Lecture: CPSC 516 Computational Geometry

Instructor: William Evans

Scribe: Seyed Ali Tabatabaee

November 8, 2022

We study the problem of Range Query. Given n points in $\Re^d$, the target is to create a structure that helps us to report the points within a given range for multiple queries. In one dimension, we know there exists a structure with $O(\log(n) + k)$ query time (where k is the number of output points), $O(n \log(n))$ preprocessing time, $O(\log(n))$ update, and $O(n)$ space. In this lecture, we focus on the problem in two dimensions. We investigate three solutions for this problem: kd trees, range trees, and quad trees.

1 kd Trees

The idea for this data structure is to alternatively split points in half vertically and then horizontally until one point remains. It is important to note that we always split through a point in this lecture. Figure 1 provides an example.

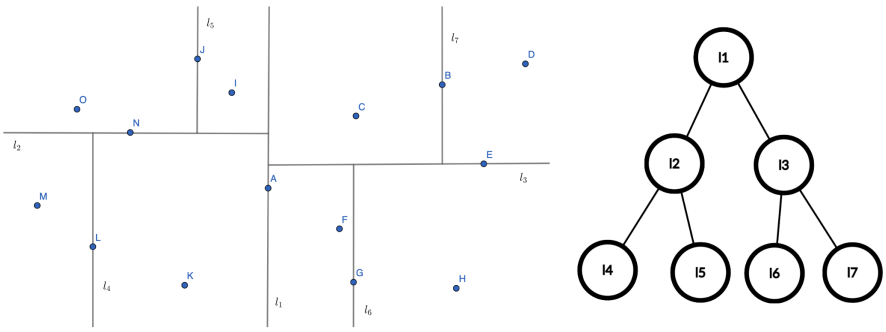

Figure 1: An example for kd trees.

The structure takes $O(n \log(n))$ preprocessing time and $O(n)$ space. Now, we analyze the query time. Let $q(n)$ denote the number of regions in kd tree intersected by a vertical line on n points. We have

$$q(n) = \begin{cases} 1, & \text{if } n = 1. \\ 2 + 2 \cdot q(\frac{n}{4}), & \text{if } n > 1. \end{cases} \tag{1}$$

Therefore, we have $q(n) = O(\sqrt{n})$. Hence, the query time is $O(\sqrt{n} + k)$.

2 Range Trees

For this structure, the idea is to build a balanced binary tree on the x-coordinate (points at leaves sorted by the x-coordinate). Then, at every node v, we store points in $slab(v)$ (x-range of points in subtree v), sorted by the y-coordinate in the array $A(v)$. To answer a query (x_1, x_2, y_1, y_2), we search the tree for x_1 and x_2. We want points in leaves between $l(x_1)$ and $l(x_2)$ that have y-coordinate in $[y1, y2]$ .We perform a binary search in $A(v)$ for all the right children on the path to $l(x_1)$ and all the left children on the path to $l(x_2)$ (See Figure 2).

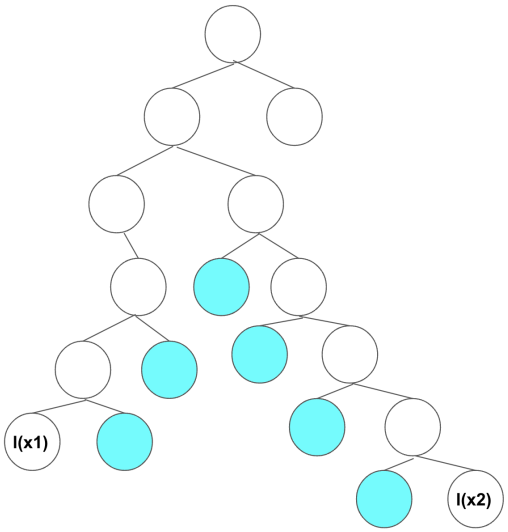

Figure 2: Blue vertices are the ones that we perform a binary search in.

We note that each node is in $A(v)$ for at most one vertex v per level of the tree. Hence, the preprocessing time and space are both $O(n \log(n))$. If we perform a binary search in $A(v)$ for any vertex v, the running time is $O(\log(n) + OUTPUT)$. Considering that we perform a binary search in $O(\log(n)$ vertices, the total running time of the algorithm is $O(\log^2(n) + k)$.

2.1 Fractional Cascading

In the range tree structure, for every non-leaf node w, $A(w)$ contains $A(w.l)$ and $A(w.r)$. Therefore, by adding the appropriate pointers (Figure 3), search in parent gives the location of the same search in children in $O(1)$. Therefore, we only need to perform the complete search in the root and hence, the cost of each query would be $O(\log(n) + k)$.

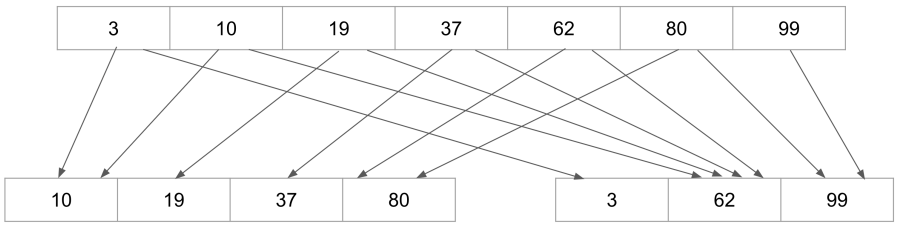

Figure 3: Fractional cascading.

3 Quad Trees

In this structure, all the points are initially in a big square. At every step, if there are more than one point in a square, we split the into four squares of equal size. Figure 4 provides an example.

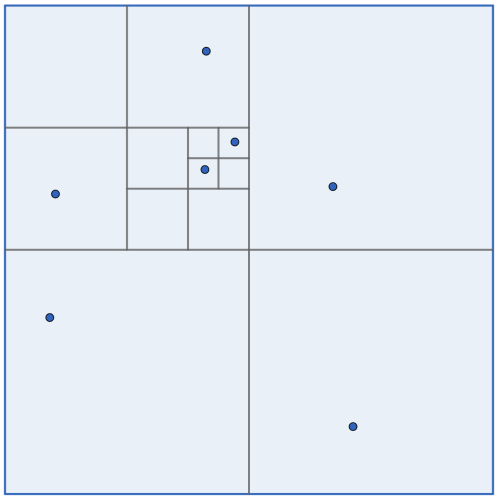

Figure 4: An example for quad trees.

We prove that the depth of the tree is less that or equal to $\log\left(\frac{s}{c}\right) + \frac{3}{2}$ where c is the smallest distance between points and s is the size of the initial square. For every $i \geq 0$, the square side length at depth i is $\frac{s}{2^i}$. The maximum distance in such a square is $\frac{\sqrt{2} \cdot s}{2^i}$. For all internal nodes at depth i, we have $\frac{\sqrt{2} \cdot s}{2^i} \geq c$. Hence, $i \leq \log\left(\frac{s\sqrt{2}}{c}\right) = \log\left(\frac{s}{c}\right) + \frac{1}{2}$. Hence, the depth of the tree is less that or equal to $\log\left(\frac{s}{c}\right) + \frac{3}{2}$.

3.1 Balanced Quad Trees

We first saw the slides from a presentation on terrain approximation by the instructor, Dr. Will Evans. The slides are available on the webpage of the course. This provided motivation for why we need balanced quad trees.

In balanced quad trees, the diference between adjacent cell sizes is at most a factor of 2.