Adaptive Data Structures for Permutations and Binary Relations*

Francisco Claude[1,2,3] and J. Ian Munro[3]

[1] Akori S.A.
Santiago, Chile
[2] Escuela de Informática y Telecomunicaciones
Universidad Diego Portales, Chile
[3] David R. Cheriton School of Computer Science
University of Waterloo, Canada

Abstract. We present new data structures for representing binary relations in an adaptive way, that is, for certain classes of inputs we achieve space below the general information theoretic lower bound, while achieving reasonable space complexities in the worst case. Our approach is derived from a geometric data structure [Arroyuelo et al., TCS 2011]. When used for representing permutations, it converges to a previously known adaptive representation [Barbay and Navarro, STACS 2009]. However, this new way of approaching the problem shows that we can support range searching in the adaptive representation. We extend this approach to representing binary relations, where no other adaptive representations using this chain decomposition have been proposed.

1 Introduction

Binary relations and permutations arise in many applications in computer science. Examples include text indexing [12] and graph representations [8], among others. These fundamental objects have been heavily studied [11,4,5,6], and very efficient data structures supporting a wide range of operations have emerged. However, most of them remain bounded by the information theoretic lower bound in their space consumption, even in the cases where the objects have exploitable properties; for example Web Graphs [7]. Some exceptions are all the developments for compressed suffix arrays [12] and the work by Barbay and Navarro [6] on more general permutations.

In this paper, we present space-efficient data structures that have adaptive space and time complexities. Our approach comes from a geometric perspective, and for permutations, converges to the representation by Barbay and Navarro [6]. However, our new approach brings a new perspective, showing how to support range searching operations. We show that the work of [6] serves to represent binary relations with Theorem 1, and also prove an alternative tradeoff based on our own formulation of their structure.

* First author funded in part by Google U.S./Canada PhD Fellowship. Second author funded in part by NSERC and the Canada Research Chairs Programme.

O. Kurland, M. Lewenstein, and E. Porat (Eds.): SPIRE 2013, LNCS 8214, pp. 64–71, 2013.

The paper is organized as follows. In Section 2 we present related work on representing permutations and binary relations, we also include some background on data structure for range searching proposed by Arroyuelo et al. [1]. In Section 3 we present our adaptive representation for permutations, and show how this representation converges to the one by Barbay and Navarro. Next, in Section 4, we extend the representation for permutations to cover binary relations in general, presenting two different approaches. Then, in Section 4.2, we present one simple application of our structure. Finally, in Section 5, we present our conclusions.

2 Related Work

We present the related work in the next two subsections. The first one presents previous results on representing permutations and binary relations; the second covers recent results on adaptive range searching.

2.1 Permutation and Binary Relations

The most common queries for a permutation Π over $[n]$[1] are: (1) $\pi(i)$: obtain the value of $\Pi[i]$; (2) $\pi^{-1}(j)$: find i such that $j = \Pi[i]$; (3) $\pi^k(i)$: apply π k times, similarly we define $\pi^{-k}(j)$; and (4) $\mathcal{R}_\Pi(i_1, i_2, j_1, j_2)$: find elements i such that $i_1 \leq i \leq i_2$ and $j_1 \leq \pi(i) \leq j_2$.

One efficient representation for arbitrary permutations is that of Munro et al. [11]. This representation achieves $(1 + \epsilon)n \lg n(1 + o(1))$ bits. It supports π in $O(1)$ time and π^{-1} in $O(\frac{1}{\epsilon})$ time. They also showed that $\pi^{\pm k}$ can be supported in the same time as the time required to perform both π and π^{-1} by using only $O(n)$ extra bits. This extension applies to any representation, and thus, to our results too.

The $\mathcal{R}$ operation is less commonly required, but also of interest. For instance, consider a position-restricted search using a suffix array. The suffix array is a permutation and searching for a pattern P between positions p_1 and p_2 is just the result of doing a range query over the range of suffixes starting with P (i.e., $[i_1, i_2]$) and those pointing to positions in $[p_1, p_2]$. Mäkinen and Navarro showed how to use wavelet trees to solve this operation and all others in $O(\lg n)$ time within $n \lg n(1 + o(1))$ bits of space [10].

Prior to this paper the only adaptive representation for permutations was that proposed by Barbay and Navarro [6]. They show many possible decompositions into monotonic sequences and subsequences, and give their space/time complexities in term of the entropy of such sequences. As we will see later, we converge to the same structure at the end of Section 3.

A natural representation for a permutation is a binary matrix of $n \times n$ where we mark the coordinates (i, j) with a 1 iff $\pi(i) = j$. We will use this conceptual representation in our construction. For a binary relation $\mathcal{B}$ over two sets $[n_1]$

[1] We use $[n]$ to represent $\{1, 2, \ldots, n\}$.

Table 1. Operations and implementations supported by the representation of Barbay et al. [4,5]. The space requirement is $t(\lg n_2 + o(\lg n_2))$ bits.

Operation	Implementation 1	Implementation 2
$\text{rowrank}_{\mathcal{B}}(i,j)$: number of 1s in row i up to position j (included).	$O(\lg\lg n_2 \lg\lg\lg n_2)$	$O(\lg\lg n_2)$
$\text{rowselect}_{\mathcal{B}}(i,p)$: p-th 1 in row i, or ∞ if $\text{rowrank}_{\mathcal{B}}(i,n_2) < p$.	$O(\lg\lg n_2)$	$O(1)$
$\text{rowcount}_{\mathcal{B}}(i)$: number of 1s in row i.	$O(1)$	$O(1)$
$\text{colrank}_{\mathcal{B}}(i,j)$: number of 1s in column j up to position i (included).	$O(\lg\lg n_2)$	$O(\lg\lg n_2 \lg\lg\lg n_2)$
$\text{colselect}_{\mathcal{B}}(p,j)$: p-th 1 in column j, or ∞ if $\text{colrank}_{\mathcal{B}}(n_1,j) < p$.	$O(1)$	$O(\lg\lg n_2)$
$\text{colcount}_{\mathcal{B}}(j)$: number of 1s in column j.	$O(1)$	$O(1)$
$\text{relaccess}_{\mathcal{B}}(i,j)$: true iff $(i,j) \in \mathcal{B}$.	$O(\lg\lg n_2)$	$O(\lg\lg n_2)$

and $[n_2], n_2 \leq n_1$, with $t = |\mathcal{B}|$ elements in the relation, we can also use the same conceptual representation. $\mathcal{B}$ is represented as a matrix of n_1 rows by n_2 columns with t ones. A one in position (i,j) indicates that i relates to j in $\mathcal{B}$.

Barbay et al. [4,5] presented a structure for representing binary relations that requires $t(\lg n_2 + o(\lg n_2))$ bits of space and supports the operations, offering two tradeoffs, as shown in Table 1.

In a follow-up work, Barbay et al. [7] proved a set of reductions for many operations on binary relations, and presented two structures supporting a core of operations allowing to answer efficiently this extended set. An important operation that allows us to support many of the operations in the extended set is $\text{relrange}_{\mathcal{B}}$. The operation $\text{relrange}_{\mathcal{B}}$ takes a range $[i_1,i_2] \times [j_1,j_2]$ and returns all the coordinates in that range containing a one. From the structures proposed in Barbay et al.'s work [7], the first structure achieves $t \lg n_2 + o(t \lg n_2)$ bits, slightly different from the previous proposal in the lower order term. The second structure achieves $\lg(1+\sqrt{2})t\mathcal{H}(\mathcal{B}) + o(t\mathcal{H}(\mathcal{B}))$ bits, where $\mathcal{H}$ corresponds to the general information theoretic lower bound, supporting most operations of interest in $O(\lg n_2)$ time per element retrieved. In this case $\mathcal{H}(\mathcal{B}) = t \lg \frac{n_1 n_2}{t}$ corresponds to the information theoretic lower bound for representing a binary relation with the characteristics of $\mathcal{B}$.

2.2 Monotonic Decomposition of Sequences

Arroyuelo et al. [1] presented an adaptive data structure for range searching that decomposes the set of points into non-crossing ascending and descending chains. Let k be the number of chains generated by the decomposition, the search time for a range query is $O(\lg k \lg n + k' + \text{output})$ time, where k' corresponds to the number of chains intersecting the query rectangle and output is the number of points in the answer. The main idea behind the search strategy is to first search for a chain that crosses the query rectangle (or discard all of them). Since the

chains do not cross, we can binary search the chains, at $O(\lg n)$ cost each probe. Once a chain is found, we have to traverse neighbouring chains until leaving the rectangle in order to retrieve all points.

The decomposition into non-crossing chains can be computed in polynomial time if we are given an optimal decomposition into monotonic subsequences [1]. The optimal decomposition into monotonic subsequences is NP-Hard [15], yet it is interesting that the optimal decomposition for a permutation of length n is bounded by $c\sqrt{n}$, where $c \leq 2$, and that we can get a constant factor approximation in polynomial time [16]. In this work we consider the optimal decomposition and show how this allows for a representation that is adaptive in the number of monotonic subsequences into which a permutation or binary relation can be decomposed. The results as stated apply also for the case when we compute a constant factor approximation, thus making the data structure feasible in practice.

3 Representing Permutations

Our representation works by decomposing the permutation into ascending and descending subsequences. A simple way to visualize this is to consider the representation of the permutation in a grid. Every row represents the index i, the columns represent the value of $\Pi[i]$. It is easy to see that the inverse permutation corresponds just to the transposed matrix. In order to simplify the presentation of this work, we will only consider ascending subsequences, the results extend easily to the general case.

First, we show how to represent a chain using bitmaps that support rank, select and access operations.

Definition 1. *A chain* $[(x_1, y_1), (x_2, y_2) \ldots, (x_n, y_n)]$ *is ascending iff* $x_i \leq x_{i+1}$, $1 \leq i < n$ *and* $y_i \leq y_{i+1},\ 1 \leq i < n$.

From this definition it is easy to prove our first result, stated in the following lemma. In order to present this result in a general way we use $S(n, m)$ as the space requirement (in bits) for representing a bitmap of length n with m ones that supports rank in t_r, select in t_s, and access in t_a time. We use t_b as $max(t_r, t_s, t_a)$.

Lemma 1. *Given an ascending chain* $\mathcal{C} = [(i_1, j_1), (i_2, j_2), \ldots, (i_m, j_m)]$, *of length* m, *where the values do not exceed* n, *we can represent the chain in* $2S(n, m)$ *bits and support the following queries:*

- $\text{getj}_{\mathcal{C}}(i)$*: gets* j *such that* $(i, j) \in \mathcal{C}$ *or* $\perp$ *if such pair does not exist. We also define* $\text{geti}_{\mathcal{C}}(j)$ *in an analogous way. Both queries are supported in time* $O(t_b)$.
- $\text{range}_{\mathcal{C}}(i_1, i_2, j_1, j_2)$*: find the* $(i, j) \in \mathcal{C}$ *such that* $(i, j) \in [i_1, i_2] \times [j_1, j_2]$ *or* $\perp$ *if such a point does not exist. This runs in time* $O(t_b)$.

We can represent each chain using Lemma 1, this leads to the following theorem:

Lemma 2. *Let m_i be the number of elements in chain i. The total space of the structure for a permutation that can be decomposed into χ chains is $2\sum_{i=1}^{\chi} S(n, m_i)$, and supports range queries in $O(t_b \lg \chi + t_b \chi' + output)$, where χ' is the number of chains that intersect the range. The next table summarizes some of the tradeoffs we can achieve.*

Bitmap Representation	*Total Space*	t_b
Pătraşcu [14]	$2n \lg \chi + O\left(\chi \lg n + \frac{\chi n}{\lg^c n}\right)$	$O(c)$
Okanohara and Sadakane [13]	$2n \lg \chi + O(\chi \lg n + n)$	$O\left(\lg \frac{n}{m_i} + \frac{\lg^4 m_i}{\lg n}\right)$

The complexity for range queries follows from the work by Arroyuelo et al. [1]. The computation of the final space is similar to the one used in proof of Theorem 2.

This representation of course is only useful for very small values of χ, otherwise the structure can be asymptotically bigger than the information theoretic minimum. This can be improved by the following observation.

Observation 1. *Given a set of χ bitmaps of length n, where the total number of ones in the set is n and no two bitmaps contain a 1 in the same position, we can represent them as a sequence of length n over an alphabet of size χ. Furthermore, any sequence representation supporting rank, select, and access in times t_r, t_s, and t_a, allows us to support the same operations in each individual bitmap within the same time.*

This not only allows us to lower the space, but it also simplifies the π and π^{-1} queries. To know which chain contains the value associated with $\pi(i)$ we just need to know which bitmap is referred to in the sequence representation by accessing its position.

It is interesting that we can represent our structure using two sequences (x and y coordinates), and they correspond exactly to S_π and $S_{\pi^{-1}}$. Furthermore, they also correspond to the representation proposed by Barbay and Navarro [6], which was originally proposed using wavelet trees, but can be modified to work with any representation, offering a wider set of tradeoffs [2]. Currently, the most interesting tradeoff is that of of Barbay at el. [3,2].

Another point to highlight, is that this shows that the original structure of Barbay and Navarro also supports adaptive range searching. This particular searching algorithm has proven to be efficient in practice [9]. This allows to state the following corollary.

Corollary 1. *Given (i) a permutation Π, that can be decomposed into χ monotonically ascending and descending chains, and (ii) a sequence representation that requires $S(n, \sigma)$ for representing a sequence of length n over an alphabet of size σ supporting rank, select and access queries in $O(t_b)$ time, there exists a structure requiring $2S(n, \chi)$ bits that supports computing π and π^{-1} in $O(t_b)$ and range search queries in $O(t_b \lg \chi + t_b \chi' + output)$, where χ' is the number of chains that touch the query rectangle, and output the size of the output.*

4 Representing Binary Relations

We use the same approach on the grid representing the binary relation. Given a binary relation $\mathcal{B}$, the pair (i, j) is marked iff i relates to j in $\mathcal{B}$. We follow the notation of the previous sections. Recall that σ is the number of rows, n the number of columns, and t the number of pairs in $\mathcal{B}$.

We assume all columns and rows have at least one element, as we can trivially map the problem when we accept empty row/columns adding a bitmap of length $n + \sigma$ supporting rank, select, and access.

We focus mostly on three operations: (1) Iterating over $\text{rowselect}_{\mathcal{B}}(i, p = 1 \ldots \text{rowcount}_{\mathcal{B}}(i))$; (2) Iterating over $\text{colselect}_{\mathcal{B}}(i, p = 1 \ldots \text{colcount}_{\mathcal{B}}(i))$; and (3) Obtaining all pairs in $[i_1, i_2] \times [j_1, j_2]$ (i.e., $\text{relrange}_{\mathcal{B}}$).

The technique presented in Section 3 does not apply directly to this case. The main problem is that a chain could contain many elements that are in the same row or column, and would result in multiple chains in the same position in a bitmap. We first give a representation that matches the result for the permutations in time and space and then we show how to potentially improve the space by using a more elegant technique. This new approach has a worse query time.

4.1 Using Permutations

We show how to transform a binary relation into a permutation by just considering a simple row/column addition algorithm that moves points around and allows one to answer the queries of interest.

The main idea is to create multiples copies of rows and columns having more than one point and then distribute the points across them so that each of them has only one point, leading to a permutation on t elements. This is inefficient in terms of space, but it allows us to match the performance of our structure for permutations. In order to be able to extract the original information we need to add $2t+o(t)$ bits, stored in B_1 and B_2. These bitmaps tell the length, in unary, of each expanded row/column. Using these two bitmaps, we can answer $\text{rowcount} = X_a.\texttt{select}(1, x+1) - X_a.\texttt{select}(1, x) - 1$ and $\text{colcount} = X_b.\texttt{select}(1, y+1) - X_b.\texttt{select}(1, y) - 1$ in constant time. We omit the algorithm pseudo-code for lack of space. We also omit the proof that our procedure generates a permutation.

This yields a theorem similar to that of Barbay et al. [4], but supporting a different subset of operations.

Theorem 1. *A binary relation $\mathcal{B} \subseteq \{(i, j) | i \in [n_1], j \in [n_2]\}$, where $t = |\mathcal{B}|$, that can be decomposed into k monotonic chains, can be represented as a permutation of length t with $(n_1 + n_2)(1 + o(1))$ extra bits. Furthermore, the resulting permutation can be decomposed into k monotonic chains, and the operations* rowselect, colselect *and* relrange *can be mapped to π, π^{-1} and $\mathcal{R}$ operations on the permutation, respectively. The counting operations can be solved using bitmaps B_1 and B_2.*

4.2 Using Chains Directly

An alternative method can be obtained by decomposing the binary relation directly. A chain could now contain more than one occurrence of a given row or column, and because of that, the transformation that converges to the structure by Barbay and Navarro does not work. At this point, the departure from the original proposal by Barbay and Navarro pays off, allowing the representation of a class wider than that of permutations.

We first take a look at the space consumption of our structure when decomposing $\mathcal{B}$ into chains. For that, we present an alternative representation for the chains. For simplicity, we will use the bitmaps representation by Pătraşcu [14], yet the results translate in a similar way as for Lemma 2, and thus, we can offer a wide set of bounds.

Lemma 3. *An ascending chain of length m with at most $\bar{n} = n + \sigma$ points in $[n] \times [\sigma]$ can be represented in $2m \lg \frac{\bar{n}}{m} + O\left(m + \frac{\bar{n}}{\lg^c n}\right)$ bits of space, but now* geti, getj *and* range *take $O(c \lg m)$ time.*

If we represent the structure using these chains, we obtain the following theorem:

Theorem 2. *A binary relation $\mathcal{B}$ over $[n_1] \times [n_2]$, where $t = |\mathcal{B}|$, can be represented in $2t \lg \frac{nk}{t} + 2t \lg k + O(\frac{kn}{\lg^c n} + k \lg t)$ bits, where $n = \max(n_1, n_2)$. Within this space, we can list elements in $O(r)$ time per datum retrieved, and answer range queries in $O(r(\lg k + k') + output)$ time, where k is the number of chains, k' the number of chains hitting the query rectangle, output the size of the output, and $r = \max(c \lg k, c \lg \lg n)$.*

We could try merging the sequences marking with a bitmap where each position starts, and this would lead to the result obtained in Theorem 1.

Another observation regarding the representation presented in Theorem 2 is that we can answer range minimum queries (RMQs) over the binary relation in the same time as for relrange.

Lemma 4. *By adding $O(t)$ extra bits to the representation from Lemma 2, or Theorems 1 and 2, and adding weights to each pair in the permutation/relation, we can support range minimum queries in the same complexity as the one required for answering* relaccess.

5 Conclusions and Future Work

We presented an alternative formulation for the representation by Barbay and Navarro. This new approach allows to show how to support range searching and provides some different tradeoffs. We then extended the results to the case of binary relations. We proposed two alternatives, both achieving interesting tradeoffs supporting navigation and range searching. It is worth noting that for *easy instances* we obtain smaller and faster representations, which is clearly an interesting behaviour.

References

1. Arroyuelo, D., Claude, F., Dorrigiv, R., Durocher, S., He, M., López-Ortiz, A., Munro, J.I., Nicholson, P.K., Salinger, A., Skala, M.: Untangled monotonic chains and adaptive range search. TCS 412(32), 4200–4211 (2011)
2. Barbay, J., Claude, F., Gagie, T., Navarro, G., Nekrich, Y.: Efficient fully-compressed sequence representations. Algorithmica (to appear, 2013)
3. Barbay, J., Gagie, T., Navarro, G., Nekrich, Y.: Alphabet partitioning for compressed rank/select and applications. In: Cheong, O., Chwa, K.-Y., Park, K. (eds.) ISAAC 2010, Part II. LNCS, vol. 6507, pp. 315–326. Springer, Heidelberg (2010)
4. Barbay, J., Golynski, A., Munro, J.I., Rao, S.S.: Adaptive searching in succinctly encoded binary relations and tree-structured documents. TCS 387(3), 284–297 (2007)
5. Barbay, J., He, M., Munro, J.I., Rao, S.S.: Succinct indexes for strings, binary relations and multi-labeled trees. In: SODA, pp. 680–689 (2007)
6. Barbay, J., Navarro, G.: Compressed representations of permutations, and applications. In: STACS, pp. 111–122 (2009)
7. Barbay, J., Claude, F., Navarro, G.: Compact rich-functional binary relation representations. In: López-Ortiz, A. (ed.) LATIN 2010. LNCS, vol. 6034, pp. 170–183. Springer, Heidelberg (2010)
8. Claude, F., Navarro, G.: Fast and compact Web graph representations. TWEB 4(4), article 16 (2010)
9. Claude, F., Munro, J.I., Nicholson, P.K.: Range queries over untangled chains. In: Chavez, E., Lonardi, S. (eds.) SPIRE 2010. LNCS, vol. 6393, pp. 82–93. Springer, Heidelberg (2010)
10. Mäkinen, V., Navarro, G.: Rank and select revisited and extended. TCS 387, 332–347 (2007)
11. Munro, J.I., Raman, R., Raman, V., Rao, S.S.: Succinct representations of permutations. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 345–356. Springer, Heidelberg (2003)
12. Navarro, G., Mäkinen, V.: Compressed full-text indexes. ACM Computing Surveys 39(1), article 2 (2007)
13. Okanohara, D., Sadakane, K.: Practical entropy-compressed rank/select dictionary. In: ALENEX (2007)
14. Pătraşcu, M.: Succincter. In: FOCS, pp. 305–313 (2008)
15. Wagner, K.: Monotonic coverings of finite sets. Elektron. Informationsverarb. Kybernet. 20, 633–639 (1984)
16. Yang, B., Chen, J., Lu, E., Zheng, S.Q.: A comparative study of efficient algorithms for partitioning a sequence into monotone subsequences. In: Cai, J.-Y., Cooper, S.B., Zhu, H. (eds.) TAMC 2007. LNCS, vol. 4484, pp. 46–57. Springer, Heidelberg (2007)