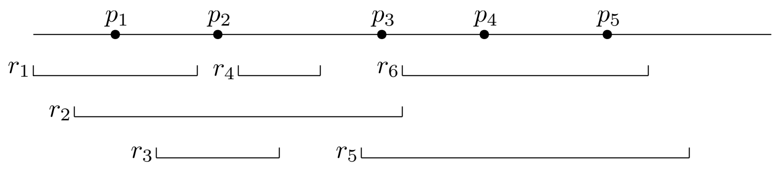
INFS 4205/7205: Exercise Set 6

Prepared by Yufei Tao and Junhao Gan

Problem 1. Consider the following instance of 1D *sorted* rectangles-join-points problem. Recall that we discussed an algorithm in the class, which maintains a linked list L as it processes the intervals and points from left to right. Answer the following questions:

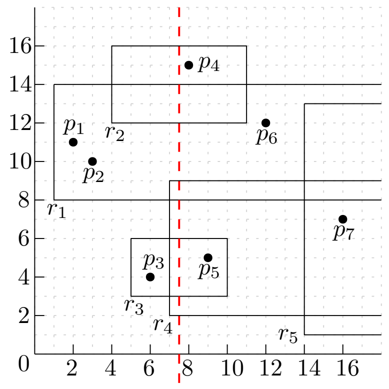
- What are the contents right *before* the algorithm processes point p_3?
- What are the contents right *after* processing point p_3?



Solution. $L = \{r_2, r_3, r_4, r_5\}$ right before processing p_3. $L = \{r_2, r_5\}$ right after processing p_3.

Problem 2. Consider running the 2D rectangles-join-points algorithm (as discussed in the class) on the input shown in the figure below (5 rectangles and 7 points). The algorithm first divides the data space using the red line, and then recursively processes the left and right of the line, respectively. Accordingly, in the recursion hierarchy, there are two child nodes of the root: corresponding to the processing on left and right of the line, respectively.

Let us focus on the right child of the root in the recursion hierarchy, where the algorithm processes (i) rectangles r_1, r_2, r_3, r_4, and r_5 (note that rectangles r_1, r_4, r_5 are "3-sided", namely, their right edges are aligned with the right boundary of the data space), and (ii) points p_4, p_5, p_6, and p_7. Here, the algorithm needs to construct a 1D instance of the rectangles-join-points problem. What are the intervals and points in this instance?



Solution. The 1D instance at the highest level of recursion consists of: (i) a set R of intervals $\{[8, 14], [2, 9]\}$ (i.e., y-projections of rectangles r_1, r_4), and (ii) a set P of 1D points $\{15, 5, 12, 7\}$ (i.e., y-projections of points p_4, p_5, p_6, p_7).

Problem 3. In the *segment join problem*, we are given a set H of horizontal segments, and a set V of vertical segments, both in $\mathbb{R}^2$. We want to report all pairs of $(h, r) \in H \times V$ such that h

intersects r. Design an algorithm to do so in $O(n \log n + k)$ time, where $n = |H| + |V|$, and k is the number of pairs in the result.

Solution. The algorithm accepts inputs: H, V (as stated in the problem), and a set X of x-coordinates. At the beginning of the algorithm, X consists of (i) the x-coordinates of all the segments in V, (ii) the value x_1 for every interval $[x_1, x_2] \times y$ in H, provided that $x_1 \neq -\infty$, and (iii) the value x_2 for every interval $[x_1, x_2] \times y$ in H, provided that $x_2 \neq \infty$. If an horizontal interval $[x_1, x_2] \times y$ has either x_1 or x_2 in X, we say that it *contributes to* X.

- Let R_{span} be the set of horizontal segments of H that do not contribute to X.
- Construct an instance of the 1D rectangles-join-points problem with a set R of intervals and a set P of points decided as follows. R is obtained by projecting all the segments of V onto y-axis, and P is obtained by projecting all the segments of R_{span} onto y-axis. Solve this 1D instance.
- Divide X into two disjoint subsets X_1 and X_2 with the same size (by respecting the ordering) by a vertical line ℓ.
- Let H_1 (or H_2) be the set of segments in H that intersect with the left (or right, resp.) side of ℓ. Let V_1 (or V_2) be the set of segments in V that fall on the left (or right, resp.) side of ℓ.
- Solve the left sub-problem with inputs H_1, V_1 and X_1, and the right sub-problem with inputs H_2, V_2 and X_2.

Problem 4. In the *2D spatial join problem*, we are given two sets—R and S respectively—of axis-parallel rectangles in $\mathbb{R}^2$, and want to report all pairs $(r, s) \in R \times S$ such that r intersects s. Design an algorithm to do so in $O(n \log n + k)$ time, where $n = |R| + |S|$, and k is the number of pairs in the result. Note that the running time should hold in the worst case (as opposed to the expected case).

Solution. Define:

- $H(R)$ as the set of horizontal edges of the rectangles in R;
- $V(R)$ as the set of vertical edges of the rectangles in R;
- $P(R)$ as the set of corner points of the rectangles in R;
- $H(S), V(S)$, and $P(S)$ similarly for S.

For a pair of intersecting rectangles (r, s), at least one of the following four cases holds: (i) a horizontal boundary of r intersect a vertical boundary of s, (ii) a vertical boundary r intersect a horizontal boundary of s, (iii) r is fully contained in s, or (iv) s is fully contained in r. Thus, the 2D spatial join problem can be settled by solving the following four problems:

- A segment join problem with inputs of $H(R)$ and $V(S)$.
- A segment join problem with inputs of $H(S)$ and $V(R)$.
- A 1D rectangles-join-points problem with inputs R and $P(S)$.
- A 1D rectangles-join-points problem with inputs S and $P(R)$.

A naive implementation of the above strategy would report (r, s) multiple times. To avoid this, we utilize the observation that, when (r, s) is discovered by one of these four sub-problems, we can determine in $O(1)$ time whether it needs to be reported in the other sub-problems as well. Hence, we can report (r, s) only once by enforcing a reporting convention like "report the pair in sub-problem i, only if it will not be reported in any sub-problems $j < i$". A similar convention can also be adopted to avoid reporting (r, s) multiple times within each sub-problem.

Problem 5. In this problem, we extend the rectangles-join-points problem to arbitrary dimensionality d. We are given a set R of axis-parallel rectangles, and a set P of points, both in $\mathbb{R}^d$. The objective is to report all pairs $(r, p) \in R \times P$ such that rectangle r covers point p. Design an algorithm to do so in $O(n \log^{d-1} n + k)$ time, where $n = |R| + |P|$, and k is the number of pairs in the result.

Solution. To solve the d-dimensional rectangles-join-points problem, we just need to modify the 2D algorithm described in the slides to construct an instance of the $(d-1)$-dimensional (rather than 1D) rectangles-join-points problem for R_{span}. By a similar analysis and plugging in the fact that the 2D case can be solved in $O(n \log n + k)$ time, the running time of the d-dimensional rectangles-join-points algorithm is bounded by $O(n \log^{d-1} n + k)$.