Submitted to the SIAM JOURNAL ON DISCRETE MATHEMATICS, 1994.

Queue Layouts and Staircase Covers of Matrices

Lenwood S. Heath* Sriram V. Pemmaraju†

June 15, 1994

Abstract

A connection between a queue layout of an undirected graph and a staircase cover of its adjacency matrix is established. The connection is exploited to establish a number of combinatorial results relating the number of vertices, the number of edges, and the queuenumber of a queue layout. The staircase notion is generalized to that of an (h, w)-staircase, and an efficient algorithm to optimally cover a matrix with (h, w)-staircases is presented. The algorithm is applied to problems of monotonic subsequences and to the maxdominance problem of Atallah and Kosaraju.

1 Introduction

A queue layout of a graph linearizes the vertices of the graph and assigns its edges to some number of queues. Queue layouts are first studied in Heath and Rosenberg [11] and

*Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0106,heath@cs.vt.edu

†Department of Computer Science, University of Iowa, Iowa City, IA 52242

Heath, Leighton, and Rosenberg [9]. Queue layouts of graphs have applications in VLSI design [18], fault-tolerant parallel processing [18], scheduling parallel processors [3], sorting networks [19], and matrix computations on data driven networks [10].

In numerical computations, the bandwidth of a matrix is the typical measure on the structure of the nonzero entries in the matrix (see Cuthill and Mckee [4, 5]). One may think of bandwidth as the minimum number of contiguous diagonals required to cover the nonzero entries. Leiserson [14] utilizes the bandwidth of a matrix as the basis for a scheme to perform matrix-vector multiplication on a systolic array. Melhem [15] generalizes Leiserson's scheme by covering the nonzero entries in a matrix with stripes; in general, the number of stripes required may be much less than the bandwidth of the matrix. Heath, Pemmaraju, and Ribbens [10] further generalize this by utilizing staircase covers as the basis of a scheme to perform matrix-vector computations on a data driven network; in general, the number of staircases required may be much less than the number of stripes required. The resulting advantage is that the scheme proposed by Heath, Pemmaraju, and Ribbens has a smaller hardware requirement than the schemes proposed by Leiserson and Melhem, yet requires no additional time. See Section 5 for more details on this application.

In this paper, we establish a close connection between queue layouts of a graph and staircase covers of the adjacency matrix of the graph. We exploit this connection to prove combinatorial results related to queue layouts. The connection also leads to an algorithm that solves a class of matrix covering problems efficiently. We apply this algorithm to simplify algorithms for some previously studied problems.

More specifically, the paper is organized as follows. In Section 2, we provide definitions and basic properties of queue layouts and staircase covers of matrices. Section 3 establishes the connection between the queue layouts of a graph and the staircase covers of its adjacency matrix. Section 4 exploits this connection to obtain combinatorial results about queue layouts. In Section 5, we generalize staircases to (h, w)-staircases and develop an efficient algorithm to construct a minimum-size (h, w)-staircase cover of a matrix. In Section 6, we reduce two well-known problems—*largest monotonic subsequence* [6, 16, 2] and *maxdominance* [1]—to (h, w)-staircase covering problems. As a result, we obtain algorithms for these problems that are simpler, and often more efficient, than previous algorithms. Section 7 concludes with some direction for future research.

2 Definitions and Basic Properties

In this section, we define both queue layouts of graphs and staircase covers of matrices. Two partial orders are defined on the positions in a matrix; these turn out to be central concepts. We also review some basic properties of queue layouts that are required later in the paper.

2.1 Queue Layouts

Let $G = (V, E)$ be a simple graph with n vertices. A *total order* σ on V is a one-to-one, onto mapping from the set $\{1, 2, \ldots, n\}$ to V. We use the notation $u <_\sigma v$ to denote $\sigma^{-1}(u) < \sigma^{-1}(v)$.

Heath and Rosenberg [11] define queue layouts and related concepts in an operational style. Here we define the same concepts in a more combinatorial fashion. A queue layout of G starts with a total order σ on V. Two edges $(s,t) \in E$, $s <_\sigma t$, and $(u,v) \in E$, $u <_\sigma v$ are said to *nest* (according to σ) if $s <_\sigma u <_\sigma v <_\sigma t$ or $u <_\sigma s <_\sigma t <_\sigma v$. A *$k$-queue layout* of G (according to σ) colors E with k colors so that no two edges of the same color nest. Each color is called a *queue.* (The edges of any one color can be processed by a single queue, hence the queue layout terminology; see [11].) A coloring that minimizes k defines the *queuenumber* $\mathrm{QN}(G,\sigma) = k$ of G according to σ. The *queuenumber* $\mathrm{QN}(G)$ of G is the minimum $\mathrm{QN}(G,\sigma)$ taken over all total orders σ.

Suppose that σ is a total order on V. The *adjacency matrix* $\mathrm{ADJ}(G,\sigma) = (g_{i,j})_{n\times n}$ of G according to σ is

$$g_{i,j} = \begin{cases} 1 & \text{if } (\sigma(i), \sigma(j)) \in E; \\ 0 & \text{otherwise.} \end{cases}$$

Since G is undirected and simple, $\mathrm{ADJ}(G,\sigma)$ is a symmetric binary matrix with zeros on its main diagonal.

Figure 1 shows a graph with 7 vertices. A 2-queue layout of this graph according to the total order $\sigma = (1,2,3,4,5,6,7)$ is shown in Figure 2. The edges of the graph that are drawn as arches above the line of vertices are assigned to one queue, while the edges drawn as arches below the line of vertices are assigned to a second queue. The reader is invited to verify that this graph does not have a 1-queue layout and therefore its queuenumber is 2.

We review some basic results related to queue layouts from Heath and Rosenberg [11].

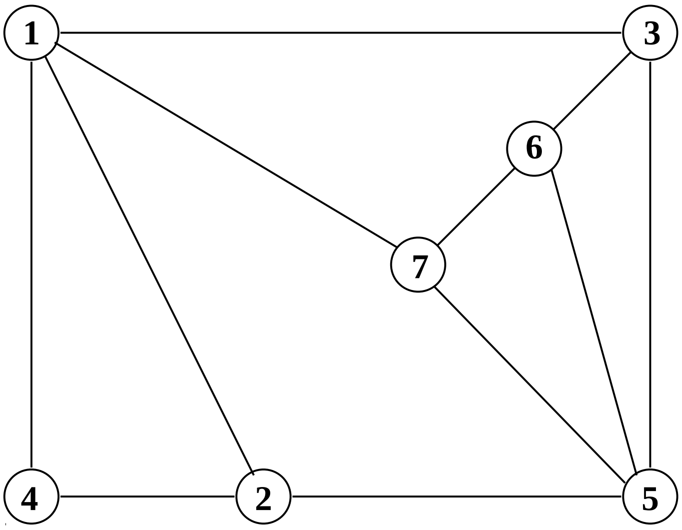

Figure 1: A graph with 7 vertices

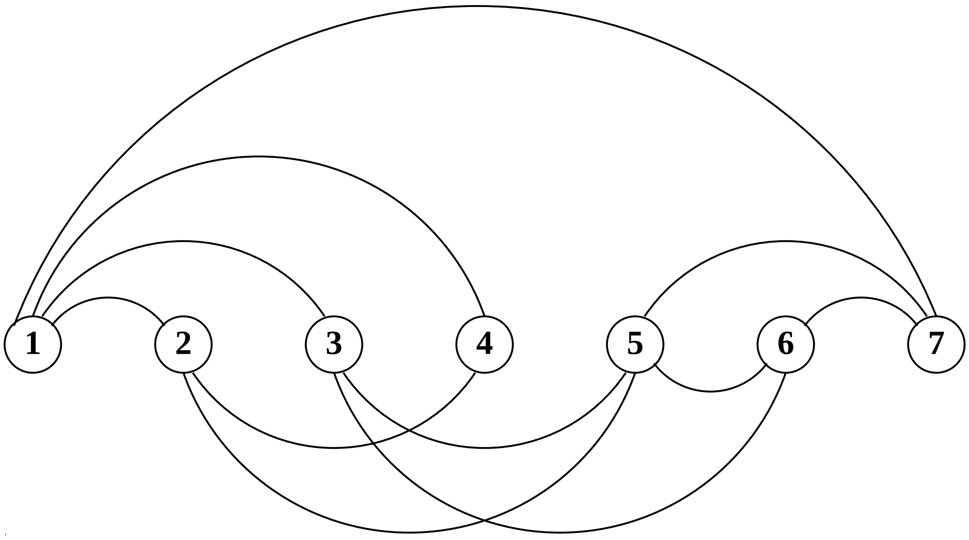

Figure 2: A 2-queue layout of the graph in Figure 1

Given a total order σ on V, a *rainbow* is a set of edges $\left\{(a_i, b_i) \mid 1 \le i \le k\right\}$ such that

$$a_1 <_\sigma a_2 <_\sigma \cdots <_\sigma a_{k-1} <_\sigma a_k <_\sigma b_k <_\sigma b_{k-1} <_\sigma \ldots <_\sigma b_2 <_\sigma b_1.$$

In other words, a rainbow is a matching in G such that every pair of edges nests. From the definition of queuenumber according to σ, the size of the largest rainbow in a layout is a lower bound on the number of queues required for that layout. In fact, Heath and Rosenberg [11] prove the following stronger result in which they show that the size of the largest rainbow is in fact equal to the queue number of the layout. Moreover, they also provide an efficient algorithm to compute the queuenumber of a fixed layout of a graph.

Proposition 1 (Heath and Rosenberg [11]) *Suppose that $G = (V, E)$ is a graph and σ is a total order on V. If the largest rainbow in σ is of size k, then a k-queue layout of G according to σ can be found in time $O(|E| \log\log |V|)$.*

2.2 Staircase Covers

Throughout the paper, we use $M = (m_{i,j})_{n\times n}$ to denote an $n \times n$ matrix and $J_n = \{1, 2, \ldots, n\}$ to denote the set of row and column indices of M. Each ordered pair $(i, j) \in J_n^2$ is called a *position index* of M. The position index (i, j) is said to *cover* the corresponding entry $m_{i,j}$ of M. Let $\Gamma \subseteq J_n^2$ be the set of position indices that cover nonzero entries of M; that is, $\Gamma = \{(i, j) \mid m_{i,j} \ne 0\}$. Define the *southeast poset* $(J_n^2, \le_{se})$ by $(i_1, j_1) \le_{se} (i_2, j_2)$ if and only if $i_1 \le i_2$ and $j_1 \le j_2$. A *staircase* in M is a chain Ψ in the poset $(\Gamma, \le_{se})$. A *staircase cover* of M is a set of staircases that collectively cover Γ. The *staircase number* of M, denoted $\mathrm{STAIR}(M)$, is the size of the smallest staircase cover of M.

Just as Heath and Rosenberg [11] identify a rainbow as an obstacle to minimizing the number of queues for a queue layout, we identify an obstacle for minimizing the number of staircases required to cover the nonzero entries of a matrix M. Define the *strictly northeast poset* $(J_n^2, \leq_{ne})$ by $(i_1, j_1) <_{ne} (i_2, j_2)$ if and only if $i_1 > i_2$ and $j_1 < j_2$. (*Strict* here means simply that no two comparable elements appear in the same row or the same column.) An *antistaircase* in M is a chain in the poset $(\Gamma, \leq_{ne})$. Recall that if two position indices (i_1, j_1) and (i_2, j_2) belong to a staircase in M, then either $i_1 \leq i_2$ and $j_1 \leq j_2$ or $i_2 \leq i_1$ and $j_2 \leq j_1$. This implies that two position indices belong to an antistaircase in M if and only if they are not in a common staircase of M. Thus, a set of position indices is an antistaircase in M if and only if it is an antichain in $(\Gamma, \leq_{se})$. The following relationship between the size of the minimum staircase cover of M and the size of the largest antistaircase in M relies on Dilworth's Theorem.

Theorem 2 *The size of a largest antistaircase in M equals* $\mathrm{STAIR}(M)$.

Proof. $\mathrm{STAIR}(M)$ is the minimum number of chains in $(J_n^2, \leq_{se})$ required to cover $\Gamma \subseteq J_n^2$. Hence, $\mathrm{STAIR}(M)$ is the minimum number of chains in the poset $(\Gamma, \leq_{se})$ required to cover Γ. Dilworth's Theorem [7] states that the minimum number of chains required to cover all the elements of a poset equals the size of a largest antichain. In the case of $(\Gamma, \leq_{se})$, this largest antichain is just a largest antistaircase in M, as desired. □

3 The Connection

In this section, we establish the connection between queue layouts of a graph and the staircase covers of its adjacency matrix (Theorem 5). The basic idea behind this connection is that a pair of edges of the graph that do no nest in a layout correspond to a pair of nonzeros in the adjacency matrix of the graph that can be covered by a staircase. The complication in presenting the connection arises from the fact that every edge in an adjacency matrix is represented twice.

We first present a result that allows us to focus on the upper triangle of the adjacency matrix of G. Let $\mathrm{UPPER}(M) = (u_{i,j})_{n\times n}$ such that $u_{i,j} = m_{i,j}$ if $i < j$; otherwise $u_{i,j} = 0$. Therefore $\mathrm{UPPER}(M)$ represents the portion of M strictly above the diagonal. Similarly, define $\mathrm{LOWER}(M) = (\ell_{i,j})_{n\times n}$ such that $\ell_{i,j} = m_{i,j}$ if $j < i$; otherwise $\ell_{i,j} = 0$. The *reflection* of a set of position indices $T \subseteq J_n^2$ in M is

$$\mathrm{REFLECT}(T) = \{(i,j) \mid (j,i) \in T\}.$$

The following proposition makes precise the observation that staircases and antistaircases in an adjacency matrix remain invariant under reflection.

Proposition 3 *These properties hold for an arbitrary matrix M.*

1. *If Ψ is a staircase in M, then* $\mathrm{REFLECT}(\Psi)$ *is also a staircase in M.*
2. *If Γ is symmetric (that is, closed under reflection), and A is an antistaircase in M, then* $\mathrm{REFLECT}(A)$ *is also an antistaircase in M.*

We have the following result for adjacency matrices.

Lemma 4 *Let M be* $\mathrm{ADJ}(G, \sigma)$ *for some graph $G = (V, E)$ and total order σ on V. Then*

$$\begin{aligned} 2 \cdot \mathrm{STAIR}(\mathrm{LOWER}(M)) &= 2 \cdot \mathrm{STAIR}(\mathrm{UPPER}(M)) \\ &= \mathrm{STAIR}(M). \end{aligned}$$

Proof. Because Γ is symmetric, the set

$$\Pi = \{\Psi_1, \Psi_2, \ldots, \Psi_k\}$$

is a staircase cover of $\mathrm{UPPER}(M)$ if and only if

$$\mathrm{REFLECT}(\Pi) = \{\mathrm{REFLECT}(\Psi_1), \mathrm{REFLECT}(\Psi_2), \ldots, \mathrm{REFLECT}(\Psi_k)\}$$

is a staircase cover of $\mathrm{LOWER}(M)$. Therefore,

$$\mathrm{STAIR}(\mathrm{UPPER}(M)) = \mathrm{STAIR}(\mathrm{LOWER}(M)).$$

Furthermore, if Π is a staircase cover of $\mathrm{UPPER}(M)$, then $\Pi \cup \mathrm{REFLECT}(\Pi)$ is a staircase cover of M (since no diagonal entries are in Γ).

Let $A \subseteq \Gamma$ be a largest antistaircase in M. Decompose A into $A_1 \cup A_2$, where A_1 consists of position indices in the upper triangle and A_2 consists of position indices in the lower triangle. Suppose that $|A_1| > |A_2|$. Then $A_1 \cup \mathrm{REFLECT}(A_1)$ is a larger antistaircase in M than A – a contradiction. A similar contradiction is obtained if we assume that $|A_1| < |A_2|$. Hence, $|A_1| = |A_2|$. If we suppose that A_1' is an antistaircase in $\mathrm{UPPER}(M)$ larger than A_1, then we obtain an antistaircase $A_1' \cup \mathrm{REFLECT}(A_1')$ in M that is larger than A – a contradiction. Hence A_1 is a largest antistaircase in UPPER(M) and similarly A_2 is a largest antistaircase in LOWER(M). By Theorem 2,

$$\begin{aligned}|A| &= \mathrm{STAIR}(M)\\ |A_1| &= \mathrm{STAIR}(\mathrm{UPPER}(M))\\ |A_2| &= \mathrm{STAIR}(\mathrm{LOWER}(M)).\end{aligned}$$

Hence

$$\begin{aligned}\mathrm{STAIR}(M) &= \mathrm{STAIR}(\mathrm{UPPER}(M)) + \mathrm{STAIR}(\mathrm{LOWER}(M))\\ &= 2\cdot\mathrm{STAIR}(\mathrm{UPPER}(M))\\ &= 2\cdot\mathrm{STAIR}(\mathrm{LOWER}(M)).\end{aligned}$$

The theorem follows. □

Figure 3 illustrates the pattern of nonzero entries in the adjacency matrix of the graph and layout of Figure 2. The nonzero entries are indicated by $\times$. The staircases that cover UPPER(M) and LOWER(M) are reflections of each other. The staircase number of M is 4, while the staircase number of both UPPER(M) and LOWER(M) is 2.

The connection between matrix covers and queue layouts is embodied in the next theorem.

Theorem 5 *Let $G = (V, E)$ be a graph and let σ be a total order on V. Then*

$$2\cdot\mathrm{QN}(G,\sigma) = \mathrm{STAIR}\Big(\mathrm{ADJ}(G,\sigma)\Big).$$

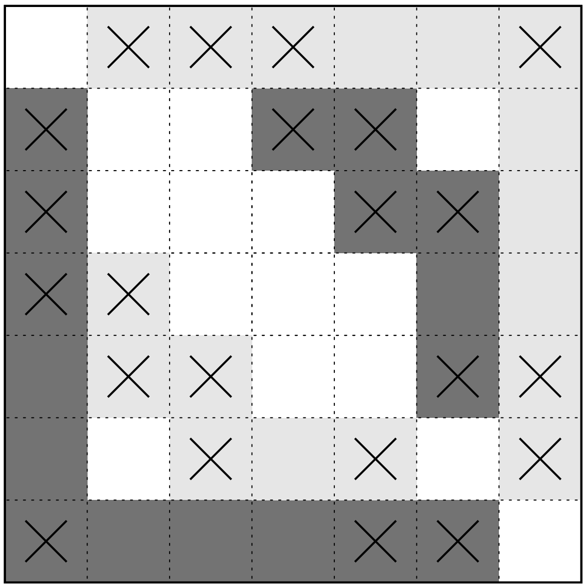

Figure 3: A 4-staircase covering of $\mathrm{ADJ}(G, \sigma)$.

Proof. Let $M = \mathrm{ADJ}(G, \sigma)$. Consider any two position indices $(i_1, j_1), (i_2, j_2) \in \Gamma$ that are in the upper triangle of M. It is easy to see that (i_1, j_1) and (i_2, j_2) are comparable with respect to $\leq_{ne}$ if and only if the corresponding edges $(\sigma(i_1), \sigma(j_1))$ and $(\sigma(i_2), \sigma(j_2))$ nest. Hence the size of a largest antistaircase in $\mathrm{UPPER}(\mathrm{ADJ}(G, \sigma))$ is exactly the size of a largest rainbow according to σ. Combining Proposition 1 and Lemma 4, the theorem follows. □

The following corollary is useful subsequently.

Corollary 6 *For any graph $G = (V, E)$ and total order σ on V, we have the following*

$$\mathrm{QN}(G, \sigma) = \mathrm{STAIR}(\mathrm{UPPER}(\mathrm{ADJ}(G, \sigma))) = \mathrm{STAIR}(\mathrm{LOWER}(\mathrm{ADJ}(G, \sigma))).$$

Figure 3 shows the adjacency matrix $\mathrm{ADJ}(G, \sigma)$ of the graph $G = (V, E)$ shown in

Figure 1 according to the total order $\sigma = (1, 2, 3, 4, 5, 6, 7)$ on V. Notice the correspondence between the assignment of edges to queues in the 2-queue layout of of G (according to $\sigma = (1, 2, 3, 4, 5, 6, 7)$) and the assignment of nonzero entries to staircases in the 4-staircase covering of $\mathrm{ADJ}(G, \sigma)$ shown in Figure 3.

4 Combinatorial Results

The formulation of a queue layout as a staircase covering of an adjacency matrix leads to new combinatorial results and simplifies the proofs of existing results related to queue layouts. For this section, we fix the graph $G = (V, E)$, set $n = |V|$, and set $m = |E|$. We exploit the connection between queue layouts and staircase covers to derive:

- A tight upper bound on the number of edges in a 1-queue graph as a function of n (Theorem 8).
- A tight upper bound on the number of edges in a k-queue graph as a function of k and n (Theorem 11).
- An existentially tight lower bound on the queuenumber of a graph as a function of m and n (Theorem 12).
- The exact number of maximal 1-queue layouts as a function of n (Theorem 14).

The proofs exploit the connection established between queue layouts and staircase covers.

Heath and Rosenberg [11] show that the maximum number of edges possible in a 1-queue graph is $2n-3$. Their argument is based on characterizing 1-queue graphs as arched leveled-planar graphs and then counting the number of edges in a maximal arched leveled-planar graph. As shown below, a rather different and substantially simpler proof based on the connection between queue layouts and staircase covers establishes the same upper bound.

Define the *source* of a nonempty staircase Ψ in M as the smallest position index, according to $\leq_{se}$, in Ψ. Similarly, define the *destination* of Ψ as the largest position index, according to $\leq_{se}$, in Ψ. The following result establishes an upper bound on the number of elements in a staircase as a function of its source and destination.

Lemma 7 *Let Ψ be a nonempty staircase in M with source (i_1, j_1) and destination (i_2, j_2). Then*

$$|\Psi| \leq (i_2 - i_1) + (j_2 - j_1) + 1.$$

Proof. This is seen by noting that the maximum number of elements in Ψ is one more than the length of a path from (i_1, j_1) to (i_2, j_2) consisting of a series of alternating horizontal and vertical line segments. The length of any such path is simply $(i_2 - i_1) + (j_2 - j_1) + 1$, the distance between (i_1, j_1) and (i_2, j_2) measured in the L_1 or Manhattan metric, as desired. $\square$

With the above proposition, we establish the following upper bound on the number of edges in a 1-queue graph.

Theorem 8 *If G is a 1-queue graph, then $m \leq 2n-3$.*

Proof. Let σ be an ordering of V that yields a 1-queue layout of G. By Theorem 2, $\mathrm{STAIR}(\mathrm{ADJ}(G,\sigma)) = 2$. Let $\Pi = \{\Psi, \mathrm{REFLECT}(\Psi)\}$ be a staircase cover of $\mathrm{ADJ}(G,\sigma)$ such that Ψ only contains position indices in the strict upper triangle of $\mathrm{ADJ}(G,\sigma)$. Therefore, $|\Psi|$ is an upper bound on the number of edges in G. $|\Psi|$ is maximized when it has source $(1,2)$ and destination $(n-1,n)$. By Lemma 7,

$$|\Psi| \leq (n-1-1) + (n-2) + 1 = 2n-3,$$

and hence $m \leq 2n-3$. □

A *maximal 1-queue graph* is a 1-queue graph with $m = 2n-3$ edges. A *maximal staircase* is a staircase Ψ in which for each position index $(i,j) \in \Psi$, with the exception of its destination, either $(i+1,j) \in \Psi$ or $(i,j+1) \in \Psi$. The following is a corollary to the proof of Theorem 8.

Corollary 9 *If G is a maximal 1-queue graph and σ is a 1-queue layout of G, then the matrix* $\mathrm{UPPER}(\mathrm{ADJ}(G,\sigma))$ *requires a maximal staircase with source* $(1,2)$ *and destination* $(n-1,n)$ *to cover all its nonzero entries.*

That the upper bound shown in Theorem 8 is tight can be seen by considering the graph $G = (V,E)$ with

$$\begin{aligned} V &= \{v_1, v_2, \ldots, v_n\} \\ E &= \{(v_1, v_i) \mid 2 \leq i \leq n\} \cup \{(v_n, v_i) \mid 2 \leq i \leq n-1\}. \end{aligned}$$

$S_0 := \Gamma$;
$\ell := 1$;
while $(S_{\ell-1} \neq \phi)$ **do**
    let Ψ_ℓ be the set of maxima in $(S_{\ell-1}, <_{ne})$;
    $S_\ell := S_{\ell-1} - \Psi_\ell$;
    $\ell := \ell + 1$;
    endwhile
return $\{\Psi_1, \Psi_2, \ldots, \Psi_{\ell-1}\}$

Figure 4: The **Staircase Cover Algorithm (SCA)**

G is the complete bipartite graph on $\{v_1, v_n\}$ and $\{v_2, \ldots, v_{n-1}\}$ with the additional edge (v_1, v_n). G has $2n-3$ edges and a 1-queue layout of G is obtained by laying out V in the order $v_1, v_2, \ldots, v_n$. The resulting staircase of size $2n-3$ first proceeds from $(1,2)$ right to $(1,n)$ and then proceeds from $(1,n)$ down to $(n-1,n)$.

Theorem 8 implies an upper bound of $k(2n-3)$ on the number of edges in a k-queue graph. Heath and Rosenberg [11] use this observation to derive a lower bound of $m/(2n-3)$ on the queuenumber of G. For $k \geq 2$, this upper bound is not tight. We show a tight bound in Theorem 11. In order to show this bound we use an algorithm called the **Staircase Cover Algorithm (SCA)** (see Figure 4) that takes as input the set of nonzero position indices Γ of a matrix M (not necessarily an adjacency matrix) and produces as output a set of nonempty staircases

$$\{\Psi_1, \Psi_2, \ldots, \Psi_k\}$$

that forms a minimal staircase cover of M. (In fact, $\{\Psi_1, \Psi_2, \ldots, \Psi_k\}$ is a partition of Γ.) A more general version of this algorithm is detailed in Section 5 where the issue of time complexity is addressed. This result is established in the following Lemma.

Lemma 10 *SCA takes as input the set of nonzero position indices of M and produces as output a set*

$$\Pi = \{\Psi_1, \Psi_2, \ldots, \Psi_k\},$$

which is a minimal staircase cover of M.

Proof. The proof consists of demonstrating that Ψ is a staircase cover of M and that Ψ is a minimal staircase cover.

Π is a staircase cover of M. No two position indices (i_1, j_1) and (i_2, j_2) belong to the same set Ψ_ℓ, $1 \leq \ell \leq k$ if $(i_1, j_1) <_{ne} (i_2, j_2)$ or $(i_2, j_2) <_{ne} (i_1, j_1)$. Therefore, for each ℓ, $1 \leq \ell \leq k$, Ψ_ℓ is an antichain in $(\Gamma, \leq_{ne})$ and a chain in the poset $(J_n^2, \leq_{se})$. Hence Ψ_ℓ is a staircase in M. When SCA terminates, every element in Γ has been placed in some set Ψ_ℓ, $1 \leq \ell \leq k$. Hence, $\{\Psi_1, \Psi_2, \ldots, \Psi_k\}$ is a staircase cover of M.

Π is a minimal staircase cover. If $k \leq 1$, then the result is clear. Hence suppose that $k \geq 2$. For an arbitrary element $(i_k, j_k) \in \Psi_k$, there exists at least one element $(i_{k-1}, j_{k-1}) \in \Psi_{k-1}$ such that $(i_k, j_k) <_{ne} (i_{k-1}, j_{k-1})$. Otherwise, (i_k, j_k) would have been included in Ψ_{k-1}. Extending this argument inductively, we obtain a sequence of position indices

$$(i_k, j_k) <_{ne} (i_{k-1}, j_{k-1}) <_{ne} \ldots <_{ne} (i_1, j_1)$$

such that $(i_\ell, j_\ell) \in \Psi_\ell \subseteq \Gamma$, for all ℓ. Hence $\{(i_k, j_k), (i_{k-1}, j_{k-1}), \ldots, (i_1, j_1)\}$ is an antistaircase of size k in M. By Theorem 2, $\{\Psi_1, \Psi_2, \ldots, \Psi_k\}$ is a minimal staircase cover of M.

□

We now establish a tight upper bound on the number of edges in a k-queue graph.

Theorem 11 *If G is a k-queue graph with n vertices and m edges, then*

$$m \quad \leq \quad k(2n-3) - 2k(k-1).$$

For each k and for $n = 2k$, there exists a k-queue graph with n vertices and $k(2n-3) - 2k(k-1)$ edges.

Proof. The two statements are proved separately, as follows.

The Upper Bound. Let σ be an ordering of V that yields a k-queue layout of G. Let

$$M \quad = \quad \mathrm{UPPER}(\mathrm{ADJ}(G, \sigma)).$$

We know by Lemma 4 and Theorem 5 that $\mathrm{STAIR}(M) = k$.

Let $\Pi(M) = \{\Psi_1, \Psi_2, \ldots, \Psi_k\}$ be the staircase cover of M produced by the algorithm SCA. We claim that for each ℓ, $1 \leq \ell \leq k$,

$$|\Psi_\ell| \leq (2n-3) - 4(\ell - 1).$$

Let the source of Ψ_ℓ be (i_ℓ, j_ℓ) and the destination be (i'_ℓ, j'_ℓ). There exists an element $(i_{\ell-1}, j_{\ell-1}) \in \Psi_{\ell-1}$ such that $(i_\ell, j_\ell) <_{ne} (i_{\ell-1}, j_{\ell-1})$. Otherwise, (i_ℓ, j_ℓ) would have been included in $\Psi_{\ell-1}$. This can be inductively extended to show that there exists an antistaircase

$$(i_\ell, j_\ell) <_{ne} (i_{\ell-1}, j_{\ell-1}) <_{ne} \ldots <_{ne} (i_1, j_1) \tag{1}$$

such that $(i_p, j_p) \in \Psi_p$ for all p, $1 \leq p \leq \ell$. Similarly, there exists an antistaircase of size ℓ:

$$(i'_\ell, j'_\ell) <_{ne} (i'_{\ell-1}, j'_{\ell-1}) <_{ne} \ldots <_{ne} (i'_1, j'_1) \tag{2}$$

From equation 1 we obtain

$$i_\ell > i_{\ell-1} > \ldots i_2 > i_1 \geq 1$$

and hence $i_\ell \geq \ell$. From equation 2 we obtain

$$j'_\ell < j'_{\ell-1} < \ldots j'_2 < j'_1 \leq n$$

and hence $j'_\ell \leq n - \ell + 1$. These conditions along with the fact that (i_ℓ, j_ℓ) and (i'_ℓ, j'_ℓ) belong to the strict upper triangle of $\mathrm{ADJ}(G, \sigma)$ imply that

$$\ell \leq i_\ell < j_\ell \quad \text{and} \quad i'_\ell < j'_\ell \leq n - \ell + 1.$$

Recall that $|\Psi_\ell| \leq (i'_\ell - i_\ell) + (j'_\ell - j_\ell) + 1$. Thus, $|\Psi_\ell|$ is maximized when

$$(i_\ell, j_\ell) = (\ell, \ell + 1) \quad \text{and} \quad (i'_\ell, j'_\ell) = (n - \ell, n - \ell + 1).$$

Therefore,

$$\begin{aligned} |\Psi_\ell| &\leq [(n - \ell) - \ell] + [(n - \ell + 1) - (\ell + 1)] + 1 \\ &= (2n - 3) - 4(\ell - 1). \end{aligned}$$

Since each edge in G corresponds to a distinct element in Ψ_ℓ, $1 \leq \ell \leq k$, the number of edges in a k-queue graph can be at most

$$\begin{aligned} \sum_{\ell=1}^{k} |\Psi_\ell| &\leq \sum_{\ell=1}^{k} (2n - 3) - 4(\ell - 1) \\ &= k(2n - 3) - 2k(k - 1), \end{aligned}$$

as claimed.

The Upper Bound is Existentially Tight. For each $k \geq 1$, choose $n = 2k$ and consider the complete graph on n vertices K_n. Heath and Rosenberg [11] show that $\mathrm{QN}(K_n) = \lfloor \frac{n}{2} \rfloor = k$ and hence K_n is a k-queue graph. The upper bound of $k(2n-3) - 2k(k-1)$ dictates that K_n have at most

$$\begin{aligned} & \frac{n}{2}(2n-3) - n\left(\frac{n}{2} - 1\right) \\ = \; & \frac{n}{2}(n-1) \end{aligned}$$

edges. This is precisely the number of edges in K_n. □

From the upper bound in Theorem 11, we derive bounds on the queuenumber of G as a function of m and n.

Theorem 12 *For any G with n vertices and m edges, the following inequalities hold:*

$$\frac{1}{4}\left((2n-1) - \sqrt{(2n-1)^2 - 8m}\right) \leq \mathrm{QN}(G) \leq \frac{1}{4}\left((2n-1) + \sqrt{(2n-1)^2 - 8m}\right).$$

Proof. Let $k = \mathrm{QN}(G)$. From Theorem 11, we have that $m \leq k(2n-3) - 2k(k-1)$. Simplification leads to

$$2k^2 + (1-2n)k + m \leq 0.$$

Solving this quadratic inequality for k leads to the claimed bounds. □

When G is a complete graph, we have $m = n(n-1)/2$, and the bounds proved in Theorem 12 simplify to

$$\frac{n-1}{2} \leq \mathrm{QN}(G) \leq \frac{n}{2}.$$

This leads to the following corollary.

Corollary 13 *The queuenumber of the complete graph K_n is*

$$\mathrm{QN}(K_n) = \left\lfloor \frac{n}{2} \right\rfloor .$$

Heath and Rosenberg [11] show that $\mathrm{QN}(K_n) = \lfloor n/2 \rfloor$ by establishing a lower bound of $\lfloor n/2 \rfloor$ on $\mathrm{QN}(K_n)$ and then explicitly constructing a queue layout of K_n with $\lfloor n/2 \rfloor$ queues. In contrast with their proof, the above corollary follows simply from the number of edges in K_n.

As a final application of the connection between queue layouts and staircase covers, we consider maximal 1-queue layouts (layouts of n-vertex, 1-queue graphs with $2n-3$ edges). The following theorem counts the number of maximal 1-queue layouts.

Theorem 14 *The number of maximal n-vertex, 1-queue layouts is the Catalan number*

$$C_{n-1} = \binom{2n-2}{n-1} \frac{1}{n}.$$

Proof. Let $G = (V, E)$ be an n-vertex, 1-queue graph with $2n-3$ edges, and let σ be a total order on V that yields a 1-queue layout of G. Counting the number of maximal 1-queue layouts is equivalent to counting the number of distinct maximal staircases with source $(1, 2)$ and destination $(n-1, n)$ that stay above the main diagonal.

Let Ψ be a maximal staircase in $\mathrm{UPPER}(\mathrm{ADJ}(G, \sigma))$ with source $(1, 2)$ and destination $(n-1, n)$ containing the elements

$$(i_1, j_1) = (1, 2) \leq_{se} (i_2, j_2) \leq_{se} \cdots \leq_{se} (i_{2n-3}, j_{2n-3}) = (n-1, n).$$

We transform this sequence into a string of balanced parentheses. As observed earlier for all ℓ, $1 \leq \ell \leq 2n-2$, either

1. $i_{\ell+1} = i_\ell + 1$ and $j_{\ell+1} = j_\ell$ or

2. $i_{\ell+1} = i_\ell$ and $j_{\ell+1} = j_\ell + 1$.

Scan the elements of Ψ in the ascending order of $\leq_{se}$. On moving from one position index to the next, if (1) holds then write down a left parenthesis and if (2) holds then write down a right parenthesis. It is easily verified that what has been written down is a string of $(2n-2)$ balanced parentheses that is unique for each maximal staircase with source $(1,2)$ and destination $(n-1,n)$. Similarly, every string of $(2n-2)$ balanced parentheses gives rise to a maximal staircase of the desired form. We know that the number of balanced parentheses expressions with $n-1$ left parentheses and $n-1$ right parentheses is the Catalan number $C_{n-1} = \binom{2n-2}{n-1}\frac{1}{n}$ (see Graham, Knuth, and Patashnik [8] Section 7.5). The theorem follows. $\square$

5 Enhancing the Connection

In this section, we generalize the connection between the queue layouts of a graph and the staircase covers of its adjacency matrix to construct an algorithm that solves a large class of matrix covering problems in $O(|\Gamma| \log\log n)$ time. For $h, w \geq 1$, an (h,w)-*staircase* is a staircase that contains at most h nonzero elements in any column and at most w nonzero elements in any row. An (h,w)-*staircase cover* of M is a set of (h,w)-staircases that covers

Γ. The size of a smallest (h, w)-staircase cover of M is the (h, w)-*staircase number* of M and is denoted by $\text{STAIR}_{h,w}(M)$. In subsequent subsections, we present: a more general staircase cover problem (Section 5.1), a more general definition of antistaircase (Section 5.2), and an algorithm that generalizes SCA and solves the more general staircase cover problem (Section 5.3).

5.1 A General Problem

From the notion of an (h, w)-staircase arises this more general staircase cover problem:

> (h, w)-**Staircase Cover Problem ((h, w)-SCCOVER)**
>
> Given a matrix $M = (m_{i,j})_{n \times n}$, find a smallest possible (h, w)-staircase cover Π of M.

In the context of an adjacency matrix, we consider only the upper triangle portion of the matrix. The more general problem, (h, w)-SCCOVER restricts queue layouts to having no more than h edges incident to a particular vertex in any single queue that go to the right and no more than w edges incident to a particular vertex in any single queue that go to the left.

Heath, Pemmaraju, and Ribbens [10] give one motivation for solving (h, w)-SCCOVER. They present a scheme for performing sparse matrix-vector multiplication on a linear array of processing elements (PEs). As shown in that paper, given positive integers h and w, an $n \times n$ matrix M and an n-vector x can be multiplied using $\text{STAIR}_{h,w}(M)$ PEs by feeding elements in each (h, w)-staircase to a distinct PE. As h and w increase, $\text{STAIR}_{h,w}(M)$

decreases, and therefore so does the hardware requirement. But, on the other hand as h and w increase the storage requirement for each PE increases. Therefore, given a particular set of constraints on the number of PEs available and the amount of storage available in each PE, there are certain choices of h and w that are optimal. To make this choice and to subsequently perform the matrix-vector multiplication, (h, w)-SCCOVER is the problem to solve. In the application described above, M is sparse, which typically means that $|\Gamma| = O(n)$. For such an application, the algorithm for (h,w)$-SCCOVER$ in Section 5.3 can construct a minimal (h, w)-staircase cover of M in $O(n \log\log n)$ time. Moreover, our algorithm handles the strip cover problem of Melhem [15] as the special case $h = 1$ and $w = 1$ and handles the normal staircase cover problem as the special case $h = n$ and $w = n$.

5.2 A More General Antistaircase

Now we define the notion of an (h, w)-antistaircase and show that $\text{STAIR}_{h,w}(M)$ is equal to the size of the largest (h, w)-antistaircase in M. For row i, $1 \leq i \leq n$, and two columns j_1 and j_2, $1 \leq j_1, j_2 \leq n$, let $R_i(j_1, j_2)$ be the set of position indices of nonzeros in row i between columns j_1 and j_2; that is,

$$R_i(j_1, j_2) \quad = \quad \Gamma \cap \{(i, j) \mid j_1 \leq j \leq j_2\}.$$

Similarly, for column j, $1 \leq j \leq n$, and two rows i_1 and i_2, $1 \leq i_1, i_2 \leq n$, let $C_j(i_1, i_2)$ be the set of position indices of nonzeros in column j between rows i_1 and i_2; that is,

$$C_j(i_1, i_2) \quad = \quad \Gamma \cap \{(i, j) \mid i_1 \leq i \leq i_2\}.$$

We extend the $\leq_{ne}$ partial order to a binary relation $\leq_{ne,h,w}$ on Γ as follows. We say that $(i_1, j_1) \leq_{ne,h,w} (i_2, j_2)$ if

1. $(i_1, j_1) \leq_{ne} (i_2, j_2)$; or
2. $i_1 = i_2$ and $|R_{i_1}(j_1, j_2)| > w$; or
3. $j_1 = j_2$ and $|C_{j_1}(i_2, i_1)| > h$.

The binary relation $<_{ne,h,w}$ is the relation $\leq_{ne,h,w}$ minus the equality relation. For the matrix shown in Figure 3,

$$(7, 1) \leq_{ne,2,2} (7, 6) \leq_{ne,2,2} (3, 6) \leq_{ne,2,2} (1, 7).$$

The next two lemmas provide the tools necessary to construct an algorithm to solve (h, w)-SCCOVER.

Lemma 15 $(\Gamma, \leq_{ne,h,w})$ *is a poset.*

Proof. We show that $\leq_{ne,h,w}$ is reflexive, antisymmetric, and transitive.

$\leq_{ne,h,w}$ **is Reflexive.** By alternative (1) in the definition, we always have $(i, j) \leq_{ne,h,w} (i, j)$; hence $\leq_{ne,h,w}$ is reflexive.

$\leq_{ne,h,w}$ **is Antisymmetric.** Assume that $(i_1, j_1) \leq_{ne,h,w} (i_2, j_2)$ and $(i_1, j_1) \neq (i_2, j_2)$. If $(i_1, j_1) \leq_{ne} (i_2, j_2)$, then $(i_2, j_2) \not\leq_{ne} (i_1, j_1)$ and $i_1 > i_2$ and $j_1 < j_2$. Therefore

$$(i_2, j_2) \not\leq_{ne,h,w} (i_1, j_1).$$

On the other hand, if $(i_1, j_1) \leq_{ne,h,w} (i_2, j_2)$ but $(i_1, j_1) \not\leq_{ne} (i_2, j_2)$, then either $i_1 = i_2$ and $|R_{i_1}(j_1, j_2)| > w$ or $j_1 = j_2$ and $|C_{j_1}(i_2, i_1)| > h$. In this case also $(i_2, j_2) \not\leq_{ne,h,w} (i_1, j_1)$. Therefore $\leq_{ne,h,w}$ is anti-symmetric.

$\leq_{ne,h,w}$ **is Transitive.** To show that $\leq_{ne,h,w}$ is transitive assume that

$$(i_1, j_1) \quad \leq_{ne,h,w} \quad (i_2, j_2)$$

and

$$(i_2, j_2) \quad \leq_{ne,h,w} \quad (i_3, j_3).$$

If either $(i_1, j_1) = (i_2, j_2)$ or $(i_2, j_2) = (i_3, j_3)$, then immediately $(i_1, j_1) \leq_{ne,h,w} (i_3, j_3)$.

Therefore, assume that $(i_1, j_1) <_{ne,h,w} (i_2, j_2)$ and $(i_2, j_2) <_{ne,h,w} (i_3, j_3)$. This implies that $i_3 \leq i_2 \leq i_1$ and $j_1 \leq j_2 \leq j_3$. If either $(i_1, j_1) <_{ne} (i_2, j_2)$ or $(i_2, j_2) <_{ne} (i_3, j_3)$, then $(i_1, j_1) <_{ne} (i_3, j_3)$ and $(i_1, j_1) <_{ne,h,w} (i_3, j_3)$.

So assume that $(i_1, j_1) \not<_{ne} (i_2, j_2)$ and $(i_2, j_2) \not<_{ne} (i_3, j_3)$. There are two possible relationships between (i_1, j_1) and (i_2, j_2), as considered in Case 1 and Case 2.

Case 1: $i_1 = i_2 = i$ **and** $|R_{i_1}(j_1, j_2)| > w$**.** Notice that $j_2 > j_1 + w$. There are two possible relationships between (i_2, j_2) and (i_3, j_3). If $i_2 = i_3$ and $|R_{i_2}(j_2, j_3)| > w$, then $i_1 = i_3$ and $|R_{i_1}(j_1, j_3)| > w$. Therefore $(i_1, j_1) <_{ne,h,w} (i_3, j_3)$. The other possibility is that $j_2 = j_3$ and $|C_{j_2}(i_2, i_3)| > h$. In this case, $i_1 = i_2 > i_3 + h$ and $j_3 = j_2 > j_1 + w$. Therefore $i_1 > i_3$ and $j_1 < j_3$ implying that $(i_1, j_1) <_{ne} (i_3, j_3)$ and hence $(i_1, j_1) <_{ne,h,w} (i_3, j_3)$.

Case 2: $j_1 = j_2 = j$ **and** $|C_{j_1}(i_2, i_1)| > h$**.** After switching the roles of rows and columns, the argument is the same as for Case 1.

This shows that $<_{ne,h,w}$ is transitive.

As $\leq_{ne,h,w}$ is reflexive, antisymmetric, and transitive, it is a partial order, and the theorem follows. $\square$

An (h, w)-*antistaircase* is a chain in the poset $(\Gamma, \leq_{ne,h,w})$. In the next lemma, we show that an (h, w)-antistaircase is indeed an obstacle to minimizing the (h, w)-staircase number of M. More precisely, we show that $\text{STAIR}(M, h, w)$ is bounded below by the size of the largest (h, w)-antistaircase in M.

In Section 2.2, we observe that no pair of position indices in an antistaircase can be elements in the same staircase (the case $h = w = n$). This observation immediately implies that the length of a largest antistaircase in M is a lower bound on $\text{STAIR}(M)$. The more general observation that no two position indices in an (h, w)-antistaircase in M can be elements in the same (h, w)-staircase would be useful. Unfortunately, as illustrated in the following example, this is not true. For $w < n$, consider the position indices (i, j_1) and $(i, j_1 + w)$, and assume that $\{(i, j) \mid j_1 \leq j \leq j_1 + w\} \subseteq \Gamma$. In other words, there is a block of nonzero entries in row i, from column j_1 to column $j_1 + w$. Therefore $(i, j_1) <_{ne,h,w} (i, j_1 + w)$. But, the position indices (i, j_1) and $(i, j_1 + w)$ could be elements in the same (h, w)-staircase, provided some of the position indices in the set $\{(i, j) \mid j_1 < j < j_1 + w\}$ belong to other (h, w)-staircases. This example does *not* deal a fatal blow to our plans. We are still able to show, in a weaker form, that an (h, w)-antistaircase is indeed an obstacle to minimizing the (h, w)-staircase number of M.

Lemma 16 is a generalization of Theorem 2.

Lemma 16 *If M contains an (h, w)-antistaircase of size k, then*

$$\mathrm{STAIR}(M, h, w) \geq k.$$

Proof. Let Q denote an (h, w)-antistaircase in M of size k, written as an increasing sequence with respect to $\leq_{ne,h,w}$. The sequence Q can be uniquely represented as some number of contiguous subsequences

$$Q_1, Q_2, \ldots, Q_r$$

such that

- The first element of Q_1 is the first element of Q;
- The last element of Q_r is the last element of Q;
- The first element of Q_ℓ, $2 \leq \ell \leq r$, is the last element of $Q_{\ell-1}$;
- Each subsequence Q_ℓ, $1 \leq \ell \leq r$, is of one of three types:

  Type 1: Q_ℓ is an antistaircase consisting of m elements:

$$(i_1, j_1) <_{ne} (i_2, j_2) <_{ne} \ldots <_{ne} (i_m, j_m).$$

  Type 2: All elements of Q_ℓ are in the same row i:

$$(i, j_1) <_{ne,h,w} (i, j_2) <_{ne,h,w} \ldots <_{ne,h,w} (i, j_m);$$

therefore,

$$|R_i(j_p, j_{p+1})| > w$$

for all p, $1 \leq p < m$.

Type 3: All elements of Q_ℓ are in the same column j:

$$(i_1, j) <_{ne,h,w} (i_2, j) <_{ne,h,w} \ldots <_{ne,h,w} (i_m, j);$$

therefore,

$$|C_j(i_{p+1}, i_p)| > h$$

for all p, $1 \leq p < m$.

- Two adjacent subsequences Q_ℓ and $Q_{\ell+1}$ are of different types.

Thus, $|Q| = 1 - r + \sum_{\ell=1}^{r} |Q_\ell|$. Let Π be an arbitrary (h, w)-staircase cover of M. If any $\Psi \in \Pi$ contains an element from two distinct subsequences, then the element is the last element in subsequence Q_ℓ and is the first element in subsequence $Q_{\ell+1}$ for some ℓ, $1 \leq \ell \leq r - 1$.

Let λ_ℓ be the number of (h, w)-staircases in Π required to cover subsequence Q_ℓ. From the above observations, a lower bound on the cardinality of Π is

$$|\Pi| \geq 1 - r + \sum_{\ell=1}^{r} \lambda_\ell.$$

We now show that $\lambda_\ell \geq |Q_\ell|$ for all ℓ, $1 \leq \ell \leq r$. If Q_ℓ is of type 1, then it is clear that $\lambda_\ell \geq |Q_\ell|$. By symmetry between types 2 and 3, it suffices to show that $\lambda_\ell = |Q_\ell|$ for a subsequence Q_ℓ of type 2. Suppose Q_ℓ consists of the following position indices in row i:

$$(i, j_1) <_{ne,h,w} (i, j_2) <_{ne,h,w} \cdots <_{ne,h,w} (i, j_m).$$

We know that $|R_i(j_p, j_{p+1})| > w$ for all p, $1 \leq p < m = |Q_\ell|$. Therefore,

$$|R_i(j_1, j_m)| > w(|Q_\ell| - 1).$$

Thus, at least $|Q_\ell|$ (h, w)-staircases are required to cover all the elements in Q_ℓ. Therefore

$$|\Pi| \geq 1 - r + \sum_{\ell=1}^{r} |Q_\ell| = |Q| = k.$$

The theorem follows. □

5.3 The General Algorithm

The inspiration for (h, w)-SCA is derived from the algorithm of Heath and Rosenberg [11] that constructs a minimal queue layout of a graph according to a given fixed order on the vertices (see Proposition 1). Given the connection between queue layouts and staircase covers established in Section 3, the algorithm of Heath and Rosenberg generalizes into an algorithm to compute a minimal staircase cover of a matrix (SCA). We generalize this further to construct an algorithm to solve (h, w)-SCCOVER.

The algorithm (h, w)-SCA processes M row by row starting at row 1 and proceeding to row n. Within each row, (h, w)-SCA processes the entries one by one starting at the entry in column n and proceeding to the entry in column 1. For purposes of time complexity, we assume that the next element of Γ with respect to this order of processing can be accessed in constant time.

The algorithm builds an (h, w)-staircase cover $\Pi = \{\Psi_1, \Psi_2, \ldots, \Psi_k\}$. Let $M(i, j)$ be the submatrix of M that is to the northeast of (i, j) (inclusive of row i and column j). When a nonzero entry $m_{i,j}$ is encountered, (h, w)-SCA inserts (i, j) into Ψ_ℓ if the size of a largest (h, w)-antistaircase in $M(i, j)$ is ℓ. This implies that Π has size k only if the size of the largest (h, w)-antistaircase in M is k. By Lemma 16, we know that Π is a minimum size (h, w)-staircase cover of M.

Finding ℓ, the size of a largest (h, w)-antistaircase in $M(i, j)$, is possible because (h, w)-SCA has processed the rest of submatrix $M(i, j)$ before processing element (i, j). To find ℓ, (h, w)-SCA does the work to find ℓ in two separate phases. Define

$$\Gamma' = \Gamma \cap \{(p, q) \mid p < i \text{ and } q \geq j\}.$$

Γ' is the set of non-zero entries to the north-east of (i, j) including those in column j, but excluding those in row i. In phase 1, (h, w)-SCA determines ℓ_1, the size of a largest (h, w)-antistaircase whose smallest element is (i, j) (with respect to $<_{ne,h,w}$) and whose second smallest element is in Γ'. In phase 2, (h, w)-SCA determines ℓ_2, the size of a largest (h, w)-antistaircase whose smallest element is (i, j) (with respect to $<_{ne,h,w}$) and whose second smallest element is in $\Gamma - \Gamma'$. Clearly, the size of a largest (h, w)-antistaircase in $M(i, j)$ is given by $\ell = \max\{\ell_1, \ell_2\}$.

Figure 5 shows (h, w)-SCA. Observe that the largest (h, w)-staircase cover that needs to be accommodated has size $2n - 1$. In phase 1, ℓ_1 is computed using two integer arrays: $R[0..2n]$ and $Y[1..2n-1]$. At any stage in the algorithm, $R[t]$ is the largest column index of

```
(* Initialization *)
R[0] := n + 1; R[1..2n − 1] := 0; Y[1..2n − 1] := 0;
for i := 1 to n do                    (* Process row i *)
    c := 0; s := 0;
    for j := n downto 1 do            (* Process column j in row i *)
        if (m_{i,j} ≠ 0) then
            Find t such that R[t] ≥ j > R[t + 1];                     (*)
(* Finding ℓ_1 *)
            if (R[t] > j) then
                ℓ_1 := t + 1
            else if(R[t] = j) and (Y[ℓ] = h) then
                ℓ_1 := t + 1
            else (R[t] = j) and (Y[ℓ] < h) then
                ℓ_1 := t;
(* Finding ℓ_2 *)
            if (c = 0) then
                ℓ_2 := 1
            else if (c > 0) and (s < w) then
                ℓ_2 := c
            else if (c > 0) and (s = w) then
                ℓ_2 := c + 1;
(* Finding ℓ and inserting (i, j) into Ψ_ℓ *)
            ℓ := max(ℓ_1, ℓ_2); Ψ_ℓ := Ψ_ℓ ∪ {(i, j)};
(* Updating Y *)
            if (R[t] > j) then
                Y[ℓ] := 1;
            else if (R[t] = j) and (ℓ = t) then
                Y[ℓ] := Y[ℓ] + 1;
(* Updating c and s *)
            if (ℓ = c) then
                s := s + 1
            else if (ℓ > c) then
                s := 1; c := ℓ;
            endif (* m_{i,j} ≠ 0 *)
    endfor (* j loop *)
    Update array R to accommodate elements in row i;
endfor (* i loop *)
```

Figure 5: The (h, w)-**Staircase Cover Algorithm** (**(h, w)-SCA**)

any element in Ψ_t, excluding the elements in the row i (the row currently being processed). ($R[0]$ and $R[2n]$ represent dummy (h, w)-staircases included for simplicity of exposition.) R satisfies the following invariant:

- $R[t] \geq R[t+1]$ for all t, $0 \leq t \leq 2n - 1$.
- There exists a t, $1 \leq t \leq 2n$ such that $R[t] = 0$ and $R[t-1] > R[t]$.

At any stage in the algorithm, $Y[t]$ contains the number of elements most recently inserted into Ψ_t that belong to the same column. Like R, the values in Y also ignore the elements in the current row. In phase 2, we only need two variables s and c to compute ℓ_2. At any stage in the algorithm, $c = t$ if the most recently processed position index in the current row was inserted into Ψ_t. The variable s contains the number of most recently processed position indices in the current row that have been inserted into the same (h, w)-staircase. For more details on this algorithm see [17].

The final comment that we make about the (h, w)-SCA relates to its time complexity. It is easy to see that each step inside the two loops can be executed in constant time, except for the step marked (*), which can be accomplished in $O(\log n)$ time because R is a sorted array of size $O(n)$. With the priority queue of Johnson [12], step (*) can be performed in $O(\log \log n)$ time. Johnson's priority queue allows for initialization, insertion, and deletion in $O(\log \log n)$ time if the input is from a "small" restricted domain of natural numbers (in this case $[0..2n]$). Clearly, the time complexity of (h, w)-SCA depends on the representation of the input. Given our assumption of constant time access to elements of Γ, we have the following result:

Theorem 17 *Given a list of non-zero entries Γ of M sorted in ascending order by row and in descending order by column within rows, (h, w)-SCA constructs a minimal (h, w)-staircase cover of M in time $O(|\Gamma| \log \log n)$.*

Karlsson and Overmars [13] use Johnson's priority queue in a similar manner to obtain an $O(n \log \log n)$ algorithm for the maximal layers problem (see Theorem 3.1 in [13]). The solution to (h, w)-SCCOVER generalizes their result by introducing constraints (h and w) on the structure of maximal layers.

6 Applications of the Algorithm

In this section, we show how efficient computation of a minimum (h, w)-staircase cover of a matrix leads to efficient solutions to several well-known problems. We begin by showing in the following theorem that (h, w)-SCA can be used to provide efficient algorithms for problems related to sequences.

Theorem 18 *Given a sequence σ of length n consisting of numbers drawn from the restricted domain $[1..n]$, the following problems can be solved in $O(n \log \log n)$ time.*

1. *Find a smallest partition of σ into strictly increasing (decreasing) subsequences.*

2. *Find a smallest partition of σ into increasing (decreasing) subsequences.*

3. *Find a largest strictly increasing (decreasing) subsequence of σ.*

4. Find a largest increasing (decreasing) subsequence of σ.

Proof. Define the matrix $M(\sigma) = (m_{i,j})$ as follows:

$$m_{i,j} = \begin{cases} 1 & \text{if } \sigma(j) = i; \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to see that a transformation of σ into a list of non-zero entries of $M(\sigma)$ in the order required by Theorem 17 can be accomplished in $O(n)$ time. We consider problems 1 and 2 together and problems 3 and 4 together.

Problems 1 and 2. The decreasing case being similar, we consider only the increasing case. It is easy to see that if the input to (h, w)-SCA is $M(\sigma)$, then it computes the smallest partition of σ into increasing subsequences if $h = w = n$ and the smallest partition of σ into strictly increasing subsequences if $h = w = 1$. Hence these problems can be solved in $O(n \log\log n)$ time.

Problems 3 and 4. The increasing case being similar, we consider only the decreasing case. A largest strictly decreasing subsequence of σ corresponds to a $(1, 1)$-antistaircase in $M(\sigma)$, while a decreasing subsequence corresponds to an (n, n)-antistaircase. It is easy to adapt (h, w)-SCA to find a largest (h, w)-antistaircase in addition to an (h, w)-staircase cover. As the time complexity does not increase, these problems can be solved in $O(n \log\log n)$ time. □

Note that for a sequence σ of arbitrary numbers, the transformation of σ into a list of non-zero entries of $M(\sigma)$ typically requires $O(n \log n)$. In that case, the time complexity of each of the problems in Theorem 18 increases to $O(n \log n)$.

The algorithm (h, w)-SCA can be used to provide an alternate and simpler solution to the maxdominance problem of Atallah and Kosaraju [1]. We denote the co-ordinates of a point p in the Cartesian plane by $(p.x, p.y)$. Given a pair of points p and q in the plane, p *dominates* q if $p.x > q.x$ and $p.y > q.y$. Let P be a set of points in a plane. $\mathrm{MAX}(P)$ is the subset of points in P not dominated by any other point in P. For an arbitrary point p, $\mathrm{DOM}_P(p)$ is the subset of points in P that are dominated by p. The maxdominance problem of Atallah and Kosaraju [1] is:

Maxdominance

Compute $\mathrm{MAX}(\mathrm{DOM}_P(p))$ for all $p \in P$.

Let $|P| = n$, and let $t = \sum_{p \in P} |\mathrm{MAX}(\mathrm{DOM}_P(p))|$. Atallah and Kosaraju provide an $O(t+n \log n)$ time algorithm for Maxdominance. We provide a simpler $O(t+n \log n)$ solution to Maxdominance using (n, n)-SCA. If P is already sorted lexicographically by coordinates, then our solution has time complexity $O(t + n \log\log n)$. This is an improvement over the algorithm of Atallah and Kosaraju because their divide and conquer algorithm requires $\Theta(t + n \log n)$ times even if the input is sorted.

From the set of points P, a matrix $M(P)$ can be constructed as follows. Obtain a sequence σ, by sorting the points in P, in the increasing order of their x-coordinates. Obtain another sequence δ, by sorting the points in P, in the decreasing order of their y-coordinates. Thus, each point $p \in P$, is identified uniquely, by its position in the two sequences. More precisely, with each point $p \in P$, we associate the pair $(\delta^{-1}(p), \sigma^{-1}(p)) \in J_n^2$. The matrix

$M(P) = (m_{i,j})_{n \times n}$ is defined as

$$m_{i,j} = \begin{cases} 1 & \text{if } (\delta^{-1}(p), \sigma^{-1}(p)) = (i,j) \\ 0 & \text{otherwise} \end{cases}$$

Thus the transformation of P into a list of non-zero entries in $M(P)$ requires $O(n \log n)$ time. Let $\Pi = \{\Psi_1, \Psi_2, \ldots, \Psi_k\}$ be a minimum (n,n)-staircase cover of $M(P)$. Rather than think of each staircase Ψ_t, $1 \leq t \leq k$, as containing position indices corresponding to points in P, we think of Ψ_t as containing the points themselves. Let $p \in \Psi_t$. The elements of $\text{MAX}(\text{DOM}_P(p))$ are in $\Psi_{t+1}, \ldots, \Psi_k$; for $t+1 \leq \ell \leq k$, the elements in $\text{MAX}(\text{DOM}_P(p)) \cap \Psi_\ell$ are a contiguous subsequence of Ψ_ℓ with respect to $\leq_{se}$. Figure 6 shows the algorithm to compute $\text{MAX}(\text{DOM}_P(p))$. The correctness of the algorithm is easy to verify (for details see [17]). The steps marked (1) and (2) can be accomplished in $O(\log(|\Psi_\ell|))$ time using binary search, because Ψ_ℓ, $1 \leq \ell \leq k$ is sorted according to $\leq_{se}$. Therefore computing $\text{MAX}(\text{DOM}_P(p))$ for a single $p \in P$ takes $O(\log n)$ time, and Maxdominance can be solved in $O(t + n \log n)$ time. In calculating $\text{MAX}(\text{DOM}_P(p))$ for all $p \in P$, we can eliminate the binary search by doing a linear scan through Ψ_ℓ as we process all $p \in \Psi_t$. If σ is in sorted order, then this reduces the time complexity to $O(t + n \log n \log n)$.

7 Conclusions

One possible direction for future research would follow a connection between stack layouts and matrix covers. This connection does not seem as natural as the one for queue layouts

$S = \emptyset$;
for $\ell := t+1$ **to** k **do**
  if $(S = \emptyset)$ **then**
    $S := \{s \in \Psi_\ell \mid s.x < p.x \text{ and } s.y < p.y\}$ (1)
  else
    Let q be the smallest element in S with respect to $\leq_{se}$;
    Let r be the largest element in S with respect to $\leq_{se}$;
    $S := S \cup \{s \in \Psi_\ell \mid r.x \leq s.x < p.x\}$
    $\cup \{s \in \Psi_\ell \mid q.y \leq s.y < p.y\}$ (2)
$\mathrm{MAX}(\mathrm{DOM}_P(p)) = S$

Figure 6: Algorithm to solve **Maxdominance**

and staircase covers. However, Theorem 14 is actually more natural for stack layouts, giving the same number of 1-stack layouts as there are 1-queue layouts. One intriguing direction is to study properties preserved by the correspondence between 1-stack layouts and 1-queue layouts induced by the intermediary of strings of balanced parentheses.

Acknowledgments

This research was partially supported by National Science Foundation Grant CCR-9009953.

References

[1] M. J. Atallah and S. R. Kosaraju, *An efficient algorithm for maxdominance, with applications*, Algorithmica, 4 (1989), pp. 221–236.

[2] R. Bar-Yehuda and S. Fogel, *Monotone sequences with applications.* Typescript,

1990.

[3] S. N. Bhatt, F. R. K. Chung, F. T. Leighton, and A. L. Rosenberg, *Scheduling tree-dags using FIFO queues: A control-memory tradeoff*, Tech. Rep. 93-48, Department of Computer Science, University of Massachusetts, 1993.

[4] E. Cuthill and J. McKee, *Reducing the bandwidth of sparse symmetric matrices*, in Proceedings of the 24th National ACM Conference, 1969, pp. 157–172.

[5] ——, *Several strategies for reducing the bandwidth of matrices*, in Sparse Matrices and their Application, D. Rose and R. Willoughby, eds., Plenum Press, 1972.

[6] E. W. Dijkstra, *Some beautiful arguments using mathematical induction*, Acta Informatica, 13 (1980), pp. 1–8.

[7] R. P. Dilworth, *A decomposition theorem for partially ordered sets*, Annals of Mathematics, 51 (1950), pp. 161–166.

[8] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, Addison-Wesley Publishing Company, Reading, MA, 1988.

[9] L. S. Heath, F. T. Leighton, and A. L. Rosenberg, *Comparing queues and stacks as mechanisms for laying out graphs*, SIAM Journal of Discrete Mathematics, 5 (1992), pp. 398–412.

[10] L. S. Heath, S. V. Pemmaraju, and C. J. Ribbens, *Sparse matrix-vector multiplication on a small linear array*. Submitted, 1993.

[11] L. S. Heath and A. L. Rosenberg, *Laying out graphs using queues*, SIAM Journal of Computing, 21 (1992), pp. 927–958.

[12] D. B. Johnson, *A priority queue in which initialization and queue operations take $O(\log\log D)$ time*, Mathematical Systems Theory, 15 (1982), pp. 295–309.

[13] R. G. Karlsson and M. H. Overmars, *Scanline algorithms on a grid*, BIT, 28 (1988), pp. 227–241.

[14] C. E. Leiserson, *Area-Efficient VLSI Computation*, MIT Press, Cambridge, MA, 1982.

[15] R. Melhem, *Parallel solution of linear systems with striped sparse matrices*, Parallel Computing, 6 (1988), pp. 165–184.

[16] M. Orlowski and M. Pachter, *An algorithm for the determination of a longest increasing subsequence in a sequence*, Computers Math. Applic., 17 (1989), pp. 1073–1075.

[17] S. V. Pemmaraju, *Exploring the powers of stacks and queues via graph layouts*, PhD thesis, Virginia Polytechnic Institute and State University, 1992.

[18] A. L. Rosenberg, *The DIOGENES approach to testable fault-tolerant arrays of processors*, IEEE Transactions on Computers, 10 (1983), pp. 902–910.

[19] R. E. TARJAN, *Sorting using networks of stacks and queues*, Journal of the ACM, 19 (1972), pp. 341–346.