Journal of Discrete Algorithms 36 (2016) 63–75

Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

Space efficient data structures for nearest larger neighbor [☆]

Varunkumar Jayapaul [a], Seungbum Jo [b], Rajeev Raman [c], Venkatesh Raman [d], Srinivasa Rao Satti [b,*]

[a] *Chennai Mathematical Institute, India*
[b] *Seoul National University, South Korea*
[c] *University of Leicester, United Kingdom*
[d] *The Institute of Mathematical Sciences, India*

A R T I C L E   I N F O

Article history:
Available online 13 January 2016

Keywords:
Nearest larger neighbor
Succinct data structures
Indexing model
Encoding model
Cartesian tree
Range minimum query

A B S T R A C T

Given a sequence of n elements from a totally ordered set, and a position in the sequence, the nearest larger neighbor (NLN) query returns the position of the element which is closest to the query position, and is larger than the element at the query position. The problem of finding the nearest larger neighbors of all the elements in a one-dimensional array has attracted interest due to its applications for parenthesis matching and in computational geometry [1–3]. We consider a data structure version of this problem, which is to preprocess a given sequence of elements to construct a data structure that can answer NLN queries efficiently. We consider time-space tradeoffs for the problem in both the *indexing* and the *encoding* models when the input is in a one dimensional array, and also initiate the study of this problem on two-dimensional arrays.

1. Introduction and motivation

Given a sequence of n elements from a totally ordered set, and a position in the sequence, the *nearest largest neighbor* (NLN) query asks for the position of an element which is closest to the query position, and is larger than the element at the query position. More formally, given a one-dimensional (1-D) array $A[1\dots n]$ of length n containing elements from a totally ordered set, and a position i in A, we define the query:

NLN(i, A) return the index j such that $A[j] > A[i]$ and $|i-j| = \min\{k : A[i+k] > A[i]$ or $A[i-k] > A[i]$ for $k > 0\}$. Ties are broken to the left, and if there is no element greater than the query element, the query returns the answer ∞.

We omit the second parameter when the array A is clear from the context. In a similar way, we can the define NLRN (*right* nearest larger neighbor), and NLLN (*left* nearest larger neighbor) queries, which return the position of the nearest larger neighbor to the right and left, respectively, of the query position. In a symmetric way, one can also define nearest *smaller* neighbor problems. In this paper, we will stick to the version that seeks the larger neighbors.

[☆] Preliminary version of these results have appeared in the proceedings of the 25th International Workshop on Combinatorial Algorithms (IWOCA-2014) [15] and in the proceedings of International Workshop on Algorithms and Computation (WALCOM-2015) [16].
[*] Corresponding author.
E-mail addresses: varunkumarj@cmi.ac.in (V. Jayapaul), sbcho@tcs.snu.ac.kr (S. Jo), r.raman@leicester.ac.uk (R. Raman), vraman@imsc.res.in (V. Raman), ssrao@cse.snu.ac.kr (S.R. Satti).

http://dx.doi.org/10.1016/j.jda.2016.01.001
1570-8667/© 2016 Elsevier B.V. All rights reserved.

We exhibit connections between the NLN problem and the well-studied problem of supporting *range maximum* queries on a given array. Given an array A, the query $\mathsf{RMQ}(i, j)$ (range maximum query) returns a position k between i and j such that $A[k]$ is a maximum element among $A[i, \ldots, j]$.

Two-dimensional NLN We also consider a natural extension of the NLN problem to two-dimensional (2-D) arrays. Here, we define the NLN of a query position as the closest position in the 2-D array, in terms of the L_1 distance, that contains an element larger than the element at the query position. More formally, given a position (i, j) in $A[1 \ldots n][1 \ldots n]$, $\mathsf{NLN}((i, j), A) = (i', j')$ such that $A[i', j'] > A[i, j]$, and $|i - i'| + |j - j'| = \min\{|x| + |y| : A[i + x, j + y] > A[i, j]\}$. If there is no element greater than the query element, the query returns the answer (∞, ∞). Again, when the array A is clear from the context, we omit the second parameter.

Encoding and indexing models This paper is concerned with the *data structure* versions of the above queries: given the array A, we aim to preprocess it into a data structure that answers a (long) sequence of queries. Our concern in this paper is the time taken to answer queries and the space used by the data structure after pre-processing. We assume a standard *word RAM* model [17] with $O(1)$-time arithmetic and bitwise boolean operations on words of $O(\lg n)$ bits each,[1] and we count space in terms of the number of bits used. In this model, for all the Nearest Larger Neighbor problems considered, there are trivial data structures that use a linear (in the size of A) number of words of space and answer queries in $O(1)$ time: just precompute and store the answer for all possible queries. However, we seek significantly more space-efficient, or *succinct*, solutions to this problem.

We consider these problems in two different models that have been studied in the succinct data structures literature, namely the *indexing* and *encoding* models [5]. In the indexing model, the queries can be answered by probing the data structure as well as the input data, whereas in the encoding model, the query algorithm cannot access the input data. The size of the data structure in the encoding model is also referred to as the *effective entropy* [13] (of the input data, with respect to the problem).

Previous work and motivation These kinds of problems have attracted much attention. In addition to the data structuring problems, the *off-line* variants, usually called *All Nearest Larger Neighbors* (or similar), which consist in computing answers for all possible input positions, have also been studied. For example, Berkman et al. [3] gave efficient parallel algorithms for the 1-D off-line problem and showed their importance as a preprocessing routine for answering range minimum queries, triangulation algorithms, reconstructing a binary tree from its traversal orders and matching a sequence of balanced parentheses [3].

Fischer et al. [12] considered the data structuring problem of supporting NLRN and NLLN queries, and showed how a data structure supporting these two queries can be used in obtaining an *entropy-bounded compressed suffix tree* representation. (They considered the nearest smaller version of the problem instead of the nearest larger version, and named the operations NSV and PSV, for the next and previous smaller values, respectively.) They considered the problem of supporting NLRN and NLLN in the indexing model, and obtained the following time-space tradeoff result. For any $1 \le c, \ell \le n$, one can construct an index of size

$$O\left(\frac{n}{c}\lg c + \ell\frac{n\lg\lg n}{\lg n} + \frac{n\lg n}{c^\ell}\right)$$

bits, to answer queries in $O(c\ell)$ time. Fischer et al. [12] also gave an encoding that supports the PSV and NSV queries in constant time, using $4n + o(n)$ bits. The encoding size was later reduced to the optimal $2.54n + o(n)$ bits by Fischer [10].

For the case of binary sequences, the data structure version of the NLN problem can be solved by building an auxiliary structure to support *rank* and *select* queries on the bit vector [19]. This uses $o(n)$ bits of extra space, in addition to the input array, and answers NLN (and also NLRN and NLLN) queries in $O(1)$ time.

Given a 2-D array A, Asano et al. [2] recently considered the off-line problem and showed that this problem can be solved in $O(n^2 \lg n)$ time (and more generally, for any d-dimensional array for $d > 1$ in $O(n^d \lg n)$ time when all elements are distinct). They also gave time-space trade-offs for this problem. To the best of our knowledge, the data structure version of the two-dimensional NLN problem has not been considered earlier.

Our results For the 1-D case, we first observe in Section 2.1, a $2n - o(n)$-bit lower bound for NLRN in the encoding model, by relating the problems to the RMQ problem. We also give a lower bound via directly counting the number of distinct configurations, which may be of independent interest. In terms of upper bounds in the encoding model, a $2n + o(n)$-bit structure to answer NLRN queries, and a $2.54n + o(n)$-bit structure to answer NLN queries exist through a structure of Fischer et al. [12]. We develop a $2n + o(n)$-bit structure for NLN when all elements in A are distinct.

Then in Section 2.2, we look at the problems in the indexing model. We give a lower bound for time-space tradeoff for NLN and NLRN queries adapting the lower bound tradeoff proof of Brodal et al. [5] for RMQ. We also provide an algorithm that matches the tradeoff for NLN – an index that uses $O(n/c)$ bits and supports queries in $O(c)$ time, for any parameter

[1] We use $\lg n$ to denote $\log_2 n$.