6.897 Advanced Data Structures (Spring'05)

Prof. Erik Demaine TA: Mihai Pătraşcu

Problem 8 – Solution

From RMQ to 3-sided queries. The relation is immediate. For each $y \in \{1, \dots, n\}$, let $A[y] = \min\{x \mid (x, y) \in S\}$. When we want to query the range $[0, b] \times [c, d]$, we use an RMQ query to find $t = \min\{A[y] \mid y \in [c, d]\}$. If $t > b$, the range is empty; otherwise, it contains at least one point.

From 3-sided to general queries. Without loss of generality, n is a power of two. Consider a perfect binary tree over the x-coordinate. Each node represents a vertical strip of space; say this is $[a, b] \times [1, n]$. For each such strip (except the root, which gives the entire space), we build a structure for 3-sided queries. If the node is the right child of its parent, this structure assumes the left side of the rectangle is on the a abscissa. If the node is a left child, it assume the right side in fixed to b. Even though we have described 3-sided queries as fixing one side to 0, we can actually fix it to anything, by horizontal translation, and possibly a reflection.

For every 3-sided structure that we build, we must also include a predecessor structure. This is needed because the 3-sided structure has less than n points. Thus, we must convert from the original rank space to this new (sparser) rank space. This can be done by building a predecessor structure on the set of y-coordinates of the points in the 3-sided structure. We then run two predecessor queries to convert the y boundaries of the rectangle to the new rank space. We implement these predecessor structures using y-fast trees. Note that the universe is $\{1, \dots, n\}$, so a query takes $O(\lg \lg n)$ time.

The space for each of the 3-sided structures is σ times the number of points in the structure. Every point appear in exactly $\lg n$ structures (one for each ancestor in the tree of its x coordinate). Thus, the total space is $O(n \lg n \cdot \sigma)$. The predecessor structures take space linear in the number of points of the 3-sided structures, so they form an additional constant factor.

Now assume we want to query the range $[a, b] \times [c, d]$. We find the lowest common ancestor (call it v) of a and b. This can be done in constant time: we take the xor of a and b and find the most significant set bit (which was in our standard set of bit tricks). Now, the left child of v contains a; let m be the rightmost abscissa under this node. The right child of v contains b; the leftmost abscissa under it is $m + 1$. We have broken our range query into two queries: $[a, m] \times [c, d]$ and $[m + 1, b] \times [c, d]$. Since we are doing existential queries, we take the or of the two answers. Both of these queries are 3-sided queries, in the left and right children of v, respectively. We use the predecessor structures to convert c and d into the rank space of the 3-sided structures, and then run the 3-sided queries. Thus, our running time is $2\tau + O(\lg \lg n)$.