graph H_3 with its labels. The edges are colored black if they belong to the underlying complete binary tree. The other edges are colored by the level of their parent; edges with parent in level 0 are orange and edges with parent in level 1 are purple. The vertices 011 and 100 are the left and right central descendant of ϵ in level 3, respectively. At the same time, the vertices 010 and 011 are the left and right central descendant of 01 in level 3, respectively.

descendant of v in level j is the vertex with label $l10^{j-i-1}$. Consider two vertices on the same level with labels l and l' . We say that the vertex with label l is *to the left* of the vertex with label l' if the corresponding binary value of l is lower than that of l' . Conversely, we say that the vertex with label l' is *to the right* of the vertex with label l . Let v be a vertex of H_r with label l in level i and let $\Delta \in [r - i]$. We define the neighborhood of v in level $i + \Delta$: There are equally many among the vertices with prefix $l0$ as with prefix $l1$. The binary values of the labels of the neighbors are contiguous and the central descendants in level $i + \Delta$ are neighbors of v . The number of neighbors in each half of the descendants in level $i + \Delta$ is given by the function $f(\Delta) = \lfloor \sqrt{2^\Delta} \rfloor$. See Figure 4.10 for an illustration of the graph H_3 .

The number of vertices in the graph H_r is $n = \sum_{i=0}^r 2^i = 2^{r+1} - 1$. Therefore, we know that r is in $\Theta(\log(n))$.

If v is a vertex with label l , by our definition, the descendants including v are precisely the vertices whose labels have the prefix l . This allows us to make an observation.

Observation 4.8: Let $v = v_{i,x}$ be a vertex of H_r with $r \geq i \geq 0$. Consider level $j \geq i$. The indices of vertices of the descendants including v range from $2^{j-i} \cdot x$ to $2^{j-i} \cdot (x + 1) - 1$ for vertices in level $j \geq i$.

4.2.1 Clique Number of the Graphs H_r

The stack number of complete graphs is known (see Lemma 2.8) and therefore, the clique number provides a lower bound for the stack number. Thus, in preparation for proving the stack number of H_r in Theorem 4.19, we show that the clique number of H_r is in $\Theta(\log(\log(n))) = \Theta(\log(r))$ (Lemma 4.15).

It is important to know when a vertex and a descendant no longer have common neighbors for levels that are at least as large as some given level. In order to simplify the argumentation, we make the observation that the central descendants are sort of the “worst case” for a descendant of a specific level with respect to having common neighbors with the ancestor.

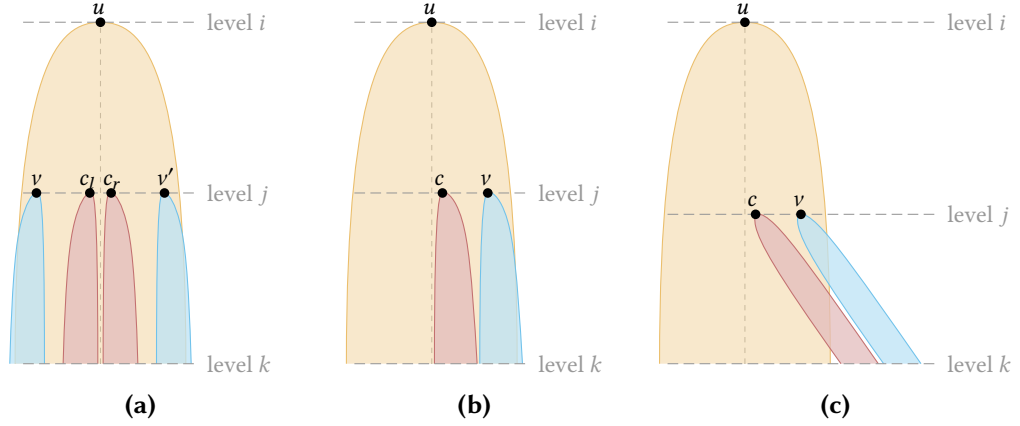


Figure 4.11: Left: Illustrates that the vertices v and v' behave symmetrically, serving as a visual argument why it suffices to consider the case that the label of v has the prefix $l_u 1$ where l_u is the label of u . The label of v' is obtained by flipping the last $j - i$ bits in the label of v . Center + Right: Illustrates the claim of Lemma 4.9. The range of the neighborhood of the vertices is colored as follows: u in orange, c in red and v in blue.

Lemma 4.9: Let $u = v_{i,x_u}$ be a vertex of H_r in level $i \in \mathbb{N}_0$, and let $k > j > i$ be some larger levels. Let $v = v_{j,x_v}$ be a vertex in level j and $c = v_{j,c_r}$ be the right central descendant of u in level j . If the vertices c and u have no common neighbor in level k , then the vertices u and v do not have a common neighbor in level k either.

This result seems quite believable just by looking at Figures 4.11b and 4.11c, as one can see that neighborhoods of vertices on the same level are located somewhat next to each other. This is the key insight of the proof. The task in the proof is rather calculating that this intuition is correct.

Proof. When put into relation to the neighborhood of the vertex u , the neighborhood of the left central descendant is mirrored to the neighborhood of the right central descendant. Therefore, if the right central descendant does not have a common neighbor in level k , then the left central descendant does not have a common neighbor in level k either. Additionally, for every descendant v of u that is to the left of the left central descendant of u , there is a descendant v' of u , that is to the right of the right central descendant of u , and has a symmetric behavior considering sharing neighbors with vertex u . See Figure 4.11a to find such a vertex v' for a descendant v . Thus, we assume without loss of generality, that the vertex v is in the right half of the neighborhood of u in level j , i.e., the label of vertex v has the prefix $l_u 1$ where l_u is the label of vertex u .

As the index x_v is at least c_r , the indices of neighbors of v in level k are greater or equal to those of the neighbors of c in level k . Furthermore, every one of those neighbors has the prefix $l_u 1$ in its label as both c and v have the prefix $l_u 1$ in their labels. Therefore, if u and c have no common neighbors in level k , indices of neighbors of c are too high to be a neighbor of u . Thus, the same holds for neighbors of v in level k . $\square$

As mentioned, we want to know from what level on a vertex and a descendant do not have any common neighbors in larger levels. Therefore, we show that there is a bound which provides that a vertex and its descendants in a specific level do not share neighbors in levels after the bound.

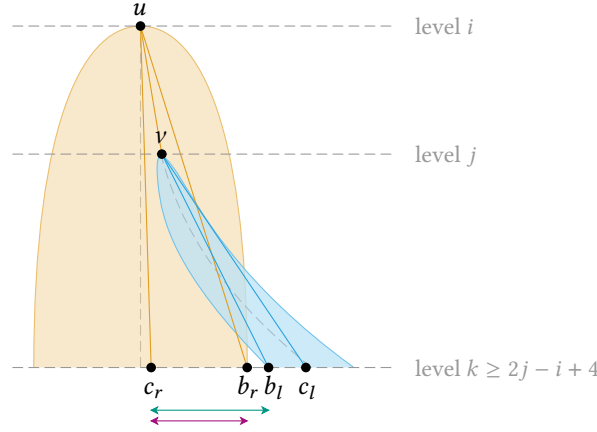


Figure 4.12: Illustrates the locations of the vertices considered in the proof of Lemma 4.10. The orange and the cyan area represent the range of vertices within the levels that are in the neighborhood in larger levels of the vertices u and v , respectively. The orange and cyan edges are present in the graph and colored by their incidence to the vertices u and v . The vertical, gray, dashed line represent the split between the central descendants of the vertices u and v in each of their neighborhoods. The horizontal, gray, dashed lines indicate the levels of the vertices. In the proof, we compare the distance of the indices of the vertices b_l (green arrow) and b_r (purple arrow) to the index of the vertex c_r to show that the vertex b_r is to the left of the vertex b_l .

Lemma 4.10: Let i and j be different levels of H_r , $0 \leq i < j \leq r$. Consider vertices u and v in levels i and j respectively. For each level $k \geq 2j - i + 4$, the vertices u and v have no common neighbor in level k .

Proof. We assume without loss of generality that the vertex v is a descendant of u , otherwise, the statement is trivial as the graph H_r is tree-based. In the case that k is greater than r , the statement is trivially true, as that level does not exist in H_r . We start by proving that the inequality $2^{k-j-1} \geq 2^{\frac{k-i+2}{2}}$ ($\star$) is equivalent to $k \geq 2j - i + 4$.

$$\begin{aligned}
 & 2^{k-j-1} \geq 2^{\frac{k-i+2}{2}} \\
 \Leftrightarrow & \quad | \log_2 \\
 & k - j - 1 \geq \frac{k - i + 2}{2} \\
 \Leftrightarrow & \quad 2k - 2j - 2 \geq k - i + 2 \\
 \Leftrightarrow & \quad k \geq 2j - i + 4
 \end{aligned}$$

The situation of the vertices considered in the proof is depicted in Figure 4.12. Let $x_u, x_v \in \mathbb{N}_0$ be the indices and l_u, l_v be the labels of vertices u, v , respectively. By Lemma 4.9, the right central descendant of u in level j not having a common neighbor with u in level k , implies that no vertex in level j has a common neighbor with u in level k . Thus, we may ignore the actual index of vertex v and assume that v is the right central descendant of u in level j . That is, the label of v is $l_v = l_u 10^{j-i-1}$. Therefore, its index is $x_v = x_u \cdot 2^{j-i} + 2^{j-i-1}$. The label of the right central descendant (c_r) of u in level k is $l_{c_r} = l_u 10^{k-i-1}$ which yields that the corresponding index is $x_{c_r} = x_u \cdot 2^{k-i} + 2^{k-i-1}$. As the vertex u has $a_i = \left\lfloor \sqrt{2^{k-i}} \right\rfloor$ neighbors with prefix $l_u 1$ in

level k , the index of the rightmost neighbor (b_r) of u in level k is $x_{b_r} = x_{c_r} + (a_i - 1)$. Similarly, the label of the left central descendant c_l of v in level k is $l_{c_l} = l_v \mathbf{0}1^{k-j-1} = l_u \mathbf{1}0^{j-i-1} \mathbf{0}1^{k-j-1}$ which yields that the corresponding index is $x_{c_l} = (x_u \cdot 2^{j-i} + 2^{j-i-1}) \cdot 2^{k-j} + (2^{k-j-1} - 1) = x_u \cdot 2^{k-i} + 2^{k-i-1} + 2^{k-j-1} - 1 = x_{c_r} + 2^{k-j-1} - 1$. Again, the vertex v has $a_j = \lfloor \sqrt{2^{k-j}} \rfloor$ neighbors with prefix $l_v \mathbf{0}$ in level k . Thus, the index of the leftmost neighbor (b_l) of v in level k is $x_{b_l} = x_{c_l} - (a_j - 1) = x_{c_r} + 2^{k-j-1} - 1 - a_j + 1 = x_{c_r} + 2^{k-j-1} - a_j$. It suffices to show that the index of the rightmost neighbor of u in level k is strictly less than the index of the leftmost neighbor of v in level k . That is, we aim to show that x_{b_r} is less than x_{b_l} . For improved clarity, we consider the distance to x_{c_r} , i.e., we subtract x_{c_r} from both sides of the inequality.

$$\begin{aligned}
 x_{b_r} - x_{c_r} &= a_i - 1 \\
 &< a_i \\
 &\stackrel{(\star)}{\leq} a_i + (2^{k-j-1} - 2^{\frac{k-i+2}{2}}) \\
 &= a_i + 2^{k-j-1} - 2 \cdot 2^{\frac{k-i}{2}} \\
 &\stackrel{(\diamond)}{\leq} a_i + 2^{k-j-1} - 2 \cdot a_i \\
 &= 2^{k-j-1} - a_i \\
 &\stackrel{(\bullet)}{\leq} 2^{k-j-1} - a_j \\
 &= x_{b_l} - x_{c_r}
 \end{aligned}$$

In the inequations, we use $2^{\frac{k-i}{2}} \geq \lfloor \sqrt{2^{k-i}} \rfloor = a_i$ for $(\diamond)$ and $a_j = \lfloor \sqrt{2^{k-j}} \rfloor \leq \lfloor \sqrt{2^{k-i}} \rfloor = a_i$ for $(\bullet)$.

Hence, the vertices u and v have no common neighbors in level k . $\square$

Using the result that vertices do not have common neighbors in levels that are large enough, we are able to observe that vertices in a clique have certain distances in levels. This then enables us to obtain an upper bound for the clique number of H_r .

We start by intuitively arguing that vertices only have logarithmically many neighbors in smaller levels. This bounds the clique number as every level forms an independent set and a clique has some vertex in a level that is the largest with respect to containing a vertex of the clique; see Figure 4.13a.

Afterwards, we provide a formal proof for the upper bound of the clique number that relies on a result in the paper by Bläsius, Dohse, Haun, and Merker [7 | 6].

We reason, why a vertex v in level k has $O(\log(k))$ neighbors in smaller levels in H_r . Without loss of generality, assume that v has neighbors in at least two different smaller levels. Let $i \in \mathbb{N}_0$ be the smallest level such that v has a neighbor in it and u_i be a neighbor of v in that level. Let j , with $i < j < k$, be some other level in which v has a neighbor, namely the vertex u_j . By Lemma 4.10, we have $k < 2j - i + 4$, and hence, $j > \frac{k+i+4}{2}$. If the level j is not large enough, the vertices u_i and u_j could not have the vertex v as a common neighbor, see Figure 4.13b for a visualization. Therefore, the levels $i+1$ to $\lfloor \frac{k+i+4}{2} \rfloor$ do not contain a neighbor of v , which are about half the levels between levels i and k . This means that neighbors in levels towards the root vertex have exponential distance in levels to k . Hence, we conclude that v has $O(\log(k))$ neighbors in smaller levels in H_r .

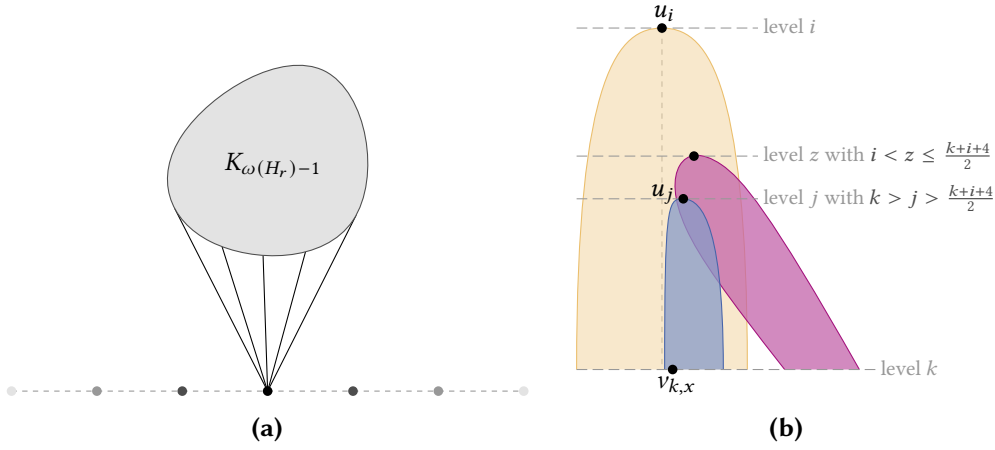


Figure 4.13: Left: Illustrates that a clique in H_r has a vertex in a largest level such that all other vertices are in smaller levels. Note that the vertices within each level form an independent set. Right: Illustrates that if a vertex $v_{k,x}$ has a neighbor u_i in level i and level j is the smallest level that is larger than i such that $v_{k,x}$ has a neighbor u_j in it, that means that level j is a lot larger than level i . The colored areas approximately show the range of indices that the vertices share as neighbors in the different levels. The vertex u_i has an orange neighborhood, the vertex in level z has a purple neighborhood and the vertex u_j has a blue neighborhood.

The following lemma is the core tool in the proof of the upper bound of the clique number of H_r .

Lemma 4.11 (Bläsius, Dohse, Haun and Merker, 2026 [6, Lemma 16 | 7, Lemma 11]): *Let $r \in \mathbb{N}_+$, $c \in \mathbb{N}_0$, and let $S \subseteq [r]$ be a set such that for every three elements $i < j < k$ of S , we have $k - j \leq j - i + c$. Then $|S| < \log_2(r) + c + 2$.*

We use Lemma 4.10 to provide the preconditions to apply Lemma 4.11 to obtain the upper bound of the clique number of H_r .

Lemma 4.12: *The clique number of H_r is in $O(\log(r))$. More precisely, the clique number of H_r is less than $\log_2(r + 1) + 5$.*

Proof. Let $X \subseteq V(H_r)$ a set of vertices that induces a clique in H_r . We consider the set S of levels which contain a vertex in X , i.e., the number i is in S if and only if level i contains a vertex that is also in the set X . As H_r is a level-based graph, the levels are independent sets and therefore, none of the vertices in X are on the same level. Thus, it suffices to consider the size of the set S as the sets S and X are of equal size. Let $i < j < k$ be some levels that appear in S , i.e., there are vertices $u_i, u_j, u_k \in X$ in levels i, j and k , respectively. As X induces a clique, the vertices u_i and u_j have the common neighbor u_k in level k , by Lemma 4.10, it holds that $k < 2j - i + 4$. We subtract j from both sides to get $k - j < j - i + 4$. As all variables have integer values, we get the condition $k - j \leq j - i + 3$. We define the set S' which shifts the elements of S by one into the natural numbers, i.e., $S' = \{x + 1 \mid x \in S\}$. As the numbers in S are between 0 and r , inclusive, the set S' is a subset of $[r + 1]$. For elements $i < j < k$ in S' , it holds $k - j = (k - 1) - (j - 1)$. By the condition on elements in S , this is at most $(j - 1) - (i - 1) + 3 = j - i + 3$. Hence, we are able to apply Lemma 4.11 with $c = 3$. This gives us that $|S'|$ is less than $\log_2(r + 1) + 3 + 2 = \log_2(r + 1) + 5$. As the size of S' is the same as the size of X , we have that the clique number of H_r is less than $\log_2(r + 1) + 5$. Hence, we conclude $\omega(H_r) \in O(\log(r))$. $\square$

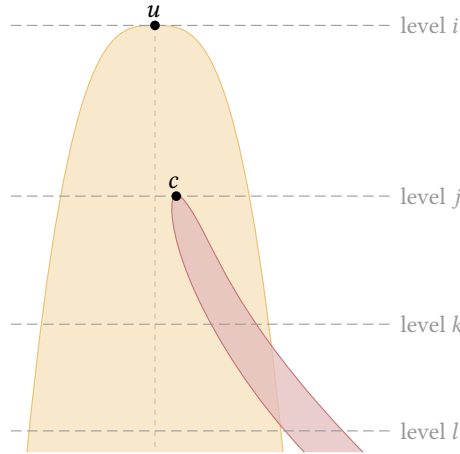


Figure 4.14: Illustrates that the range of neighboring vertices (colored red) of a right central descendant c of a vertex u stays within the range of neighboring vertices (colored orange) of said vertex u for levels that are not too large. However, there is a threshold at which those ranges diverge and have an empty intersection, i.e., the vertices c and u do not have common neighbors in those levels. Here, every neighbor of vertex c in level k is a neighbor of vertex u as well while none of the neighbors of vertex c in level l is a neighbor of vertex v . The last part refers to Lemma 4.10, if l is sufficiently large.

We have proven an upper bound on the clique number of H_r but in order to show that it is tight, we also want to prove a lower bound on the clique number of H_r . That is, we want to find cliques of that size. This is what we do for the rest of the section.

In order to find cliques, it is useful to know that the intersection of a level and the neighborhood of one vertex is contained in the neighborhood of another vertex. If we take a vertex u and the right central descendant c to u in some larger level, we show in Lemma 4.13 that the neighborhood of c is a subset of u 's neighborhood when restricted to a certain range of levels. This behavior of the neighborhood of the right central descendant is illustrated in Figure 4.14. This then allows us to pick vertices such that their neighborhood in larger levels remains within the neighborhood of some ancestors if we pick them such that their level is close enough to level r .

Lemma 4.13: *Let $0 \leq i < j < k \leq r$ be some levels of H_r . Let*

- $u = v_{i,x_u}$ be a vertex of H_r in level i ,
- $v = v_{j,x_v}$ be the right central descendant of u in level j ,
- $k \leq 2j - i - 2$ and w be a vertex in level k .

If vertex w is a neighbor of vertex v , then vertex w is a neighbor of vertex u as well, i.e., $N(v) \cap V_k(H_r) \subseteq N(u) \cap V_k(H_r)$.

Proof. It suffices to show that the indices of the neighbors of v are within those of u . The situation of the vertices considered in the proof is depicted in Figure 4.15. Let l_u be the label of vertex u . As v is the right central descendant of u in level j , we know that the label of v is $l_v = l_u 10^{j-i-1}$ and the index of v is $x_v = x_u \cdot 2^{j-i} + 2^{j-i-1}$. For the lower bound, we observe

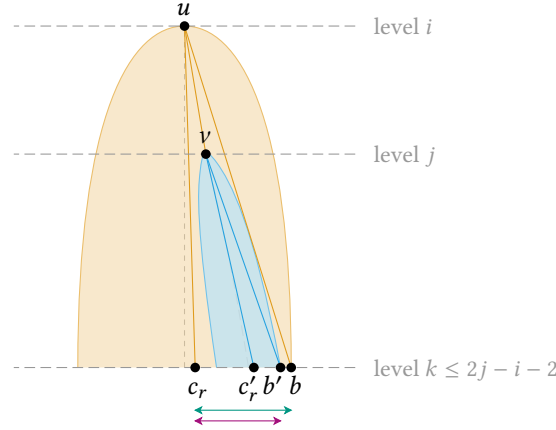


Figure 4.15: Illustrates the locations of the vertices considered in the proof of Lemma 4.13. The orange area and the cyan area represent the range of vertices within the levels that are in the neighborhood in larger levels of the vertices u and v , respectively. The orange and cyan edges are present in the graph and colored by their incidence to the vertices u and v . The vertical, gray, dashed lines represent the split between the central descendants of the vertices u and v in each of their neighborhoods. The horizontal, gray, dashed lines indicate the levels of the vertices. In the proof, we compare the distance of the indices of the vertices b (green arrow) and b' (purple arrow) to the index of the vertex c_r to show that the vertex b is to the right or equal to the vertex b' .

that every neighbor vertex v in level k has the prefix $l_u 1$ and is thus to the right of or equal to the right central descendant of u in level k . This already suffices as the leftmost neighbor of u in level k is to the left of the left descendant of u in level k .

We show that the rightmost neighbor of v in level k has to be to the left of or equal to rightmost neighbor of u in level k . To prove this, we estimate the indices of those neighbors by another. The right central descendant of u in level k is the vertex c_r with label $l_{c_r} = l_u 10^{k-i-1}$ corresponding to the index $x_{c_r} = x_u \cdot 2^{k-i} + 2^{k-i-1}$. Thus, as u has $a_i = \lfloor \sqrt{2^{k-i}} \rfloor$ neighbors in level k with prefix $l_u 1$, the rightmost neighbor of u in level k is the vertex with index $x_b = x_{c_r} + (a_i - 1)$. We call this vertex b . We consider similar vertices for the vertex v . The right central descendant of v in level k is the vertex c'_r with label $l_{c'_r} = l_v 10^{k-j-1} = l_u 10^{j-i-1} 10^{k-j-1}$ corresponding to the index $x_{c'_r} = x_u \cdot 2^{k-i} + 2^{k-i-1} + 2^{k-j-1} = x_{c_r} + 2^{k-j-1}$. Again, v having $a_j = \lfloor \sqrt{2^{k-j}} \rfloor$ neighbors in level k with prefix $l_v 1$, implies that the rightmost neighbor of v in level k is the vertex with index $x_{b'} = x_{c'_r} + (a_j - 1) = x_{c_r} + 2^{k-j-1} + a_j - 1$. We call this vertex b' .

Given the calculated indices of the vertices b and b' , we show that the vertex b is to the right of or equal to the vertex b' , thereby completing the proof. That is, we aim to show that $x_{b'}$ is at most x_b . In order to do that, we compare their distance to x_{c_r} .

$$\begin{aligned}
x_{b'} - x_{c_r} &= 2^{k-j-1} + a_j - 1 \\
&= 2^{k-j-1} + \left\lfloor \sqrt{2^{k-j}} \right\rfloor - 1 \\
&\stackrel{(1)}{\leq} 2^{k-j-1} + 2^{k-j-1} - 1 \\
&= 2^{k-j} - 1 \\
&= 2^{\frac{k-j}{2} + \frac{k-j}{2}} - 1 \\
&\stackrel{k \leq 2j-i-2}{\leq} 2^{\frac{k-j}{2} + \frac{2j-i-2-j}{2}} - 1 \\
&= 2^{\frac{k-i}{2}-1} - 1 \\
&= 2^{\frac{k-i}{2}} - 2^{\frac{k-i}{2}-1} - 1 \\
&\leq \left\lfloor \sqrt{2^{k-i}} \right\rfloor + 1 - 2^{\frac{k-i}{2}-1} - 1 \\
&= a_i - 2^{\frac{k-i}{2}-1} \\
&\leq a_i - 2^{1-1} \\
&= a_i - 1 \\
&= x_b - x_{c_r}
\end{aligned}$$

We use for the last inequality that $i < j < k$ with i, j, k being integers implies that $k - i \geq 2$.

For (1), we use the inequality $\left\lfloor \sqrt{2^{k-j}} \right\rfloor \leq 2^{k-j-1}$ which we obtain for $k - j = 1$ by observing that both sides are equal to 1 and for $k - j \geq 2$ by estimating $\left\lfloor \sqrt{2^{k-j}} \right\rfloor \leq 2^{\frac{k-j}{2}} \leq 2^{\frac{k-j}{2} + \frac{k-j}{2} - 1} = 2^{k-j-1}$. $\square$

We now use this auxiliary lemma to prove the lower bound for the clique number by picking vertices such that they induce a clique in H_r .

Lemma 4.14: *The clique number of H_r is in $\Omega(\log(r))$.*

Proof. We aim to choose a sequence of vertices $u_1, \dots, u_m$ such that for all $c \in [m-1]$ we have that $u_{c+1} \in N^+(u_c)$ and that furthermore $N^+(u_{c+1}) \subseteq N^+(u_c)$. These vertices form a clique of order m in H_r .

If u_c is a chosen vertex on level i , we pick the right central descendant of u_c in level $j = \left\lceil \frac{r+i+2}{2} \right\rceil$ as the next vertex u_{c+1} . By definition, we have $u_c u_{c+1} \in E(H_r)$. For every other level k such that $i < k \leq r$, we have that $k \leq r = r + i + 2 - i - 2 = 2 \cdot \frac{r+i+2}{2} - i - 2 \leq 2 \cdot \left\lceil \frac{r+i+2}{2} \right\rceil - i - 2 = 2j - i - 2$ and hence, we may apply Lemma 4.13. As this applies for every level larger than j , we have that every vertex that is a neighbor of u_{c+1} in a level larger than j , is also a neighbor of u_c , i.e., $N^+(u_{c+1}) \subseteq N^+(u_c)$. This approximately cuts the distance of the level of u_{c+1} to level r by half, compared to vertex u_c , and we can see that we are able to pick asymptotically $\log_2(r)$ such vertices. Therefore, we conclude that the clique number of H_r is in $\Omega(\log(r))$. $\square$

After proving the upper and lower bound of the clique number of the graphs H_r , we can combine them into a single Lemma.

Lemma 4.15: *The clique number of H_r is in $\Theta(\log(r))$.*

Proof. Apply Lemmas 4.12 and 4.14 for the lower and the upper bound respectively. $\square$

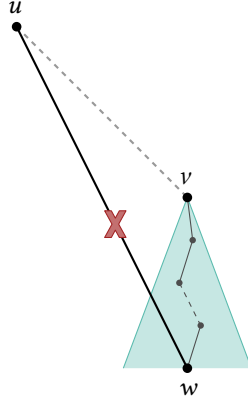


Figure 4.16: Illustrates what it means that a tree-based graph is ancestor-preserving. That is that for a vertex u and a descendant of u , vertex v , and again a descendant of v , vertex w , it holds the following. The edge uw , that is crossed out, cannot exist, if the edge uv is not in the graph. The green area illustrates the descendants of v .

4.2.2 Stack Number of the Graphs H_r

In this section, we show that the stack number of H_r is in $\Theta(\log(r)) = \Theta(\log(\log(n)))$ (Theorem 4.19).

Recall that we proved some results on ancestor-preserving tree-based graphs in Section 4.1. For a tree-based graph to be ancestor-preserving means the following. If a vertex u has a descendant v and a neighbor that is a descendant of v , then the vertices u and v are already adjacent, see also Figure 4.16 as a visual illustration. In order to determine the stack number of H_r , we want to use some of those results. Therefore, we show that the tree-based graph H_r is ancestor-preserving.

Lemma 4.16: *The tree-based graph H_r is ancestor-preserving.*

The situation of the following proof is depicted in Figure 4.17.

Proof. We assume without loss of generality that the graph H_r contains the three vertices that we pick in the following, otherwise the tree-based graph is trivially ancestor-preserving. Let $u = v_{i,x_u}$ be a vertex of H_r , $v = v_{j,x_v}$ be a descendant of u , and $w = v_{k,x_w}$ be a descendant of v . It suffices to show that if the edge uv is not in H_r that then, the edge uw is not in H_r .

Let l_u, l_v, l_w be the labels of vertices u, v, w , respectively.

Without loss of generality, because of symmetry, we assume that both vertices v and w are descendants of the right child of u , or possibly equal to the right child of u , i.e., the labels l_v and l_w have the prefix $l_u 1$.

Let $x_{c_j} = x_u \cdot 2^{j-i} + 2^{j-i-1}$ be the index of the right central descendant (c_j) of u in level j and let $x_{c_k} = x_u \cdot 2^{k-i} + 2^{k-i-1}$ be the index of the right central descendant (c_k) of u in level k . Note that $x_{c_k} = x_{c_j} \cdot 2^{k-j}$. The vertex u has $d_j = \lfloor \sqrt{2^{j-i}} \rfloor$ and $d_k = \lfloor \sqrt{2^{k-i}} \rfloor$ neighbors with prefix $l_u 1$ in levels j and k , respectively. We observe that $d_k \leq d_j \cdot 2^{k-j}$.

By construction (see Observation 4.8), the index of the leftmost descendant of v in level k is the index $x_l = x_v \cdot 2^{k-j}$. As the vertex w is a descendant of v , we have $x_w \geq x_l$. As vertices u and v are not adjacent, we know that the index of v is larger than $x_{c_j} + d_j - 1$, i.e., $x_v \geq x_{c_j} + d_j$. We multiply both sides by 2^{k-j} and get $x_w \geq x_l = x_v \cdot 2^{k-j} \geq 2^{k-j} \cdot (x_{c_j} + d_j) = x_{c_k} + 2^{k-j} \cdot d_j \geq x_{c_k} + d_k > x_{c_k} + d_k - 1$. Therefore, the vertices u and w are not adjacent. $\square$

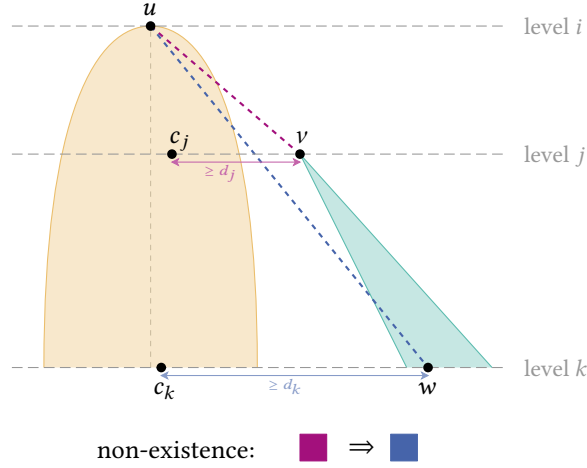


Figure 4.17: Illustrates the situation in the proof of Lemma 4.16. The vertex v is a descendant of the vertex u and the vertex w is a descendant of vertex v . We show that the non-existence of the edge uv in H_r implies the non-existence of the edge uw in H_r . In the proof we show that, if the difference of indices between vertices c_j and v is at least d_j , then the difference of indices between vertices c_k and w is at least d_k . Where d_j and d_k are half of the number of neighbors of u in levels j and k , respectively.

Given that we know that H_r is ancestor-preserving, we are able to apply lemmas about ancestor-preserving tree-based graphs in order to obtain the stack number with respect to a specific descendant-respecting vertex ordering.

Lemma 4.17: Let σ_r^{in} be the in-order vertex ordering that is obtained by using the complete binary tree within H_r . The stack number of H_r with respect to σ_r^{in} is in $\Theta(\omega(H_r))$.

Utilities for Lemma 4.17:

- 1 The graph H_r is a tree-based graph (by definition).
- 2 The vertex ordering σ_r^{in} is descendant-respecting. (given by Observation 2.13)
- 3 The tree-based graph H_r is ancestor-preserving (given by Lemma 4.16).

Proof. We start with the upper bound. H_r being an ancestor-preserving tree-based graph, allows us to apply Lemma 4.3, implying that for every vertex $v \in V(H_r)$, the size of its neighborhood in smaller levels $|N^-(v)|$ is at most the clique number $\omega(H_r)$. Using that H_r is an ancestor-preserving tree-based graph and σ_r^{in} is descendant-respecting, allows us to apply Lemma 4.2 with $k = \omega(H_r)$. This gives $\text{sn}_{\sigma_r^{\text{in}}}(H_r) \leq \omega(H_r) + 1$. Hence, $\text{sn}_{\sigma_r^{\text{in}}}(H_r) \in O(\omega(H_r))$.

We obtain the lower bound by applying Corollary 2.9 which yields that $\text{sn}(H_r) \in \Omega(\omega(H_r))$, and therefore, $\text{sn}_{\sigma_r^{\text{in}}}(H_r) \in \Omega(\omega(H_r))$ as well. Hence, we also have that $\text{sn}_{\sigma_r^{\text{in}}}(H_r) \in \Theta(\omega(H_r))$. $\square$

In preparation of the stack number of H_r in relation to the number of vertices, we derive the stack number of H_r in relation to the clique number of H_r .

Corollary 4.18: The stack number of H_r is in $\Theta(\omega(H_r))$.

Proof. We obtain the upper bound from Lemma 4.17 and the lower bound from Corollary 2.9. $\square$

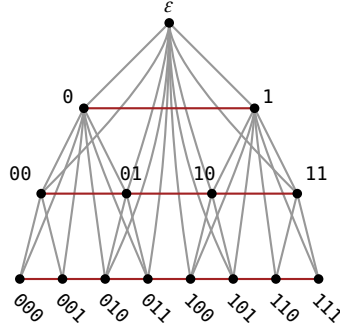


Figure 4.18: Illustration of the constructed graph H_3^P with its labels. The edges inherited from H_3 are gray and the newly added edges within the levels are red.

We conclude the section with the result that the stack number of H_r is in $\Theta(\log(\log(n)))$. We use the results proven in this section as well as the clique number of H_r , given by Lemma 4.15.

Theorem 4.19: *Let n be the number of vertices in H_r . Then the stack number of H_r is in $\Theta(\log(\log(n)))$.*

Proof. By Corollary 4.18, we get that the stack number of H_r is linear in the clique number of H_r . Furthermore, the clique number of H_r is logarithmic in r , as given in Lemma 4.15. As the number of vertices of H_r is $n = 2^{r+1} - 1$, it is obvious that $r \in \Theta(\log(n))$. Hence, we conclude $\text{sn}(H_r) \in \Theta(\omega(H_r)) = \Theta(\log(r)) = \Theta(\log(\log(n)))$. $\square$

4.2.3 The Intermediate Graphs H_r^P

Before we construct the actual graphs that are inspired by the paper [7], we create additional intermediate graphs that bring us one step closer to the graph G_r and its stack number. For each $r \in \mathbb{N}_0$, we define a graph H_r^P by adding some paths to the graph H_r . For each level of H_r , we add a path that contains all vertices of the level in the order given by their indices. We can formalize this as $H_r^P = H_r + E'$ with $E' = \left\{ ab \in \binom{V(H_r)}{2} \mid ab = v_{i,x}v_{i,x+1} \text{ for some } i, x \in \mathbb{N}_0 \right\}$. See Figure 4.18 for the result.

In order to determine the stack number of the graphs H_r^P , we start by proving that the additional edges compared to H_r only have constant stack number with respect to the in-order vertex ordering. This then allows us to conclude that the asymptotic behavior of the stack number of H_r^P is the same as for H_r .

Lemma 4.20: *Let σ_r^{in} be the in-order vertex ordering that is obtained by using the complete binary tree within H_r^P . For every $r \in \mathbb{N}_0$, the stack number of $H_r^P - E(H_r)$ with respect to σ_r^{in} is at most 3.*

We use an inductive argument for the proof of this lemma. As the vertex ordering has been obtained from an in-order of a binary tree, there is exactly one vertex outside of level r between each subsequent vertices of level r with subsequent indices (see Observation 2.15). Therefore, the situation looks like in Figure 4.19, and we see that we always have a color available when assigning the edges to stacks greedily by increasing level.

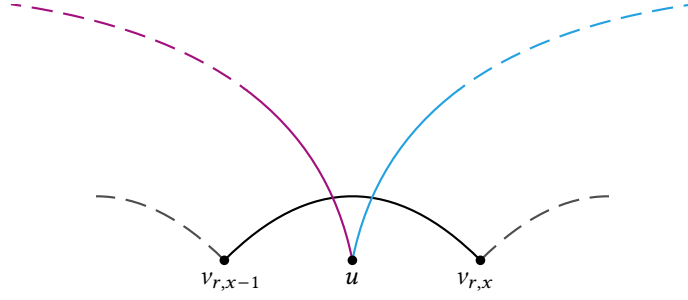


Figure 4.19: Shows an edge $e = v_{r,x-1}v_{r,x}$ of the path in level r of the graph $H_r^P - E(H_r)$ with a vertex u between the endpoints in the vertex ordering σ_r^{in} . All edges that cross the edge e are incident to vertex u in the layout σ_r^{in} . Therefore, the edge e can be colored in a color that none of the at most two edges incident to u are colored in.

Proof. For H_r^P we look at a graph $P^{(r)} = H_r^P - E(H_r)$ which consists of the intra-level edges on the same vertex set as H_r^P , i.e., we use the edges that form the paths within the levels. Observe that $\Delta(P^{(r)}) \leq 2$ for every $r \in \mathbb{N}_0$. We show $\text{sn}_{\sigma_r^{\text{in}}}(P^{(r)}) \leq 3$ by induction. The base case is $P^{(0)}$ which is just a single vertex in level 0 and therefore, trivial.

For the induction step, consider $r \in \mathbb{N}_+$. As seen in Observation 2.14, we may restrict σ_r^{in} to the vertices of $P^{(r-1)}$ to retrieve the vertex ordering σ_{r-1}^{in} . By induction, we know that there is a 3-stack layout of $P^{(r-1)}$ using vertex ordering σ_{r-1}^{in} . We aim to assign the edges $E' = E(P^{(r)}) \setminus E(P^{(r-1)})$ to the three stacks used by the layout for $P^{(r-1)}$ to obtain a 3-stack layout of $P^{(r)}$. The resulting stack layout might look like in Figure 4.20 for the graph $P^{(4)}$.

Let $x \in [2^r - 1]$, then $a = v_{r,x-1}, b = v_{r,x} \in V_r(H_r)$ are two vertices that are adjacent in $P^{(r)}$, and they are ordered $a <_{\sigma_r^{\text{in}}} u <_{\sigma_r^{\text{in}}} b$ in the vertex ordering σ_r^{in} . By Observation 2.15, there is exactly one vertex $u \in V(P^{(r-1)}) \subseteq V(P^{(r)})$ such that $a <_{\sigma_r^{\text{in}}} u <_{\sigma_r^{\text{in}}} b$. Therefore, if an edge e crosses the edge ab in the vertex ordering σ_r^{in} , then the edge e is incident to the vertex u . As the maximum degree is 2, the edge ab crosses edges in at most two stacks. Thus, there is a stack S such that none of the contained edges crosses ab . Therefore, we may assign ab to stack S .

As all edges in $E(P^{(r)}) \setminus E(P^{(r-1)})$ are like that, we may do this for all those edges independently. The edges $E(P^{(r)}) \setminus E(P^{(r-1)})$ generally do not cross as the vertices of the added path appear in the path in the same order as in the vertex ordering σ_r^{in} . Therefore, the process yields a 3-stack layout with respect to σ_r^{in} for $P^{(r)}$ based on one for $P^{(r-1)}$. Hence, we have $\text{sn}_{\sigma_r^{\text{in}}}(P^{(r)}) \leq 3$ as claimed. $\square$

We now combine the previous result with the stack number of H_r with respect to the in-order vertex ordering to obtain the result for H_r^P .

Lemma 4.21: *Let σ_r^{in} be the in-order vertex ordering that is obtained by using the complete binary tree within H_r^P . The stack number of H_r^P with respect to σ_r^{in} is in $\Theta(\omega(H_r^P))$.*

Proof. For the lower bound, we use Corollary 2.9, yielding $\text{sn}(H_r^P) \in \Omega(\omega(H_r^P))$.

For the upper bound, we consider the edges in H_r and $E(H_r^P) \setminus E(H_r)$ individually. By Lemma 4.20, we have a 3-stack layout of the edge set $E(H_r^P) \setminus E(H_r)$ with vertex ordering σ_r^{in} . By Lemma 4.17, we have that the stack number of H_r with respect to σ_r^{in} is in $\Theta(\omega(H_r))$ and together with $\omega(H_r) \leq \omega(H_r^P)$, we have that there is a stack layout of H_r using $O(\omega(H_r^P))$ stacks. We combine the partitions to obtain a stack layout of H_r^P .

Hence, we conclude $\text{sn}_{\sigma_r^{\text{in}}}(H_r^P) \in \Theta(\omega(H_r^P))$ $\square$

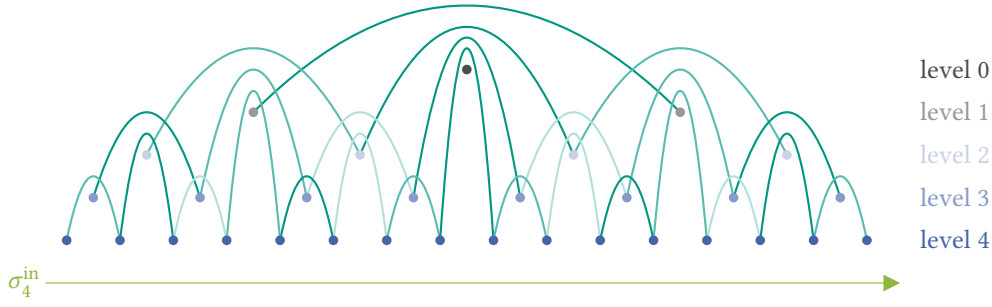


Figure 4.20: Illustrates the graph $P^{(4)} \subseteq H_4^P$ from the proof of Lemma 4.20 as well as a 3-stack layout for the vertex ordering σ_4^{in} . Each stack is represented by a different shade of green. The vertices are colored by their level in H_r . The vertices of a level are elevated a bit to improve visual clarity. As no edge is below some vertex, it is clear that it is still a stack layout.

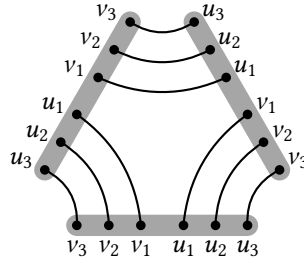


Figure 4.21: Illustrates the chaining of a graph with three copies. The copies of the graph are each marked by a gray blob.

Together with Corollary 2.9, we summarize the result as follows.

Corollary 4.22: *The stack number of H_r^P is in $\Theta(\omega(H_r^P))$.*

4.2.4 The Family $\mathcal{H}$

We define a different concept, the *chaining* of a graph. Afterwards, we use the new concept of chaining a graph in combination with our previous efforts to construct a family of graphs.

Let G be a graph and σ a vertex ordering of G . Let $k \in \mathbb{N}_+$ and $u_1, \dots, u_k, v_1, \dots, v_k$ be some vertices of G such that the vertices are ordered $v_k <_\sigma \dots <_\sigma v_1 \leq_\sigma u_1 <_\sigma \dots <_\sigma u_k$. The vertices v_1 and u_1 may be equal. Otherwise, the vertex v_1 is to the left of the vertex u_1 .

For each $p \in \mathbb{N}_+$, we define the *chaining-graph* $C = C_{G, \sigma, (u_1, \dots, u_k, v_1, \dots, v_k), p}$. We use p copies of G , denoted by G_i for each $i \in [p]$, and consider them in a cyclic order. Additionally, we insert the edge from u_j of copy G_i to v_j in the next copy G_{i+1} , indices modulo p , for all $j \in [k]$ and all copies of G . The set of edges between the u -vertices of G_i and the v -vertices of G_{i+1} , for $i \in [p]$ and indices modulo p is called the *i -th rainbow* of C . See Figure 4.21 for an abstract example of a chaining graph.

We show that the stack number of such a chaining graph is at most one more than the stack number of G with respect to σ . This is used by defining the graphs in the family $\mathcal{H}$ in a way such that they are obtained by chaining some H_r 's.

Lemma 4.23: *Let $C = C_{G, \sigma, (u_1, \dots, u_k, v_1, \dots, v_k), p}$ be a chaining-graph for some graph G with a vertex ordering σ , some $p, k \in \mathbb{N}_+$, and some vertices $u_1, \dots, u_k, v_1, \dots, v_k \in V(G)$ such that their order matches $v_k <_\sigma \dots <_\sigma v_1 \leq_\sigma u_1 <_\sigma \dots <_\sigma u_k$. Then $\text{sn}(C) \leq \text{sn}_\sigma(G) + 1$.*

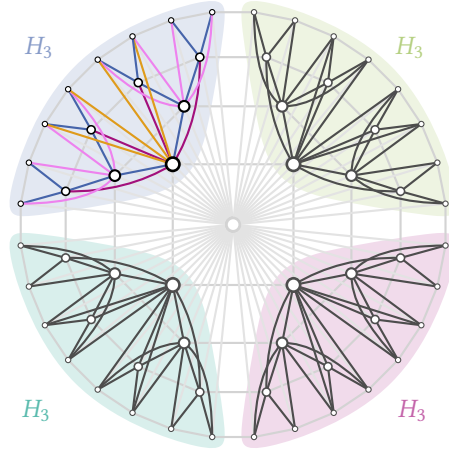


Figure 4.22: Illustrates how the copies of H_3 contribute to G_3 with the additional universal vertex and edges in light gray.

Proof. We use the vertex ordering σ' of C that is obtained by primarily ordering the vertices by the copy G_i they are part of, and secondarily we order the vertices within a copy just as they are ordered in σ .

We assign the edges within the copies of G to $\text{sn}(G)$ many stacks as the vertices of the copies of G are contiguous in σ' . The assignment of the edges is performed for each copy of G individually in the same way as it is done for G alone in the vertex ordering σ to achieve a $\text{sn}_\sigma(G)$ -stack layout for G . Edges belonging to different copies of G do not cross since the vertices of each copy appear as a block in σ' . In order to prove the upper bound, it remains to show that the edges between the copies of G do not cross, as this shows that they can be assigned to the same additional stack.

We observe that for every $i \in [p]$, the i -th rainbow of C is in fact a rainbow in the vertex ordering σ' . Furthermore, an edge of the i -th rainbow, $i \neq p$, nests below every edge in the p -th rainbow which it does not share a vertex with. In either case the edges do not cross. Similarly, if we consider edges e and e' in the i -th and the j -th rainbow, respectively, for $i < j < p$, then they share a vertex, or they are disjoint in the vertex ordering σ' . So, we have that none of the edges between copies of G cross.

Hence, the stack number of C is at most one more than the stack number of G with respect to σ . $\square$

Now that we have done all the previous work, we can assemble the graphs from our family that is inspired by the hyperbolic uniform disk graphs from the paper [7].

For each $r \in \mathbb{N}_+$, the family $\mathcal{H}$ contains a graph $G_r = (V, E)$. The graph G_r consists of four copies of H_r^P which are chained by using each the first and the last vertex of each level for the v -vertices and u -vertices, respectively. The order of the vertices for the chaining is considered by their level, e.g., the last vertex in level r corresponds to vertex u_{r+1} for the chaining. This matches with the vertex ordering σ_r^{in} of H_r as the vertices appear in exactly that order. We further add a universal vertex. The graphs G_r then look like depicted in Figure 4.22.

Given the definition of the graphs, we prove that the stack number of G_r is in linear in its clique number.

Lemma 4.24: *Let $G_r \in \mathcal{H}$. The stack number of G_r is in $\Theta(\omega(G_r))$.*

Proof. We start with the upper bound. Firstly, we use one stack for all edges incident to the universal vertex c in the center, which allows us to entirely ignore this specific vertex in the following. The graph $G_r - c$ consists of four copies of H_r^P which are chained. By Lemma 4.23, we have that the stack number of $G_r - c$ is at most one more than $\text{sn}_{\sigma_r^{\text{in}}}(H_r^P)$. Using Lemma 4.21, we get the upper bound $O(\omega(H_r^P))$. Note that, as H_r^P is a subgraph of G_r , the clique number of G_r is at least the clique number of H_r^P . Hence, $\text{sn}(G_r) \in O(\omega(G_r))$.

We obtain the lower bound by applying Corollary 2.9 which yields that $\text{sn}(G_r) \in \Omega(\omega(G_r))$. Hence, we also have that $\text{sn}(G_r) \in \Theta(\omega(G_r))$. $\square$

We conclude the section by proving that the stack number of G_r is in $\Theta(\log(\log(n)))$. We show that the clique number is the same as that of H_r up to a constant factor. This allows us to use the determined clique number of H_r (see Lemma 4.15).

Theorem 4.25: *Let $G_r \in \mathcal{H}$ with n vertices. The stack number of the graph G_r is in $\Theta(\log(\log(n)))$.*

Proof. By Lemma 4.24, we get that the stack number of G_r is linear in the clique number of G_r . Furthermore, the clique number of G_r is at least that of H_r as it contains a copy of H_r . The universal vertex only increases the clique number by a constant amount. Hence, it does not change the asymptotic behavior of the clique number of graphs in $\mathcal{H}$. On the other hand, removing the universal vertex as well as the edges of the four copies of H_r leaves just some cycles in G_r . The cycles have clique number 2. By the definition of a tree-based graph, the levels of H_r are independent sets. Thus, the vertices of a clique in G_r that are on the same level, have to induce a clique within the cycles. This implies that the clique number in G_r without the universal vertex is at most twice the clique number of H_r . Therefore, the clique number of G_r is in $\Theta(\omega(H_r))$.

Thus, by Lemma 4.15, the clique number of G_r is logarithmic in r . As the number of vertices of G_r is $n = 4 \cdot (2^{r+1} - 1) + 1 = 2^{r+3} - 3$, it follows that $r \in \Theta(\log(n))$. Hence, we conclude $\text{sn}(G_r) \in \Theta(\omega(G_r)) = \Theta(\log(r)) = \Theta(\log(\log(n)))$. $\square$

4.2.5 Queue Number of the Graphs H_r

Having thoroughly considered the stack number of H_r and its modifications, we also want to consider their queue number. As we bounded the stack number of H_r in terms of its clique number, we want to bound the queue number of H_r in terms of its clique number as well. This way, since the stack number is linear in the clique number in graphs of each of the considered graph classes, a queue number super-linear in terms of the clique number would be an example of a graph with small stack number and large queue number.

We start by proving that ancestor-preserving tree-based graphs are chordal, providing that their treewidth is approximately equal to their clique number. This allows us to bound the queue number of H_r using previous results in terms of the treewidth.

Lemma 4.26: *Let G be an ancestor-preserving tree-based graph. Then, the graph G is chordal.*

Proof. Recall the equivalence (Lemma 2.3) that a graph is chordal if and only if it has a perfect elimination scheme. To show that G is chordal, we aim to choose a vertex ordering σ such that it is a perfect elimination scheme, which then completes the proof.

For the vertex ordering σ , we order the vertices of G ascending by their level. Let v be a vertex of G . Therefore, every neighbor that is to the left of vertex v , is in a smaller level than v itself. That is, the neighbors to the left of v are a subset of the set $N^-(v)$, which in turn,

if there are any, by Lemma 4.3, induce a clique in G . Hence, if there are neighbors of v that are to the left of v , they induce a clique, which shows that the vertex ordering σ is indeed a perfect elimination scheme. $\square$

Lemma 4.26 immediately implies that the graph H_r is chordal.

Corollary 4.27: *The graph H_r is chordal.*

Proof. By Lemma 4.16, the tree-based graph H_r is ancestor-preserving. Applying Lemma 4.26 completes the proof. $\square$

We use the result by Wiechert [37] which improves upon a previous bound on the queue number of graphs with bounded treewidth by Dujmović, Morin and Wood [17].

Lemma 4.28 (Wiechert, 2017 [37, Theorem 1]): *Let G be a graph. Then $\text{qn}(G) \leq 2^{\text{tw}(G)} - 1$.*

Combining this result with the fact that H_r is chordal provides us with an upper bound for the queue number of H_r .

Lemma 4.29: *The queue number of H_r is in $\Omega(\omega(H_r)) \cap O(2^{\omega(H_r)})$.*

Proof. We obtain the lower bound by applying Corollary 2.11.

By Corollary 4.27, the graph H_r is chordal. Therefore, its treewidth is one less than its clique number (Lemma 2.2). Using Lemma 4.28, we can bound the queue number of H_r by $\text{qn}(H_r) \leq 2^{\text{tw}(H_r)} = 2^{\omega(H_r)-1}$. Hence, we have the upper bound of $O(2^{\omega(H_r)})$. $\square$

4.2.6 Modifying the Graphs H_r

The motivation is to find a family of graphs with bounded stack number and unbounded queue number. Similarly to the paper [7] in which a family similar to $\mathcal{H}$ is defined, we delete some levels of the graphs H_r in order to obtain a family with both constant stack number and constant queue number. Once again adding some intra-level paths, as with H_r^P , does not increase their stack number by too much while possibly increasing queue number.

For each $s \in \mathbb{N}_0$, we define a tree-based graph H'_s that is an induced subgraph of the tree-based graph $H = H_{2^s}$ using the same underlying rooted tree. We take the vertices of the levels that are a power of two in addition to the root vertex of the underlying rooted tree in level 0. More formally, the graph H'_s is the graph induced by the set

$$V' = V_0(H) \cup \bigcup_{k=0}^s V_{2^k}(H)$$

in the graph H , i.e., $H'_s = H[V']$. Recall that $V_i(H)$ denotes the set of vertices of H in level i . See Figure 4.23 for an illustration of H'_2 .

Recall that Lemma 4.5 states that induced subgraphs of an ancestor-preserving tree-based graph are, with the same underlying rooted tree, ancestor-preserving tree-based graphs. Since H_{2^s} is an ancestor-preserving tree-based graph (Lemma 4.16) and H'_s is an induced subgraph of H_{2^s} , Lemma 4.5 immediately implies the following observation.

Observation 4.30: *The graph H'_s is an ancestor-preserving tree-based graph with the underlying tree of the tree-based graph H_{2^s} .*

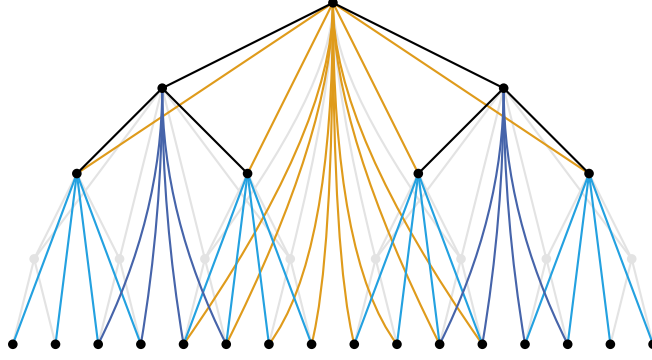


Figure 4.23: The graph H'_2 . Edges of the underlying rooted tree are colored black, other edges are colored by the level of their parent. Vertices and incident edges of deleted levels are indicated by a very light gray.

As the graph H'_s is an ancestor-preserving tree-based graph, we want to reuse previous results to first obtain an upper bound for the stack number of induced subgraphs of H_r in terms of their clique number. Therefore, to show that the graphs H'_s have bounded stack number and bounded queue number, it is sufficient to prove that the clique number of H'_s is bounded by a constant.

Lemma 4.31: *Every induced subgraph G of H_r has stack number at most $\omega(G)$ for every $r \in \mathbb{N}_0$. A vertex ordering admitting such a stack layout can be obtained by restricting the vertex ordering σ_r^{in} to the vertices of G .*

Recall that σ_r^{in} is the in-order of the vertices of the underlying rooted tree of the tree-based graph H_r .

Proof. Let σ be the vertex ordering obtained by restricting σ_r^{in} to the vertices of G . Applying Lemma 4.5 provides that G is an ancestor-preserving tree-based graph with the same underlying tree as of H_r . Observe that σ inherits being descendant-respecting from the vertex ordering σ_r^{in} . Recall that Corollary 4.4 provides that the stack number of ancestor-preserving tree-based graphs with respect to everything descendant-respecting vertex ordering is at most their clique number. Hence, there is an $\omega(G)$ -stack layout of G using vertex ordering σ . $\square$

We determine the clique number of H'_s to then apply the previous lemma.

Lemma 4.32: *The clique number of H'_s is at most 5 for every $s \in \mathbb{N}_0$.*

Note that the given proof resembles one in the paper [6, Lemma 19].

Proof. As H'_s is a tree-based graph, the vertices of a level are an independent set and every clique contains at most one vertex of each level. If $s \leq 1$, the graph H'_s has at most three levels, so the bound holds trivially. We assume $s \geq 2$ for the rest of the proof.

The levels 0, 1 and 2 each contribute at most one vertex to every clique. So, it remains to show that the clique number of the graph $G = H'_s - (V_0 \cup V_1 \cup V_2)$ is at most 2. Suppose for contradiction that there is a triangle in G , i.e., there are vertices $u, v, w \in V(G)$ which induce a triangle. Since vertices of a clique are in different levels, we may assume without

loss of generality that vertices u, v, w are in levels $2^i, 2^j, 2^k$, respectively, for $2 \leq i < j < k$ and $i, j, k \in \mathbb{N}_+$. However, we can estimate $2^k \geq 2 \cdot 2^j \stackrel{i \geq 2}{\geq} 2 \cdot 2^j - 2^i + 4$. The shared neighbor w of vertices u and v in level 2^k contradicts to Lemma 4.10 which provides that, with $2^k \geq 2 \cdot 2^j - 2^i + 4$, vertices u, v in levels $2^i, 2^j$ have no common neighbors in level 2^k .

Hence, the graph G has clique number at most 2 and H'_s has clique number at most 5. $\square$

The previous two results immediately imply the following corollary.

Corollary 4.33: *The stack number of H'_s is at most 5 for every $s \in \mathbb{N}_0$.*

Proof. Apply Lemmas 4.31 and 4.32. $\square$

Similarly, we are able to exploit the fact that the clique number of H'_s is bounded by a constant to obtain a constant upper bound for the queue number of H'_s .

Corollary 4.34: *The queue number of H'_s is at most 15 for every $s \in \mathbb{N}_0$.*

Proof. By Observation 4.30 and Lemma 4.26 we have that H'_s is chordal. Thus, applying Lemma 2.2 yields $\text{tw}(H'_s) = \omega(H'_s) - 1$ and Lemma 4.28 provides an upper bound on the queue number of $\text{qn}(H'_s) \leq 2^{\text{tw}(H'_s)} - 1$. By Lemma 4.32, the clique number of H'_s is at most 5, thereby completing the proof after performing simple remaining calculations: $\text{qn}(H'_s) \leq 2^{\text{tw}(H'_s)} - 1 = 2^{\omega(H'_s)-1} - 1 = 2^{5-1} - 1 = 15$. $\square$

Similarly to the definition of H'_s , we can delete levels of the graph H_r^P . For each $s \in \mathbb{N}_0$, we define a graph $H_s'^P$ on the same vertex set as H'_s but as the induced subgraph of $H_{2^s}^P$. One could interpret it as well as again adding some intra-level paths to the graph H'_s .

We once again begin by determining the stack number of induced subgraphs of H_r^P in terms of its clique number. This allows us to obtain a constant upper bound for the stack number of $H_s'^P$ by determining a constant upper bound for the clique number of $H_s'^P$.

Lemma 4.35: *Every induced subgraph G of H_r^P has stack number at most $\omega(G) + 3$ for every $r \in \mathbb{N}_0$.*

Proof. We use the in-order vertex ordering σ_r^{in} and restrict it to the vertices of G to get a vertex ordering σ . Let $X = V(G) \subseteq V(H_r^P)$ be the set of vertices that induces G in H_r^P . We have that the edges of $H_r[X]$ require at most $\omega(G)$ stacks with σ by Lemma 4.31. We observe that the graph $G - E(H_r[X])$ is an induced subgraph of $H_r^P - E(H_r)$. Using that the stack number of $H_r^P - E(H_r)$ with respect to σ_r^{in} is at most 3 (Lemma 4.20), we get, as σ is obtained by restricting σ_r^{in} to X , that the stack number of $G - E(H_r[X])$ with respect to σ is at most 3. Hence, the stack number of G is at most $\omega(G) + 3$. $\square$

As mentioned, we continue by providing a constant upper bound for the clique number of $H_s'^P$, deriving from our work on the clique number of H'_s .

Lemma 4.36: *The clique number of $H_s'^P$ is at most 6 for every $s \in \mathbb{N}_0$.*

Proof. Let $C \subseteq H_s'^P$ be a clique in $H_s'^P$. If C contains at most one vertex in each level, we do not use any of the intra-level edges. Hence, C is a subgraph of H'_s , i.e., we have that C consists of at most $\omega(H'_s)$ vertices, which by Lemma 4.32 is at most 5.

We consider the case that C contains multiple vertices in the same level. Let level i be the smallest such level, i.e., for every level j with $j < i$, there is at most one vertex in level j that is in C . Let u and v be two different vertices that are in level i . Then, they have no common neighbors in $H_s'^P$ that are in levels larger than level i as this would otherwise contradict to

H'_s being tree-based. Since $H'_s{}^P - E(H'_s)$ only contains intra-level edges, this implies that the clique C contains no vertices that are in levels larger than level i . In particular, there is at most one vertex in C of each level except level i . Further, since $H'_s{}^P - E(H'_s)$ consists of vertex-disjoint paths, there is no vertex besides u and v in C that is in level i . Therefore, we conclude that $C - u$ is a subgraph of H'_s , and we have $|V(C)| = |V(C - u)| + 1 \leq \omega(H'_s) + 1$. With Lemma 4.32, we have $\omega(H'_s) \leq 5$, thereby completing the proof. $\square$

We use the previous two lemmas to obtain a constant upper bound for the stack number of $H'_s{}^P$.

Corollary 4.37: *The stack number of $H'_s{}^P$ is at most 9 for every $s \in \mathbb{N}_0$.*

Proof. Apply Lemmas 4.35 and 4.36. $\square$

4.2.7 Queue Number of the Graphs H_r^P and H_s^P

The case for the queue number of the graphs H_r^P and H_s^P is a bit more difficult. Since we determine that they have high treewidth, just applying the lemma by Wiechert [37] does not yield good bounds. To prove that particularly H_s^P has super-constant treewidth, we show that it has a clique of super-constant order as a minor and then use the following lemma.

Lemma 4.38 ([12, Lemma 12.4.1]): *Let G, H be graphs such that the graph G contains an H -minor. Then $\text{tw}(H) \leq \text{tw}(G)$.*

Recall that the levels in which H_s^P contains vertices are level 0 and level 2^i for each $i \in \{0, \dots, s\}$. So there are $s + 2$ levels which contain vertices. By contracting the paths within each level, and with the fact that there are edges between vertices of every pair of levels, there is a K_{s+2} -minor in H_s^P . With Lemma 4.38, we thus obtain the following observation.

Observation 4.39: *For every $s \in \mathbb{N}_0$, the treewidth of H_s^P is at least $s + 1$.*

Since the treewidth of H_s^P is super-constant, we are not able to simply use Lemma 4.28 as an upper bound for a constant queue number of H_s^P . Instead, we provide a linear bound in the number of levels containing vertices compared to the exponential bound given by applying Wiechert's result [37].

Lemma 4.40: *Let G be a subgraph of H_r^P for some $r \in \mathbb{N}_0$. The graph G has queue number at most $2l - 1$, where l is the number of levels in H_r^P such that G contains a vertex of it.*

More formally, we set $l = \sum_{i=0}^r \min(1, |V_i(G)|)$.

Proof. Without loss of generality assume that G is an induced subgraph of H_r^P . Let σ be the vertex ordering obtained by restricting the vertex ordering σ_r^{in} of H_r to the vertices of G . See Figure 4.24 for how σ and the constructed queue layout look for $H_2^P \subseteq H_4^P$. Note that the vertices of an intra-level path in H_r^P appear in σ_r^{in} in the order of the path. Hence, no two intra-level edges in G with endpoints in the same level nest in σ and as G has vertices in exactly l levels, using l queues is sufficient for the intra-level edges of G . It remains to assign the edges of $H_r[V(G)]$ to queues.

We observe that $H_r[V(G)]$ is a tree-based graph with the same underlying tree as H_r . Recall that σ_r^{in} is a descendant-respecting vertex ordering of H_r and by restricting it, we have that σ is a descendant-respecting vertex ordering of $H_r[V(G)]$. By Lemma 4.1, this means that edges with parents on the same level do not nest. This allows us to distribute

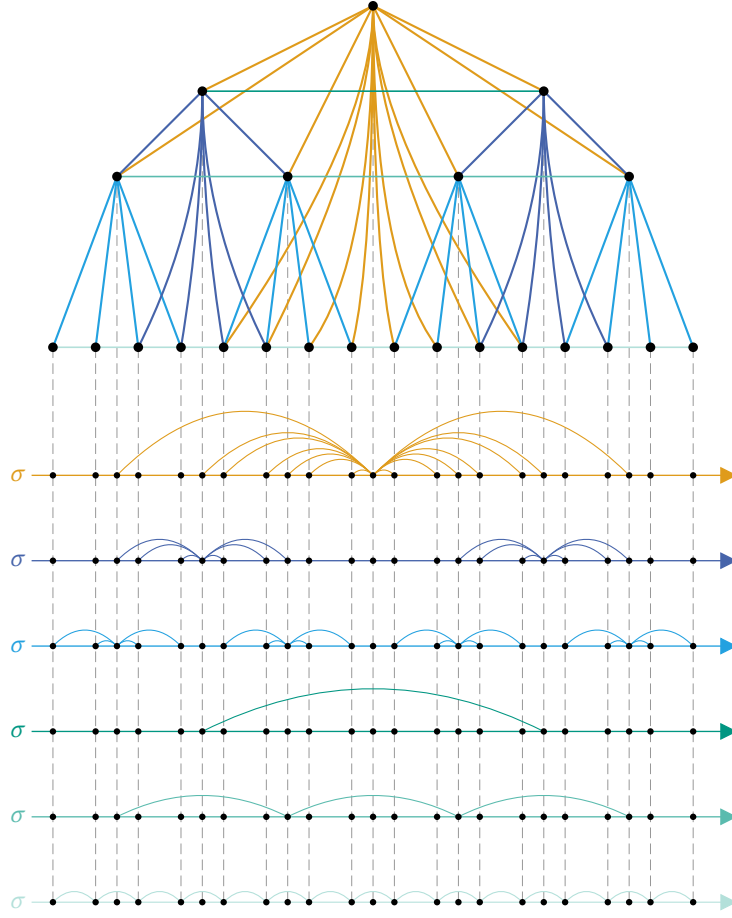


Figure 4.24: Illustration of the queue layout for $H_2^P \subseteq H_4^P$ as constructed in the proof of Lemma 4.40. The queue for the path in level 0 is omitted since it contains no edges. The vertical, dashed, gray lines indicate which vertices are the same in the vertex orderings and in the graph.

the remaining edges in l queues, one each for the edges with parents on the same level since there are vertices in at most l levels. We save one queue by the fact that the vertices in level $\max \{i \in \{0, \dots, r\} \mid V_i(G) \neq \emptyset\}$ do not have any edges for which they are parents since there are no vertices in larger levels.

All together, we assigned edges to at most $2l - 1$ queues. $\square$

Note that for H_r^P , this bound asymptotically matches the bound for the queue number of H_r since $\text{tw}(H_r) = \omega(H_r) - 1 \in \Theta(\log(r))$ and Wiechert [37] provides an upper bound of $2^{\text{tw}(H_r)} - 1$.

Corollary 4.41: *The queue number of H_r^P is at most $2r + 1$ for every $r \in \mathbb{N}_0$.*

Proof. The graph H_r^P has vertices in levels $0, \dots, r$ of H_r , i.e., $r + 1$ levels with vertices. Hence, by Lemma 4.40, we have $\text{qn}(H_r^P) \leq 2 \cdot (r + 1) - 1$. $\square$

Similarly, we derive a result for the graph $H_s'^P$ with an upper bound for the queue number of $H_s'^P$ in $O(s)$ instead of $O(2^s)$ as obtainable by applying Wiechert's result [37].

Corollary 4.42: *The queue number of $H_s'^P$ is at most $2s + 3$ for every $r \in \mathbb{N}_0$.*

Proof. The graph $H_s'^P$ has vertices in levels $0, 2^0, 2^1, \dots, 2^s$ of H_{2^s} , i.e., $s + 2$ levels with vertices. Hence, by Lemma 4.40, we have $\text{qn}(H_s'^P) \leq 2 \cdot (s + 2) - 1$. $\square$

All together, we note that the graphs $H_s'^P$ have constant stack number and unbounded treewidth. Since for large enough s , they contain a K_5 -minor, almost all of the graphs $H_s'^P$ are not planar by Wagner's theorem [36]. Hence, the upper bound on the queue number of planar graphs [16] cannot be used either. Since we do not know a constant upper bound on their queue number, they are an interesting candidate for solving Open Problem 1.2, assuming that it can be solved in the negative.

5 Queue Numbers with Restricted Vertex Orderings

In this chapter, we consider different versions of the queue number by restricting the allowed set of vertex orderings. We show a lower bound for the queue number and some negative results when considering certain vertex orderings.

5.1 BFS-orderings

Let G be a graph. Each BFS-ordering of G can be obtained by starting at a vertex and then using some other vertex ordering as a tiebreaker to sort the vertices within each BFS-level. For a vertex $r \in V(G)$ and vertex ordering σ of G , we define the (r, σ) -BFS-ordering of G , denoted $\Rightarrow_{\text{BFS}}(r, \sigma, G)$, which is obtained by primarily sorting vertices ascending by their distance to r and secondarily by the order σ . We say that the vertex ordering σ produces the BFS-ordering $\Rightarrow_{\text{BFS}}(r, \sigma, G)$. We say that σ is an *original* vertex ordering of the BFS-ordering $\Rightarrow_{\text{BFS}}(r, \sigma, G)$. See Figure 5.1 for an example how a vertex ordering differs from its produced BFS-ordering. We define a modified variant of the queue number. For a start vertex $r \in V(G)$, the r -BFS-queue number of G with start vertex r is defined via the usual queue number as the minimum queue number of G with respect to r -BFS-orderings. We denote it by

$$\text{BFSqn}(r, G) = \min_{\sigma \text{ vertex ordering of } G} \text{qn}_{\Rightarrow_{\text{BFS}}(r, \sigma, G)}(G).$$

Figure 5.1 already provides an example that the queue number with respect to a BFS-ordering can be larger than the queue number of the original vertex ordering. The question is, what is the worst case in terms of an increased requirement of queues. It turns out that using three times the number of queues is always sufficient.

Lemma 5.1: *Let G be a graph with a vertex ordering σ . Further, consider a vertex r of G and the (r, σ) -BFS-ordering $\sigma' = \Rightarrow_{\text{BFS}}(r, \sigma, G)$. Then $\text{qn}_{\sigma'}(G) \leq 3 \cdot \text{qn}_{\sigma}(G)$. If G is bipartite, we have an upper bound of $\text{qn}_{\sigma'}(G) \leq 2 \cdot \text{qn}_{\sigma}(G)$.*

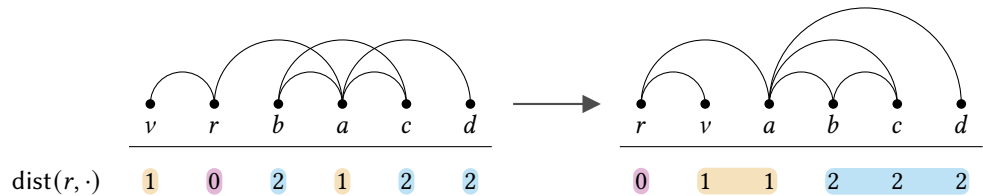


Figure 5.1: Illustrates the difference of the vertex ordering σ (left) and the (r, σ) -BFS-ordering (right). Note that while the original vertex ordering σ contains no 2-rainbow, the (r, σ) -BFS-ordering does contain a 2-rainbow.

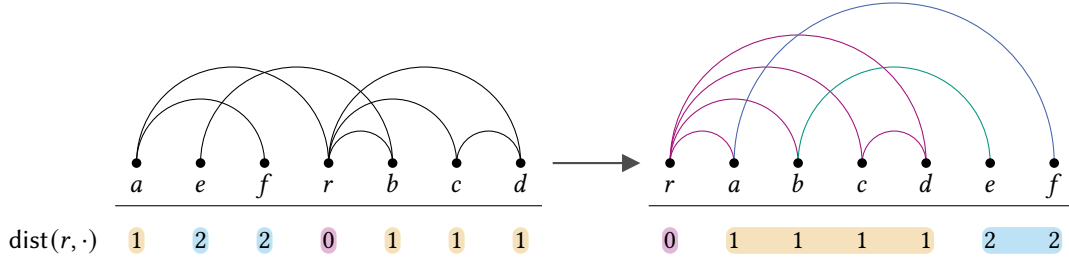


Figure 5.2: Illustrates the tightness of Lemma 5.1 for 1-queue layouts in the case of general graphs. The original vertex ordering contains no rainbow, but there is a 3-rainbow in the produced BFS-ordering. Each color represents a queue. The tightness for bipartite graphs can be observed by considering the subgraph without vertices c and d .

We are able to exploit this result in Corollary 5.5 which enables us to limit the vertex orderings that need to be considered for the asymptotic behavior of the queue number of a family of graphs. We show that the given bounds are tight in the following lemma; see Figure 5.2 for $k = 1$.

Lemma 5.2: *The bounds given by Lemma 5.1 are tight. That is, for each $k \in \mathbb{N}_+$, there is a graph G and a bipartite graph H with a k -queue layout using vertex orderings σ and τ , respectively, such that for some start vertex r ,*

- *there is a $3k$ -rainbow in the (r, σ) -BFS-ordering σ' of G ,*
- *there is a $2k$ -rainbow in the (r, τ) -BFS-ordering τ' of H .*

Proof. We generalize the example given for the case $k = 1$ (see Figure 5.2). We define a graph $G = G_k$ with a vertex ordering σ . For $k = 1$, we use the graph G_1 as visible in Figure 5.2 with the vertex ordering σ as visible in the vertex ordering in the left part of the figure. We consider the case $k \geq 2$. Use Figure 5.3a to follow along during the construction. We replace each of the vertices a, b, c, d, e, f of G_1 with a set of k vertices A, B, C, D, E, F , respectively, where $A = \{a_i \mid i \in [k]\}$ and similarly for the other vertex sets. For uniformity, we replace r with the singleton $R = \{r\}$. For each $i \in [k]$, we define $V_i = \{r, a_i, b_i, c_i, d_i, e_i, f_i\}$. We choose the edges of G_k such that the graph induced by V_i is isomorphic to the graph in Figure 5.2. More precisely, we have the edge set $E(G_k) = \bigcup_{i \in [k]} \{ra_i, rb_i, rc_i, rd_i, a_i f_i, b_i e_i, c_i d_i\}$. The order of the sets of vertices is given by the order of the vertices they replace in σ_1 : $A <_\sigma E <_\sigma F <_\sigma R <_\sigma B <_\sigma C <_\sigma D$. Internally, we order the vertices of the sets A, B, C ascending by their indices and the vertices of the sets D, E, F descending by their indices. Restricting the vertex ordering σ to V_i yields the same vertex ordering as depicted in Figure 5.2. Thus, providing a queue for the edges induced by each V_i is sufficient. Hence, we have a k -queue layout.

The (r, σ) -BFS-ordering σ' adheres to the order of σ within BFS-levels, and therefore, the sets are ordered $R <_{\sigma'} A <_{\sigma'} B <_{\sigma'} C <_{\sigma'} D <_{\sigma'} E <_{\sigma'} F$ with internal order of the vertices remaining intact within each of the vertex sets A, B, C, D, E, F . This means that the vertices are ordered $r <_{\sigma'} a_1 <_{\sigma'} \dots <_{\sigma'} a_k <_{\sigma'} b_1 <_{\sigma'} \dots <_{\sigma'} b_k <_{\sigma'} c_1 <_{\sigma'} \dots <_{\sigma'} c_k <_{\sigma'} d_k <_{\sigma'} \dots <_{\sigma'} d_1 <_{\sigma'} e_k <_{\sigma'} \dots <_{\sigma'} e_1 <_{\sigma'} f_k <_{\sigma'} \dots <_{\sigma'} f_1$. We have a $3k$ -rainbow in σ' with edges $a_i f_i, b_i e_i, c_i d_i \in E(G_k)$ for each $i \in [k]$.

For the bipartite case, we consider the graph $H = G_k - C - D$, see also Figure 5.3b. The graph H is bipartite since its vertices can be partitioned into independent sets $R \cup E \cup F$ and $A \cup B$. Further, it has queue number $\text{qn}_\tau(H) \leq \text{qn}_\sigma(G_k) \leq k$ with respect to the vertex

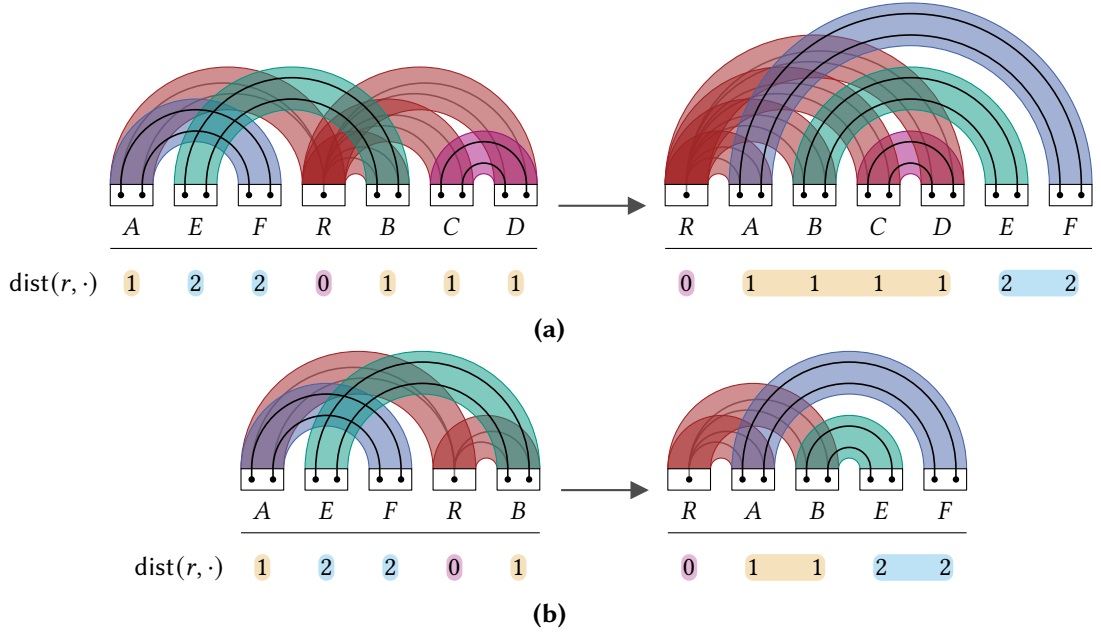


Figure 5.3: The original vertex ordering (left) and the produced BFS-ordering (right) as described in the proof of Lemma 5.2 for the general case of Lemma 5.1, (a) for arbitrary graphs and (b) for bipartite graphs. The edges incident to r are gray to indicate that they are still there, while hopefully avoiding a large deterioration of clarity. The figures depict the constructions for $k = 2$.

ordering τ which is obtained by restricting σ to the vertices of H . With $A <_{\tau} E <_{\tau} F <_{\tau} B$ and $N_H(r) = A \cup B$, we have that the sets are ordered $R <_{\tau'} A <_{\tau'} B <_{\tau'} E <_{\tau'} F$ in the (r, τ) -BFS-ordering τ' while the internal order of vertices within the sets is maintained because of equal distance to r for all vertices in each set. Therefore, the vertices of H_k are ordered $r <_{\tau'} a_1 <_{\tau'} \dots <_{\tau'} a_k <_{\tau'} b_1 <_{\tau'} \dots <_{\tau'} b_k <_{\tau'} e_k <_{\tau'} \dots <_{\tau'} e_1 <_{\tau'} f_k <_{\tau'} \dots <_{\tau'} f_1$. Thus, we have a $2k$ -rainbow in τ' with edges $a_i f_i, b_i e_i \in E(H)$ for each $i \in [k]$. $\square$

The vertex orderings considered by Lemma 5.1 are a special case of the Riffle-Lemma as mentioned by Katheder, Kaufmann, Pupyrev, and Ueckerdt [30, Lemma 6]. The Riffle-Lemma concerns a graph G with a fixed vertex ordering σ and a partition of the vertices of G into k parts $V_1, \dots, V_k$. One can use some new vertex ordering σ' of G and as long as the parts V_i are internally ordered the same in σ and σ' for every $i \in [k]$, the queue number increases at most by a factor of k^2 , i.e., $\text{qn}_{\sigma'}(G) \leq k^2 \cdot \text{qn}_{\sigma}(G)$. For bipartite graphs, the Riffle-Lemma provides even stronger bounds if $V_1, \dots, V_k$ are each a subset of a part in a bipartition of $V(G)$.

Lemma 5.3 (Riffle-Lemma [30, Lemma 6]): *Let G be a graph with vertex ordering σ and $V_1, \dots, V_k$ be a partition of $V(G)$. Then, for any vertex ordering σ' where for vertices $u, v \in V_i$ it holds that $u <_{\sigma'} v$ whenever $u <_{\sigma} v$: $\text{qn}_{\sigma'}(G) \leq k^2 \cdot \text{qn}_{\sigma}(G)$. If G is bipartite with parts A and B , $\bigcup_{i=1}^{\ell} V_i = A$ and $\bigcup_{i=\ell+1}^k V_i = B$, then $\text{qn}_{\sigma'} \leq 2 \cdot \ell \cdot (k - \ell) \cdot \text{qn}_{\sigma}(G)$.*

The BFS-orderings are a special case as the partition is given by the BFS-levels and the parts are additionally ordered. We have at most $\text{diam}(G) + 1$ BFS-levels for a connected graph G and start vertex $r \in V(G)$. Therefore, the Riffle-Lemma provides the bound $\text{qn}_{\Rightarrow \text{BFS}(r, \sigma, G)}(G) \leq$

$(\text{diam}(G) + 1)^2 \cdot \text{qn}_\sigma(G)$ and the constant factor 3 is an improvement for this special case. Similarly, the factor 2 is an improvement for bipartite graphs compared to the Riffle-Lemma for bipartite graphs.

We prepare the proof of Lemma 5.1 by observing that in some cases of BFS-orderings, when checking for rainbows in a set S of edges, it suffices to consider certain groups of edges if there are no edges in S which have both endpoints in the same level. We use the observation to limit the pairs of edges considered when showing that S is a stack as it provides necessary conditions for nesting edges.

Observation 5.4: *Let G be a graph with a vertex $r \in V(G)$ and an r -BFS-ordering σ . Let $S \subseteq E(G)$ such that for every edge $uv \in S$, it holds that $\text{dist}(r, u) \neq \text{dist}(r, v)$. Then for each two nesting edges $e = ab, e' = cd \in S$ with $a <_\sigma c <_\sigma d <_\sigma b$, we have $\text{dist}(r, a) = \text{dist}(r, c)$ and $\text{dist}(r, b) = \text{dist}(r, d) = \text{dist}(r, a) + 1$.*

Note that the observation is especially useful if G is a connected bipartite graph as the condition of endpoints of edges having different distances to r is met trivially.

Proof. The statement is trivial if S contains no nesting edges, i.e., if S is a stack in σ . So, assume that there are nesting edges $e = ab, e' = cd \in S$. Without loss of generality, we may assume $a <_\sigma c <_\sigma d <_\sigma b$. As σ is an r -BFS-ordering, this provides $\text{dist}(r, a) \leq \text{dist}(r, c) \leq \text{dist}(r, d) \leq \text{dist}(r, b)$. With endpoints of edges in S having different distances to r and σ being an r -BFS-ordering, we have $\text{dist}(r, b) = \text{dist}(r, a) + 1$ and $\text{dist}(r, d) = \text{dist}(r, c) + 1$. We derive $\text{dist}(r, d) = \text{dist}(r, c) + 1 \geq \text{dist}(r, a) + 1 = \text{dist}(r, b)$, and together with $\text{dist}(r, d) \leq \text{dist}(r, b)$, we have equality, i.e., $\text{dist}(r, d) = \text{dist}(r, b)$. Subtracting 1 from both sides provides $\text{dist}(r, c) = \text{dist}(r, d) - 1 = \text{dist}(r, b) - 1 = \text{dist}(r, a)$. This completes the proof. $\square$

The proof of Lemma 5.1 is partly inspired by the proof of the Riffle-Lemma in a paper by Katheder, Kaufmann, Pupyrev, and Ueckerdt [30, Lemma 6].

Proof of Lemma 5.1. We consider a fixed queue layout of G with vertex ordering σ and transform every queue of the layout into three queues each for a layout with respect to the vertex ordering σ' .

Let $Q \subseteq E(G)$ be a queue with vertex ordering σ . We create three new queues $Q_<, Q_=: Q_>$ with vertex ordering σ' and individually add the edges to a respective queue. Let $e = xy \in Q$ be an edge of the queue and assume that x is to the left of y in σ . The assignment of the edge e is based on the distances of the vertices x and y to the vertex r . The edge e is inserted into the queue $Q_\square$, for $\square \in \{<, =, >\}$ such that the (in)equality $\text{dist}(r, x) \square \text{dist}(r, y)$ holds.

We show for each of the three queues that they do not contain nesting edges.

We simultaneously consider the queues $Q_<$ and $Q_>$, as we similarly show that they each do not contain a 2-rainbow under the vertex ordering σ' . Let $\square \in \{<, >\}$ to allow for some common observations. Suppose for contradiction that there are nesting edges $e = ab, e' = cd \in Q_\square$ in the vertex ordering σ' such that $a <_{\sigma'} c <_{\sigma'} d <_{\sigma'} b$. By assignment to the queue $Q_\square$, we have that endpoints of an edge in $Q_\square$ have different distances to r , allowing us to use Observation 5.4. Therefore, we know that there is some $i \in \mathbb{N}_0$ such that vertices a and c are in BFS-level i and vertices b and d are in BFS-level $(i + 1)$. Hence, we know that the pairs of vertices a, c and b, d are ordered the same in the vertex orderings σ and σ' , i.e., $a <_\sigma c$ and $d <_\sigma b$. We continue depending on the queue that we are considering.

If we are considering the queue $Q_<$, the fact that $\text{dist}(r, c) = i < i + 1 = \text{dist}(r, d)$ implies that the vertices of the edge e' are ordered $c <_\sigma d$. This implies that there is the rainbow $a <_\sigma c <_\sigma d <_\sigma b$ in the vertex ordering σ , a contradiction to Q being a queue.

If we are considering the queue $Q_>$, the fact that $\text{dist}(r, b) = i + 1 > i = \text{dist}(r, a)$ implies that the vertices of the edge e are ordered $b <_\sigma a$. This implies that there is the rainbow $d <_\sigma b <_\sigma a <_\sigma c$ in the vertex ordering σ , a contradiction to Q being a queue.

So, there are, either way, no nesting edges in the queue $Q_\square$ for the vertex ordering σ' .

We assume without loss of generality that $Q_=_$ contains at least two edges. Let $e = ab, e' = cd \in Q_=_$ be two different edges. Without loss of generality, assume that they are ordered $a <_{\sigma'} b$ and $c <_{\sigma'} d$. Since the edges are in the queue $Q_=_$ both endpoints of the edge e have equal distance to vertex v and both endpoints of the edge e' have equal distance to vertex r . In the case that all four vertices have the same distance to the vertex r , they are ordered equally in the vertex orderings σ and σ' . As Q is a queue for the vertex ordering σ , the edges do not nest, and the same holds for the vertex ordering σ' . If, however, the distance from vertex r to vertex a is not the same as to vertex c , the vertices c and d are either both to the left of vertex a or both to the right of vertex b , and thus, they do not nest in σ' . So, there are no nesting edges in the queue $Q_=_$ for the vertex ordering σ' .

For the bipartite case, we observe that for two vertices $u, v \in V(G)$ with $\text{dist}(r, u) = \text{dist}(r, v) < \infty$, they have to be assigned the same color in a proper 2-coloring of G . Hence, we have $uv \notin E(G)$. Therefore, no edge in the connected component of r in G is assigned to the queue $Q_=_$. If G is connected, this implies that we only need to create the queues $Q_<, Q_>$ for each queue in the vertex ordering σ , i.e., we have that the queue number of G with respect to σ' is at most $2k$. In the case that G is not connected, let $R \subseteq V(G)$ be the set of vertices in the connected component of r . Then, by σ' being an r -BFS-ordering, we have $R <_{\sigma'} V(G) \setminus R$. By the previous case, $2k$ queues are sufficient for the connected component of r and k queues are sufficient for $V(G) \setminus R$, as those vertices are ordered identically as in σ . Since we have $R <_{\sigma'} V(G) \setminus R$, we may reuse some k queues for the queues for $V(G) \setminus R$ as edges in the component of R are disjoint to edges in the subgraph $G - R$ in σ' . Hence, we again have that the queue number of G with respect to σ' is at most $2k$. $\square$

We use Lemma 5.1, to obtain a lower bound for the queue number in the case that we are able to provide a lower bound for the queue number with respect to BFS-orderings.

Corollary 5.5: *Let G be a graph and let r be a vertex of G . Then*

$$\frac{1}{3} \cdot \text{BFSqn}(r, G) \leq \text{qn}(G) \leq \text{BFSqn}(r, G).$$

If G is bipartite, we have $\frac{1}{2} \cdot \text{BFSqn}(r, G) \leq \text{qn}(G) \leq \text{BFSqn}(r, G)$.

A slightly – by a constant additive term – weaker result may be obtained by closely looking at a lemma by Dujmović, Joret, Micek, Morin and Ueckerdt [16, Lemma 8] for the case of general graphs. For the case of bipartite graphs, a similar slightly weaker result is presented by Förster, Kaufmann, Merker, Pupyrev and Raftopoulou [25, Lemma 10]. They derived the result from the more general result for graphs [16, Lemma 8], and it is not found in the conference version of the paper [26].

Proof. The upper bound is trivial, as every vertex ordering that contributes to $\text{BFSqn}(r, G)$ is considered for $\text{qn}(G)$ as well. Let σ be a vertex ordering of G such that $\text{qn}_\sigma(G) = \text{qn}(G)$. Consider the (r, σ) -BFS-ordering $\sigma' = \Rightarrow_{\text{BFS}}(r, \sigma, G)$. By Lemma 5.1, the queue number of G with respect to σ' is at most $3 \cdot \text{qn}_\sigma(G)$. At the same time, we have $\text{BFSqn}(r, G) \leq \text{qn}_{\sigma'}(G)$. We combine both inequations to get $\text{BFSqn}(r, G) \leq 3 \cdot \text{qn}_\sigma(G)$. We divide both sides by 3 and arrive at $\frac{1}{3} \cdot \text{BFSqn}(r, G) \leq \text{qn}_\sigma(G) = \text{qn}(G)$ by the choice of the vertex ordering σ .

The proof for bipartite graphs is similar, just replace 3 by 2 and use the bipartite case of Lemma 5.1. $\square$

5.2 Queue-BFS-orderings and Lex-BFS-orderings

Given the results for BFS-orderings, it makes sense to ask whether it is possible to further restrict the vertex orderings when asymptotically determining bounds for the queue number of a graph. For example, queue-BFS-orderings provide 1-queue layouts for trees [29]. Given that we are looking for a partition of edges such that each set is processable by a queue, it appears to be believable that queue-BFS-orderings might be worth looking into. There are various applications of Lex-BFS [9] and the vertex orderings considered in the proof of Wiechert's upper bound on the queue number of graphs with treewidth k are constructed using partition refinement as well. So, Lex-BFS-orderings, as a special case of queue-BFS-orderings, seem like another interesting candidate.

Let G be a graph, r be a vertex of G and σ be a vertex ordering of G . Similar to BFS-orderings, each queue-BFS-ordering of G and each Lex-BFS-ordering of G can be obtained by performing a queue-BFS or a Lex-BFS, respectively, while using some other vertex ordering as a tiebreaker, e.g., itself. We define the (r, σ) -queue-BFS-ordering of G , denoted $\Rightarrow_{\text{qBFS}}(r, \sigma, G)$, which is obtained by performing a queue-BFS on G with start vertex r and inserting neighbors of an explored vertex by their order in σ into the queue. We define the (r, σ) -Lex-BFS-ordering of G , denoted $\Rightarrow_{\text{LexBFS}}(r, \sigma, G)$, which is obtained by performing a Lex-BFS on G with start vertex r . In the case that there are multiple vertices with the lexicographically largest label when choosing the next vertex in line 7 of Algorithm 2.1, we pick the one furthest to the left in σ . We use the same notions of *original* and *producing* vertex orderings as defined for transforming a vertex ordering into a BFS-ordering. Similarly to the r -BFS-queue number for some start vertex $r \in V(G)$, we define the r -queue-BFS-queue number of G and the r -Lex-BFS-queue number of G . We denote the r -queue-BFS-queue number of G with start vertex r by

$$\text{qBFSqn}(r, G) = \min_{\sigma \text{ vertex ordering of } G} \text{qn}_{\Rightarrow_{\text{qBFS}}(r, \sigma, G)}(G)$$

and the r -Lex-BFS-queue number of G with start vertex r by

$$\text{LexBFSqn}(r, G) = \min_{\sigma \text{ vertex ordering of } G} \text{qn}_{\Rightarrow_{\text{LexBFS}}(r, \sigma, G)}(G).$$

The concepts of r -BFS-queue number, r -queue-BFS-queue number and r -Lex-BFS-queue number can be extended by minimizing over all start vertices to consider all BFS-orderings, queue-BFS-orderings and Lex-BFS-orderings, respectively. We denote the *BFS-queue number*, *queue-BFS-queue number*, and *Lex-BFS-queue number* of a graph G by $\text{BFSqn}(G)$, $\text{qBFSqn}(G)$ and $\text{LexBFSqn}(G)$, respectively. With every r -Lex-BFS-ordering being an r -queue-BFS-ordering and every r -queue-BFS-ordering being an r -BFS-ordering, we get certain relations of the corresponding versions of the queue number of a graph.

Observation 5.6: *Let G be a graph and r a vertex of G . Then $\text{qn}(G) \leq \text{BFSqn}(r, G) \leq \text{qBFSqn}(r, G) \leq \text{LexBFSqn}(r, G)$.*

However, there are families of graphs in which some queue-BFS-orderings and Lex-BFS-orderings can have pretty large rainbows, although the queue number of the family of graphs is constant, and they are produced by vertex orderings with constant-sized largest rainbows. We define a family $\mathcal{C}$ which contains a graph $C_{(k)}$ for every $k \in \mathbb{N}_+$. We define the graph

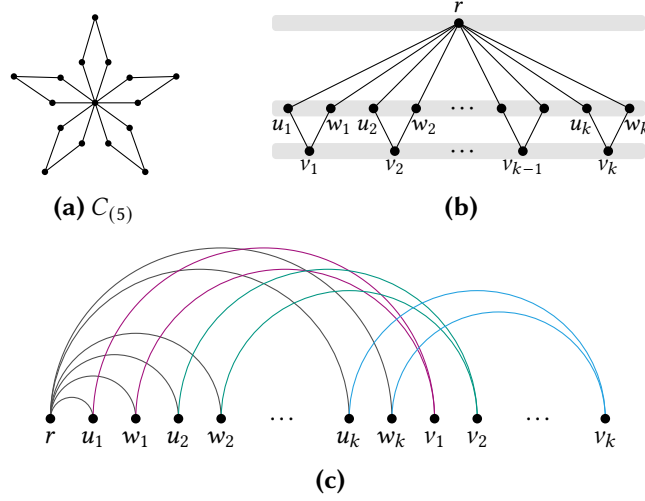


Figure 5.4: Upper Left: An illustration of the graph $C_{(5)}$ as defined before Lemma 5.7. Upper Right: A leveled-planar embedding of the graph $C_{(k)}$. Each level of the embedding is marked by a gray bar. Bottom: A 1-queue layout of the graph $C_{(k)}$. The different colors are supposed to improve clarity and do not indicate that edges are in different queues. All edges incident to the vertex r are gray. Other edges are colored based on the indices of the endpoints.

$C_{(k)} \in \mathcal{C}$ as a graph with k cycles of length four and each of the cycles sharing the same central vertex. More formally, we set $V(C_{(k)}) = \{r\} \cup \bigcup_{i \in [k]} \{u_i, v_i, w_i\}$ for pairwise distinct vertices r, u_i, v_i, w_i for all $i \in [k]$. The edge set is given by $E(C_{(k)}) = \bigcup_{i \in [k]} \{ru_i, u_i v_i, v_i w_i, w_i r\}$. See Figure 5.4a for an illustration of $C_{(5)}$. We observe that for every $k \in \mathbb{N}_+$, we have $\text{qn}(C_{(k)}) = 1$, by using a vertex ordering $\sigma = r, u_1, w_1, u_2, w_2, \dots, u_k, w_k, v_1, v_2, \dots, v_k$, see Figure 5.4c.

We show that there are Lex-BFS-orderings of graphs in $\mathcal{C}$ with arbitrarily large rainbows.

Lemma 5.7: *The family $\mathcal{C}$ has constant queue number, but for each $C_{(k)} \in \mathcal{C}$, there is a Lex-BFS-ordering σ' which contains a rainbow of size k , i.e., $\text{qn}_{\sigma'}(C_{(k)}) \geq k$.*

Note that this provides the same result for queue-BFS-orderings and BFS-orderings given the subset relations of those orderings. However, due to Lemma 5.1, we know that an original vertex ordering that produces such a BFS-ordering, also has a rainbow of size at least $\frac{k}{3}$. We show in Lemma 5.9 that this result of transforming a vertex ordering and increasing the size of largest rainbows by at most a constant factor cannot be generalized.

Proof. To show that graphs in $\mathcal{C}$ have constant queue number 1, we observe that $C_{(k)} \in \mathcal{C}$ is clearly leveled-planar with an embedding as depicted in Figure 5.4b. This allows us to apply Lemma 2.7 to obtain the result that every graph in $\mathcal{C}$ has queue number 1. The vertex ordering in Figure 5.4c is precisely the vertex ordering obtained by the proof of Lemma 2.7 using the leveled-planar embedding depicted in Figure 5.4b.

We use a Lex-BFS-ordering σ' obtained from a Lex-BFS that starts at the vertex r . As the neighborhood of r , i.e., $U \cup W$, is an independent set, we may choose their order arbitrarily. We process them in the order $w_1, w_2, \dots, w_k, u_k, \dots, u_2, u_1$. As each of the vertices v_i , for $i \in [k]$, has precisely the neighbors $N(v_i) = \{u_i, w_i\}$ and $w_i <_{\sigma'} u_i$, the vertices of V are ordered the same in terms of the indices as the vertices of W , i.e., $v_1 <_{\sigma'} v_2 <_{\sigma'} \dots <_{\sigma'} v_k$.

For every $i \in [k]$, we have the edge $u_i v_i \in E(C_{(k)})$. As the vertices in $U \cup V$ are ordered $u_k <_{\sigma'} \dots <_{\sigma'} u_2 <_{\sigma'} u_1 <_{\sigma'} v_1 <_{\sigma'} v_2 <_{\sigma'} \dots <_{\sigma'} v_k$, there is a rainbow of size k . This completes the proof. $\square$

We do have to note that when we consider a vertex ordering σ that produces the vertex ordering σ' in the proof of Lemma 5.7, then the vertex ordering σ does contain a decently large rainbow as well. To show this, we use a version of the Erdős-Szekeres theorem.

Theorem 5.8 (Erdős-Szekeres, 1935 [24]): *Let π be a sequence of n distinct integers. Then there is an increasing subsequence of length at least $\sqrt{n}$ or a decreasing subsequence of length at least $\sqrt{n}$.*

Being an r -BFS-ordering increases the maximum size of a rainbow at most by a constant factor (Lemma 5.1). Further, the tiebreaker is only relevant for vertices of the same distance to r . So we assume without loss of generality that σ is an r -BFS-ordering.

As we pick the vertices $u_1, \dots, u_k, w_1, \dots, w_k$ in a specific order, we use the tiebreaker for each of them. Thus, they are ordered the same in σ as in σ' , i.e., we have $w_1 <_{\sigma} w_2 <_{\sigma} \dots <_{\sigma} w_k <_{\sigma} u_k <_{\sigma} \dots <_{\sigma} u_2 <_{\sigma} u_1$. We do not use the tiebreaker for any of the vertices v_i . So they may be ordered arbitrarily in σ .

We consider the sequence π of the indices of the vertices v_i in σ . By the Erdős-Szekeres-Theorem 5.8, there is an increasing subsequence $\pi_{\nearrow}$ of length at least $\sqrt{k}$ or a decreasing subsequence $\pi_{\searrow}$ of length at least $\sqrt{k}$. In the case of the increasing subsequence $\pi_{\nearrow}$ of length at least $\sqrt{k}$, this leads to a rainbow of size at least $\sqrt{k}$ as the corresponding vertices v_i are ordered by increasing indices, and, thus, reversely to the corresponding vertices u_i . In the case of the decreasing subsequence $\pi_{\searrow}$ of length at least $\sqrt{k}$, this similarly leads to a rainbow of size at least $\sqrt{k}$ as the corresponding vertices v_i are ordered by decreasing indices, and, thus, reversely to the corresponding vertices w_i .

Either way, there is a rainbow of size at least $\sqrt{k}$. Hence, the initial vertex ordering was already significantly worse than an optimal queue layout.

Strengthening Lemma 5.7, this example can already be used to show that there is no constant coefficient α such that for every graph G , the queue number with respect to a Lex-BFS-ordering that is produced by an original vertex ordering σ is at most $\alpha \cdot \text{qn}_{\sigma}(G)$. We are able to make this observation by finding a vertex ordering $\tilde{\sigma}$ in which both the longest increasing subsequence and the longest decreasing subsequence have length $O(\sqrt{k})$. This leads to the queue number with respect to $\tilde{\sigma}$ being in $\Theta(\sqrt{k})$.

We now formally prove an even stronger statement, in which we use a family of graphs similar to $\mathcal{C}$ with layouts with constant queue number which, when used to produce Lex-BFS-orderings, leads to rainbows of unbounded size. Therefore, we are not able to make a statement similar to Corollary 5.5 using the same chain of arguments as for BFS-orderings. Note that the existence of corollaries similar to Corollary 5.5 for r -queue-BFS-queue number and r -Lex-BFS-queue number is not ruled out, we just note that the result cannot be achieved in the same way. We do however determine in Section 5.3 that if there is a function that bounds the r -Lex-BFS-queue number in terms of the queue number then it is at least exponential.

We change the length and the number of the cycles in each graph. We define a family $\mathcal{C}'$ which contains a graph $C'_{(k)}$ for every $k \in \mathbb{N}_+$. It contains 2^k cycles of length $2k + 2$ each. We label the cycles by index $i \in \{0, 1, \dots, 2^k - 1\}$ and usually consider i as the 0-1-bit-string with the corresponding binary value. Within the cycle i , we label the vertices $v_{i,j}$ for $j \in \{0, 1, \dots, 2k + 1\}$ such that the cycle i is $v_{i,0} v_{i,1} \dots v_{i,2k} v_{i,2k+1} v_{i,0}$. Note that the vertex

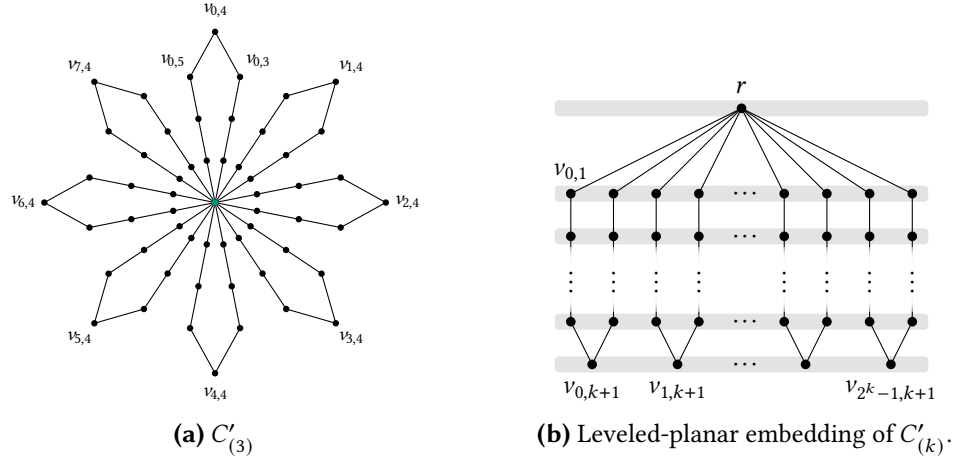


Figure 5.5: Left: Illustration of the graph $C'_{(3)} \in \mathcal{C}'$. To make it visible, the central vertex r is colored green.

Right: Illustration of a levelled-planar embedding of $C'_{(k)}$. Each level of the embedding is marked by a gray bar. The embedding is similar to the levelled-planar embedding of $C_{(k)}$, depicted in Figure 5.4b.

$v_{i,0} = r$ is the vertex shared by every cycle $i \in \{0, 1, \dots, 2^k - 1\}$. Other vertices are pairwise distinct between cycles. See Figure 5.5a for an illustration of $C'_{(3)}$. Similarly to $C_{(k)} \in \mathcal{C}$, we observe that $C'_{(k)} \in \mathcal{C}'$ is levelled-planar (see Figure 5.5b) and thus, $C'_{(k)}$ has queue number 1 for every $k \in \mathbb{N}_+$ by applying Lemma 2.7.

The concept that allows forming such a large rainbow in the produced Lex-BFS-ordering is switching blocks of vertices recursively to reverse the order of the block of vertices with index k in each cycle. In the produced Lex-BFS-ordering, they then form a rainbow of size 2^k together with the vertices with index $k + 1$ in each cycle, whose order does not change. The concept of switching blocks of vertices is depicted in Figure 5.6.

Lemma 5.9: For every $\ell \in \mathbb{N}_+$, there is a graph G with a vertex ordering σ and a vertex $r \in V(G)$ such that $\text{qn}_\sigma(G) \leq 3$, but the queue number with respect to the (r, σ) -Lex-BFS-ordering/ (r, σ) -queue-BFS-ordering $\sigma' = \Rightarrow_{\text{LexBFS}}(r, \sigma, G) = \Rightarrow_{\text{qBFS}}(r, \sigma, G)$ is at least $\text{qn}_{\sigma'}(G) \geq \ell$.

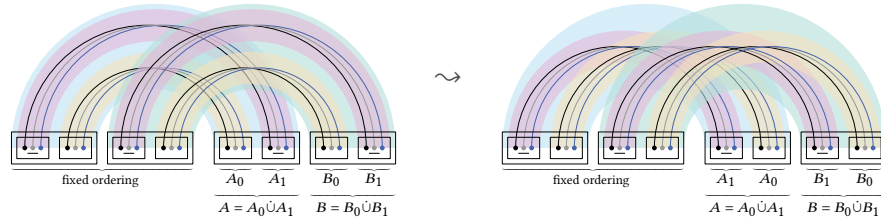


Figure 5.6: Illustrates the key concept behind the constructed vertex ordering in the proof of Lemma 5.9. The original vertex ordering is on the left-hand side. The produced queue-BFS-ordering, given that the first half of the vertices is fixed, e.g., because of other earlier vertices, is on the right-hand side. Observe that we have $A < B$ in both vertex orderings and the sets A_0, A_1, B_0, B_1 are internally ordered the same as well. Meanwhile, A_0, B_0 and A_1, B_1 switched places in the produced queue-BFS-ordering compared to the original vertex ordering, respectively.

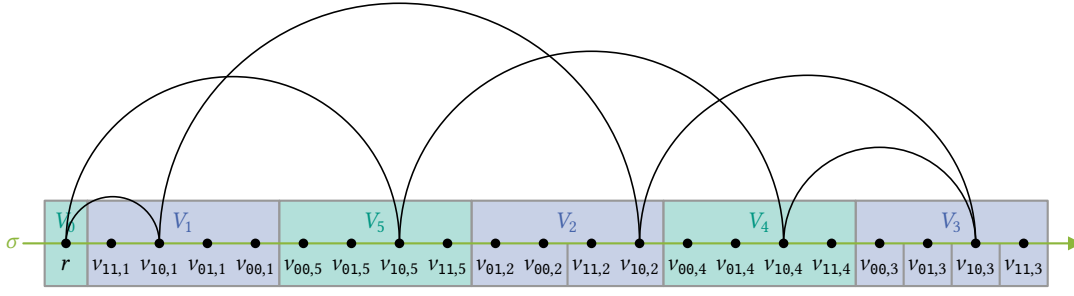


Figure 5.7: Illustrates the vertex ordering σ of $C'_{(2)}$ as constructed in the proof of Lemma 5.9 with one of the cycles of the graph $C'_{(2)}$.

It is possible to replace 2 in the definition of the graph $C'_{(k)}$ and in the proof with a constant $c \in \mathbb{N}_+$ to obtain a family of graphs with similar properties. For simplicity, we only consider the case $c = 2$. Further, note that the vertex ordering obtained by applying Lemma 2.7 to the leveled-planar embedding in Figure 5.5b is an r -Lex-BFS-ordering. Thus, we have – in contrast to the statement of the lemma – that $\text{LexBFSqn}(r, G) = 1$.

We give a short overview of the proof. We begin by choosing the graph G and constructing the vertex ordering σ . Firstly, we show that the queue number of G with respect to σ is indeed at most $2 + 1$. Secondly, we argue why the produced queue-BFS-ordering and Lex-BFS-ordering are indeed the same. We finish the proof by showing that G contains an ℓ -rainbow in σ' .

Proof. To simplify comparing the labels of the cycles in which vertices are contained, we start by introducing some notations. For a vertex $v \in V(C'_{(k)})$, $v \neq r$, let $\text{label}(v)$ denote the 0-1-bit-string of the unique cycle which v is on. We use the operator $<_{\text{lex}}$ to compare 0-1-bit-strings of the same length lexicographically, prioritizing the beginning of strings, e.g., $01 <_{\text{lex}} 10$. This is equivalent to comparing the numerical values of the binary strings but allows for more convenient operations on prefixes and suffixes. We use $\text{pre}(s, \ell)$ and $\text{post}(s, \ell)$ for the prefix and suffix of the string s of length $\ell \in \mathbb{N}_0$, respectively.

Let $k \in \mathbb{N}_+$ be an integer such that $2^k \geq \ell$. We choose the graph $G = C'_{(k)}$ and use a BFS-ordering σ with start vertex r which we define in the following. We partition the vertices of $C'_{(k)}$ by their indices in the cycles $V(C'_{(k)}) = V_0 \cup V_1 \cup \dots \cup V_{2k} \cup V_{2k+1}$ where set V_j contains the vertices with index j in each cycle for each $j \in \{0, \dots, 2k+1\}$, i.e., $V_j = \{v_{i,j} \mid i \in \{0, \dots, 2^k - 1\}\}$. Observe that the set V_0 consists of the single shared vertex r . The vertices in BFS-level $d \in [k+1]$ are precisely the vertices $V_d \cup V_{2k+2-d}$. However, for $d = k+1$, we have $d = 2k+2-d$ and thus, the BFS-level d consists of only the set V_d . We order the sets in σ for each BFS-level $d \in [k]$ as $V_d <_{\sigma} V_{2k+2-d}$. For each $d \in [k]$, we order the vertices of V_{2k+2-d} ascending by the index of their cycles. For each $d \in [k+1]$, we order the vertices of V_d based on the 0-1-bit-string of their corresponding cycles. We primarily order the vertices ascending by the prefix of length $d-1$ and secondarily descending by the suffix of length $k-(d-1)$. This means that there are 2^{d-1} blocks of vertices with the same prefix in the labels of their cycles, each containing $2^{k-(d-1)}$ vertices which are ordered descending by the 0-1-bit-string of their cycle. See Figure 5.7 for an example with the vertex ordering of $C'_{(2)}$.

We now assign the edges of $C'_{(k)}$ to three queues Q_0, Q_1, Q_2 . See Figure 5.8 for an illustration of the resulting assignment of the edges to the queues. All edges incident to V_0 or vertices of V_{2k+2-d} , for each $d \in [k]$, are assigned to the queue Q_2 . We assign the remaining edges depending on the label of their cycle and the sets $V_1, \dots, V_{k+1}$ their endpoints are in. For $i \in [k]$ and an edge e with endpoints in V_i and V_{i+1} , let s be the 0-1-bit-string of the unique

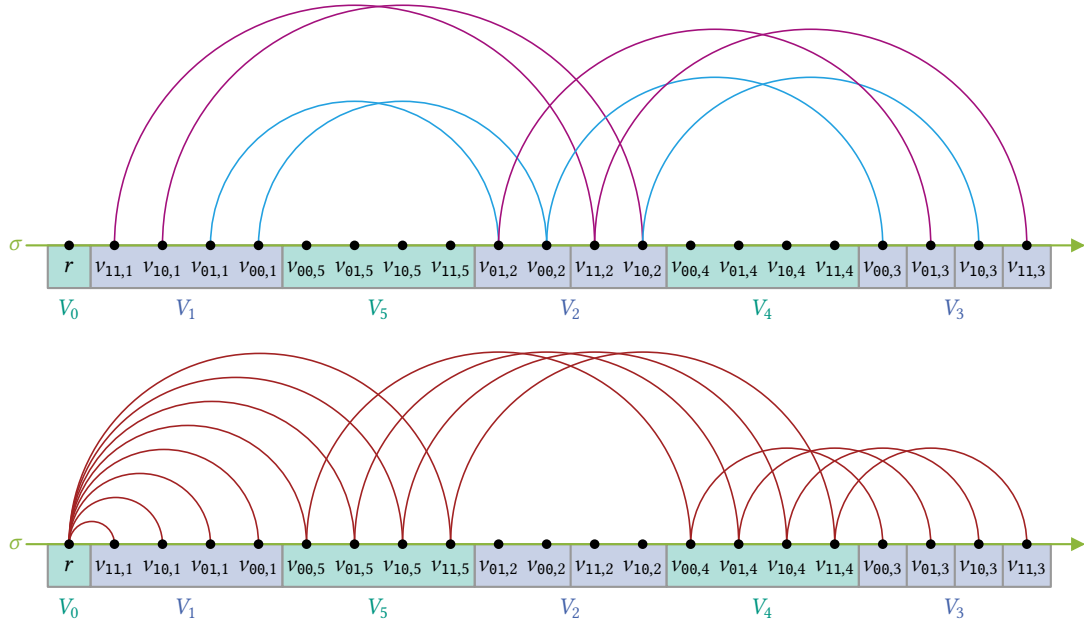


Figure 5.8: Illustrates the queue layout with respect to σ of $C'_{(2)}$ as constructed in the proof of Lemma 5.9. The lower part of the figure contains the queue Q_2 with edges colored red and the upper part of the figure contains the queues Q_0 and Q_1 with edges colored in blue and purple, respectively.

cycle containing e . Then, the edge e is assigned to the queue corresponding to the i -th bit of s , i.e., if the i -th bit is 0, edge e is assigned to queue Q_0 and otherwise to queue Q_1 . Here, we consider positions of bits in a bit-string 1-indexed from the beginning of the string.

We next aim to show that the queue number of $C'_{(k)}$ with respect to σ is at most $2 + 1$. As $C'_{(k)}$ is bipartite and connected, there are no edges with both endpoints having the same distance to r . As σ is an r -BFS-ordering, we use Observation 5.4 to obtain that we only have to consider edges between BFS-level i and BFS-level $(i + 1)$ for each $i \in \mathbb{N}_0$ when showing that there are no nesting edges in a queue. Note that this implies that there are no edges that nest with an edge incident to r , as r is the sole vertex in BFS-level 0. Let $i \in \mathbb{N}_+$. We observe that the edges in Q_2 form a 2^k -twist between BFS-level i and BFS-level $(i + 1)$ as the vertices are ordered the same with respect to their cycles in each BFS-level. If there are two edges $e = uv, e' = u'v'$ in Q_b , for some $b \in \{0, 1\}$, we show that they do not nest. We may assume without loss of generality that $u <_\sigma u'$, vertices u, u' are in BFS-level i and vertices v, v' are in BFS-level $(i + 1)$ (using Observation 5.4). Recall that $C'_{(k)}$ consists of cycles that only share the vertex r , and thus, we have that, as the edges $uv, u'v'$ are present, $\text{label}(u) = \text{label}(v)$ and $\text{label}(u') = \text{label}(v')$. As $u <_\sigma u'$, we know that it either holds that $\text{pre}(\text{label}(u), i - 1) <_{\text{lex}} \text{pre}(\text{label}(u'), i - 1)$ or that $\text{pre}(\text{label}(u), i - 1) = \text{pre}(\text{label}(u'), i - 1)$ and $\text{post}(\text{label}(u), k - i + 1) >_{\text{lex}} \text{post}(\text{label}(u'), k - i + 1)$.

In the case that $\text{pre}(\text{label}(u), i - 1) <_{\text{lex}} \text{pre}(\text{label}(u'), i - 1)$, we also have that

$$\text{pre}(\text{label}(v), i) = \text{pre}(\text{label}(u), i - 1) \cdot b <_{\text{lex}} \text{pre}(\text{label}(u'), i - 1) \cdot b = \text{pre}(\text{label}(v'), i)$$

and therefore, $v <_\sigma v'$.

In the case that $\text{pre}(\text{label}(u), i-1) = \text{pre}(\text{label}(u'), i-1)$ and $\text{post}(\text{label}(u), k-i+1) >_{\text{lex}} \text{post}(\text{label}(u'), k-i+1)$, we also have that

$$\text{pre}(\text{label}(v), i) = \text{pre}(\text{label}(u), i-1) \cdot b = \text{pre}(\text{label}(u'), i-1) \cdot b = \text{pre}(\text{label}(v'), i).$$

Furthermore, we have

$$b \cdot \text{post}(\text{label}(v), k-i) = \text{post}(\text{label}(u), k-i+1) >_{\text{lex}} \text{post}(\text{label}(u'), k-i+1) = b \cdot \text{post}(\text{label}(v'), k-i)$$

which implies $\text{post}(\text{label}(v), k-i) >_{\text{lex}} \text{post}(\text{label}(v'), k-i)$ and therefore, $v <_{\sigma} v'$.

In either case we additionally have the order $v <_{\sigma} v'$ and therefore, we know exactly that the endpoints of the edges e and e' are ordered $u <_{\sigma} u' <_{\sigma} v <_{\sigma} v'$ and thus, the edges e and e' do not nest. This yields $\text{qn}_{\sigma}(C'_{(k)}) \leq 2 + 1$ for every $k \in \mathbb{N}_+$.

We observe that every BFS-level is an independent set and that every vertex in BFS-level i , for some $i \in \mathbb{N}_+$ has at most one neighbor in BFS-level $(i+1)$. Therefore, the Lex-BFS does not have to resolve the case that there are multiple vertices available which have a common leftmost neighbor, and therefore, its structure simulates a queue for $C'_{(k)}$ and we have $\Rightarrow_{\text{qBFS}}(r, \sigma, C'_{(k)}) = \Rightarrow_{\text{LexBFS}}(r, \sigma, C'_{(k)}) = \sigma'$.

We now prove that $C'_{(k)}$ contains a 2^k -rainbow in σ' . We start by determining the vertex ordering σ' . As BFS-level 1 is an independent set, we use the tiebreaker σ for every but the last vertex in the neighborhood of r . Recall that the neighborhood of r is $N_{C'_{(k)}}(r) = V_1 \cup V_{2k+1}$ and V_1 is to the left of V_{2k+1} in σ with vertices of V_1 being ordered descending and vertices of V_{2k+1} being ordered ascending by their labels. In each BFS-level i for $2 \leq i \leq k$, each vertex has a unique neighbor in BFS-level $(i-1)$. Therefore, for each $i \in [d]$, we have in the BFS-level i with vertices $V_i \cup V_{2k+2-i}$ in the vertex ordering σ' first the vertices of V_i ordered descending and second the vertices of V_{2k+2-i} ordered ascending by their labels. Since $V_k < V_{k+2}$, the vertices in V_{k+1} are ordered by their neighbors in V_k , i.e., the vertices of V_k are ordered descending by their labels. Together with $V_{k+2} <_{\sigma'} V_{k+1}$ this gives the order $v_{0,k+2} <_{\sigma'} \dots <_{\sigma'} v_{2^k-1,k+2} <_{\sigma'} v_{2^k-1,k+1} <_{\sigma'} \dots <_{\sigma'} v_{0,k+1}$ of $V_{k+1} \cup V_{k+2}$. With $v_{j,k+1}v_{j,k+2} \in E(C'_{(k)})$ for each $j \in \{0, 1, \dots, 2^k-1\}$, we have a 2^k -rainbow in σ' , given by the edges between the vertex sets V_{k+2} and V_{k+1} . Hence, we conclude $\text{qn}_{\sigma'}(C'_{(k)}) \geq 2^k \geq \ell$. $\square$

Therefore, we know that there is no constant $c \in \mathbb{R}$ such that for every graph G with vertex ordering σ and start vertex r , it holds that $\text{qn}_{\Rightarrow_{\text{LexBFS}}(r, \sigma, G)}(G) \leq c \cdot \text{qn}_{\sigma}(G)$, like we show in Lemma 5.1 for BFS-orderings. Lemma 5.9 provides that there is not even a function f such that for every graph G with vertex ordering σ and start vertex r it holds $\text{qn}_{\Rightarrow_{\text{LexBFS}}(r, \sigma, G)}(G) \leq f(\text{qn}_{\sigma}(G))$. The same holds for $\text{qn}_{\Rightarrow_{\text{qBFS}}(r, \sigma, G)}(G)$.

Corollary 5.10: *There is no function f such that for every graph G , every vertex ordering σ of G and every start vertex $r \in V(G)$, we have $\text{qn}_{\Rightarrow_{\text{qBFS}}(r, \sigma, G)}(G) \leq f(\text{qn}_{\sigma}(G))$.*

Corollary 5.11: *There is no function f such that for every graph G , every vertex ordering σ of G and every start vertex $r \in V(G)$, we have $\text{qn}_{\Rightarrow_{\text{LexBFS}}(r, \sigma, G)}(G) \leq f(\text{qn}_{\sigma}(G))$.*

5.3 Comparing Queue Number and Lex-BFS-Queue Number

We just proved that the queue number of a graph with respect to a Lex-BFS-ordering is not bounded by a function of the queue number of an original vertex ordering that produces said Lex-BFS-ordering. However, this does not make any statement on how the queue number

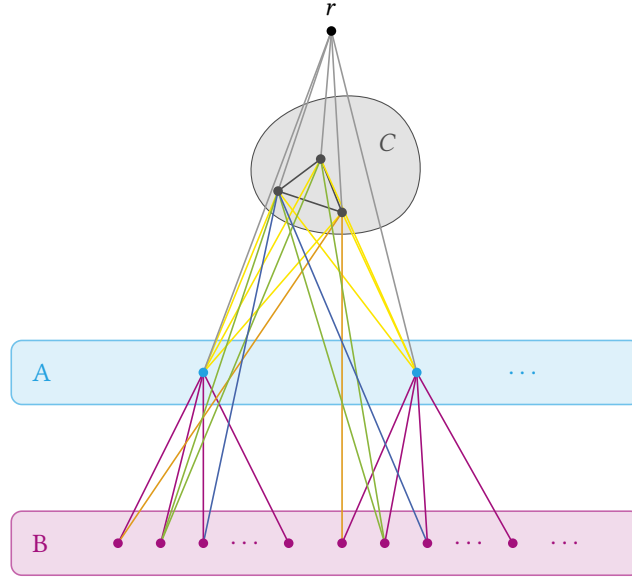


Figure 5.9: A partial sketch of Z_3 as constructed in Section 5.3. The edges in E_R are gray and the edges E_C in the clique C are in a darker gray. The edges in E_{CA} are yellow and the edges in E_{AB} are purple. The edges in E_{CB} are colored orange, green, or blue. The colors indicate that vertices in B which are incident to edges of the same color have the same set of neighbors in C .

and the Lex-BFS-queue number compare. Therefore, we want to establish a lower bound in the case that there is a function that bounds the Lex-BFS-queue number of a graph in terms of its queue number.

As an intermediate step, we define a family of chordal graphs and show that it has queue number linear in its clique number and r -Lex-BFS-queue number exponential in its clique number for some fixed start vertex r . Afterwards, we apply some modifications which allow Lex-BFS-orderings with arbitrary start vertices while guaranteeing that the queue number remains linear in the clique number and that the queue number with respect to a Lex-BFS-ordering is at least exponential in the clique number. This exponential bound is close to Wiechert's [37] upper bound on the queue number in terms of the treewidth of a graph. While our result does not show high queue number of the considered graphs, one could still derive asymptotic tightness of Wiechert's proof if a Lex-BFS-ordering is obtainable by Wiechert's construction of vertex orderings which allow the upper bound.

For every $k \in \mathbb{N}_+$, we define a graph Z_k on the vertex set $V(Z_k) = R \dot{\cup} C \dot{\cup} A \dot{\cup} B$, where $R = \{r\}$, $C = \{c_i \mid i \in [k]\}$, $A = \{a_i \mid i \in [2^k + 1]\}$ and $B = \{b_{S,i} \mid S \subseteq [k], i \in [2^k + 1]\}$. The vertices $R \cup C$ induce a clique and A and B are each independent sets with each vertex having a specific set of neighbors. See Figure 5.9 for a rough sketch of Z_k . More precisely, the edges of Z_k are defined by $E(Z_k) = E_R \cup E_C \cup E_{CA} \cup E_{CB} \cup E_{AB}$ with $E_R = \{rx \mid x \in C \cup B\}$, $E_C = \binom{C}{2}$, $E_{CA} = \{ca \mid c \in C, a \in A\}$, $E_{CB} = \{c_i b_{S,j} \mid i \in S \subseteq [k], j \in [2^k + 1]\}$ and $E_{AB} = \{a_i b_{S,i} \mid S \subseteq [k], i \in [2^k + 1]\}$. The edge sets E_{CB} and E_{AB} are chosen to guarantee that for every vertex $a \in A$ and subset of vertices $S \subseteq C$, there is at least one vertex in B that has the neighborhood $S \cup \{a\}$. This is the vertex $b_{S',i}$ where i is the index of vertex a and S' is the set

of indices of vertices in S . Also note that every vertex in B has exactly one neighbor in A and every vertex in A has exactly 2^k neighbors in B , i.e., E_{AB} consists of the edges of $|A| = 2^k + 1$ vertex-disjoint copies of stars $K_{1,2^k}$ with centers in A and leaves in B .

We show that every vertex ordering σ with $r <_\sigma C <_\sigma A <_\sigma B$ serves as a perfect elimination scheme. Since A is an independent set and every vertex of A induces a clique together with $R \cup C$, for every vertex $x \in C \cup A$, the neighbors of x which are to the left of x in σ induce a clique. Furthermore, we have that B is an independent set and that each vertex $b \in B$ has exactly one neighbor in A and some neighbors in C , i.e., $N(b)$ induces a clique. Hence, σ is indeed a perfect elimination scheme, and by Lemma 2.3, this implies that Z_k is chordal.

Observation 5.12: Z_k is chordal for every $k \in \mathbb{N}_+$.

As we want to put the r -Lex-BFS-queue number of Z_k into relation with its clique number and treewidth, we next determine the clique number of Z_k .

Lemma 5.13: Let $k \in \mathbb{N}_+$. The clique number of Z_k is $k + 2$.

Proof. Since A and $B \cup R$ are independent sets, every clique contains at most one vertex of each. Hence, a largest clique contains at most $|V(Z_k) \setminus (A \cup (B \cup R))| + 1 + 1 = |C| + 2 = k + 2$ vertices. At the same time, there is some vertex a in A , and we observe that the vertex set $\{a, r\} \cup C$ induces a clique in Z_k . Therefore, the clique number of Z_k is at least $k + 2$. $\square$

We continue by upper bounding the queue number of Z_k by upper bounding the r -queue-BFS-queue number of Z_k . Applying Corollary 2.9 provides asymptotic tightness of the result.

Lemma 5.14: Let $k \in \mathbb{N}_+$. Then $\text{qn}(Z_k) \leq \text{BFSqn}(r, Z_k) \leq \text{qBFSqn}(r, Z_k) \leq k + 2$.

Proof. We use a queue-BFS-ordering σ with $r <_\sigma A <_\sigma C <_\sigma B$. There is such a queue-BFS-ordering since $A \cup C$ is the neighborhood of r . We observe that $R \cup C$ is a vertex cover of the subgraph with edge set $E(Z_k) \setminus E_{AB} = E_R \cup E_C \cup E_{CA} \cup E_{CB}$. Hence, by Lemma 2.12, using $|R \cup C| = k + 1$ queues is sufficient for those edges. We put the edges of E_{AB} in a single additional queue and aim to show that there is no rainbow in σ with edges E_{AB} . Note that every vertex of B has exactly one neighbor in A and that this neighbor is its leftmost neighbor in σ , as $A <_\sigma C$. Therefore, the vertices of B are ordered by their neighbor in A . Furthermore, each edge of E_{AB} has one endpoint in B and one endpoint in A . Thus, for edges $e = ab, e' = a'b'$ with $a, a' \in A, b, b' \in B$, and $a \leq_\sigma a'$, this implies $b \leq_\sigma b'$, completing the order to $a \leq_\sigma a' <_\sigma b \leq_\sigma b'$. Hence, there is no rainbow in σ with edges E_{AB} . Together, we use $(k + 1) + 1 = k + 2$ queues. The other inequalities are obtained by Observation 5.6. $\square$

We finish determining the graph parameters required for our result by proving a lower bound on the r -Lex-BFS-queue number.

Lemma 5.15: Let $k \in \mathbb{N}_+$. Then $\text{LexBFSqn}(r, Z_k) \geq 2^k$.

What we are essentially doing, is finding a matching containing most of the vertices of A such that their partners in B are ordered reversely. This then serves as a rainbow of the desired size. We do this by maintaining boxes of vertices of B whose vertices each appear consecutively in a Lex-BFS-ordering of Z_k . We use each of those boxes to find a neighbor for one vertex of A .

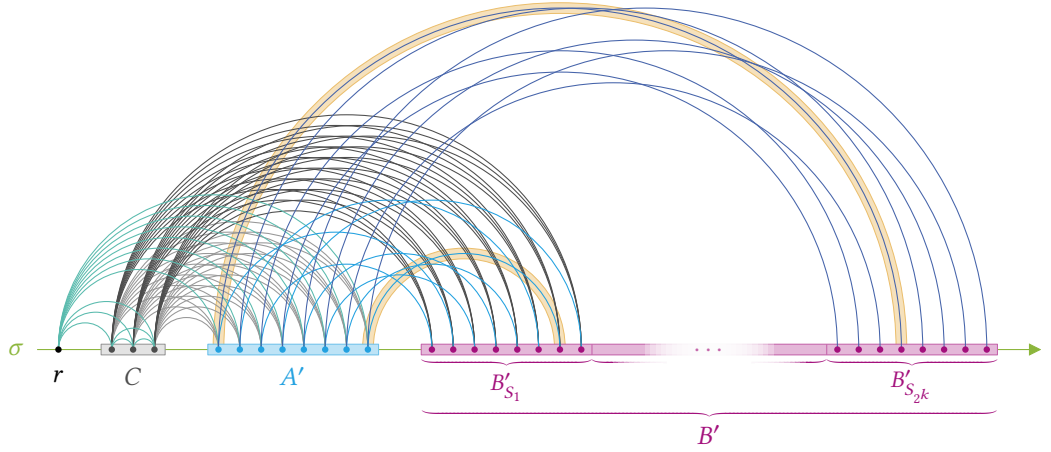


Figure 5.10: An example how the vertex ordering of the graph induced by $R \cup C \cup A' \cup B'$ of Z_k could look in the proof of Lemma 5.15. For the first and last vertex of A' , the incident edges which are chosen for the rainbow are highlighted with an orange background. The colors of the edges do not correspond to assignments to queues and are supposed to improve clarity.

Proof. Let σ be an r -Lex-BFS-ordering of Z_k . We aim to show that there is a 2^k -rainbow in σ . As the neighborhood of r is $N(r) = C \cup A$, we have $r <_\sigma C \cup A <_\sigma B$. Since vertices of A are defined symmetrically, regardless of their indices, we assume without loss of generality that the vertices of A are ordered $a_{2^k+1} <_\sigma a_{2^k} <_\sigma \dots <_\sigma a_1$.

Claim: *There is at most one vertex $\tilde{a} \in A$ such that $C \not<_\sigma \tilde{a}$.*

Proof. Suppose for contradiction that there are two different vertices $\tilde{a}, \hat{a}$ such that $C \not<_\sigma \tilde{a}$ and $C \not<_\sigma \hat{a}$. Then the rightmost vertex $c \in C$ is to the right of both of them. In notation, we have $\{\tilde{a}, \hat{a}\} <_\sigma c$. By Lemma 2.5, every Lex-BFS-ordering of a chordal graph is a perfect elimination scheme. However, we have $c\tilde{a}, c\hat{a} \in E(Z_k)$ and $\tilde{a}\hat{a} \notin E(Z_k)$ which is a contradiction to σ being a perfect elimination scheme as $\{\tilde{a}, \hat{a}\} <_\sigma c$. Hence, the claim follows. $\square$

So, we have some $A' \subseteq A$ with $|A'| = |A| - 1 = 2^k$ and $C <_\sigma A'$. Let $\tilde{a} \in A \setminus A'$ be the excluded vertex which we pick to be leftmost in the case $C <_\sigma A$. We consider $B' = B \setminus N(\tilde{a})$ instead of B for the rest of our considerations. As vertices of A are ordered descending by indices, we know $\tilde{a} = a_{2^k+1}$. In particular, this implies $A' = \{a_i \mid i \in [2^k]\}$ and $B' = \{b_{S,i} \mid S \subseteq [k], i \in [2^k]\}$. As the vertices of B' are an independent set and of maximum distance to r in Z_k , they are lexicographically ordered by all of their neighbors. Since $C <_\sigma A'$, the vertices of B' are primarily ordered by their sets of neighbors in C and secondarily by their neighbors in A' . That is, for each $S \subseteq [k]$, the vertices of B' whose neighborhood in C consists of exactly the vertices with indices in S , appear as a block in σ . We denote this set of vertices in B' by $B'_S = \{b \in B' \mid N(b) \cap C = \{c_i \in C \mid i \in S\}\} = \{b_{S,i} \mid i \in [2^k]\}$ for each set of indices $S \subseteq [k]$. Let $S_1, \dots, S_{2^k} \subseteq [k]$ such that $B'_{S_1} <_\sigma \dots <_\sigma B'_{S_{2^k}}$ while noting that $\{S_i \mid i \in [2^k]\} = 2^{[k]}$. See Figure 5.10 for how the vertex ordering of the graph could partially look like.

For every $i \in [2^k]$, the vertex $a_i \in A'$ has the neighbor $b_{S_i,i} \in B'_{S_i} \subseteq B'$. Those vertices are pairwise distinct, as we have $S_i \neq S_j$ for $i \neq j$. The sets S_i determine the order of the chosen vertices. The chosen vertices are ordered $a_{2^k} <_\sigma \dots <_\sigma a_1 <_\sigma b_{S_{1,1}} <_\sigma \dots <_\sigma b_{S_{2^k,2^k}}$ and with edges $a_i b_{S_i,i}$ for every $i \in [2^k]$, we have a 2^k -rainbow in σ . This completes the proof. $\square$

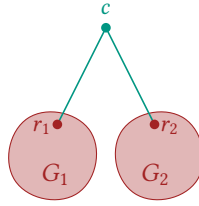


Figure 5.11: Illustrates the cherry graph $\text{CH}(r, G)$ where graphs G_1 and G_2 are the copies of G , and vertices r_1 and r_2 are the vertices corresponding to $r \in V(G)$ in the copies G_1 and G_2 , respectively.

If there is a function that bounds the r -Lex-BFS-queue number in terms of the queue number for every start vertex r , this example shows that it is asymptotically at least exponential with base 2.

5.4 Cherries

Now we want to extend the graph Z_k to still satisfy the properties proven for Z_k while removing the restriction to a specific start vertex to observe large rainbows. We define the more general notion of the cherry modification of a graph. For some graph G with a vertex $r \in V(G)$, we define the *cherry graph*, denoted $\text{CH}(r, G)$, which consists of two copies of G with an additional vertex c which is adjacent to vertex r of each copy of G . An abstract illustration of the concept is visible in Figure 5.11.

Given the definition of a cherry, we want to show that our results proven for Z_k extend to the cherry graph $\text{CH}(r, Z_k)$. We start by proving that cherry graphs inherit chordality.

Lemma 5.16: *Let G be a chordal graph and let $r \in V(G)$ be a vertex of G . Then the cherry graph $\text{CH}(r, G)$ is chordal.*

Proof. Let G_1 and G_2 be the copies of G with copies r_1, r_2 of vertex r in $\text{CH}(r, G)$ and let c be the additional vertex. Since $\text{CH}(r, G) - cr_1 - cr_2$ is the vertex-disjoint union of a single vertex and two copies of G , it is a vertex-disjoint union of chordal graphs, and hence, is chordal itself. Therefore, the edges cr_1 and cr_2 are between c and different connected components of $\text{CH}(r, G) - cr_1 - cr_2$. Thus, they do not introduce new cycles to the graph and $\text{CH}(r, G)$ is chordal since every induced cycle still has length 3. $\square$

We continue with the clique number of cherry graphs.

Lemma 5.17: *Let G be a graph that contains an edge and let $r \in V(G)$ be a vertex of G . The clique number of the cherry graph $\text{CH}(r, G)$ is $\omega(G)$.*

Proof. Since $\text{CH}(r, G)$ contains a copy of G , the clique number of $\text{CH}(r, G)$ is at least $\omega(G)$.

Every clique in $\text{CH}(r, G)$ does not contain vertices of both copies of G as there are no edges across copies. Hence, every clique contains at most vertices of one copy of G and the vertex c . If the vertex c is part of a largest clique, the clique has order at most $2 \leq \omega(G)$ since c has exactly one neighbor in each copy of G . Otherwise, the clique has order at most $\omega(G)$. $\square$

We now want to give an upper bound on the queue number of cherry graphs.

Lemma 5.18: *Let G be a graph that contains an edge and let $r \in V(G)$ be a vertex of G . The queue number of the cherry graph $\text{CH}(r, G)$ is at most $\min(\text{qn}(G) + 1, \text{BFSqn}(r, G))$.*

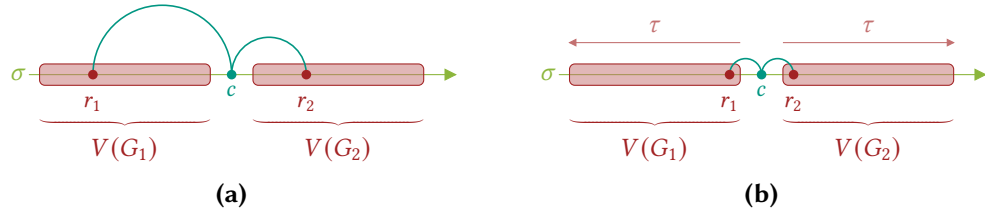


Figure 5.12: Illustrates the two vertex orderings considered in the proof of Lemma 5.18. The left vertex ordering (a) is the one for the upper bound of $\text{qn}(G) + 1$. The right vertex ordering (b) is the one for the upper bound of $\text{BFSqn}(G)$.

Proof. To provide an upper bound for the queue number of $\text{CH}(r, G)$, we use a vertex ordering σ of $\text{CH}(r, G)$ where we firstly have a copy of G , then the vertex c and finally the second copy of G . This allows us to use an optimal queue layout of G twice while using the same queues without having to worry about nesting edges between the copies. This requires $\text{qn}(G)$ queues. We use an additional queue for the edges incident to c and obtain a $(\text{qn}(G) + 1)$ -queue layout of $\text{CH}(r, G)$. See also Figure 5.12a.

Let τ be an r -BFS-ordering of G which admits a $\text{BFSqn}(r, G)$ -queue layout. If we order the second copy of G as in τ and the first copy reversely, we observe that the copies of vertex r are the vertices which are next to vertex c in the chosen vertex ordering for $\text{CH}(r, G)$. See also Figure 5.12b. Thus, there are no edges that nest below edges incident to c since there are no vertices between their endpoints by the choice of the vertex orderings within the copies of G . Furthermore, there is no edge which nests above an edge incident to vertex c since there are no edges across copies of G . Hence, we can save the additional queue for the edges incident to vertex c by inserting these edges into an arbitrary queue of a layout for G . $\square$

We finish by providing a lower bound for the Lex-BFS-queue number of cherry graphs.

Lemma 5.19: *Let G be a connected graph and let $r \in V(G)$ be a vertex of G . The Lex-BFS-queue number $\text{LexBFSqn}(\text{CH}(r, G))$ of the cherry graph $\text{CH}(r, G)$ is at least $\text{LexBFSqn}(r, G)$.*

Note that the statement and the following proof work as well for the corresponding statements for queue-BFS-queue number and BFS-queue number: $\text{qBFSqn}(\text{CH}(r, G)) \geq \text{qBFSqn}(r, G)$ and $\text{BFSqn}(\text{CH}(r, G)) \geq \text{BFSqn}(r, G)$.

Proof. No matter the start vertex of the Lex-BFS, in one of the copies of G , the vertex corresponding to vertex r is the first vertex of the copy of G which is encountered. Therefore, the vertices of the copy are ordered as in an r -Lex-BFS-ordering of G . This completes the proof. $\square$

Having proven estimations for the parameters determined for Z_k , we summarize our results in the following theorem using that G is clearly connected and contains an edge.

Theorem 5.20: *For each integer $\kappa \geq 2$, there is a chordal graph with clique number $\kappa + 1$, treewidth κ , queue number at most $\kappa + 1$ and Lex-BFS-queue number at least $2^{\kappa-1}$.*

Proof. We use the cherry graph $C = \text{CH}(r, Z_k)$ with $k = \kappa - 1$ and start vertex $r \in V(Z_k)$. By Observation 5.12 and Lemma 5.16, we have that C is chordal. Lemmas 5.13 and 5.17 yield that the clique number of C is $k + 2 = \kappa + 1$. By Lemma 2.2, the treewidth of a chordal graph is one less than the clique number, i.e., we have $\text{tw}(C) = \kappa$. Using $\text{BFSqn}(r, Z_k) \leq k + 2$ by Lemma 5.14, lets us apply Lemma 5.18 to obtain $\text{qn}(C) \leq \text{BFSqn}(r, Z_k) \leq k + 2 = \kappa + 1$. Finally, applying Lemmas 5.15 and 5.19 yields $\text{LexBFSqn}(C) \geq \text{LexBFSqn}(r, Z_k) \geq 2^k = 2^{\kappa-1}$. $\square$

The theorem yields that if there is a function that bounds the Lex-BFS-queue number in terms of the queue number, then it is at least exponential with base 2.

Corollary 5.21: *If the Lex-BFS-queue number is bounded by a function of the queue number, i.e., if there is a function f such that it holds for every graph G that $\text{LexBFSqn}(G) \leq f(\text{qn}(G))$, then we know $f(n) \in \Omega(2^n)$.*

We consider it possible that there is a family of graphs, each with treewidth k and Lex-BFS-queue number in $\Omega(2^k)$ such that there is a Lex-BFS-ordering which can be constructed in a way such that it is considered by the proof of Wiechert [37], thus showing the tightness of the proof. Note that this would only show that there is some vertex ordering with large rainbows, prohibiting an asymptotically better bound considering all vertex orderings constructible in the described way. The queue number of graphs with treewidth k may as well still be bounded by some function in $o(2^k)$. Most importantly, assuming that Wiechert's bound may be improved by more than a constant factor, Theorem 5.20 yields that the vertex orderings considered for a new upper bound may not contain a Lex-BFS-ordering for every graph.

We generally know that a function that bounds the r -Lex-BFS-queue number in terms of the queue number also bounds the Lex-BFS-queue number in terms of the queue number. Using the results of this section, we are able to prove a slightly weaker result for the reverse direction.

Theorem 5.22: *If there is a function f such that it holds for every graph G that $\text{LexBFSqn}(G) \leq f(\text{qn}(G))$, then there is a function g such that for every graph G and vertex $r \in V(G)$ it holds that $\text{LexBFSqn}(r, G) \leq g(\text{qn}(G))$.*

Similarly to Lemma 5.19, we are able to analogously prove the corresponding result for the queue-BFS-queue number (and the BFS-queue number).

Proof. Suppose for contradiction that there is such a function f and that for every function g there is a graph G and a vertex $r \in V(G)$ such that $\text{LexBFSqn}(r, G) > g(\text{qn}(G))$. We assume without loss of generality that the function f is non-decreasing, i.e., it holds for every $x \in \mathbb{N}_0$ that $f(x+1) \geq f(x)$. Otherwise, we modify the function to be $f_{\text{new}}(x) = \max\{f(a) \mid a \in \mathbb{N}_0, a \leq x\}$ which serves as an upper bound as well since it holds $\text{LexBFSqn}(G) \leq f(\text{qn}(G)) \leq f_{\text{new}}(\text{qn}(G))$ for every graph G .

Let $\hat{f}$ be the function given by $\hat{f}(n) = f(n+1)$ for every $n \in \mathbb{N}_0$. By assumption, there is a graph G and a vertex $r \in V(G)$ such that $\text{LexBFSqn}(r, G) > \hat{f}(\text{qn}(G))$. We may assume without loss of generality that this graph is connected. Let $C = \text{CH}(r, G)$ be the cherry graph as defined in Section 5.4 for graph G and vertex r . By Lemmas 5.18 and 5.19, we have $\text{qn}(C) \leq \text{qn}(G) + 1$ and $\text{LexBFSqn}(C) \geq \text{LexBFSqn}(r, G)$. Putting everything together, we get:

$$\text{LexBFSqn}(C) \geq \text{LexBFSqn}(r, G) > \hat{f}(\text{qn}(G)) = f(\text{qn}(G) + 1) \geq f(\text{qn}(C)).$$

This is a contradiction to $\text{LexBFSqn}(C) \leq f(\text{qn}(C))$, and thus, completes the proof. $\square$

This theorem means that we can prove the seemingly simpler statement that the Lex-BFS-queue number is bounded by a function of the queue number and additionally get the stronger version that there is a function of the queue number that bounds the r -Lex-BFS-queue number for every start vertex r . If such a function exists, this implies that to prove that a family of graphs has unbounded queue number, it is sufficient to show that it has unbounded r -Lex-BFS-queue number for some fixed start vertex r which we may pick individually for each graph of the family.

6 Conclusion

In this thesis, we looked at the stack number and queue number of some graphs that appear interesting with respect to Open Problem 1.2. In particular, we determined a family of graphs with bounded stack number and unbounded treewidth for which we do not know an easy constant upper bound on the queue number. We believe that there is a chance that this family has unbounded queue number but were unfortunately unable to prove it. We further showed that restricting the vertex orderings considered for the queue number to BFS-orderings increases the queue number of a graph at most by a constant factor, even when further restricting to a fixed start vertex. Hence, it can be used as a tool to asymptotically determine the queue number of arbitrary graph classes. We especially want to reiterate the open problem whether there is a function that bounds the queue number in terms of the stack number since it was the motivation for this thesis.

Open Problem 1.2: *Is the queue number bounded by a function of the stack number?*

We now give some questions that arose during the work on this thesis and could be a base for future research.

The most intriguing question is that of the queue number of the graphs $H'_s{}^P$ from Chapter 4. If it is super-constant, the result would, together with Corollary 4.37, solve Open Problem 1.2 in the negative.

Question 6.1: *Is there a constant $c \in \mathbb{N}_+$ such that the queue number of $H'_s{}^P$ is at most c for every $s \in \mathbb{N}_0$?*

For a better understanding, it could be helpful to look at the queue number of the intermediate graphs that we considered on our way to $H'_s{}^P$ and tighten our bounds. In Chapter 4, we first considered the graph H_r and its supergraph H_r^P , whereby $H'_s{}^P$ is an induced subgraph of H_{2s}^P .

Question 6.2: *What is the queue number of H_r and H_r^P ?*

To answer the last two questions it could be useful to look at our work on comparing the r -BFS-queue number to the queue number of a graph since it reduces the number of vertex orderings that have to be considered for proving a lower bound for the queue number of these graphs. It is also interesting to consider whether there are other ways to restrict the queue number while preserving being bounded by a function of the queue number. We ask whether there is such a function for the queue-BFS-queue number and the Lex-BFS-queue number. Of course, one could consider completely different kinds of restrictions on the vertex orderings.

Question 6.3: *Is the queue-BFS-queue number bounded by a function of the queue number?*

Question 6.4: *Is the Lex-BFS-queue number bounded by a function of the queue number?*

Note that Question 6.4 being answered in the positive implies that Question 6.3 is answered in the positive as it holds for every graph G that $\text{qbFSqn}(G) \leq \text{LexBFSqn}(G)$. With Theorem 5.22, the question of existence of such a function is equivalent to the existence of such a function when additionally restricting to a fixed start vertex. This seems to be a stronger result since it further limits the vertex orderings that require consideration.

We only consider restricting the vertex orderings for the queue number. It would be interesting if there are similar results for the stack number for example when restricting the vertex orderings to those obtainable by a depth-first search.

When proving that if there is a function that bounds the Lex-BFS-queue number in terms of the queue number, it is at least exponential, we constructed a graph with treewidth κ and Lex-BFS-queue number at least $2^{\kappa-1}$. Since this somewhat resembles the upper bound on the queue number given by Wiechert [37] for graphs of bounded treewidth, it leads us to consider whether a similar graph could show tightness of Wiechert's proof.

Question 6.5: *Is the proof of Wiechert's result [37] tight?*

We do not ask whether the bound is tight, this is a separate question, but rather whether there are chordal graphs with treewidth k and vertex orderings constructible in Wiechert's proof while having rainbows of size $\Theta(2^k)$. We believe that there is such a family and that it can be obtained using a construction similar to the one we used in Section 5.3. But of course tightening the gap between the bounds of the queue number of graphs with bounded treewidth would be even more exciting. If there is such a family that shows that Wiechert's proof is tight, one would need some new ideas that avoid Lex-BFS-orderings to asymptotically improve Wiechert's bound.

Bibliography

- [1] Michael Bekos, Michael Kaufmann, Fabian Klute, Sergey Pupyrev, Chrysanthi Raftopoulou, and Torsten Ueckerdt. “Four pages are indeed necessary for planar graphs”. In: *Journal of Computational Geometry* Volume 11 (Aug. 2020), pp. 332–353. DOI: [10.20382/jocg.v11i1a12](https://doi.org/10.20382/jocg.v11i1a12).
- [2] Claude Berge. “Some Classes of Perfect Graphs”. In: *Graph theory and theoretical physics*. Edited by Frank Harary. Academic Press, 1967, pp. 155–166. URL: <https://archive.org/details/graphtheorytheor0000hara>.
- [3] Frank Bernhart and Paul C Kainen. “The book thickness of a graph”. In: *Journal of Combinatorial Theory, Series B* Volume 27 (1979), pp. 320–331. ISSN: 0095-8956. DOI: [10.1016/0095-8956\(79\)90021-2](https://doi.org/10.1016/0095-8956(79)90021-2).
- [4] Robin Blankenship and Bogdan Oporowski. *Drawing subdivisions of complete and complete bipartite graphs on books*. Department of Mathematics, Louisiana State University, 1999. URL: <https://web.archive.org/web/20070824003537/https://www.math.lsu.edu/~preprint/1999/rbbo991.ps>.
- [5] Robin Leigh Blankenship. “Book embeddings of graphs”. Louisiana State University, 2003. DOI: [10.31390/gradschool_dissertations.3734](https://doi.org/10.31390/gradschool_dissertations.3734).
- [6] Thomas Bläsius, Emil Dohse, Deborah Haun, and Laura Merker. *Product Structure and Treewidth of Hyperbolic Uniform Disk Graphs*. 2026. arXiv: [2603.18997](https://arxiv.org/abs/2603.18997).
- [7] Thomas Bläsius, Emil Dohse, Deborah Haun, and Laura Merker. “Product Structure and Treewidth of Hyperbolic Uniform Disk Graphs”. In: *42nd International Symposium on Computational Geometry (SoCG 2026)*. Edited by Hee-Kap Ahn, Michael Hoffmann, and Amir Nayyeri. Vol. 367. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2026, 18:1–18:17. ISBN: 978-3-95977-418-5. DOI: [10.4230/LIPIcs.SocG.2026.18](https://doi.org/10.4230/LIPIcs.SocG.2026.18).
- [8] Jonathan F. Buss and Peter W. Shor. “On the pagenummer of planar graphs”. In: *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*. New York, NY, USA: Association for Computing Machinery, 1984, pp. 98–100. ISBN: 0897911334. DOI: [10.1145/800057.808670](https://doi.org/10.1145/800057.808670).
- [9] Derek G. Corneil. “Lexicographic Breadth First Search – A Survey”. In: *Graph-Theoretic Concepts in Computer Science*. Edited by Juraj Hromkovič, Manfred Nagl, and Bernhard Westfechtel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 1–19. ISBN: 978-3-540-30559-0. DOI: [10.1007/978-3-540-30559-0_1](https://doi.org/10.1007/978-3-540-30559-0_1).
- [10] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer International Publishing, 2015. DOI: [10.1007/978-3-319-21275-3](https://doi.org/10.1007/978-3-319-21275-3).
- [11] Josep Díaz, Jordi Petit, and Maria Serna. “A survey of graph layout problems”. In: *ACM Comput. Surv.* Volume 34 (Sept. 2002), pp. 313–356. ISSN: 0360-0300. DOI: [10.1145/568522.568523](https://doi.org/10.1145/568522.568523).

- [12] Reinhard Diestel. *Graph Theory*. 6th ed. Springer Berlin, Heidelberg, 2025. DOI: 10.1007/978-3-662-70107-2.
- [13] G. A. Dirac. “On rigid circuit graphs”. In: *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* Volume 25 (Apr. 1961), pp. 71–76. ISSN: 1865-8784. DOI: 10.1007/BF02992776.
- [14] Vida Dujmović. “Graph Product Structure Theory with Applications”. In: *Courses in Discrete and Computational Geometry*. Edited by János Pach and Géza Tóth. Cham: Springer Nature Switzerland, 2026, pp. 3–44. ISBN: 978-3-032-10503-5. DOI: 10.1007/978-3-032-10503-5_1.
- [15] Vida Dujmović, David Eppstein, Robert Hickingbotham, Pat Morin, and David R. Wood. “Stack-Number is Not Bounded by Queue-Number”. In: *Combinatorica* Volume 42 (Apr. 2022), pp. 151–164. ISSN: 1439-6912. DOI: 10.1007/s00493-021-4585-7.
- [16] Vida Dujmović, Gwenaél Joret, Piotr Micek, Pat Morin, Torsten Ueckerdt, and David R. Wood. “Planar Graphs Have Bounded Queue-Number”. In: *J. ACM* Volume 67 (Aug. 2020). ISSN: 0004-5411. DOI: 10.1145/3385731.
- [17] Vida Dujmovic, Pat Morin, and David R. Wood. “Layout of Graphs with Bounded Tree-Width”. In: *SIAM Journal on Computing* Volume 34 (2005), pp. 553–579. DOI: 10.1137/S0097539702416141.
- [18] Vida Dujmovic and David R. Wood. “Graph Treewidth and Geometric Thickness Parameters”. In: *Discrete & Computational Geometry* Volume 37 (May 2007), pp. 641–670. ISSN: 1432-0444. DOI: 10.1007/s00454-007-1318-7.
- [19] Vida Dujmović and David R. Wood. “On Linear Layouts of Graphs”. English. In: *Discrete Mathematics & Theoretical Computer Science* Volume vol. 6:2 (Jan. 2004). ISSN: 1365-8050. DOI: 10.46298/dmtcs.317.
- [20] Vida Dujmović and David R. Wood. “Stacks, Queues and Tracks: Layouts of Graph Subdivisions”. English. In: *Discrete Mathematics & Theoretical Computer Science* Volume Vol. 7 (Jan. 2005). ISSN: 1365-8050. DOI: 10.46298/dmtcs.346.
- [21] David Eppstein. *Separating Geometric Thickness from Book Thickness*. 2001. arXiv: math/0109195.
- [22] David Eppstein. “Separating Thickness from Geometric Thickness”. In: *Graph Drawing*. Edited by Michael T. Goodrich and Stephen G. Kobourov. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 150–162. ISBN: 978-3-540-36151-0. DOI: 10.1007/3-540-36151-0_15.
- [23] David Eppstein, Robert Hickingbotham, Laura Merker, Sergey Norin, Michał T. Seweryn, and David R. Wood. “Three-Dimensional Graph Products with Unbounded Stack-Number”. In: *Discrete & Computational Geometry* Volume 71 (June 2024), pp. 1210–1237. ISSN: 1432-0444. DOI: 10.1007/s00454-022-00478-6.
- [24] P. Erdős and G. Szekeres. “A combinatorial problem in geometry”. In: *Compositio Mathematica* Volume 2 (1935), pp. 463–470. URL: https://www.numdam.org/item/CM_1935__2__463_0/.
- [25] Henry Förster, Michael Kaufmann, Laura Merker, Sergey Pupyrev, and Chrysanthi Raftopoulou. *Linear Layouts of Bipartite Planar Graphs*. 2023. arXiv: 2305.16087.

-
- [26] Henry Förster, Michael Kaufmann, Laura Merker, Sergey Pupyrev, and Chrysanthi Raftopoulou. “Linear Layouts of Bipartite Planar Graphs”. In: *Algorithms and Data Structures*. Edited by Pat Morin and Subhash Suri. Cham: Springer Nature Switzerland, 2023, pp. 444–459. ISBN: 978-3-031-38906-1. DOI: [10.1007/978-3-031-38906-1_29](https://doi.org/10.1007/978-3-031-38906-1_29).
- [27] Joseph L. Ganley and Lenwood S. Heath. “The pagenumber of k-trees is $O(k)$ ”. In: *Discrete Applied Mathematics* Volume 109 (2001), pp. 215–221. ISSN: 0166-218X. DOI: [10.1016/S0166-218X\(00\)00178-5](https://doi.org/10.1016/S0166-218X(00)00178-5).
- [28] Lenwood S. Heath, Frank Thomson Leighton, and Arnold L. Rosenberg. “Comparing Queues and Stacks As Machines for Laying Out Graphs”. In: *SIAM Journal on Discrete Mathematics* Volume 5 (1992), pp. 398–412. DOI: [10.1137/0405031](https://doi.org/10.1137/0405031).
- [29] Lenwood S. Heath and Arnold L. Rosenberg. “Laying Out Graphs Using Queues”. In: *SIAM Journal on Computing* Volume 21 (1992), pp. 927–958. DOI: [10.1137/0221055](https://doi.org/10.1137/0221055).
- [30] Julia Katheder, Michael Kaufmann, Sergey Pupyrev, and Torsten Ueckerdt. “Transforming Stacks into Queues: Mixed and Separated Layouts of Graphs”. In: *42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025)*. Edited by Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng. Vol. 327. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025, 56:1–56:18. ISBN: 978-3-95977-365-2. DOI: [10.4230/LIPIcs.STACS.2025.56](https://doi.org/10.4230/LIPIcs.STACS.2025.56).
- [31] Sriram Venkata Pemmaraju. “Exploring the powers of stacks and queues via graph layouts”. Virginia Polytechnic Institute and State University, 1992. DOI: [10919/38418](https://doi.org/10.1091/38418).
- [32] Sergey Pupyrev. *A collection of existing results on stack and queue numbers*. 2020. URL: <https://spupyrev.github.io/linearlayouts.html>.
- [33] Neil Robertson and P.D Seymour. “Graph minors. II. Algorithmic aspects of tree-width”. In: *Journal of Algorithms* Volume 7 (1986), pp. 309–322. ISSN: 0196-6774. DOI: [10.1016/0196-6774\(86\)90023-4](https://doi.org/10.1016/0196-6774(86)90023-4).
- [34] Donald J Rose. “Triangulated graphs and the elimination process”. In: *Journal of Mathematical Analysis and Applications* Volume 32 (1970), pp. 597–609. ISSN: 0022-247X. DOI: [10.1016/0022-247X\(70\)90282-9](https://doi.org/10.1016/0022-247X(70)90282-9).
- [35] Donald J. Rose, R. Endre Tarjan, and George S. Lueker. “Algorithmic Aspects of Vertex Elimination on Graphs”. In: *SIAM Journal on Computing* Volume 5 (1976), pp. 266–283. DOI: [10.1137/0205021](https://doi.org/10.1137/0205021).
- [36] K. Wagner. “Über eine Eigenschaft der ebenen Komplexe”. In: *Mathematische Annalen* Volume 114 (Dec. 1937), pp. 570–590. ISSN: 1432-1807. DOI: [10.1007/BF01594196](https://doi.org/10.1007/BF01594196).
- [37] Veit Wiechert. “On the Queue-Number of Graphs with Bounded Tree-Width”. In: *The Electronic Journal of Combinatorics* Volume 24 (Mar. 2017), #P1.65. DOI: [10.37236/6429](https://doi.org/10.37236/6429).
- [38] Mihalis Yannakakis. “Embedding planar graphs in four pages”. In: *Journal of Computer and System Sciences* Volume 38 (1989), pp. 36–67. ISSN: 0022-0000. DOI: [10.1016/0022-0000\(89\)90032-9](https://doi.org/10.1016/0022-0000(89)90032-9).
- [39] Mihalis Yannakakis. “Planar graphs that need four pages”. In: *Journal of Combinatorial Theory, Series B* Volume 145 (2020), pp. 241–263. ISSN: 0095-8956. DOI: [10.1016/j.jctb.2020.05.008](https://doi.org/10.1016/j.jctb.2020.05.008).