The Range 1 Query (R1Q) Problem

Rezaul Chowdhury[*], Pramod Ganapathi[†], Yuan Tang[‡]

Abstract: Given a bit array of size n and two indices i and j, find efficiently if there is a 1 in the subarray $[i, j]$. We call this problem the "Range 1 Query" (R1Q) problem. The problem can be easily generalized to higher dimensions. We give exact and approximation algorithms to solve the problem in 1D and higher dimensions. We can also answer queries for right triangles, isosceles triangles and in general, polygons with certain constraints. The applications include the Pochoir stencil compiler [1], where we have to answer octagonal queries to select optimized code clones.

Exact Algorithm: The 1D algorithm is as follows. We divide the input array into blocks of size 2^p for different values of p greater than some threshold. We preprocess these blocks so that any query of length exactly 2^p that crosses a block boundary or lies completely inside a block can be answered in constant time. To answer intra-block queries, we use Four Russians trick.

To answer the query R1Q(i, j), we find at most two possibly overlapping blocks of size $2^k <= j - i + 1$ (for some positive integer k) that completely cover the range, and then compute the answer from the two queries. For input size of n bits, preprocessing takes $o(n)$ bits of space and $O(n)$ time and query execution takes $\mathcal{O}(1)$ time. The algorithm can be extended to higher dimensions.

Approximation Algorithm: As in the exact algorithm, we preprocess the blocks of size 2^p for different values of p, and this time we use Cormode-Muthukrishnan's Count Min (CM) sketch data structure [2] to store the preprocessed data. We create separate CM sketches for different block sizes so that queries of different sizes can be answered approximately in sublinear space and constant time.

By setting the value of the error rate, the space usage of the approximation algorithm can be made arbitrarily small compared to the exact algorithm. The query execution is similar to the exact algorithm.

Other Shapes: We answer right triangular queries as follows. We preprocess all the right triangles with horizontal base in eight orientations, with either base

[*]Stony Brook University, U.S.A.; `rezaul@cs.stonybrook.edu`
[†]Stony Brook University, U.S.A.; `pganapathi@cs.stonybrook.edu`
[‡]Fudan University, China; `yuantang@fudan.edu.cn`

length or height a power of two. Now we split a general horizontal base right triangle into two overlapping right triangles with base length or height a power of two, and possibly a rectangle. We find the answers to these triangles and rectangle using the preprocessed data and hence answer the query triangle.

We answer polygonal queries as follows. The polygon is split into a collection of right triangles and rectangles that completely lie inside the given polygon and completely cover it, and each of which can be answered using preprocessed data. The results of these right triangular and rectangular queries are combined to answer the input polygonal query.

References

[1] Y. Tang, R. Chowdhury, B. Kuszmaul, C. K. Luk, C. Leiserson. *The Pochoir Parallel Stencil Compiler*. Proceedings of the SPAA, pp. 117-128, 2011.

[2] G. Cormode, S. Muthukrishnan. *An Improved Data Stream Summary: The Count-Min Sketch and Its Applications*. Journal of Algorithms, vol. 55, pp. 58-75, 2005.