Journal of Discrete Algorithms ••• (••••) •••–•••

Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

Range selection and predecessor queries in data aware space and time [☆]

M. Oğuzhan Külekci [a,*], Sharma V. Thankachan [b]

[a] Informatics Institute, Istanbul Technical University, Istanbul, Turkey
[b] Department of Computer Science, University of Central Florida, Orlando, USA

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Data structures
Range searching
Predecessor search
Data compression

ABSTRACT

On a given vector $X = \langle x_1, x_2, \dots, x_n \rangle$ of integers, the *range selection* (i, j, k) query is finding the k-th smallest integer in $\langle x_i, x_{i+1}, \dots, x_j \rangle$ for any (i, j, k) such that $1 \le i \le j \le n$, and $1 \le k \le j - i + 1$. Previous studies on the problem proposed data structures that occupied additional $O(n \cdot \log n)$ bits of space over the X itself that answer the queries in logarithmic time. In this study, we replace X and encode all integers in it via a single wavelet tree by using $S = n \cdot \log u + \sum \log x_i + o(n \cdot \log u + \sum \log x_i)$ bits, where u is the number of distinct $\lfloor \log x_i \rfloor$ values observed in X. Notice that u is at most 32 (64) for 32-bit (64-bit) integers and when $x_i > u$, the space used for x_i in the proposed data structure is less than the Elias-δ coding of x_i. Besides data-aware coding of X, the range selection is performed in $O(\log u + \log x')$ time where x' is the k-th smallest integer in the queried range. This result is *adaptive* and achieves the range selection regardless of the size of X, and the time-complexity depends on the answer itself. Additionally, we can answer range predecessor queries (i, j, x'): return the largest element $y \le x'$ in $\langle x_i, x_{i+1}, \dots, x_j \rangle$, in $O(\log u + \log x')$ time. In summary, to the best of our knowledge, we present the first algorithms using data-aware space and time for the general range selection and predecessor problems.

1. Introduction and previous work

Range selection (or quantile) on a given vector $X = \langle x_1, x_2, \dots, x_n \rangle$ of integers is the process of selecting the k-th smallest integer among the subsequence $\langle x_i, x_{i+1}, \dots, x_j \rangle$ of X for any (i, j, k) conforming $1 \le i \le j \le n$, and $1 \le k \le j - i + 1$. The special case of the problem when $k = \lceil \frac{j-i+1}{2} \rceil$ is called the range median query, which is detecting the element appearing in the middle position when the sequence is sorted. Since *median* is an important parameter in statistics, it has received significant attention in the related studies [3,9,2,16] among with a wide range of applications such as document listing, or 2-d searching, just to name a few.

The naive solution of the problem is sorting the elements in the queried range, which can be achieved in $O(d \cdot \log d)$ time, for $d = j - i + 1$, and then simply picking up the integer appearing at k-th position of the sorted list. That sorting process may be repeated $O(n^2)$ times on all possible (i, j) ranges, and the results may be stored in quadratic space so that

[☆] Early part of this work appeared in the *Proceedings of Data Compression Conference* (DCC), 2015 [18].
[*] Corresponding author.
E-mail addresses: kulekci@itu.edu.tr (M.O. Külekci), sharma.thankachan@ucf.edu (S.V. Thankachan).

http://dx.doi.org/10.1016/j.jda.2017.01.002
1570-8667/© 2017 Elsevier B.V. All rights reserved.

Table 1
Time and space complexities of the solutions proposed to answer general range selection query (i, j, k) on a given integer sequence $X = \langle x_1, x_2, \ldots, x_n \rangle$, where σ is the number of distinct elements in X, u is the number of distinct $\lfloor \log x_i \rfloor$ values observed on X, and x' is the k-th smallest value we seek in the range (answer of the query). $S(X)$ denotes the total space used to represent the integers in X, which is typically $n \cdot \log \max\{x_i \in X\}$ bits. Notice that $O(n)$ *words* space is equal to $O(n \log n)$ bits in word-RAM model assuming the word size is $\Theta(\log n)$.

	Time-complexity	Space complexity (in bits)
Gagie et al. (2009) [9]	$O(\log \sigma)$	$S(X) + n \log \sigma + o(n \log \sigma)$
Brodal et al. (2011) [2]	$O(\log n / \log \log n)$	$S(X) + O(n \log n)$
Jørgensen & Larsen (2011) [16]	$O(\log k / \log \log n + \log \log n)$	$S(X) + O(n \log n)$
Chan & Wilkinson (2013) [4]	$O(\log k / \log \log n + 1)$	$S(X) + O(n \log n)$
This study	$O(\log u + \log x')$	$(n \log u + \sum_{\forall i} \log x_i)(1 + o(1))$

any query can be answered in $O(1)$ time. Such a path is not of interest due to the heavy load of sorting, and also the requirement of quadratic space, a combination that would be impractical for relatively large n.

It is noteworthy that the *quickselect* algorithm, that was introduced in an early study by Hoare [14] achieves range selection in linear time by modifying the *quicksort* [15]. In *quickselect*, the queried range is split into two by selecting a random pivot, and then collecting the items that are less than this pivot in one bucket and rest in the other, akin to the *quicksort*. Same splitting is recursively applied on the bucket that contains the kth smallest item until the answer is reached. Since at each step only one of the two buckets is processed, the overall time complexity becomes $O(d)$ for the range select assuming the selected pivot splits the range into equal sized partitions. In the worst case, where the pivot just decreases the range only by one, the time-complexity becomes $O(d^2)$.

The naive solution and the *quickselect* algorithm described above forms a base-line for evaluating the solutions of the general range selection queries, where it is seen that given a dynamic range the linear-time solution is trivial. Thus, the research in that direction focused on achieving logarithmic time-complexities as we review in section 2.

Range predecessor is a related problem, where the query (i, j, x') asks to report the largest element $y \leq x'$ in $\langle x_i, x_{i+1}, \ldots, x_j \rangle$, in $O(\log u + \log x')$ time. Here also, $O(1)$ query time is trivial using quadratic space. However, our aim is to develop space and time efficient solutions for both problems with data-dependent wavelet trees, that take into account of the compressibility of the data.

1.1. Previous studies

In [16], the authors classify the solutions on range selection problem into two as polynomial-space and near-linear-space algorithms according to the space occupied by their data structures. Although the polynomial-space algorithms provide constant time query handling, they are not of interest to us due to their large memory consumption. Table 1 lists the time and space complexities of some of the solutions offered to date for range selection.

In [3], following the approach introduced in [10], the authors proposed to create a binary search tree over X in a bottom-up approach, where they first sort X, create the leaf nodes associated with an integer $x_i \in X$. In the inner nodes of the tree they keep two indices per integer included in the leaves of that node (see [3] for details). They achieve $O(\log n / \log \log n)$ time-complexity, and the data structure they use occupies $O(n)$ words. Similarly, [2] achieves the same time-complexity $O(\log n / \log \log n)$.

In Gagie et al. [9], instead of a bottom-up approach, a top-down scheme was preferred by creating a wavelet tree. At each node of the wavelet tree a bitmap is maintained in which the integers represented in that node that are smaller than the pivot value are marked with 0, and others by 1. By selecting the pivot equal to the median of the sequence at each step, the tree becomes a balanced tree with height $\lceil \log \sigma \rceil$, where σ is the number of distinct elements in X. The processing of a range selection query begins at the root node by counting the number of 0s and 1s from the i-th bit to the j-th bit. In case k is larger than the number of 0s then the right child is selected and k is updated accordingly. Otherwise, left child is followed without updating k value. In each case the (i, j) values are updated properly to keep track on the next node. These operations require `rank` queries to be run efficiently, and that can be computed in $O(1)$ time by using an additional $o(.)$ space [22,23,20].

Jørgensen and Larsen introduced the first adaptive solution to the problem. In their scheme the space usage was again linear in the size of the input X sequence. However, the time-complexity was improved as $O(\log k / \log \log n + \log \log n)$. Finally Chan and Wilkinson [4] presented a linear space data structure with optimal $O(\log k / \log \log n)$ query time.[1]

On a related node, a generalization of this problem called tree path selection queries is studied in [13,24]. Here the integers are associated with the nodes in a tree and an (i, j, k) query asks to report the k-th smallest integer among the integers associated with the nodes on the path from node i to j.

For the range predecessor problem, there are several space–time trade-offs known (refer to Table 2). Firstly, the quadratic space can be reduced to $O(n^{1+\epsilon})$ for any constant $\epsilon > 0$ and keep the query time constant [5]. On the other hand, wavelet trees can handle the queries in $O(\log \sigma)$ time [8]. Among linear space solutions, we have an $O(\log n / \log \log n)$ time solution

[1] Note that the space and the query time for their range counting structure is $O(n \log \log n)$ and $O(\log k / \log \log n + \log \log n)$ respectively.