Computational Geometry
[csci 3250]

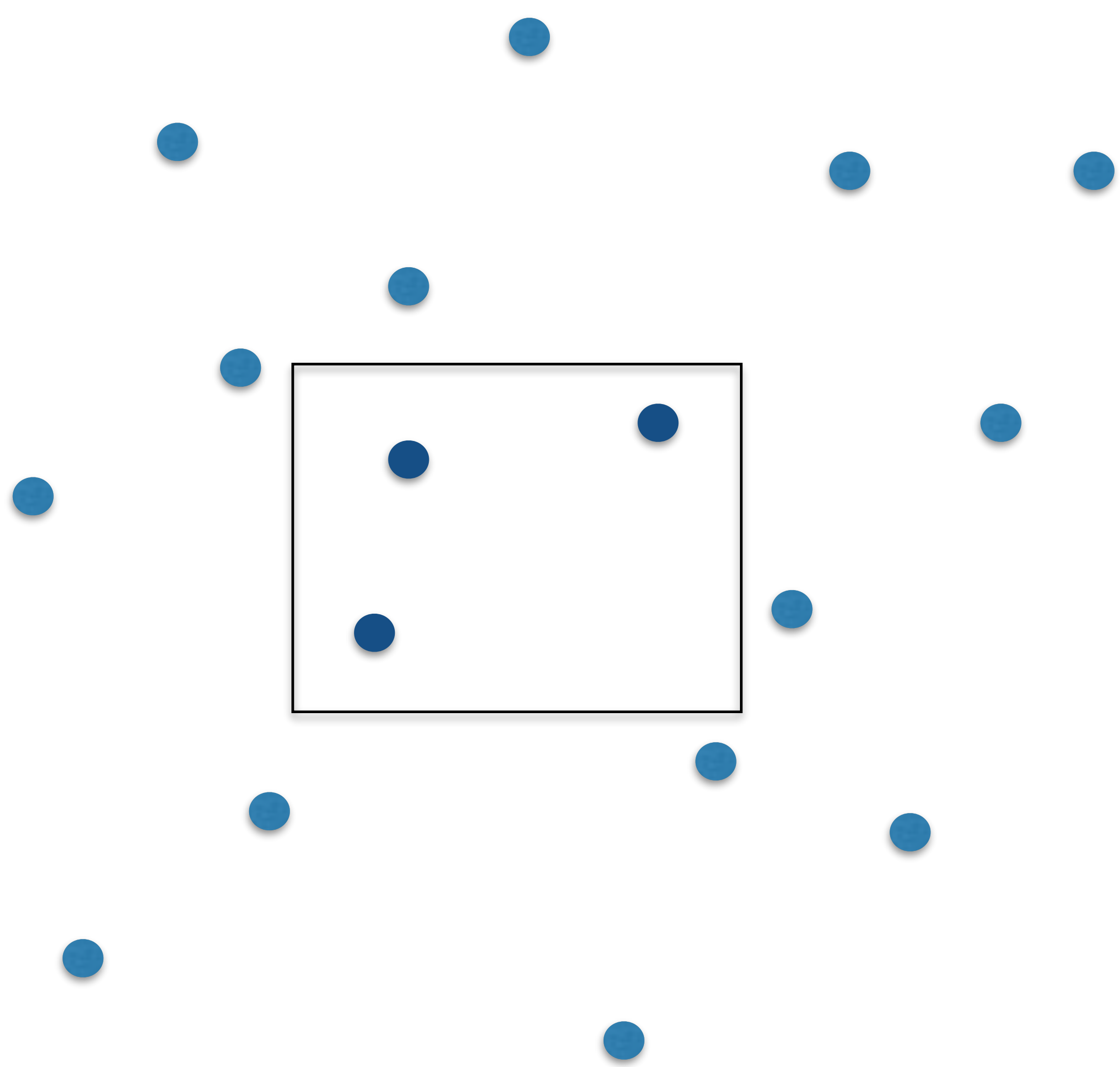
Laura Toma

Bowdoin College

Range searching with Range Trees

Range searching

Given a set of points, preprocess them into a data structure to support fast range queries.



2D

1D

- BBST
  - Build: O(n lg n)
  - Space: O(n)
  - Range queries: O(lg n +k)

2D

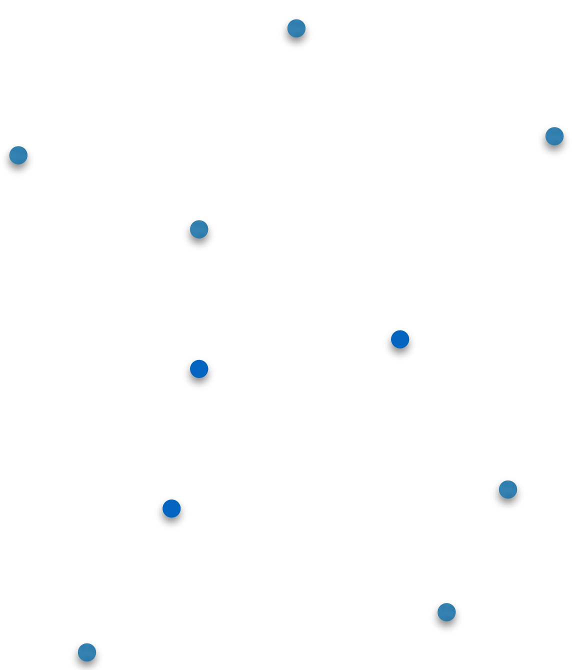
- kd-trees
  - Build: O(n lg n)
  - Space: O(n)
  - Range queries: $O(\sqrt{n} + k)$

- Range trees
  - Build: O(n lg n)
  - Space: O(n lg n)
  - Range queries: O(lg n + k)

Different trade-offs!

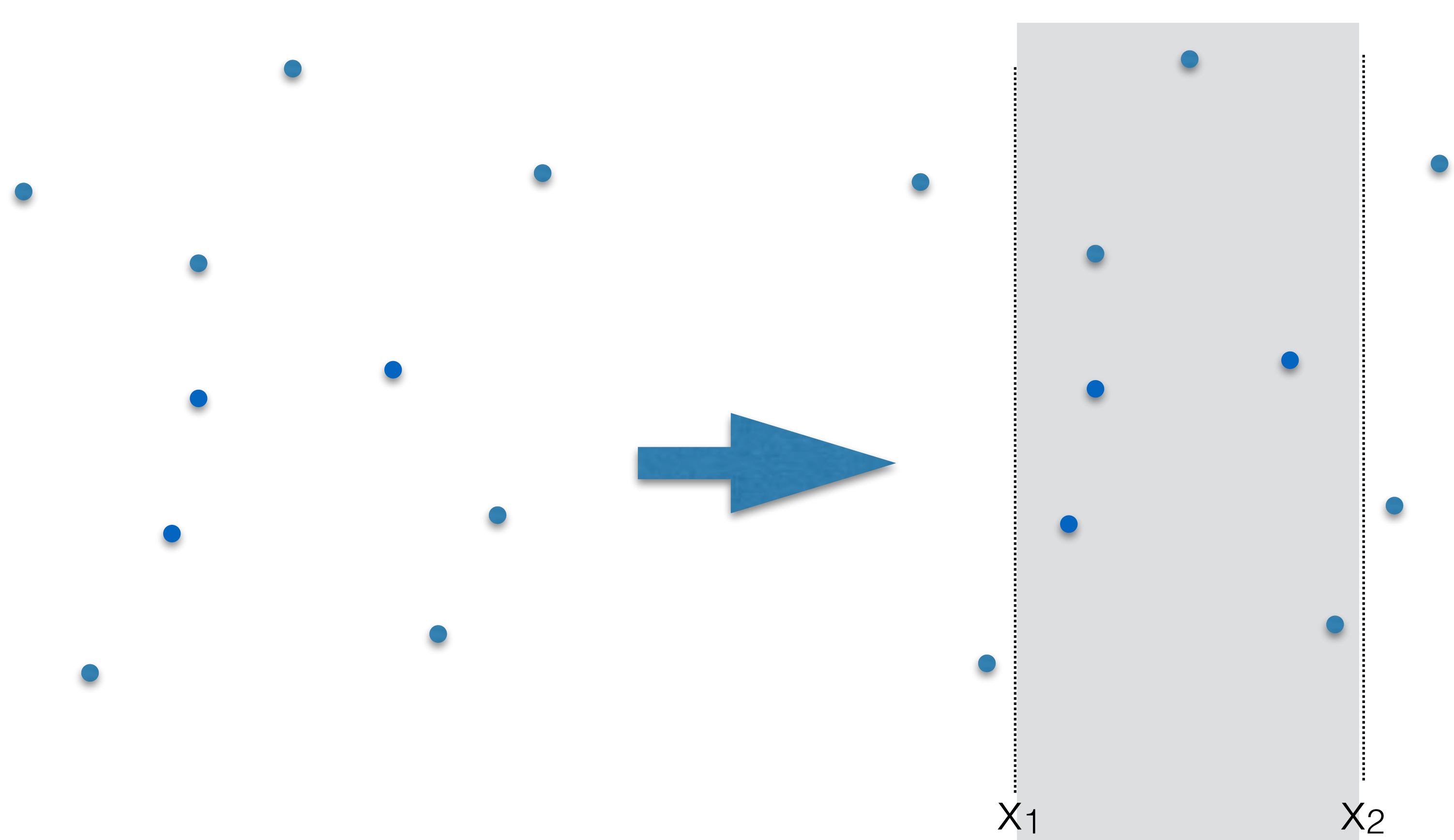
Towards 2D Range Trees

- Denote query $[x_1, x_2] \times [y_1, y_2]$
- Idea
  - Find all points with x-coordinates in the correct range $[x_1, x_2]$
  - Of all these points, find all points with y-coord in the correct range $[y_1, y_2]$



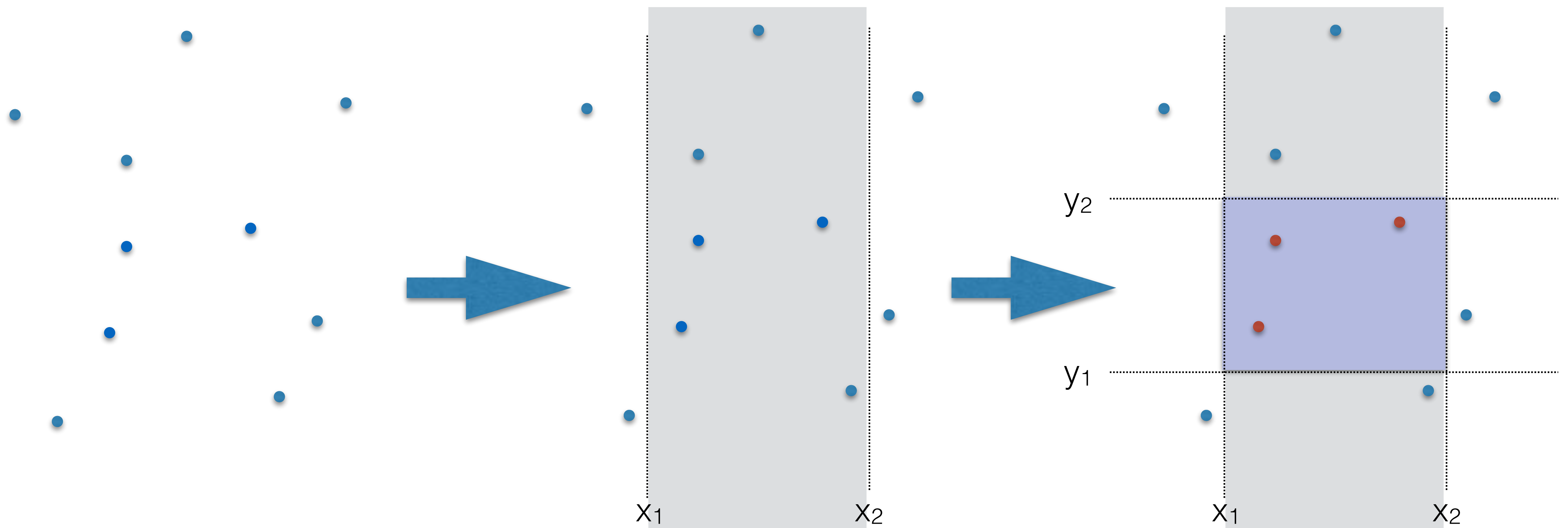
Towards 2D Range Trees

- Denote query $[x_1, x_2] \times [y_1, y_2]$
- Idea
    - Find all points with x-coordinates in the correct range $[x_1, x_2]$
    - Of all these points, find all points with y-coord in the correct range $[y_1, y_2]$



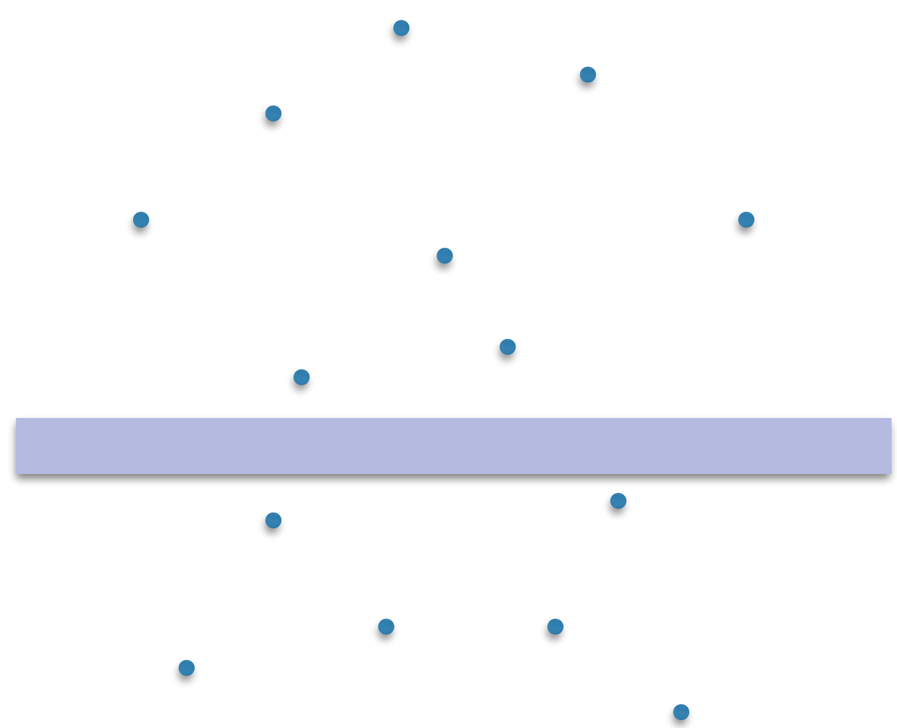
Towards 2D Range Trees

- Denote query $[x_1, x_2] \times [y_1, y_2]$
- Idea
  - Find all points with x-coordinates in the correct range $[x_1, x_2]$
  - Of all these points, find all points with y-coord in the correct range $[y_1, y_2]$



Towards 2D Range Trees

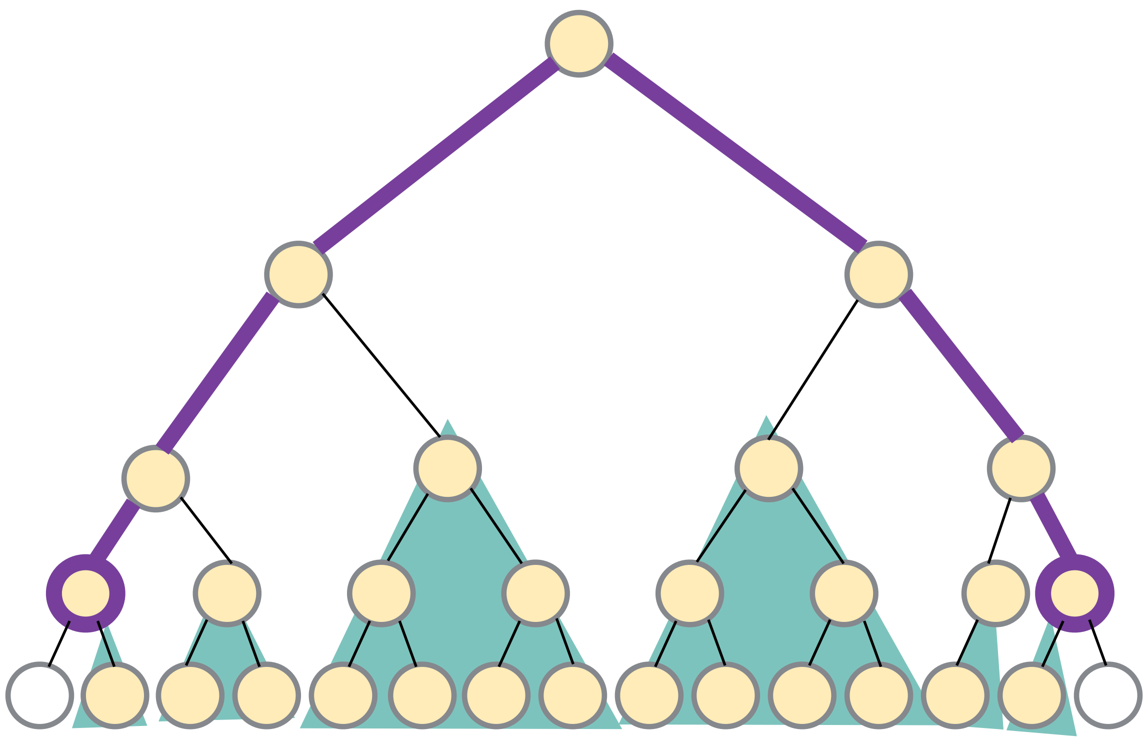
- Store points in a BBST by x-coord
- Range queries:
    - Use BBST to find all points with the x-coordinates in $[x_1, x_2]$: O(lg n + n')
    - Of all these points, find all points with y-coord in $[y_1, y_2]$: O(n')



Works, but slow. n' = O(n)

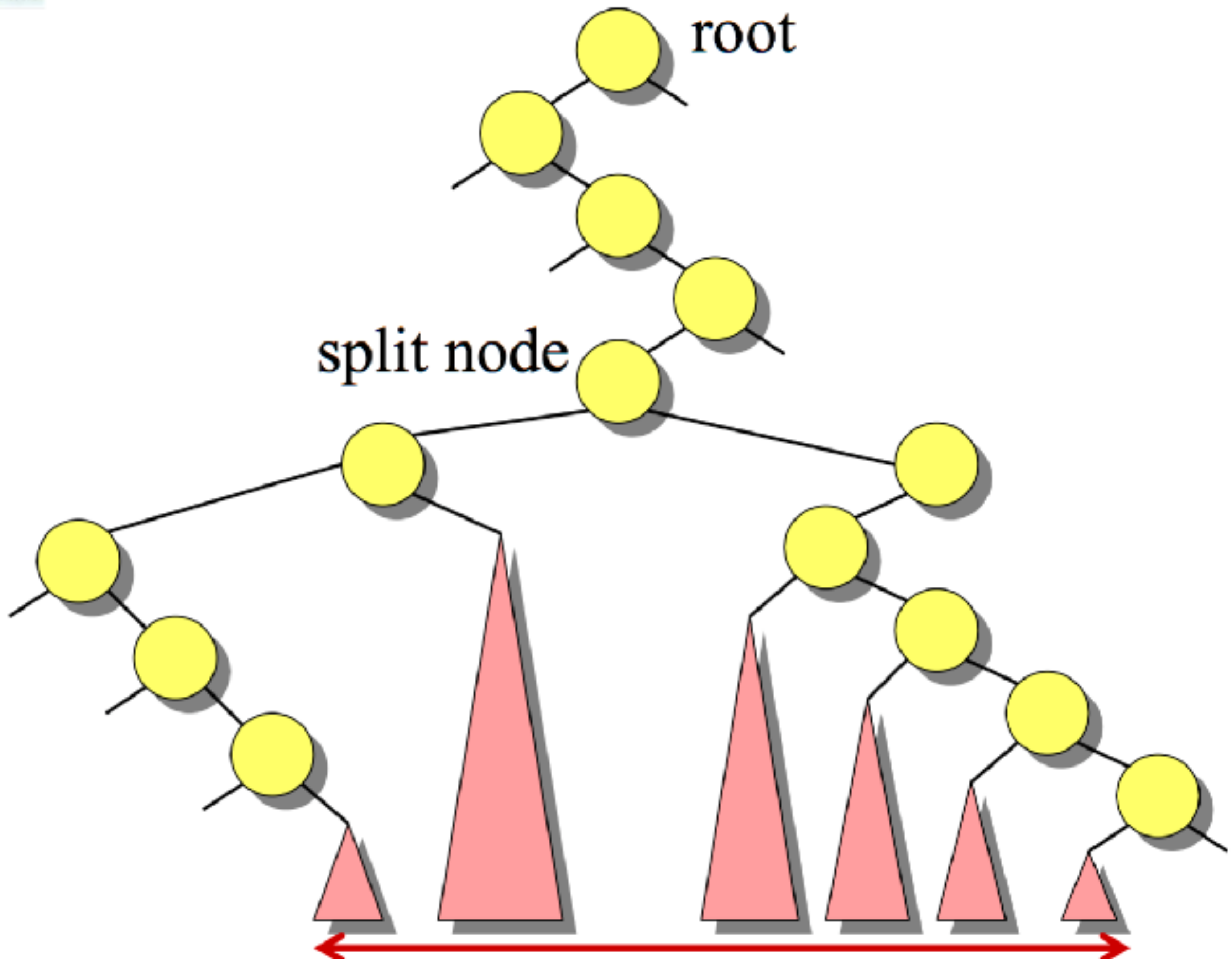
Towards 2D Range Trees

- Store points in a BBST by x-coord
- Range queries:
  - Use BBST to find all points with the x-coordinates in $[x_1, x_2]$
  - Of all these points, find all points with y-coord in $[y_1, y_2]$

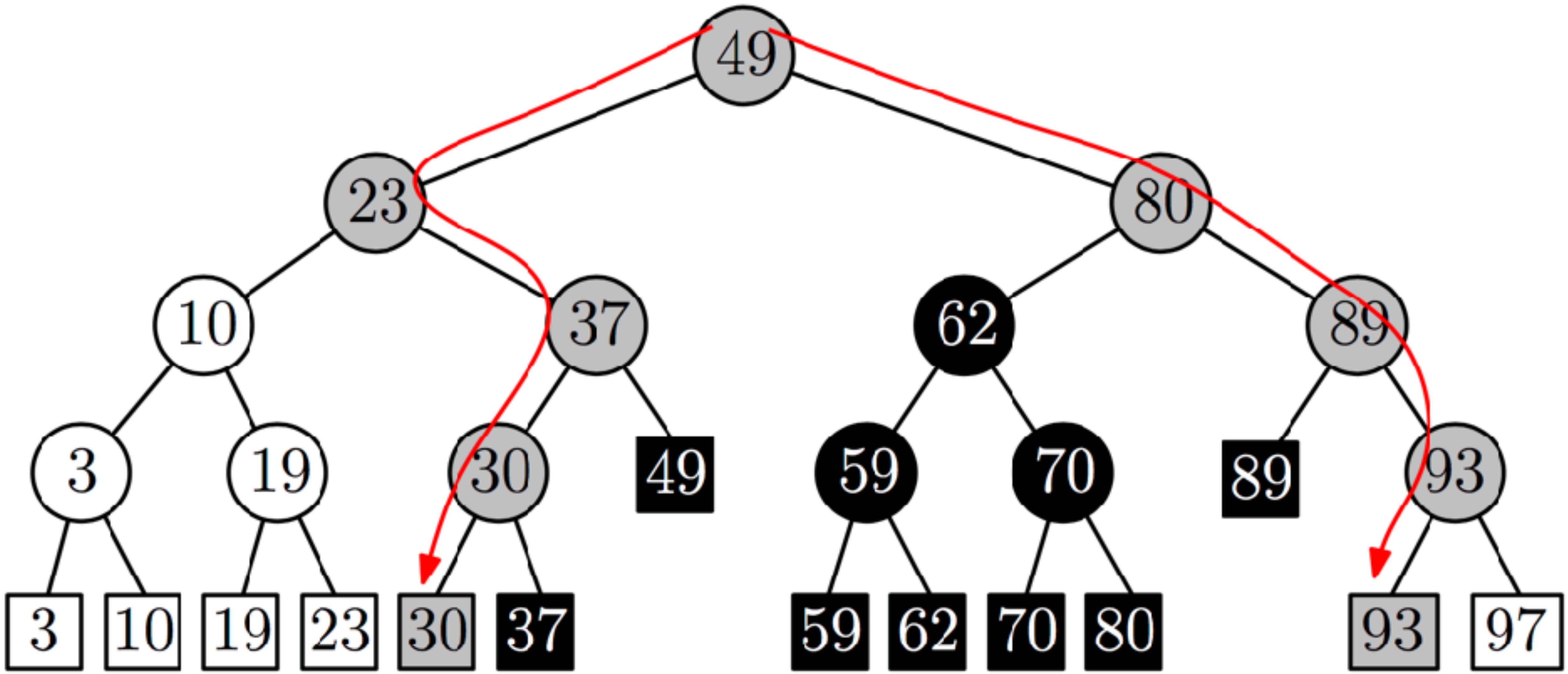


They are sitting in O(lg n) subtrees

General 1D range query



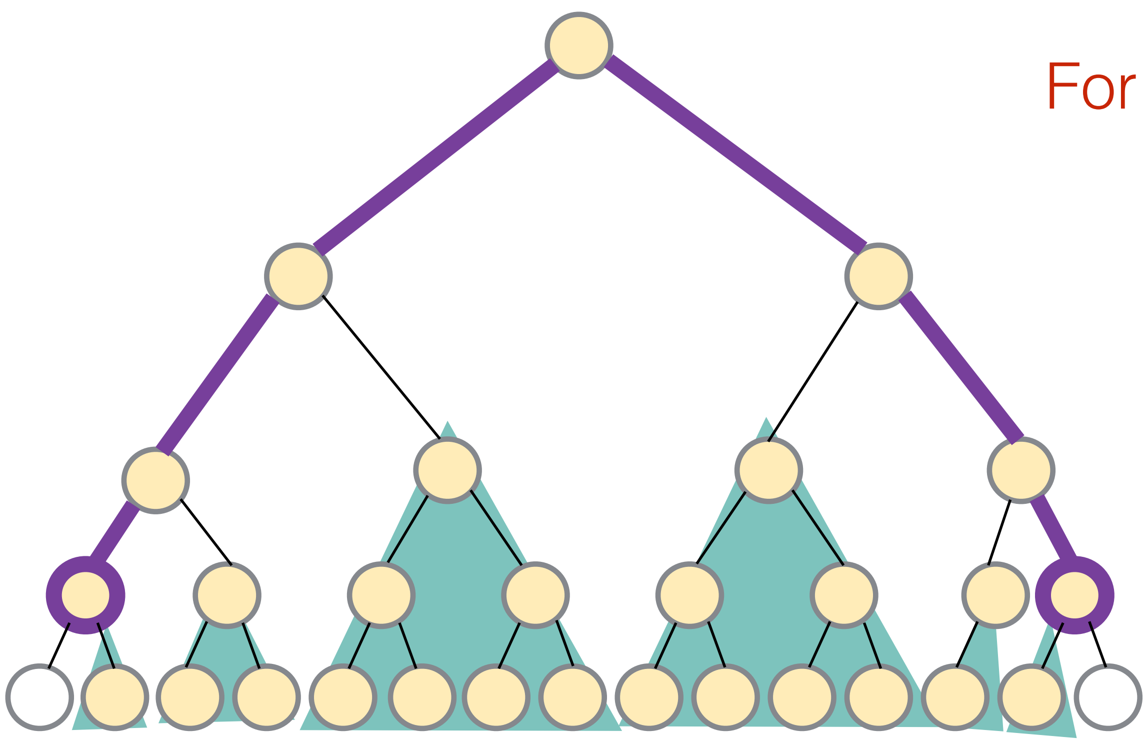
A 1-dimensional range query with $[25, 90]$



screen shot from Mark van Kreveld slides, http://www.cs.uu.nl/docs/vakken/ga/slides5b.pdf)

Towards 2D Range Trees

- Store points in a BBST by x-coord
- Range queries:
    - Use BBST to find all points with the x-coordinates in $[x_1, x_2]$
    - Of all these points, find all points with y-coord in $[y_1, y_2]$

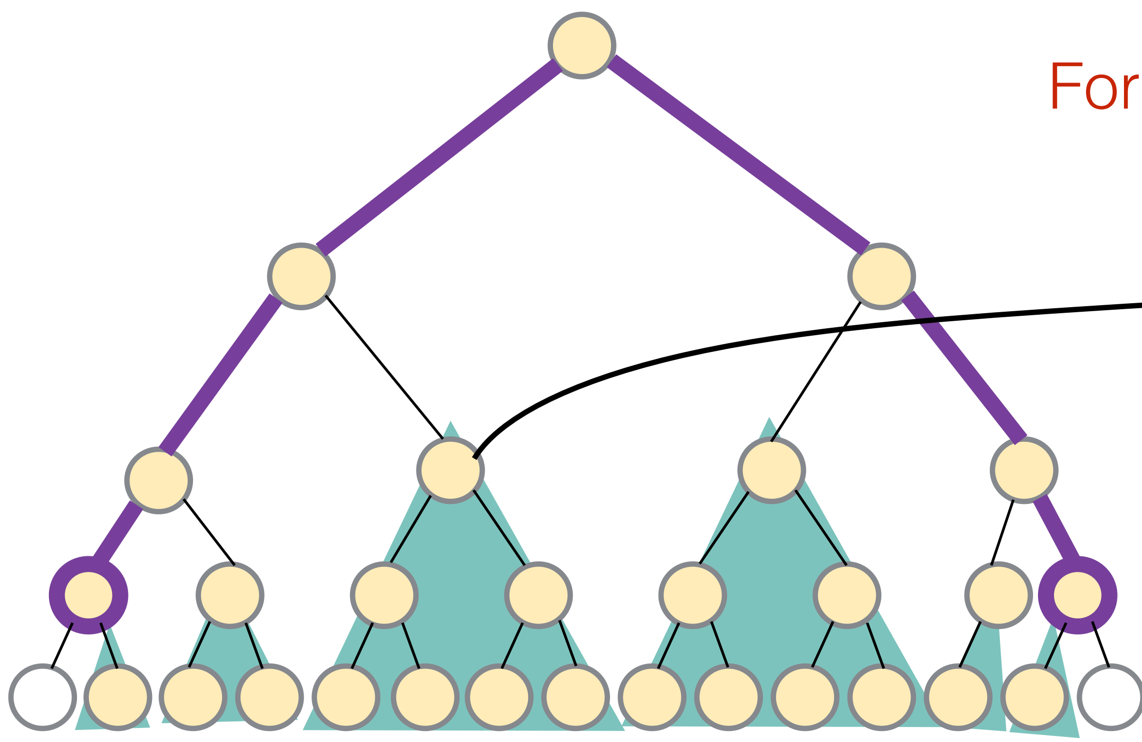


For each subtree we need all points in $[y_1, y_2]$

They are sitting in O(lg n) subtrees

Towards 2D Range Trees

- Store points in a BBST by x-coord
- Range queries:
    - Use BBST to find all points with the x-coordinates in $[x_1, x_2]$
    - Of all these points, find all points with y-coord in $[y_1, y_2]$



For each subtree we need all points in $[y_1, y_2]$

data structure for range searching on y-coord

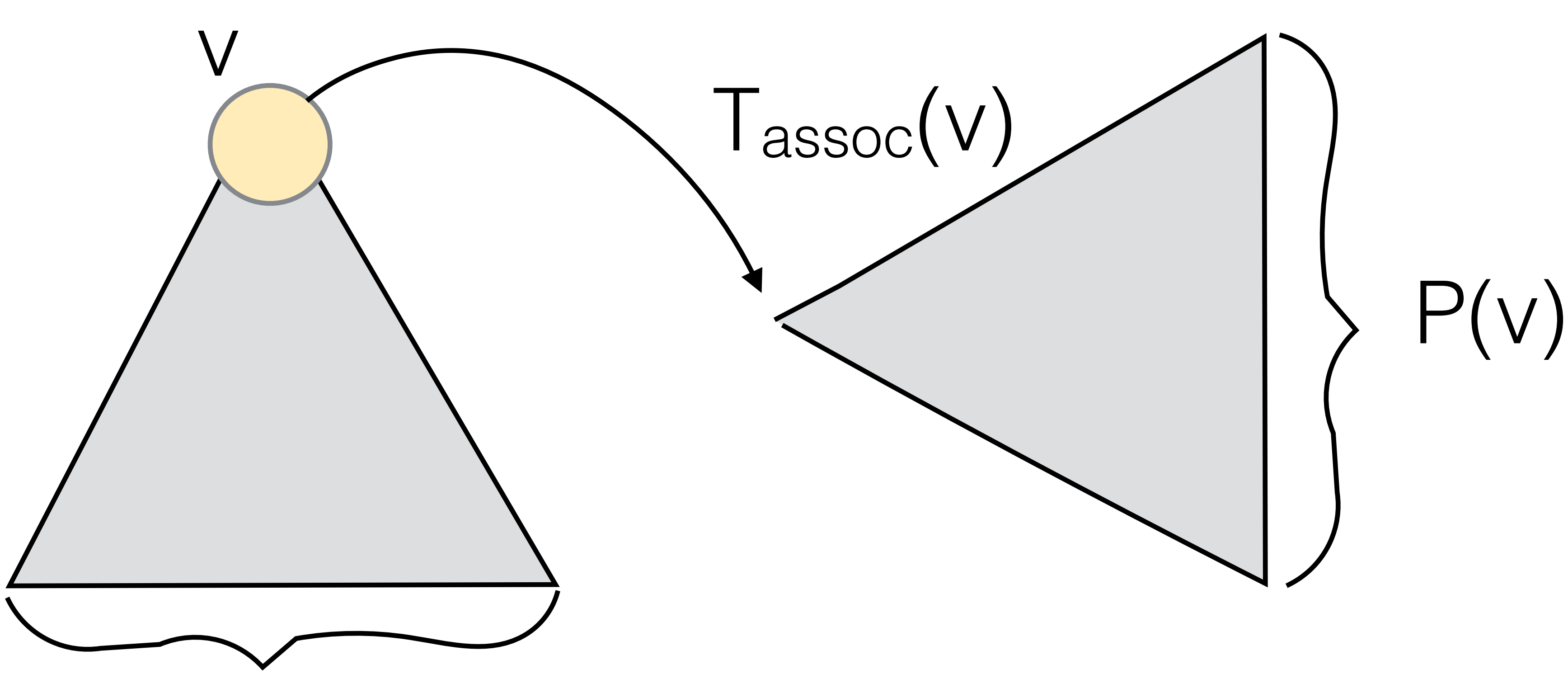
What is a good data structure for range search on y?

The 2D Range Tree

P: set of points

RangeTree(P) is

- A BBST T of P on x-coord
- Any node v in T stores a BBST T_{assoc} of P(v), by y-coord



P(v): all points in subtree rooted at v

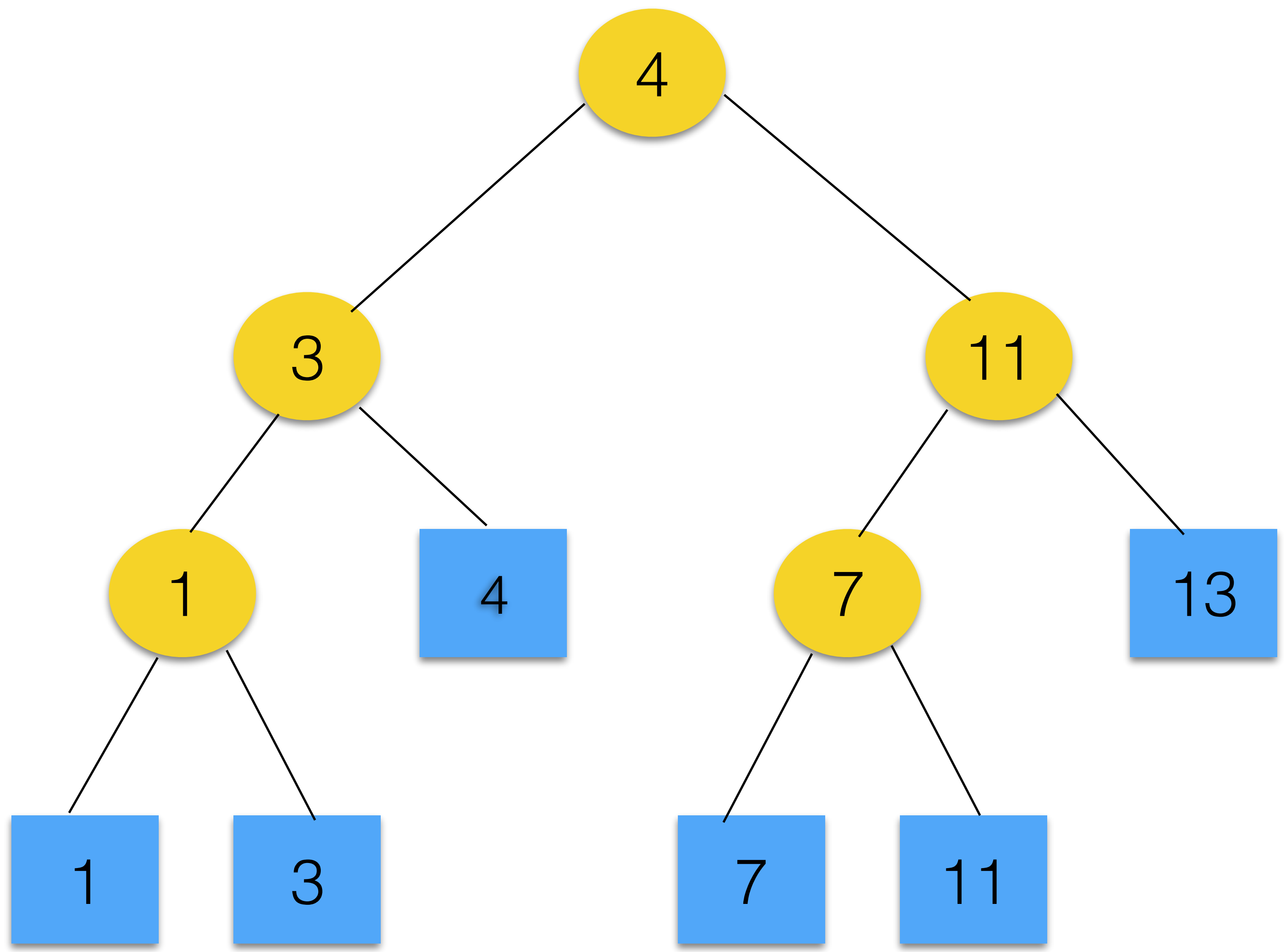
The 2D Range Tree

- We'll use a variant of BBSTs that store all data in leaves
    - It makes the details simpler

The 2D Range Tree

- We'll use a variant of BBSTs that store all data in leaves

Example: $P = \{ 1, 3, 4, 7, 11, 13 \}$



Class work

- Show the BBST with all data in leaves for P = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

- Write pseudocode for the algorithm to build BBST(P)

```
//create and return a BBST of P with all data in leaves
BuildBBST ( P )
```

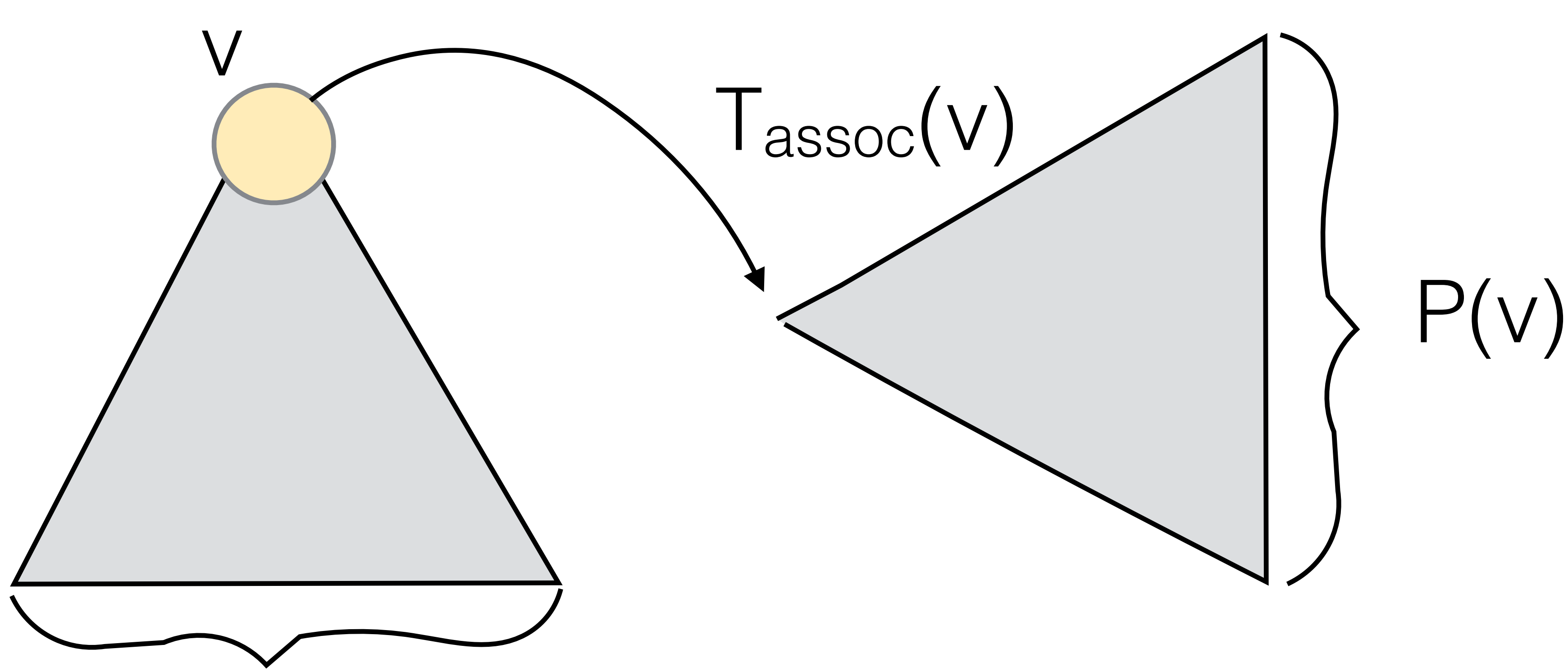
- Analysis?
- What if P is sorted?

The 2D Range Tree

P: set of points

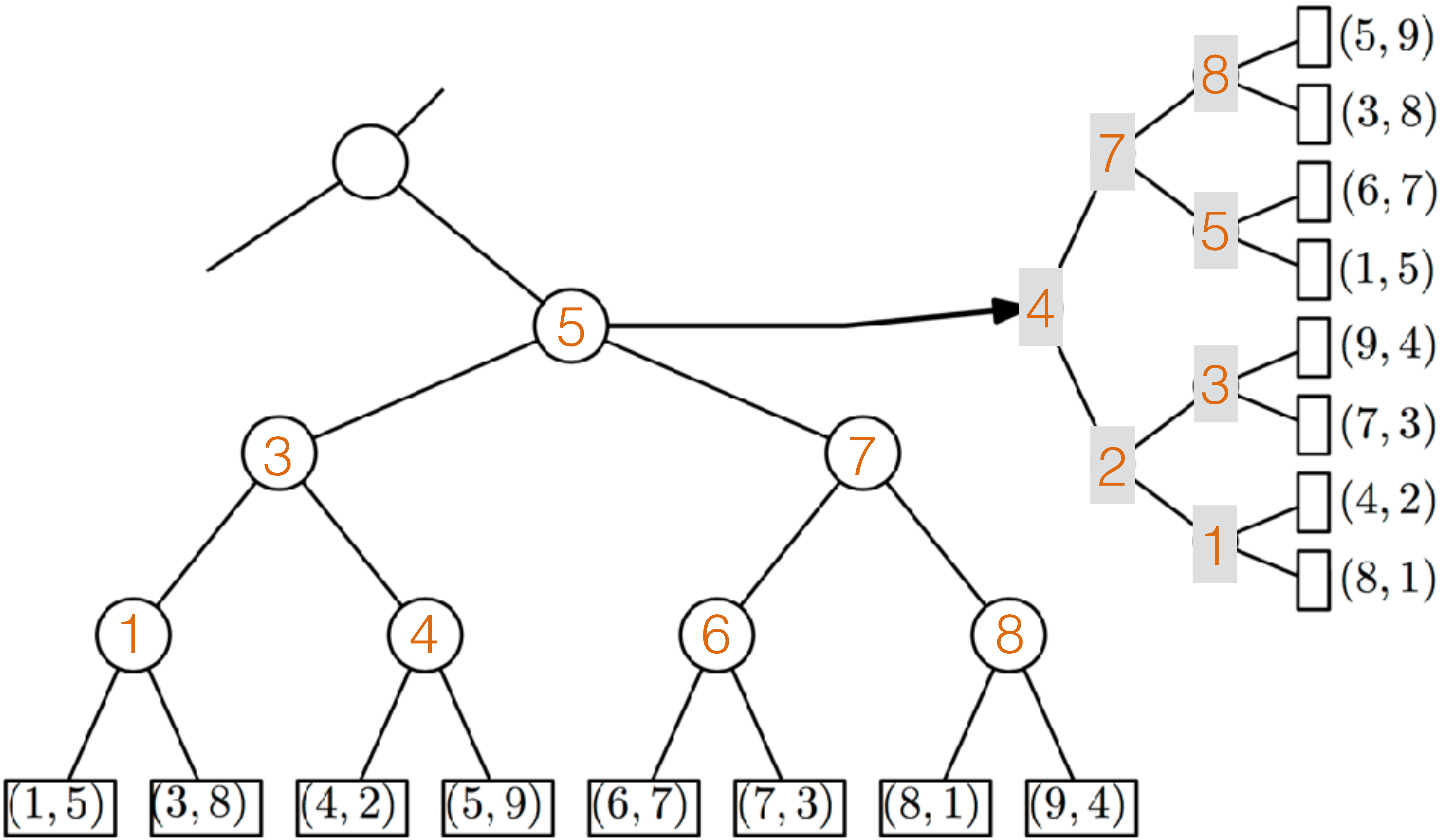
RangeTree(P) is

- A BBST T of P on x-coord
- Any node v in T stores a BBST T_{assoc} of P(v), by y-coord



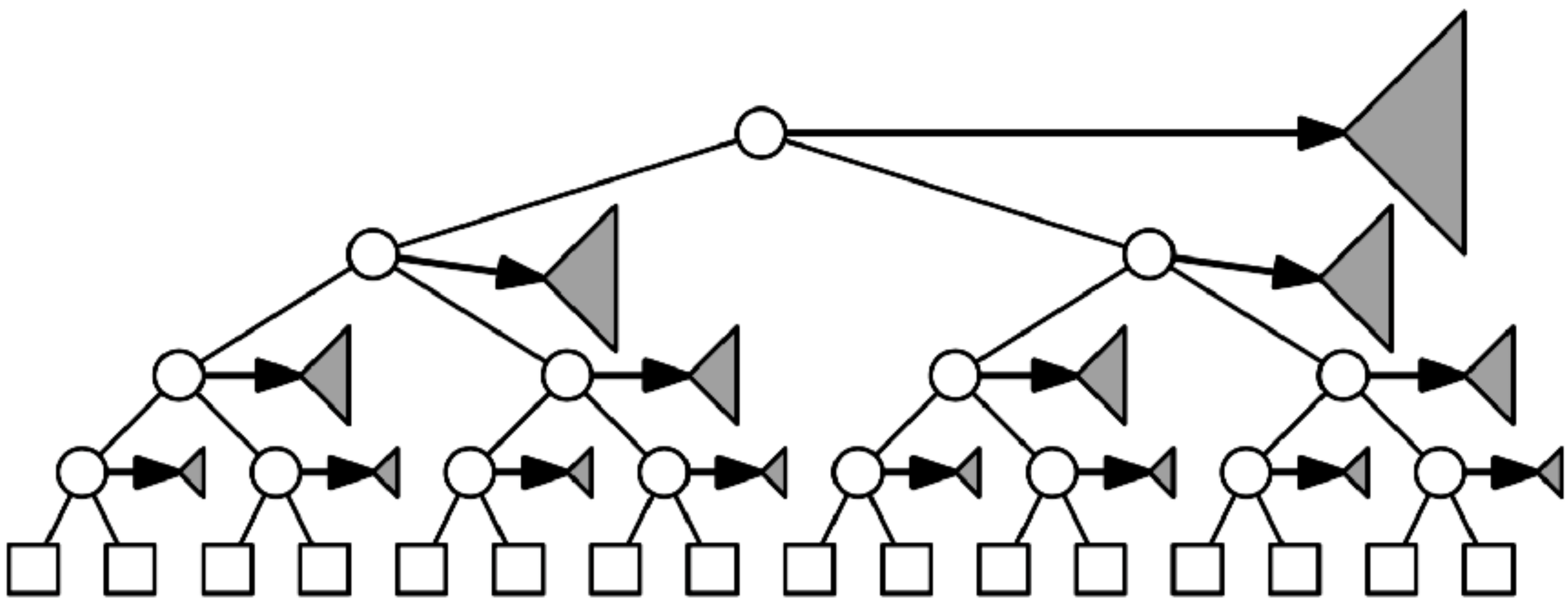
P(v): all points in subtree rooted at v

screen shot from Mark van Kreveld slides, http://www.cs.uu.nl/docs/vakken/ga/slides5b.pdf)



screen shot from Mark van Kreveld slides, http://www.cs.uu.nl/docs/vakken/ga/slides5b.pdf)

Every internal node stores a whole tree in an *associated structure*, on y-coordinate



Building a 2D Range Tree

Building a 2D Range Tree

Let $P = \{p_1, p_2, \ldots p_n\}$. Assume P sorted by x-coord.

Algorithm Build2DRT(P)

1. if P contains only one point:

   create a leaf v storing this point, create its T_{assoc} and return v

2. else

   1. Construct the associated structure: build a BBST T_{assoc} on the set of y-coordinates of P
   2. Partition P into 2 sets w.r.t. the median coordinate x_{middle}:

      $P_{left} = \{p \text{ in } P.\ p_x <= x_{middle}\}$, $P_{right} = \ldots$

   3. v_{left} = Build2DRT(P_{left})
   4. v_{right} = Build2DRT(P_{right})
   5. Create a node v storing x_{middle}, make v_{left} its left child, make v_{right} its right child, make T_{assoc} its associate structure
   6. return v

Class work

- Let $P = \{(1,4), (5,8), (4,1), (7,3), (3, 2), (2, 6), (8,7)\}$.

  Show the range tree of P.

The 2D Range Tree

Questions

- How to build it and how fast?
- How much space does it take?
- How do you answer range queries and how fast?

Building a 2D Range Tree

- Let $T(n)$ be the time of **Build2DRT(P)** , where P has n points
- Constructing a BBST on an unsorted set of keys takes $O(n \lg n)$
- Then

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

- This solves to $O(n \lg^2 n)$

Building a 2D Range Tree

- Common trick: pre-sort P and pass it as argument

  //P_x is set of points sorted by x-coord

  //P_y is set of points sorted by y-coord

  Build2DRT(P_x, P_y)

- Maintain the sorted sets through recursion

  P_1 sorted-by-x, P_1 sorted-by-y

  P_2 sorted-by-x, P_2 sorted-by-y

- Fact: If keys are in order, a BBST can be built in O(n)
- We have

  $T(n) = 2T(n/2) + O(n)$ which solves to $O(n \lg n)$

The 2D Range Tree

- How much space does a range tree use?

The 2D Range Tree

- How much space does a range tree use?

Two arguments can be made:

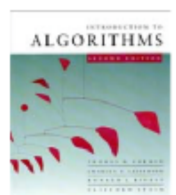
- At each level in the tree, each point is stored exactly once (in the associated structure of precisely one node). So every level stores all points and uses $O(n)$ space => $O(n \lg n)$

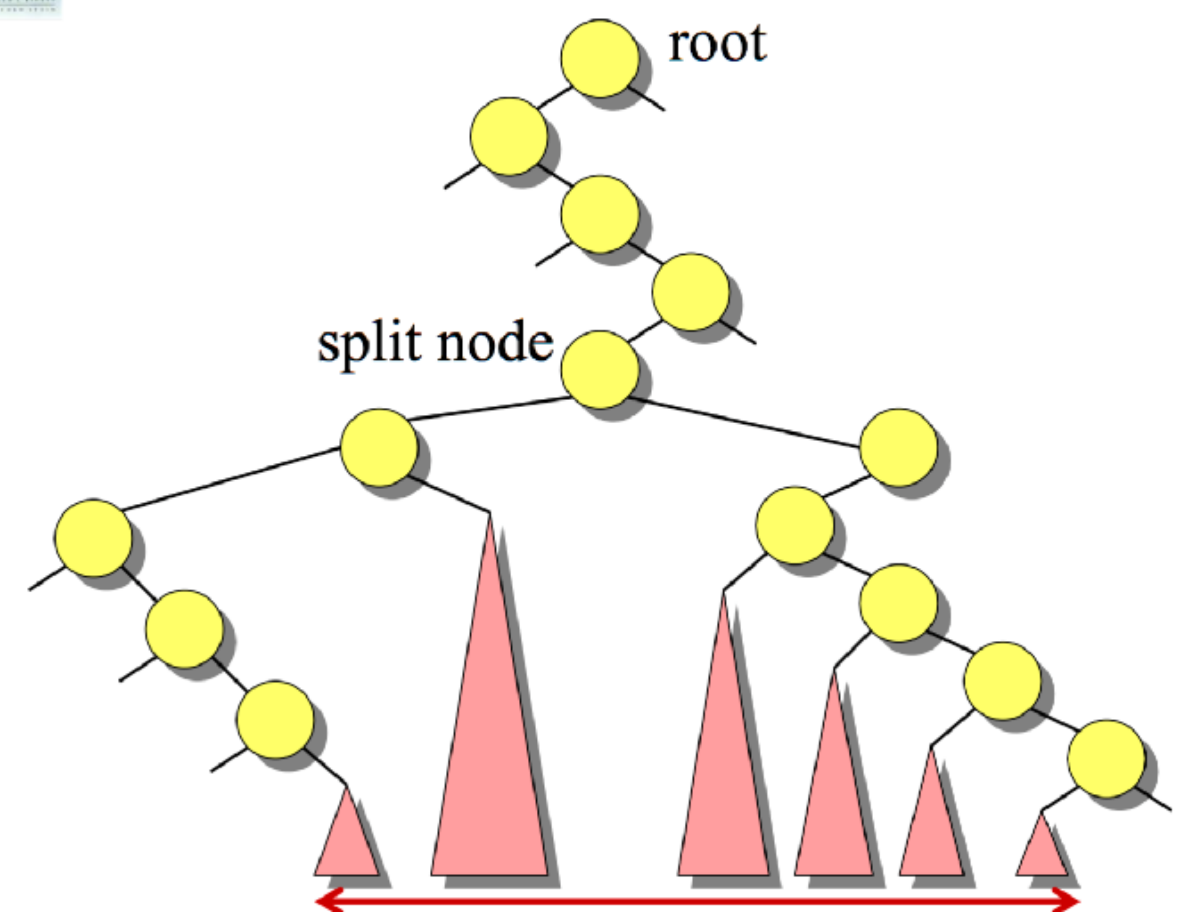
or

- Each point p is stored in the associated structures of all nodes on the path from root to p. So one point is stored $O(\lg n)$ times => $O(n \lg n)$

The 2D Range Tree

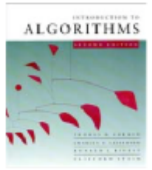
- How to answer range queries with a range tree, and how fast?

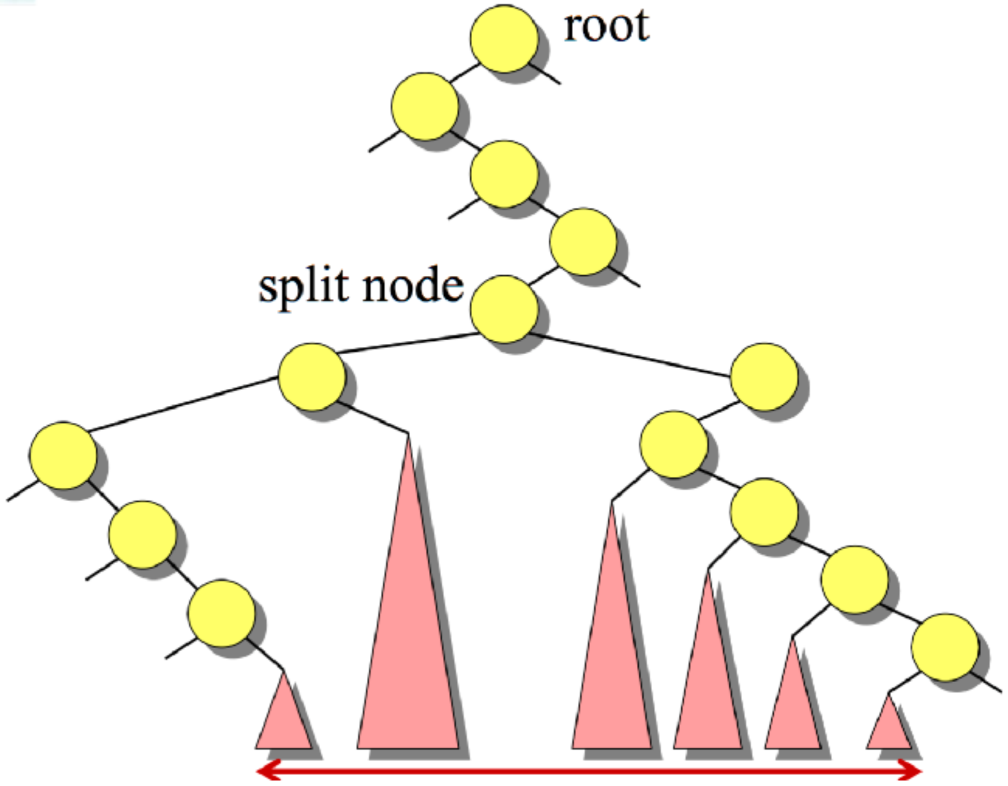




Range queries with the 2D Range Tree

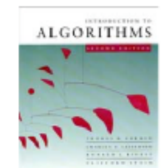
- Find the split node x_{split} (where the search paths for x_1 and x_2 split)
- Follow path root to x_1: for each node v to the **right** of the path, query its associated structure $T_{assoc}(v)$ with $[y_1,y_2]$
- Follow path root to x_2: for each node v to the **left** of the path, query its associated structure $T_{assoc}(v)$ with $[y_1,y_2]$





How long does this take?

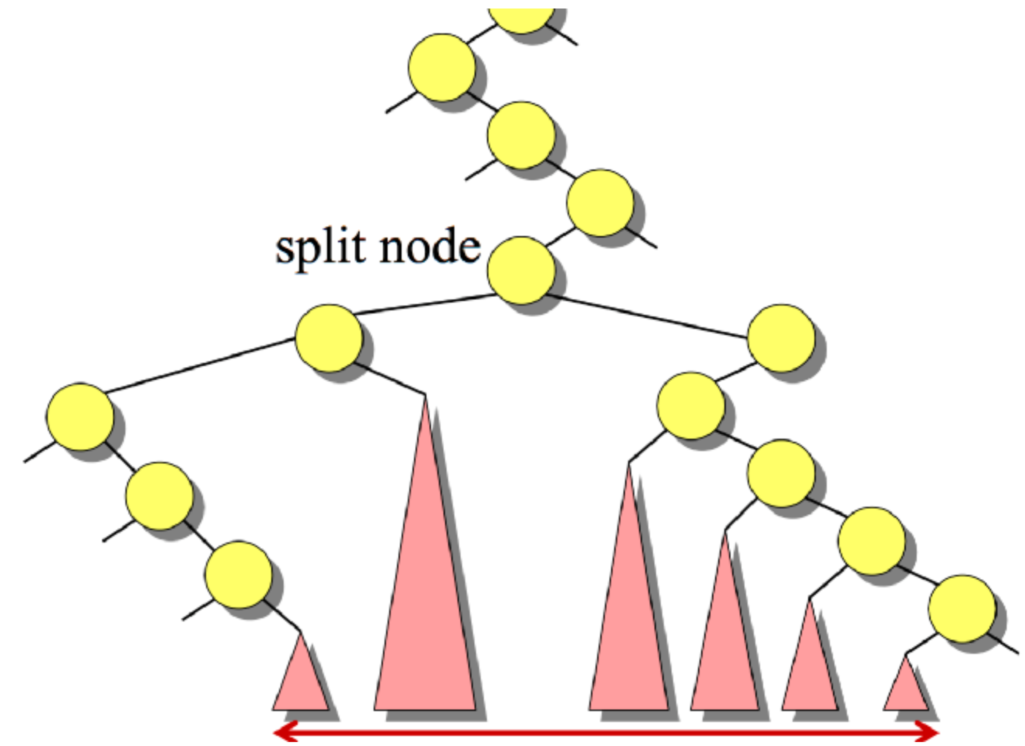
Range queries with the 2D Range Tree



How long does a range query take?

- There are O(lg n) subtrees in between the paths
- We query each one of them using its associated structure
- Querying T_{assoc} takes O(lg n_v + k')
- Overall it takes

  SUM {O(lg n_v + k') } = O($\lg^2 n$+k)



n_v: number of points in T_{assoc}

k': number of points in T_{assoc} that are in [y1,y2]

Comparison

2D

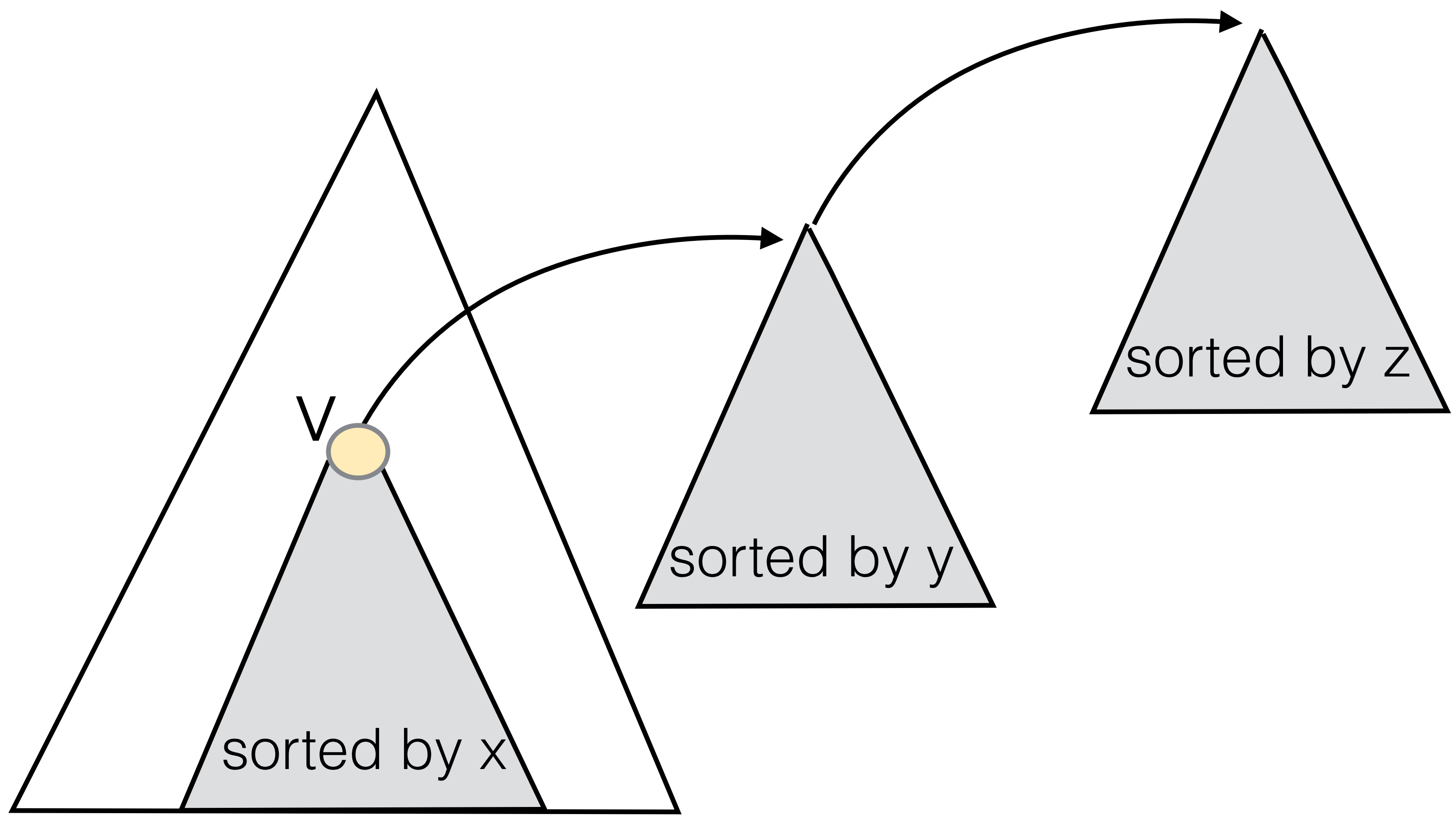
n	$\log n$	$\log^2 n$	$\sqrt{n}$
16	4	16	4
64	6	36	8
256	8	64	16
1024	10	100	32
4096	12	144	64
16384	14	196	128
65536	16	256	256
1M	20	400	1K
16M	24	576	4K

screen shot from Mark van Kreveld slides, http://www.cs.uu.nl/docs/vakken/ga/slides5b.pdf)

3D Range Trees

P: set of points in 3D

- 3DRangeTree(P)
  - Construct a BBST on x-coord
  - Each node v will have an associated structure that's a 2D range tree for P(v) on the remaining coords



3D Range Trees

Size:

- An associated structure for n points uses O(n lg n) space. Each point is stored in all associated structures of all its ancestors => $O(n \lg^2 n)$

Let's try this recursively

- Let $S_3(n)$ be the size of a 3D Range Tree of n points
- Find a recurrence for $S_3(n)$
  - Think about how you build it : you build an associated structure for P that's a 2D range tree; then you build recursively a 3D range tree for the left and right half of the points
  - $S_3(n) = 2S_3(n/2) + S_2(n)$
  - This solves to $O(n \lg^2 n)$

3D Range Trees

Build time:

- Think recursively
- Let $B_3(n)$ be the time to build a 3D Range Tree of n points
- Find a recurrence for $B_3(n)$
    - Think about how you build it : you build an associated structure for P that's a 2D range tree; then you build recursively a 3D range tree for the left and right half of the points
    - $B_3(n) = 2B_3(n/2) + B_2(n)$
    - This solves to $O(n \lg^2 n)$

3D Range Trees

Query:

- Query BBST on x-coord to find O(lg n) nodes
- Then perform a 2D range query in each node

Time?

- Let $Q_3(n)$ be the time to answer a 3D range query
- Find a recurrence for $Q_3(n)$
  - $Q_3(n) = O(\lg n) + O(\lg n) \times Q_2(n)$
  - This solves to $O(\lg^3 n + k)$

Comparison RangeTree and kdtree

4D

n	$\log n$	$\log^4 n$	$n^{3/4}$
1024	10	10,000	181
65,536	16	65,536	4096
1M	20	160,000	32,768
1G	30	810,000	5,931,641
1T	40	2,560,000	1G

screen shot from Mark van Kreveld slides, http://www.cs.uu.nl/docs/vakken/ga/slides5b.pdf)