Computational Geometry

Winter term 2017/18

Line-Segment Intersection

or

Map Overlay

Lecture #2

Steven Chaplick

Chair for Informatics I
Map Overlay in Geographic Information Systems (GIS)

Here:

= bridge
Line-Segment Intersection

Definition: Is there an intersection?

Answer: Depends...

Problem: Given a set $S$ of $n$ closed non-overlapping line segments in the plane, compute...

- all points where at least two segments intersect and
- for each such point report all segments that contain it.

Task: Discuss with your neighbor: how would you do it?
Example

Brute Force?

$O(n^2)$ ... can we do better?

Idea:
Process segments top-to-bottom using a “sweep line”.
Which active segments should be compared?
Data Structures

1) event (-point) queue $\mathcal{Q}$

\[ p \prec q \iff_{\text{def.}} y_p > y_q \text{ or } (y_p = y_q \text{ and } x_p < x_q) \]

Store event pts in \textit{balanced binary search tree} acc. to $\prec$

$\Rightarrow$ nextEvent() and del/insEvent() take $O(\log |\mathcal{Q}|)$ time

2) (sweep-line) status $\mathcal{T}$

Store the segments intersected by $\ell$ in left-to-right order.

\textbf{How?} In a balanced binary search tree!
Pseudo-code

findIntersections($S$)

**Input:** set $S$ of $n$ non-overlapping closed line segments

**Output:** – set $I$ of intersection pts
– for each $p \in I$ every $s \in S$ with $p \in s$

$Q \leftarrow \emptyset$; $T \leftarrow \langle$ vertical lines at $x = -\infty$ and $x = +\infty \rangle$  // sentinels

\[
\begin{align*}
\text{foreach } s \in S \text{ do} \\
&\quad \text{foreach endpoint } p \text{ of } s \text{ do} \\
&\quad\quad \text{if } p \notin Q \text{ then } Q.\text{insert}(p); L(p) = U(p) = \emptyset \\
&\quad\quad \text{if } p \text{ lower endpt of } s \text{ then } L(p).\text{append}(s) \\
&\quad\quad \text{if } p \text{ upper endpt of } s \text{ then } U(p).\text{append}(s)
\end{align*}
\]

while $Q \neq \emptyset$ do
\[
\begin{align*}
p \leftarrow Q.\text{nextEvent}() \\
Q.\text{deleteEvent}(p) \\
\text{handleEvent}(p)
\end{align*}
\]

This subroutine does the real work – how would you implement it?
Handling an Event

```
handleEvent(event p)
if |U(p) ∪ L(p) ∪ C(p)| > 1 then
  report intersection in p, report segments in U(p) ∪ L(p) ∪ C(p)
delete L(p) ∪ C(p) from T // consecutive in T!
insert U(p) ∪ C(p) into T in their order slightly below ℓ
if U(p) ∪ C(p) = ∅ then
  bleft /bright = left/right neighbor of p in T
  findNewEvent(bleft, bright, p)
else
  sleft/sright = leftmost/rightmost segment in U(p) ∪ C(p)
  bleft = left neighbor of sleft in T
  bright = right neighbor of sright in T
  findNewEvent(bleft, sleft, p)
  findNewEvent(bright, sright, p)
```
Correctness

**Lemma.** findIntersections() correctly computes all intersection points & the segments that contain them.

**Proof.** Let $p$ be an intersection pt. Assume (by induction):
- Every int. pt $q \prec p$ has been computed correctly.
- $T$ contains all segments intersecting $\ell$ in left-to-right order.

**Case I:** $p$ is not an interior pt of a segment.

$\Rightarrow$ $p$ has been inserted in $Q$ in the beginning.
Segm. in $U(p)$ and $L(p)$ are stored with $p$ in the beginning.

When $p$ is processed, we output all segm. in $U(p) \cup L(p)$.
$\Rightarrow$ All segments that contain $p$ are reported.
Correctness (Case II)

Case II: $p$ is an interior point of some segment, i.e., $C(p) \neq \emptyset$. If $p$ is not an endpt, need that $p$ is inserted into $Q$ before $\ell$ reaches $p$.

Let $s, s' \in C(p)$ be neighbors in the circular ordering of $C(p) \cup \{\ell\}$ around $p$. Imagine moving $\ell$ slightly back in time. Then $s, s'$ were neighbors in the left-to-right order on $\ell$ (in $T$). At the beginning of the algorithm, they weren’t neighbors in $T$. ⇒ There was some moment when they became neighbors!

This is when $\{p\} = s \cap s'$ was inserted into $Q$. □
\( Q \leftarrow \emptyset; \quad T \leftarrow \langle \text{vertical lines at } x = -\infty \text{ and } x = +\infty \rangle \quad // \text{sentinels} \\
\text{foreach } s \in S \text{ do} \\
\quad \text{foreach endpoint } p \text{ of } s \text{ do} \\
\quad \quad \text{if } p \not\in Q \text{ then } Q.\text{insert}(p); \quad L(p) = U(p) = \emptyset \\
\quad \quad \text{if } p \text{ lower endpt of } s \text{ then } L(p).\text{append}(s) \\
\quad \quad \text{if } p \text{ upper endpt of } s \text{ then } U(p).\text{append}(s) \\
\text{while } Q \neq \emptyset \text{ do} \\
\quad p \leftarrow Q.\text{nextEvent()} \\
\quad Q.\text{deleteEvent}(p); \quad \text{handleEvent}(p) \\
\text{handleEvent(event } p) \\
\quad \text{if } |U(p) \cup L(p) \cup C(p)| > 1 \text{ then} \\
\quad \quad \text{report intersection in } p, \text{ report segments in } U(p) \cup L(p) \cup C(p) \\
\quad \quad \text{delete } L(p) \cup C(p) \text{ from } T \quad // \text{consecutive in } T! \\
\quad \text{insert } U(p) \cup C(p) \text{ into } T \text{ in their order slightly below } \ell \\
\quad \text{if } U(p) \cup C(p) = \emptyset \text{ then} \\
\quad \quad b_{\text{left}}/b_{\text{right}} = \text{left/right neighbor of } p \text{ in } T \\
\quad \quad \text{findNewEvent}(b_{\text{left}}, b_{\text{right}}, p) \quad \Rightarrow \{p\} = s \cap s' \\
\quad \quad \text{if } x \not\in Q \text{ then } Q.\text{insert}(p) \\
\quad \text{else} \\
\quad \quad s_{\text{left}}/s_{\text{right}} = \text{leftmost/rightmost segment in } U(p) \cup C(p) \\
\quad \quad b_{\text{left}} = \text{left neighbor of } s_{\text{left}} \text{ in } T \\
\quad \quad b_{\text{right}} = \text{right neighbor of } s_{\text{right}} \text{ in } T \\
\quad \quad \text{findNewEvent}(b_{\text{left}}, s_{\text{left}}, p) \\
\quad \quad \text{findNewEvent}(b_{\text{right}}, s_{\text{right}}, p) \\
\text{Running time?}
Proof. Let \( p \) be an event pt, 
\[ m(p) = |L(p) \cup U(p) \cup C(p)| \] and 
\[ m = \sum_p m(p). \]
Then it’s clear that the runtime is \( O((m + n) \log n) \).

We show that \( m \in O(n + I) \). (\( \Rightarrow \) lemma)

Define (geometric) graph \( G = (V, E) \) with 
\[ V = \{\text{endpts, intersection pts}\} \Rightarrow |V| \leq 2n + I. \]
For any \( p \in V: m(p) \leq \deg(p). \)
\[ \Rightarrow m \leq \sum_p \deg(p) = 2|E| \leq 2 \cdot (3|V| - 6) \]
\[ \in O(n + I) \]
Euler (\( G \) is planar!!)
Today’s Main Result

**Theorem.** We can report all \( I \) intersection points among \( n \) non-overlapping line segments in the plane and report the segments involved in the intersections in \( O((n + I) \log n) \) time and \( O(n) \) space.

**Sure?** The event-point queue \( Q \) contains

- all segment end pts below the sweep line
- all intersection pts below the sweep line

\( \Rightarrow \) (worst-case) space consumption \( \in \Theta(n + I) \) :-(

**Can we do better?**

- insert \( s \cap s' \) into \( Q \)
- remove \( s \cap s' \) from \( Q \)
- re-insert \( s \cap s' \) into \( Q \)

\( \Rightarrow \) need just \( O(n) \) space; (asymptotic) running time doesn’t change