Algorithm Theory
2 Divide and Conquer: Line Intersection

Christian Schindelhauer

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Wintersemester 2007/08
Divide and Conquer: Line Intersection
The Principle of Divide and Conquer

- Remember Quicksort?
- Formulation and analysis of the principle
- Geometric Divide-and-Conquer
  - Closest-Pair
  - Line segment intersection
  - Voronoi diagram
- Fast Fourier Transformation
Divide and Conquer Paradigm

Divide-and-conquer method for solving a problem of size $n$

1. Divide:

   $n > c$: Divide the problem into $k$ sub-problems of sizes $n_1, \ldots, n_k$ ($k \geq 2$)

   $n \leq c$: Use direct solution

2. Conquer:

   Recursively solve the $k$ sub-problems (using Divide and Conquer)

3. Merge:

   Combine the $k$ partial solutions to get the overall solution
Line Segment Intersection

Find all pairs of intersection line segments

Naive approach:
$O(n^2)$

Output sensitive:
$O(n \log n + \#\text{intersections})$
Line Segment Intersection

Find all pairs of intersection line segments

We represent horizontal line segments by their endpoints. Then we introduce a vertical partitioning of all objects.
ReportCuts

Input: Set \( S \) of vertical line segments and endpoints of horizontal line segments

Output: All intersections of vertical line segments with horizontal line segments for which at least one endpoint is in \( S \).

1. Divide
   
   if \(|S| > 1\)
   
   then use a vertical line \( L \) to divide \( S \) into equally sized sets \( S_1 \) (left of \( L \)) and \( S_2 \) (right of \( L \))

   else \( S \) contains no intersections
1. Divide-Step

\[ \text{ReportCuts}(S_1); \text{ReportCuts}(S_2) \]

2. Conquer

\[ \text{ReportCuts}(S_1); \text{ReportCuts}(S_2) \]
3. Merge: ???

Possible intersection of a horizontal line segment $h$ in $S_1$

**Case 1:** both endpoints in $S_1$
Case 2: only one endpoint of $h$ in $S_1$

2 a) right endpoint in $S_1$
ReportCuts

2 b) left endpoint of $h$ in $S_1$

right endpoint in $S_2$

$S_1 \quad S_2$

Intersections not reported

right endpoint not in $S_2$

$S_1 \quad 1S_2$
Procedure: ReportCuts(S)

3. Merge:
Return the intersection of vertical line segments in \( S_2 \) with horizontal line segments in \( S_1 \), for which the left endpoint is in \( S_1 \) and the right endpoint is neither in \( S_1 \) nor \( S_2 \). Proceed analogously for \( S_1 \).
Implementation

Set S

$L(S)$: y-coordinates of all left endpoints in $S$ for which the corresponding right endpoints is not in $S$

$R(S)$: y-coordinates of all right endpoints in $S$ for which the corresponding left end point is not in $S$

$V(S)$: y-intervals of all vertical line-segments in $S$
Base cases

S contains only one elements $s$

Case 1: $s = (x,y)$ is a left endpoint

$L(S) = \{y\}$  $R(S) = \emptyset$  $V(S) = \emptyset$

Case 2: $s = (x,y)$ is a right endpoint

$L(S) = \emptyset$  $R(S) = \{y\}$  $V(S) = \emptyset$

Case 3: $s = (x,y_1,y_2)$ is a vertical line-segment

$L(S) = \emptyset$  $R(S) = \emptyset$  $V(S) = \{[y_1,y_2]\}$
Merge-Step

Assume that \( L(S_i), R(S_i), V(S_i) \) are known \( i=1,2 \)

\[ S = S_1 \cup S_2 \]

\[ L(S) = (L(S_1) \setminus R(S_2)) \cup L(S_2) \]

\[ R(S) = (R(S_2) \setminus L(S_1)) \cup R(S_1) \]

\[ V(S) = V(S_1) \cup V(S_2) \]

\( L,R: \) ordered by increasing y-coordinates
linked lists

\( V: \) ordered by increasing lower endpoints
linked lists
Output of the intersections

Runtime: $|V(S_2)| + \#\text{intersections}$
Running Time

Initially, the input (vertical line segments, left/right endpoints of horizontal line segments) has to be sorted and stored in an array.

Divide-and-Conquer:

\[ T(n) = 2T(n/2) + an + \text{size of output} \]
\[ T(1) = O(1) \]

\[ O(n \log n + k) \quad k = \#\text{intersections} \]
Computation of a Voronoi Diagram

**Input:** A set of sets

**Output:** Partition of the plane into regions, each consisting of the points closer to a site than any other site
Definition of a Voronoi Diagram

*P*: Set of sites

\[ H(p \mid p') = \{ x \mid x \text{ is closer to } p \text{ than to } p' \} \]

Voronoi region of *p*

\[ VR(p) = \bigcap_{p' \in P \setminus \{p\}} H(p \mid p') \]
Computation of a Voronoi Diagram

**Divide:** Partition the set of sites into equally sized sets

**Conquer:** Recursive computation of the two smaller Voronoi diagrams

**Stopping conditions:** The Voronoi diagram of a single is the whole plane

**Merge:** Connect the diagrams by adding new edges
Computation of the Voronoi Diagram

Output: The complete Voronoi diagram

Running time: $O(n \log n)$, where $n$ is the number of sites
Algorithm Theory
2 Divide and Conquer: Line Intersection

Christian Schindelhauer

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Wintersemester 2007/08