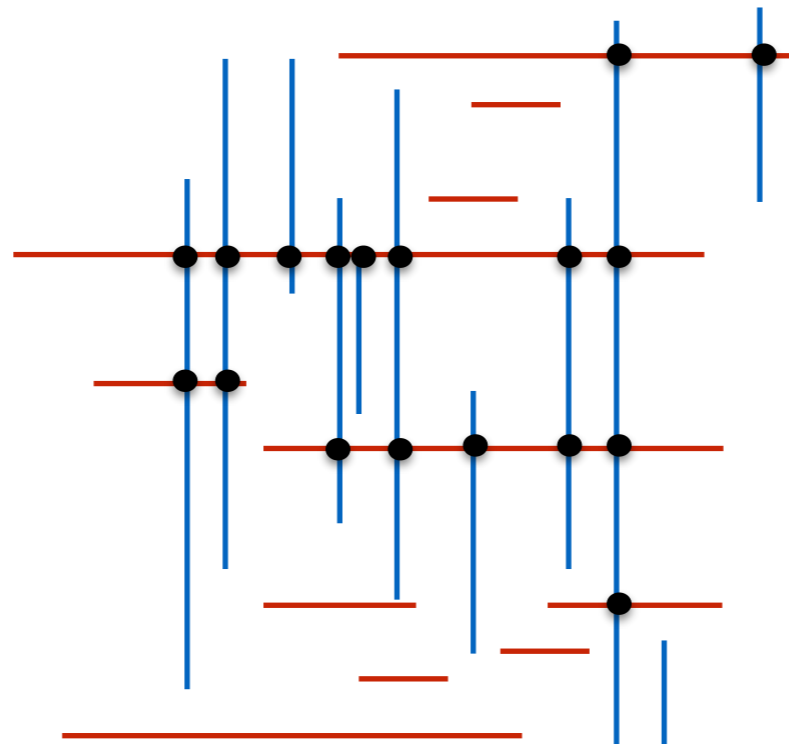


Orthogonal line segment intersection



Computational Geometry [csci 3250]

Laura Toma

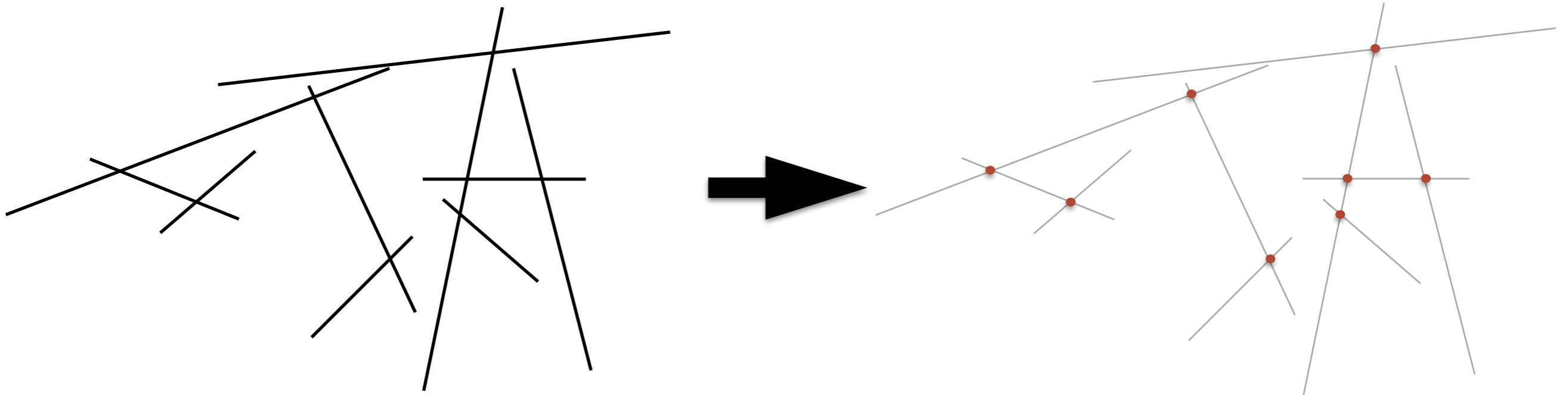
Bowdoin College

Line segment intersection

- The problem (what)
- Applications (why)
- Algorithms (how)
 - A special case: Orthogonal line segments
 - General case and Bentley-Otman line sweep algorithm

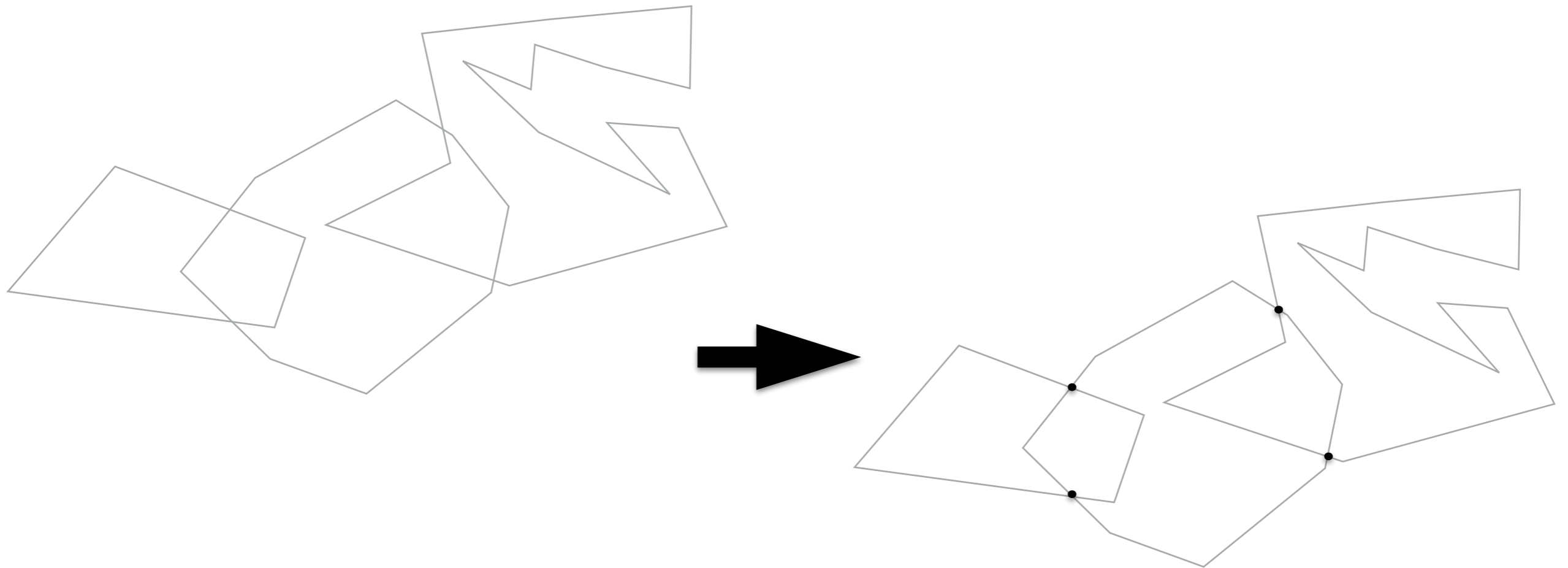
Line segment intersection

Problem: Given a set of line segments in 2D, find all their pairwise intersections.



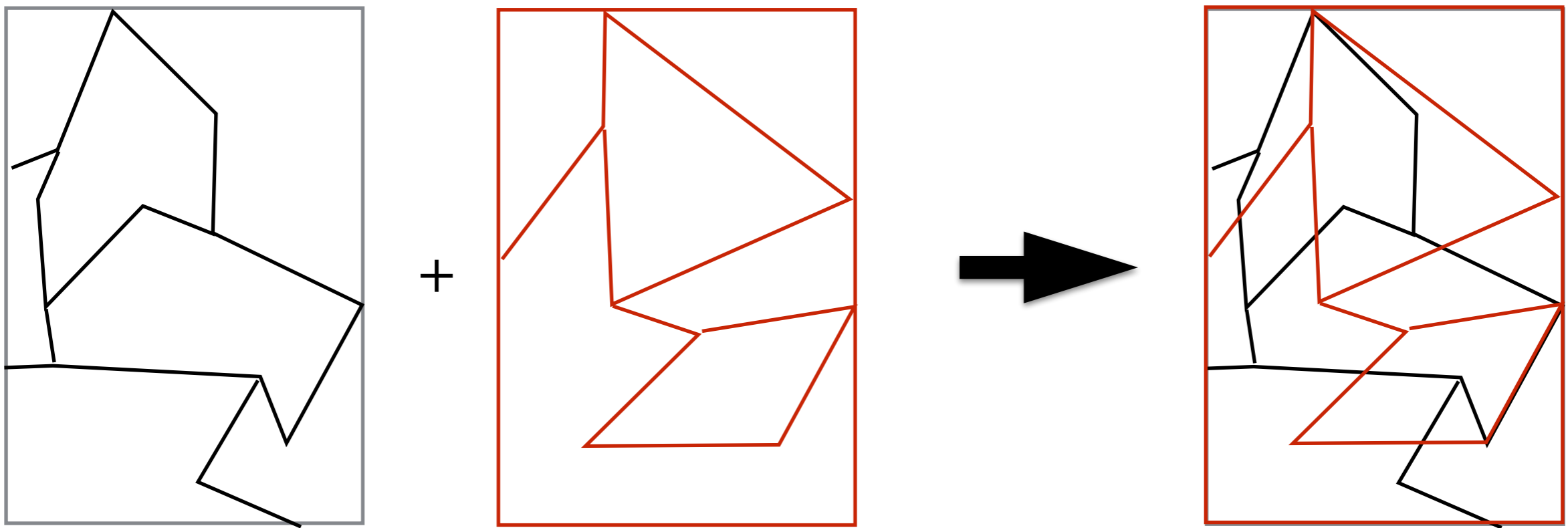
Line segment intersection

Problem: Given a set of line segments in 2D, find all their pairwise intersections.



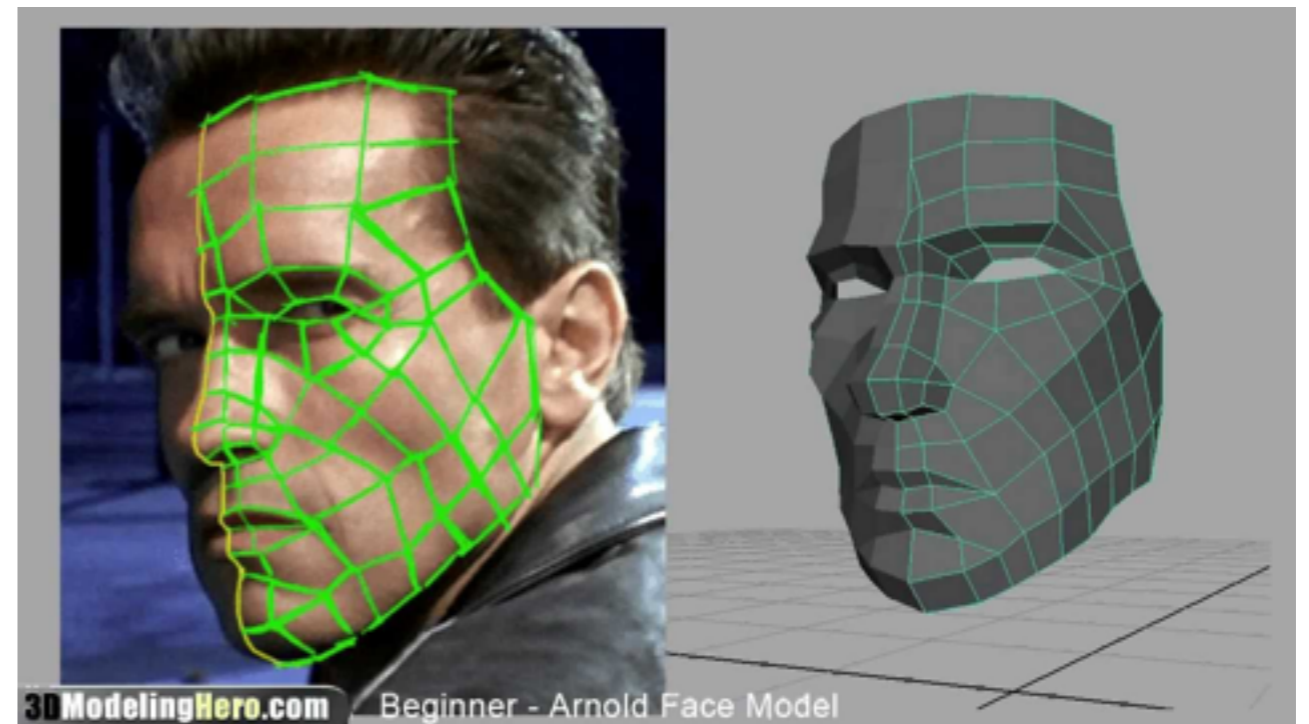
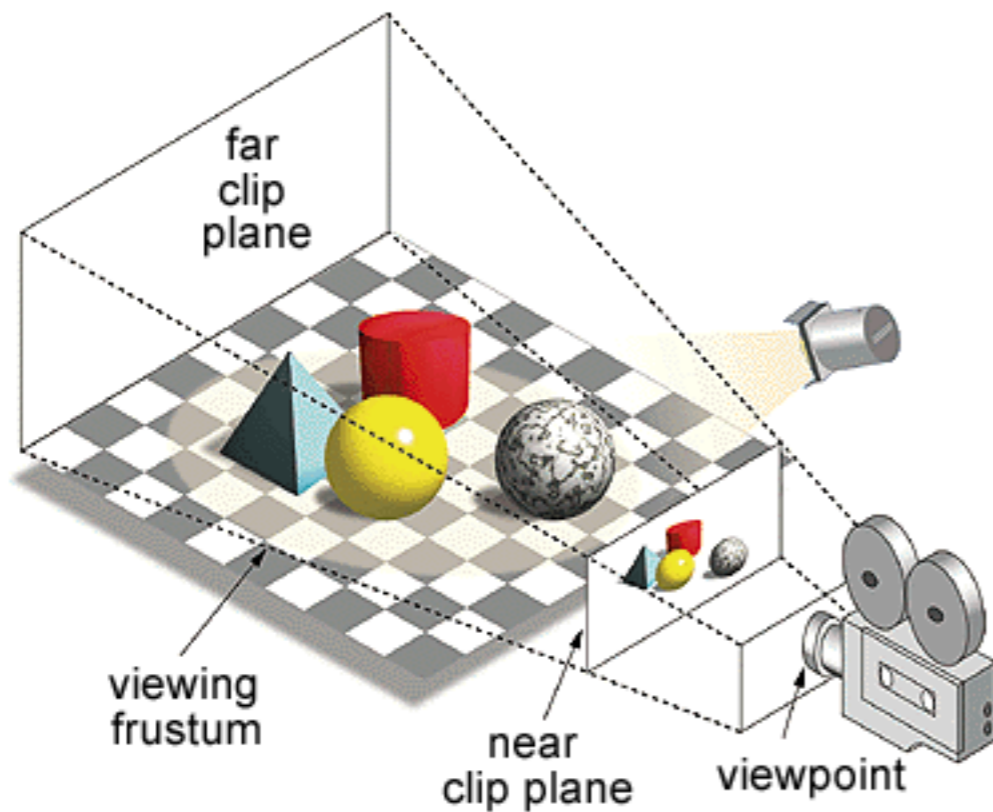
Line segment intersection

Problem: Given a set of line segments in 2D, find all their pairwise intersections.



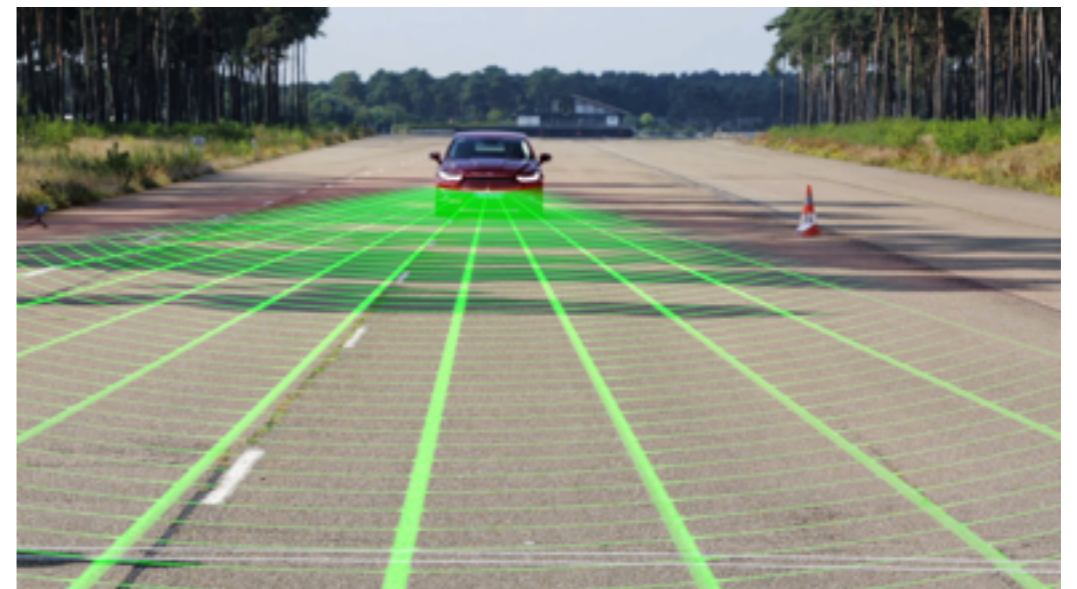
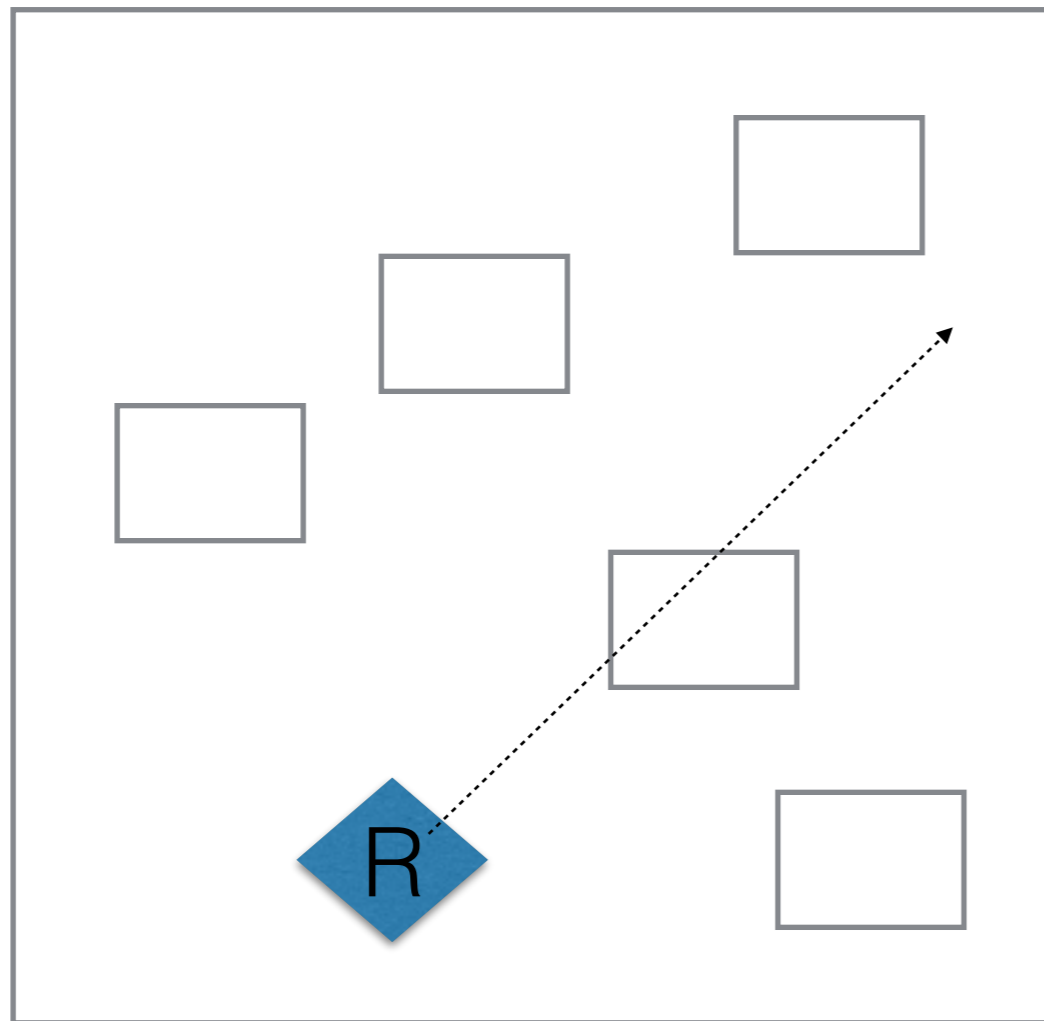
Applications

Graphics: rendering => hidden surfaces ==> intersections



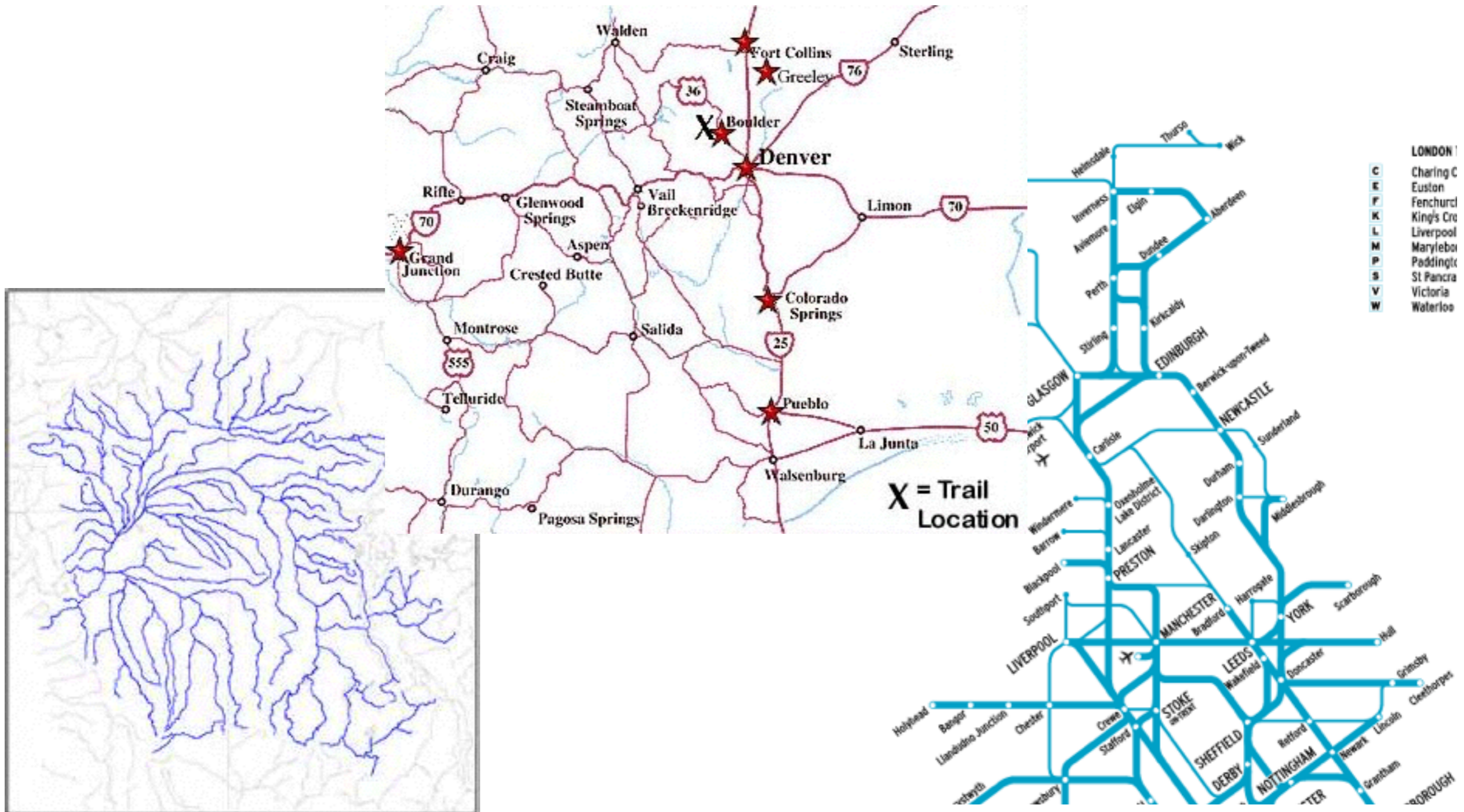
Applications

Motion planning and collision detection in autonomous systems/robotics



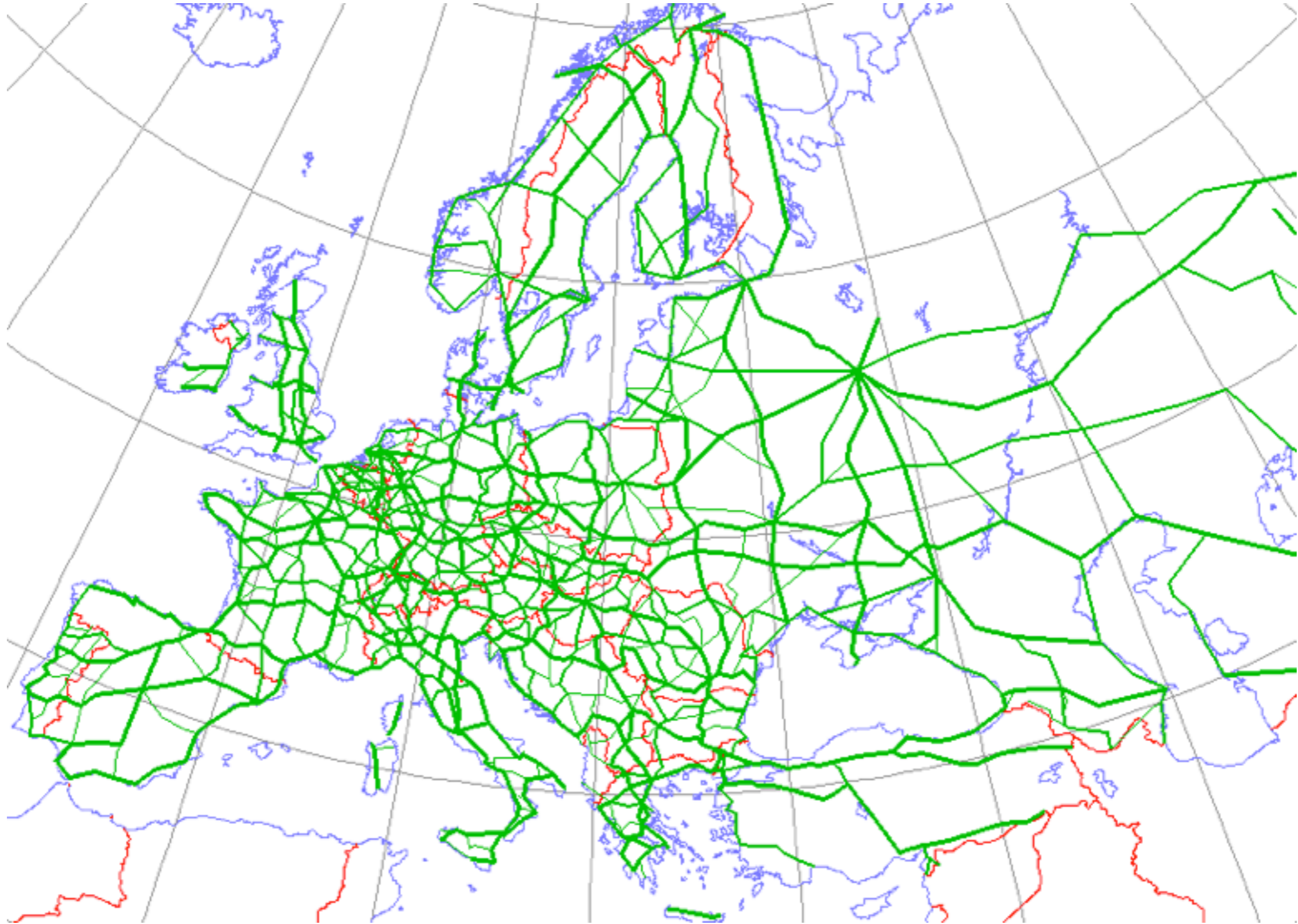
Applications

Geographical data: River networks, road networks, railways, ..



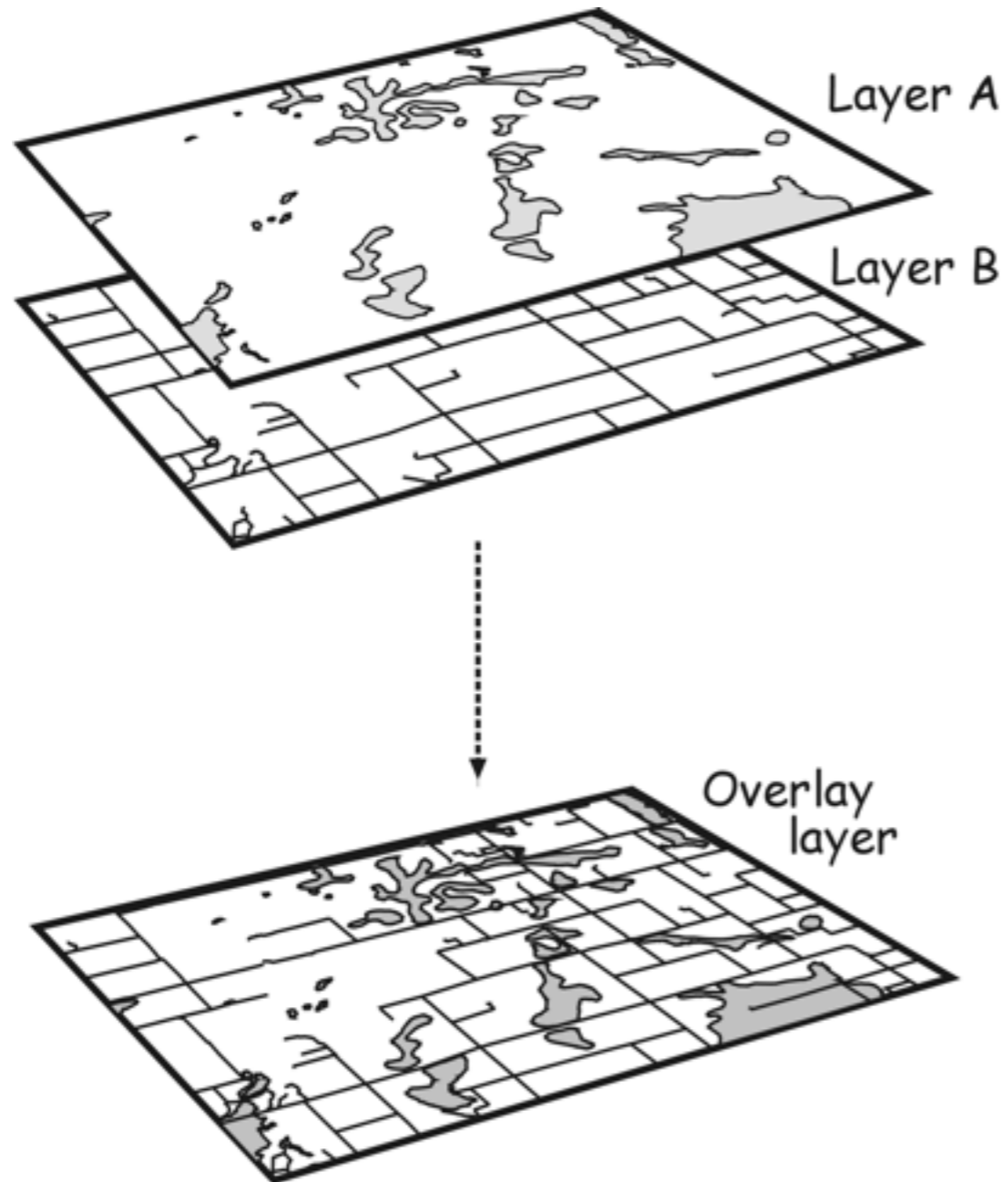
Applications

Geographical data: River networks, road networks, railways, ..



Applications

Map overlay in GIS



Algorithms

Naive

Notation

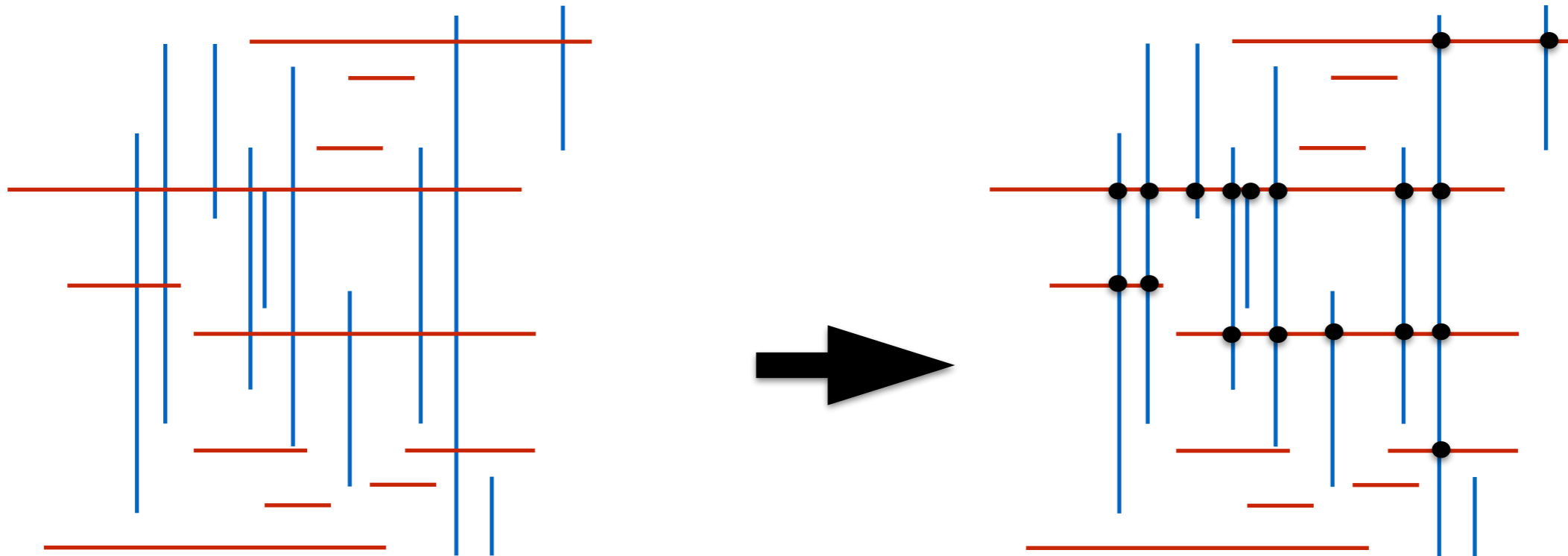
- n : size of the input (number of segments)
- k : size of output (number of intersections)

Problem: Given a set of n line segments in 2D, find all their pairwise intersections.

Exercises:

- Give upper and lower bounds for k , draw examples that achieve these bounds.
- Give a straightforward algorithm that computes all intersections and analyze its running time. Give scenarios when this algorithm is efficient/inefficient.
- What is your intuition of an upper bound for this problem? (how fast would you hope to be able to solve it?)

A special case: Orthogonal line segment intersection



Exercises

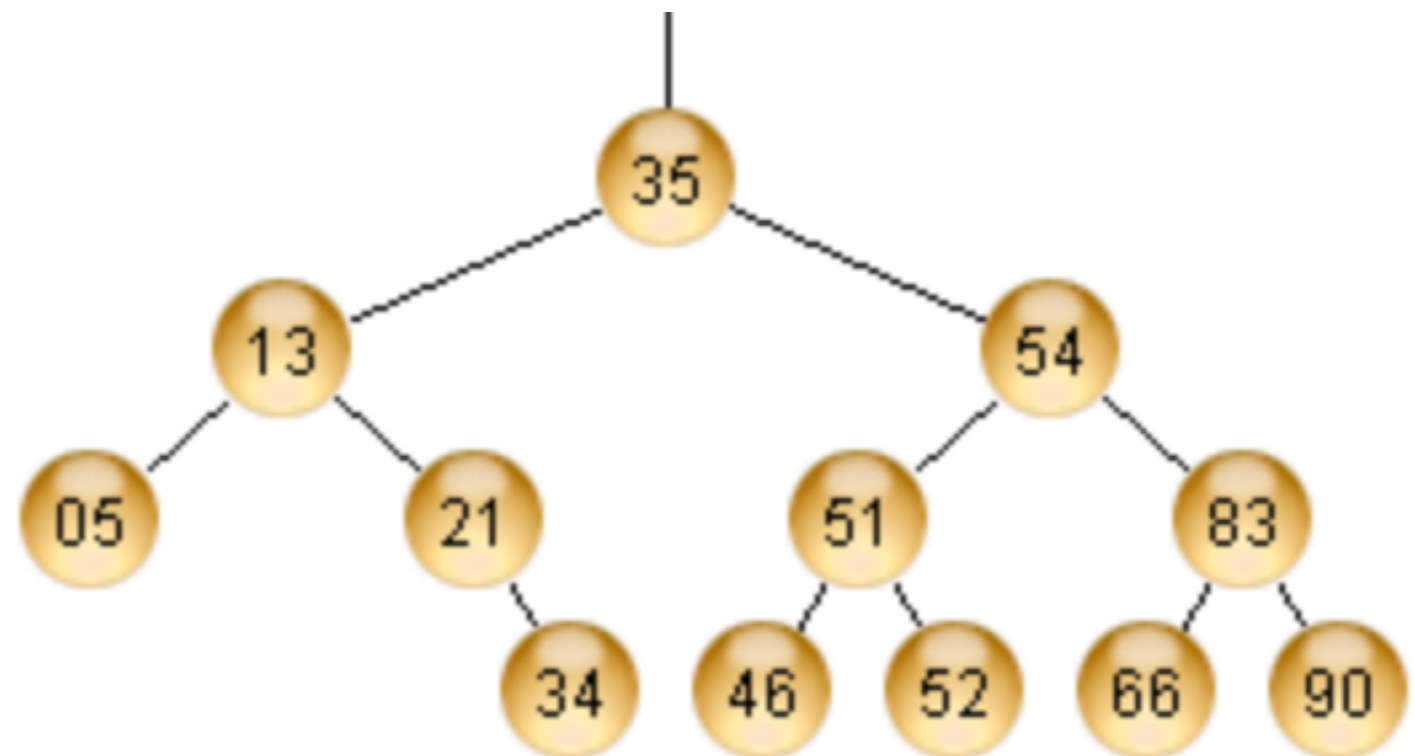
- Come up with a straightforward algorithm and analyze its time
- Improved algorithm?

Balanced Binary Search Trees

- review -

Binary Search Trees (BST)

- Operations
 - insert
 - delete
 - search
 - successor, predecessor
 - traversals (in order, ..)
 - min, max

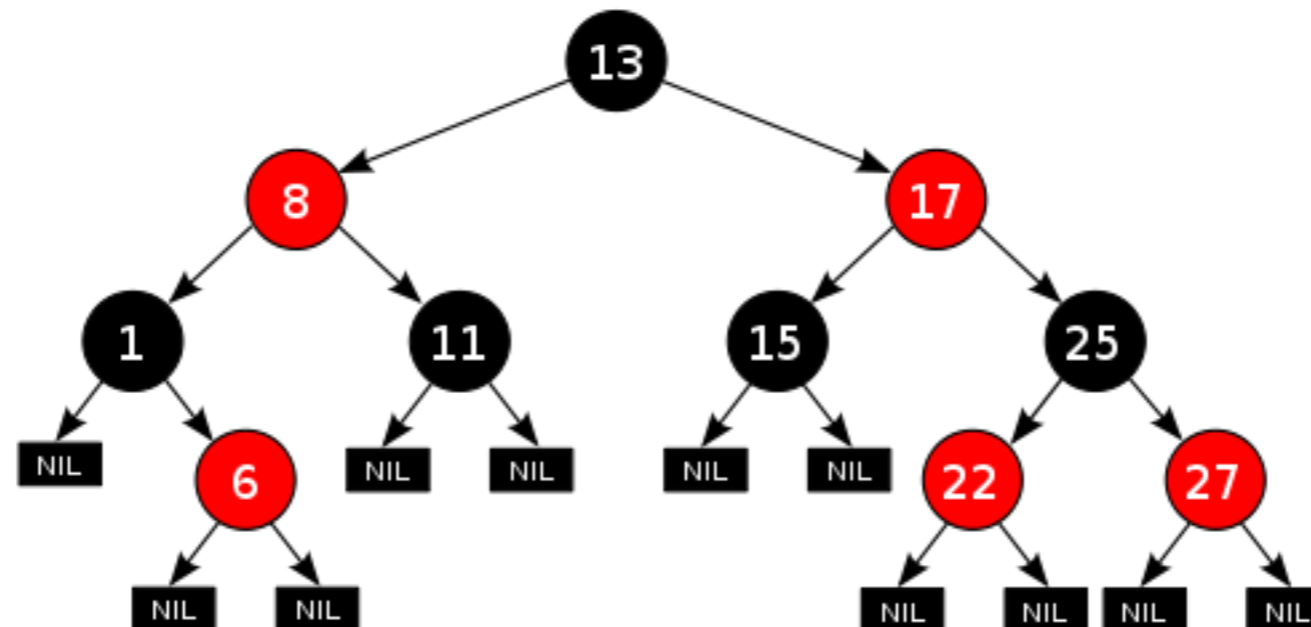


Balanced Binary Search Trees (BBST)

- Binary search trees + invariants that constrain the tree to be balanced (and thus have logarithmic height)
- These invariants have to be maintained when inserting and deleting (so we can think of the tree as self-balancing)
- BBST variants
 - red-black trees
 - AVL trees
 - B-trees
 - (a,b) trees
 - ...

Example: Red-Black trees

- Binary search tree, and
 - Each node is Red or Black
 - The children of a Red node must be Black
 - The number of Black nodes on any path from the root to any node that does not have two children must be the same

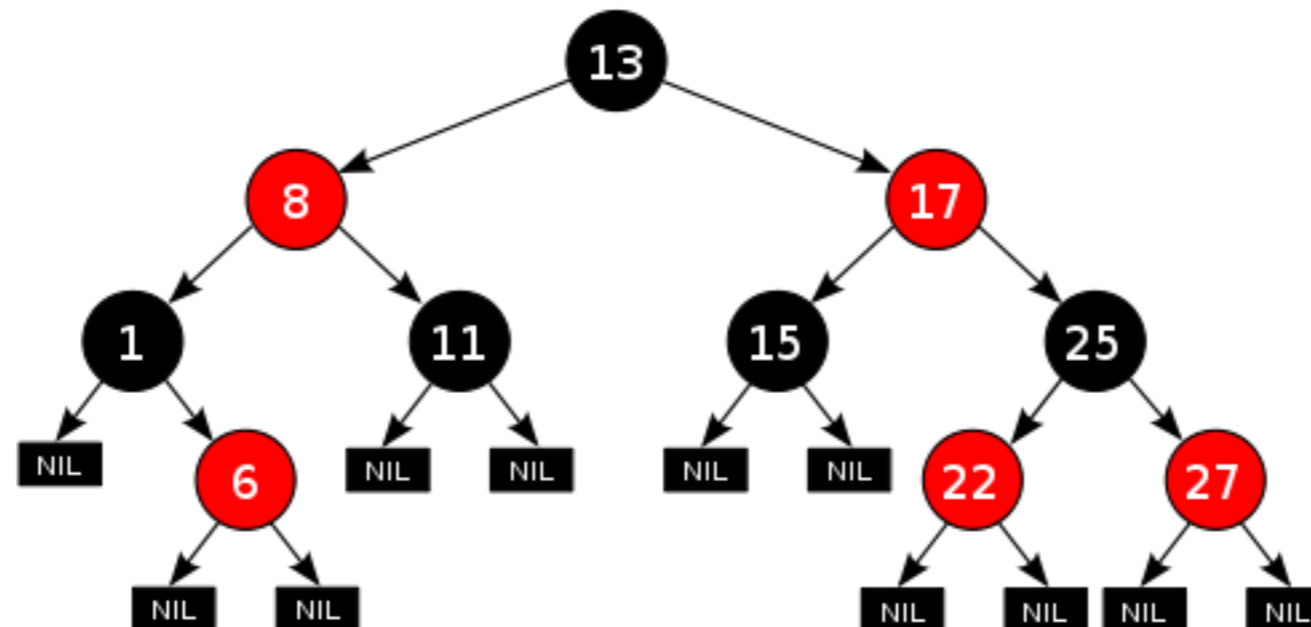


Note:

- easier to conceptualize the tree as containing explicit NULL leaves, all Black
- the number of Black nodes on any root-to-leaf path must be the same

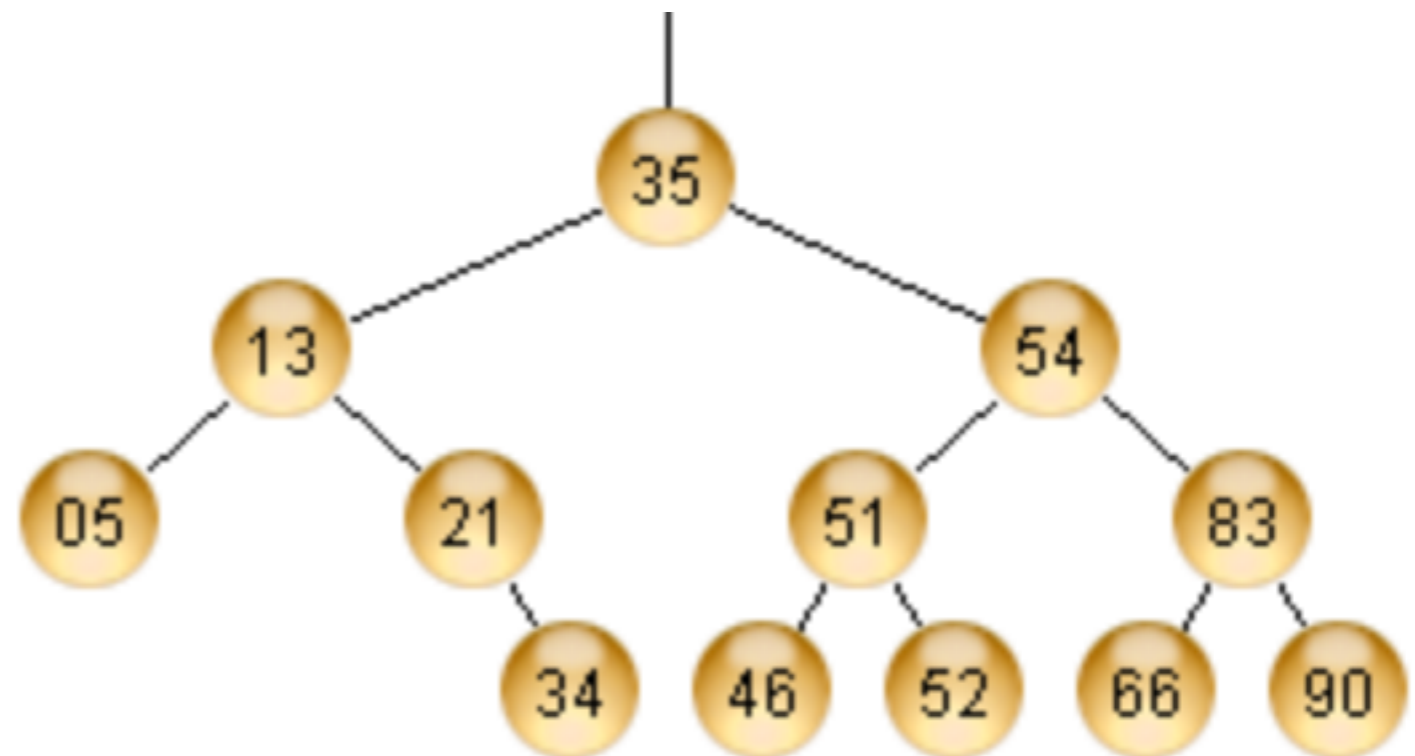
Example: Red-Black trees

- **Theorem:**
 - After an insertion or a deletion, the RB tree invariants can be maintained in additional $O(\lg n)$ time. This is done by performing rotations and recoloring nodes on the path from the inserted/deleted node to the root.



Binary Search Trees

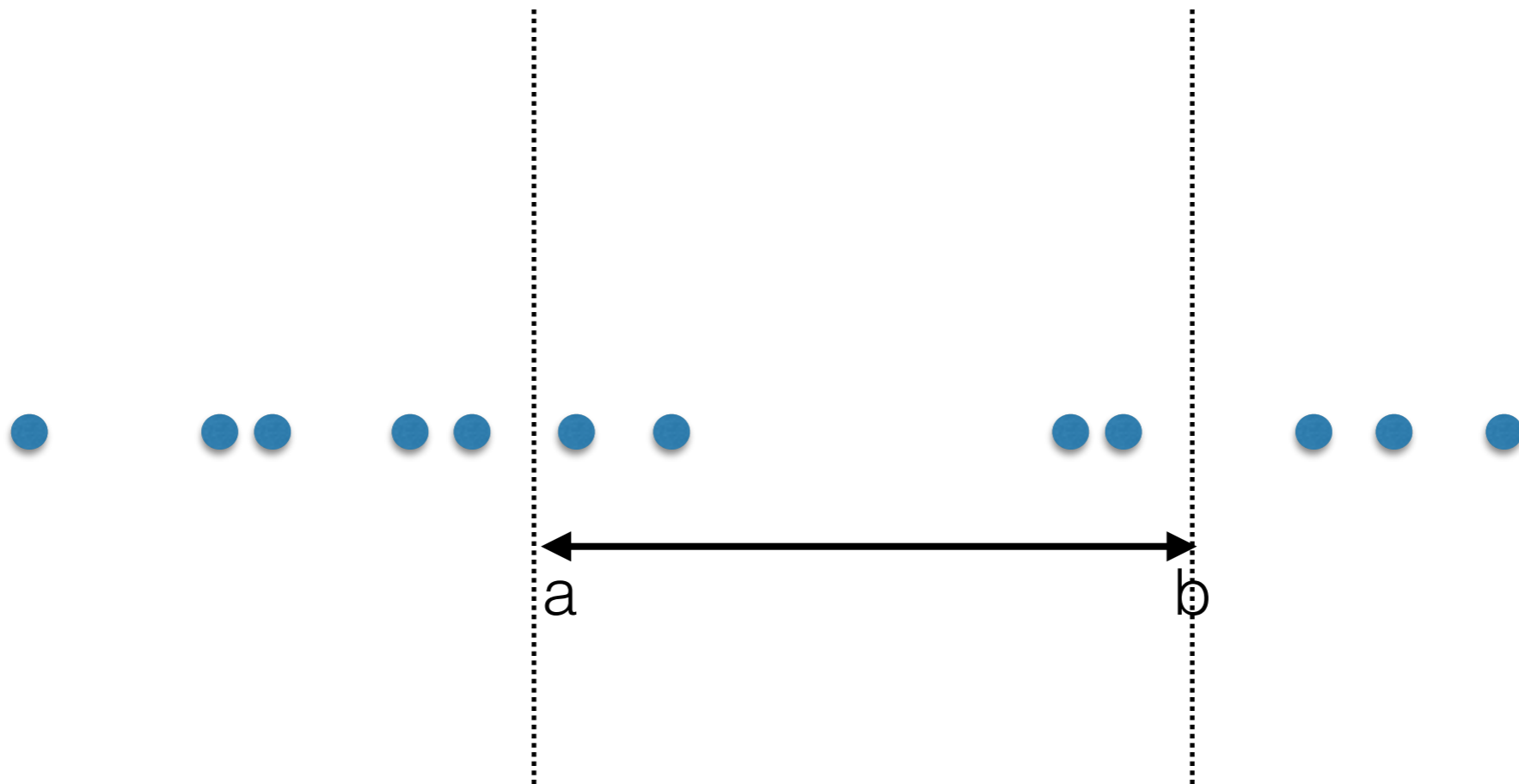
- Operations
 - insert
 - delete
 - search
 - successor, predecessor
 - traversals (in order, ..)
 - min, max
 - range search (1D)



1D Range Searching

- Given a set of values $P = \{x_1, x_2, x_3, \dots, x_n\}$
- Pre-process it in order to answer

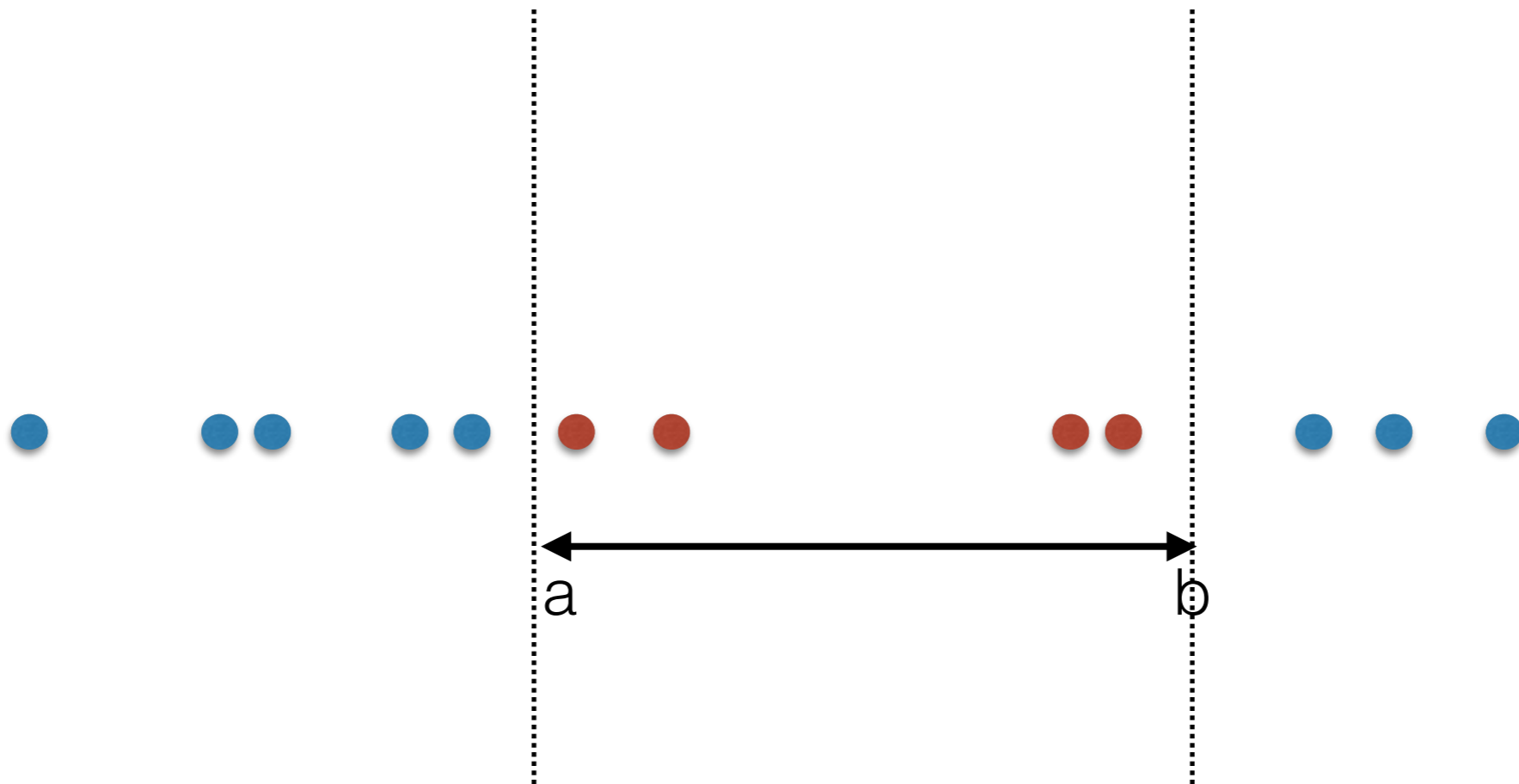
`rangeSearch(a,b)`: return all elements in P in interval (a,b)



1D Range Searching

- Given a set of values $P = \{x_1, x_2, x_3, \dots, x_n\}$
- Pre-process it in order to answer

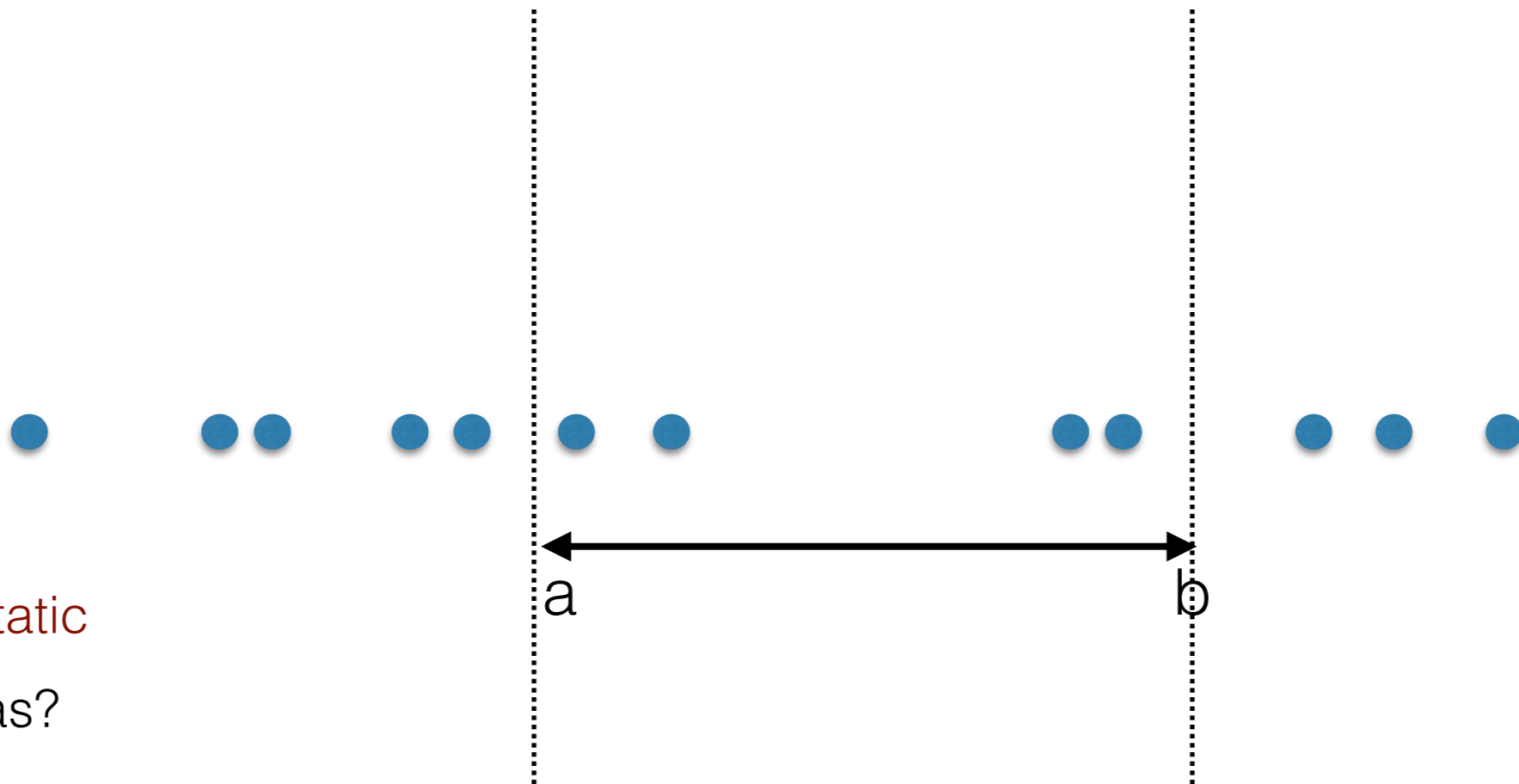
rangeSearch(a,b): return all elements in P in interval (a,b)



1D Range Searching

- Given a set of values $P = \{x_1, x_2, x_3, \dots, x_n\}$
- Pre-process it in order to answer

rangeSearch(a,b): return all elements in P in interval (a,b)

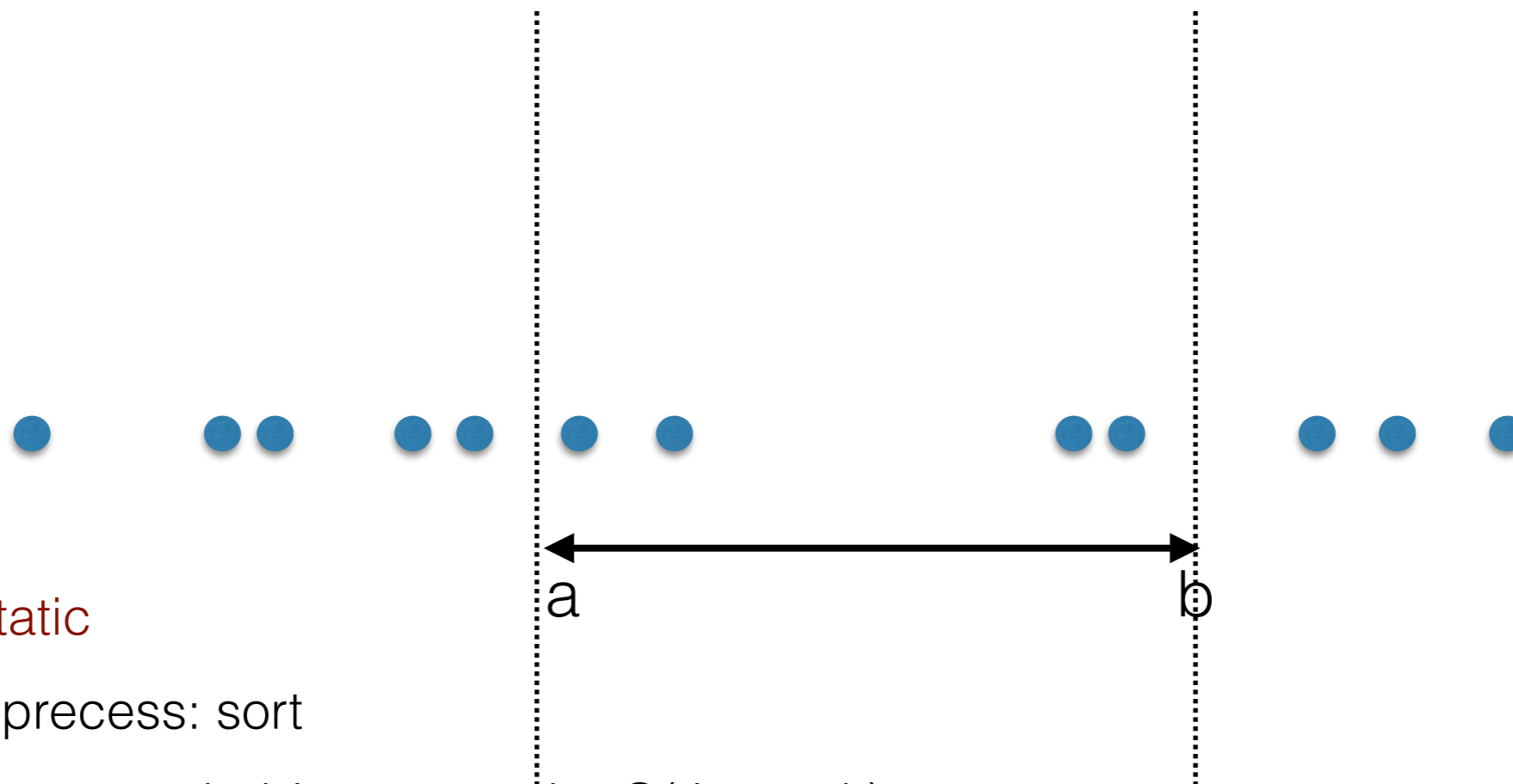


- If P is static
 - Ideas?

1D Range Searching

- Given a set of values $P = \{x_1, x_2, x_3, \dots, x_n\}$
- Pre-process it in order to answer

rangeSearch(a,b): return all elements in P in interval (a,b)

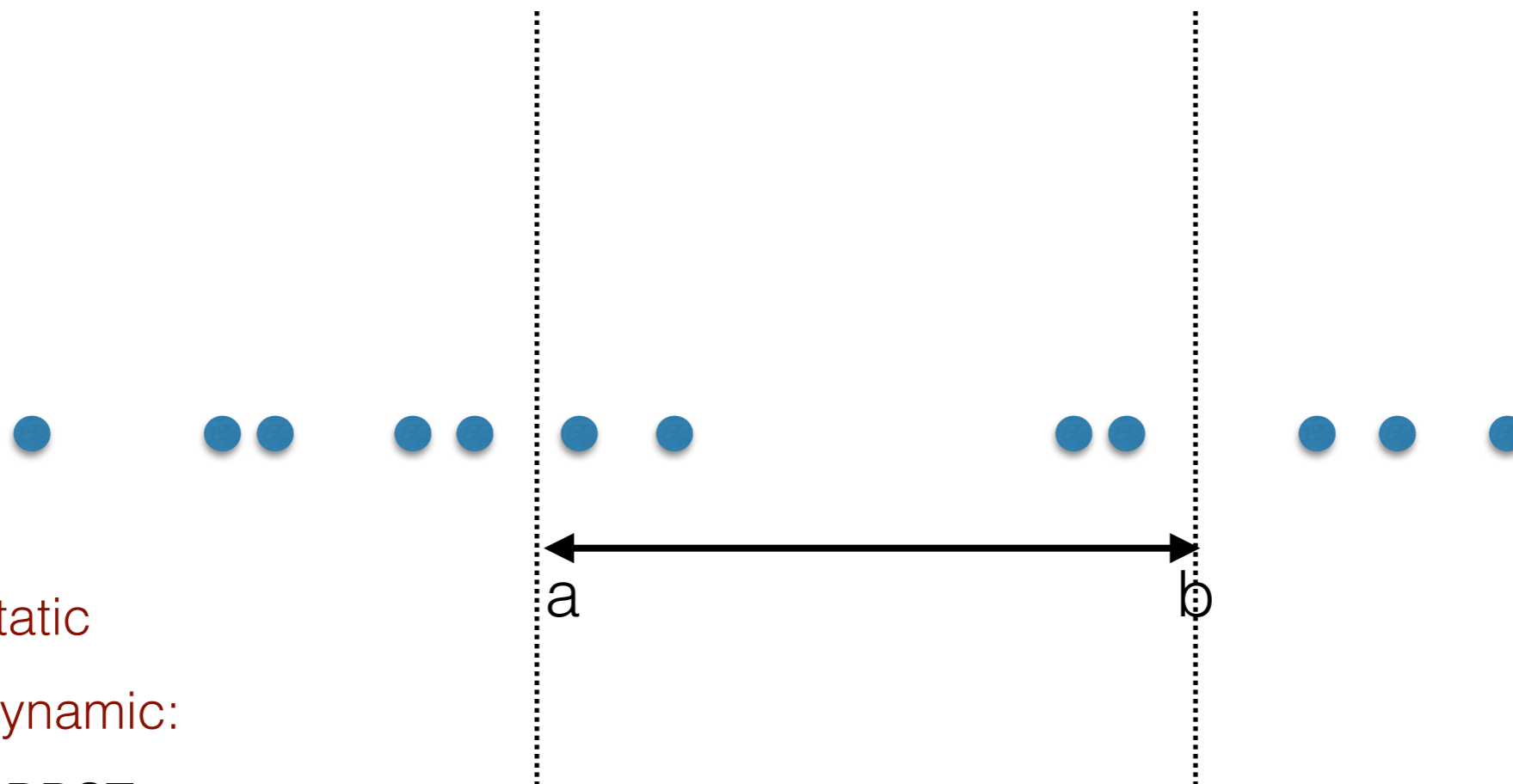


- If P is static
 - Pre-process: sort
 - Range search: binary search , $O(\lg n + k)$ per query

1D Range Searching

- Given a set of values $P = \{x_1, x_2, x_3, \dots, x_n\}$
- Pre-process it in order to answer

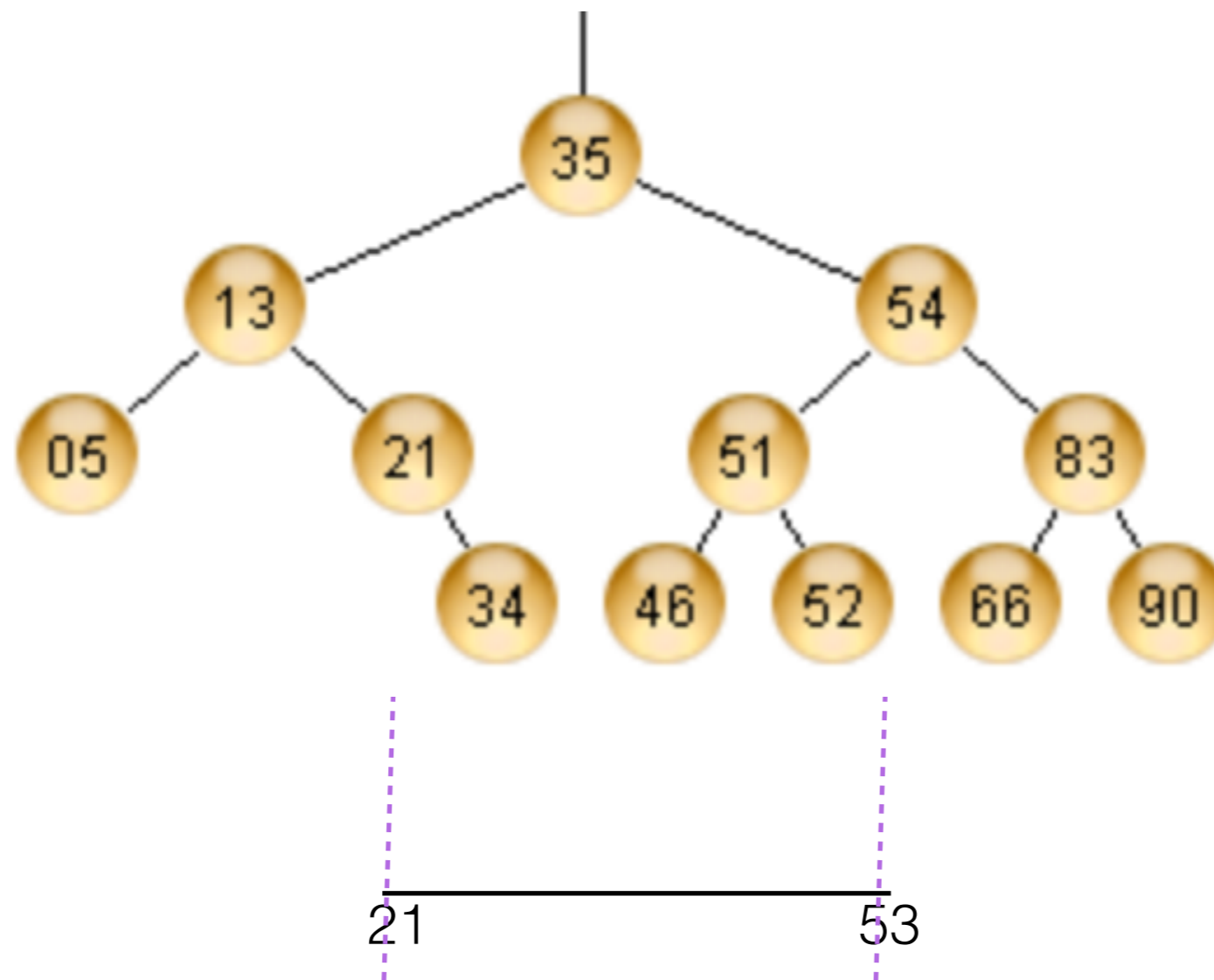
rangeSearch(a,b): return all elements in P in interval (a,b)



- If P is static
- If P is dynamic:
 - use BBST

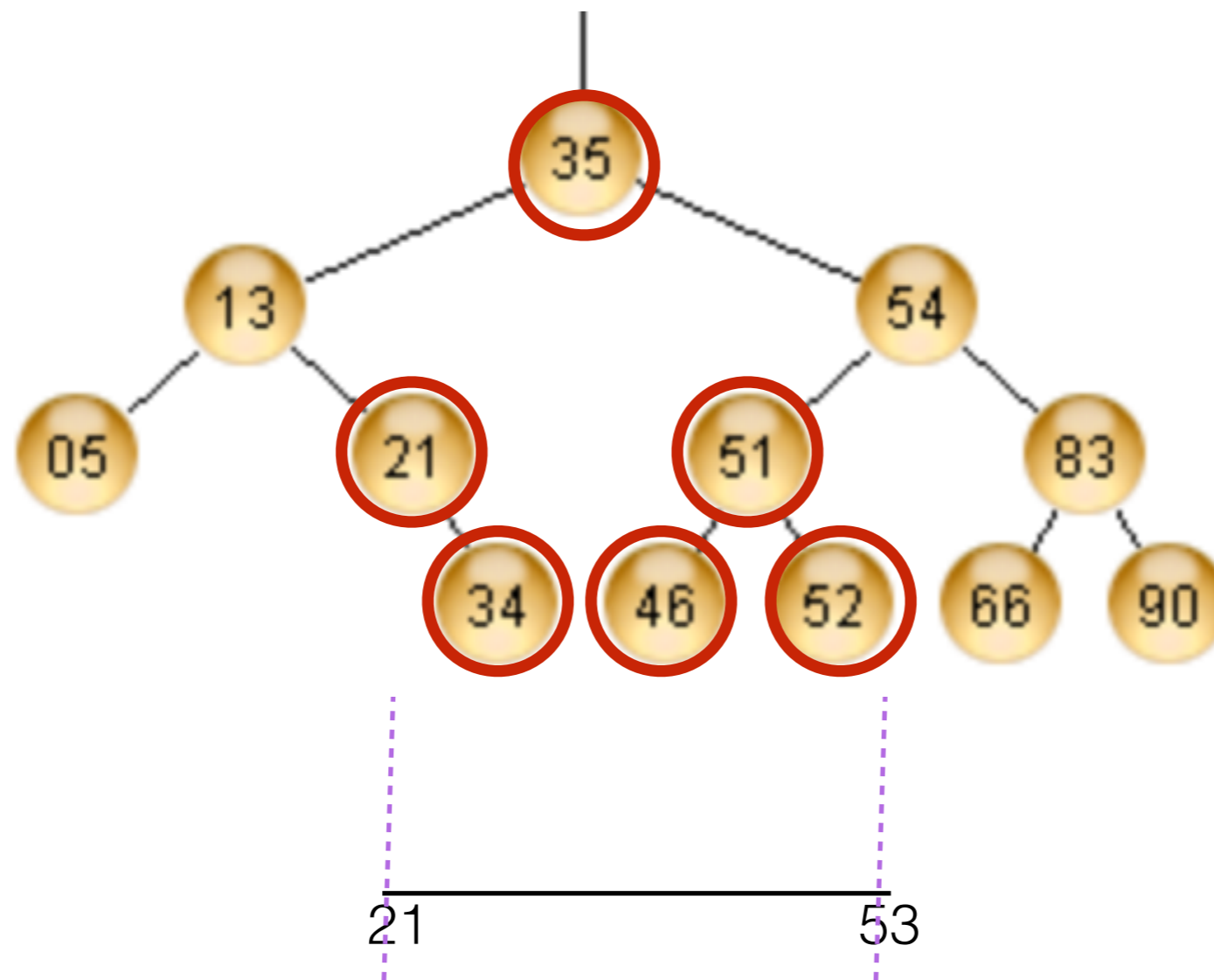
1D range searching with Binary Search Trees

Example: `range_search(21, 53)`: return 21, 34, 35, 46, 51, 52



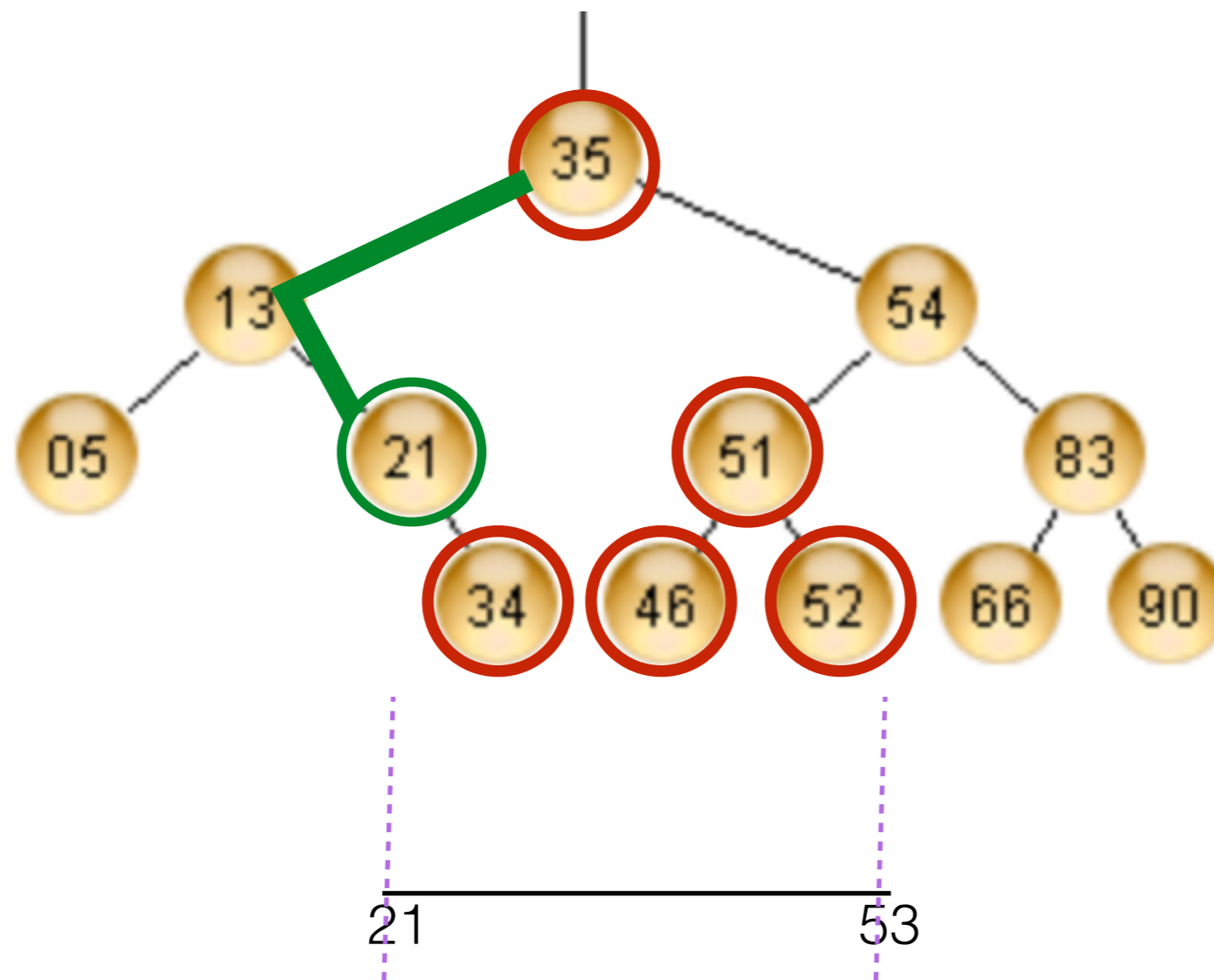
1D range searching with Binary Search Trees

Example: `range_search(21, 53)`: return 21, 34, 35, 46, 51, 52



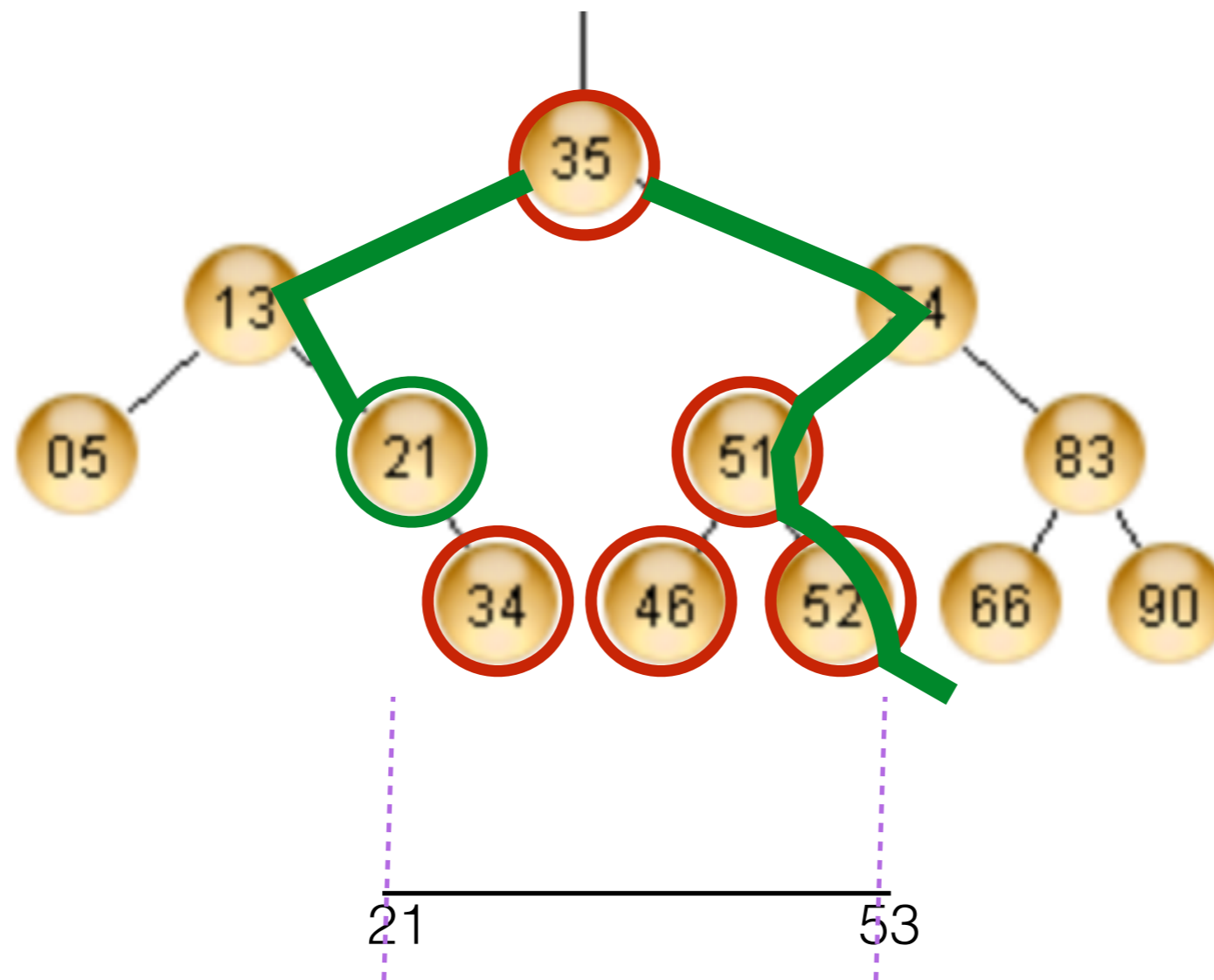
1D range searching with Binary Search Trees

Example: `range_search(21, 53)`: return 21, 34, 35, 46, 51, 52



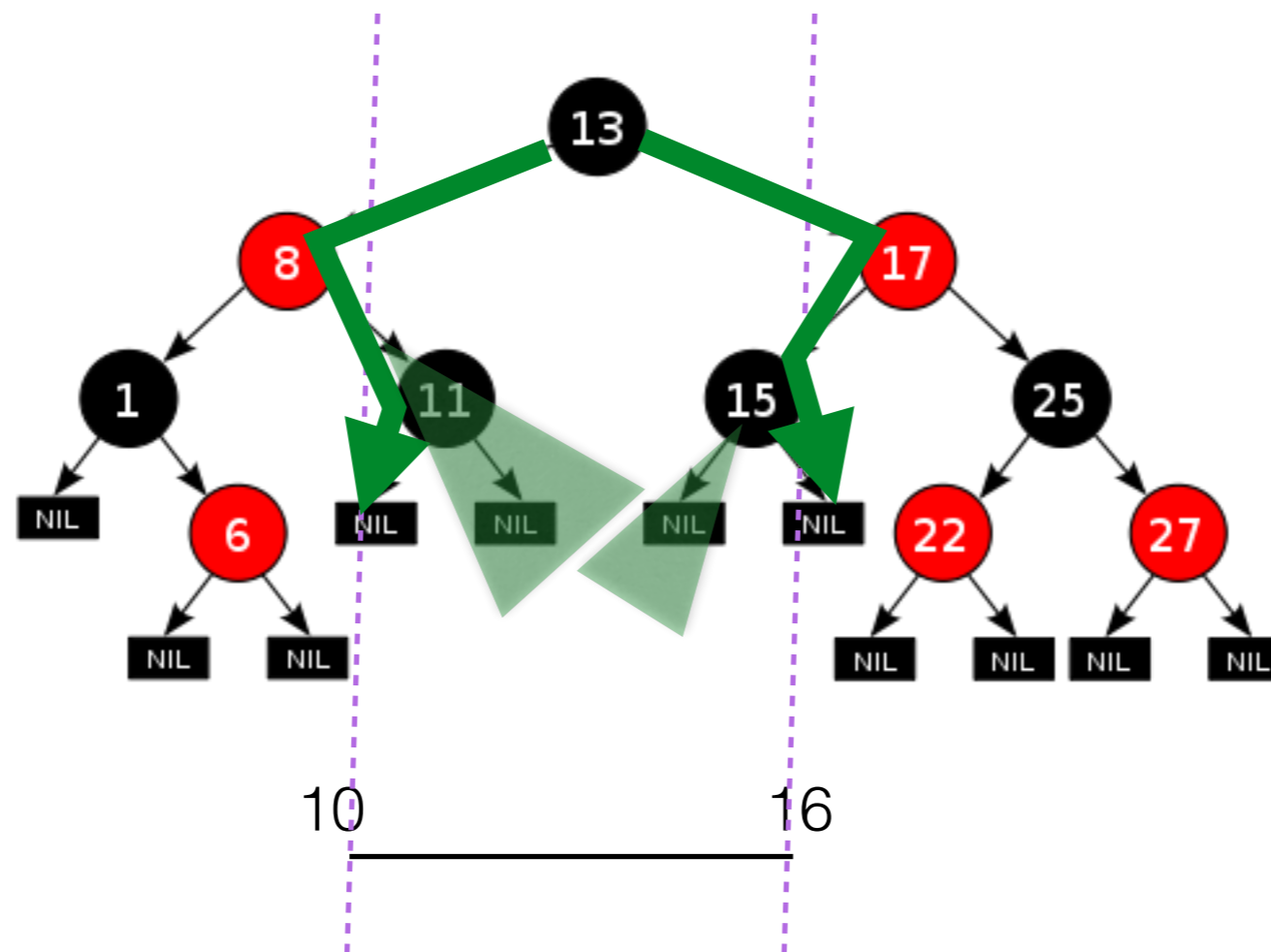
1D range searching with Binary Search Trees

Example: `range_search(21, 53)`: return 21, 34, 35, 46, 51, 52



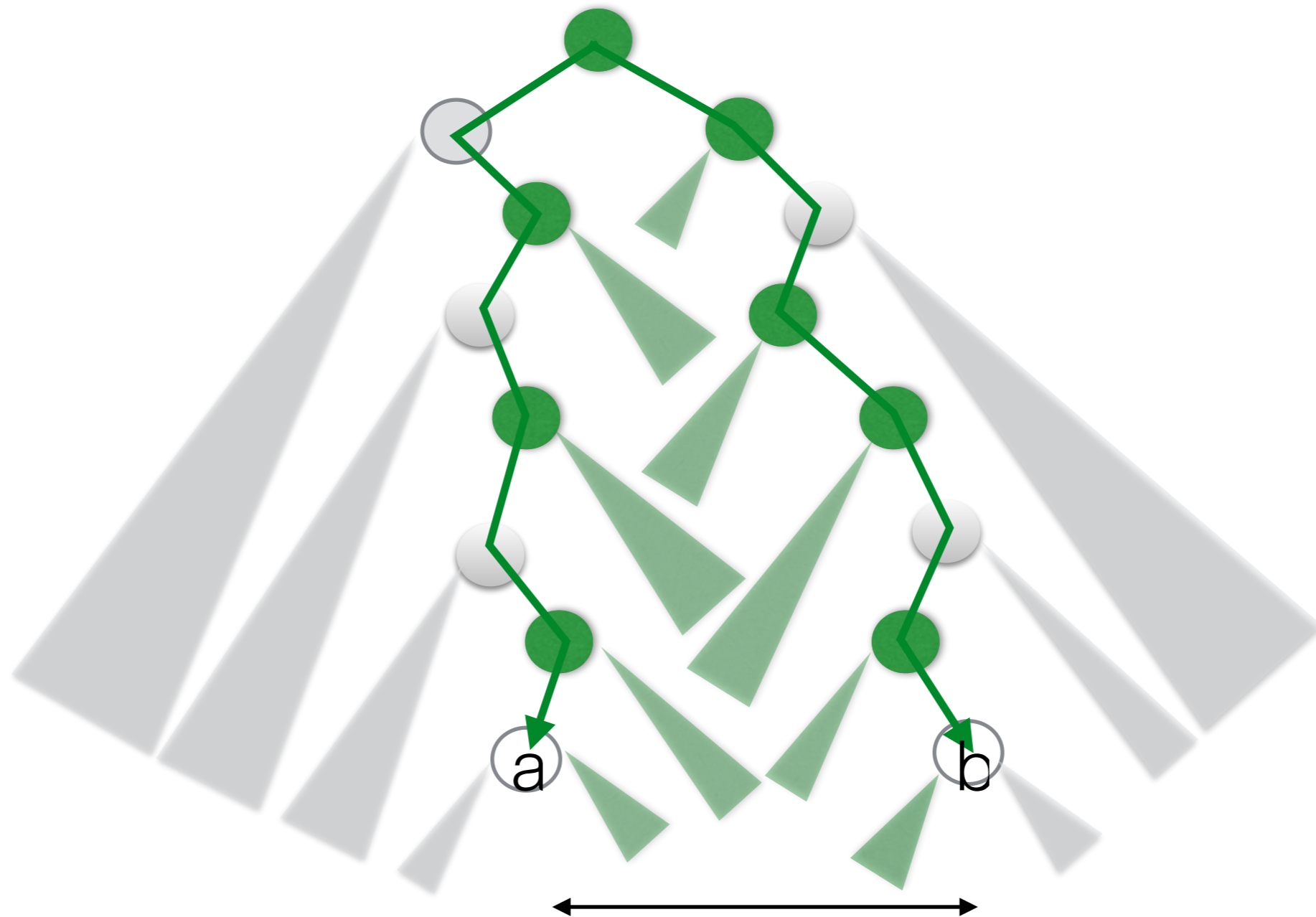
1D Range Searching with Red-Black Trees

Example: `range_search(10, 16)`: return 11, 13, 15



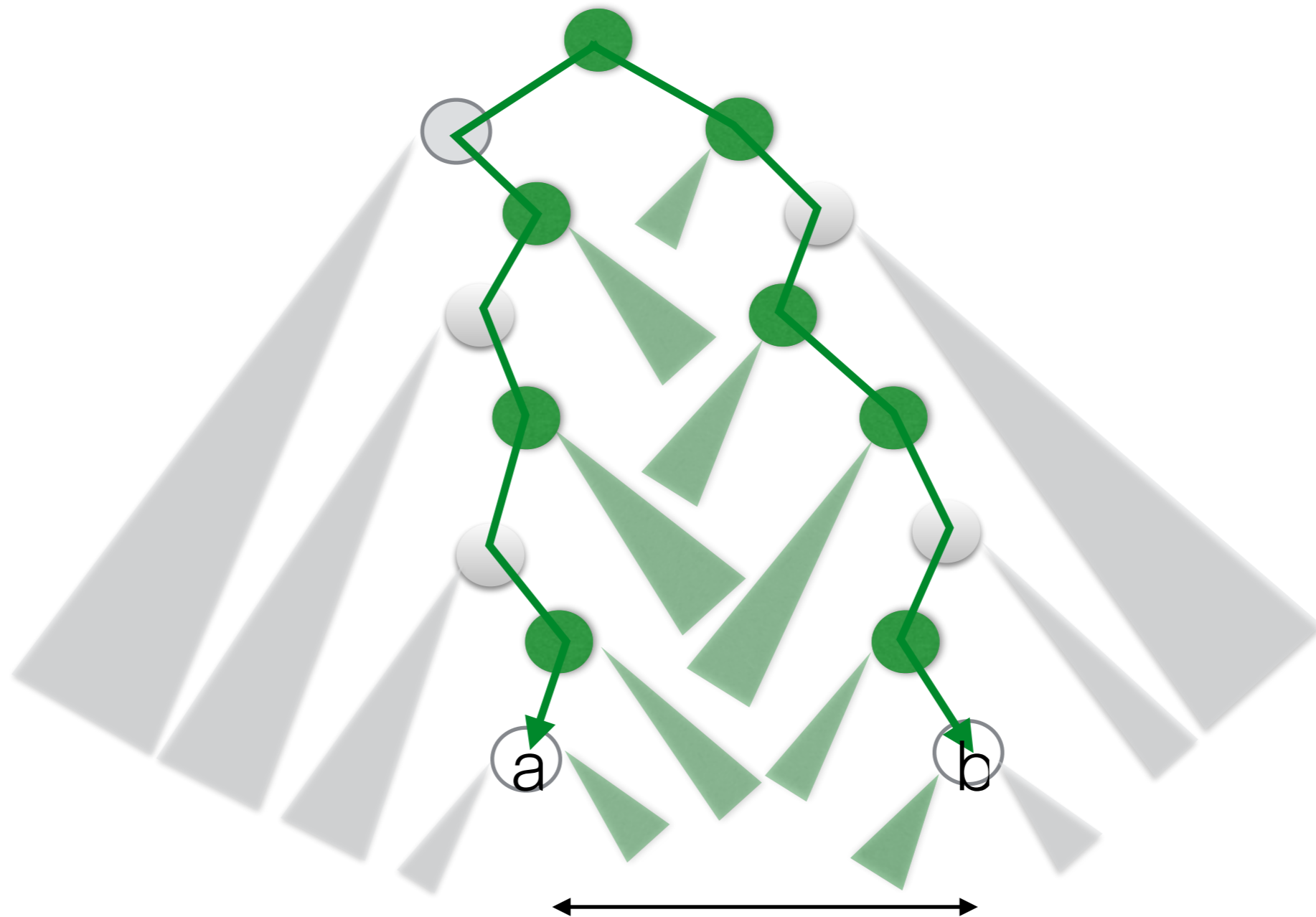
1D range searching with Binary Search Trees

- Range search (a,b): return all elements in this interval

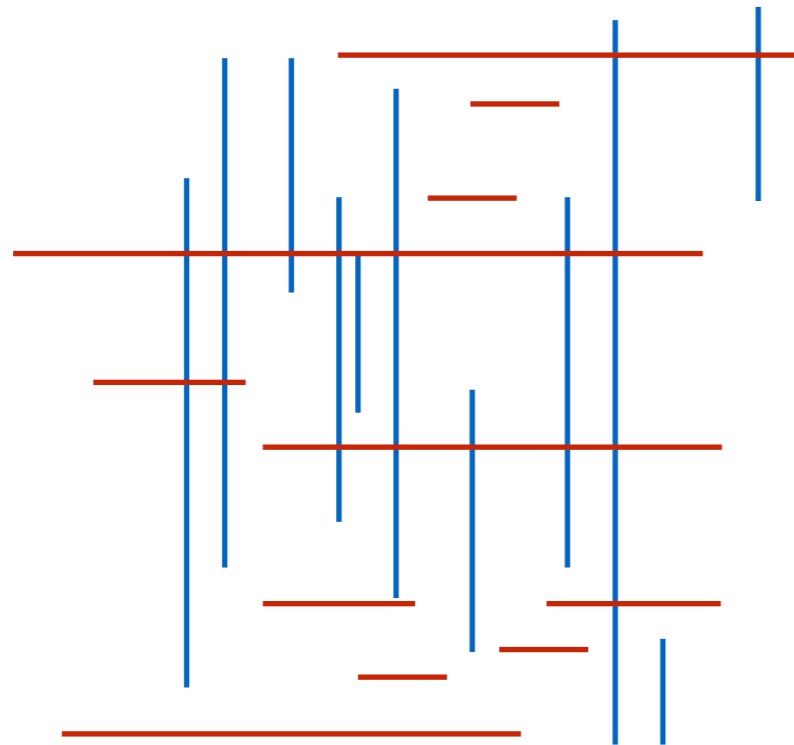


1D range searching with Binary Search Trees

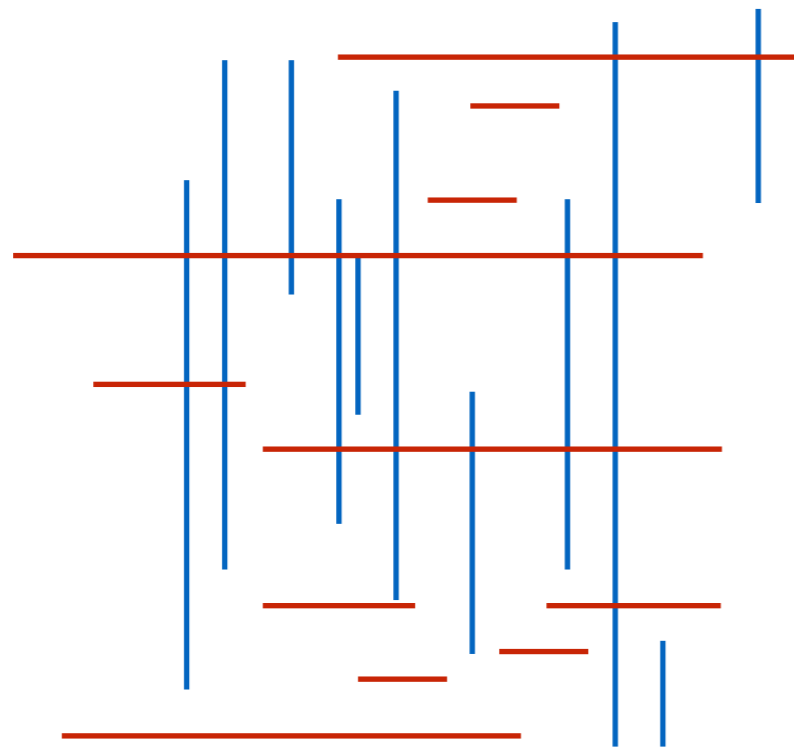
- Range search (a,b): return all elements in this interval
- Can be answered in $O(\lg n + k)$, where $k = O(n)$ is the size of output



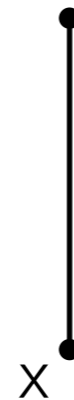
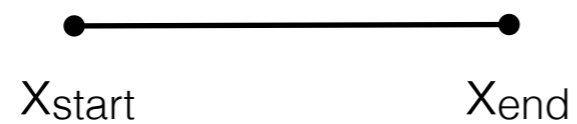
Orthogonal line segment intersection



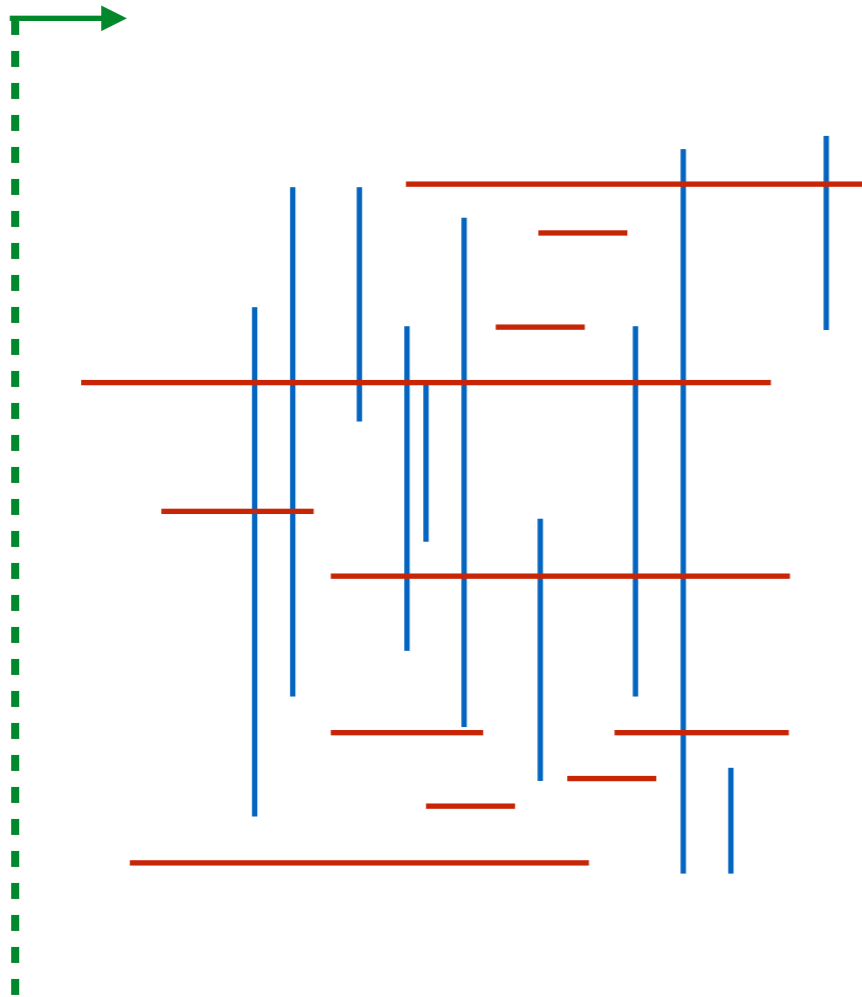
Orthogonal line segment intersection



- Let X be the set of x -coordinates of all segments //the “events”



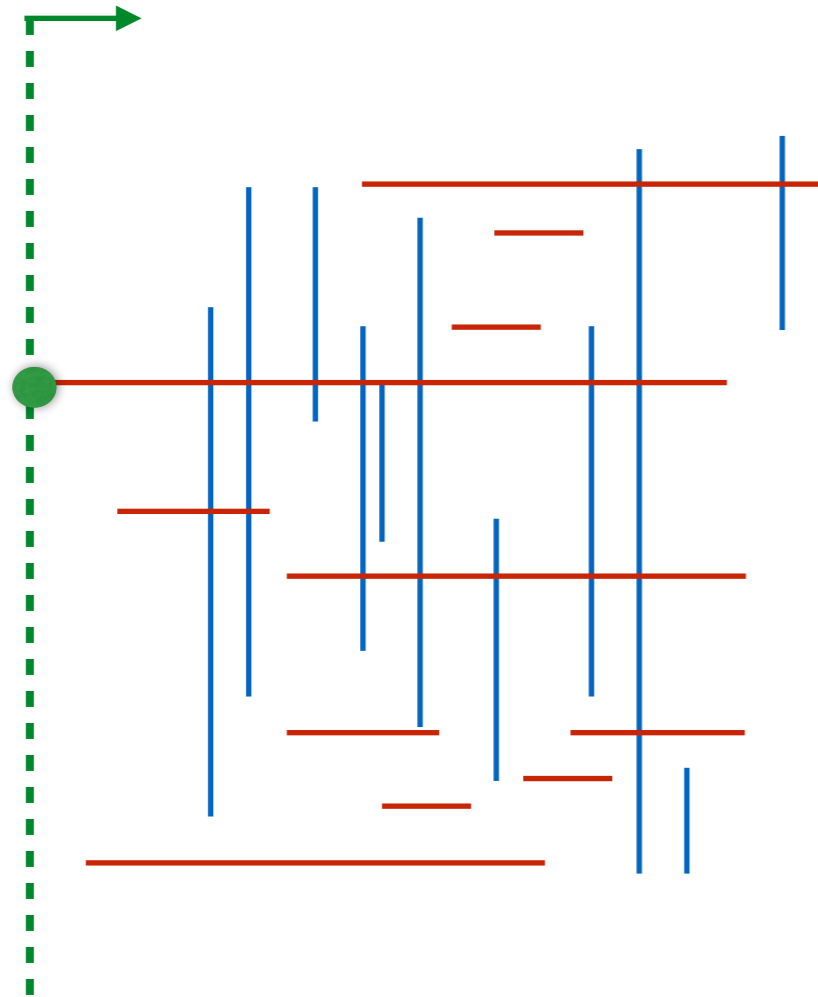
Orthogonal line segment intersection



line sweep technique
solve the problem behind the line

- Let X be the set of x-coordinates of all segments //our “events”
- Sort X and traverse the events in order

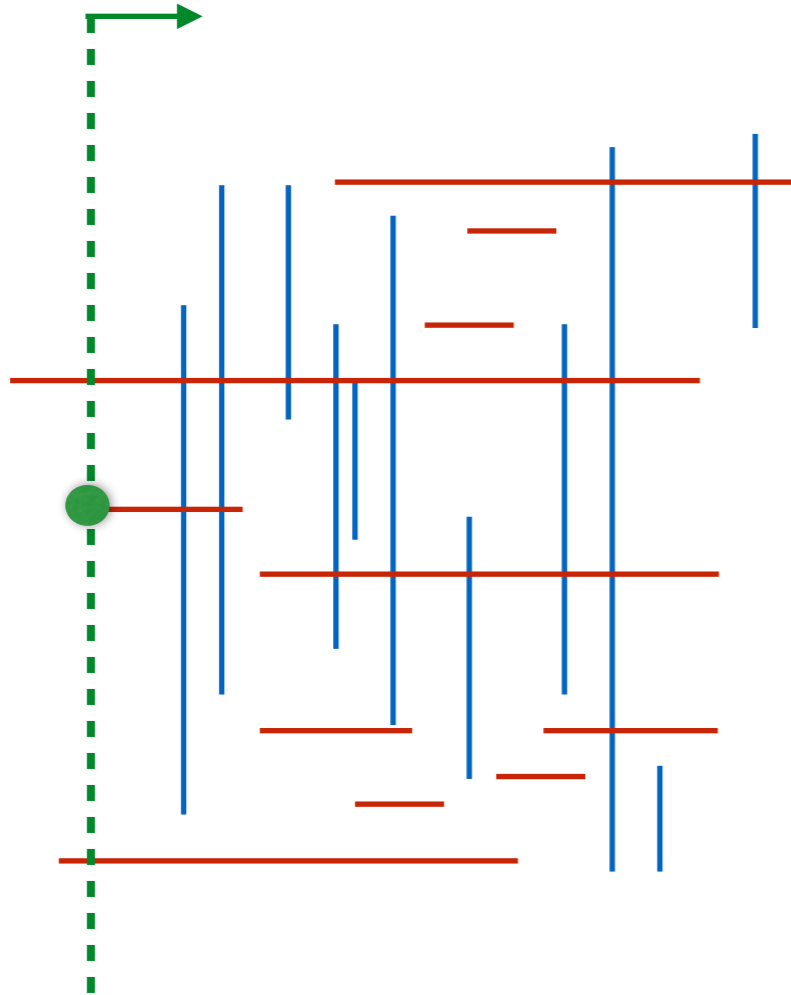
Orthogonal line segment intersection



line sweep technique
solve the problem behind the line

- Let X be the set of x-coordinates of all segments //our “events”
- Sort X and traverse the events in order

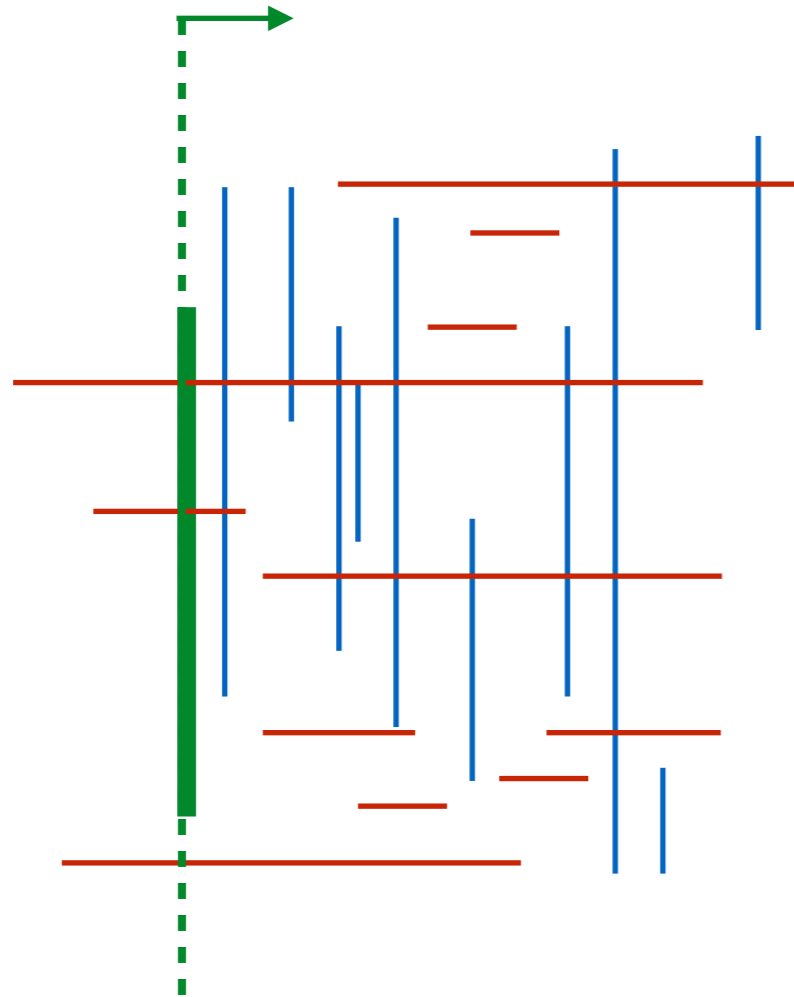
Orthogonal line segment intersection



line sweep technique
solve the problem behind the line

- Let X be the set of x-coordinates of all segments //our “events”
- Sort X and traverse the events in order

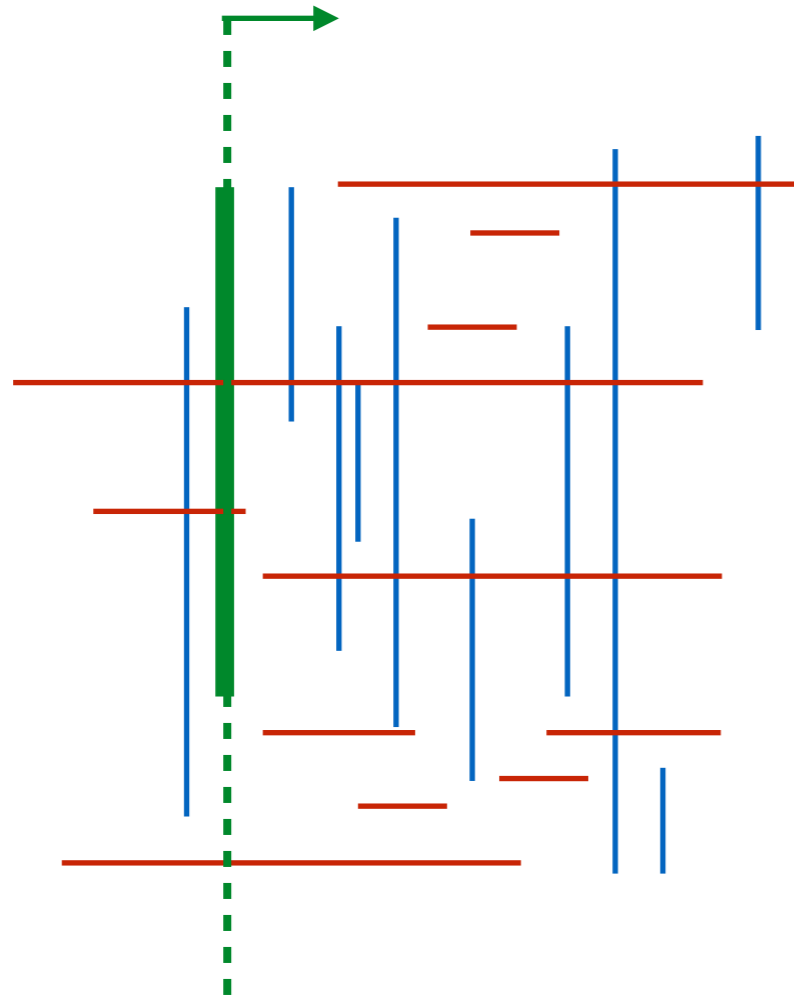
Orthogonal line segment intersection



line sweep technique
solve the problem behind the line

- Let X be the set of x -coordinates of all segments //our “events”
- Sort X and traverse the events in order

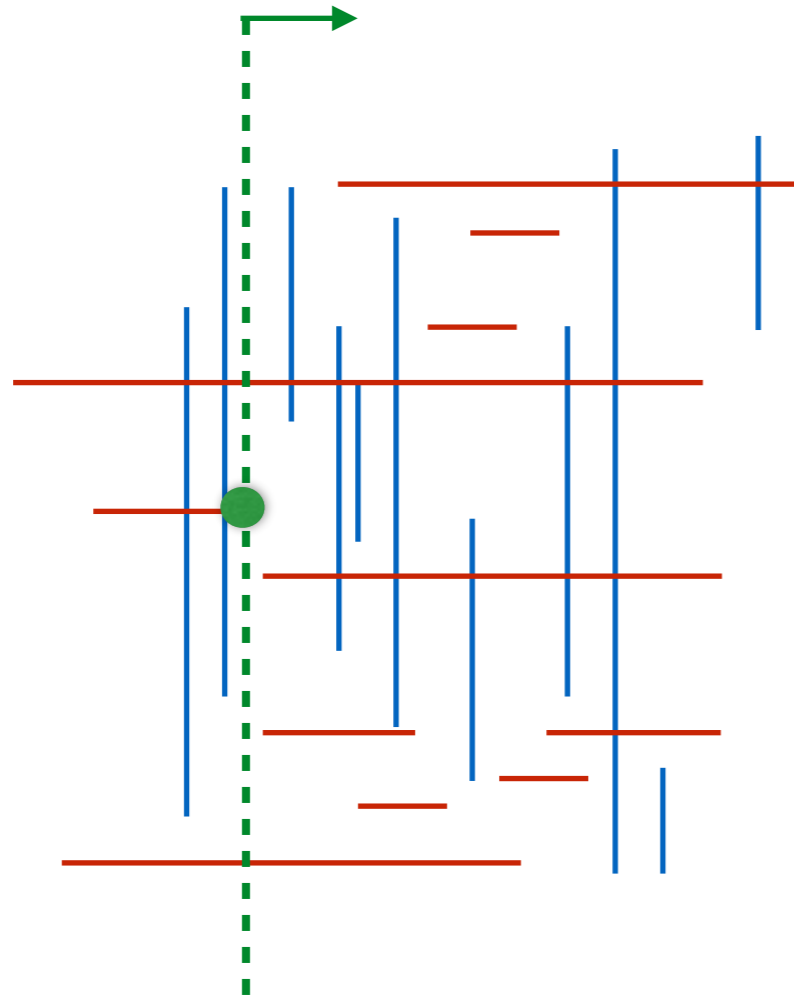
Orthogonal line segment intersection



line sweep technique
solve the problem behind the line

- Let X be the set of x -coordinates of all segments //our “events”
- Sort X and traverse the events in order

Orthogonal line segment intersection



line sweep technique
solve the problem behind the line

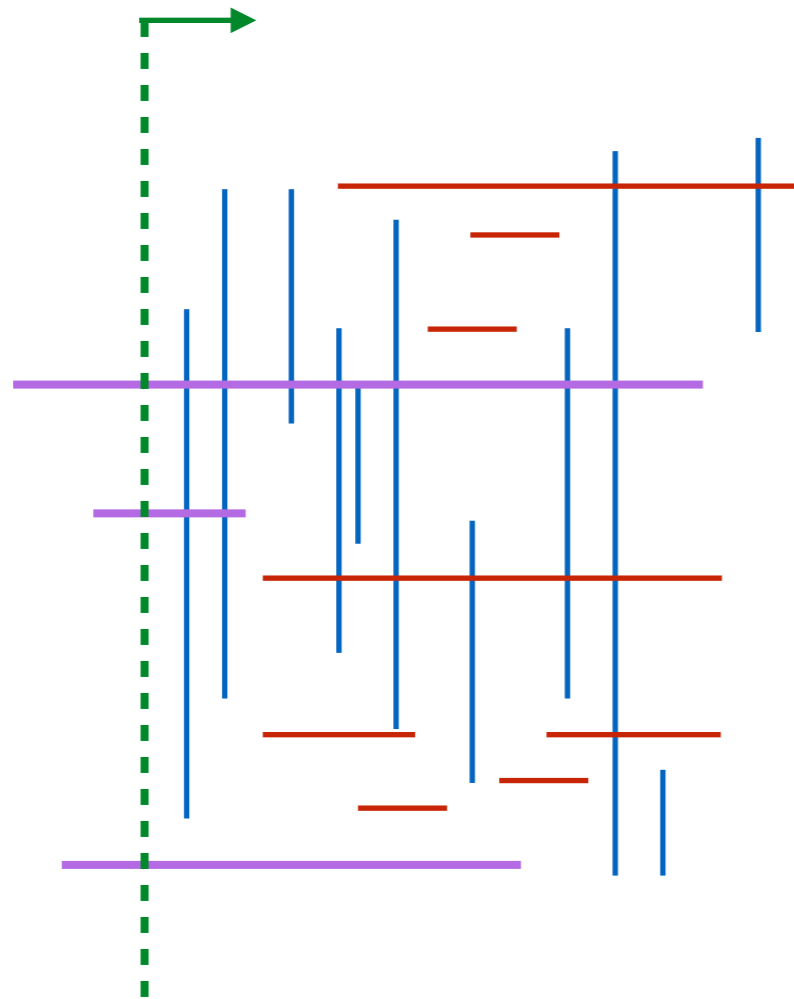
Events

beginning of a horizontal segment

end of a horizontal segment

vertical segment

Orthogonal line segment intersection



Events

beginning of a horizontal segment

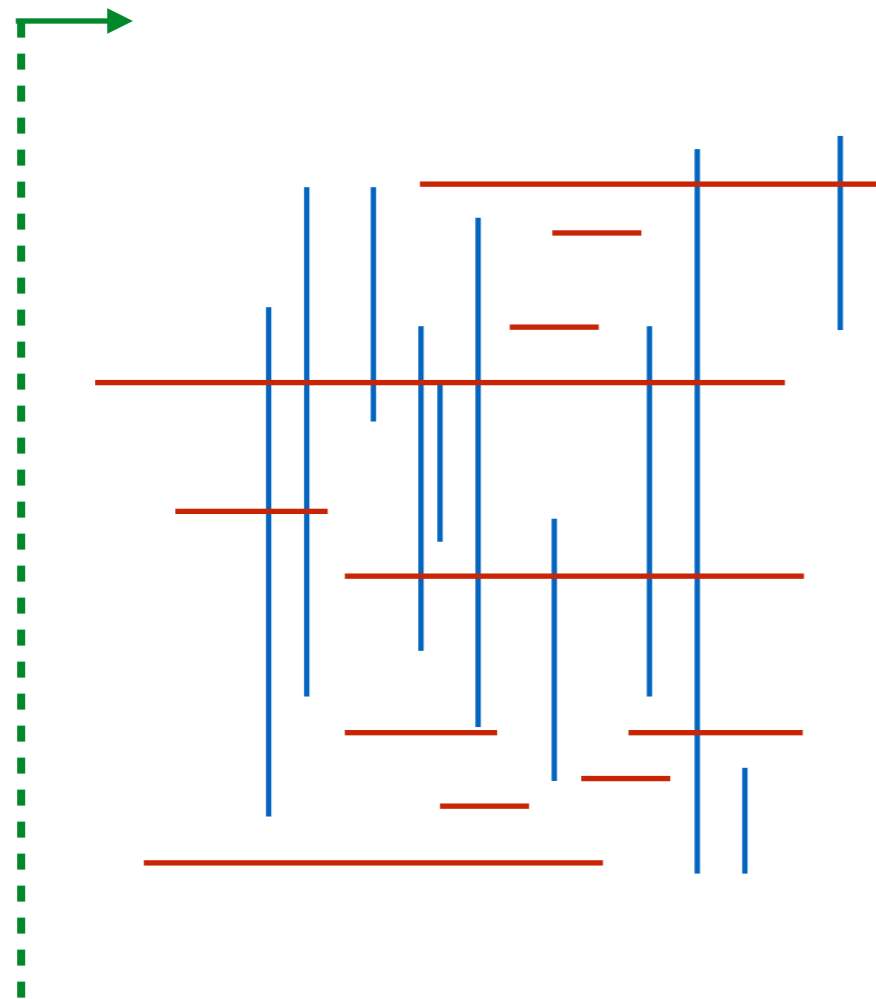
end of a horizontal segment

vertical segment

Line sweep technique

- Events
- Traverse events in order and maintain an Active Structure (AS)
 - AS contains objects that are “active” (started but not ended) in other words they are intersected by the present sweep line
 - at certain events, insert in AS
 - at certain events, delete from AS
 - at other events, query AS

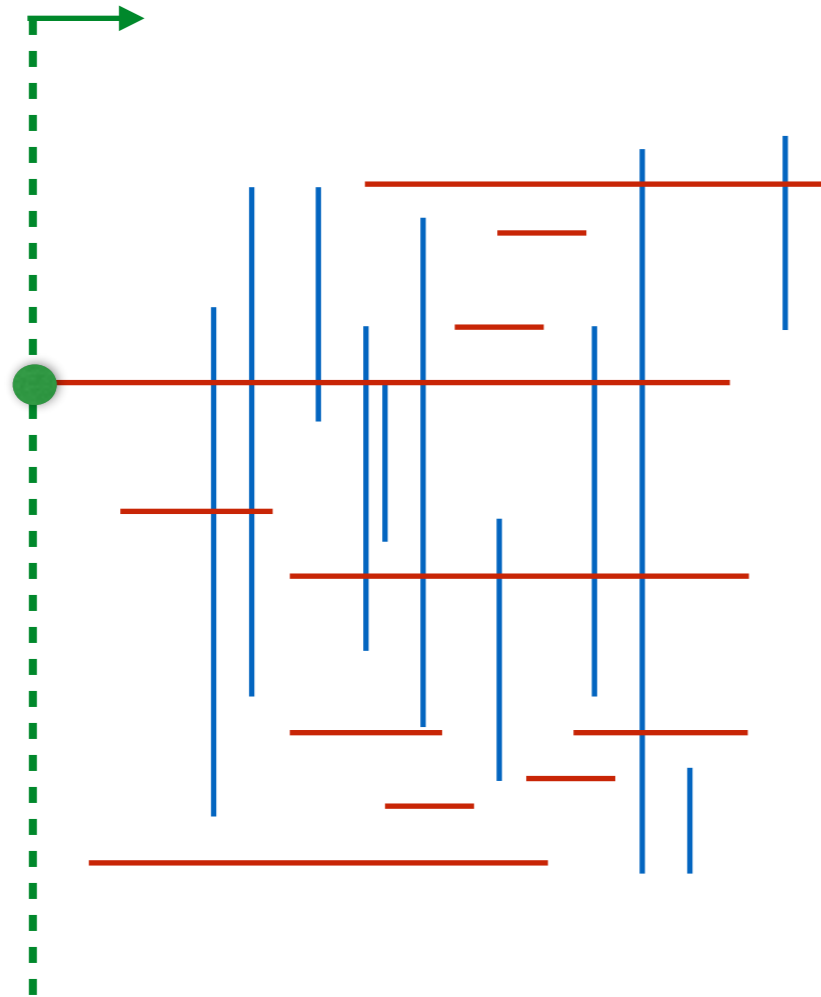
Orthogonal line segment intersection



AS=?
in order to do this efficiently

- Let X be the set of x-coordinates of all segments //the events
- Initialize $AS = \{\}$
- Sort X and traverse the events in sorted order; let x be the next event in X
 - if x is start of horizontal segment (x, x', y) :
//segment becomes active
insert segment (x,x',y) in AS
 - if x is end of horizontal segment (x, x', y) :
//segment stops being active
delete segment (x,x',y) from AS
 - if x corresponds to a vertical segment (y, y', x) :
//All active segments start before x and end after x . We need those whose y is in $[y,y']$
search AS for all segments with y -value in given range $[y,y']$ and report intersections

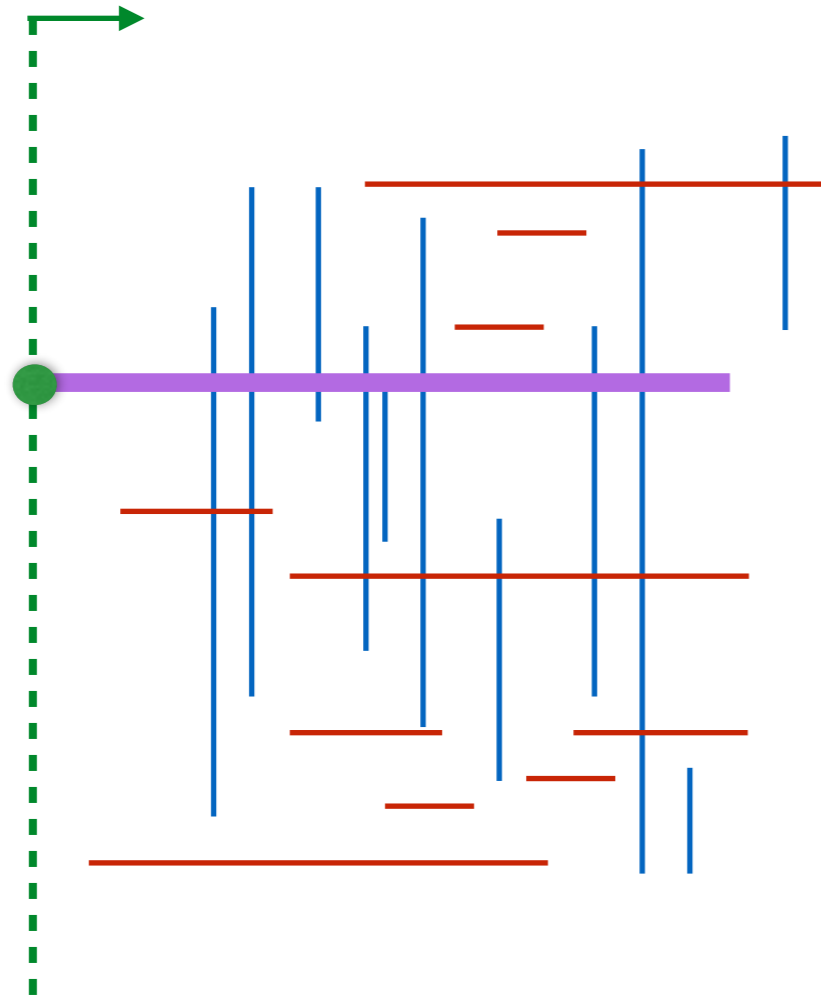
Orthogonal line segment intersection



AS=?
in order to do this efficiently

- Let X be the set of x -coordinates of all segments
//the events
- Initialize $AS = \{\}$
- Sort X and traverse the events in sorted order; let x be the next event in X
 - if x is start of horizontal segment (x, x', y) :
//segment becomes active
insert segment (x, x', y) in AS
 - if x is end of horizontal segment (x, x', y) :
//segment stops being active
delete segment (x, x', y) from AS
 - if x corresponds to a vertical segment (y, y', x) :
//All active segments start before x and end after x . We need those whose y is in $[y, y']$
search AS for all segments with y -value in given range $[y, y']$ and report intersections

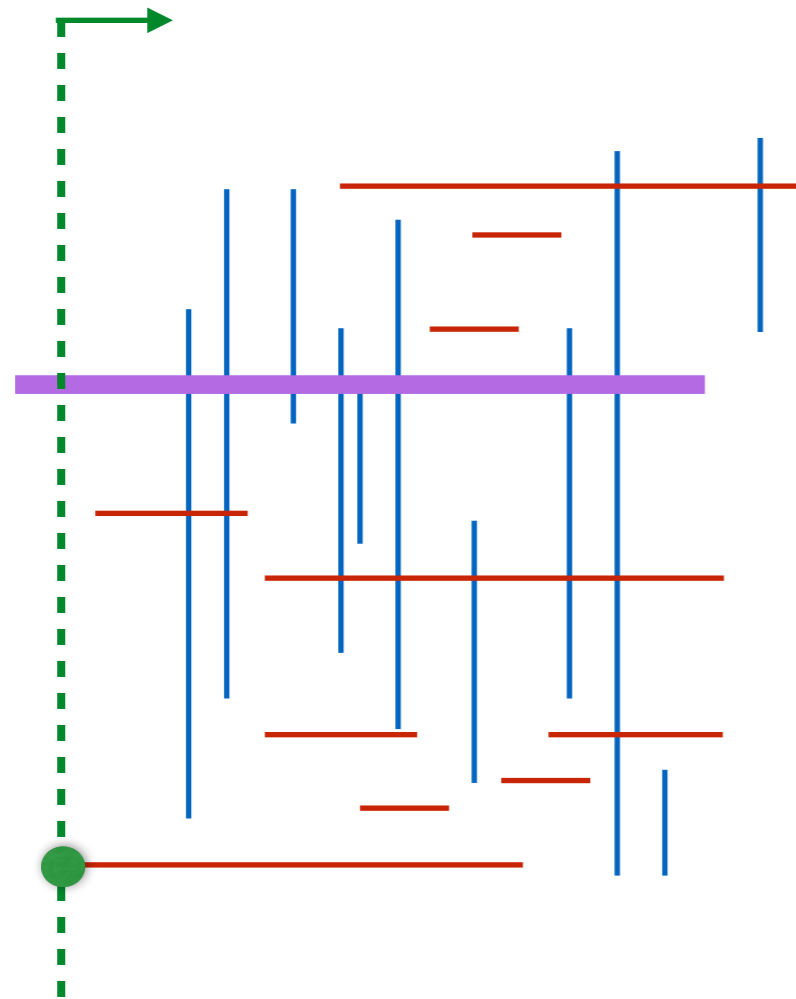
Orthogonal line segment intersection



AS=?
in order to do this efficiently

- Let X be the set of x -coordinates of all segments
//the events
- Initialize $AS = \{\}$
- Sort X and traverse the events in sorted order; let x be the next event in X
 - if x is start of horizontal segment (x, x', y) :
//segment becomes active
insert segment (x, x', y) in AS
 - if x is end of horizontal segment (x, x', y) :
//segment stops being active
delete segment (x, x', y) from AS
 - if x corresponds to a vertical segment (y, y', x) :
//All active segments start before x and end after x . We need those whose y is in $[y, y']$
search AS for all segments with y -value in given range $[y, y']$ and report intersections

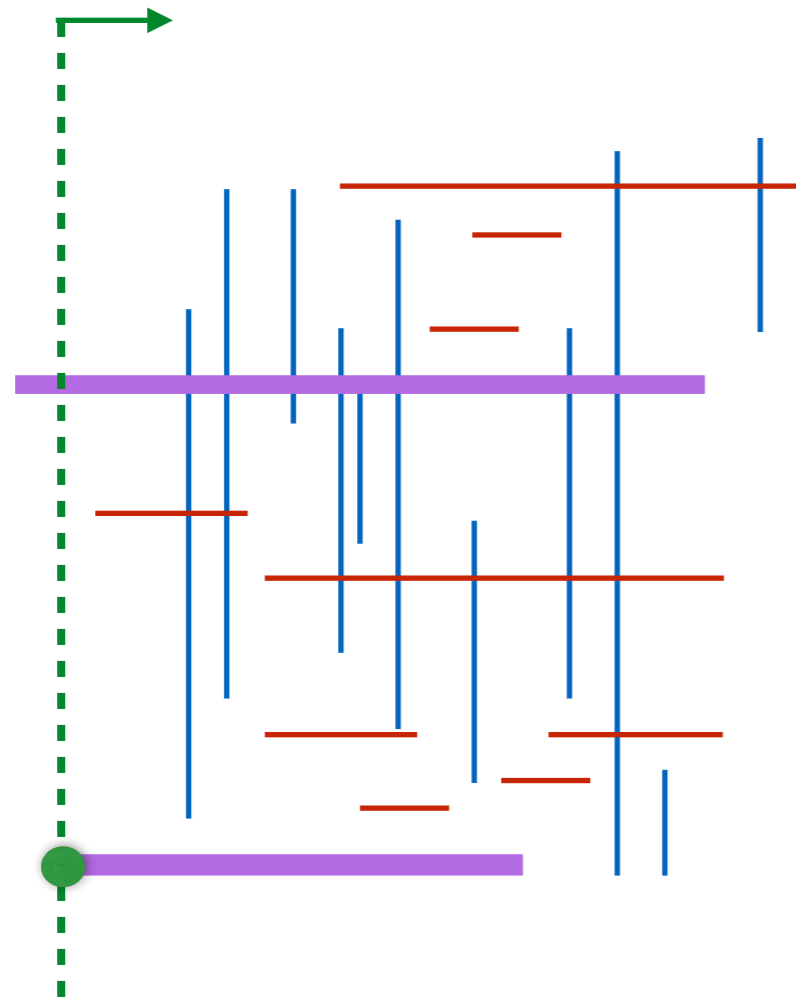
Orthogonal line segment intersection



AS=?
in order to do this efficiently

- Let X be the set of x -coordinates of all segments
//the events
- Initialize $AS = \{\}$
- Sort X and traverse the events in sorted order; let x be the next event in X
 - if x is start of horizontal segment (x, x', y) :
//segment becomes active
insert segment (x, x', y) in AS
 - if x is end of horizontal segment (x, x', y) :
//segment stops being active
delete segment (x, x', y) from AS
 - if x corresponds to a vertical segment (y, y', x) :
//All active segments start before x and end after x . We need those whose y is in $[y, y']$
search AS for all segments with y -value in given range $[y, y']$ and report intersections

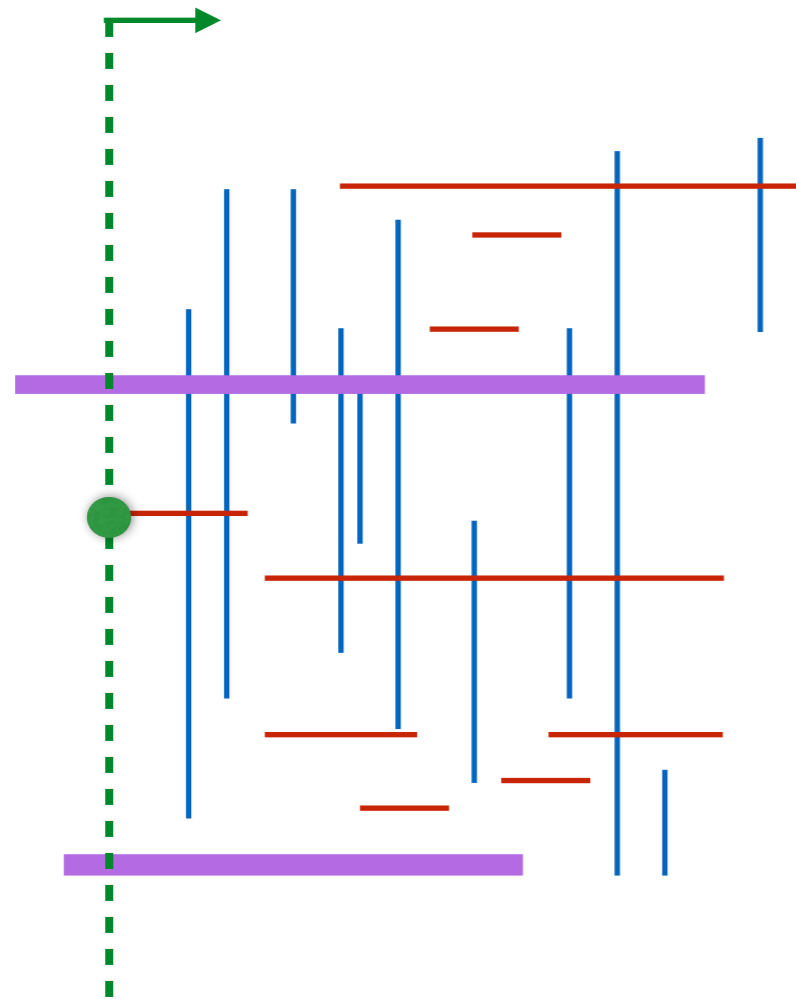
Orthogonal line segment intersection



AS=?
in order to do this efficiently

- Let X be the set of x -coordinates of all segments
//the events
- Initialize $AS = \{\}$
- Sort X and traverse the events in sorted order; let x be the next event in X
 - if x is start of horizontal segment (x, x', y) :
//segment becomes active
insert segment (x, x', y) in AS
 - if x is end of horizontal segment (x, x', y) :
//segment stops being active
delete segment (x, x', y) from AS
 - if x corresponds to a vertical segment (y, y', x) :
//All active segments start before x and end after x . We need those whose y is in $[y, y']$
search AS for all segments with y -value in given range $[y, y']$ and report intersections

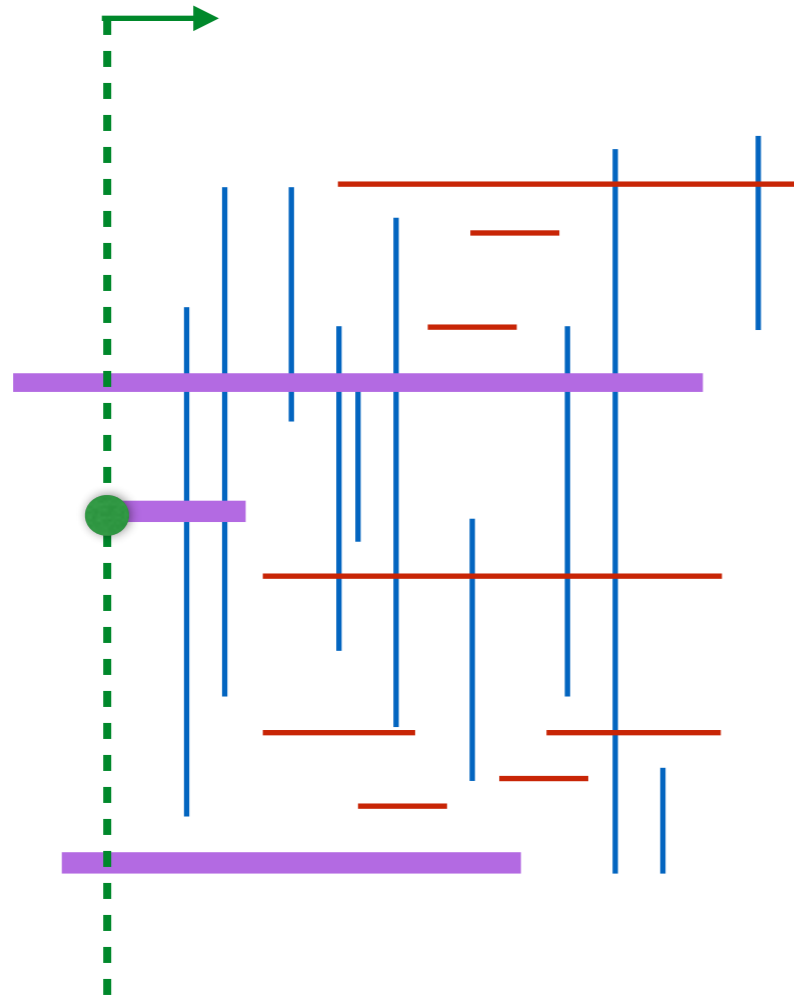
Orthogonal line segment intersection



AS=?
in order to do this efficiently

- Let X be the set of x -coordinates of all segments
//the events
- Initialize $AS = \{\}$
- Sort X and traverse the events in sorted order; let x be the next event in X
 - if x is start of horizontal segment (x, x', y) :
//segment becomes active
insert segment (x, x', y) in AS
 - if x is end of horizontal segment (x, x', y) :
//segment stops being active
delete segment (x, x', y) from AS
 - if x corresponds to a vertical segment (y, y', x) :
//All active segments start before x and end after x . We need those whose y is in $[y, y']$
search AS for all segments with y -value in given range $[y, y']$ and report intersections

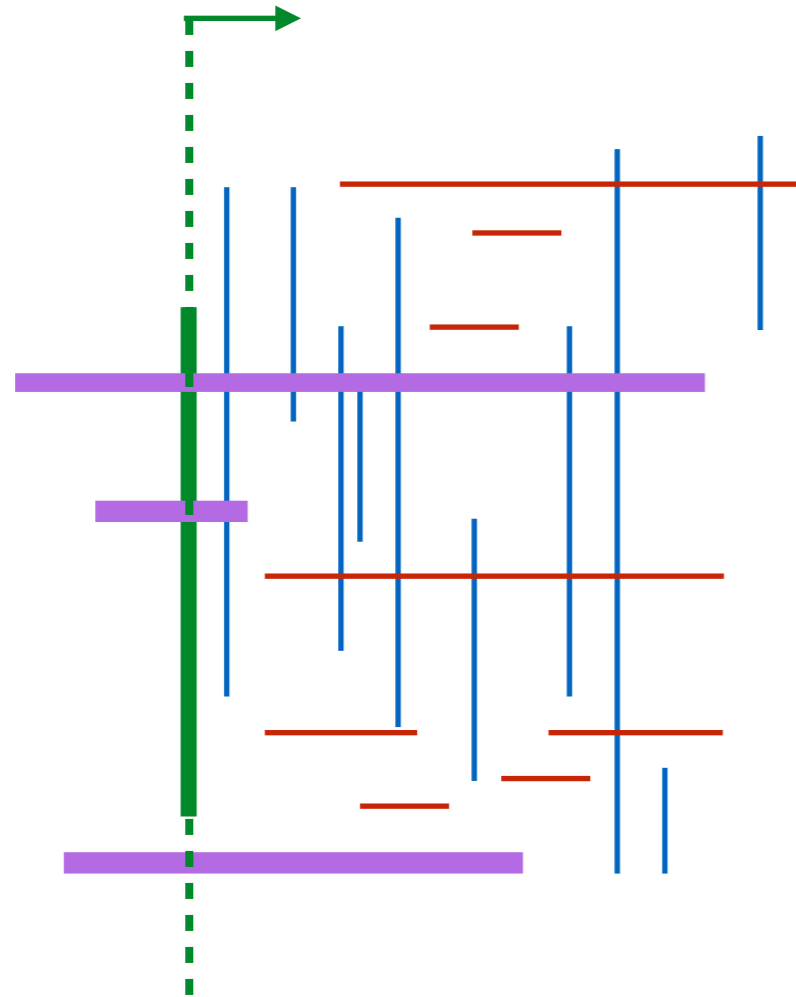
Orthogonal line segment intersection



AS=?
in order to do this efficiently

- Let X be the set of x -coordinates of all segments
//the events
- Initialize $AS = \{\}$
- Sort X and traverse the events in sorted order; let x be the next event in X
 - if x is start of horizontal segment (x, x', y) :
//segment becomes active
insert segment (x, x', y) in AS
 - if x is end of horizontal segment (x, x', y) :
//segment stops being active
delete segment (x, x', y) from AS
 - if x corresponds to a vertical segment (y, y', x) :
//All active segments start before x and end after x . We need those whose y is in $[y, y']$
search AS for all segments with y -value in given range $[y, y']$ and report intersections

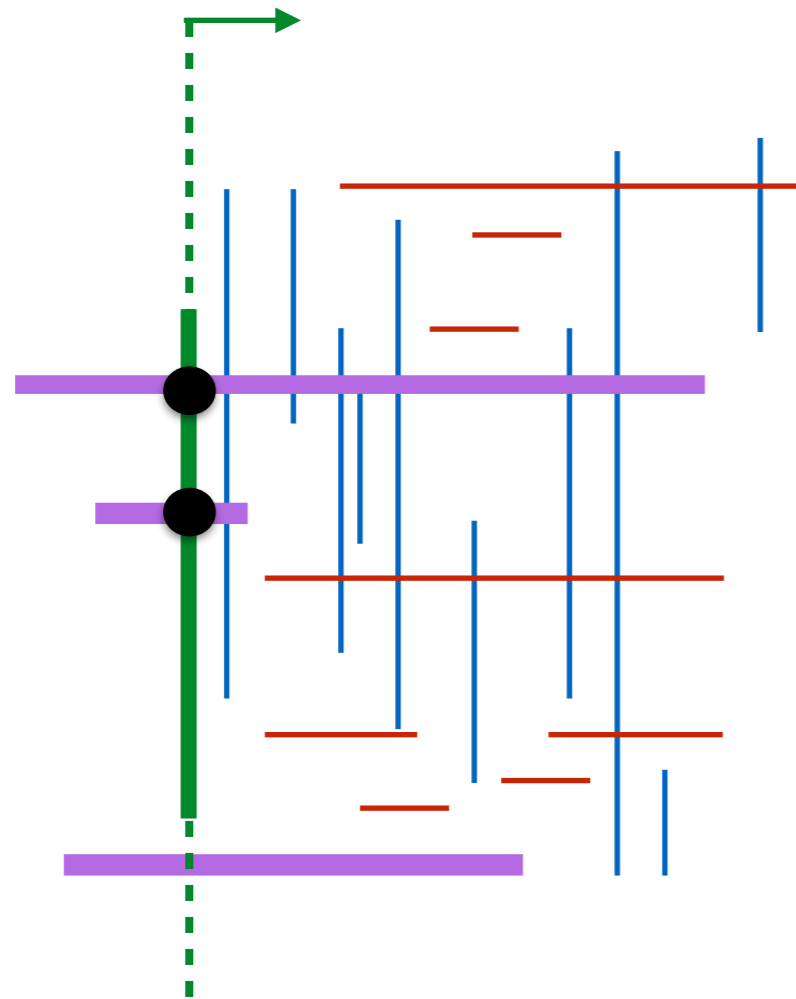
Orthogonal line segment intersection



AS=?
in order to do this efficiently

- Let X be the set of x -coordinates of all segments
//the events
- Initialize $AS = \{\}$
- Sort X and traverse the events in sorted order; let x be the next event in X
 - if x is start of horizontal segment (x, x', y) :
//segment becomes active
insert segment (x, x', y) in AS
 - if x is end of horizontal segment (x, x', y) :
//segment stops being active
delete segment (x, x', y) from AS
 - if x corresponds to a vertical segment (y, y', x) :
//All active segments start before x and end after x . We need those whose y is in $[y, y']$
search AS for all segments with y -value in given range $[y, y']$ and report intersections

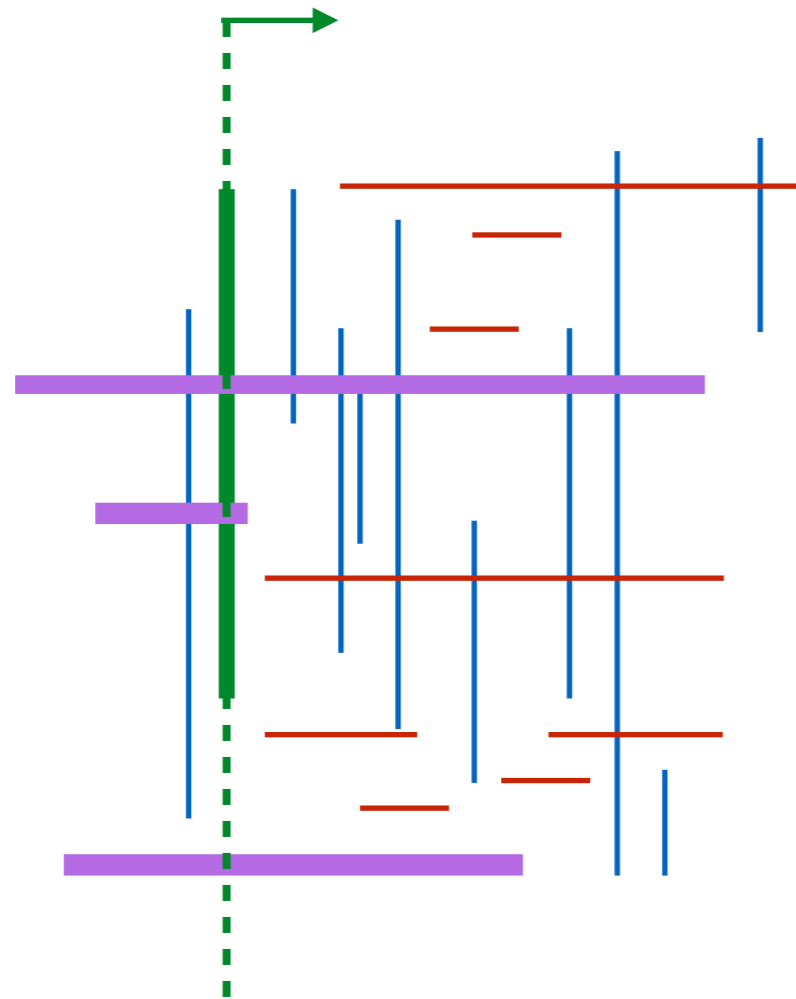
Orthogonal line segment intersection



AS=?
in order to do this efficiently

- Let X be the set of x -coordinates of all segments
//the events
- Initialize $AS = \{\}$
- Sort X and traverse the events in sorted order; let x be the next event in X
 - if x is start of horizontal segment (x, x', y) :
//segment becomes active
insert segment (x, x', y) in AS
 - if x is end of horizontal segment (x, x', y) :
//segment stops being active
delete segment (x, x', y) from AS
 - if x corresponds to a vertical segment (y, y', x) :
//All active segments start before x and end after x . We need those whose y is in $[y, y']$
search AS for all segments with y -value in given range $[y, y']$ and report intersections

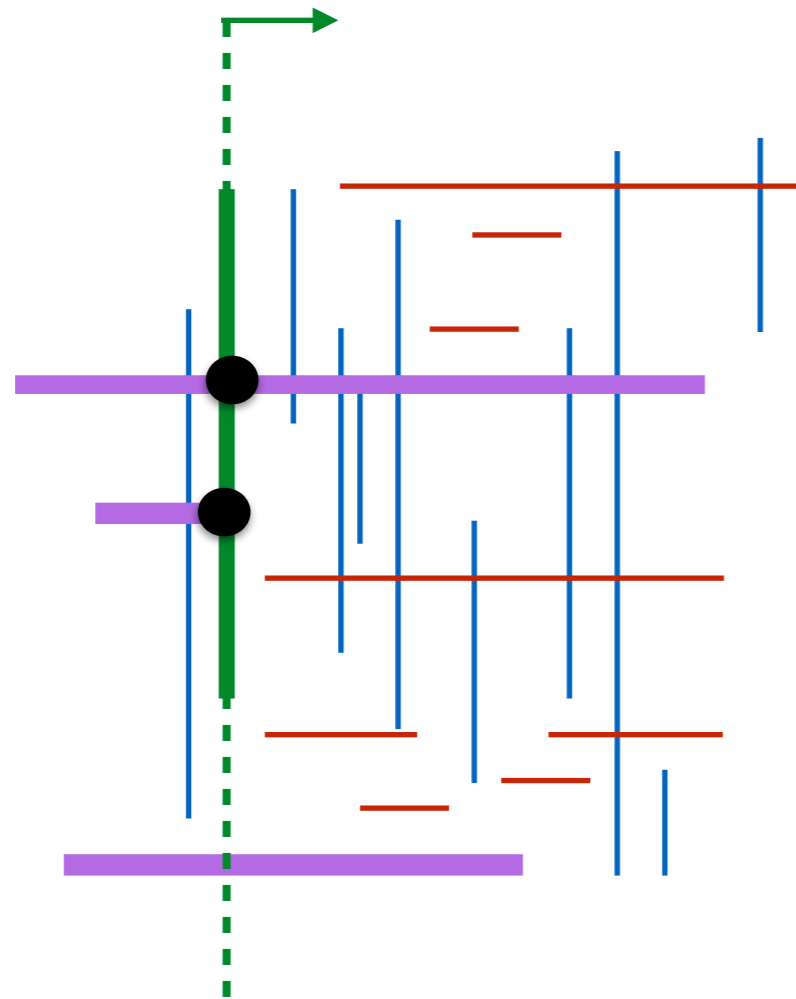
Orthogonal line segment intersection



AS=?
in order to do this efficiently

- Let X be the set of x -coordinates of all segments
//the events
- Initialize $AS = \{\}$
- Sort X and traverse the events in sorted order; let x be the next event in X
 - if x is start of horizontal segment (x, x', y) :
//segment becomes active
insert segment (x, x', y) in AS
 - if x is end of horizontal segment (x, x', y) :
//segment stops being active
delete segment (x, x', y) from AS
 - if x corresponds to a vertical segment (y, y', x) :
//All active segments start before x and end after x . We need those whose y is in $[y, y']$
search AS for all segments with y -value in given range $[y, y']$ and report intersections

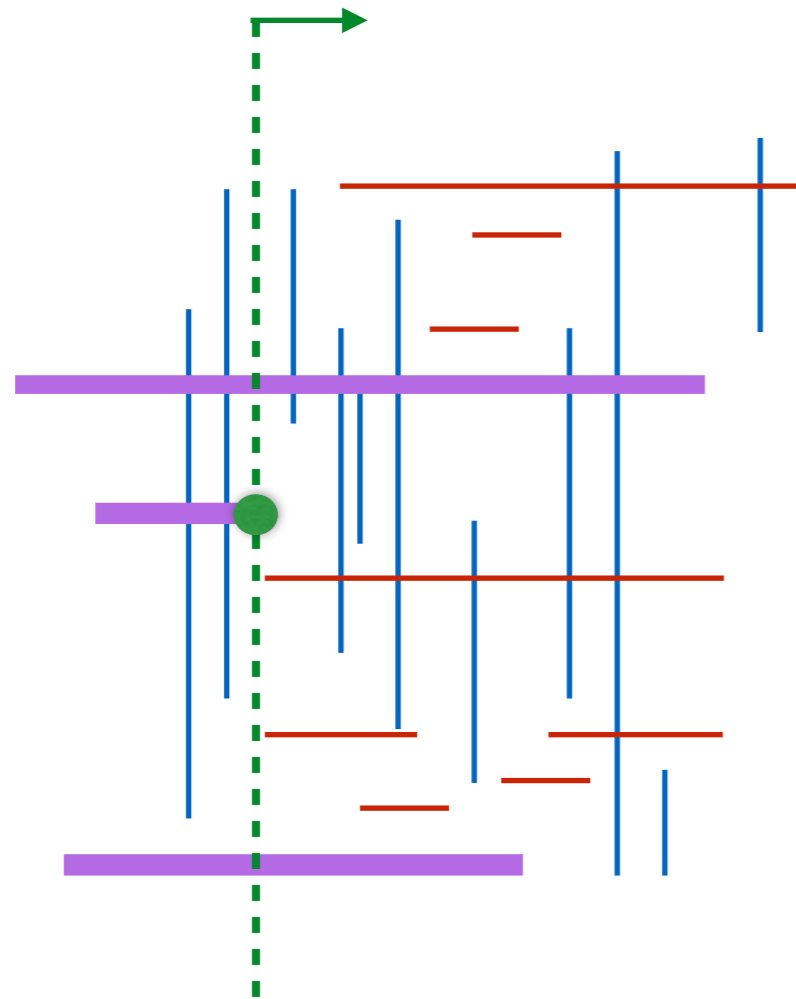
Orthogonal line segment intersection



AS=?
in order to do this efficiently

- Let X be the set of x -coordinates of all segments
//the events
- Initialize $AS = \{\}$
- Sort X and traverse the events in sorted order; let x be the next event in X
 - if x is start of horizontal segment (x, x', y) :
//segment becomes active
insert segment (x, x', y) in AS
 - if x is end of horizontal segment (x, x', y) :
//segment stops being active
delete segment (x, x', y) from AS
 - if x corresponds to a vertical segment (y, y', x) :
//All active segments start before x and end after x . We need those whose y is in $[y, y']$
search AS for all segments with y -value in given range $[y, y']$ and report intersections

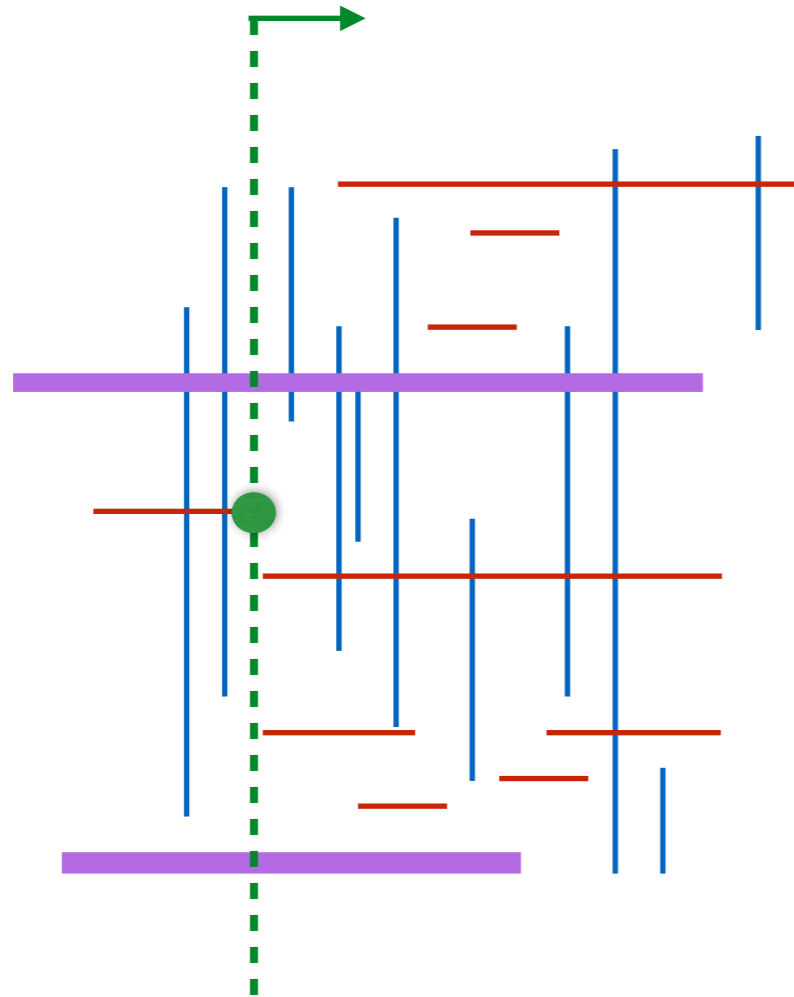
Orthogonal line segment intersection



AS=?
in order to do this efficiently

- Let X be the set of x -coordinates of all segments
//the events
- Initialize $AS = \{\}$
- Sort X and traverse the events in sorted order; let x be the next event in X
 - if x is start of horizontal segment (x, x', y) :
//segment becomes active
insert segment (x, x', y) in AS
 - if x is end of horizontal segment (x, x', y) :
//segment stops being active
delete segment (x, x', y) from AS
 - if x corresponds to a vertical segment (y, y', x) :
//All active segments start before x and end after x . We need those whose y is in $[y, y']$
search AS for all segments with y -value in given range $[y, y']$ and report intersections

Orthogonal line segment intersection



AS=?
in order to do this efficiently

- Let X be the set of x -coordinates of all segments
//the events
- Initialize $AS = \{\}$
- Sort X and traverse the events in sorted order; let x be the next event in X
 - if x is start of horizontal segment (x, x', y) :
//segment becomes active
insert segment (x, x', y) in AS
 - if x is end of horizontal segment (x, x', y) :
//segment stops being active
delete segment (x, x', y) from AS
 - if x corresponds to a vertical segment (y, y', x) :
//All active segments start before x and end after x . We need those whose y is in $[y, y']$
search AS for all segments with y -value in given range $[y, y']$ and report intersections

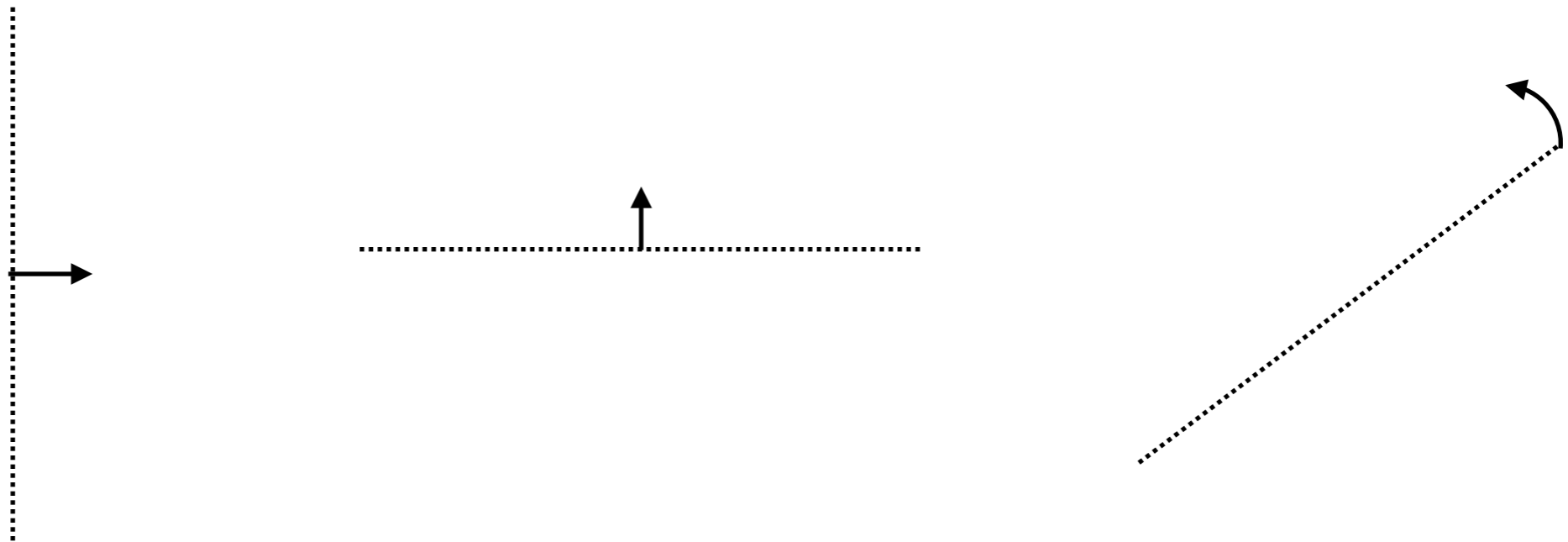
Orthogonal line segment intersection

- Pick an example and simulate the algorithm
- How do you implement the AS?
- Analysis?

- Let X be the set of x -coordinates of all segments
//the events
- Initialize $AS = \{\}$
- Sort X and traverse the events in sorted order; let x be the next event in X
 - if x is start of horizontal segment (x, x', y) :
//segment becomes active
insert segment (x, x', y) in AS
 - if x is end of horizontal segment (x, x', y) :
//segment stops being active
delete segment (x, x', y) from AS
 - if x corresponds to a vertical segment (y, y', x) :
//All active segments start before x and end after x . We need those whose y is in $[y, y']$
search AS for all segments with y -value in given range $[y, y']$ and report intersections

Line sweep

- Frequently used technique
- Line can be horizontal or vertical or radial or



- Traverse events in order and maintain an Active Structure (AS)
 - AS maintains objects that are “active” (started but not ended) in other words they are intersected by the present sweep line
 - at certain events, insert in AS
 - at certain events, delete from AS
 - at other events, query AS