Computational Geometry 3rd set

Emmanouil Thomas Chatzakis 2021030061

May 2025

Exercise 1

Describe an algorithm for the following problem: given a set S of rectangles with sides parallel to the axes, report all pairs of rectangles that intersect. Your algorithm **must run** in time $O(n \log n + k)$.
Hint: use plane sweeping and the Priority Search Tree

Solution

Each rectangle R is defined by four values: $(x_{\min}, x_{\max}, y_{\min}, y_{\max})$. Two rectangles intersect if and only if their intervals on both the x-axis and y-axis. overlap.

First, we sort all rectangles in our set by their $x_{\min}$ coordinate. Then, for each rectangle, we create two events: one at $x_{\min}$ (left edge) and one at $x_{\max}$ (the right edge). As we sweep from left to right across the plane, these events allow us to detect when we enter and exit each rectangle, so we can maintain the set of rectangles currently intersected by the sweep line.

Event Type	**Description**
Start event	At $x_{\min}(R)$ (left edge of R)
End event	At $x_{\max}(R)$ (right edge of R)

During the plane sweep, we use a Priority Search Tree (PST) to maintain the set of *active* rectangles,that is, rectangles whose start event has been encountered by the sweeping, but whose end event has not yet been passed.

- **Insertion:** When the sweep line encounters the left edge ($x_{\min}$) of a rectangle R, we insert its y-interval $[y_{\min}(R), y_{\max}(R)]$ into the PST.
- **Querying:** Before inserting R's y-interval, we query the PST to find all active rectangles whose y-intervals overlap with $[y_{\min}(R), y_{\max}(R)]$. Each such overlap represents an intersection between R and another active rectangle.
- **Deletion:** When the sweep line reaches the right edge ($x_{\max}$) of rectangle R, we remove its y-interval from the PST, as R is no longer active.

Algorithm Outline

1. For each rectangle R, create two events: a *start* event at $x_{\min}(R)$ and an *end* event at $x_{\max}(R)$.
2. Sort all events in increasing order of x-coordinate.
3. Initialize an empty Priority Search Tree (PST) to store active y-intervals, and an empty list to record intersecting pairs.
4. For each event, in order:
   (a) **If it is a start event for rectangle R:**
      i. Search the PST for all active intervals that overlap with $[y_{\min}(R), y_{\max}(R)]$.
      ii. For each overlapping interval found, record the corresponding pair of rectangles as intersecting.
      iii. Insert $[y_{\min}(R), y_{\max}(R)]$ into the PST.
   (b) **If it is an end event for rectangle R:**
      i. Remove $[y_{\min}(R), y_{\max}(R)]$ from the PST.
5. After processing all events, the list contains all pairs of intersecting rectangles.

Complexity Analysis

- For each rectangle there are two events. So for n rectangles we got 2n events
- Each event involves a PST query, which takes $O(\log n + i)$ time, where i is the number of overlaps reported for that event.
- Insertion and deletion in the PST each take $O(\log n)$ time.
- The total running time is $O(n \log n + k)$, where k is the total number of intersecting pairs.

Exercise 2

For the space cost of the range tree, prove that the following recurrence has solution

$$S(d, n) = \begin{cases} 2S(d, n/2) + S(d-1, n) & \text{if } d > 1 \\ \Theta(n) & \text{if } d = 1 \end{cases}$$

Solution

We will prove the result using mathematical induction on d.

- **Base case:** $d = 2$**:**

  $$S(2, n) = 2S(2, n/2) + \Theta(n)$$

  According to the Master Theorem (case 2), we have:

  $$S(2, n) = \Theta(n \log n)$$

  Thus, the base case holds.

- **Inductive step:** Assume the statement is true for $d = k - 1$,

  $$S(k-1, n) = \Theta(n \log^{k-2} n)$$

  Now, consider $d = k$:

  $$S(k, n) = 2S(k, n/2) + S(k-1, n)$$

  Substitute the inductive hypothesis:

  $$S(k, n) = 2S(k, n/2) + \Theta(n \log^{k-2} n)$$

  Again, by the Master Theorem (case 2), it follows that:

  $$S(k, n) = \Theta(n \log^{k-1} n)$$

Therefore, by induction, the result holds for all $d \in \mathbb{N}$:

$$S(d, n) = \Theta(n \log^{d-1} n)$$

Exercise 3

In some applications we are only interested in the number of keys falling within a range. Describe a d-dimensional range tree, changed so that it can report the number of keys in the range $[a, b]$ $(a, b \in \mathbb{R}^d)$ in time $O(\log^d n)$. Prove the query cost and the construction cost.

Solution

To find the number of keys that lie within a given interval, it is sufficient for each node of the main tree and the auxiliary trees to store, along with the key and the pointer to the auxiliary tree, the number of nodes in its subtree, including itself.The search process in the range tree is similar to that of a standard range query, when a range counting query is performed. However, whenever a subtree

is found to be completely contained within the query range, instead of recursively visiting all its nodes, we can immediately add the stored subtree size to our count.

The query proceeds recursively: at each level, the d-dimensional problem is reduced to a set of $(d-1)$-dimensional subproblems using the auxiliary trees stored at each node.

Query Time Complexity: By storing subtree sizes, the range counting query in a d-dimensional range tree can be answered in $O(\log^d n)$ time, where n is the number of points stored in the tree. This is because, at each level of recursion, only $O(\log n)$ nodes need to be considered, and the process is repeated for each dimension.

Construction and Space Complexity: The construction process of the d-dimensional range tree is similar to the standard range tree, but with the addition that each node stores a field representing the size of its subtree. This field can be computed using the recursion for a node u:

$$u.\text{size} = 1 + u.\text{left.size} + u.\text{right.size}$$

Therefore, the total construction time for a d-dimensional range tree with n points is:

$$O(n \log^{d-1} n)$$

Range Tree

Exercise 4

Assume that, for any N large enough, we can construct a dataset $\mathcal{I}_N$ of N 2-d points over the integer domain $D = [0 : N)^2$, such that every rectangle of area a which lies inside D contains approximately $\Theta\left(\frac{aN}{D}\right)$ points.
Using the redundancy theorem, prove that the indexability trade-off for rectangular range queries is

$$r = \Omega\left(\frac{\log(N/B)}{\log A}\right)$$

Solution

We follow the main idea from the references. First, we create a set of queries $q_1, \ldots, q_k$, where each query covers at least $B/2$ points. We focus on rectangles that have area $a = BN/2$, so each one contains about $\Theta(B/2)$ points.

We'll examine rectangles with sides c^i and a/c^i. Both sides must be less than or equal to N, so i must be between $\log_c \frac{B}{2}$ and $\log_c N$. This gives us about $\log_c \frac{2N}{B}$ different rectangle shapes.

For each shape, we can cover the whole grid D with non-overlapping rectangles, and there are N^2/a rectangles for each shape. So, the total number of queries is $k = \frac{N^2}{a} \log_c \frac{2N}{B}$.

Before we use the redundancy theorem, we need to check how much two queries can overlap. If two rectangles have different shapes, their overlap is at most $B/(16A^2)$ points. If we set $c = 16A^2$, this overlap is small enough.

Now, we apply the redundancy theorem. This gives us:

$$r \geq \frac{1}{12N} \frac{N^2}{a} \log_c \frac{2N}{B} \cdot \Theta\left(\frac{B}{2}\right) = \Theta\left(\frac{1}{12} \log_c \frac{2N}{B}\right)$$

This simplifies to:

$$r = \Omega\left(\frac{\log(N/B)}{\log c}\right)$$

and since c is related to A, we get:

$$r = \Omega\left(\frac{\log(N/B)}{\log A}\right)$$

On a Model of Indexability and Its Bounds for Range Queries

A Lower Bound Theorem for Indexing Schemes and its Application to Multidimensional Range Queries *