Line segment intersection (I):
Orthogonal line segment intersection
Outline

• The problem (what)

• Applications (why)

• Algorithms (how)
  • A special case: Orthogonal line segments
  • General case: Bentley-Otman line sweep algorithm
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.
Line segment intersection:

Applications
Applications

Graphics: rendering $\Rightarrow$ hidden surfaces $\Rightarrow$ intersections
Applications

Motion planning and collision detection in autonomous systems/robotics
Applications

Geographic data: River networks, road networks, railways, ..
Applications

Map overlay in GIS

from: www.geo.hunter.cuny.edu/aierulli/gis2/lectures/Lecture2/fig9-30_raster_overlay.gif
Applications

Map overlay in GIS

from: www.geo.hunter.cuny.edu/aierulli/gis2/lectures/Lecture2/fig9-30_raster_overlay.gif
Applications

Geographic data: River networks, road networks, railways, ..
Computing line segment intersection:

Algorithms
**Naive**

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

**Class work:**
- 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?)

**Notation**
- $n$: size of the input (number of segments)
- $k$: size of output (number of intersections)
A special case: Orthogonal line segment intersection

Problem: Given a set of orthogonal line segments in 2D, find all their pairwise intersections.
A special case: Orthogonal line segment intersection

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

Exercises

• Come up with a straightforward algorithm and analyze its time
• Can you do better?
Balanced Binary Search Trees
- crash course -
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
  - 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:**
  - A Red-Black tree of \( n \) nodes has height \( \Theta(\lg n) \).
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 \( P = \{x_1, x_2, x_3, \ldots x_n \} \)
• Pre-process it in order to answer

\[
\text{rangeSearch}(a,b): \text{ return all elements in } P \text{ in interval } (a,b)
\]
1D Range Searching

- Given a set of values $P = \{x_1, x_2, x_3, \ldots x_n\}$
- Pre-process it in order to answer
  $$\text{rangeSearch}(a, b): \text{return all elements in } P \text{ in interval } (a, b)$$

- If $P$ is static
  
  $P$ is known ahead and does not change
1D Range Searching

- Given a set of values $P = \{x_1, x_2, x_3, \ldots x_n \}$
- Pre-process it in order to answer
  
  $\text{rangeSearch}(a, b)$: return all elements in $P$ in interval $(a, b)$

- If $P$ is static
  - sort, then binary search for $a$ and walk. $O(\lg n + k)$ per query
1D Range Searching

- Given a set of values $P = \{x_1, x_2, x_3, \ldots x_n\}$
- Pre-process it in order to answer
  
  $\text{rangeSearch}(a,b)$: return all elements in $P$ in interval $(a,b)$

- If $P$ is dynamic
  - use a BBST

  $P$ changes by adding and deleting values
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
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 \((a, b)\):
1D range searching with Binary Search Trees

- Range search \((a,b)\):
- Can be answered in \(O(\lg n+k)\), where \(k = O(n)\) is the size of output
Balanced Binary Search Trees
- end -
Orthogonal line segment intersection
Orthogonal line segment intersection

• Let X be the set of x-coordinates of all segments: these are the “events”

Events

beginning of a horizontal segment
end of a horizontal segment
vertical segment
Orthogonal line segment intersection

- Events: Let X be the set of x-coordinates of all segments. Sort X.
- Traverse the events in sorted order
Events: Let $X$ be the set of x-coordinates of all segments. Sort $X$.

Traverse the events in sorted order.
Orthogonal line segment intersection

- Events: Let X be the set of x-coordinates of all segments. Sort X.
- Traverse the events in sorted order
Orthogonal line segment intersection

- Events: Let X be the set of x-coordinates of all segments. Sort X.
- Traverse the events in sorted order
Orthogonal line segment intersection

Events: Let $X$ be the set of $x$-coordinates of all segments. Sort $X$.

• Traverse the events in sorted order
Orthogonal line segment intersection

• Events: Let X be the set of x-coordinates of all segments. Sort X.
• Traverse the events in sorted order
Orthogonal line segment intersection

- 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 current sweep line
- At some events, insert in AS
- At some events, delete from AS
- At some events, query AS

Events
- beginning of a horizontal segment
- end of a horizontal segment
- vertical segment

Line sweep technique
Orthogonal line segment intersection

- 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

$AS=?$
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
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

- 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

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

Initialize $AS = \emptyset$

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
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
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

$AS = ?$

in order to do this efficiently
Orthogonal line segment intersection

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

$AS = ?$
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

- 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

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

$AS=?$
in order to do this efficiently
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
Line sweep algorithms

- Powerful, elegant, frequently used technique
- Line can be horizontal or vertical or radial or ….

- 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 current sweep line
  - at some events, insert in AS
  - at some events, delete from AS
  - at some events, query AS