CSL 356 Lecture 27 Oct 3

Range Searching → contd.

K-d trees. (Complete Binary Tree)

1. Each node contains information related to

   (i) the enclosing rectangle that it represents

   canonical rectangles

   (ii) pointer to the leaf nodes in its subtree

   (iii) a count of all points in subtree

   (iv) A horizontal/vertical line corresponding to whether its children are obtained using horizontal/vertical split

2. The leaf nodes correspond to each of the given points in P and they are linked in a chain from left to right

To answer a query rectange R, we intersect it with the current node η (starting with root)

<u>Case 1</u> : It completely contains the $R(\eta)$: the rectange corresp to η

Then report/count all points in the subtree

<u>Case 2</u> : No intersection with $R(\eta)$

then no points are reported

<u>Case 3</u> : Partial overlap Then split R into two subrectangles R^1 and R^2 (either horizontally or vertically)

and call Search(R^1), Search(R^2)

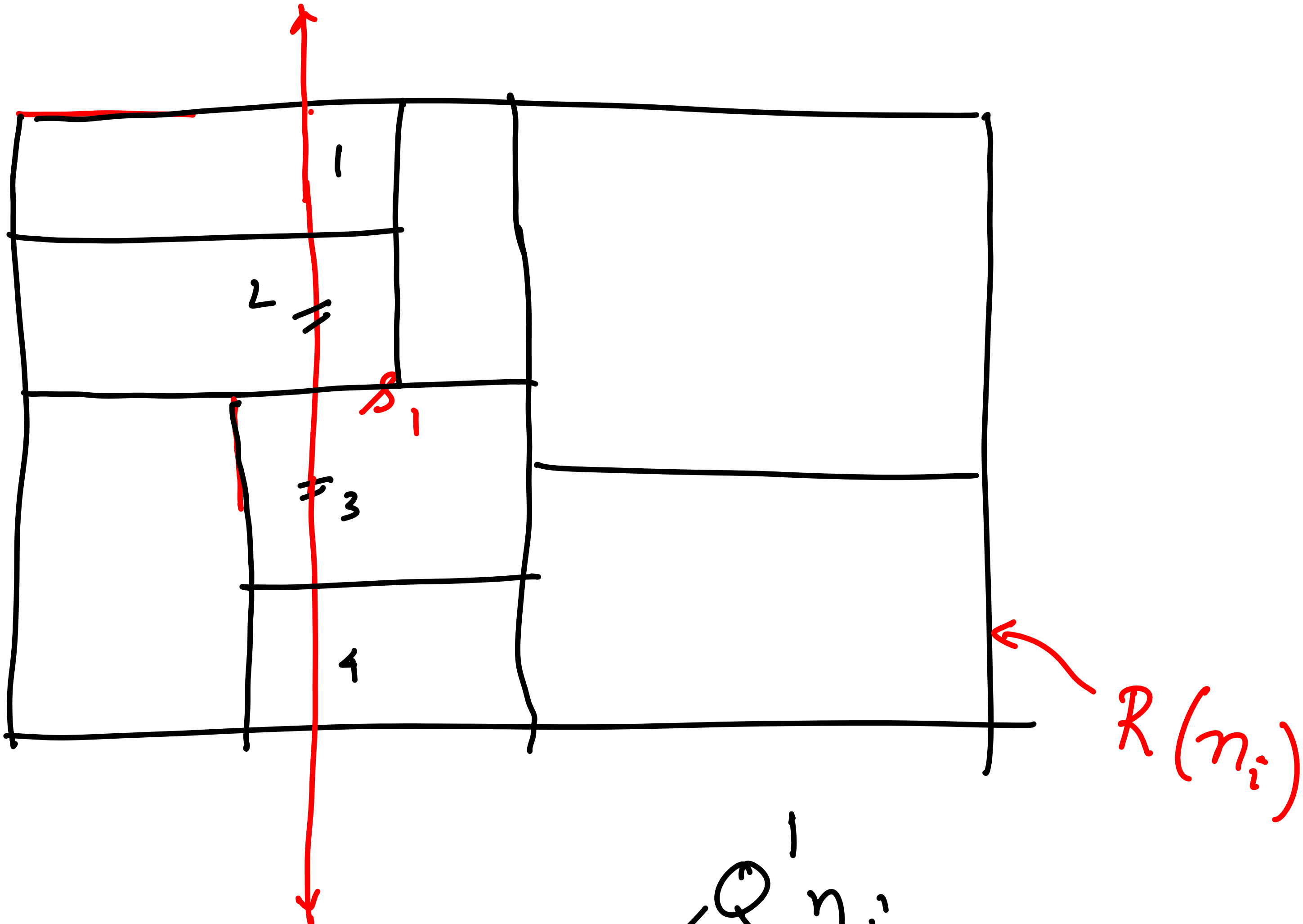
Search Time on a k-d tree (2 dim).

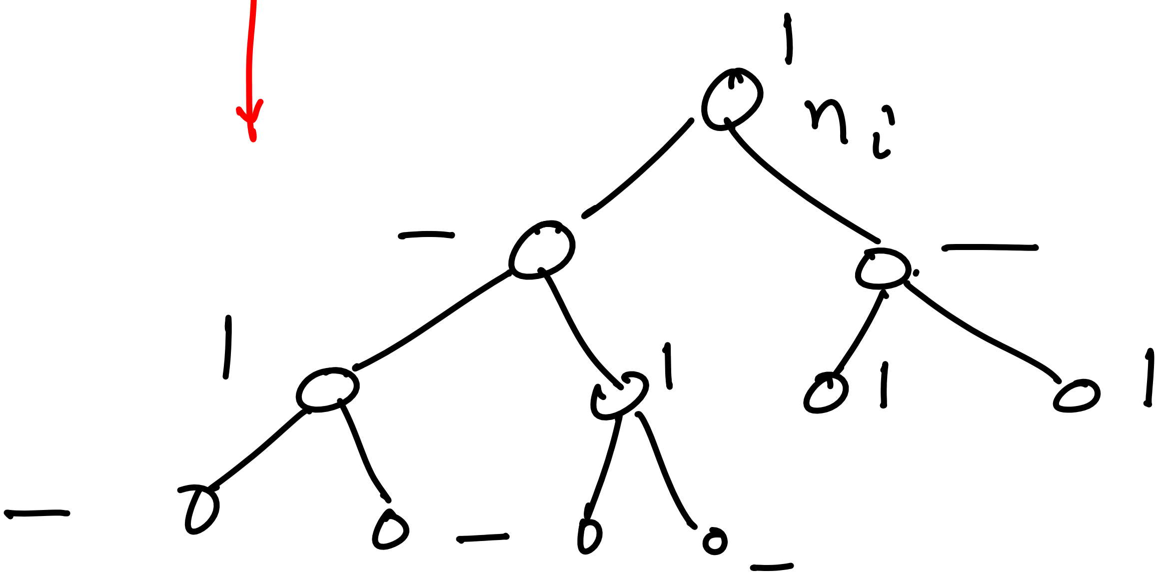
Consider the 4 sides of the query rectangle R, say s_1, s_2, s_3, s_4

We will count the max no. of partitioning lines intersected by s_1 (also for s_2, s_3, s_4)

How many partitioning lines does s_1 intersect

Assume wlog s_1 is vertical, so we count the # of horizontal part lines





Let $A(j)$ be the # of subsegments of s_1 in level j of the k-d-tree

Then $A(j+2) \leq 2 \cdot A(j)$

$\Rightarrow A(i) \leq 2^{i/2}$ $\qquad A(0) = 1$

root node

$$A(\log n) \leq 2^{\log n/2} = \sqrt{n}$$

$\Rightarrow$ We can visit atmost $4\sqrt{n}$ nodes of the k-d tree

So query time is $O(\sqrt{n})$

In d dimensions the k-d tree has a query time

$$O\left(n^{1-\frac{1}{d}}\right) \text{ for } d>1$$

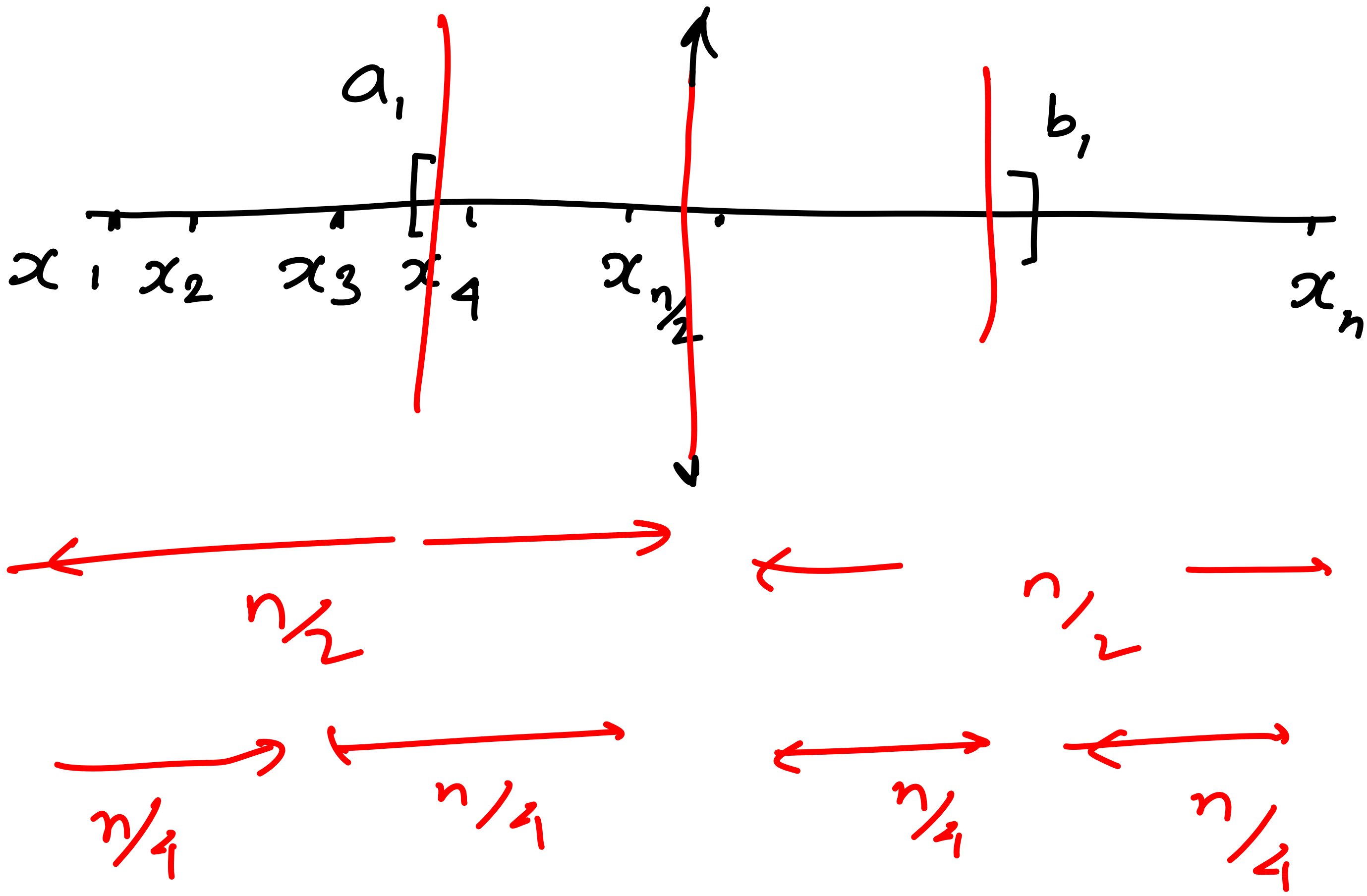
d : dim.

Preprocessing $O(n \log n)$

Range Searching

Can we do better?

K-d tree answers query by union of subrectangles (disjoint)



How many subintervals does $[a_1, b_1]$ get split into?

Any interval can be expressed as union of atmost $2 \log n$ canonical intervals

The generic technique for range searching involves splitting the point set P into "canonical partition" ← related to space

The answers are precomputed for canonical partition

The query rectangle is expressed (search process) as union of canonical partitions (the fewer the better)

search time