New Bounds and Extended Relations Between Prefix Arrays, Border Arrays, Undirected Graphs, and Indeterminate Strings*

F. Blanchet-Sadri[1] Michelle Bodnar[2] Benjamin De Winkle[3]

October 23, 2015

Abstract

We extend earlier works on the relation of prefix arrays of indeterminate strings to undirected graphs and border arrays. If integer array y is the prefix array for indeterminate string w, then we say w satisfies y. We use a graph theoretic approach to construct a string on a minimum alphabet size which satisfies a given prefix array. We relate the problem of finding a minimum alphabet size to finding edge clique covers of a particular graph, allowing us to bound the minimum alphabet size by $n + \sqrt{n}$ for indeterminate strings, where n is the size of the prefix array. When we restrict ourselves to prefix arrays for partial words, we bound the minimum alphabet size by $\lceil\sqrt{2n}\rceil$. Moreover, we show that this bound is tight up to a constant multiple by using Sidon sets. We also study the relationship between prefix arrays and border arrays. We give necessary and sufficient conditions for a border array and prefix array to be satisfied by the same indeterminate string. We show that the slowly-increasing property completely characterizes border arrays for indeterminate strings, whence there are exactly C_n distinct border arrays of size n for indeterminate strings (here C_n is the nth Catalan number). We give an algorithm to enumerate all prefix arrays for partial words of a given size, n. Our algorithm has a time

*This material is based upon work supported by the National Science Foundation under Grant No. DMS–1060775. Part of this paper was presented at STACS'14 [4].

[1]Department of Computer Science, University of North Carolina, P.O. Box 26170, Greensboro, NC 27402–6170, USA, blanchet@uncg.edu

[2]Department of Mathematics, University of California, San Diego, 9500 Gilman Drive #0112, La Jolla, CA 92093–0112, USA, mbodnar@ucsd.edu

[3]Department of Mathematics, Tufts University, 503 Boston Avenue, Medford, MA 02155, USA, benjamin.de_winkle@tufts.edu

complexity of n^3 times the output size, that is, the number of valid prefix arrays for partial words of length n. We also bound the number of prefix arrays for partial words of a given size using Stirling numbers of the second kind.

Keywords: Indeterminate strings; Partial words; Prefix arrays; Border arrays; Undirected graphs.

1 Introduction

Strings are sequences of letters from a given alphabet. They have been extensively studied and several generalizations have been proposed in the literature which include *indeterminate strings* and *strings with don't cares* [11, 1, 3]. An indeterminate string allows positions to be subsets of cardinality greater than one of a given alphabet, while a string with don't cares allows positions to be the given alphabet. For example, $a\{a,b\}bb\{a,c\}$ is an indeterminate string of length 5 on the alphabet $\{a,b,c\}$ and $a\{a,b,c\}bb\{a,b,c\}$ is a string with don't cares of same length on that alphabet. Indeterminate strings where each position is either a singleton subset or the full alphabet (in other words, a don't care) are also referred to as *partial words* and the don't care symbol is often represented by the $\diamond$ symbol, or *hole* symbol, which represents the alphabet. An alternative way of writing our example $a\{a,b,c\}bb\{a,b,c\}$ is $a\diamond bb\diamond$. Strings where each position is a singleton subset are referred to as *regular strings*.

The fundamental concept of *border array* has played an important role in pattern matching for over four decades [7, 16]. If non-empty strings u_1, u_2, v_1, v_2 exist such that $w = u_1v_1 = v_2u_2$ and u_1 matches u_2, denoted by $u_1 \approx u_2$, then string w has a *border* of length $|u_1| = |u_2|$. The border array β corresponding to a string w of length n is an integer array of same length such that for $j \in [0..n-1]$, $\beta[j]$ is the length of the longest border of $w[0..j]$. For example, $a\{a,b\}bb\{a,c\}$ and $a\diamond bb\diamond$ give rise to the border arrays 01231 and 01234, respectively.

For a regular string w, any border of a border of w is also a border of w; thus w's border array gives all the borders of every prefix of w. This desirable property is lost however, when we consider indeterminate strings or partial words, due to the lack of the transitivity of $\approx$ (e.g., $a \approx \{a,b\}$ and $\{a,b\} \approx b$, but $a \not\approx b$ implying that $w = a\{a,b\}b$ has a border of length 2 having a border of length 1, but w has no border of length 1). Smyth and Wang [17] showed that for indeterminate strings, the concept of *prefix array* provides more information than the one of border array and specifies all the borders of every prefix. The prefix array y corresponding to a string

w of length n is an integer array of same length such that for $j \in [0..n-1]$, $y[j]$ is the length of the longest prefix of $w[j..n)$ that matches a prefix of w. For example, $a\{a,b\}bb\{a,c\}$ and $a\diamond bb\diamond$ give rise to the prefix arrays 53001 and 54001, respectively. Main and Lorentz [14] described the first algorithm for computing the prefix array of any given regular string as a routine in their well-known algorithm for finding all repetitions in a regular string, and Smyth and Wang [17] described an algorithm for efficiently computing the prefix array of any given indeterminate string.

The reverse problem of the one of designing an algorithm that computes the prefix array of any given string is the problem of designing an algorithm that tests if an integer array is the prefix array of some string and, if so, constructs such a string. Clément et al. [6] described an $O(n)$ time algorithm that tests if an integer array of size n is the prefix array of some regular string and, if so, constructs the lexicographically least string having it as a prefix array, the alphabet size of the string being bounded by $\log_2 n$. Recently, Christodoulakis et al. [5] described an algorithm for computing an indeterminate string corresponding to a given prefix array. Such algorithmic characterizations of prefix arrays are not only interesting from a theoretical point of view, but also from a practical point of view, e.g., they help in the design of methods for randomly generating prefix arrays for software testing.

Christodoulakis et al. [5] established quite unexpected connections between indeterminate strings, prefix arrays, and undirected graphs. Among them, they proved the surprising result that every feasible array is the prefix array of some string. In this paper, we extend connections between indeterminate strings, prefix/border arrays, and undirected graphs, which yield combinatorial insights.

The contents of our paper are as follows: In Section 2, we review some basics. In Section 3, we revisit the problem of constructing an indeterminate string on a minimal alphabet satisfying a given prefix array y. We describe two methods: the first one relies on a graph $\mathcal{Q}_y$ built from y's associated prefix graph $\mathcal{P}_y$, while the second one examines induced subgraphs of $\mathcal{P}_y$. It turns out that the minimum alphabet size is the chromatic number of $\mathcal{Q}_y$ (the chromatic number of a graph is the smallest number of colors needed to color the vertices so that no two adjacent vertices share the same color) and is also the size of the smallest induced positive edge cover of $\mathcal{P}_y$. We bound the minimum alphabet size for an array of size n by $n + \sqrt{n}$ using results of Alon, Erdős et al., and Lovász on edge clique covers. In Section 4, we explore the relationship between prefix arrays and border arrays. In particular, we show that every slowly-increasing array is the border array for some indeterminate string, whence the number of border arrays of size

n for indeterminate strings is the nth Catalan number. In Section 5, we restrict prefix arrays to partial words. We give a characterization of such prefix arrays y in terms of the prefix graph $\mathcal{P}_y$. Moreover, we give a method to construct a partial word on the smallest possible alphabet for a given prefix array. We bound the minimum alphabet size for an array of size n by $\left\lceil \sqrt{2n} \right\rceil$, this bound being tight (up to a constant multiple) using results of Erdős and Túran on Sidon sets. We give an algorithm which enumerates all prefix arrays of size n which are valid for partial words. Our algorithm runs in time n^3 times the output size. We also bound the number of prefix arrays of a given size valid for partial words using Stirling numbers of the second kind. Finally in Section 6, we conclude with some suggestions for future work.

2 Preliminaries

Throughout the paper, we use many graph theoretical concepts and constructions. We refer the reader to [12] for an introduction to these ideas.

A *string* w on alphabet A is a sequence of non-empty subsets of A, or may be empty. If A has cardinality μ, we also say that w is a string on μ letters. We call a 1-element subset of A a *regular* letter and larger subsets *indeterminate* letters. A string of all regular letters is called a *regular string* (also referred to as a *word*), and a string which contains at least one indeterminate letter is called an *indeterminate string*. A *hole*, denoted by $\diamond$, is an indeterminate letter which consists of the full alphabet, A. A *partial word* is a string which consists of only regular letters and holes. We denote the length of string w by $|w|$.

For non-negative integers i and j, $i \leq j$, we denote by $[i..j]$ the set of integers $\{i, i+1, \ldots, j\}$ and by $[i..j)$ the set $\{i, i+1, \ldots, j-1\}$. For a string w and integer i such that $0 \leq i < |w|$, the letter (regular or indeterminate) at position i of w is denoted by $w[i]$. We denote by $w[i..j]$ the factor $w[i]w[i+1]\cdots w[j]$ of w when $0 \leq i \leq j < |w|$, and by $w[i..j)$ the factor $w[i]w[i+1]\cdots w[j-1]$ of w when $0 \leq i \leq j \leq |w|$.

Two non-empty subsets of A, α and α', *match* if they have non-empty intersection. We denote this by $\alpha \approx \alpha'$. Similarly, two strings w and w' *match* if $|w| = |w'|$ and $w[i] \approx w'[i]$ for each $i \in [0..|w|-1]$. As before, this is denoted by $w \approx w'$.

An integer array y of size n such that $y[0] = n$ and for every $i \in [1..n-1]$, $0 \leq y[i] \leq n - i$, is called *feasible*. The *prefix array* of a string w of length n is an array y of integers such that $y[j]$ is the length of the longest prefix

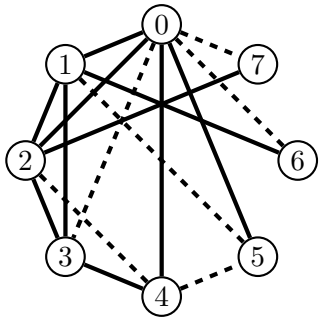

Figure 1: Prefix graph $\mathcal{P}_y$ for $y = 84201300$. Solid lines indicate positive edges and dashed lines indicate negative edges.

of $w[j..n)$ that matches a prefix of w. Note that $y[0]$ is the size of y for any prefix array y. If y is the prefix array of some regular string, then y is called *regular*. If y is the prefix array of some partial word, then y is called *valid for partial words*. If y is the prefix array of a string w, then w *satisfies* y.

Lemma 1. *[5] An integer array y of size n is the prefix array of a string w of length n if and only if for each position $i \in [0..n-1]$, the following two conditions hold: (1) $w[0..y[i]) \approx w[i..i+y[i])$ and (2) if $i + y[i] < n$, then $w[y[i]] \not\approx w[i+y[i]]$.*

The most important graph construction that we use is that of the *prefix graph*, which is introduced in [5]. The prefix graph of a prefix array, y, of size n is denoted by $\mathcal{P}_y$ and is constructed as follows. Its vertex set, $V(\mathcal{P}_y)$, is $[0..n)$. Its edge set consists of two types of edges. Let $i \in [1..n-1]$. For $j \in [0..y[i]-1]$, $\{j, i+j\}$ is a *positive edge*, while for $i+y[i] < n$, $\{y[i], i+y[i]\}$ is a *negative edge* (refer to Lemma 1).

Let $E^+(\mathcal{P}_y)$ be the set of positive edges of $\mathcal{P}_y$ and $E^-(\mathcal{P}_y)$ be the set of negative edges of $\mathcal{P}_y$ (note we may write just E^+ or E^-, respectively, when $\mathcal{P}_y$ is clear from context). We write $\mathcal{P}_y^+ = (V(\mathcal{P}_y), E^+(\mathcal{P}_y))$ (i.e., the graph with the same vertex set, but with only the positive edges) and $\mathcal{P}_y^- = (V(\mathcal{P}_y), E^-(\mathcal{P}_y))$ (same vertex set, only the negative edges). A string w *satisfies* negative edge $\{i, j\}$ if $w[i] \not\approx w[j]$ and w *violates* $\{i, j\}$ if $w[i] \approx w[j]$. Similarly, w *satisfies* positive edge $\{i, j\}$ if $w[i] \approx w[j]$ and w *violates* $\{i, j\}$ if $w[i] \not\approx w[j]$. The graph $\mathcal{P}_y$ encodes all the information of the prefix array y, that is to say, that string w satisfies y if and only if w satisfies all positive and negative edges of $\mathcal{P}_y$. Figure 1 shows an example.

Lemma 2. *Let y be a prefix array and w be an indeterminate string satisfying y. If in $\mathcal{P}_y$, j_0 and j_k are joined by a negative edge and $j_0, j_1, \ldots, j_k$ is a path on only positive edges, then there exists some $i \in [0..k]$ such that $w[j_i]$ is an indeterminate letter.*

Proof. Assume, by way of contradiction, that $w[j_i]$ is a regular letter for each $i \in [0..k]$. Note $w[j_i] \approx w[j_{i+1}]$ for each $i \in [0..k-1]$, because $\{j_i, j_{i+1}\}$ is a positive edge. Moreover, because $w[j_i]$ and $w[j_{i+1}]$ are regular letters, it must be that $w[j_i] = w[j_{i+1}]$. This implies $w[j_0] = w[j_1] = \cdots = w[j_k]$. However, this is a contradiction, because $\{j_0, j_k\}$ is a negative edge, which implies $w[j_0] \not\approx w[j_k]$. □

3 Constructing Indeterminate Strings for Prefix Arrays

Returning to Figure 1, $\{a, c, e\}\{a, b\}\{a, b\}\{b, d\}\{c, d\}ebb$ satisfies the prefix array $y = 84201300$. It is constructed on an alphabet of five letters a, b, c, d, e. Is the alphabet size minimal? The answer is no since

$$\{a, b\}\{a, c\}\{b, c\}\{c, d\}\{a, d\}bcc$$

also satisfies y but is constructed using only the four letters a, b, c, d. We describe two methods for constructing indeterminate strings on a minimum alphabet size that satisfy a given prefix array. For ease of notation, if y is a prefix array, let $\mu(y)$ denote the minimum alphabet size for an indeterminate string that satisfies y.

3.1 Constructing strings with minimum alphabet size: First method

Let us describe our first method.

Let V^+ be the set of vertices of $\mathcal{P}_y$ which are incident to a positive edge. We construct a new graph $\mathcal{Q}$ from $\mathcal{P}_y$ as follows: $V(\mathcal{Q}) = E^+ \cup \{\{i, i\} \mid i \in V \setminus V^+\}$, and $\{\{i_1, j_1\}, \{i_2, j_2\}\} \in E(\mathcal{Q})$ if and only if there exists some $\{r, s\} \in E^-$ such that $r \in \{i_1, j_1\}$ and $s \in \{i_2, j_2\}$. Since a prefix array y defines a unique $\mathcal{P}_y$, which in turn defines a unique $\mathcal{Q}$, we can call this graph $\mathcal{Q}_y$. We show how proper colorings of $\mathcal{Q}_y$ and indeterminate strings satisfying y are related.

Figure 2 gives an example. Since $\mathcal{Q}_y$ has chromatic number 2, associate a with vertices $\{0, 2\}$ and $\{0, 3\}$, and b with vertices $\{1, 3\}$, $\{1, 4\}$, and

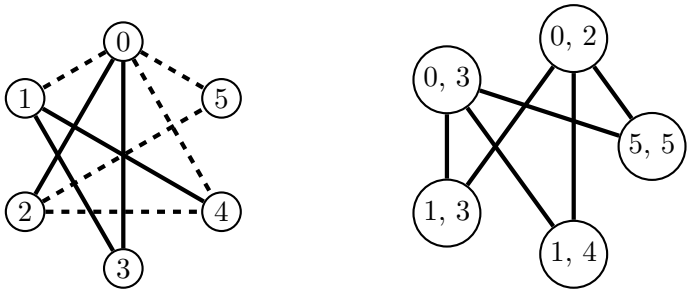

Figure 2: $\mathcal{P}_y$ (left) and $\mathcal{Q}_y$ (right) for the prefix array $y = 602200$, where solid lines indicate positive edges and dashed lines indicate negative edges.

$\{5, 5\}$. By assigning letters to each vertex in $\mathcal{P}_y$ corresponding to the letters associated with its incident positive edges, we obtain the indeterminate $\{a\}\{b\}\{a\}\{a, b\}\{b\}\{b\}$ which satisfies 602200.

Theorem 1. *Let y be a prefix array and let μ be the size of some smallest alphabet on which a string (or an indeterminate string) satisfying y can be constructed. Then $\chi(\mathcal{Q}_y) = \mu$, where $\chi(\mathcal{Q}_y)$ denotes the chromatic number of $\mathcal{Q}_y$.*

Proof. Let $\mathcal{P} = \mathcal{P}_y$ and $\mathcal{Q} = \mathcal{Q}_y$. Suppose w is a string on μ letters which satisfies y, and associate a distinct color to each letter. For each edge $\{i, j\} \in E^+ \cup \{\{i, i\} \mid i \in V \setminus V^+\}$, color the vertex $\{i, j\}$ in $\mathcal{Q}$ with the color associated to the first element in $w[i] \cap w[j]$ (we employ some arbitrary but fixed order on the symbols of the alphabet). The intersection is always nonempty because positive edges and loops represent matchings. Now suppose there is an edge connecting the vertices $\{i, j\}$ and $\{r, s\}$ in $\mathcal{Q}$. Then there is a negative edge in $\mathcal{P}$ connecting one endpoint of $\{i, j\}$ to an endpoint of $\{r, s\}$. Without loss of generality, assume $\{i, r\} \in E^-$. This implies $w[i] \cap w[r] = \emptyset$, so $\{i, j\}$ and $\{r, s\}$ must have different colors. Thus, we have obtained a proper coloring of $\mathcal{Q}$ with μ colors, so $\chi(\mathcal{Q}) \leq \mu$.

Now suppose we are given a proper coloring of $\mathcal{Q}$ using $\chi(\mathcal{Q})$ colors. We may think of each color as a letter and construct an indeterminate string w by assigning to $w[i]$ the color of each $\{k, j\} \in V(Q)$ such that $i = k$ or $i = j$. Let $\{i, j\}$ be any positive edge of $\mathcal{P}$. Then $w[i]$ and $w[j]$ both contain the color given to $\{i, j\}$, so it is satisfied. It remains to show that each negative edge is also satisfied. Suppose $\{i, j\} \in E^-$. Then $w[i] = \{c \mid c \text{ is the color on some } \{i, r\} \in E^+\}$, and $w[j] = \{c \mid c \text{ is the color on some } \{j, s\} \in E^+\}$. Moreover, if $\{i, r\}, \{j, s\} \in E^+$, then

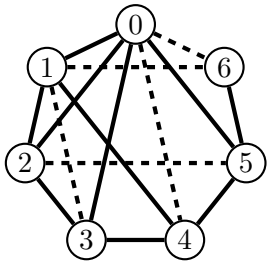

Figure 3: $\mathcal{P}_y$ for $y = 7612010$, where solid lines indicate positive edges and dashed lines indicate negative edges. One IPEC for $\mathcal{P}_y$ is given by the sets $V_0 = \{0, 1, 5\}$, $V_1 = \{0, 2, 3\}$, $V_2 = \{1, 2, 4\}$, and $V_3 = \{3, 4, 5, 6\}$. The construction in Theorem 2 gives the indeterminate string $w = \{a, b\}\{a, c\}\{b, c\}\{b, d\}\{c, d\}\{a, d\}d$, which satisfies y.

they are connected by an edge in $\mathcal{Q}_y$, so they have a different color. Hence $w[i] \cap w[j] = \emptyset$, so $\{i, j\}$ is satisfied. Therefore, w satisfies y. Moreover, w uses at most $\chi(\mathcal{Q})$ letters, which implies $\mu \leq \chi(\mathcal{Q})$. □

3.2 Constructing strings with minimum alphabet size: Second method

Let us describe our second method.

Suppose indeterminate string w on alphabet $A = \{a_0, a_1, \ldots, a_{\mu-1}\}$ satisfies prefix array y, and define $V_i = \{j \mid a_i \in w[j]\}$, i.e., V_i is the set of positions at which the symbol a_i occurs. Notice that the subgraph of $\mathcal{P}_y$ induced by V_i contains no negative edges. Moreover, each positive edge is in the subgraph induced by some V_i. This observation motivates the following definitions.

A subgraph of $\mathcal{P}_y$ is *negative-free* if it does not contain any negative edge. We use the notation $\mathcal{P}_y[V_i]$ to denote the subgraph of $\mathcal{P}_y$ induced by V_i. A set $\{V_0, V_1, \ldots, V_k\}$, where $V_i \subseteq V(\mathcal{P}_y)$, is an *induced positive edge cover* (IPEC) of $\mathcal{P}_y$ if $\mathcal{P}_y[V_i]$ is negative-free for all $i \in [0..k]$, each positive edge of $\mathcal{P}_y$ is in some $\mathcal{P}_y[V_i]$, and each vertex of $\mathcal{P}_y$ is in some $\mathcal{P}_y[V_i]$. Figure 3 gives an example of an IPEC.

Theorem 2. *Let y be a prefix array. The minimum alphabet size for an indeterminate string satisfying y is exactly the size of the smallest IPEC of $\mathcal{P}_y$.*

Proof. Let μ be the minimum alphabet size for an indeterminate string satisfying y, and let σ be the size of the smallest IPEC of $\mathcal{P}_y$. Suppose w is

an indeterminate string that satisfies y on the alphabet $\{a_0, a_1, \ldots, a_{\mu-1}\}$. Let V_i be as defined above. We claim $\{V_0, V_1, \ldots, V_{\mu-1}\}$ is an IPEC of $\mathcal{P}_y$. It is clear that each vertex is in some V_i, because each position of w is non-empty. Suppose $\{i, j\}$ is a negative edge of $\mathcal{P}_y$. Since w satisfies y, it must satisfy $\{i, j\}$, so $w[i] \cap w[j] = \emptyset$. Hence there is no V_k which contains both i and j, which implies $\{i, j\}$ is not in $\mathcal{P}_y[V_k]$ for any k. This holds for any negative edge, so each $\mathcal{P}_y[V_k]$ is negative-free. Now suppose $\{i, j\}$ is a positive edge of $\mathcal{P}_y$. As before, w must satisfy $\{i, j\}$, which implies there exists some $a_k \in w[i] \cap w[j]$. Then $i, j \in V_k$, so $\{i, j\}$ is in the subgraph induced by V_k. This proves our claim and shows $\sigma \leq \mu$.

Now suppose $\mathcal{C} = \{V_0, V_1, \ldots, V_{\sigma-1}\}$ is an IPEC of $\mathcal{P}_y$. Let

$$\{a_0, a_1, \ldots, a_{\sigma-1}\}$$

be a collection of distinct letters and construct an indeterminate string w by setting $w[i] = \{a_j \mid i \in V_j\}$. Since each i is in some V_j, $w[i]$ is non-empty for all i. We claim w satisfies y. Suppose i and j are connected by a negative edge in $\mathcal{P}_y$. Then there is no $V_k \in \mathcal{C}$ which contains both i and j, so by construction, $w[i] \cap w[j] = \emptyset$. Hence all negative edges of $\mathcal{P}_y$ are satisfied. Now suppose i and j are connected by a positive edge. This edge is in $\mathcal{P}_y[V_k]$ for some $V_k \in \mathcal{C}$, which implies $a_k \in w[i] \cap w[j]$, satisfying the positive edge. Thus all edges of $\mathcal{P}_y$ are satisfied, which proves our claim. Note w uses σ letters, so $\mu \leq \sigma$. Therefore, $\mu = \sigma$. □

We can use this construction to bound $\mu(y)$, the minimum alphabet size for an indeterminate string satisfying y, but first we introduce a few concepts. Given a graph G, an *edge clique cover* of G is a set of cliques in G such that each edge of G is in at least one of these cliques. The *edge clique cover number* of G is the size of the smallest edge clique cover of G, which we denote by $\theta(G)$. We denote the complement of G by $\overline{G}$, i.e., the graph defined by $V(\overline{G}) = V(G)$ and two vertices of $\overline{G}$ are joined by an edge if and only if they are not joined by an edge in G.

Lemma 3. *Let y be a prefix array of size n such that $y \neq n00\cdots0$ and $y \neq n(n-2)(n-3)\cdots0$. Then $\mu(y) \leq \theta\left(\overline{\mathcal{P}_y^-}\right)$, where $\overline{\mathcal{P}_y^-}$ contains exactly all positive and non-edges of $\mathcal{P}_y$ as edges.*

Proof. Let $\theta = \theta\left(\overline{\mathcal{P}_y^-}\right)$ and let $\mathcal{C} = \{V_0, V_1, \ldots, V_{\theta-1}\}$ be an edge clique cover of $\overline{\mathcal{P}_y^-}$ (that is, each $V_i \subseteq V(\mathcal{P}_y^-)$). Notice that each positive edge of $\mathcal{P}_y$ is also an edge of $\overline{\mathcal{P}_y^-}$, so each positive edge is in some $\mathcal{P}_y[V_k]$. Moreover, since

each V_k is a clique in $\overline{\mathcal{P}_y^-}$, it must be that $\mathcal{P}_y[V_k]$ is negative-free. Suppose that there exists some vertex, i, in $\mathcal{P}_y$ such that $i \notin V_k$ for all $V_k \in \mathcal{C}$. This implies that i is connected to every other vertex by a negative edge. Notice that by our construction of $\mathcal{P}_y$, if $\{i-1, i\}$ and $\{i, i+1\}$ are both edges in $\mathcal{P}_y$, they cannot both be negative. However, given our condition on i they must both be negative, unless either $i = 0$ or $i = n-1$ (in which case one of these edges does not exist). Thus, $i = 0$ or $i = n-1$, but in these cases, y must be one of the excluded prefix arrays. So, each vertex of $\mathcal{P}_y$ is in some V_k. Hence $\mathcal{C}$ is also an IPEC of $\mathcal{P}_y$, and, by Theorem 2, $\mu(y) \leq \theta$. □

Edge clique covers have been well studied, and we can use results on them to bound $\mu(y)$. Specifically, we use the following results.

Theorem 3 ([9, Theorem 2])**.** *Let G be a graph on n vertices. Then $\theta(G) \leq \lfloor \frac{n^2}{4} \rfloor$.*

Theorem 4 ([13, Theorem 5])**.** *Let G be a graph on n vertices with $m \geq \lfloor \frac{n^2}{4} \rfloor$ edges. Further, set $k = \binom{n}{2} - m$ (i.e., k is the number of edges in $\overline{G}$) and let t be the greatest integer such that $t^2 - t \leq k$. Then $\theta(G) \leq k + t$.*

Theorem 5 ([2, Theorem 1.4])**.** *Let G be a graph on n vertices with maximum degree d. Then $\theta(\overline{G}) \leq 2e^2(d+1)^2 \ln n$, where e is the base of the natural logarithm.*

Now we can state a bound on $\mu(y)$.

Theorem 6. *Let y be a prefix array of size n and let r be the number of negative edges in $\mathcal{P}_y$. Also let e be the base of the natural logarithm and let d be the maximum degree of a vertex in $\mathcal{P}_y^-$. Then $\mu(y) \leq \min\{r + \sqrt{r} + 1, 2e^2(d+1)^2 \ln n\}$,*

Proof. We use Lemma 3, so we first check the cases $y = n00\cdots0$ and $y = n(n-2)(n-3)\cdots0$. Suppose $y = n00\cdots0$. Notice that y is satisfied by $abb\cdots b$. Similarly, if $y = n(n-2)(n-3)\cdots0$, then y is satisfied by $aa\cdots ab$. In both of these cases, any string satisfying y must have at least two letters. Moreover, in both of these cases $r = n-1$, hence $\mu(y) = 2 \leq \min\{r + \sqrt{r} + 1, 2e^2(r+1)^2 \ln n\}$ and the result holds.

Thus, by Lemma 3, we may assume that $\mu(y) \leq \theta\left(\overline{\mathcal{P}_y^-}\right)$. Let $\theta = \theta\left(\overline{\mathcal{P}_y^-}\right)$. It follows from Theorem 5 that $\theta \leq 2e^2(d+1)^2 \ln n$. To get the $r + \sqrt{r} + 1$ bound, we apply Theorem 4, but this requires that $\overline{\mathcal{P}_y^-}$ has at

least $\lfloor \frac{n^2}{4} \rfloor$ edges. Note that $\overline{\mathcal{P}_y^-}$ has $\binom{n}{2} - r$ edges. Since $r < n$, the number of edges in $\overline{\mathcal{P}_y^-}$ is at least $\binom{n}{2} - (n-1) = \frac{n^2-3n+2}{2}$.

Define the function $f(n) = \frac{n^2-3n+2}{2} - \frac{n^2}{4} = \frac{n^2-6n+4}{4}$. Whenever f is positive, $\overline{\mathcal{P}_y^-}$ has at least $\frac{n^2}{4}$ edges. Note that $f(6) = 1 > 0$, and $f'(n) = \frac{n-3}{2}$ is positive for any $n > 3$. Hence f is positive for all $n \geq 6$. Note that the complement of $\overline{\mathcal{P}_y^-}$ is $\mathcal{P}_y^-$, which has r edges. Hence, by Theorem 4, if $n \geq 6$ and t is the greatest integer such that $t^2 - t \leq r$, then $\theta \leq r + t$. Notice that $t < \sqrt{r} + 1$, so $\theta < r + \sqrt{r} + 1$. This just leaves the cases where $n < 6$. Note that in these cases, $r + \sqrt{r} + 1 < 2e^2(d+1)^2 \ln n$, because $r - 1 < n$. Moreover, we can easily check that for each combination of n and r, either $\overline{\mathcal{P}_y^-}$ has at least $\frac{n^2}{4}$ edges, or $\frac{n^2}{4} < r + \sqrt{r} + 1$. In the former case, we can apply Theorem 4 to give $\theta < r + \sqrt{r} + 1$. In the latter case, Theorem 3 gives us $\theta \leq \frac{n^2}{4}$, implying $\theta < r + \sqrt{r} + 1$. □

Since $r \leq n - 1$, we get the following corollary.

Corollary 1. *Let y be a prefix array of size n. Then $\mu(y) \leq n + \sqrt{n}$.*

The following proposition exhibits a prefix array whose negative graph contains a complete graph K_n and which requires at least n letters to be satisfied (K_n has chromatic number n), showing that there exist prefix arrays which require an arbitrarily large alphabet.

Proposition 1. *For all $n \geq 1$, there exists a prefix array y of size 2^{n-1} such that $\mathcal{P}_y^-$ contains a K_n.*

Proof. We provide a construction of such an array. For each $n > 1$ construct the prefix array y_n recursively as follows: let $y_1 = 1$ and $y_n = 2^{n-1} y'_{n-1} (2^{n-2} - 1) y'_{n-1}$, i.e., y_n starts with the number 2^{n-1} followed by the array y'_{n-1} followed by the number $2^{n-2} - 1$ followed by the array y'_{n-1}, where $y'_k = y_k[1..2^{k-1})$, i.e., y'_k is the factor of y_k starting at position 1 and ending at position $2^{k-1} - 1$, for all $k \geq 1$. Clearly, y_n has size 2^{n-1}. Assume that y_n creates a K_n in $\mathcal{P}_{y_n}^-$ on the vertices labeled 0, 1, $2^2 - 1$, $2^3 - 1, \ldots, 2^{n-1} - 1$ and consider the prefix array $y_{n+1} = 2^n y'_n (2^{n-1} - 1) y'_n$. The induced subgraph of $\mathcal{P}_{y_{n+1}}^-$ on its first 2^{n-1} vertices is identical to $\mathcal{P}_{y_n}^-$, which we assumed contains a K_n at vertices whose labels are $2^k - 1$ for $0 \leq k < n$. We show that every such vertex is connected to $2^n - 1$ by a negative edge, which creates a K_{n+1}.

Recall that every negative edge is of the form $\{y[i], i + y[i]\}$. Since every vertex $2^k - 1$ in $\mathcal{P}_{y_n}^-$ is connected to $2^{n-1} - 1$ by a negative edge, i.e., an edge of the form $\{2^k - 1, 2^{n-1} - 1\}$, we have $y_n[2^{n-1} - 2^k] = 2^k - 1$ for $0 \leq k \leq n - 2$.

Since every entry of $y_n[1..2^{n-1})$ occurs in y_{n+1} at its original position plus 2^{n-1} we have $y_{n+1}[2^{n-1}+2^{n-1}-2^k] = y_{n+1}[2^n-2^k] = 2^k-1$ for $0 \leq k < n$, so there is a negative edge from 2^k-1 to $2^n-2^k+2^k-1 = 2^n-1$. These connect all vertices of the K_n created by y_n to the last vertex in $\mathcal{P}^-_{y_{n+1}}$, creating the desired K_{n+1}. By induction on n, we conclude that y_n is a prefix array of size 2^{n-1} whose negative prefix graph contains a K_n. □

4 Connecting Prefix Arrays and Border Arrays

An indeterminate string w of length n has a *border* of length $\ell \in [0..n-1]$ if $w[0..\ell) \approx w[n-\ell..n)$. The *border array* $\beta[0..n)$ of an indeterminate string w is an integer array such that $\beta[0] = 0$ and for $i > 0$, $\beta[i]$ is the length of the longest border of $w[0..i]$. For example, $a\{a,b\}\{a,b\}bac$ has border array $\beta = 012330$. A prefix array y *satisfies* an array β if all strings with prefix array y also have border array β.

Theorem 7. *Let β be an array of size n. Then a prefix array y satisfies β if and only if the following two conditions hold: (1) $\beta[j] \leq y[j-\beta[j]+1]$ for all $j \in [0..n-1]$, and (2) $y[i] \leq j-i$ for all $j \in [0..n-1], i \leq j-\beta[j]$.*

Proof. ($\Rightarrow$) Let y be a prefix array that satisfies β and w be any string with prefix array y. Since $w[0..j]$ has a maximal border of length $\beta[j]$, we have $w[0..\beta[j]) \approx w[j-\beta[j]+1..j]$. Since $y[j-\beta[j]+1]$ gives the length of the longest prefix of w that matches a prefix of $w[j-\beta[j]+1..|w|)$, we have $y[j-\beta[j]+1] \geq \beta[j]$ which gives (1). Now let $i \leq j-\beta[j]$ and $y[i] = j-i+r$ for some r. Then $w[0..j-i+r) \approx w[i..j+r)$. If $r > 0$ then $w[0..j-i] \approx w[i..j]$, so $w[0..j]$ has a border of length at least $j-i+1$. However, since $\beta[j] \leq j-i$, $w[0..j]$ has a maximal border of length $j-i$, so $r \leq 0$. Therefore $y[i] \leq j-i$, so (2) must hold.

($\Leftarrow$) For the reverse implication, let y be a prefix array satisfying (1) and (2), and w be a string with prefix array y. We show that w must have border array β. Let $j \in [0..n-1]$ be arbitrary. By (1) the factor of w of length $\beta[j]$ starting at position $j-\beta[j]+1$ matches $w[0..\beta[j])$, so $w[0..j]$ has a border of length $\beta[j]$. To see that this border is maximal, suppose there exists a border of $w[0..j]$ with length $\beta[j]+r$ for some $r \geq 1$. Then $y[j-\beta[j]-r+1] \geq \beta[j]+r$ which contradicts (2). Thus, y satisfies β. □

This characterization of prefix arrays which satisfy a given array leads to a natural question: Given an array, what degree of freedom do we have in creating a prefix array which satisfies it? The following corollary answers

this question, but requires one property of border arrays referred to as the *slowly-increasing property*, stated in the following proposition.

Proposition 2. *For any border array $\beta = \beta[0..n)$, $\beta[j+1] \leq \beta[j] + 1$ for all $j \in [0..n-2]$.*

Proof. Let w be an indeterminate string such that β is its border array. Suppose $\beta[j+1] = i$ for some integer i. This means that $w[0..i) \approx w[j+1-(i-1)..j+1]$. So, $w[0..i-1) \approx w[j+1-(i-1)..j]$, which implies that $\beta[j] \geq i-1$. Thus, $\beta[j+1] = i \leq \beta[j]+1$. □

Corollary 2. *Let y be a prefix array that satisfies array β. Then β completely determines $y[i]$, where $i > 0$, if and only if $\beta[i] = 0$ or there exists some j such that $i = j - \beta[j] + 1$. Moreover, if $i = j - \beta[j] + 1$ for some j, then $y[i] = \beta[j]$ for the largest j with this property.*

Proof. First note that by Theorem 7(2), $y[i]$ is completely determined by β to be 0 if and only if $\beta[i] = 0$. Now suppose $y[i]$ is completely determined by β and $y[i] > 0$. Since β forces a lower bound on $y[i]$, there must exist some j such that $i = j - \beta[j] + 1$. Finally, assume $i = j - \beta[j] + 1$ for some j. Moreover, assume that j is the largest integer with this property. By Theorem 7(1), $\beta[j] \leq y[i]$. If $j = n-1$, then $i = n - \beta[j]$, so $n - i = \beta[j] \leq y[i]$. Since $n - i$ is the greatest feasible value for $y[i]$, it follows that $y[i] = \beta[j]$. Hence we may assume $j < n-1$. By the slowly-increasing property, $\beta[j+1] \leq \beta[j] + 1$; however, if $\beta[j+1] = \beta[j] + 1$, then $i = (j+1) - \beta[j+1] + 1$, which contradicts the maximality of j. Thus $\beta[j+1] \leq \beta[j]$. This implies $i = j - \beta[j] + 1 \leq (j+1) - \beta[j+1]$. Hence by Theorem 7(2), $y[i] \leq (j+1) - i = (j+1) - (j - \beta[j] + 1) = \beta[j]$. Thus $y[i] = \beta[j]$. □

The slowly-increasing property characterizes border arrays of indeterminate strings.

Theorem 8. *Every slowly-increasing array is the border array for some indeterminate string.*

Proof. Let β be any slowly-increasing array. Since every prefix array is satisfied by some indeterminate string, it will suffice to show that there exists a prefix array y such that y satisfies β. Conditions (1) and (2) from Theorem 7 impose many constraints on possible assignments of values to $y[i]$ for $i \in [0..n-1]$: (1) gives lower bounds, and (2) gives upper bounds. We need to show that no lower bound ever strictly exceeds any upper bound for some position of y, and that no lower bound ever forces an entry of y to be so large that it becomes infeasible.

Suppose that for some position i, a lower bound on $y[i]$ from (1) strictly exceeds an upper bound on $y[i]$ from (2). In particular, Conditions (1) and (2) give

$$\begin{cases} \beta[j] \le y[i] & j \in [0..n-1], \\ y[i] \le j' - i & j' \in [0..n-1] \text{ and } i \le j' - \beta[j']. \end{cases}$$

If the bounds are contradictory, we must have $\beta[j] > j' - i$, where i takes the form $i = j - \beta[j] + 1$. In particular, $\beta[j] > j' - j + \beta[j] - 1$, which is equivalent to $j + 1 > j'$. However, we must have $i \le j' - \beta[j']$ which means $j - \beta[j] + 1 \le j' - \beta[j']$, or $j - j' + 1 \le \beta[j] - \beta[j']$. However, this contradicts the slowly-increasing property because it implies that the array has increased by strictly more than $j - j'$ between positions j' and j. Since we assume β is slowly-increasing, this can never happen.

Second, we must check that no lower bound ever forces an entry of y to be so large that it becomes infeasible. This could only happen if $\beta[j] > n - (j - \beta[j] + 1)$, which is equivalent to $j + 1 > n$. However, this can never occur because we assume $j \in [0..n-1]$. Therefore there must exist a satisfying assignment of entries to a feasible array y which satisfies Conditions (1) and (2) of Theorem 7, so y satisfies β. □

The following theorem counts the total number of slowly-increasing arrays of a given size.

Theorem 9. *For all $n \ge 1$, the number of slowly-increasing arrays of size n is $C_n = \frac{1}{n+1}\binom{2n}{n}$, the nth Catalan number.*

Proof. All border arrays $\beta = \beta[0..n)$ have the slowly-increasing property. The number of binary trees with n nodes is known to be C_n, so we exhibit a bijection between these trees and slowly-increasing arrays. Refer to Figure 4 to see an example of the construction outlined below. Define the *right-depth* of a node in a binary tree to be the number of right children that must be traversed from the root to reach that node.

Slowly-increasing arrays to trees: Consider a slowly-increasing array $\beta = \beta[0..n)$. We choose for $\beta[0]$ to correspond to the root of the tree. Then, for each $j \in [0..n-2]$, if $\beta[j+1] \le \beta[j]$, begin from the deepest node of right-depth $\beta[j+1]$ on the path from the root to the node for $\beta[0..j]$ and add a left child corresponding to $\beta[0..j+1]$. If $\beta[j+1] = \beta[j] + 1$, add a right child to the node corresponding to $\beta[0..j]$, and make the new node correspond to $\beta[0..j+1]$. All that remains to be shown is that a node with the appropriate right-depth exists, and that the child that we need to add is not already present in the tree. The appropriate right-depth exists on the path because

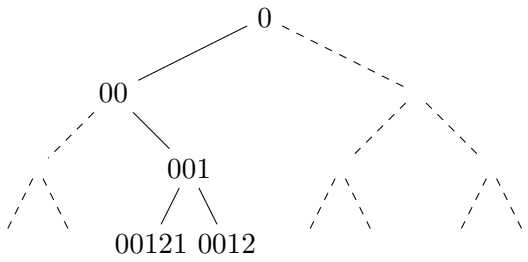

Figure 4: Binary tree corresponding to border array 00121.

in this construction, each node is inserted at right depth $\beta[j+1]$. Also, if a node acquires a right child, then it does so before it acquires a left child, so the child that needs to be added is not already present when it is time to add the child. Hence, we construct a tree from β.

Trees to slowly-increasing arrays: Given a binary tree T with n nodes, we construct a slowly-increasing array. Start with an empty array β. We perform a depth-first traversal of T where we go right before going left. Whenever we encounter a node for the first time, we append its right-depth to the end of β. Clearly, the right-depth cannot increase by more than one at a time when doing this traversal, so β is slowly-increasing. □

This gives us the following corollary.

Corollary 3. *The number of distinct border arrays of size n for indeterminate strings is exactly the n^{th} Catalan number, $C_n = \frac{1}{n+1}\binom{2n}{n}$.*

5 Restricting Prefix Arrays to Partial Words

The next theorem gives a characterization of prefix arrays which can be satisfied by a partial word. It is similar to a characterization of regular prefix arrays given by [5, Lemma 10]. We first define $V^-(\mathcal{P}_y)$ (or just V^- if $\mathcal{P}_y$ is clear from context) to be the set of vertices in $\mathcal{P}_y$ which are incident to a negative edge. The proof idea is that $w[i]$ is a regular letter for all $i \in V^-$, which implies that the entire connected component of i in $\mathcal{P}_y^+[V^-]$ must be the same letter. Figure 5 gives an example of a prefix array valid for partial words.

Theorem 10. *Let y be a prefix array and let V^- be the set of vertices in $\mathcal{P}_y$ which are incident to a negative edge. The following are equivalent: (1)*

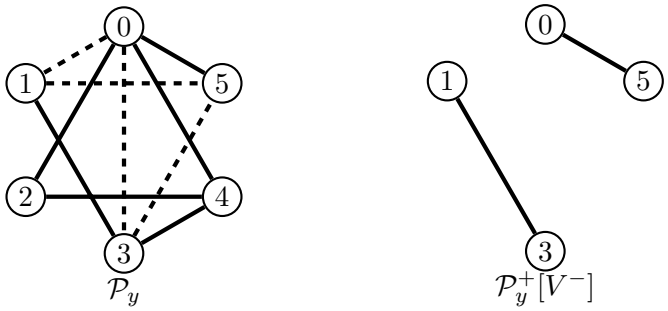

Figure 5: $\mathcal{P}_y$ (left) and $\mathcal{P}_y^+[V^-]$ (right) for the prefix array $y = 603011$ satisfied by the partial word $ab\diamond b\diamond a$, where solid lines indicate positive edges and dashed lines indicate negative edges.

the array y is the prefix array for some partial word, (2) every cycle in $\mathcal{P}_y$ which contains exactly one negative edge has at least one vertex which is not in V^-, and (3) every negative edge of $\mathcal{P}_y$ connects vertices in two different connected components of $\mathcal{P}_y^+[V^-]$.

Proof. First we prove (1) implies (3) by contraposition. Assume $\{i, j\}$ is a negative edge in $\mathcal{P}_y$ which connects vertices i and j, where i and j lie in the same connected component of $\mathcal{P}_y^+[V^-]$. Then there exists a path, p, in $\mathcal{P}_y^+[V^-]$ from i to j. Further suppose w is an indeterminate string which satisfies y. Since each edge in path p is a positive edge, Lemma 2 implies there exists some k such that k is in this path and $w[k]$ is an indeterminate letter. However, $k \in V^-$, so $w[k]$ cannot be a hole. Since holes are the only indeterminate letters allowed in a partial word, w is not a partial word.

Next we prove (2) implies (3), again by contraposition. Suppose $\{i, j\}$ is a negative edge such that i and j are in the same connected component of $\mathcal{P}_y^+[V^-]$. Then there exists a path from j to i which lies in $\mathcal{P}_y^+[V^-]$. Note that all the edges in this path are positive, and all the vertices are in V^-. Then concatenating $\{i, j\}$ to this path creates a cycle in $\mathcal{P}_y$ which contains exactly one negative edge, but whose vertices all are in V^-.

Finally, we prove (3) implies (1). Assume every negative edge of $\mathcal{P}_y$ connects vertices in two different connected components of $\mathcal{P}_y^+[V^-]$. Let $C_0, C_1, \ldots, C_{\ell-1}$ be the connected components of $\mathcal{P}_y^+[V^-]$, and construct a partial word w on the alphabet $\{a_0, a_1, \ldots, a_{\ell-1}\}$ by setting $w[i] = a_j$ if $i \in C_j$ and $w[i] = \diamond$ if $i \notin V^-$. Note that this construction does indeed assign one letter (or hole) to each position of w, and we claim that w satisfies y.

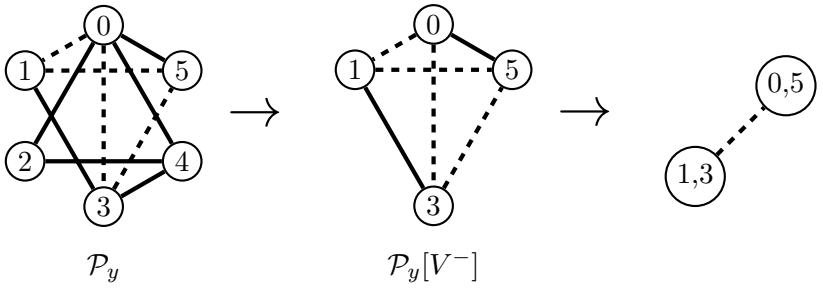

Figure 6: Example construction of the graph $\mathcal{C}_y$ for prefix array $y = 603011$.

The proof for (3) implies (2) is symmetric.

Suppose $\{i, j\}$ is a negative edge in $\mathcal{P}_y$. By assumption, i and j are in different connected components of $\mathcal{P}_y^+[V^-]$. Because of $w[i] \not\approx w[j]$, this edge is satisfied. Now suppose $\{i, j\}$ is a positive edge in $\mathcal{P}_y$. If $i \notin V^-$, then $w[i] = \diamond$, which implies $w[i] \approx w[j]$ and w satisfies $\{i, j\}$. A symmetric argument holds for $j \notin V^-$. Now assume $i, j \in V^-$. This implies i and j are in the same connected component of $\mathcal{P}_y^+[V^-]$, so $w[i] = w[j]$, satisfying $\{i, j\}$. Therefore w satisfies all edges of $\mathcal{P}_y$, proving that y is the prefix array for the partial word w. □

5.1 Constructing partial words for prefix arrays

Based upon the construction given in the above proof, construct the graph $\mathcal{C}_y$ as follows: make one vertex in $\mathcal{C}_y$ for each connected component in $\mathcal{P}_y^+[V^-]$ and join two vertices in $\mathcal{C}_y$ if and only if there exists a negative edge in $\mathcal{P}_y$ between their corresponding components. Figure 6 illustrates this graph construction.

If y is the prefix array for some partial word, $\mu_\diamond(y)$ will denote the minimum alphabet size for a partial word satisfying y.

Theorem 11. *Let y be the prefix array for some partial word. Then $\mu_\diamond(y) = \chi(\mathcal{C}_y)$.*

Proof. Let $C_0, C_1, \ldots, C_{\ell-1}$ be the connected components of $\mathcal{P}_y^+[V^-]$ and assume we have a valid coloring of $\mathcal{C}_y$. We treat these colors as letters and construct w using a similar method to the one given in the proof of Theorem 10. We let $w[i]$ be the color of C_k if $i \in C_k$ and let $w[i] = \diamond$ otherwise. The proof that w satisfies y follows exactly the last part of the proof of Theorem 10. Hence $\mu_\diamond(y) \leq \chi(\mathcal{C}_y)$.

Let w be a partial word satisfying y on an alphabet A of minimum size. Suppose i and i' are in the same connected component of $\mathcal{P}_y^+[V^-]$. Then there exists a path, $i = j_1, j_2, \ldots, j_k = i'$, of positive edges from i to i' such that each vertex in this path is in V^-. Clearly, $w[j_\ell]$ must be a regular letter for each $\ell \in [1..k]$. Then, since j_ℓ and $j_{\ell+1}$ are joined by a positive edge for each $\ell \in [1..k-1]$, it follows that $w[j_1] = w[j_2] = \cdots = w[j_k]$. Hence $w[i] = w[i']$. This implies that all positions of w associated with vertices in a given connected component of $\mathcal{P}_y^+[V^-]$ must have the same letter.

Now we give a coloring of $\mathcal{C}_y$ using the letters in A. Color a vertex v of $\mathcal{C}_y$ with $a \in A$ if there exists some i in the connected component of $\mathcal{P}_y^+[V^-]$ represented by v such that $w[i] = a$. It follows from the above discussion that this coloring operation is well-defined. We claim that this is a valid coloring. Suppose u and v are vertices of $\mathcal{C}_y$ joined by an edge. This edge corresponds to some negative edge $\{i, j\}$ in $\mathcal{P}_y$, where i is in the connected component of $\mathcal{P}_y^+[V^-]$ represented by u and j is in the connected component of $\mathcal{P}_y^+[V^-]$ represented by v. Since i and j are connected by a negative edge, it must be that $w[i] \not\approx w[j]$ which implies u and v have different colors. Hence this is a valid coloring and $\chi(\mathcal{C}_y) \leq \mu_\diamond(y)$. □

It is well known that if a graph G has e edges, then $\frac{\chi(G)(\chi(G)-1)}{2} \leq e$. We can use this fact to bound $\mu_\diamond(y)$.

Corollary 4. *Let y be the prefix array for some partial word such that $|y| = n$. Then $\mu_\diamond(y) \leq \left\lceil \sqrt{2n} \right\rceil$.*

Proof. Recall from Section 2 that $\mathcal{P}_y$ has at most $n-1$ negative edges, and since each edge of $\mathcal{C}_y$ corresponds to at least one negative edge in $\mathcal{P}_y$, we know $\mathcal{C}_y$ also has at most $n-1$ edges. Further, by Theorem 11, $\mu_\diamond(y) = \chi(\mathcal{C}_y)$. So, $\frac{\mu_\diamond(y)(\mu_\diamond(y)-1)}{2} = \frac{\chi(\mathcal{C}_y)(\chi(\mathcal{C}_y)-1)}{2} < n$. The result follows from algebraic manipulation of the above inequality, which is equivalent to $\mu_\diamond(y) - \frac{1}{2} \leq \sqrt{2n - \frac{7}{4}}$. □

We can show that this bound is tight within a constant multiple using Sidon sets. In number theory, a set $S = \{s_0, s_1, \ldots, s_{m-1}\}$ of natural numbers is a finite Sidon set, named after the Hungarian mathematician Simon Sidon, if the pairwise sums $s_i + s_j$, $i \leq j$, are all different. It is easy to show that Sidon sets have the property that the pairwise differences $|s_i - s_j|$, $i < j$, are also all different.

Proposition 3. *For every $n > 0$, there exists a prefix array y of size n such that $\mu_\diamond(y) = (1 - o(1))\sqrt{n}$.*

Proof. Erdős and Turán [8, 10] showed that, for every $n > 0$, there exists a Sidon set with $(1 - o(1))\sqrt{n}$ elements such that each element is less than n. Let $S = \{s_0, s_1, \ldots, s_{m-1}\}$ be such a set, where $m = (1 - o(1))\sqrt{n}$. Define $y = y[0..n)$ by

$$y[i] = \begin{cases} s_j & \text{if } i = s_k - s_j, \text{ for some } s_j, s_k \in S, s_j < s_k, \\ n - i & \text{otherwise.} \end{cases}$$

We show that $\mathcal{P}_y^-$ contains an m-clique. Consider $s_r, s_t \in S$, where $s_r < s_t$. This implies $y[s_t - s_r] = s_r$, and since $s_t - s_r + s_r = s_t < n$, it follows that $\{y[s_t - s_r], s_t - s_r + y[s_t - s_r]\} = \{s_r, s_t\} \in E^-$, where $\{s_r, s_t\}$ indicates an edge between the vertices indexed by s_r and s_t. Moreover, this holds for any pair of elements in S, so the vertices indexed by S form an m-clique in $\mathcal{P}_y^-$. Since by Theorem 11, $\mu_\diamond(y) = \chi(\mathcal{C}_y)$, and $\mathcal{P}_y^-$ contains an m-clique which requires at least m distinct letters for a proper coloring, we deduce $\mu_\diamond(y) \geq m$.

Now construct a partial word w on the alphabet $A = \{a_0, a_1, \ldots, a_{m-1}\}$ by $w[i] = a_j$ if $i = s_j \in S$, while $w[i] = \diamond$ otherwise. We claim that w satisfies y. Since $i + y[i] = n$ whenever $y[i] \notin S$, the only negative edges of $\mathcal{P}_y$ are of the form $\{s_r, s_t\}$ where $s_r, s_t \in S$. Note that $w[s_r] = a_r \not\approx a_t = w[s_t]$, so w satisfies all of these negative edges. Moreover, since S is an m-clique in $\mathcal{P}_y^-$, any positive edge must be incident to some vertex i where $i \notin S$. Then $w[i] = \diamond$, which matches anything, so this positive edge must be satisfied. Therefore w satisfies y on m letters, implying $\mu_\diamond(y) \leq m$. □

Referring to the proof of Proposition 3, the Sidon set $\{0, 1, 4, 6\}$ determines the prefix array $y = 7041010$. Note that $\mathcal{P}_y^-$ contains the 4-clique $\{0, 1, 4, 6\}$.

It follows from Corollary 4 and Proposition 3 that the maximum of the $\mu_\diamond(y)$'s such that y is the prefix array for a partial word of length n is $\Theta(\sqrt{n})$, while the maximum of the $\mu(y)$'s such that y is a prefix array of size n is $\Omega(\sqrt{n})$.

5.2 Enumerating all prefix arrays of a given size

We present an algorithm which enumerates all prefix arrays of size n valid for partial words. Let $\text{pref}_\diamond(n)$ be the number of such prefix arrays.

Recall from Theorem 10 that p is a valid prefix array for a partial word if and only if every negative edge in $\mathcal{P}_p$ connects vertices in two different connected components of $\mathcal{P}_p^+[V^-]$. We call this the *CC condition*.

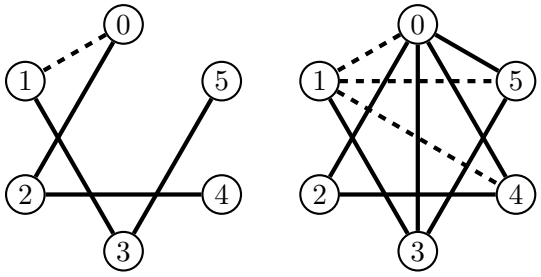

Figure 7: Spanning subgraph for array 604 (left) of prefix graph for lengthening 604111 (right).

An array p' is a *lengthening* of an array p if the first $|p|$ entries of p' and p are identical (here $|p|$ denotes the length of p). If p is an array which has some feasible lengthening y, denote by $\mathcal{P}_p$ the spanning subgraph of $\mathcal{P}_y$ containing only edges determined by the first $|p|$ elements of y. Figure 7 gives an example.

Let us first give an overview of our algorithm, Algorithm 1, that constructs all prefix arrays of size n valid for partial words by building arrays one entry at a time. If $\mathcal{P}_p$ violates the CC condition then $\mathcal{P}_{p'}$ also violates the CC condition for any lengthening p', so any partial prefix candidate is never lengthened in error by more than one position. When an array of size n satisfying the CC condition is reached, it must be a valid prefix array for some partial word, and all such arrays are eventually reached by our algorithm. Our algorithm builds a tree of height $n-1$, an example is shown in Figure 8. Some facts about the tree construction are (1) every leaf occurs at depth $n-1$ (assuming the root is at depth 0), (2) every node has either 0 or 1 as a child, and (3) the set of paths from root to leaf is in bijective correspondence with the set of prefix arrays valid for partial words. Figure 9 provides an example of how our algorithm works.

We give the pseudocode for Algorithm 1. It calls Procedure 2, which in turn calls Procedure 3. The entry $C_d[i]$ is the positive integer label (if it exists) assigned to vertex i at depth d, i.e., the label of the connected component of i at depth d in $\mathcal{P}_p^+[V^-]$, and is zero if i has not yet been labeled. The entry $F_d[i]$ gives, at depth d, the set of all positive labels j such that some vertex in the component labeled j is connected to a vertex in the component of i by a negative edge. In other words, a set of components which are forbidden from being merged with the component of i. We denote by $\mathcal{P}_p^+$ and $\mathcal{P}_p^-$ the adjacency matrices of $\mathcal{P}_p$ for its positive and negative edges, respectively. Note that the line "reset $\mathcal{P}_p^+$ and $\mathcal{P}_p^-$" in Procedure 2 means

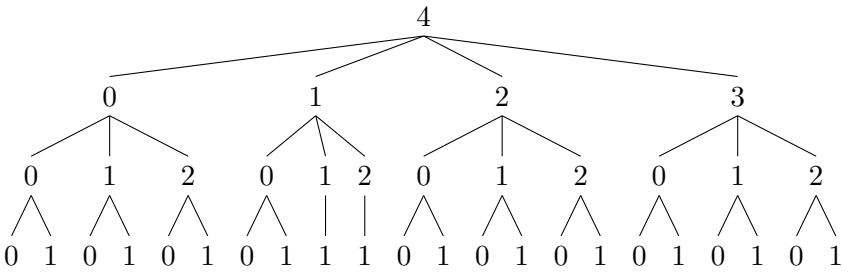

Figure 8: Tree construction for prefix arrays of size 4 valid for partial words (4110 and 4120 are not valid).

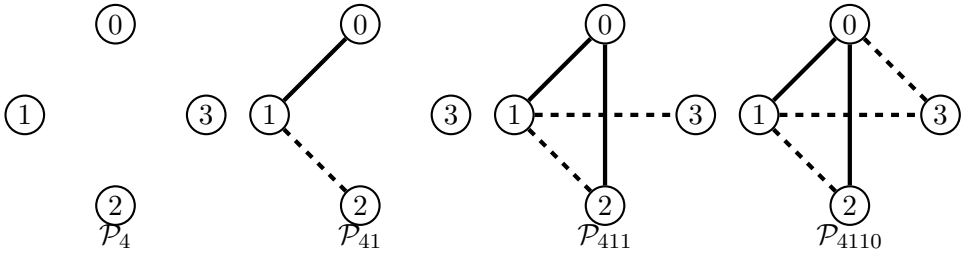

Figure 9: Constructing array 4110. Here $\mathcal{P}_4$, $\mathcal{P}_{41}$, and $\mathcal{P}_{411}$ do satisfy the CC condition, but $\mathcal{P}_{4110}$ does not.

Algorithm 1 Enumerate Prefix Arrays of Size n Valid for Partial Words

Require: integer n

1: initialize $n \times n$ arrays $\mathcal{P}_p^+$, and $\mathcal{P}_p^-$ to all zeros
2: initialize an n-element array p and an n-element collection of n-element arrays $C = \{C_d\}_{d=0}^{n-1}$ to zeros
3: initialize an n-element collection of n-element arrays of sets $F = \{F_d\}_{d=0}^{n-1}$ to empty sets
4: $p[0] \leftarrow n$
5: $d \leftarrow 0$
6: $IncreaseDepth(p, d, C, F, \mathcal{P}_p^+, \mathcal{P}_p^-)$
7: **return**

Procedure 2 IncreaseDepth

Require: p, d, C, F, $\mathcal{P}_p^+$, $\mathcal{P}_p^-$

1: $d \leftarrow d + 1$
2: **for** $i = 0$ to $n - d$ **do**
3: $\quad C_d \leftarrow C_{d-1}$
4: $\quad F_d \leftarrow F_{d-1}$
5: $\quad p[d] = i$
6: $\quad$ **if** $ValidPref(p, d, C, F, \mathcal{P}_p^+, \mathcal{P}_p^-)$ and $d < n$ **then**
7: $\quad\quad IncreaseDepth(p, d, C, F, \mathcal{P}_p^+, \mathcal{P}_p^-)$
8: $\quad$ reset $\mathcal{P}_p^+$ and $\mathcal{P}_p^-$
9: **return**

to reverse only the changes made by the most recent call to Procedure 3.

Theorem 12. *Algorithm 1 prints every prefix array of size n valid for partial words in lexicographic order exactly once.*

Proof. The algorithm uses the characterization of valid prefix arrays given in Theorem 10 and attempts to build a prefix array p one entry at a time. If there exists a lengthening of p to a valid prefix array for a partial word, then the algorithm finds it and extends p. If not, then the algorithm never again considers a prefix array starting with p.

Procedure 2 serves to reset arrays and adjacency matrices, and try possible additions to p. We say something occurs at depth d in the algorithm if the current array, p, being checked has d elements. To check the CC condition, we use C_d and F_d. It suffices to check that Procedure 3 determines whether or not p satisfies the CC condition and can be extended, then correctly updates C, F, $\mathcal{P}_p^+$, and $\mathcal{P}_p^-$ corresponding to p.

Procedure 3 ValidPref

Require: p, d, C, F, $\mathcal{P}_p^+$, $\mathcal{P}_p^-$

$i \leftarrow p[d]$
if $i < n - d$ **then**
  if $C_d[i] = C_d[i+d] > 0$ **then**
    return false
  $\mathcal{P}_p^-[i][i+d] \leftarrow 1$ and $\mathcal{P}_p^-[i+d][i] \leftarrow 1$
  if $C_d[i] = 0$ **then**
    if $C_d[j] > 0$ for some j such that $\mathcal{P}_p^+[i][j] = 1$ **then**
      $C_d[i] \leftarrow C_d[j]$
    else
      $C_d[i] \leftarrow \max\{C_d[k] \mid 0 \leq k < n\} + 1$
    if there exist $j_1 < j_2$ such that $\mathcal{P}_p^+[i][j_1] = \mathcal{P}_p^+[i][j_2] = 1$ and $0 < C_d[j_1] \neq C_d[j_2] > 0$ **then**
      if $C_d[j_2] \in F_d[C_d[j_1]]$ **then**
        return false
      else
        for k such that $C_d[k] = C_d[j_2]$ **do**
          $C_d[k] \leftarrow C_d[j_1]$
        $F_d[C_d[j_1]] \leftarrow F_d[C_d[j_1]] \cup F_d[C_d[j_2]]$
  if $C_d[i+d] = 0$ **then**
    if $C_d[j] > 0$ for some j such that $\mathcal{P}_p^+[i+d][j] = 1$ **then**
      $C_d[i+d] \leftarrow C_d[j]$
    else
      $C_d[i+d] \leftarrow \max\{C_d[k] \mid 0 \leq k < n\} + 1$
    if there exist $j_1 < j_2$ such that $\mathcal{P}_p^+[i+d][j_1] = \mathcal{P}_p^+[i+d][j_2] = 1$ and $0 < C_d[j_1] \neq C_d[j_2] > 0$ **then**
      if $C_d[j_2] \in F_d[C_d[j_1]]$ **then**
        return false
      else
        for k such that $C_d[k] = C_d[j_2]$ **do**
          $C_d[k] \leftarrow C_d[j_1]$
        $F_d[C_d[j_1]] \leftarrow F_d[C_d[j_1]] \cup F_d[C_d[j_2]]$
  $F_d[C_d[i]] \leftarrow F_d[C_d[i]] \cup F_d[C_d[i+d]]$
for $j = 0$ to $i - 1$ **do**
  if $C_d[j+d] \in F_d[j]$ **then**
    return false
  $\mathcal{P}_p^+[j][j+d] \leftarrow 1$ and $\mathcal{P}_p^+[j+d][j] \leftarrow 1$
  for k such that $C_d[k] = C_d[j+d]$ **do**
    $C_d[k] \leftarrow C_d[j]$
  $F_d[C_d[j]] \leftarrow F_d[C_d[j]] \cup F_d[C_d[j+d]]$
if $d = n$ **then**
  print p
return true

Suppose we extend our array by appending i to the end. If $i < n - d$ then the addition introduces a negative edge into $\mathcal{P}_p^-$. Lines 2–4 check that this does not connect vertices in the same labeled connected component of $\mathcal{P}_p^+[V^-]$. After updating $\mathcal{P}_p^-$ in Line 5, we must check whether i and $i + d$ have already been assigned component labels. If they have not, then we must assign labels. Lines 6–10 do this for i, and Lines 18–22 work identically for $i + d$, so we focus our attention on i. If i is connected by a positive edge to a vertex which has already been labeled, then we assign that label to i. Otherwise, assign it a brand new label. Then we check whether any two labeled components connected to it are different (Line 11). If they are forbidden from being connected then the CC condition is violated and we return false (Lines 12–13). Otherwise we merge the components and their corresponding F sets (lines 14–17).

Next we consider each positive edge and check that it satisfies the CC condition (Lines 31–37). Lines 32–33 rule out the possibility of merging components which are not allowed to be joined. Then we update $\mathcal{P}_p^+$ (Line 34) and finally merge the components connected by the positive edge (Lines 35–37).

If we are at depth n, then we have a prefix array of size n which is valid for partial words so we print it (Lines 38–39). Since p satisfies the CC condition after each lengthening, it is clear that it satisfies it once it reaches length n and thus all arrays which we print are valid prefix arrays for partial words. Moreover, the arrays are printed in increasing lexicographic order so it is clear that no array is printed more than once. □

Proposition 4. *Let p be an array such that $|p| < p[0]$ and $p[i] \leq p[0] - i$. If p satisfies the CC condition, then either $p0$ or $p1$ also satisfy the CC condition.*

Proof. Let p be an array satisfying our assumptions and set $|p| = k$. Construct the prefix graph $\mathcal{P}$ of p. There are two cases to consider. First, suppose 0 and $p[k-1]$ are both incident to negative edges, so they are both vertices in $\mathcal{P}^+[V^-]$. If they are in the same connected component, we can append a 1 to p. If they are in different connected components, we can append a 0 to p. Now, suppose either 0 or $p[k-1]$ is only incident to positive edges. This vertex does not appear in $\mathcal{P}^+[V^-]$, and the addition of a positive edge does not change this, so we may append a 1 to p without violating the CC condition. □

Construct a labeled tree T as follows: Label the root n. Assume we have the tree constructed up to depth d. Then for each vertex v at depth d, let p

be the array whose ith entry is the ith vertex in the path from the root to v. Create a child of v labeled j if pj, we append a j to p, satisfies the CC condition. The following connections can be made between T, Algorithm 1, and the set of prefix arrays of size n valid for partial words:

1. There is one node in T for each time Procedure 3 returns true. Listing the labels as they are seen for the first time in an in-order traversal of T is identical to listing, in the order that Algorithm 1 calls it, the lengthenings that Procedure 3 returns true on.

2. Every leaf in T occurs at depth $n-1$. Indeed, every node at depth d where $d < n-1$ satisfies the CC condition, so by Proposition 4 it has at least one lengthening.

3. The set of all paths from root to leaf is in bijective correspondence with the set of prefix arrays of size n valid for partial words.

Proposition 5. *Algorithm 1 runs in $O(n^3 \operatorname{pref}_\diamond(n))$ time.*

Proof. We use the correspondence between the algorithm and the tree T described above. All $\operatorname{pref}_\diamond(n)$ leaves occur at depth $n-1$, so the total number of vertices in the tree is bounded above by $n \operatorname{pref}_\diamond(n)$. The time it takes to get from one node to the next is the time it takes to run Procedure 3. Checking the CC condition after the addition of a single positive or negative edge takes $O(n)$ time. Since lengthening an array introduces at most 1 negative and $n-1$ positive edges to the prefix graph, we conclude that Procedure 3 runs in $O(n^2)$ time. Thus, the entire algorithm is $O(n^3 \operatorname{pref}_\diamond(n))$. □

Next, we use the following notion to prove a bound for $\operatorname{pref}_\diamond(n)$. If partial word w of length n uses alphabet A and partial word w' of length n uses alphabet B, then w and w' are *p-equivalent* if there exists a bijection $f : A \cup \{\diamond\} \to B \cup \{\diamond\}$ with $f(\diamond) = \diamond$, extended in the obvious way to $(A \cup \{\diamond\})^*$, such that $f(w) = w'$. If two partial words are not p-equivalent, we say they are *p-distinct*. We start with a lemma.

Lemma 4. *The number of p-distinct partial words of length n with h holes and μ, $\mu \geq 2$, letters is $\binom{n}{h}\left\{{n-h \atop \mu}\right\}$, where $\left\{{n \atop \mu}\right\}$ denotes a Stirling number of the second kind.*

Proof. We borrow from [15] and use the infinite *standard alphabet*, $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_\mu, \ldots\}$, which has subalphabets $\Lambda_\mu = \{\lambda_1, \lambda_2, \ldots, \lambda_\mu\}$ for each integer $\mu \geq 2$. The standard alphabet uses the natural ordering: $\lambda_i < \lambda_j$

for all $i < j$. In addition, we consider $\diamond < \lambda_1$, and let $\Lambda_\diamond = \Lambda \cup \{\diamond\}$. We call a partial word w *p-canonical* if $w \in \Lambda_\diamond^*$ and, for every $j \geq 2$, no λ_j appears before all of $\lambda_1, \ldots, \lambda_{j-1}$ have appeared at least once. In other words, $\lambda_1, \ldots, \lambda_{j-1}$ must occur at some positions before the first occurrence of λ_j in the word. For example, $\lambda_1\lambda_2\diamond\lambda_1\diamond\lambda_3\lambda_3$ is p-canonical, but $\lambda_1\diamond\lambda_2\lambda_1\lambda_4$ is not because λ_4 appears before λ_3 has appeared. It is important to note that every partial word is p-equivalent to exactly one p-canonical partial word.

For nonnegative integers h, μ and positive integer n, we extend a definition of [15] to define $p'[\mu, n, h]$ as the number of p-distinct partial words of length n that use exactly μ letters and have h holes. For each p-canonical regular word w of length $n - h$ that uses exactly μ, $\mu \geq 2$, letters, there are $\binom{n}{h}$ partial words resulting from inserting h holes into w. For any p-canonical regular word $w' \neq w$ with the same length and letters used, adding h holes results in partial words that are distinct from those arising out of w. Also, every p-canonical word of length n using exactly μ letters with h holes can be generated through this process. Thus, $p'[\mu, n, h] = \binom{n}{h} p'[\mu, n - h, 0]$. Moreover from [15], $p'[\mu, n, 0] = \left\{ {n \atop \mu} \right\}$, where $\left\{ {n \atop \mu} \right\}$ is a Stirling number of the second kind. Therefore, $p'[\mu, n, h] = \binom{n}{h} \left\{ {n-h \atop \mu} \right\}$. □

We now prove our bound.

Proposition 6. *For sufficiently large n,*

$$\mathrm{pref}_\diamond(n) \leq \lceil \sqrt{2n} \rceil \left\{ {n+1 \atop \lceil \sqrt{2n} \rceil + 1} \right\},$$

where $\left\{ {n+1 \atop \lceil \sqrt{2n} \rceil + 1} \right\}$ denotes a Stirling number of the second kind.

Proof. By Corollary 4, any prefix array valid for partial words can be satisfied by a partial word with at most $\left\lceil \sqrt{2n} \right\rceil$ letters. Partial words which are p-equivalent have the same prefix array, so different prefix arrays are necessarily satisfied by p-distinct partial words. We may bound the number of prefix arrays valid for partial words by the number of p-distinct partial words on at most $\left\lceil \sqrt{2n} \right\rceil$ letters. By Lemma 4, the number of p-distinct partial words of length n with h holes and μ, $\mu \geq 2$, letters is $\binom{n}{h} \left\{ {n-h \atop \mu} \right\}$. Summing over all possible numbers of holes we have $\sum_{h=0}^{n-\mu} \binom{n}{h} \left\{ {n-h \atop \mu} \right\} = \sum_{j=\mu}^{n} \binom{n}{j} \left\{ {j \atop \mu} \right\} = \left\{ {n+1 \atop \mu+1} \right\}$.

Consider p-distinct partial words using less than $\left\lceil \sqrt{2n} \right\rceil$ letters. The following fact about Stirling numbers is useful. For fixed n, $\left\{ {n \atop \mu} \right\}$ is a unimodal

sequence with mode asympototically approaching $\frac{n}{\log n}$. Thus for sufficiently large n, we have $\left\{ {n+1 \atop \lceil\sqrt{2n}\rceil+1} \right\} \geq \left\{ {n+1 \atop j} \right\}$ for all $j \leq \left\lceil \sqrt{2n} \right\rceil$. Summing over each possible number of letters and using the unimodality of Stirling numbers,

$$\mathrm{pref}_\diamond(n) \leq \sum_{\mu=2}^{\lceil\sqrt{2n}\rceil} \left\{ {n+1 \atop \mu+1} \right\} \leq \lceil\sqrt{2n}\rceil \left\{ {n+1 \atop \lceil\sqrt{2n}\rceil+1} \right\}.$$

□

6 Conclusion and Future Work

In Section 3, we demonstrated two methods for constructing an indeterminate string which satisfies a given prefix array using the smallest alphabet possible. Moreover, we showed that the minimum alphabet size for an indeterminate string satisfying a prefix array y of size n is at most $n + \sqrt{n}$. Indeed, we showed that the minimum alphabet size is at most $r + \sqrt{r} + 1$, where r is the number of negative edges in $\mathcal{P}_y$. One suggestion for future work is to improve this bound or show it is tight. Since there are many results bounding chromatic numbers, we believe the method involving $\mathcal{Q}_y$ may be useful in lowering this bound. Examining many prefix arrays has led us to the following conjecture.

Conjecture 1. *Let y be a prefix array of size n. Then $\mu(y) < n$.*

Another suggestion for future work, as mentioned in [5], is to develop an efficient algorithm to compute a string on an alphabet of minimum size for a given prefix array.

In section 5, we restricted ourselves to considering prefix arrays for partial words. We gave a characterization of prefix arrays y valid for partial words in terms of the prefix graph $\mathcal{P}_y$. Moreover, we gave a method to construct a partial word on the smallest possible alphabet for a given prefix array. We showed that the minimum alphabet size for a partial word satisfying a given prefix array of size n is at most $\left\lceil \sqrt{2n} \right\rceil$, and we showed that this bound is tight (up to a constant multiple) using Sidon sets. We also bounded $\mathrm{pref}_\diamond(n)$, the number of prefix arrays of size n valid for partial words, for large enough n, in terms of Stirling numbers of the second kind. Based on experimental data, it seems that our bound is not tight, and we believe that there is actually an exponential upper bound for $\mathrm{pref}_\diamond(n)$.

Conjecture 2. *For all n,* $\mathrm{pref}_\diamond(n) \leq 4^{n-1}$.

We gave an algorithm which enumerates all prefix arrays of size n which are valid for partial words. Our algorithm runs in $n^3 \operatorname{pref}_{\diamond}(n)$ time. In the case of regular strings, all prefix arrays of size n can be enumerated in constant time with respect to the output size [15]. So one final suggestion for future work is to develop a more efficient algorithm in the partial word case.

In addition, we established a World Wide Web server interface at

`http://www.uncg.edu/cmp/research/arrays`

for automated use of a program that produces an indeterminate string with the minimum number of letters for a given prefix array.

Acknowledgements

We thank Professor W. F. Smyth for sending us his paper on indeterminate strings, prefix arrays, and undirected graphs. We thank Brian Bowers and Nathan Fox for the slowly-increasing property, the number of slowly-increasing arrays of size n equalling the nth Catalan number, and the fact that the number of p-distinct partial words of length n with h holes and μ, $\mu \geq 2$, letters is $\binom{n}{h}\left\{{n-h \atop \mu}\right\}$, where $\left\{{n \atop \mu}\right\}$ denotes a Stirling number of the second kind. We also thank the referees of a preliminary version of this paper for their very valuable comments and suggestions.

References

[1] K. Abrahamson. Generalized string matching. *SIAM Journal on Computing*, 16:1039–1051, 1987.

[2] N. Alon. Covering graphs by the minimum number of equivalence relations. *Combinatorica*, 6:201–206, 1986.

[3] F. Blanchet-Sadri. *Algorithmic Combinatorics on Partial Words*. Chapman & Hall/CRC Press, Boca Raton, FL, 2008.

[4] F. Blanchet-Sadri, M. Bodnar, and B. De Winkle. New bounds and extended relations between prefix arrays, border arrays, undirected graphs, and indeterminate strings. In E. Mayr and N. Portier, editors, *STACS 2014, 31st International Symposium on Theoretical Aspects of Computer Science, Lyon, France*, volume 25 of *LIPIcs Schloss Dagstuhl-Leibniz-Zentrum für Informatik*, pages 162–173, Germany, 2014. Schloss Dagstuhl.

[5] M. Christodoulakis, P. J. Ryan, W. F. Smyth, and S. Wang. Indeterminate strings, prefix arrays & undirected graphs. arXiv:1406.3289v1 [cs.DM] 12 Jun 2014.

[6] J. Clément, M. Crochemore, and G. Rindone. Reverse engineering prefix tables. In S. Albers and J.-Y. Marion, editors, *STACS 2009, 26th International Symposium on Theoretical Aspects of Computer Science, Freiburg, Germany*, volume 3 of *LIPIcs*, pages 289–300. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.

[7] M. Crochemore, C. Hancart, and T. Lecroq. *Algorithms on Strings*. Cambridge University Press, 2007.

[8] P. Erdős. On a problem of Sidon in additive number theory and on some related problems. Addendum. *Journal of the London Mathematical Society, Second Series*, 19:208, 1944.

[9] P. Erdős, A. W. Goodman, and L. Pósa. The representation of a graph by set intersections. *Canadian Journal of Mathematics*, 18:106–112, 1966.

[10] P. Erdős and P. Turán. On a problem of Sidon in additive number theory, and on some related problems. *Journal of the London Mathematical Society, Second Series*, 16:212–215, 1941.

[11] M. Fischer and M. Paterson. String matching and other products. In R. Karp, editor, *7th SIAM-AMS Complexity of Computation*, pages 113–125, 1974.

[12] J. L. Gross and J. Yellen. *Handbook of Graph Theory*. CRC Press, 2004.

[13] L. Lovász. On covering of graphs. In *Theory of Graphs (Proceedings of the Colloquium, Tihany, 1966)*, pages 231–236. Academic Press, New York, 1968.

[14] M. G. Main and R. J. Lorentz. An O(nlog n) algorithm for finding all repetitions in a string. *Journal of Algorithms*, 5:422–432, 1984.

[15] D. Moore, W. F. Smyth, and D. Miller. Counting distinct strings. *Algorithmica*, 23:1–13, 1999.

[16] W. F. Smyth. *Computing Patterns in Strings*. Pearson Addison-Wesley, 2003.

[17] W. F. Smyth and S. Wang. New perspectives on the prefix array. In *15th Symposium on String Processing and Information Retrieval*, volume 5280 of *Lecture Notes in Computer Science*, pages 133–143. Springer-Verlag, Berlin, Heidelberg, 2008.