Applications of Forbidden 0-1 Matrices to Search Tree and Path Compression-Based Data Structures

Seth Pettie*
University of Michigan

Abstract

In this paper we improve, reprove, and simplify several theorems on the performance of data structures based on path compression and search trees. We apply a technique very familiar to computational geometers but still foreign to many researchers in (non-geometric) algorithms and data structures, namely, to bound the complexity of an object via its *forbidden substructures*.

To analyze an algorithm or data structure in the forbidden substructure framework one proceeds in three discrete steps. First, one *transcribes* the behavior of the algorithm as some combinatorial object M; for example, M may be a graph, sequence, permutation, matrix, set system, or tree. (The size of M should ideally be linear in the running time.) Second, one shows that M excludes some forbidden substructure P, and third, one bounds the size of any object avoiding this substructure. The power of this framework derives from the fact that M lies in a more pristine environment and that upper bounds on the size of a P-free object M may be reused in different contexts.

Among our results, we present the first asymptotically sharp bound on the length of arbitrary path compressions on arbitrary trees, improving analyses of Tarjan [35] and Seidel and Sharir [31]. We reprove the linear bound on postordered path compressions, due to Lucas [23] and Loebel and Nešetřil [22], the linear bound on deque-ordered path compressions, due to Buchsbaum, Sundar, and Tarjan [5], and the sequential access theorem for splay trees, originally due to Tarjan [38]. We disprove a conjecture of Aronov et al. [3] related to the efficiency of their data structure for half-plane proximity queries and provide a significantly cleaner analysis of their structure. With the exception of the sequential access theorem, all our proofs are exceptionally simple. Notably absent are calculations of any kind.

1 Introduction

In this paper we demonstrate a simple and effective method for analyzing dynamic data structures that is not based on potential functions, structural invariants, or the like. We show that the running time of several notorious and obscure data structures can be tightly bounded by the number of 1s that can be packed into a 0-1 matrix avoiding certain forbidden 0-1 submatrices or *patterns*. Forbidden submatrices have been successfully applied to various problems in discrete and computational geometry [13, 4, 14, 27, 8, 28], and, since forbidden submatrix theory largely subsumes the theory of Davenport-Schinzel sequences [1] and Turán-type subgraph avoidance problems, one can often restate proofs based on these concepts in terms of forbidden 0-1 matrices.

Despite the pervasive use of the *forbidden substructure method* in computational geometry, the method has had minimal impact on (non-geometric) algorithms and data structures research. This should not be too surprising. Geometric axioms naturally lend themselves to forbidden substructure arguments (two lines do not cross twice, etc.) whereas non-geometric algorithms and data structures are not ruled by axioms in the same way. In this paper we use the forbidden substructure method to analyze several data structures that are based on some form of binary search tree or path compression heuristic. Rather than account for the actual running time of the data structure, we simple find an impossible configuration (often involving no more than 3 objects from the data structure and 2 operations) that can never occur under any execution. We represent the evolution of an n-element data structure under m operations as an $m \times n$ incidence matrix where the (i, j)th entry is 1 if the ith operation *touches* the jth element and 0 otherwise, i.e., the running time of the data structure is linear in the weight of the matrix. The impossible configuration then corresponds to a forbidden 0-1 matrix.

Our Results. We present a new proof that m path compressions on an n-node tree have total length $(1+o(1))m \log_{1+m/n} n$, which matches asymptotic upper

bounds of Tarjan [35] and Seidel and Sharir [31] and is the first analysis to match Fischer's lower bound [10] up to lower order terms. We present considerably simpler proofs that *postordered* path compressions with "rising roots" take $O(m + n)$ time (matching Loebl and Nešetřil [22] and Lucas [23]) and that *deque-ordered* compressions with rising roots take $O(m + n)$ time, matching an upper bound of Buchsbaum, Sundar, and Tarjan [5]. Buchsbaum et al. used deque-ordered path compressions in their data structure for min-deques with catenation. Aronov et al. [3] proposed a data structure for half-plane proximity queries that used a very specialized operation on a binary search tree. They proved each operation took amortized logarithmic time and conjectured the true bound was constant. We disprove the conjecture and give logarithmic upper and lower bounds on the cost of such operations; our upper bound proof is significantly simpler than that of [3]. The operation performed by Aronov et al.'s data structure is actually a generalization of what Sundar [33] called a *twist* in a binary search tree. Improving on Sundar's analysis, we prove that one can only perform a linear number of $(\log n)$-twists in a binary search tree. Finally, we present yet another proof of Tarjan's [38] sequential access theorem for splay trees, which says that splaying the elements of a splay tree in order takes linear time. All proofs are presented in their entirety.

Precursors. It has been known for some time that a specific path compression system could be characterized by forbidden substructure. As an intermediate step in Hart and Sharir's [17] proof that $(ababa)$-free Davenport-Schinzel sequences over an n-letter alphabet have length $\Theta(n\alpha(n))$ they showed an equivalence between $(ababa)$-free DS sequences and what they called generalized postordered path compressions.[1] The appearance of the inverse-Ackermann function in this context suggested a hidden connection to Tarjan's [35] upper bound of $O(n\alpha(n))$ on the length of n arbitrary compressions on a *balanced* n-node tree. Hart and Sharir found no such connection (nor has anyone since) and, rather unexpectedly, there was no followup work exploring the relationship between other brands of path compression and other forbidden substructures. The author [29] recently obtained an $O(n\alpha^*(n))$ bound on *deque-ordered* access sequences in splay trees using generalized Davenport-Schinzel sequences [20, 26], which is a very natural approach for tackling the problem, given Hart and Sharir's [17] work. It is well known [24] that

splaying is an analogue of path compression for binary search trees. Demaine et al. [7] showed an interesting equivalence between the intrinsic cost of an access sequence in a binary search tree and a problem on 0-1 matrices that does not involve forbidden substructure.[2]

Pros and Cons of The Method. Several questions can be raised about the simplicity and necessity of the forbidden substructure method. By appealing to the bounds on the maximum weight of 0-1 matrices, isn't one just sweeping the complexity of a proof under the rug? In principle the answer is *yes*, that's the point of the method. However, in practice this is not an actual concern. For example, in Theorems 2.1, 2.4, and 2.5 we make use of a forbidden pattern used in several geometric applications [13, 4, 27, 8]. Füredi's [13] asymptotically tight analysis of this pattern occupies all of 1/3 of a page and Tardos's [34] analsysis, tight up to lower order terms, occupies less than a page. In Theorems 2.2 and 2.3 we make use of Marcus and Tardos's [25] tight analysis of permutation matrices. Their leisurely proof occupies all of 1 page and a terser version would be half that. The proofs of [13, 34, 25] are completely elementary and represent a powerful set of reusable tools.

Would it not still be simpler (and more illuminating) to "unroll" a proof in this framework and refer directly to features of the algorithm rather than features of a 0-1 matrix? Again, in principle the answer to this question could be yes, but in our experience the opposite is true. To begin with, it is generally not easy to unroll a proof in this framework since we can manipulate 0-1 matrices in ways that do not make sense in the context of a given data structure. For example, rotating a 0-1 matrix is clearly a trivial act; however, when applied to an operation-object incidence matrix for some dynamic data structure, *rotation* exchanges the roles of time and space. Our analysis of arbitrary path compressions, in Theorem 2.1, is invariant under rotation, meaning that any unrolled version of the proof must, at some level, remain indifferent to the distinction between path compressions (time) and nodes (space).

Outline. In Section 1.1 we review some results on forbidden 0-1 matrices. In Sections 2.1–2.5 we exhibit the technique on various problems involving path compression schemes and operations on binary search trees. In Section 3 we conclude with some open problems. The appendix contains some simple generalizations of existing results on square 0-1 matrices [13, 14, 34] to rectan-

---

[1] *Generalized* here means that any subset of the vertices on a path may participate in the compression. *Postordered* means the first node on successive compressions are consistent with some postordering of the tree and that compressions are postorder-preserving.

[2] In their problem the input is an $m \times n$ 0-1 matrix representing the access sequence with exactly one 1 in each row. The problem is to identify the minimum number of 0s that, when flipped, make the matrix "orthogonally satisfied." The worst case is $\Theta(m \log n)$ but could be significantly smaller for individual inputs.

gular matrices.

1.1 Forbidden Submatrices Let A and P be two 0-1 matrices. We say A *avoids* P if there is no submatrix A' of A, having the same dimensions as P, such that each 1 in P corresponds to a 1 in A', i.e., the 0s in the pattern P are "don't cares." Let $\mathrm{Ex}(P, m, n)$ be the maximum weight (number of 1s) in an $m \times n$ matrix avoiding P, where P can be a single matrix or set of forbidden matrices, and let $\mathrm{Ex}(P, n) = \mathrm{Ex}(P, n, n)$. The extremal function $\mathrm{Ex}(P, m, n)$ is obviously invariant under vertical and horizontal reflection of A, but, in general, not invariant under rotations unless $m = n$.

Füredi [13] and Bienstock and Györi [4] initiated the study of this problem and showed that both $\mathrm{Ex}(P_1, n)$ and $\mathrm{Ex}(P_2, n)$ are $\Theta(n \log n)$ [13, 4, 34]. Following convention we write 0-1 matrices with bullets for 1s and blanks for 0s.

$$P_1 = \begin{pmatrix} & \bullet & \bullet \\ \bullet & & \bullet \end{pmatrix} \qquad P_2 = \begin{pmatrix} & & \bullet \\ \bullet & \bullet & \bullet \end{pmatrix}$$

Füredi and Hajnal [14] and Tardos [34] gave tight asymptotic bounds on all forbidden submatrices with weight 4, as well as some tight bounds when there are multiple forbidden submatrices. At present we have an incomplete understanding of weight-5 forbidden matrices [34, 19, 12, 28, 15, 30].

All the known upper and lower bounds on $\mathrm{Ex}(P, n)$, for various P, are either linear, quasilinear (of the form $n2^{\alpha^{O(1)}(n)}$, where α is the inverse-Ackermann function), $\Theta(n\mathrm{polylog}(n))$, or polynomial, i.e., $\Theta(n^c)$ for some $1 < c < 2$. However, it is unknown whether these four categories are exhaustive or too broad[3], nor do we have a perfect characterization for any one of the categories. Nonetheless, large swaths of forbidden submatrices have successfully been categorized. Marcus and Tardos [25], answering a problem posed by Füredi and Hajnal [14], showed that every permutation matrix P has $\mathrm{Ex}(P, n) = O(n)$. Recently Geneson [15] showed that double permutation matrices are also linear. From the quasilinear bounds on generalized Davenport-Schinzel sequences [20, 26] it follows that $\mathrm{Ex}(P, n)$ is either linear or quasilinear if P has one 1 in each column. A 0-1 pattern P can be interpreted as the incidence matrix of a bipartite graph. If P contains a cycle then $\mathrm{Ex}(P, n)$ is polynomial, i.e., $\Omega(n^c)$ for some $1 < c < 2$. This

follows directly from known lower bounds on the size of graphs with a specified girth. The author [30] recently disproved a conjecture of Füredi and Hajnal [14] stating that all acyclic patterns have extremal functions in $O(n \log n)$. In particular there is an acyclic pattern X for which $\mathrm{Ex}(X, n) = \Omega(n \log n \log \log n)$. Pach and Tardos [28] made the weaker and highly plausible conjecture that $\mathrm{Ex}(P, n) = O(n\mathrm{polylog}(n))$ for all acyclic P. (Note that there is no known non-trivial upper bound on $\mathrm{Ex}(P, n)$ for acyclic patterns P.) The Pach-Tardos conjecture has been verified for all weight-5 patterns and all but two weight-6 patterns [28].

It is currently known [19, 15, 30] that there are infinitely many minimal[4] nonlinear forbidden patterns, though only two can be identified. There are also infinitely many minimal non*quasi*linear patterns, though only four such patterns can be identified. See [30] for a discussion of these problems.

2 Applications of the Forbidden Substructure Method

In Sections 2.1–2.3 we reprove some earlier results concerning arbitrary path compressions [35, 31] and postorder and deque-ordered path compressions [23, 22, 5]. Below we review these standard concepts.

Postorder. Let T be a tree rooted at r, which has children $c_1, \ldots, c_k$. A *postorder* of T consists of the concatenation of postorders of the trees rooted at $c_1, \ldots, c_k$ followed by r. (This corresponds to a depth first traversal of T.) Depending on context the children of a vertex may or may not be ordered from left to right. If they are then the postorder is unique.

Path Compression. Let $C = (x_1, x_2, \ldots, x_k)$ be a sequence of nodes in T, where x_{i+1} is the parent of x_i. *Compressing* C yields a new tree T' in which the edges (x_i, x_{i+1}) are replaced by (x_i, x_k). We say that C *originates* at x_1, *terminates* at x_k, and *touches* $x_1, \ldots x_{k-1}$. The *cost* of C is $k - 1$. If T is ordered we say C is *order preserving* if it is possible to arrange $x_1, \ldots, x_{k-1}$ in the order of x_k's chilidren such that the postordering of the *leaves* of T matches their postorder in T'. In particular, any compression that originates from a node on the leftmost or rightmost path of T is order preserving. If u and v are nodes in a tree, $u \lhd v$ means u is a strict descendant of v and $u \unlhd v$ means $u \lhd v$ or $u = v$.

2.1 A Set-Union Data Structure We consider a standard data structure supporting the operations *makeset*, *union*, and *find*, in which elements correspond

---
[3]For example, there may be patterns with extremal functions between the second and third categories, say $\Theta(n \log \log n)$, or functions between the third and fourth, say of the form $n2^{\Theta(1/\sqrt{\log n})}$. A further issue is whether the $n\mathrm{polylog}(n)$ category is too big, i.e., it may be that if $\mathrm{Ex}(P, n) = n\mathrm{polylog}(n)$ then it is also $O(n \log^c n)$ for some fixed c. The case $c = 1$ has been ruled out [30].

---
[4]Minimal with respect to containment and a trivial *stretching* operation; see [30].

to nodes and sets correspond to rooted, unordered trees. A $union(A, B)$ operation makes the root of tree A a child of the root of B. A $find(x)$ operation returns the root r of the tree containing x, but only after performing the path compression $(x, \ldots, r)$. Makeset and union have unit cost and the cost of a find is equal to the cost of the associated path compression. It is well known that the cost of a sequence of $m \geqslant n$ finds on an n-element universe is $O(m \log_{1+m/n} n)$ [35, 31]. With the additional *union-by-rank* or *union-by-size* heuristics the cost becomes $\Theta(n + m\alpha(m, n))$ [35, 36, 31, 21, 11, 18].

THEOREM 2.1. *The cost of n makesets, $n - 1$ unions, and m finds is: $O(n) + \mathrm{Ex}(\{P, P^{\oslash}\}, m, n)$, where:*

$$P = \begin{pmatrix} \vdots & & \cdot \\ & \cdot & \end{pmatrix} \quad and \quad P^{\oslash} = \begin{pmatrix} & & \cdot \\ \cdot & & \cdot \end{pmatrix}$$

The proofs of Theorems 2.1–2.3 reference the *compression-node incidence matrix A*, defined below. The nodes $u_1, \ldots, u_n$ are postordered with respect to some tree T and the compressions are, by default, ordered by time of compression. (Theorem 2.3 arranges compressions in a different order.) Let $x \lessdot y$ indicate that x is equal to or precedes y, where x and y are nodes or compressions.

$$A(i, j) = \begin{cases} 1 & \text{if the } i\text{th compression touches } u_j \\ 0 & \text{otherwise} \end{cases}$$

Let $A(R; C)$ be the submatrix of A restricted to rows R and columns C.

Proof. (Theorem 2.1) Let T be the tree representing all edges formed by the union operations. Notice that since T is not ordered, its postorder is not uniquely determined by the sequence of unions.

P-Freeness. Suppose that $A(a, b; u, v, w)$ contained P, that is, compressions a and b touch u, w and u, v, respectively, where $a \lessdot b$ and $u \lessdot v \lessdot w$. Since u participates in compressions with both v and w it follows that $u \lhd v \lhd w$. See Figure 1, left. After compression
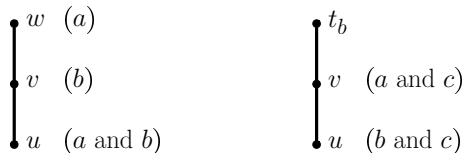

Figure 1: Instances of the patterns P, left, and $P^{\oslash}$, right. The compressions affecting each node are indicated in parentheses.

a, u and v are no longer related so they cannot both be

touched by compression b.

$P^{\oslash}$-Freeness. Now suppose that $A(a, b, c; u, v)$ contained $P^{\oslash}$, that is, u is touched by compressions b and c, v is touched by a and c, $a \lessdot b \lessdot c$, and $u \lessdot v$. Since u and v participate in compression c it follows (from the postorder) that $u \lhd v$ in T. Moreover, since non-roots cannot acquire new descendants, v must be ancestral to u in the union-find forest just before the ath compression. The bth compression terminates at a strict ancestor of v in T, say t_b, thereby making u and v unrelated after compression b. See Figure 1, right. They obviously cannot both be touched by a subsequent compression, namely c.

Tardos [34] showed that $\mathrm{Ex}(P, n) = \mathrm{Ex}(P^{\oslash}, n) = n \log n + O(n)$. Our bound on n path compressions improves an exceptionally careful analysis of Seidel and Sharir [31] by a factor of 2 and matches a lower bound of Fischer [10]. In the appendix we generalize Tardos's proof to rectangular matrices and show that for $m \geqslant n$, $\mathrm{Ex}(P, m, n), \mathrm{Ex}(P^{\oslash}, m, n)$, and $\mathrm{Ex}(\{P, P^{\oslash}\}, m, n)$ are all $(1 + o(1))m \log_{1+m/n} n$. (In other words, the proof of Theorem 2.1 is twice as long as it needs to be; either half suffices to get a $(1+o(1))m \log_{1+m/n} n$ bound.) This bound on the length of m path compressions sharpens analyses of [35, 31] and matches, asymptotically, a lower bound of Tarjan and van Leeuwen [37].

2.2 Postordered Path Compression A sequence of path compressions has the "rising roots" property [23, 5] if, after a compression terminates at a node u, no subsequent compression terminates at a strict descendant of u.

THEOREM 2.2. *The cost of m rising root postordered path compressions on an initial n-node tree is on the order of $\mathrm{Ex}(Q, m, n)$, where:*

$$Q = \begin{pmatrix} & & \cdot \\ & \cdot & \cdot \\ \cdot & & \end{pmatrix}$$

Proof. Let o_i and t_i be the origin and termination point (first and last nodes) of the ith compression. The postorder and rising root conditions imply that $o_i \lessdot o_{i+1}$ and $t_i \lesseqgtr t_{i+1}$. Suppose that $A(a, b, c, d; u, v, w, x)$ contained the pattern Q. In other words, $o_a \lhd x$, $\{o_b, o_c, o_d\} \lhd v \lhd w \lhd x$, and compressions a, b, c, d touch nodes x, v, w and u, respectively. See Figure 2. At the time of the bth compression we still have $\{o_b, o_c\} \lhd v \lhd w$; otherwise the two nodes touched by either compression b or c would be unrelated. The rising roots condition forces $x \lhd t_a \lesseqgtr t_b$. Thus, after the bth compression $v \not\lesseqgtr w$ and in particular, $o_c \not\lesseqgtr w$, a contradiction.
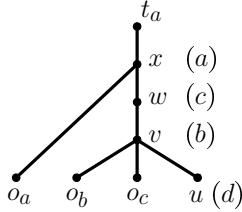
Figure 2: An occurrence of the pattern Q. The compression touching each node is indicated in parentheses.

Notice that Q is a permutation matrix, all of which are known to be linear [25]. An earlier proof of Füredi and Hajnal [14] showed that $\mathrm{Ex}(Q, m, n) = O(m+n)$ but their proof does not extend to other permutation matrices.

2.3 Deque-ordered Path Compression Buchsbaum, Sundar, and Tarjan [5] designed an optimal data structure for *mindeques with catenation* whose analysis amounted to bounding the length of a sequence of *deque-ordered* path compressions. (See [5, §2.1] for several applications of mindeques.) We give an operational definition of deque-ordered rising-root path compressions. Initially there are n single-node, rooted trees. Two trees with roots r, r' may be linked by making r the leftmost or rightmost child of r'. If l is the leftmost or rightmost leaf of a tree rooted at r we may delete l or perform the path compression $(l, \ldots, r)$, which is clearly order-preserving. Buchsbaum et al. [5] proved that the total cost of m such path compressions is $O(m+n)$.

THEOREM 2.3. *The length of m rising-root deque-ordered path compressions on an n-node tree is on the order of $\mathrm{Ex}(R, m, n)$, where:*

$$R = \begin{pmatrix} & & \bullet & & \\ & & & \bullet & & \bullet \\ & & & & \bullet \\ & \bullet & & \bullet & \\ \bullet & & & & \end{pmatrix}$$

Proof. Let T be the ordered tree formed by all linking operations. We bound the cost of all path compressions originating at leftmost leaves; the same analysis can obviously be applied to compressions from rightmost leaves. Let o_i and t_i be the points of origin and termination of compression i. We use the compression-node incidence matrix, where compressions are ordered by point of origin, breaking ties by point of termination, i.e., $i < j$ only if $(o_i, t_i) < (o_j, t_j)$ lexicographically. Now suppose that $A(a, b, c, d, e, f, g; s, u, v, w, x, y, z)$ contains the pattern R, that is, compressions a, b, c, d, e, f, g touch, respectively, u, z, v, y, x, w, s, where $a < \cdots < g$ and $u < \cdots <$

z. It follows from the postordering $o_a < \cdots < o_g < s$ that $\{o_a, o_b, o_c, o_d, o_e, o_f, s\} \vartriangleleft u \vartriangleleft v \vartriangleleft w \vartriangleleft x \vartriangleleft y \vartriangleleft z$. See Figure 3, left. We also claim that compressions
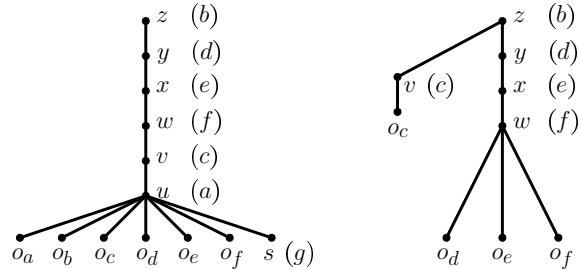

Figure 3: An occurrence of the pattern R. Left: the situation before compression a. Right: before compression c. The compression touching each node is indicated in parentheses.

a, b, c, d, e, and f are performed in that order. If some compression $q \in \{b, c, d, e, f\}$ were performed before a, o_q would have been a leftmost leaf at the time, which implies that o_a (preceding o_q) was not a member of o_q's tree at the time. Since compression q terminates above u and non-root nodes cannot acquire new descendants, o_a could never become a descendant of u. Thus, o_b, o_c, o_d, o_e, and o_f must be in o_a's tree at the time of compression a, and, therefore, be performed in the order b, c, d, e, f. Now consider the position of o_d at the time of compression c. If o_d were still a descendant of v, compression c would cause it to become unrelated to y since, by the rising-roots property, $v \vartriangleleft y \vartriangleleft t_b \trianglelefteq t_c$. This rules out the possibility that o_d is still a descendant of v. Moreover, it implies that o_d is positioned to the right of v. See Figure 3, right. Since o_f is still positioned to the right of d and still a descendant of w, o_d and o_e must also be descendants of w. Thus, compression d causes o_e to become unrelated to x, a contradiction since both must be touched by compression e.

The matrix R obviously represents a permutation, all of which are known to be linear [25].

2.4 Dilbagging a Search Tree Aronov et al. [3] presented a data structure for halfplane proximity queries, one of whose components reshapes a binary tree via special operations we call *dilbags*.[5] They gave an $O(n \log n)$ upper bound on the total cost of n dilbags and conjectured that the worst case cost is actually $O(n)$. We give a significantly simpler proof of Aronov

et al.'s $O(n \log n)$ upper bound using forbidden matrices. We also observe that this bound is *tight*, disproving Aronov et al.'s conjecture. As a byproduct of these results we are also able to solve a problem addressed by Sundar [33], namely, to bound the cost of performing a sequence of *twists* in a binary search tree.

As defined in [3], a dilbag alters a binary search tree by (1) making the tree root the left child of a new root node, followed by (2) reshaping any connected portion of the tree containing the root into a right path. To be more consistent with the presentation from Sections 2.1–2.3 we consider the time- and orientation-reversed version of dilbags. Henceforth, a dilbag consists of $(1')$ reshaping the left path of the binary tree in such a way that the leftmost node becomes the new root, followed by $(2')$ deletion of the root. The cost of a dilbag is *not* the number of rotations needed to effect the reshaping in step $(1')$ but the number of nodes on the left path before step $(1')$ whose parent changes due to step $(1')$.

In Theorem 2.4 we give a significantly simpler proof of Aronov et al.'s $O(n \log n)$ upper bound, which, it turns out, is asymptotically tight.

THEOREM 2.4. *The cost of a sequence of dilbags on an initial n node binary tree is on the order of:* $\mathrm{Ex}(S, n)$, *where:*

$$S = \begin{pmatrix} & \bullet & \bullet \\ \bullet & & \bullet \end{pmatrix}$$

Proof. A node on the left path is *affected* by a dilbag if the identity of the node's parent changes. Call a node u *touched* if u is affected and there are no affected nodes in u's left subtree after the dilbag. The number of touched nodes is at least half the number of affected nodes and therefore at least half the cost of the dilbag. See Figure 4 for an example dilbag, with affected and touched nodes marked. We claim the dilbag-node incidence matrix A is
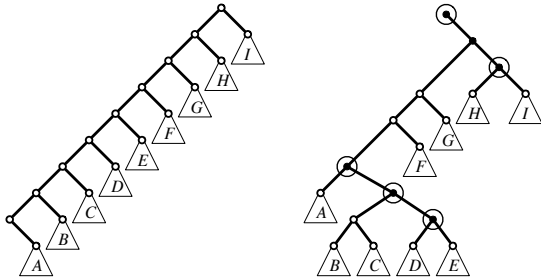

Figure 4: A single dilbag operation. The blackened nodes are *affected*. Those with a halo are *touched*.

S-free, where $A(i, j) = 1$ if the ith dilbag touches node j and 0 otherwise. Suppose $A(a, b, c; u, v, w)$ contained

S. Since v and w are both touched by dilbag a and $v < w$, v cannot be a descendant of w after the dilbag. Just before dilbag b touches w, w is on the leftmost path, which implies that v and everything to the left of v has already been deleted by previous dilbags. Thus, a subsequent dilbag c cannot touch any u preceding v.

The proof above actually shows that A avoids both $P^{\oslash}$ and $P^{\odot}$, in addition to S, where:

$$P^{\oslash} = \begin{pmatrix} \bullet & \bullet \\ & \\ \bullet & \bullet \\ \bullet & \end{pmatrix} \quad \text{and} \quad P^{\odot} = \begin{pmatrix} & \bullet & \bullet \\ \bullet & & \bullet \end{pmatrix}$$

The dilbag operation is a slight generalization of what Sundar [33] called a right twist, in which a selection of edges on the leftmost path are rotated to the right. A twist that rotates k edges is called a k-twist. Sundar proved that the number of k-twists in a sequence of twists is $O(kn^{1+1/k})$, which is tight for fixed k. However, it only implies that the number of $(\log n)$-twists is $O(n \log n)$. The same argument used in the proof of Theorem 2.4 shows that Sundar's bound is not tight.

THEOREM 2.5. *Starting with any initial n-node binary search tree, in any sequence of right twists the number of $(\log n)$-right twists is $\Theta(n)$.*

Proof. Let m be the maximum number of $(\log n)$-twists and let A be the $m \times n$ twist-node incidence matrix for such a sequence of m twists. The weight of A is at least $m \log n$. By the same argument from Theorem 2.4, A is $\{S, P^{\oslash}, P^{\odot}\}$-free and therefore has weight at most $\mathrm{Ex}(P^{\odot}, m, n)$, which Theorem A.1 implies is at most $2m(1 + \log_{1+m/n} n)$. The inequality $m \log n \leqslant 2m(1 + \log_{1+m/n} n)$ has no solutions for $m \geqslant 4n$ and large enough n, giving the upper bound. The lower bound of $\Omega(n)$ on the number of $(\log n)$-right twists is due to Sundar [33].

2.5 Sequential Access in Splay Trees A *splay tree* is a simple binary search tree that reshapes itself in response to each access. After accessing the element x the tree rotates x to the root position by repeatedly performing so-called *zigzig*, *zigzag*, and *zig* steps, which are distinguished by the relationship between x and its current parent p and grandparent g. See Figure 5. If x and p are both left children (or both right children) a zigzig is applied, rotating (p, g) and (x, p) in that order. If x is a left child and p a right child (or vice versa) a zigzag is applied, rotating (x, p) and (x, g) in that order. If g does not exist a zig rotates (x, p), making x the root. In this section we reprove the sequential access theorem, which states that the time to splay each node in order is $O(n)$. Our proof is fairly complex relative other proofs

of the sequential access theorem [33, 9]. The purpose of this section is to illustrate that forbidden 0-1 matrices can be used in situations where there is no apparent forbidden substructure.
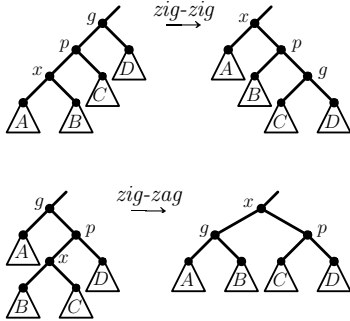

Figure 5: The effect of zigzig and zigzag splay steps.

We assume, for notational simplicity, that nodes are deleted after they are accessed. This is without loss of generality since, after the $(k+1)$th access, nodes $1, \ldots, k$ are never seen again; see Figure 6. If $(x_1, \ldots, x_j)$ is the left path, oriented
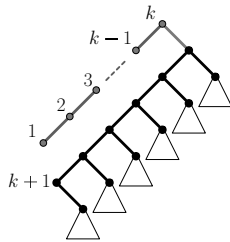

Figure 6: The splay tree after k accesses.

towards the root, splaying x_1 causes the edges $(x_2, x_3), (x_4, x_5), \ldots, (x_{2\lceil(j-1)/2\rceil}, x_{2\lceil(j-1)/2\rceil+1})$ to be rotated and x_1 to be deleted, i.e., replaced by its right subtree. We actually consider a slightly more general semi-splay operation [32]. Let L be a contiguous portion of the left path of the splay tree that includes the root. *Semi-splaying* L consists of rotating every other edge (i.e., it may rotate the even or odd edges), *possibly* followed by the deletion of the leftmost node in the tree.

THEOREM 2.6. *A sequential access of the elements of an n-node splay tree takes time on the order of:* $\mathrm{Ex}(T, n)$, *where:*

$$T = \begin{pmatrix} & \bullet & \\ \bullet & & \\ \bullet & & \bullet \end{pmatrix}$$

Proof. Let V be the set of nodes in the splay tree. Call a subset Z of V *cohesive* if $x, y \in Z$ implies $\mathrm{lca}(x, y) \in$ Z. One can easily prove by induction that during a sequential access of V, any cohesive Z set remains cohesive and the tree induced by Z is subject exclusively to semi-splays on its left path. We actually prove that for any cohesive set Z, performing any number of semi-splays on V requires $O(\mathrm{Ex}(T, |Z|))$ rotations between pairs of nodes in Z.

We begin by partitioning Z into n/B blocks of B consecutive nodes, where $B = O(1)$ will be fixed later, and partitioning time into n/B periods, where the ith period begins when the leftmost node in Z is in block i (the *active* block) and ends when all elements from this block have been deleted. During the ith period a cohesive set $Z_i \subset Z$ is grown, where Z_i is typically not disjoint from $\bigcup_{j<i} Z_j$. We divide the edge rotations performed on Z during a semi-splay into five categories: (i) edges between nodes in the (active) block i, (ii) the edge with exactly one endpoint in block i, (iii) edges between nodes in a common inactive block, (iv) edges between nodes in Z_j, for $j \leq i$, and (v) all other rotations. The cost of (i) is at most $n \log B$ since the left depth[6] of such nodes (w.r.t. block i) is halved in each semi-splay. The cost of (ii) is at most n since there must be fewer than n semi-splays; each semi-splay increases the number of nodes with left depth at most 1. We ignore type (i) and type (ii) rotations for the accounting scheme below.

We call type (iii) and (iv) rotations *paid* and type (v) *unpaid*. The number of type (iii) rotations is at most n since each such rotation reduces the number of nodes on the left path of an inactive block.[7] The number of type (iv) rotations may be bounded inductively. In the analysis below we build an $n/B \times n/B$ period-block incidence matrix A, which is initially zero.

Each *paid* rotation pays for the cost of itself and one other rotation of type (v). Suppose that in period i there is a semi-splay containing unpaid rotations $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$. If the total number of rotations is at least $2k$ then the cost of this semi-splay has already been accounted for and we are done. Otherwise there must be fewer than $4k$ nodes touched by the semi-splay. For each x_l in block j_l, $1 \leq l \leq k$, we set $A(i, j_l) = 1$. (We call this *labeling* block j_l with i.) We include in Z_i *all* nodes touched by the semi-splay, including the root but excluding those in block i. See Figure 7. Let $\mathcal{T}(n)$ be the maximum number of rotations performed by a sequence of left path semi-splays and leftmost node deletions in a cohesive set of n nodes. The above analysis shows that $\mathcal{T}(n) \leq$

---

[6]The left depth of a node x is the number of its strict ancestors greater than x.

[7]The cost of (iii) is already counted in (i) but we keep it distinct nonetheless.
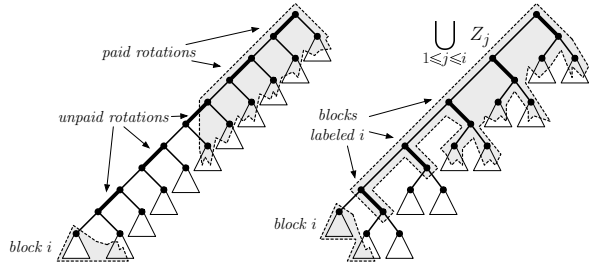
Figure 7: The lower shaded region represents nodes in block i, the upper shaded region to nodes in $\bigcup_{1 \leqslant j \leqslant i} Z_j$.

$n(\log B + 2) + 2 \sum_{i=1}^{n/B} \mathcal{T}(|Z_i|)$. The cost $\mathcal{T}(|Z_i|)$ needs to be doubled since each rotation internal to Z_i is *paid*, and therefore can be used to pay for a rotation outside of Z_i. We claim that A is T-free, which implies that $\sum_i |Z_i| \leqslant 4 \cdot \mathrm{Ex}(T, n/B) < 8n/B$. ($\mathrm{Ex}(T, n')$ is trivially less than $2n'$ [14].) If $A(a, b; p, q)$ contained T then at the beginning of period b blocks p, q are still in the tree and inactive. When a semi-splay in period a performed the unpaid rotation (x, y), where x was in block p, it caused x and *all* ancestors to be placed in Z_a. Thus, in period b no semi-splay could perform an unpaid rotation involving a node in block $q > p$, i.e., involving a node ancestral to x. Assuming inductively that $\mathcal{T}(n) < cn$ we have:

$$
\begin{aligned}
\mathcal{T}(n) &\leqslant n(\log B + 2) + 2 \sum_{i=1}^{n/B} \mathcal{T}(|Z_i|) \\
&\text{where} \quad \sum_{i=1}^{n/B} |Z_i| \leqslant 4 \cdot \mathrm{Ex}(T, n/B) \\
&< n(\log B + 2) + 16cn/B \\
&< cn \quad \{\text{for } B = 45 \text{ and } c = 12\}
\end{aligned}
$$

This concludes the proof of Theorem 2.6.

3   Conclusions

In this paper we illustrated the power of the *forbidden substructure method* to simplify and/or improve the analysis of several data structures based on search trees and path compression. Our analyses typically consisted of nothing more than defining an operation-node incidence matrix A, proving that A avoids some submatrix P, and applying a known bound on the size of a P-free A. Absent from our proofs are excessive definitions, notation, case analyses, and calculations of any kind.[8] At a very high level, the data structures we considered operate on a partial order (represented by

a tree) that is updated through rotations, in the case of search trees, or path compressions. Since trees and partial orders are omnipresent objects in algorithms and data structures, we are certain that these techniques will find many applications outside of discrete and computational geometry.

There are any number of poorly understood algorithms that could succumb to a forbidden substructure type analysis, of which we name just a few. A very appealing data structure of Haeupler et al. [16] maintains an explicit topological order over the strongly connected components of a graph as edges are inserted one by one. It seems likely that the ways in which this permutation evolves could be captured by a forbidden substructure argument. Haeupler et al. proved their algorithm runs in $O(n^{2.5})$ time and conjectured the true bound to be $\tilde{O}(n^2)$. Very recently Demaine et al. [7] demonstrated that a greedy offline binary search tree could be simulated online using a technique that resembles path compression. Is it possible to prove that this algorithm has logarithmic access time using some forbidden substructure argument? Given the existing applications of the method to splay trees (Theorem 2.6 and [29]), one would hope that Cole's notorious proof of the dynamic finger theorem [6] could be simplified. A couple corollaries of the dynamic optimality conjecture for splay trees, namely the split [24] and traversal [32] conjectures, strongly resemble the deque-ordered path compression systems analyzed in Section 2.3. We are optimistic that these conjectures could be attacked using large forbidden permutation matrices, as in the proof of Theorem 2.3.

The viability of the forbidden substructure method ultimately depends on a diverse library of forbidden substructure theorems. The most important open problems, in our opinion, are understanding the properties of forbidden 0-1 matrices that cause a qualitative shift in the extremal function. Despite some recent strong results [19, 25, 15] we currently have no characterization of forbidden matrices P for which $\mathrm{Ex}(P, n)$ is linear, quasilinear, $O(n \mathrm{polylog}(n))$, or $n^{1+\Omega(1)}$.

References

[1] P. Agarwal and M. Sharir. *Davenport-Schinzel Sequences and their Geometric Applications.* Cambridge University Press, 1995.

[2] A. Aggarwal, M. Klawe, S. Moran, P. Shor, and R. Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2:195–208, 1987.

[3] B. Aronov, P. Bose, E. Demaine, J. Gudmundsson, J. Iacono, S. Langerman, and M. Smid. Data structures for halfplane proximity queries and incremental

---

[8] (the proof of the sequential access theorem being the exception that proves the rule.)

voronoi diagrams. In *Proceedings 7th Latin American Theoretical Informatics Symposium*, pages 80–92, 2006.

[4] D. Bienstock and E. Györi. An extremal problem on sparse 0-1 matrices. *SIAM J. Discr. Math.*, 4(1):17–27, 1991.

[5] A. Buchsbaum, R. Sundar, and R. E. Tarjan. Data-structural bootstrapping, linear path compression, and catenable heap-ordered double-ended queues. *SIAM J. Comput.*, 24, 1995.

[6] R. Cole. On the dynamic finger conjecture for splay trees II: The proof. *SIAM J. Comput.*, 30(1):44–85, 2000.

[7] E. Demaine, D. Harmon, J. Iacono, D. Kane, and M. Pătraşcu. The geometry of binary search trees. In *Proc. 20th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 496–505, 2009.

[8] A. Efrat and M. Sharir. A near-linear algorithm for the planar segment center problem. In *Proceedings 5th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 87–97, 1994.

[9] A. Elmasry. On the sequential access theorem and deque conjecture for splay trees. *Theoretical Computer Science*, 314(3):459–466, 2004.

[10] M. J. Fischer. Efficiency of equivalence algorithms. In *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*, pages 153–167, 187–212. Plenum, New York, 1972.

[11] M. L. Fredman and M. Saks. The cell probe complexity of dynamic data structures. In *Proc. 21st annual ACM Symposium on Theory of Computing*, pages 345–354, 1989.

[12] R. Fulek. Linear bound on extremal functions of some forbidden patterns in 0-1 matrices. *Discrete Mathematics*, 309:1736–1739, 2009.

[13] Z. Füredi. The maximum number of unit distances in a convex *n*-gon. *J. Comb. Theory, Ser. A*, 55(2):316–320, 1990.

[14] Z. Füredi and P. Hajnal. Davenport-Schinzel theory of matrices. *Discrete Mathematics*, 103(3):233–251, 1992.

[15] J. T. Geneson. Extremal functions of forbidden double permutation matrices. *J. Combin. Theory, Series A*, 116(7):1235–1244, 2009.

[16] B. Haeupler, T. Kavitha, R. Mathew, S. Sen, and R. Tarjan. Faster algorithms for incremental topological ordering. In *Proceedings 35th Int'l Colloq. on Automata, Languages and Programming (ICALP)*, pages 421–433, 2008.

[17] S. Hart and M. Sharir. Nonlinearity of Davenport-Schinzel sequences and of generalized path compression schemes. *Combinatorica*, 6(2):151–177, 1986.

[18] H. Kaplan, N. Shafrir, and R. E. Tarjan. Meldable heaps and Boolean union-find. In *Proceedings 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 573–582, 2002.

[19] B. Keszegh. On linear forbidden submatrices. *J. Combin. Theory, Series A*, 116(1):232–241, 2009.

[20] M. Klazar. The Füredi-Hajnal conjecture implies the Stanley-Wilf conjecture. In *Formal power series and algebraic combinatorics (Moscow, 2000)*, pages 250–255. Springer, Berlin, 2000.

[21] H. LaPoutré. Lower bounds for the union-find and the split-find problem on pointer machines. *J. Comput. Syst. Sci.*, 52:87–99, 1996.

[22] M. Loebl and J. Nešetřil. Linearity and unprovability of set union problem strategies. I. Linearity of strong postorder. *J. Algor.*, 23(2):207–220, 1997.

[23] J. M. Lucas. Postorder disjoint set union is linear. *SIAM J. on Computing*, 19(5):868–882, October 1990.

[24] J. M. Lucas. On the competitiveness of splay trees: Relations to the union-find problem. In Lyle A. McGeoch and Daniel D. Sleator, editors, *On-line Algorithms*, volume 7 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 95–124, 1991.

[25] A. Marcus and G. Tardos. Excluded permutation matrices and the Stanley-Wilf conjecture. *J. Combin. Theory Ser. A*, 107(1):153–160, 2004.

[26] G. Nivasch. Improved bounds and new techniques for Davenport-Schinzel sequences and their generalizations. In *Proceedings 20th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1–10, 2009.

[27] J. Pach and M. Sharir. On vertical visibility in arrangements of segments and the queue size in the Bentley-Ottmann line sweeping algorithm. *SIAM J. Comput.*, 20(3):460–470, 1991.

[28] J. Pach and G. Tardos. Forbidden paths and cycles in ordered graphs and matrices. *Israel J. Math.*, 155:359–380, 2006.

[29] S. Pettie. Splay trees, Davenport-Schinzel sequences, and the deque conjecture. In *Proc. 19th ACM-SIAM Symposium on Discrete Algorithms*, pages 1115–1124, 2008.

[30] S. Pettie. On nonlinear forbidden 0-1 matrices: A refutation of a Füredi-Hajnal conjecture. In *Proceedings 21st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2010.

[31] R. Seidel and M. Sharir. Top-down analysis of path compression. *SIAM J. Comput.*, 34, 2005.

[32] D. D. Sleator and R. E. Tarjan. Self-adjusting binary search trees. *J. ACM*, 32(3):652–686, 1985.

[33] R. Sundar. On the deque conjecture for the splay algorithm. *Combinatorica*, 12(1):95–124, 1992.

[34] G. Tardos. On 0-1 matrices and small excluded submatrices. *J. Combin. Theory Ser. A*, 111(2):266–288, 2005.

[35] R. E. Tarjan. Efficiency of a good but not linear set merging algorithm. *J. ACM*, 22:215–225, 1975.

[36] R. E. Tarjan. A class of algorithms which require nonlinear time to maintain disjoint sets. *J. Comput. Syst. Sci.*, 18(2):110–127, 1979.

[37] R. E. Tarjan and J. van Leeuwen. Worst-case analysis of set union algorithms. *J. ACM*, 31(2):245–281, 1984.

[38] R.E. Tarjan. Sequential access in play trees takes linear time. *Combinatorica*, 5(4):367–378, 1985.

APPENDIX

A   Upper Bounds for Rectangular 0-1 Matrices Avoiding Forbidden Patterns

The results in this section refer to the following matrices: P and three matrices derived from it by rotation or reflection, $S^{\oslash}$ (a reflection of S, defined in Theorem 2.4) and U:

$$P = \begin{pmatrix} \bullet & & \bullet \\ & \bullet & \end{pmatrix} \qquad P^{\oslash} = \begin{pmatrix} & \bullet & \\ \bullet & & \bullet \end{pmatrix}$$

$$P^{\ominus} = \begin{pmatrix} \bullet & \bullet & \\ & & \bullet \end{pmatrix} \qquad P^{\circlearrowleft} = \begin{pmatrix} \bullet & & \\ \bullet & & \bullet \end{pmatrix}$$

$$U = \begin{pmatrix} \bullet & & \\ & \bullet & \bullet \end{pmatrix} \qquad S^{\oslash} = \begin{pmatrix} \bullet & & \bullet \\ & \bullet & \end{pmatrix}$$

Theorem A.1 generalizes a result of Tardos [34] to rectangular matrices.

THEOREM A.1. *Let $d > 1$ be any real and $c = \min_\epsilon \{\log_d(\frac{1+\epsilon}{\epsilon}) + \log_{1+\frac{1}{d-1}}(1+\epsilon)\}$. For any m, n:*

$$\mathrm{Ex}\,(P, m, n) \leqslant 2m + c^{-1}(m \log_d n + n \log_{1+\frac{1}{d-1}}(n/e)).$$

Proof. Let A be an $m \times n$ 0-1 matrix avoiding P with weight $\mathrm{Ex}(P, m, n)$. Let $l(i) = \min\{j \mid A(i,j) = 1\}$ and $r(i) = \max\{j \mid A(i,j) = 1\}$ be the positions of the leftmost and rightmost 1 in row i. Let $s_i(j) = \min\{j' > j \mid A(i,j') = 1\}$ and $p_i(j) = \max\{j' < j \mid A(i,j') = 1\}$ be the next 1 in row i after position j and before position j, respectively, with the convention that $\min \varnothing = \infty$.

Define the following weight functions over entries in A. Note that w_1 and w_2 assign zero weight to the last two 1s in each row.

$$w_1(i,j) = \begin{cases} 0 & \text{if } A(i,j) = 0 \text{ or} \\ & s_i(j) = \infty \text{ or} \\ & s_i(s_i(j)) = \infty \\ \log_d\left(\frac{r(i)-j}{r(i)-s_i(j)}\right) & \text{otherwise} \end{cases}$$

$$w_2(i,j) = \begin{cases} 0 & \text{as above} \\ \log_{1+\frac{1}{d-1}}\left(\frac{r(i)-j}{s_i(j)-j}\right) & \text{otherwise} \end{cases}$$

$$w(i,j) = w_1(i,j) + w_2(i,j)$$

Note that the non-zero terms in $\sum_j w_1(i,j)$ telescope, which is true for any 0-1 matrix whether it avoids P or not. Summing w_1 over A we have:

$$\sum_{i,j\,:\,w_1(i,j)\neq 0} w_1(i,j) = \sum_i \sum_j \log_d\left(\frac{r(i)-j}{r(i)-s_i(j)}\right)$$
$$= \sum_i \log_d\left(\frac{r(i)-l(i)}{r(i)-p_i(r(i))}\right)$$
$$< m \log_d(n-1)$$

We now consider the sum $\sum_{j,i} w_2(i,j)$. Notice that for any fixed j and $i < i'$ for which $w_2(i,j), w_2(i',j)$ are non-zero, it must be the case that $r(i) < r(i')$ and $s_{i'}(j) \geqslant r(i)$. If not then $A(i,j)$, $A(i,r(i))$, $A(i',j)$, $A(i', s_{i'}(j))$ form an instance of P. See the matrix below with rows and columns labeled:

$$\begin{pmatrix} & j & s_i(j) & r(i) & s_{i'}(j) & r(i') \\ i & \bullet & \bullet & \bullet & & \\ i' & \bullet & & & \bullet & \bullet \end{pmatrix}$$

This observation implies that the w_2-sum of each column also telescopes, since $s_{i'}(j) - j \leqslant r(i) - j$ whenever $i < i'$ and both $w_2(i,j)$ and $w_2(i',j)$ are non-zero. Let $t(j) = \min\{i \mid w(i,j) \neq 0\}$ and $b(j) = \max\{i \mid w(i,j) \neq 0\}$ be the positions of the topmost and bottommost 1s in column j that are not the last or second to last 1 in their row. Then:

$$\sum_{j,i\,:\,w_2(i,j)\neq 0} w_2(i,j) = \sum_{j,i} \log_{1+\frac{1}{d-1}}\left(\frac{r(i)-j}{s_i(j)-j}\right)$$
$$\leqslant \sum_j \log_{1+\frac{1}{d-1}}\left(\frac{r(b(j))-j}{s_{t(j)}(j)-j}\right)$$
$$\leqslant \sum_j \log_{1+\frac{1}{d-1}}(n-j)$$
$$< n \log_{1+\frac{1}{d-1}}(n/e)$$

The second to last inequality follows since $r(b(j)) \leqslant n$ and $s_{t(j)}(j) - j \geqslant 1$ and the last inequality follows since $(n-1)! < (n/e)^n$ for $n \geqslant 7$. For a given $A(i,j) = 1$ let $\frac{r(i)-j}{s_i(j)-j} = 1+\epsilon$. Then :

$$w(i,j) = w_1(i,j) + w_2(i,j)$$
$$= \log_d(\tfrac{1+\epsilon}{\epsilon}) + \log_{1+\frac{1}{d-1}}(1+\epsilon) \quad \{\text{Defn. of } \epsilon\}$$
$$\geqslant c \hspace{4.2cm} \{\text{Defn. of } c\}$$

To put this a different way, for *any* i,j, $A(i,j) \leqslant c^{-1}w(i,j)$ unless $A(i,j)$ is one of the last two 1s in row i. We can now bound the weight of A in terms of m, n, d, and c:

$$\sum_{i,j} A(i,j) \leqslant 2m + \sum_{i,j} c^{-1}w(i,j)$$
$$\leqslant 2m + c^{-1}\left[m \log_d n + n \log_{1+\frac{1}{d-1}}(n/e)\right]$$

Note that it is always the case that $c > 1$. For $\epsilon \geqslant 1/(d-1)$, $w_2(i,j) \geqslant 1$, and for $\epsilon \leqslant 1/(d-1)$, $w_1(i,j) \geqslant \log_d((d-1)(1+1/(d-1))) = 1$.

There is no clean expression for the base d (as a function of the density m/n) that minimizes the bound of Theorem A.1). Once d is fixed the constant c becomes

$\log_d(\frac{1+\epsilon}{\epsilon}) + \log_{1+\frac{1}{d-1}}(1+\epsilon)$, where $\epsilon = \log_d(1 + \frac{1}{d-1})$. When $d = 2 = 1 + \frac{1}{d-1}$, c is also 2, and as d grows c tends to 1.

COROLLARY A.1. *[34]* $\mathrm{Ex}(P, n) < n(\log n + 2n - \frac{\log e}{2})$.

It is not clear *a priori* that $\mathrm{Ex}(P, m, n)$ should be asymptotically equivalent to $\mathrm{Ex}(P, n, m)$ since the transpose of P is neither P nor any reflection of P. However, note that the proof of Theorem A.1 *is* essentially symmetric with respect to the matrix dimensions: the function $d \mapsto 1 + 1/(d-1)$ is an involution.

COROLLARY A.2. *For* $m \geqslant n$, $\mathrm{Ex}(P, m, n) = (1 + o(1))m\log_{1+m/n} n$, *where the* $o(1)$ *is a function of* m/n. *For* $m \leqslant n$, $\mathrm{Ex}(P, m, n) = (1 + o(1))n\log_{1+n/m} m$, *where the* $o(1)$ *is a function of* n/m.

Proof. For $m \geqslant n$, apply Theorem A.1 with $d = 1 + \frac{m}{n \log^2(1+m/n)}$. For $m \leqslant n$, apply Theorem A.1 with $d = 1 + \frac{m \log^2(1+n/m)}{n}$.

The bound from Corollary A.2 is asymptotically tight. Tarjan and van Leeuwen [37] (generalizing a construction of Fischer [10]) showed that for $m \in [n, n^2]$, there is a sequence of m path compressions on an n-node tree with length $\Theta(m \log_{1+m/n} n)$; moreover, these path compressions have the "rising roots" property. It follows (see Theorem 2.1) that their $m \times n$ compression-node incidence matrix is both P-free and $P^{\oslash}$-free and that the transpose $n \times m$ node-compression incidence matrix is both P^T-free and $(P^{\oslash})^T$-free. Since $(P^{\oslash})^T$ is a reflection of P it follows that both $\mathrm{Ex}(P, m, n)$ and $\mathrm{Ex}((P^{\oslash})^T, n, m) = \mathrm{Ex}(P, n, m)$ are $\Omega(m \log_{1+m/n} n)$.[9] We can construct matrices avoiding $\{P, P^{\oslash}\}$ directly, without going through path compressions.

THEOREM A.2. *For integers k, l with $2 \leqslant k < l$*

1. $\mathrm{Ex}(\{P, P^{\oslash}, S^{\circledcirc}\}, \binom{l}{k}, \binom{l}{k-1})) \geqslant k\binom{l}{k}$

2. $\mathrm{Ex}(\{P^{\ominus}, P^{\circlearrowleft}, U\}, \binom{l}{k}, \binom{l}{k-1})) \geqslant k\binom{l}{k}$

Proof. Parts (1) and (2) are derived by selecting a rectangular submatrix from a standard square matrix from the literature [14, 34] avoiding the given patterns. Identify the rows of a matrix A with l-bit strings, k of which are 1, and the columns of A with l-bit strings, $k-1$

of which are 1. Both the rows and columns are ordered lexicographically. The entries of A are as follows:

$$A(i, j) = \begin{cases} 1 & \text{if } i \text{ and } j \text{ differ in one bit position} \\ 0 & \text{otherwise} \end{cases}$$

The number of 1s in A is clearly $k\binom{l}{k}$. It is known that A is $\{P, P^{\oslash}, S^{\circledcirc}\}$-free [14]. Let the matrix A' be defined exactly as A except that the columns are ordered anti-lexicographically (i precedes i' if the reversal of i precedes the reversal of i' lexicographically) while the rows are ordered lexicographically. It is known that A' is $\{P^{\ominus}, P^{\circlearrowleft}, U\}$-free [34].

When Theorem A.2(2) is applied to square matrices, i.e., when $n = \binom{2k-1}{k-1}$ and $m = \binom{2k-1}{k} = n$, the number of 1s in the matrix is $km = kn \approx n(\log n)/2$. That is, Theorem A.2 achieves roughly half the upper bound from Corollary A.1; which bound is closer to the truth is an interesting open problem. Note, however, that Theorem A.2 matches the upper bound from Corollary A.2 up to lower order terms for matrices with very imbalanced dimensions. E.g., for fixed k and $m = n^{1+1/k}/(k+1)$, the construction from Theorem A.2 demonstrates that $\mathrm{Ex}(P, m, n) \geqslant n^{1+1/k} - O(n) = m(1 + \log_{m/n} n) - O(n)$.

---
[9]When $m \gg n$, one obtains an improvement of roughly $m - 2n$ by using $P^{\oslash}$-freeness rather than P-freeness. The additive $2m$ from Theorem A.1 (using P-freeness) becomes $2n$ (using $P^{\oslash}$-freeness) and the $m \log_d n$ term becomes $m \log_d(m/e) \approx m(1 + \log_d n)$ for $d \approx m/n$.