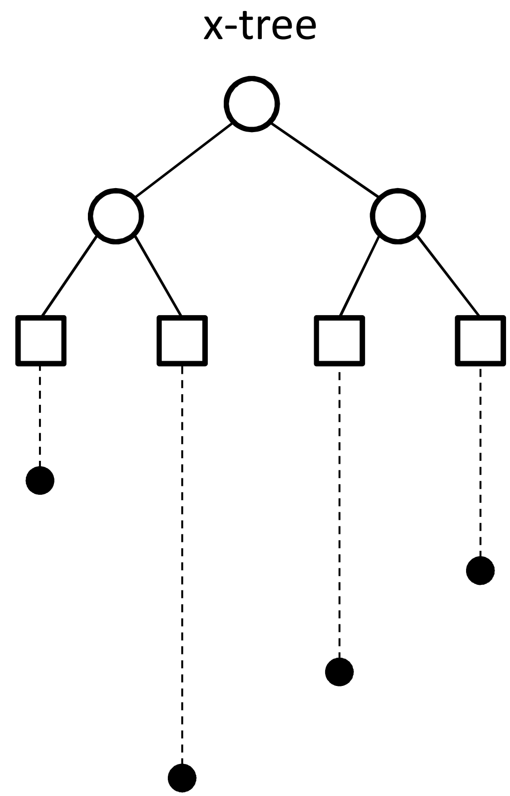
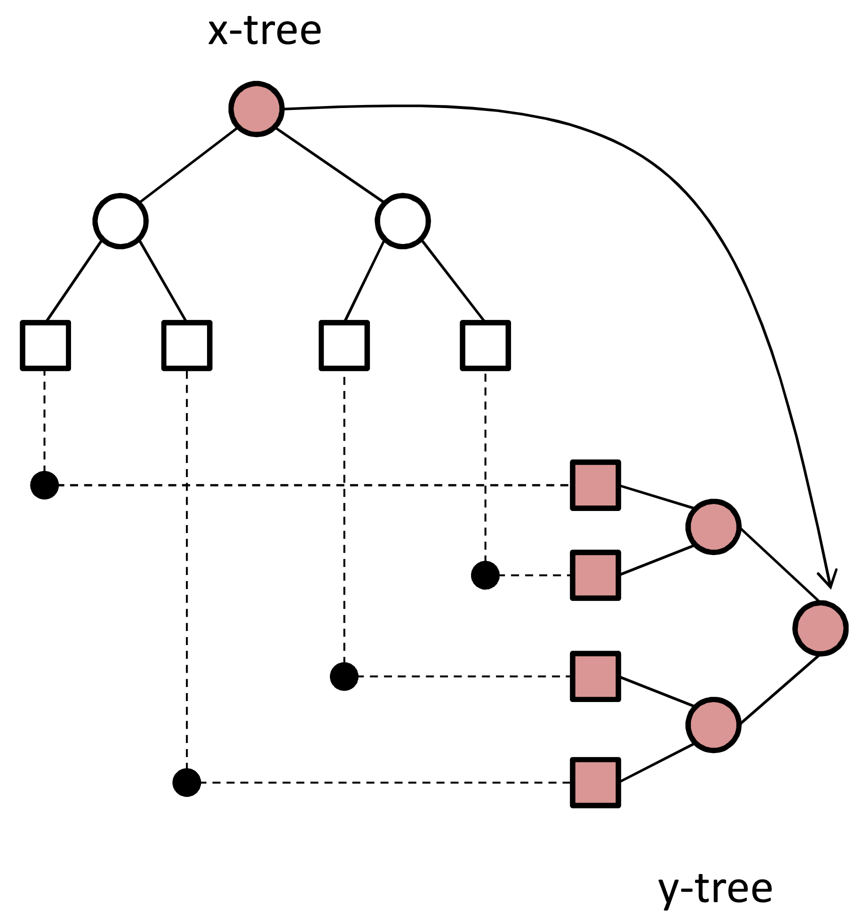
Range Tree

- One binary tree in X (x-tree)



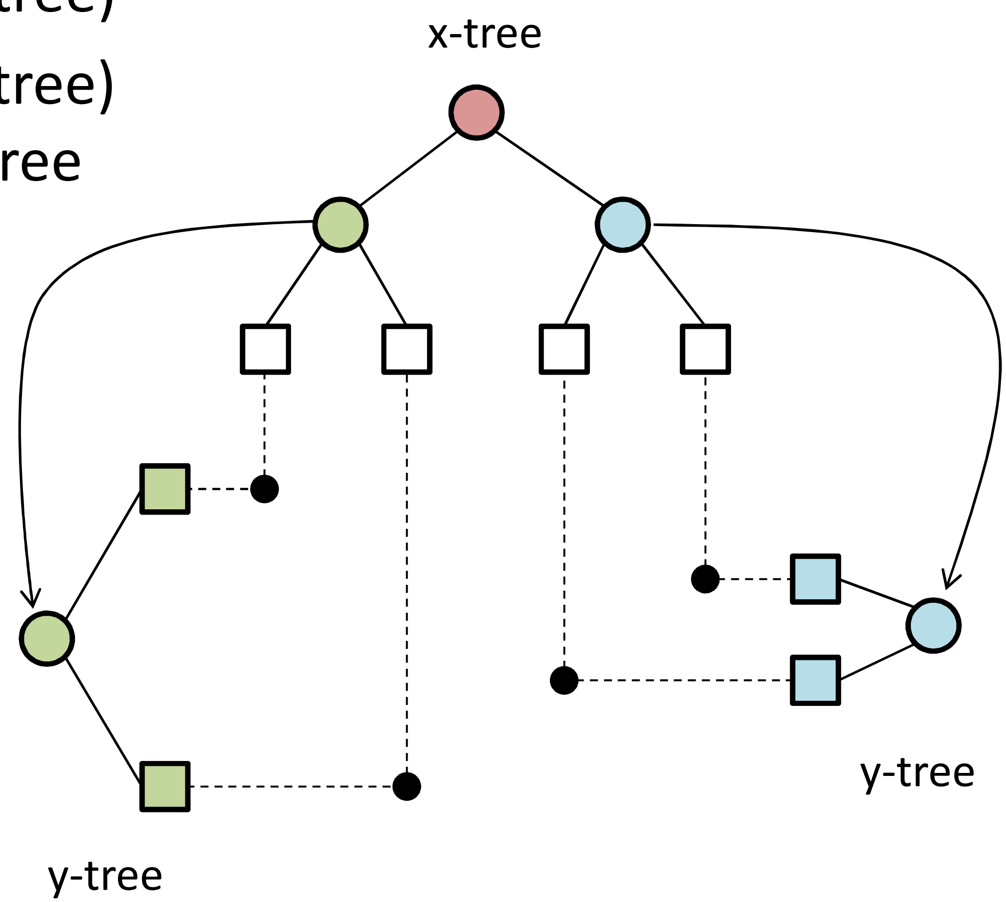
Range Tree

- One binary tree in X (x-tree)
- One binary tree in Y (y-tree) for each node in the x-tree



Range Tree

- One binary tree in X (x-tree)
- One binary tree in Y (y-tree) for each node in the x-tree

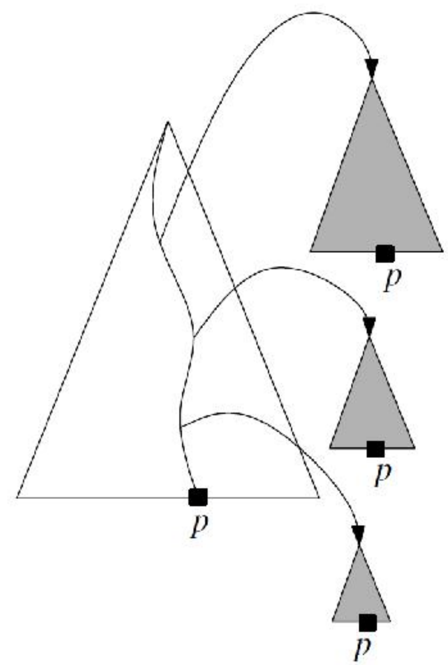


Range Tree

- Space complexity:
  - Size of each tree (x- or y-) is linear to # of leaves
  - Let T_i be # of trees of which p_i is a leaf, total space is

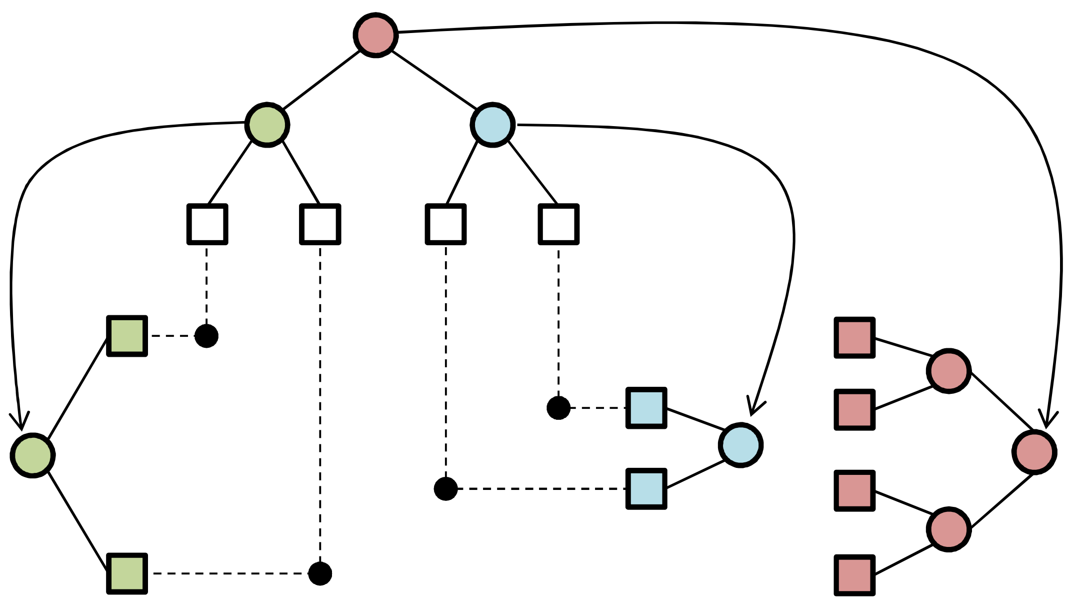
$$O\left(\sum_{i=1}^{n} T_i\right)$$

  - $T_i = O(\log n)$
  - Total space is $O(n \log n)$



Range Tree

How to build it?



Range Tree

- If t is a node of x-tree:
  - t.val: cut value
  - t.left, t.right: child
  - t.ytree: y-tree
- If t is a leaf of x-tree:
  - t.pt: point
  - t.ytree: a y-tree with a single point

BuildXTree (S)   //S: point set

1. If |S|=1, return leaf t where
   1. t.pt and t.ytree are the point of S

O(n)

2. x be median of X coordinates of all points in S
3. L (R) be subset of S whose X coordinates are no greater than (greater than) x
4. Return node t where
   1. t.val = x

   2T(n/2)

   2. t.left = BuildXTree (L)
   3. t.right = BuildXTree (R)

   O(n)

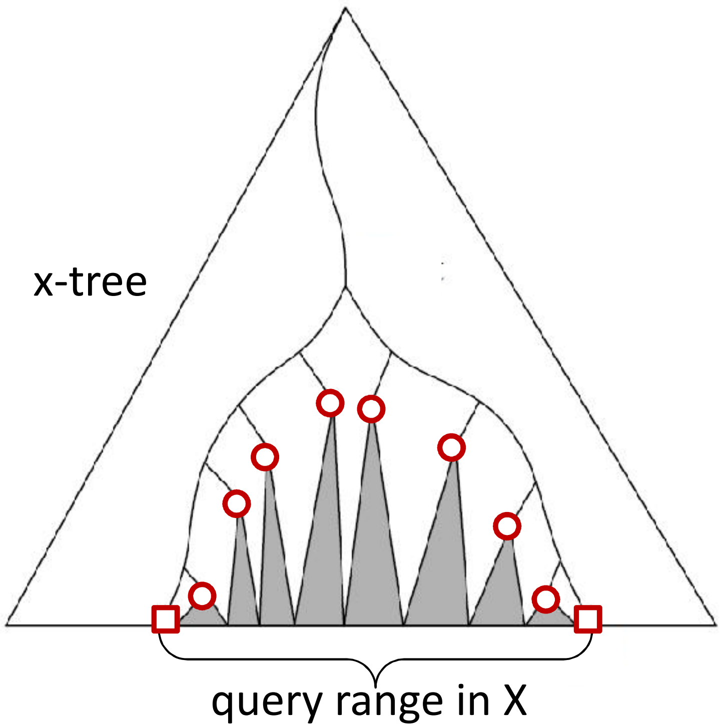
   4. t.ytree = MergeYTree (t.left.ytree, t.right.ytree)

$T(n) = O(n) + 2T(n/2)$
$= O(n \log n)$

Range Tree

- Space complexity: O(n Log n)
- Building time: O(n Log n)

Query a range Tree



Complexity of QueryY(): $O(\text{Log } n_t + k_t)$
Query() calls: O (Log n)
Total complexity: $O(log^2 n + k)$

Query (t, rX, rY)
//rX, rY: query range in X and Y
1. If t is a leaf
    1. If t.pt is inside {rX,rY}, return t.pt
    2. Else return NULL
2. If t.range is inside rX
    1. QueryY (t.ytree, rY)
3. Else if t.range intersects rX
    1. Return Query (t.left, rX, rY) ∪ Query (t.right, rX, rY)

1D range query

Can be improved to O(log n + k)
(using *fractional cascading*, see book/note)