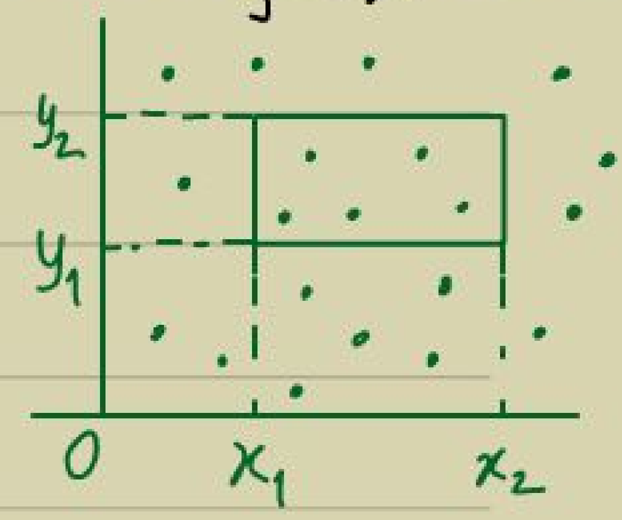
- Another example of augmented AVL tree:

Orthogonal Range Searching

- Input: A set of points $T \subset \mathbb{R}^2$ & a rectangle $(x_1, y_1, x_2, y_2) =: R$.

Output: All points in the rectangle, i.e. $T \cap R$.



- Brute-force: It can be solved in $O(n)$ time.

  Qn: Can it be done significantly faster?

  Let $k := |T \cap R|$.

- Easier question: Find the points on the line (x_1, x_2)?
  - Store the points in an AVL tree wrt the x-coordinate.
  - Search x_1, x_2 in T & find the least common ancestor (lca) y.

▷ In the tree T the blue shaded nodes are exactly the points in $[x_1, x_2]$.

▷ If $|T \cap [x_1, x_2]| =: \ell$ then these points can be found in $O(\ell + \lg n)$ time.

- Next, how do we find the points in $T \cap R$?

- Ans: Augment each node v by adding another copy of tree(v) as: An AVL tree organized wrt y-coordinates.

Call this Ytree(v).

- This inspires the following pseudocode for RangeSearch(T, x_1, x_2, y_1, y_2):
  - For root v of each blue shaded subtree {
    Do RangeSearch(Ytree(v), y_1, y_2) }
  - For the other blue vertices v on the

We may make $O(\lg n)$ such calls.

search path: Check whether $v \in ? T \cap R$.

- Output the ones found in $T \cap R$.

▷ Orthogonal Range Search can be done in $O(k + \lg^2 n)$ time.

Pf: (Exercise) $\square$

- Note that preprocessing time taken is $O(n \lg n)$.

But, the query time is significantly lower!

- Note that the space required by the augmented AVL tree is $\approx$

$$\sum_{v \in T} |\text{tree}(v)| = \sum_{u \in T} \#(v \in T : v \text{ is an ancestor of } u)$$

$$\leq |T| \cdot \text{depth}(T)$$

$$= O(n \lg n).$$