Plane Sweep

Pat Morin
COMP2402/2002

Carleton University
Input: Given a set $S$ of $n$ line segments
Line Segment Intersection Finding

- **Input:** Given a set $S$ of $n$ line segments
- **Output:** All pairs $s, t \in S$ such that $s$ intersects $t$
The Trivial Algorithm

The trivial algorithm:

1. for each \( s, t \in \binom{S}{2} \)
   - if \( s \) intersects \( t \) then add \((s, t)\) to the output

Running time is proportional to \( (n^2) = n(n-1)/2 = O(n^2) \)

Can we do better?

In the worst case, no, every pair in \( S \) might intersect

Then the size of the output is \( (n^2) = \Omega(n^2) \)
The trivial algorithm:

1. for each $s, t \in \binom{S}{2}$
   - if $s$ intersects $t$ then add $(s, t)$ to the output

Running time is proportional to $\binom{n}{2} = n(n - 1)/2 = O(n^2)$
The Trivial Algorithm

- The trivial algorithm:
  1. for each $s, t \in \binom{S}{2}$
     - if $s$ intersects $t$ then add $(s, t)$ to the output
- Running time is proportional to $\binom{n}{2} = n(n - 1)/2 = O(n^2)$
- Can we do better?
The Trivial Algorithm

- The trivial algorithm:
  1. for each \( s, t \in \binom{S}{2} \)
     - if \( s \) intersects \( t \) then add \((s, t)\) to the output
  - Running time is proportional to \( \binom{n}{2} = n(n-1)/2 = O(n^2) \)
  - Can we do better?
  - In the worst case, no, every pair in \( S \) might intersect
The Trivial Algorithm

The trivial algorithm:

1. for each \( s, t \in \binom{S}{2} \)
   - if \( s \) intersects \( t \) then add \((s, t)\) to the output

Running time is proportional to \( \binom{n}{2} = n(n-1)/2 = O(n^2) \)

Can we do better?

In the worst case, no, every pair in \( S \) might intersect

Then the size of the output is \( \binom{n}{2} = \Omega(n^2) \)
The $\Omega(n^2)$ lower-bound on the size of the output is unsatisfactory.
Output-Sensitive Algorithms

- The $\Omega(n^2)$ lower-bound on the size of the output is unsatisfactory
  - In many cases, the number of intersecting pairs is much smaller than $\binom{n}{2}$
Output-Sensitive Algorithms

- The $\Omega(n^2)$ lower-bound on the size of the output is unsatisfactory
  - In many cases, the number of intersecting pairs is much smaller than $\binom{n}{2}$
- An output-sensitive algorithm is an algorithm whose running-time is sensitive to the number, $k$, of intersecting pairs

Pat Morin  COMP2402/2002  Plane Sweep
The $\Omega(n^2)$ lower-bound on the size of the output is unsatisfactory

- In many cases, the number of intersecting pairs is much smaller than $\binom{n}{2}$

An output-sensitive algorithm is an algorithm whose running-time is sensitive to the number, $k$, of intersecting pairs

The Bently-Ottman plane-sweep algorithm runs in time $O((n + k) \log n)$ where $k$ is the number of intersecting pairs of segments
Output-Sensitive Algorithms

- The $\Omega(n^2)$ lower-bound on the size of the output is unsatisfactory
  - In many cases, the number of intersecting pairs is much smaller than $\binom{n}{2}$
- An output-sensitive algorithm is an algorithm whose running-time is sensitive to the number, $k$, of intersecting pairs
- The Bently-Ottman plane-sweep algorithm runs in time $O((n + k) \log n)$ where $k$ is the number of intersecting pairs of segments
- This is much faster when $k \ll \binom{n}{2}$
The Bentley-Ottman Plane Sweep Algorithm

- The plane sweep algorithm runs a simulation
The Bentley-Ottman Plane Sweep Algorithm

- The plane sweep algorithm runs a simulation
- A vertical line (the sweep line) sweeps from left to right
The Bentley-Ottman Plane Sweep Algorithm

- The plane sweep algorithm runs a simulation
- A vertical line (the *sweep line*) sweeps from left to right
- The sweep line pauses at
The Bentley-Ottman Plane Sweep Algorithm

- The plane sweep algorithm runs a simulation
- A vertical line (the *sweep line*) sweeps from left to right
- The sweep line pauses at
  - The endpoints of segments (endpoint events)
The Bentley-Ottman Plane Sweep Algorithm

- The plane sweep algorithm runs a simulation
- A vertical line (the sweep line) sweeps from left to right
- The sweep line pauses at
  - The endpoints of segments (endpoint events)
  - The intersection points (intersection events)
The Bentley-Ottman Plane Sweep Algorithm

- The plane sweep algorithm runs a simulation
- A vertical line (the *sweep line*) sweeps from left to right
- The sweep line pauses at
  - The endpoints of segments (endpoint events)
  - The intersection points (intersection events)
- During intersection events, we record the intersecting pairs
The Bentley-Ottman Plane Sweep Algorithm

- The plane sweep algorithm runs a simulation
- A vertical line (the *sweep line*) sweeps from left to right
- The sweep line pauses at
  - The endpoints of segments (endpoint events)
  - The intersection points (intersection events)
- During intersection events, we record the intersecting pairs
- Simplifying assumptions:
The Bentley-Ottman Plane Sweep Algorithm

- The plane sweep algorithm runs a simulation
- A vertical line (the *sweep line*) sweeps from left to right
- The sweep line pauses at
  - The endpoints of segments (endpoint events)
  - The intersection points (intersection events)
- During intersection events, we record the intersecting pairs
- Simplifying assumptions:
  - No segment is vertical
The Bentley-Ottman Plane Sweep Algorithm

- The plane sweep algorithm runs a simulation
- A vertical line (the *sweep line*) sweeps from left to right
- The sweep line pauses at
  - The endpoints of segments (endpoint events)
  - The intersection points (intersection events)
- During intersection events, we record the intersecting pairs
- Simplifying assumptions:
  - No segment is vertical
  - No three segments intersect in the same point
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Plane Sweep Algorithm

- The algorithm maintains two data structures

  - The sweep-line status is a `SortedSet` that stores the segments that currently intersect the sweep line, ordered from top to bottom (`y`-coordinate)

  - The event queue is a `PriorityQueue` that stores events (segment endpoints and intersections) ordered from left to right (`x`-coordinate)
The Plane Sweep Algorithm

- The algorithm maintains two data structures
- The *sweep-line status* is a SortedSet that stores the segments that currently intersect the sweep line, ordered from top to bottom (y-coordinate)
The Plane Sweep Algorithm

- The algorithm maintains two data structures
- The *sweep-line status* is a `SortedSet` that stores the segments that currently intersect the sweep line, ordered from top to bottom (y-coordinate)
- The *event queue* is a `PriorityQueue` that stores events (segment endpoints and intersections) ordered from left to right (x-coordinate)
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
To initialize

- set the \( x \)-coordinate of the sweep line to \(-\infty\)
Initialization

- To initialize
  - set the $x$-coordinate of the sweep line to $-\infty$
  - add all $2n$ segment endpoints to the event queue
Processing endpoint events

- To process an endpoint event
  
  ▶ For the left endpoint of a segment:
    ▶ Add to the sweep line status
    ▶ Check if it intersects the segment above or below it and add a crossing event to the event queue if necessary
  
  ▶ For the right endpoints of a segment:
    ▶ Remove from the sweep line status
    ▶ Check if the element above and below cross and add a crossing event to the event queue if necessary
Processing endpoint events

- To process an endpoint event
- For the left endpoint of a segment $s$:
  - Add $s$ to the sweep line status
  - Check if $s$ intersects the segment above or below it and add a crossing event to the event queue if necessary

For the right endpoints of a segment $s$:
  - Remove $s$ from the sweep line status
  - Check if the element above and below $s$ cross and add a crossing event to the event queue if necessary
Processing endpoint events

- To process an endpoint event
- For the left endpoint of a segment $s$:
  - Add $s$ to the sweep line status
- For the right endpoints of a segment $s$:
  - Remove $s$ from the sweep line status
  - Check if the element above and below $s$ cross and add a crossing event to the event queue if necessary
Processing endpoint events

- To process an endpoint event
- For the left endpoint of a segment $s$:
  - Add $s$ to the sweep line status
  - Check if $s$ intersects the segment above or below it and add a crossing event to the event queue if necessary

- For the right endpoints of a segment $s$:
  - Remove $s$ from the sweep line status
  - Check if the element above and below $s$ cross and add a crossing event to the event queue if necessary
Processing endpoint events

- To process an endpoint event
  - For the left endpoint of a segment $s$:
    - Add $s$ to the sweep line status
    - Check if $s$ intersects the segment above or below it and add a crossing event to the event queue if necessary
  - For the right endpoints of a segment $s$:
Processing endpoint events

- To process an endpoint event
- For the left endpoint of a segment $s$:
  - Add $s$ to the sweep line status
  - Check if $s$ intersects the segment above or below it and add a crossing event to the event queue if necessary
- For the right endpoints of a segment $s$:
  - Remove $s$ from the sweep line status
Processing endpoint events

- To process an endpoint event
- For the left endpoint of a segment $s$:
  - Add $s$ to the sweep line status
  - Check if $s$ intersects the segment above or below it and add a crossing event to the event queue if necessary
- For the right endpoints of a segment $s$:
  - Remove $s$ from the sweep line status
  - Check if the element above and below $s$ cross and add a crossing event to the event queue if necessary
Processing crossing events

- To process a crossing event where \( s \) and \( t \) cross:
  
  - Switch the order of \( s \) and \( t \) in the sweep line status
  
  - Check if \( s \) or \( t \) intersects the new elements above and below them in the sweep line and add crossing events to the event queue if necessary
Processing crossing events

- To process a crossing event where $s$ and $t$ cross:
  - Switch the order of $s$ and $t$ in the sweep line status
Processing crossing events

- To process a crossing event where $s$ and $t$ cross:
  - Switch the order of $s$ and $t$ in the sweep line status
  - Check if $s$ or $t$ intersects the new elements above and below them in the sweep line and add crossing events to the event queue if necessary
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Bentley-Ottman Plane Sweep Algorithm
The Plane Sweep Algorithm is correct because any pair \( s, t \) that crosses will eventually become adjacent in the sweep-line status structure.
Analysis

- The Plane Sweep Algorithm is correct because any pair $s, t$ that crosses will eventually become adjacent in the sweep-line status structure.
  - When they become adjacent, their crossing event is added to the event queue.
We process $2n + k$ events.
Analysis

- We process $2n + k$ events
- Each event requires
Analysis

- We process $2n + k$ events
- Each event requires
  - Adding an element to the event queue: $O(\log n)$
Analysis

- We process $2n + k$ events
- Each event requires
  - Adding an element to the event queue: $O(\log n)$
  - Getting an element from the event queue: $O(\log n)$
We process $2n + k$ events

Each event requires

- Adding an element to the event queue: $O(\log n)$
- Getting an element from the event queue: $O(\log n)$
- Searching the sweepline status: $O(\log n)$

Total running time is therefore $(2n + k) \cdot O(\log n) = O((n + k) \log n)$
We process $2n + k$ events

Each event requires
- Adding an element to the event queue: $O(\log n)$
- Getting an element from the event queue: $O(\log n)$
- Searching the sweepline status: $O(\log n)$

Total running time is therefore
$$(2n + k) \cdot O(\log n) = O((n + k) \log n)$$
Theorem: The Bentley-Ottman Plane Sweep Algorithm can compute all pairs of intersecting segments in $O((n + k) \log n)$ time, where $k$ is the number of pairs of segments that intersect.
Theorem: The Bentley-Ottman Plane Sweep Algorithm can compute all pairs of intersecting segments in $O((n + k) \log n)$ time, where $k$ is the number of pairs of segments that intersect.

Plane-sweep algorithms can solve many other problems:
Theorem: The Bentley-Ottman Plane Sweep Algorithm can compute all pairs of intersecting segments in $O((n + k) \log n)$ time, where $k$ is the number of pairs of segments that intersect.

Plane-sweep algorithms can solve many other problems:

- Given any set of objects, determine if any pair in the set intersect: $O(n \log n)$ time
Summary

Theorem: The Bentley-Ottman Plane Sweep Algorithm can compute all pairs of intersecting segments in $O((n + k) \log n)$ time, where $k$ is the number of pairs of segments that intersect.

Plane-sweep algorithms can solve many other problems:
- Given any set of objects, determine if any pair in the set intersect: $O(n \log n)$ time
- Find the closest pair of points among $n$ points: $O(n \log n)$
Summary

- **Theorem**: The Bentley-Ottman Plane Sweep Algorithm can compute all pairs of intersecting segments in $O((n + k) \log n)$ time, where $k$ is the number of pairs of segments that intersect.

- Plane-sweep algorithms can solve many other problems:
  - Given any set of objects, determine if any pair in the set intersect: $O(n \log n)$ time.
  - Find the closest pair of points among $n$ points: $O(n \log n)$.
  - A data structure for the planar point location problem: $O(n \log n)$ space and $O(\log n)$ query time.