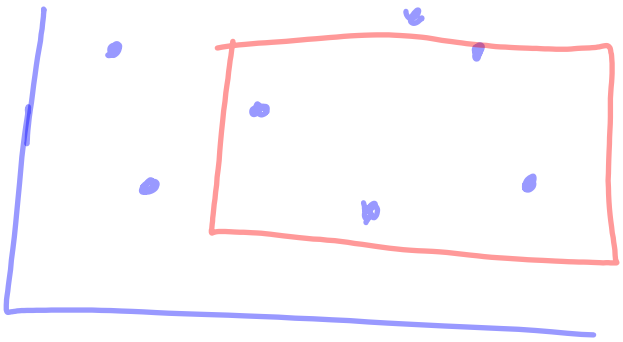
Today we will discuss **orthogonal range searching** in which we are given n points in a d dimensional geometric space.

We will perform queries with an axis-aligned box (a rectangle in 2D), and we wish to answer questions regarding the points contained inside of the box.

- How many points are in the box?
- List all of the points in the box.

This is an important application in databases. Suppose we have a database with n records each with d numeric fields. Each record will be represented as a point in the d dimensional geometric space, and value assigned to field i is the coordinate of the point in the ith dimension.

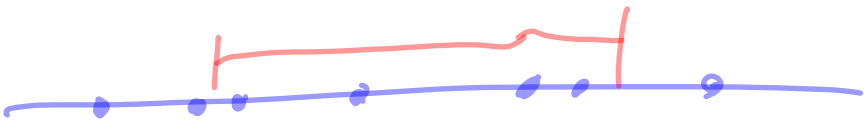


We want to maintain a data structure which will allow us to answer queries quickly.

We will consider a *static data structure* (i.e. we will not add or delete points from our data structure).

We are allowed to spend time preprocessing our input (i.e. the preprocessing time is one time operation and so it can be more expensive than the query time).

Consider the problem in 1D (all of the points are along a line).



Since we are allowed preprocessing time, we can sort the points according to their left to right ordering in $O(n \log n)$ time.

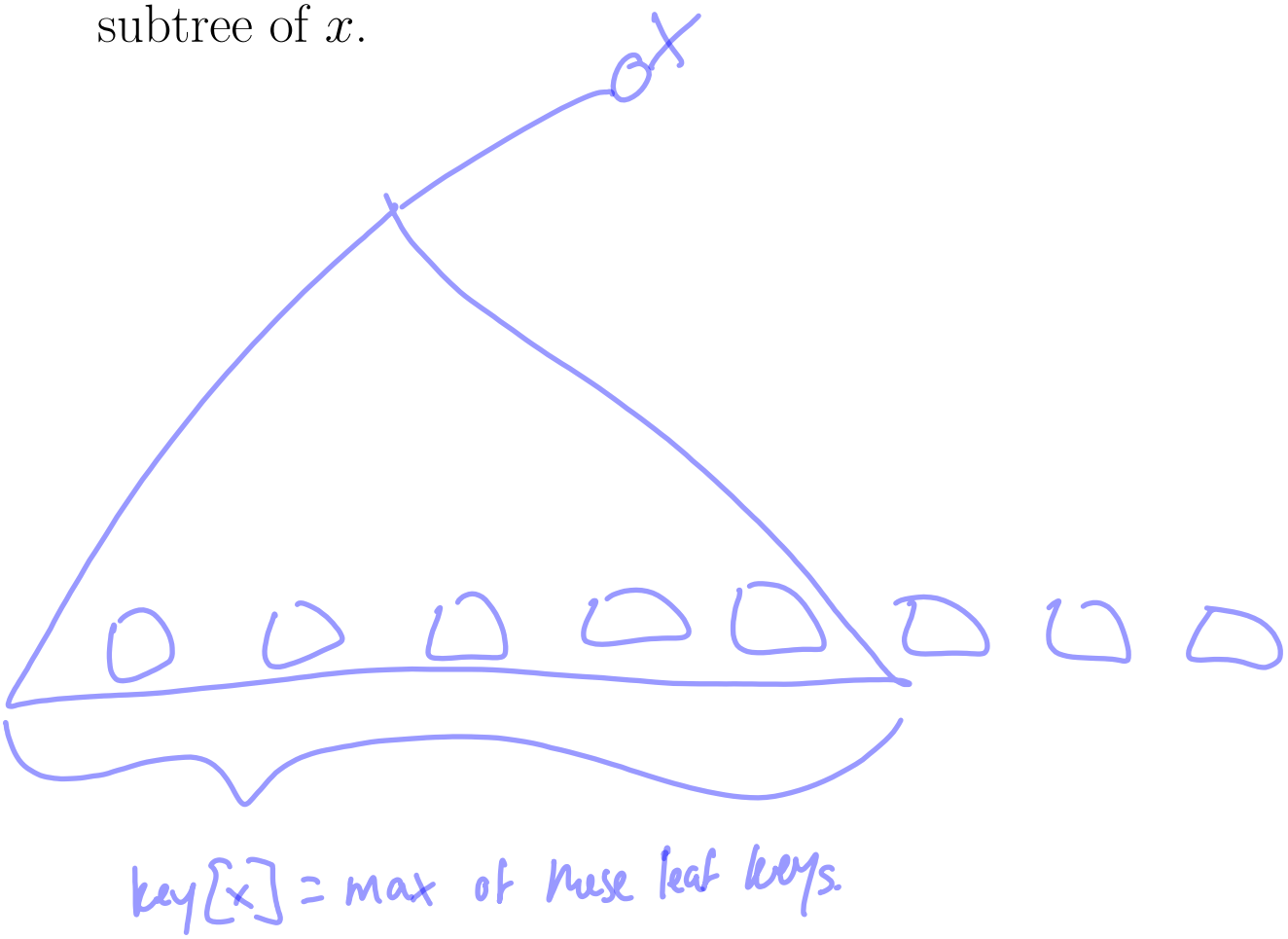
- Since the points are sorted, we can find the first point by binary search in $O(\log n)$ time.
- If there are k points in the query range, we can step through the sorted list and report all of the k points in $O(k)$ time.
- Total running time is $O(\log n + k)$.

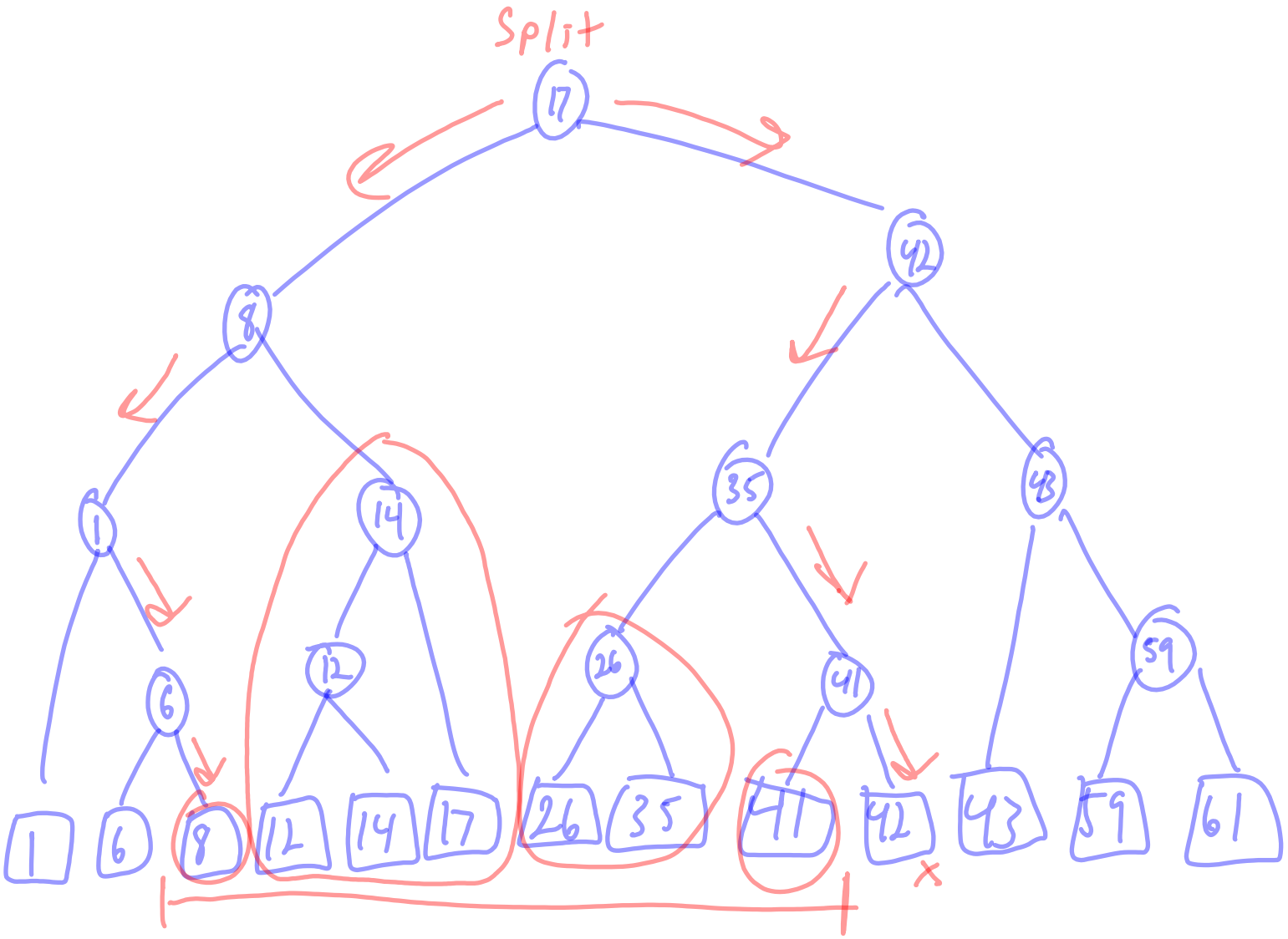
Nice simple idea, but it does not generalize to higher dimensions, and the ultimate goal is to get something that works in any d dimensional space.

Another approach to the 1D problem: use a search tree.

Store each of the points in the *leaves* of the tree.

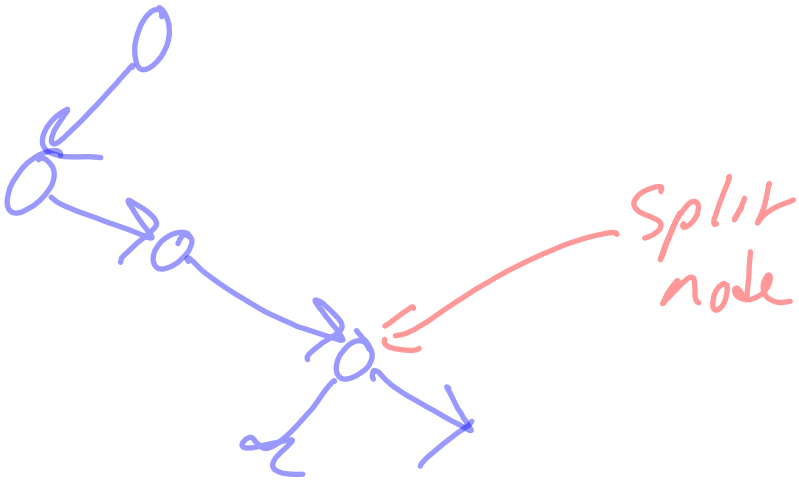
To help us search the tree, each internal node x stores $key[x]$ which is the maximum key of any leaf in the left subtree of x.





Query [7, 41]

Split node: first node we find whose key is in our query range.



We need to traverse $O(\log n)$ subtrees.

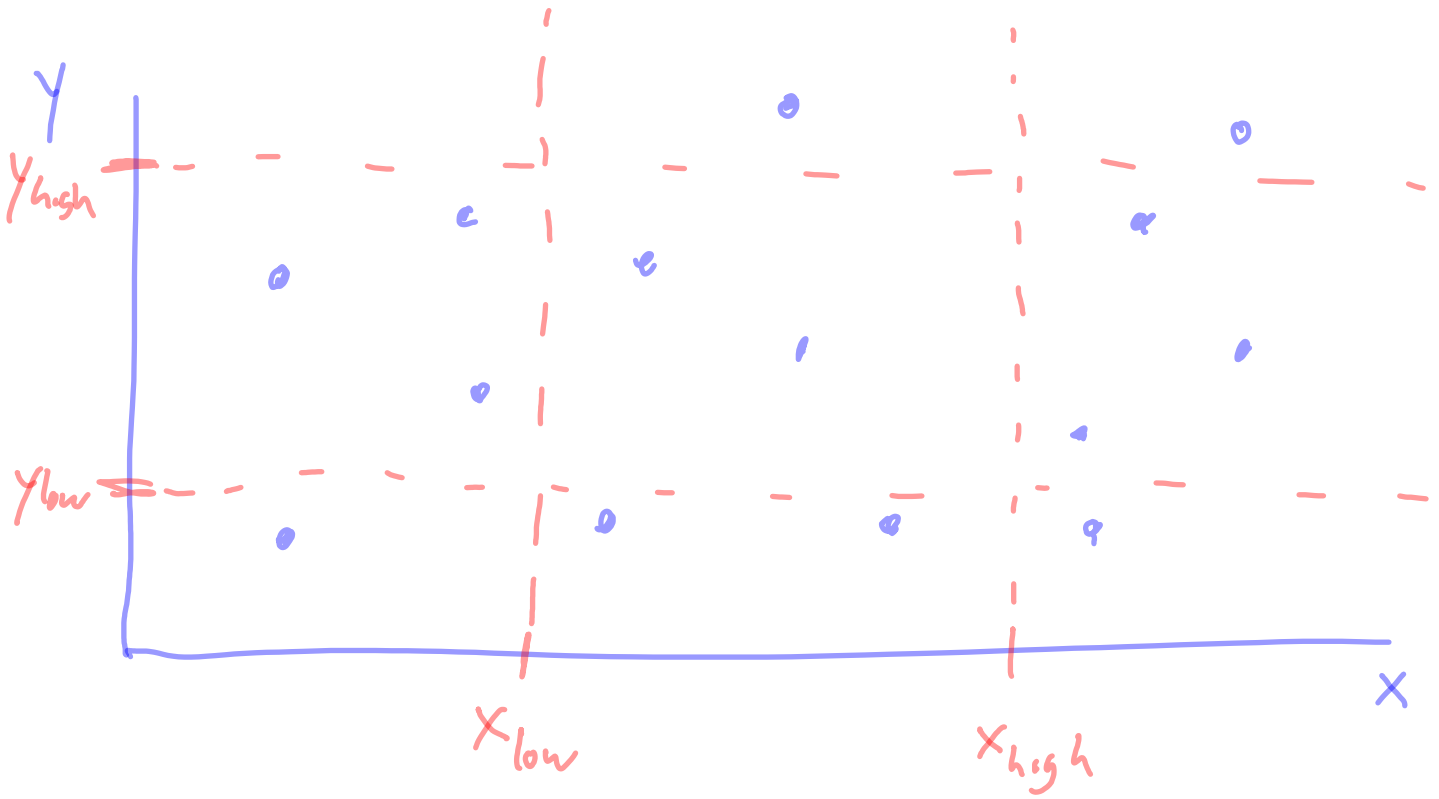
We report k points found in $O(k)$ time

Total Running time $O(\log n + k)$

- Same as Sorting technique

Extending to 2D

Query is 2 intervals: $[x_{low}, x_{high}]$, $[y_{low}, y_{high}]$



We search a "primary" range tree based on x-coordinate. For each point that "survives" this query, we then search in a "secondary" range tree based on y-coordinate.

In 2D, we query in $O(\log^2 n + k)$ time.

In d-dimensions, we query in $O(\log^d n + k)$ time.