Encoding Data Structures for Range Queries on Arrays

Seungbum Jo ✉ 🆔
Chungnam National University, Daejeon, South Korea

Srinivasa Rao Satti ✉ 🆔
Norwegian University of Science and Technology, Trondheim, Norway

── **Abstract** ──────────────────────────────────

Efficiently processing range queries on arrays is a fundamental problem in computer science, with applications spanning diverse domains such as database management, computational biology, and geographic information systems. A range query retrieves information about a specific segment of an array, such as the sum, minimum, maximum, or median of elements within a given range. The challenge lies in designing data structures that allow such queries to be answered quickly, often in constant or logarithmic time, while keeping space overhead (and preprocessing time) small. Encoding data structures for range queries has emerged as a pivotal area of research due to the increasing demand for high-performance systems handling massive datasets. These structures consider the data together with the queries and aim to store only as much information about the data as is needed to answer the queries. The data structure does not need to access the original data to answer the queries. Encoding-based solutions often leverage techniques from succinct data structures, bit manipulation, and combinatorial optimization to achieve both space and time efficiency. By encoding the array in a manner that preserves critical information, these methods strike a balance between query time and space usage. In this survey article, we explore the landscape of encoding data structures for range queries on arrays, providing a comprehensive overview of some important results on space-efficient encodings for various types of range query.

1 Introduction

Efficiently processing range queries on arrays is a fundamental problem in computer science, with applications spanning diverse domains such as database management, computational biology, geographic information systems, and data analytics. A range query retrieves specific information about a contiguous segment of an array, such as the sum, minimum, maximum, or median of elements within a given range. Designing data structures to process such queries efficiently is critical, especially as datasets grow in size and complexity. The challenge lies in enabling fast query responses – often aiming for constant or logarithmic time – while minimizing the space overhead and preprocessing time required by the data structures.

Encoding data structures for range queries have emerged as a crucial area of research to address these challenges. Unlike traditional approaches that rely on augmenting the array with auxiliary structures or preprocessing, encoding-based methods focus on storing a compact representation of the data tailored specifically to the types of queries to be answered. These structures are designed to store only the information necessary for query resolution, eliminating the need to access the original array during query processing. The encoding data structures typically combine techniques from succinct/compressed data structures, combinatorial optimization and algorithmic design.

This survey aims to provide a comprehensive and structured overview of the key developments in the field of encoding data structures for range queries on arrays. We categorize the encoding solutions based on the specific types of range queries they support, such as range minimum/maximum queries, range top-k queries, range mode queries, and range majority/minority queries. The survey article by Skala [51] from 2013 gives a detailed summary of various results for "range queries on arrays". Thus we mainly focus on the new results since 2013, briefly summarizing the previous results.

Data structures can be classified into two categories: *indexing* and *encoding* data structures. For an indexing data structure, we preprocess the data and build an *index* so that subsequent queries can be answered efficiently by probing the index and the input data. On the other hand, for an encoding data structure, we preprocess the input data and build an *encoding* of the input so that subsequent queries can be answered by probing only the encoding (i.e., with no access to the input at query time). In this survey, we mainly focus on the results on encoding data structures, but occasionally mention a few results on indexing data structures for comparison. For many problems, the size of an encoding can be smaller than the size of the input, which is in fact the case for many range queries that we consider.

For example, given a query range, a range minimum query returns the minimum element within the query range. Range median, range mode, range selection and top-k, range mode and majority/minority queries are defined analogously. With this definition of range queries, one can reconstruct the input array by asking point queries with range that contains a single element, and hence the size of an encoding is at least the size of the input, and storing the input explicitly gives an optimal space encoding. To avoid this, we define the range queries as follows. For a query range, the range minimum query returns a position of the minimum element within the query range (and analogously for other range queries), instead of the value at the position. With this definition, one can obtain an encoding of size linear in bits for range minimum queries, which is asymptotically less than the space required to store the input array in the word RAM model.

In this article, we consider the encoding data structures for a 1D array $A[1, n]$, and a 2D array $A[1, m][1, n]$, where $m \leq n$. We assume that the array indices start from 1. For 2D arrays, we use the term *range* to mean an rectangular range which is defined as a Cartesian product of a given set of intervals in each dimension. We assume a word RAM model with word-size $\Theta(\log n)$ bits. For the encoding structures where we do not mention the query times, queries can be supported in polynomial time by essentially decoding the entire structure.

2 Range minimum queries

Given an input array and a query range, a range minimum query (RMQ) returns the position of the minimum element within the query range. From its wide applications (e.g., constructing an indexing structure on a string [15]), designing a data structure to answer RMQ has intensive attention from the community. For the 1D case, any encoding for RMQ requires at least $2n - o(n)$ bits due to its bijective relationship with the Cartesian tree of the input [53]. The best-known result for a worst-case input is the $(2n + o(n))$-bit data structure by Fisher and Heun that supports $O(1)$ query time[1]. For earlier results, see [51].

When the input is highly compressible, there are some results whose space usages are parameterized based on the compressed size of the input while still supporting efficient query time. Here, compressibility is considered in two ways: (1) The compressibility of the

---

[1] considering the $o(n)$-term, the current best result is Navarro and Sadakane's $(2n + O(n/(\frac{\log n}{t})^t))$-bit data structure that supports queries in $O(t)$ time [45].

input [2, 18, 21, 49], and (2) the compressibility of the Cartesian tree of the input [21, 43]. Note that data structures in (1) can access the input, as they maintain the input array in a compressed form. In contrast, the results in (2) cannot access to the input as they maintain the compressed Cartesian tree with some auxiliary structures for the query support.

There also has been progress on space-query time trade-off lower bounds for the problem under the cell-probe model, which measures query time by counting the number of memory cell accesses only. Liu and Yu showed that any data structure answering RMQ in $O(t)$ time requires at least $2n + O(n/(\log n)^{O(t^2 \log^2 t)})$ bits of space [42]. Later, Liu improved the space lower bound to $2n + O(n/(\log n)^{O(t \log^2 t)})$ bits [41].

2.1 Range minimum and maximum queries

To address range minimum and maximum queries on 1D array simultaneously, a straightforward approach is to use separate data structures for each query. Since any data structure designed for range minimum queries can be easily adapted for range maximum queries, the data structure of Fischer and Heun [17] provide a $(4n + o(n))$-bit data structure that can answer both queries in $O(1)$ time. Gawrychowski and Nicolson [26] showed that if an array contains no consecutive equal values, there exists a data structure that uses $3n + o(n)$ bits and supports both queries in $O(1)$ time. Furthermore, they proved that any encoding data structure for answering both queries requires at least $3n - \Theta(\log n)$ bits, as any Baxter permutation can be fully reconstructed using only range minimum and maximum queries.

When the input array contains consecutive equal values, a straightforward solution is to use two separate data structures proposed by Fischer [14] for range minimum and maximum queries. This approach uses $5.08n + o(n)$ bits of space and supports both queries in $O(1)$ time. Additionally, for any given p, the data structure also can answer the position of the p-th leftmost minimum or maximum value within a query range in $O(1)$ time [35]. Jo and Satti [35] improved the space usage of the data structure to $4.585n + o(n)$ bits while maintaining the same $O(1)$ query time. Subsequently, Tsur [52] further reduced the space to $3.701n$ bits with $O(n)$ time for queries. Recently, Jo and Kim [33] proposed a data structure that uses $3.701n + o(n)$ bits while supporting all queries in $O(\log^{(\ell)} n)$ time for any positive integer ℓ (here $\log^{(\ell)}$ denotes logarithm iterated ℓ times for any constant $\ell \geq 1$). They also showed that any data structure for answering these queries requires at least $3.16n - \Theta(\log n)$ bits when the input contains consecutive equal values.

2.2 Range minimum queries on non-permutation input

Consider a 1D array with duplicate entries. In this case, some ranges may have multiple answers for RMQ. The data structures discussed in Skala's survey [51] return either the leftmost or rightmost position among the possible answers. Motivated by the efficient construction of compressed suffix trees [50], Fischer and Heun [16] considered the problem of finding the position of an approximated median among all positions of minimums within a given range. Specifically, for any constant $1 < c < 1/2$, they proposed a $(\log (3 + 2\sqrt{2})n + o(n/c) \approx 2.54n + o(n/c))$-bit data structure that can answer the position of the r-th leftmost minimum in $O(1/c)$ time, where $r \in [\frac{1}{2}(1/2 - c), \frac{1}{2}(1/2 + c)]$. The data structure uses a super Cartesian tree, a Cartesian tree in which each edge is colored either red or blue.

2.3 Range minimum queries on 2D array

When the input is an $m \times n$ 2D array with $m \leq n$, Demaine et al. [11] showed that there is no Cartesian tree-like structure for the 2D case. Specifically, no structure exists that fully encodes the answer to all queries and can be constructed in linear time. They further

showed that when $m = n$, any encoding data structure for answering RMQ queries requires $\Omega(n^2 \log n)$ bits. Since the input array can be stored using $O(n^2 \log n)$ bits, this implies that any encoding data structure uses asymptotically the same space as indexing data structures [1, 5, 54] when $m = \Theta(n)$. In this survey, we focus on the case where $m = o(n)$.

Brodal et al. [5] proposed an $O(nm \cdot \min(m, \log n))$-bit data structure with $O(1)$ query time. The $O(nm^2)$-bit data structure is achieved by maintaining $O(m^2)$ 1D RMQ structures to support queries over the range $[i, j] \times [1, n]$ for all $1 \leq i \leq j \leq m$ along with the 1D RMQ structures on the columns. Additionally, they proved that any encoding for answering RMQ requires at least $\Omega(nm \log m)$ bits. When m is 2 or 3, Golin et al. [29] improved the encoding space of the result in [5] (see Table 1) by introducing the joint Cartesian tree of the input. This structure is used to compare two minimum elements in ranges where the row ranges are $[1, m-1]$ and m, respectively. Furthermore, when $m = 2$, they proposed a data structure that answers queries in $O(t)$ time using $5n + O(n \log t/t)$ bits of space, for any $t = (\log n)^{O(1)}$.

Table 1 Encoding space for RMQ queries on $m \times n$ 2D array with $m = 2$ or 3.

Input	Space (in bits)	ref
$2 \times n$	$7n - O(\log n)$	[5]
	$5n - O(\log n)$	[29]
$3 \times n$	$(12 + \log 5)n - O(\log n)$	[5]
	$(6 + \log 5)n + o(n)$	[29]

Also for the 2D array case with $1 \leq i \leq m$ and $1 \leq j \leq n$, the query range can be restricted as follows: (1) 1-sided: $[1, m] \times [1, j]$, (2) 2-sided: $[1, i] \times [1, j]$, (3) 3-sided: $[1, i] \times [j_1, j_2]$ for $1 \leq j_1 \leq j_2 \leq n$, and (4) 4-sided: any rectangular range. Golin et al.[29] provided upper bounds and matching lower bounds for the encoding space required to answer RMQ under these four restricted query cases (see Table 2). Here the encoding space is expressed as an expected value, assuming that the input is arranged in row- or column-major order, uniformly chosen from all permutations of size mn.

Table 2 Expected encoding space (in bits) for RMQ with restricted query ranges [29].

1-sided	2-sided	3-sided	4-sided
$\Theta(\log^2 n)$	$\Theta(\log^2 n \log m)$	$\Theta(n \log^2 m)$	$\Theta(mn)$

For general m and query ranges, Brodal et al. [6] proposed an $O(nm \log m)$-bit encoding for answering RMQ. Based on the their lower bound result of [6], this encoding is asymptotically optimal. For $m = o(n)$, the problem of designing a $o(nm \log n)$-bit data structure for a 2D array that answers queries in sublinear time remains an open problem.

RMQ on (partial) Monge Matrices. A 2D matrix (array) M is called Monge if $M[i_1, j_1] + M[i_2, j_2] \geq M[i_1, j_2] + M[i_2, j_1]$ for any $1 \leq i_1 \leq i_2 \leq m$ and $1 \leq j_1 \leq j_2 \leq n$ (it is more common to define a Monge matrix with $\leq$ rather than $\geq$. In this case, a matrix defined with $\geq$ is referred to as an inverse Monge matrix. All the results presented in this survey can be easily adapted for inverse Monge matrices [38]). Solving RMQ on Monge matrices is of interest due to their various applications in combinatorial optimization and computational geometry [38]. Due to the property of Monge matrices, it is possible to design encoding data structures for RMQ that are more space-efficient than those for general 2D arrays.

■ **Table 3** Data structures on $n \times n$ Monge and partial Monge matrices. Here $\alpha(n)$ denotes the inverse Ackermann function.

Input	Space (in bits)	Query time	ref
Monge	$O(n \log^2 n)$	$O(\log^2 n)$	[37]
	$O(n \log n)$	$O(\log n)$	[22]
	$O(n^{1+\epsilon})$	$O(1)$	
	$O(n \log n)$	$O(\log \log n)$	[23]
Partial Monge	$O(n \log^2 n \cdot \alpha(n))$	$O(\log^2 n)$	[37]
	$O(n \log n)$	$O(\log n \cdot \alpha(n))$	[22]
	$O(n \log n)$	$O(\log \log n)$	[23]

The first non-trivial result was proposed by Kaplan et al. [37] (the journal version of the paper published in 2017 [38]). They showed that for an $n \times n$ Monge matrix, there exists an $O(n \log^2 n)$-bit data structure that can answer RMQ in $O(\log^2 n)$ time.

The result was later improved by Gawrychowski et al. [22] who proposed a data structure using $O(n \log n)$ bits while supporting the query in $O(\log n)$ time. Furthermore, they showed that with $O(n^{1+\epsilon})$ bits of space for any $0 < \epsilon < 1$, the query can be answered in $O(1)$ time. This was further improved in a subsequent work by Gawrychowski et al. [23], where they presented a data structure using $O(n \log n)$ bits of space while supporting $O(\log \log n)$ query time. Additionally, they provided a lower bound of the data structure by showing that any data structure of size $O(n \cdot polylog(n))$ bits requires $\Omega(\log \log n)$ time to answer RMQ on an $n \times n$ Monge matrix. This lower bound was derived by reducing the predecessor problem to RMQ, implying that their data structure is asymptotically optimal (the journal version of the results in [22] and [23] was published in 2020 [24]).

Monge matrices can be generalized to partial Monge matrices, which are Monge matrices with some undefined entries, and the defined entries in each row and column form a contiguous interval. Solving RMQ on partial Monge matrices has applications in areas such as algorithms for maximum flow in planar graphs [38]. In [37], as well as in subsequent works [22] and [23], data structures for partial Monge matrices were proposed by extending those designed for Monge matrices. A summary of these results is presented in Table 3.

3 Range top-k queries

Range selection and range top-k queries are natural extensions of the RMQ, defined as follows: Given a positive integer k and a query range, a range selection query (denoted as sel-k) returns the position of the k-th largest value within the range of the input. Similarly, a range top-k query (denoted as top-k) returns the positions of the k'-largest values within the range for all $k' \le k$. From these definitions, RMQ can be considered a special case of sel-k or top-k with $k = 1$. There are two variations of top-k: (1) sorted top-k reports the answers in sorted order, based on the corresponding values in the input, and (2) unsorted top-k reports the answers in an arbitrary order. For 1D array, Gawrychowski and Nicholson [25] showed that the space lower bound for answering sorted and unsorted top-k are the same within additive lower order terms when $k = o(n)$. Throughout this survey, we use top-k to refer to the sorted one.

For $k = 2$, Davoodi et al. [10] proposed a $(3.272n + o(n))$-bit data structure that can answer top-2 in $O(1)$ time. Their solution is based on the Cartesian tree of the input, combined with additional information to support the queries. Furthermore, they showed that at least $2.656n - O(\log n)$ bits are required to answer top-2.

■ **Table 4** Data structures for top-k on a 1D array.

Query	Space (in bits)		Query time	ref
	Upper bounds			
top-2	$3.272n + o(n)$		$O(1)$	[10]
	$2.755n + o(n)$			[26]
top-k	$O(n \log k)$		$O(k)$	[31]
	$n \log k + n(k+1) \log(1 + 1/k) + o(n \log k)$		$O(\log n)$	[26]
	$1.5n \log k - \Theta(n)$		$poly(k \log n)$	[25]
	Lower bounds			
top-2	$2.656n - O(\log n)$			[10]
	$2.755n - o(n)$			[26]
top-k	$n \log k - O(n + k \log k)$			[31]
	$n \log k + n(k+1) \log(1 + 1/k) - o(n \log k)$			[26]

For general k, the first non-trivial result was introduced by Grossi et al.[31]. This work is an extended journal version of two earlier conference papers published in 2013 [32] and 2014 [44]. They first gave a space lower bound result that at least $n \log k - O(n + k \log k)$ bits are necessary to answer sel-k or top-k, even when the query range is restricted to being 1-sided (i.e., a prefix of the input). Then using the concept of shallow cuttings [9], they design two $O(n \log k)$-bit data structures. These structures support: (1) sel-k in $O(1 + \log k' / \log \log n)$ time, and (2) top-k in $O(k')$ time, for any $1 \le k' \le k$. Hence, both data structures use asymptotically optimal space. Moreover, the query time for (1) is also optimal for any data structures using $O(n \cdot polylog(n))$ space, as shown by the lower bound result of Jørgensen and Larsen [36]. However, when the query range is 1-sided, they show that the time lower bound on range selection queries can be circumvented. Specifically, for 1-sided sel-k, they proposed two data structures that (1) uses $n \log k + o(n \log k) + n$ bits and supports queries in any $\omega(1)$ time, or (2) uses $(1 + \epsilon)n \log k$ bits and supports queries in $O(1/\epsilon)$ time, for any constant $0 < \epsilon < 1$.

The space upper and lower bounds for answering top-k from [31] were later improved by Gawrychowski and Nicholson [23]. They showed that at least $n \log k + n(k + 1) \log(1 + 1/k)$ bits are necessary to answer top-k and proposed an encoding scheme whose space usage is optimal up to lower-order additive terms. For instance, when $k = 2$, their encoding uses $2.755n + o(n)$ bits of space, improving upon the result of Davoodi et al.[10]. In the extended version of their paper [25], they also presented a $(1.5n \log k - \Theta(n))$-bit data structure that can answer top-k in $poly(k \log n)$ time. See Table 4 for a summary of the results on data structures for top-k in a 1D array.

The approximated selection query is to find the position of an element whose rank lies between $k - \alpha s$ and $k + \alpha s$ for a constant $0 < \alpha < 1/2$, where s denotes the length of the query range. In the special case where $k = 1/2$, the query is referred to as an approximate range median query. Bose et al. [4] introduced a data structure that uses $O(n \log n/\alpha)$ bits of space, which can answer approximate range median queries in $O(1)$ time. For general k, El-Zein et al. [13] proposed a data structure with size $O(n/\alpha^3)$ bits that also supports $O(1)$ query time. When k is fixed, they showed that the size of the data structure can be reduced to $O(n/\alpha^2)$ bits while maintaining the same query time. Therefore, the result improves the space usage of [4] for approximated range median queries. Additionally, they showed that both data structures use asymptotically optimal space for constant α by proving an $\Omega(n)$-bit encoding lower bound for approximate range median queries.

Jo et al. [34] studied the top-k on an $m \times n$ 2D array with $m \leq n$. For queries restricted to the range $[1, m] \times [1, j]$, they proposed an $O(\min nk \lg m, nm \lg k)$-bit encoding for sorted top-k and an $O(\min nk \lg(m/k), nm \lg(k/m))$-bit encoding for unsorted top-k. This result implies a space gap between the encodings for sorted and unsorted top-k in 2D, unlike the 1D case, even when $k = o(mn)$. For arbitrary rectangular query ranges, they presented an $(m \log \binom{(k+1)n}{n} + 2nm(m-1) + o(n))$-bit encoding to answer top-k. Compared to the $O(nm \log n)$-bit trivial encoding, which explicitly stores the input, their encoding uses less space when $m = o(\log n)$.

4 Range mode

A *mode* of a multiset S is an element of S that occurs at least as frequently as any other element in S. In the range mode problem, we are given an array A of n elements which we can preprocess so as to answer *range mode queries* efficiently. Given a query range (i, j), the range mode query returns any position, between i and j, of a mode of the multiset $\{A[i], A[i+1], \cdots, A[j]\}$. Krizanc et al. [40] were the first to consider the data structure version of the range mode problem. They proposed two structures achieving different time-space tradeoffs: (i) a data structure that takes $O(n^{2-2\epsilon})$ words and supports queries in $O(n^\epsilon \log n)$ time, for any $0 < \epsilon \leq 1/2$, and (ii) a data structures that takes $O(n^2 \log \log n / \log n)$ words and supports range mode queries in $O(1)$ time. The space bound of the second structure was improved to $O(n^2 / \log n)$ words by Petersen [47]. Subsequently, Petersen and Grabowski [48] improved both the tradeoffs to shave-off a log factor, to obtain the following results: (i) an $O(n^{2-2\epsilon})$ space structure that supports queries in $O(n^\epsilon)$ time, for any $0 < \epsilon < 1/2$, and (ii) an $O(n^2 \log \log n / \log^2 n)$ space structure that supports the queries in $O(1)$ time.

Greve et al. [30] showed that any data structure that uses S memory cells of w bits needs $\Omega(\frac{\log n}{\log(Sw/n)})$ time to answer range mode queries. Chan et al. [7] designed a data structure that uses $O(n)$ words of space and answers range mode queries in $O(\sqrt{n/\log n})$ time. Also, by reducing the Boolean matrix multiplication problem to the range mode problem, they showed that any data structure for range mode must have either $\Omega(n^{\omega/2})$ preprocessing time or $\Omega(n^{\omega/2-1})$ query time in the worst case, where ω denotes the matrix multiplication exponent.

As all the above data structures use at least linear space, they can store the input as part of the data structure. One can improve the space usage significantly by considering approximate versions of the query as described in the next subsection.

4.1 Approximate range mode

In the approximate range mode problem, given a query range (i, j) and a parameter $c \geq 1$, we are interested in returning a position k such that the element $A[k]$ occurs at least $1/c$ times the number of occurrences of the mode of the query range.

Bose et al. [4] were the first to consider this problem whose proposed data structures achieve constant query time for $c = 2, 3$ and 4, using storage space of $O(n \log n)$, $O(n \log \log n)$ and $O(n)$ words, respectively. They also give another data structure that takes $O(n/\epsilon)$ words and answers $(1 + \epsilon)$-approximate range mode queries in $O(\log \log_{1+\epsilon} n)$ time. This gives a linear space data structure that answers c-approximate range mode queries in $O(\log \log n)$ time, for constant c. Greve et al. [30] propose an improved data structure that uses linear space and answers 3-approximate range mode queries in $O(1)$ time. Using this data structure,

they design another data structure that takes $O(n/\epsilon)$ words and answers $(1+\epsilon)$-approximate range mode queries in $O(\log(1/\epsilon))$ time. This gives a linear space data structure that answers c-approximate range mode queries in $O(1)$ time, for constant c.

Finally, El-Zein et al. [13] designed an encoding data structure for approximate range mode queries that occupies $O(n/\epsilon)$ bits of space and answers $(1+\epsilon)$-approximate range mode queries in $O(\log(1/\epsilon))$ time. This improves the space usage of Greve et al. [30] by a factor of $\log n$ while maintaining the query time. They also show that the space usage of their structure is asymptotically optimal for constant ϵ by proving a matching lower bound.

5 Range majority and minority

Range majority and range minority queries are fundamental problems in data mining and theoretical computer science. They involve preprocessing a sequence such that, given a range (i, j), one can efficiently determine elements that occur frequently (majority) or infrequently (minority) within the range. These problems are closely related to range mode queries, which aim to find the most frequent element but are computationally harder.

Range majority. Range majority problems are mainly studied under the assumption that one can access either the original or a compressed version of the input array. For the case when τ is fixed at preprocessing time, Karpinski and Nekrich [39] gave a data structure that takes $O(n/\tau)$ words and supports τ-majority queries in $O((\log \log n)^2/\tau)$ time. Durocher et al. [12] independently considered the same problem and obtained an improved result which takes $O(n \log(1/\tau))$ words and supports queries in optimal $O(1/\tau)$ time. For the case when τ is not fixed at preprocessing time, Chan et al. [8] gave a structure that uses $O(n \log n)$ words and supports queries in optimal $O(1/\tau)$ time. Gagie et al. [19] gave another structure for this case that takes $O(n(H_0 + 1))$ words while supporting queries in optimal time (here H_k denotes the k-th order empirical entropy of the input). Belazzougui et al. [3] designed two improved structures: one that takes $nH_0 + o(n)(H_0 + 1)$ bits and supports queries in $(1/\tau) \cdot \omega(1)$ time, for any slowly growing function, and another structure that takes $(1 + \epsilon)nH_0 + o(n)$ bits, for any constant $\epsilon > 0$ and supports queries in $O(1/\tau)$ time. For the case when the alphabet size σ satisfies $\log \sigma = O(\log w)$, they also gave another structure that uses $nH_0 + o(n)$ bits and supports queries in $O(1/\tau)$ time.

For encoding data structures, Navarro and Thankachan [46] were the first to consider the encoding version of the τ-majority problem They obtained an encoding for range τ-majority queries that takes $O(n\lceil \log(1/\tau)\rceil$ bits and supports range τ'-majority queries, for any $\tau < \tau' < 1$, in time $O((1/\tau) \log \log_w(1/\tau) \log n)$, where $w = \Omega(\log n)$ is the word size. Moreover, they showed that the space usage is optimal by showing that any encoding for range τ-majority queries must use $\Omega(n\lceil \log(1/\tau)\rceil)$ bits. They also propose another structure that takes $O(n\lceil \log(1/\tau)\rceil + n \log \log n)$ bits and answers range τ'-majority queries in $O((1/\tau) \log \log_w(1/\tau))$ time. Finally, Gawrychowski and Nicholson [27] improved the query time of the first structure above of Navarro and Thankachan to obtain a structure that uses $O(n \log(1/\tau))$ bits and supports queries in $O(1/\tau)$ time. Moreover they showed that the space bound is optimal even for a weaker query in which one must decide whether the query range contains at least one τ-majority element. Gawrychowski and Nicholson [28] also showed that for an array of size $2n \log^c n$, for a large constant c, any data structure for checking an existence of element with $1/\log^c n$-majority either needs $\Omega(n^2)$ space or $\Omega(\log^{c-1} n)$ query time, through a reduction from the set intersection problem. This implies that it is unlikely that one can improve the query time to the output sensitive bound of $O(occ + 1)$ when returning $occ = o(1/\tau)$ positions for range τ-majority queries.

Range minority. Parameterized range minority problem was introduced by Chan et al. [8]. In this problem, we need to preprocess a given array such that given a parameter τ and a range (i, j), we need to return an element within the range that is not one of its τ-majorities, if there exists one. Currently, there are no encoding results for range minority queries. Also encoding data structure for minority queries are likely harder than the encoding data structures for majority queries. This is because at most $1/\tau$ elements can be candidates for τ-majority, whereas such a lower bound doesn't exist for minority queries. Here, we mention some indexing data structures for range τ-minority queries.

Chan et al. gave a structure that takes $O(n)$ words and supports queries in $O(1/\tau)$ time. By exploiting the duality of this problem with range τ-majorities problem, Belazzougui et al. [3] obtain exactly the same tradeoffs they obtained for the τ-majority problem, mentioned above. Also, analogous to their range majority structure, Gagie et al. [20] propose a data structure that takes $nH_k + 2n + o(n \log \sigma)$ bits for any $k = o(\log_\sigma n)$, and answers range τ-minority queries in $O((\log \log_w \sigma)/\tau)$ time, where $w = \Omega(\log n)$ is the word size.

References

1 Amihood Amir, Johannes Fischer, and Moshe Lewenstein. Two-dimensional range minimum queries. In *CPM*, volume 4580 of *LNCS*, pages 286–294. Springer, 2007. `doi:10.1007/978-3-540-73437-6_29`.

2 Jérémy Barbay, Johannes Fischer, and Gonzalo Navarro. LRM-trees: Compressed indices, adaptive sorting, and compressed permutations. *Theor. Comput. Sci.*, 459:26–41, 2012. `doi:10.1016/J.TCS.2012.08.010`.

3 Djamal Belazzougui, Travis Gagie, J. Ian Munro, Gonzalo Navarro, and Yakov Nekrich. Range majorities and minorities in arrays. *Algorithmica*, 83(6):1707–1733, 2021. `doi:10.1007/S00453-021-00799-7`.

4 Prosenjit Bose, Evangelos Kranakis, Pat Morin, and Yihui Tang. Approximate range mode and range median queries. In *STACS*, volume 3404 of *LNCS*, pages 377–388. Springer, 2005. `doi:10.1007/978-3-540-31856-9_31`.

5 Gerth Stølting Brodal, Pooya Davoodi, Moshe Lewenstein, Rajeev Raman, and Srinivasa Rao Satti. Two dimensional range minimum queries and fibonacci lattices. *Theor. Comput. Sci.*, 638:33–43, 2016. `doi:10.1016/J.TCS.2016.02.016`.

6 Gerth Stølting Brodal, Pooya Davoodi, and S. Srinivasa Rao. On space efficient two dimensional range minimum data structures. *Algorithmica*, 63(4):815–830, 2012. `doi:10.1007/S00453-011-9499-0`.

7 Timothy M. Chan, Stephane Durocher, Kasper Green Larsen, Jason Morrison, and Bryan T. Wilkinson. Linear-space data structures for range mode query in arrays. *Theory Comput. Syst.*, 55(4):719–741, 2014. `doi:10.1007/S00224-013-9455-2`.

8 Timothy M. Chan, Stephane Durocher, Matthew Skala, and Bryan T. Wilkinson. Linear-space data structures for range minority query in arrays. *Algorithmica*, 72(4):901–913, 2015. `doi:10.1007/S00453-014-9881-9`.

9 Timothy M. Chan and Bryan T. Wilkinson. Adaptive and approximate orthogonal range counting. *ACM Trans. Algorithms*, 12(4):45:1–45:15, 2016. `doi:10.1145/2830567`.

10 Pooya Davoodi, Gonzalo Navarro, Rajeev Raman, and S. Srinivasa Rao. Encoding range minima and range top-2 queries. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 372(2016):20130131, 2014. `doi:10.1098/rsta.2013.0131`.

11 Erik D. Demaine, Gad M. Landau, and Oren Weimann. On cartesian trees and range minimum queries. In *Automata, Languages and Programming, 36th International Colloquium, ICALP Proceedings, Part I*, volume 5555 of *LNCS*, pages 341–353. Springer, 2009. `doi:10.1007/978-3-642-02927-1_29`.

12   Stephane Durocher, Meng He, J. Ian Munro, Patrick K. Nicholson, and Matthew Skala. Range majority in constant time and linear space. *Inf. Comput.*, 222:169–179, 2013. `doi:10.1016/J.IC.2012.10.011`.

13   Hicham El-Zein, Meng He, J. Ian Munro, Yakov Nekrich, and Bryce Sandlund. On approximate range mode and range selection. In *ISAAC*, volume 149 of *LIPIcs*, pages 57:1–57:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. `doi:10.4230/LIPICS.ISAAC.2019.57`.

14   Johannes Fischer. Combined data structure for previous- and next-smaller-values. *Theor. Comput. Sci.*, 412(22):2451–2456, 2011. `doi:10.1016/J.TCS.2011.01.036`.

15   Johannes Fischer and Volker Heun. Theoretical and practical improvements on the rmq-problem, with applications to LCA and LCE. In *Combinatorial Pattern Matching, 17th Annual Symposium, CPM 2006, Barcelona, Spain, July 5-7, 2006, Proceedings*, volume 4009 of *LNCS*, pages 36–48. Springer, 2006. `doi:10.1007/11780441_5`.

16   Johannes Fischer and Volker Heun. Finding range minima in the middle: Approximations and applications. *Math. Comput. Sci.*, 3(1):17–30, 2010. `doi:10.1007/S11786-009-0007-8`.

17   Johannes Fischer and Volker Heun. Space-efficient preprocessing schemes for range minimum queries on static arrays. *SIAM J. Comput.*, 40(2):465–492, 2011. `doi:10.1137/090779759`.

18   Johannes Fischer, Veli Mäkinen, and Gonzalo Navarro. An(other) entropy-bounded compressed suffix tree. In *CPM*, volume 5029 of *LNCS*, pages 152–165. Springer, 2008. `doi:10.1007/978-3-540-69068-9_16`.

19   Travis Gagie, Meng He, J. Ian Munro, and Patrick K. Nicholson. Finding frequent elements in compressed 2d arrays and strings. In *SPIRE*, volume 7024 of *LNCS*, pages 295–300. Springer, 2011. `doi:10.1007/978-3-642-24583-1_29`.

20   Travis Gagie, Meng He, and Gonzalo Navarro. Compressed dynamic range majority and minority data structures. *Algorithmica*, 82(7):2063–2086, 2020. `doi:10.1007/S00453-020-00687-6`.

21   Pawel Gawrychowski, Seungbum Jo, Shay Mozes, and Oren Weimann. Compressed range minimum queries. *Theor. Comput. Sci.*, 812:39–48, 2020. `doi:10.1016/J.TCS.2019.07.002`.

22   Pawel Gawrychowski, Shay Mozes, and Oren Weimann. Improved submatrix maximum queries in monge matrices. In *ICALP (1)*, volume 8572 of *LNCS*, pages 525–537. Springer, 2014. `doi:10.1007/978-3-662-43948-7_44`.

23   Pawel Gawrychowski, Shay Mozes, and Oren Weimann. Submatrix maximum queries in monge matrices are equivalent to predecessor search. In *Automata, Languages, and Programming – 42nd International Colloquium, ICALP Proceedings, Part I*, volume 9134 of *LNCS*, pages 580–592. Springer, 2015. `doi:10.1007/978-3-662-47672-7_47`.

24   Pawel Gawrychowski, Shay Mozes, and Oren Weimann. Submatrix maximum queries in monge and partial monge matrices are equivalent to predecessor search. *ACM Trans. Algorithms*, 16(2):16:1–16:24, 2020. `doi:10.1145/3381416`.

25   Pawel Gawrychowski and Patrick K. Nicholson. Optimal encodings for range min-max and top-k. *CoRR*, abs/1411.6581, 2014. `arXiv:1411.6581`.

26   Pawel Gawrychowski and Patrick K. Nicholson. Optimal encodings for range top-k , selection, and min-max. In *ICALP (1)*, volume 9134 of *LNCS*, pages 593–604. Springer, 2015. `doi:10.1007/978-3-662-47672-7_48`.

27   Pawel Gawrychowski and Patrick K. Nicholson. Optimal query time for encoding range majority. In *Algorithms and Data Structures – 15th International Symposium, WADS Proceedings*, volume 10389 of *LNCS*, pages 409–420. Springer, 2017. `doi:10.1007/978-3-319-62127-2_35`.

28   Pawel Gawrychowski and Patrick K. Nicholson. Optimal query time for encoding range majority. *CoRR*, abs/1704.06149, 2017. `arXiv:1704.06149`.

29   Mordecai J. Golin, John Iacono, Danny Krizanc, Rajeev Raman, Srinivasa Rao Satti, and Sunil M. Shende. Encoding 2d range maximum queries. *Theor. Comput. Sci.*, 609:316–327, 2016. `doi:10.1016/J.TCS.2015.10.012`.

30   Mark Greve, Allan Grønlund Jørgensen, Kasper Dalgaard Larsen, and Jakob Truelsen. Cell probe lower bounds and approximations for range mode. In *ICALP (1)*, volume 6198 of *LNCS*, pages 605–616. Springer, 2010. `doi:10.1007/978-3-642-14165-2_51`.

31   Roberto Grossi, John Iacono, Gonzalo Navarro, Rajeev Raman, and S. Srinivasa Rao. Asymptotically optimal encodings of range data structures for selection and top-k queries. *ACM Trans. Algorithms*, 13(2):28:1–28:31, 2017. `doi:10.1145/3012939`.

32   Roberto Grossi, John Iacono, Gonzalo Navarro, Rajeev Raman, and Srinivasa Rao Satti. Encodings for range selection and top-k queries. In *Algorithms – ESA 2013 – 21st Annual European Symposium, Proceedings*, volume 8125 of *LNCS*, pages 553–564. Springer, 2013. `doi:10.1007/978-3-642-40450-4_47`.

33   Seungbum Jo and Geunho Kim. Space-efficient data structure for next/previous larger-/smaller value queries. In *LATIN 2022: Theoretical Informatics – 15th Latin American Symposium, Proceedings*, volume 13568 of *LNCS*, pages 71–87. Springer, 2022. `doi:10.1007/978-3-031-20624-5_5`.

34   Seungbum Jo, Rahul Lingala, and Srinivasa Rao Satti. Encoding two-dimensional range top-k queries. *Algorithmica*, 83(11):3379–3402, 2021. `doi:10.1007/S00453-021-00856-1`.

35   Seungbum Jo and Srinivasa Rao Satti. Simultaneous encodings for range and next/previous larger/smaller value queries. *Theor. Comput. Sci.*, 654:80–91, 2016. `doi:10.1016/J.TCS.2016.01.043`.

36   Allan Grønlund Jørgensen and Kasper Green Larsen. Range selection and median: Tight cell probe lower bounds and adaptive data structures. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 805–813. SIAM, 2011. `doi:10.1137/1.9781611973082.63`.

37   Haim Kaplan, Shay Mozes, Yahav Nussbaum, and Micha Sharir. Submatrix maximum queries in monge matrices and monge partial matrices, and their applications. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 338–355. SIAM, 2012. `doi:10.1137/1.9781611973099.31`.

38   Haim Kaplan, Shay Mozes, Yahav Nussbaum, and Micha Sharir. Submatrix maximum queries in monge matrices and partial monge matrices, and their applications. *ACM Trans. Algorithms*, 13(2):26:1–26:42, 2017. `doi:10.1145/3039873`.

39   Marek Karpinski and Yakov Nekrich. Searching for frequent colors in rectangles. In *CCCG*, 2008.

40   Danny Krizanc, Pat Morin, and Michiel H. M. Smid. Range mode and range median queries on lists and trees. *Nord. J. Comput.*, 12(1):1–17, 2005.

41   Mingmou Liu. Nearly tight lower bounds for succinct range minimum query. *CoRR*, abs/2111.02318, 2021. `arXiv:2111.02318`.

42   Mingmou Liu and Huacheng Yu. Lower bound for succinct range minimum query. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 1402–1415. ACM, 2020. `doi:10.1145/3357713.3384260`.

43   J. Ian Munro, Patrick K. Nicholson, Louisa Seelbach Benkner, and Sebastian Wild. Hypersuccinct trees – New universal tree source codes for optimal compressed tree data structures and range minima. In *ESA*, volume 204 of *LIPIcs*, pages 70:1–70:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPICS.ESA.2021.70`.

44   Gonzalo Navarro, Rajeev Raman, and Srinivasa Rao Satti. Asymptotically optimal encodings for range selection. In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS*, volume 29 of *LIPIcs*, pages 291–301. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014. `doi:10.4230/LIPICS.FSTTCS.2014.291`.

45   Gonzalo Navarro and Kunihiko Sadakane. Fully functional static and dynamic succinct trees. *ACM Trans. Algorithms*, 10(3):16:1–16:39, 2014. `doi:10.1145/2601073`.

46   Gonzalo Navarro and Sharma V. Thankachan. Optimal encodings for range majority queries. *Algorithmica*, 74(3):1082–1098, 2016. `doi:10.1007/S00453-015-9987-8`.

47   Holger Petersen. Improved bounds for range mode and range median queries. In *SOFSEM 2008: 34th Conference on Current Trends in Theory and Practice of Computer Science, Proceedings*, volume 4910 of *LNCS*, pages 418–423. Springer, 2008. `doi:10.1007/978-3-540-77566-9_36`.

48   Holger Petersen and Szymon Grabowski. Range mode and range median queries in constant time and sub-quadratic space. *Inf. Process. Lett.*, 109(4):225–228, 2009. `doi:10.1016/J.IPL.2008.10.007`.

49   Luís M. S. Russo, Gonzalo Navarro, and Arlindo L. Oliveira. Fully compressed suffix trees. *ACM Trans. Algorithms*, 7(4):53:1–53:34, 2011. `doi:10.1145/2000807.2000821`.

50   Kunihiko Sadakane. Compressed suffix trees with full functionality. *Theory Comput. Syst.*, 41(4):589–607, 2007. `doi:10.1007/S00224-006-1198-X`.

51   Matthew Skala. Array range queries. In *Space-Efficient Data Structures, Streams, and Algorithms*, volume 8066 of *LNCS*, pages 333–350. Springer, 2013. `doi:10.1007/978-3-642-40273-9_21`.

52   Dekel Tsur. The effective entropy of next/previous larger/smaller value queries. *Inf. Process. Lett.*, 145:39–43, 2019. `doi:10.1016/J.IPL.2019.01.011`.

53   Jean Vuillemin. A unifying look at data structures. *Commun. ACM*, 23(4):229–239, 1980. `doi:10.1145/358841.358852`.

54   Hao Yuan and Mikhail J. Atallah. Data structures for range minimum queries in multidimensional arrays. In *SODA*, pages 150–160. SIAM, 2010. `doi:10.1137/1.9781611973075.14`.