CSCI 4560/6560 Computational Geometry

https://www.cs.rpi.edu/~cutler/classes/computationalgeometry/S22/

Lecture 13: Arrangements & Duality

Administrative Note

This course has been officially approved as “Communication Intensive”, starting this term, Spring 2022. Yay!

Outline for Today

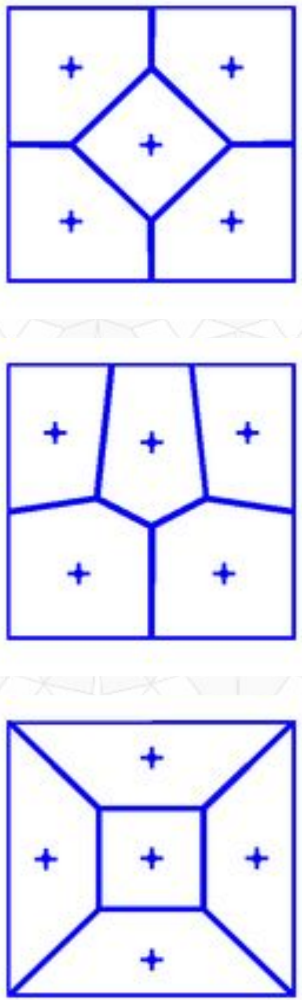
- Last Lecture: Problems that reduce to Voronoi Diagrams
- Duality: Points $\leftrightarrow$ Lines
- Arrangement of Lines
- Complexity of an Arrangement of Lines
- Algorithm to Construct Arrangement of Lines
- Arrangement Application: Ray Tracing Supersampling
- Next Time: Arrangement Application: Architectural Sketching
- Next Time: Delaunay Triangulations

Centroidal Voronoi Diagram

- *What if we could place all of the grocery stores?*
- Where should we place the grocery stores so that they are centrally located for all of their customers?
- But if you change the position of the store, the closest store will change for some customers…

- Points are at the center of mass of their cell
- Constructed using k-means clustering / Lloyd's algorithm - an iterative relaxation algorithm

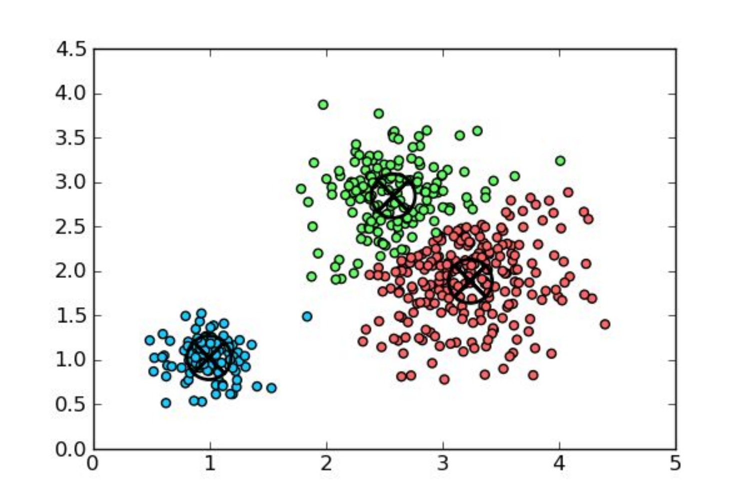
- *Note: May be multiple solutions!*



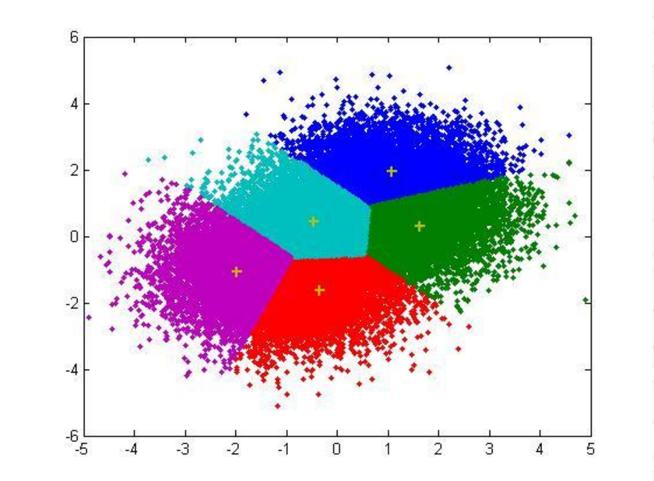
https://en.wikipedia.org/wiki/Centroidal_Voronoi_tessellation

K-Means Clustering

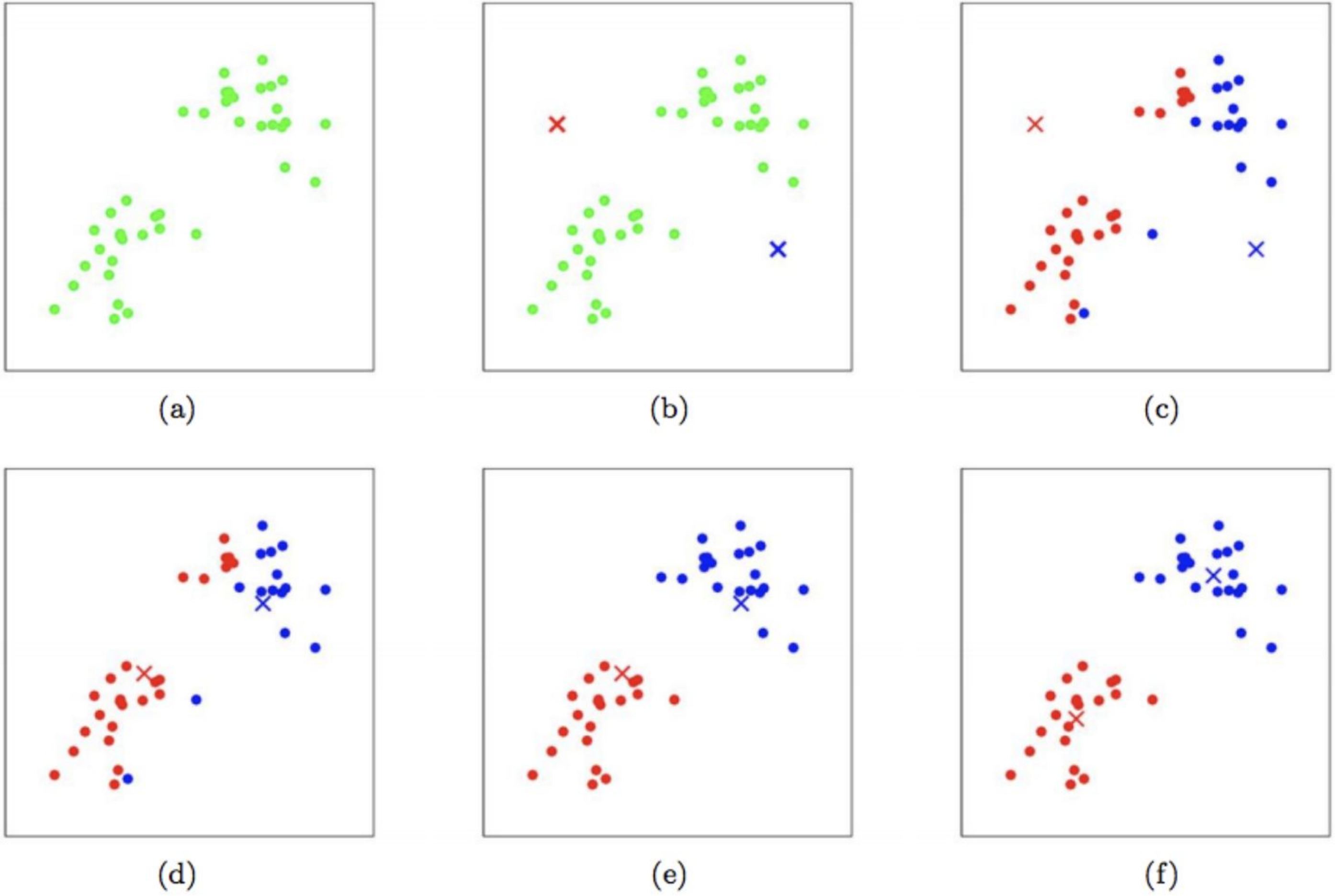
- Works quite well, when the data can be meaningfully classified (and we know how many clusters to use).
- With dense data, output is visually similar to Voronoi diagram (k-Means chooses the data points that define the cells)



http://blog.mpacula.com/2011/04/27/k-means-clustering-example-python/



"Efficient K-Means Clustering using JIT" Yi Cao

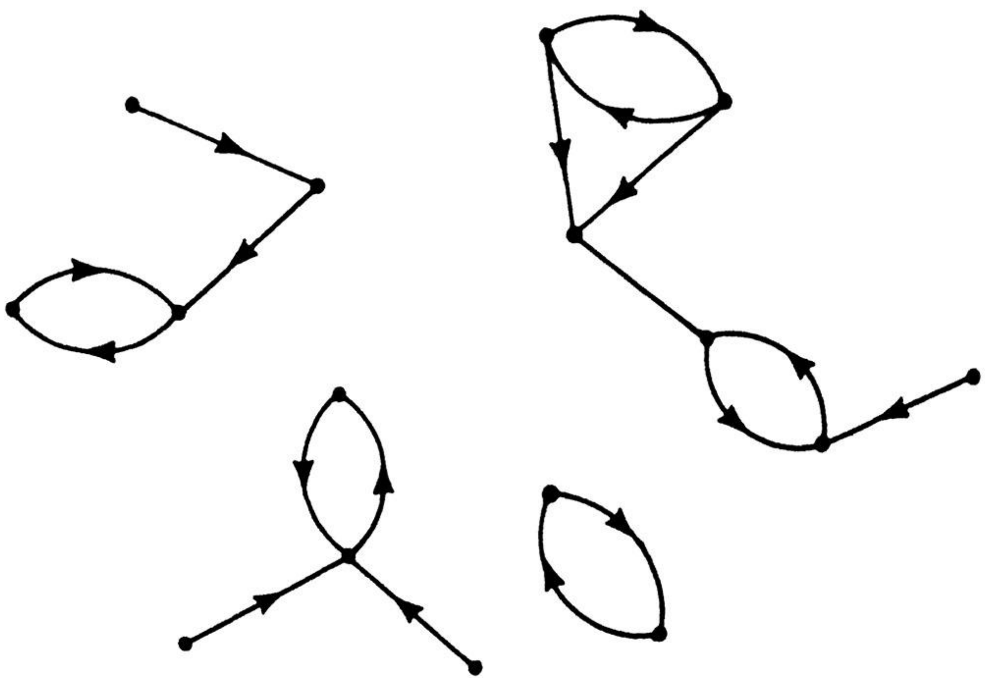


Wei Zhang
https://wei2624.github.io/MachineLearning/usv_kmeans/

Problem: Closest Pair

- Which two points are the closest?
- Applications - Collision Detection & Air Traffic Control
- Which two objects have soonest potential for collision?

- Linear loop over all edges in the All Nearest Neighbors solution to find the shortest edge
- Will be a reciprocal pair



Computational Geometry: An Introduction,
Preparata & Shamos, Figure 5.1

Problem: Euclidean Minimum Spanning Tree

- Given *n* points
- Draw *n-1* edges to create a tree, connecting all points without creating any cycles.
- Pick edges to minimize the sum of their lengths.

- *Application: Minimize cost of physical telephone lines*

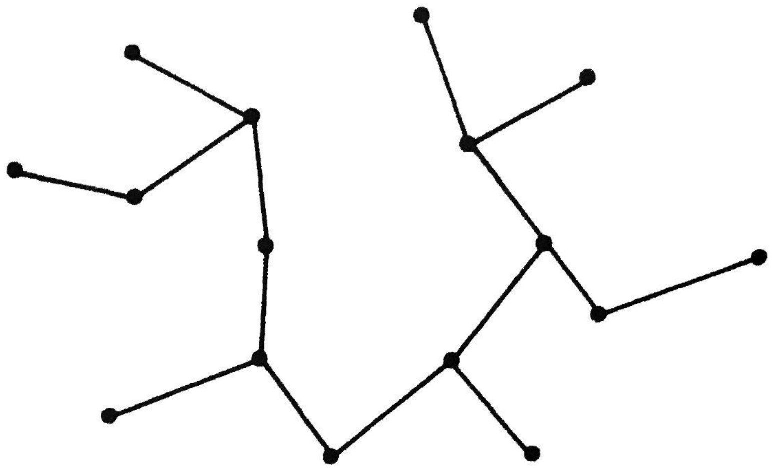


Figure 5.2 A minimum spanning tree on a planar point set.

Computational Geometry: An Introduction, Preparata & Shamos, Figure 5.2

Reduce Convex Hull to Voronoi Diagram

- Theorem: Voronoi polygon V_i is unbounded if and only if Voronoi site i is on the convex hull of all sites. (proved in Preparata & Shamos)

- O(n) to convert Voronoi Diagram to Convex Hull:
  - Start with any unbounded cell
  - Walk edges clockwise to find adjacent unbounded cell
  - Voronoi sites will trace convex hull in counter-clockwise order

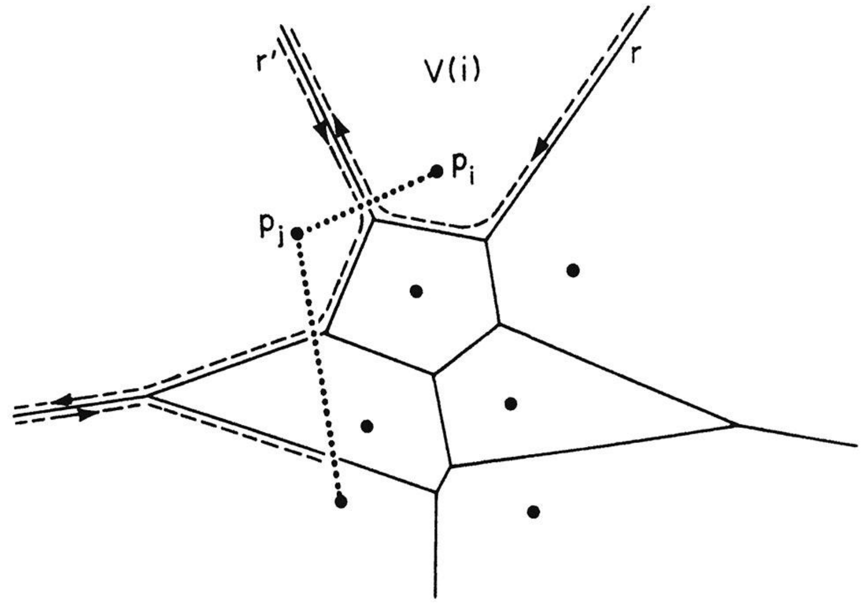
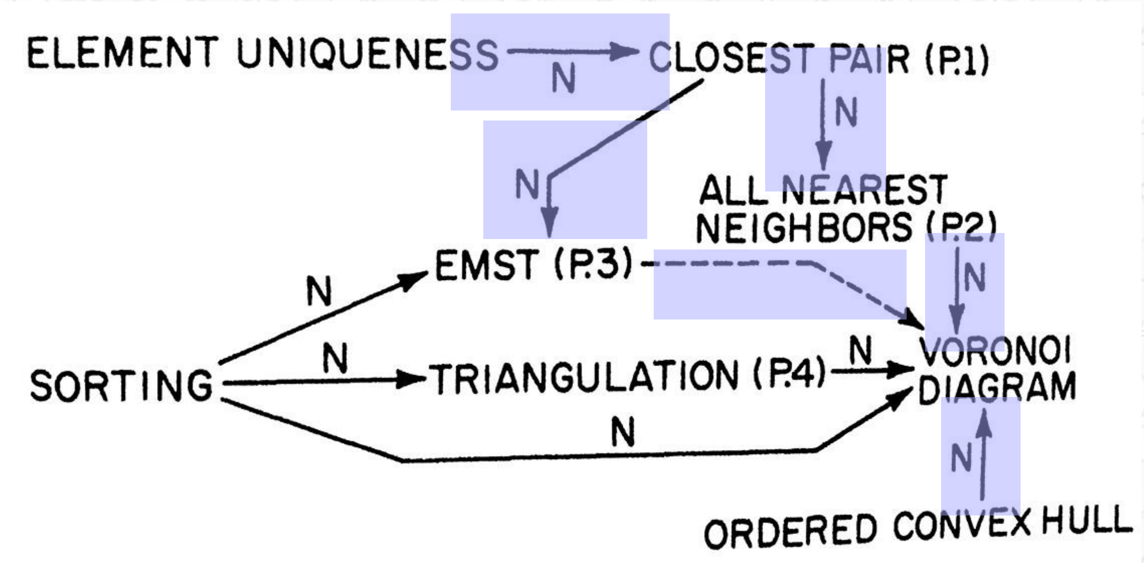


Figure 5.31 Construction of the convex hull from the Voronoi diagram.

Computational Geometry: An Introduction,
Preparata & Shamos, Figure 5.31

Problems that Reduce to Voronoi Diagram

- We can compute the Voronoi Diagram of *n* points in *O(n log n)* time and *O(n)* space.
- These other problems can be computed in *O(n)* additional time if given the Voronoi Diagram.
- *Therefore they are also O(n log n) time and O(n) space.*



Computational Geometry: An Introduction,
Preparata & Shamos, Figure 5.30

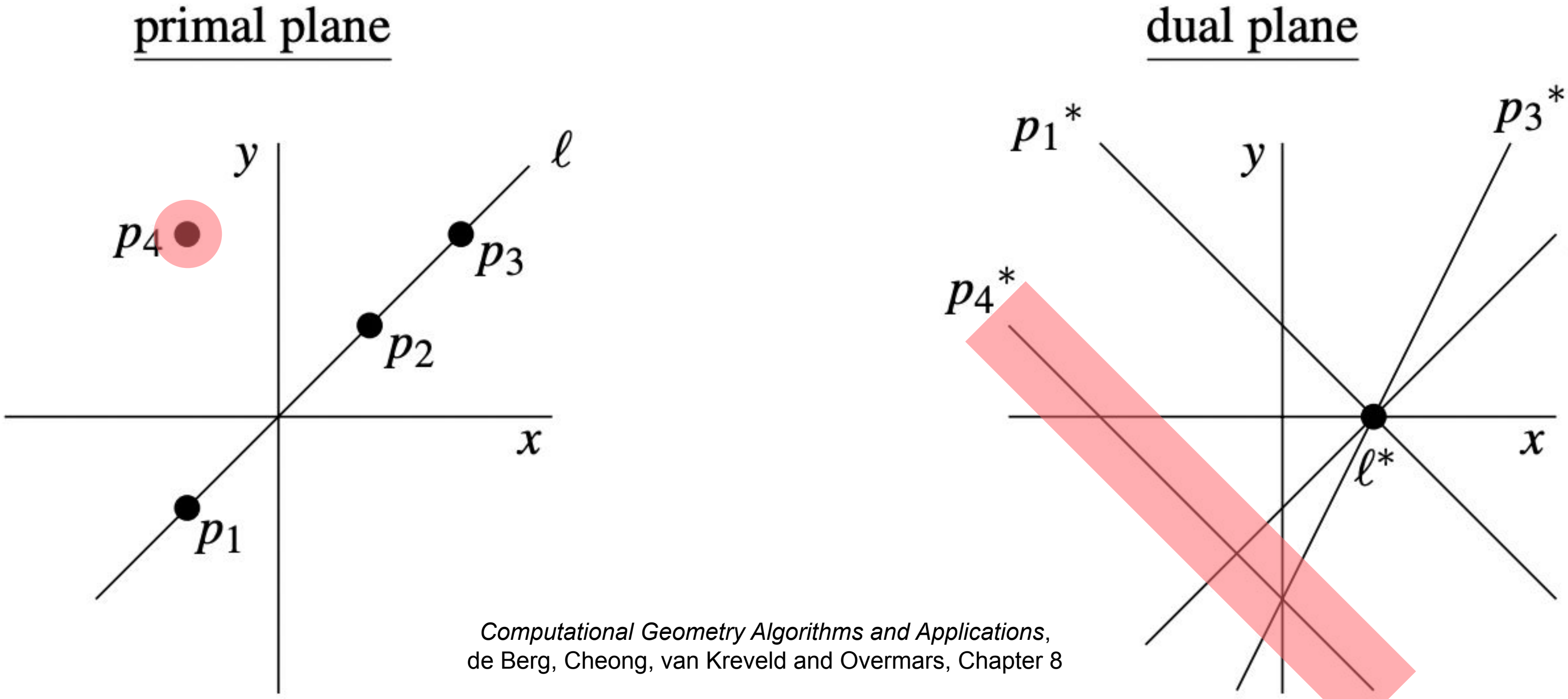
Outline for Today

- Last Lecture: Problems that reduce to Voronoi Diagrams
- Duality: Points ↔ Lines
- Arrangement of Lines
- Complexity of an Arrangement of Lines
- Algorithm to Construct Arrangement of Lines
- Arrangement Application: Ray Tracing Supersampling
- Next Time: Arrangement Application: Architectural Sketching
- Next Time: Delaunay Triangulations

Duality: Points ↔ Lines

slope **y-intercept**

Point p: (p_x, p_y) in primal plane ↔ Line p^*: $y = p_x x - p_y$ in dual plane

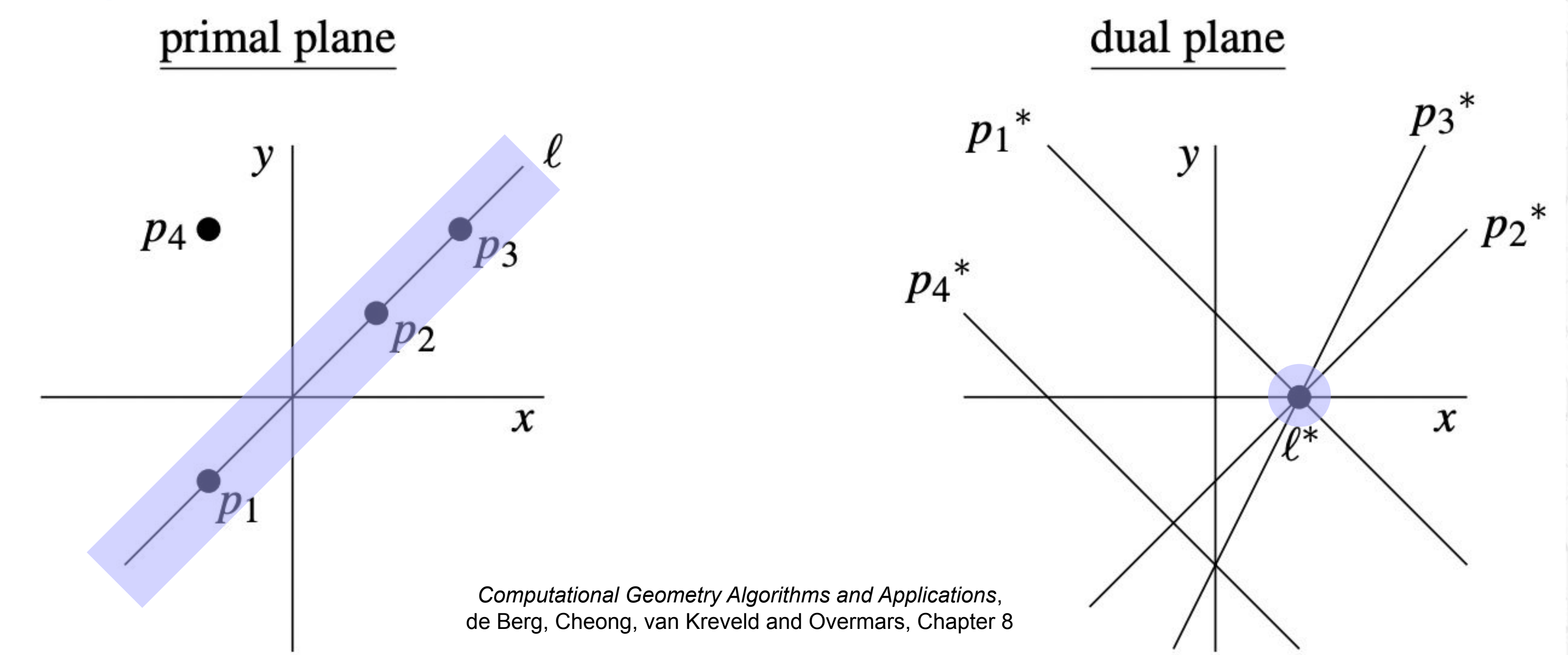


Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 8

Duality: Points ↔ Lines

Line ℓ: $y = mx + b$ in primal plane ↔ Point ℓ^*: (m,b) in dual plane

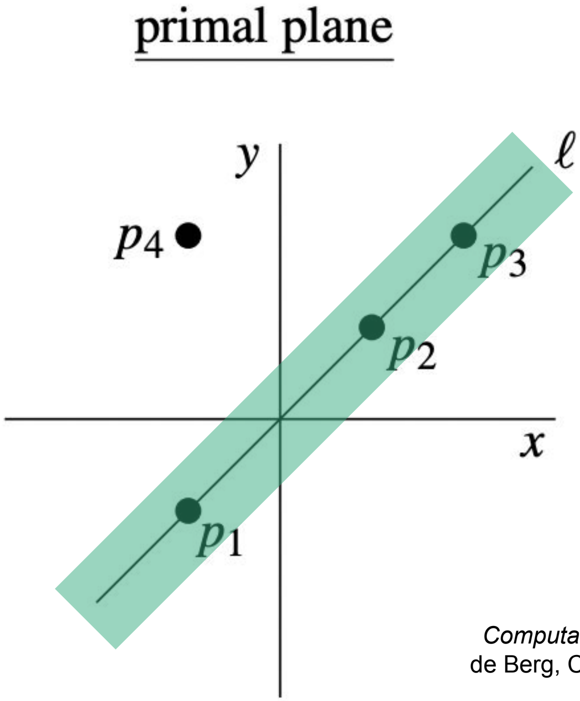
slope **y-intercept**

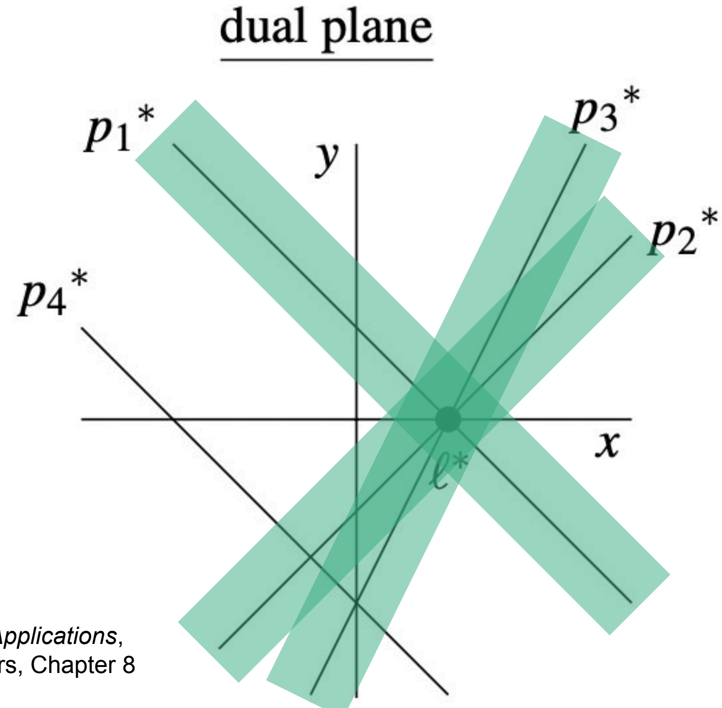


Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 8

Duality: Points ↔ Lines

Points p_1, p_2, p_3 on line ℓ in primal plane,
are lines p_1*, p_2*, p_3* that pass through point ℓ* in dual plane.

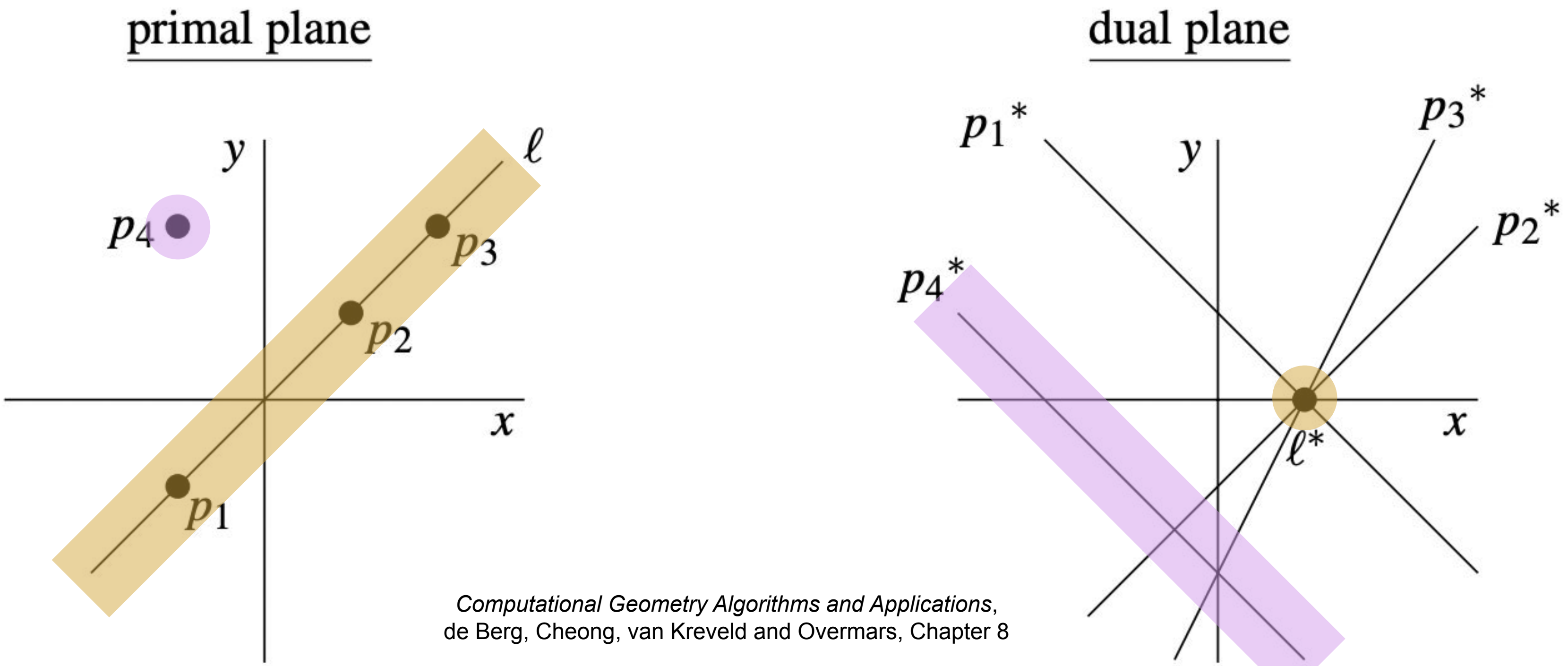




Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 8

Duality: Points ↔ Lines

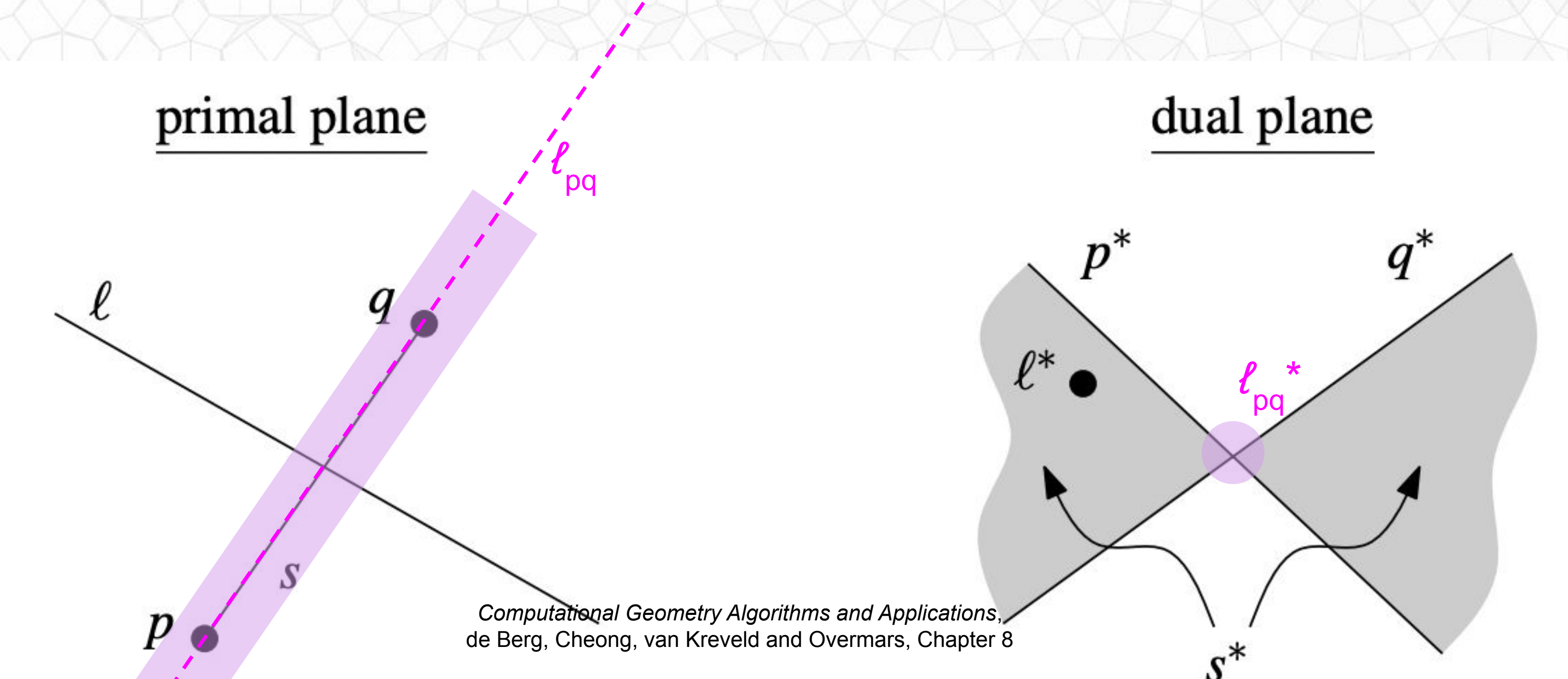
Point p_4 that lines above line ℓ in primal plane,
Is line p_4* that lies beneath point ℓ* in dual plane.



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 8

Duality: Line Segment ↔ Double Wedge

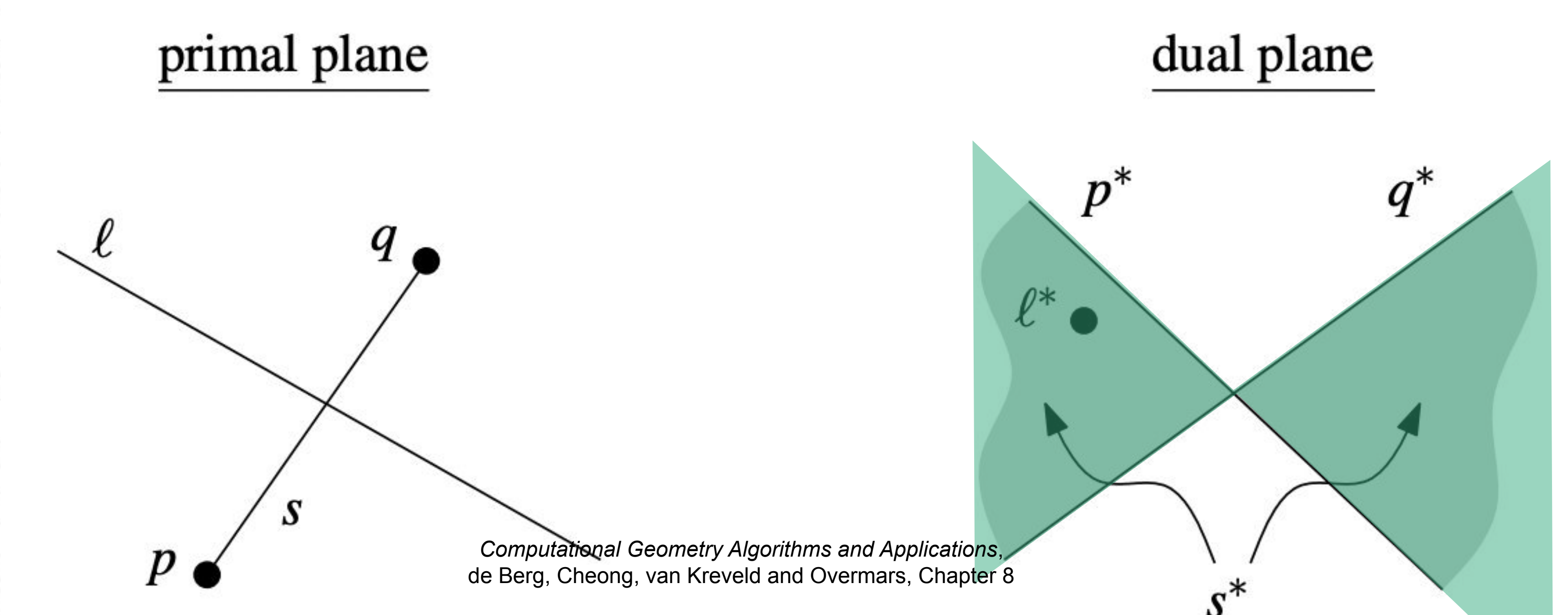
Line segment *s* between points *p* and *q*, which lies on line ℓ_{pq}, in primal plane



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 8

Duality: Line Segment ↔ Double Wedge

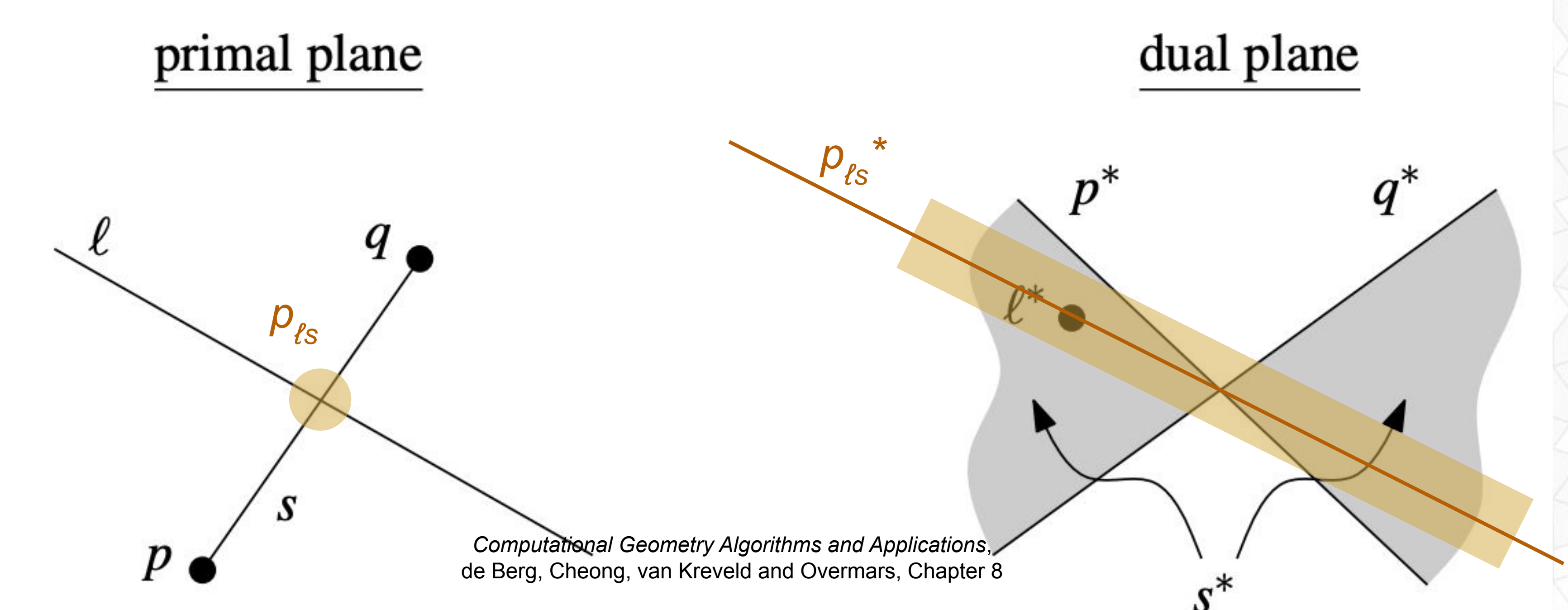
Line segment s between points p and q, which lies on line ℓ_{pq}, in primal plane
Is a double wedge s^* of area between lines p^* and q^* in the dual plane



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 8

Duality: Line Segment ↔ Double Wedge

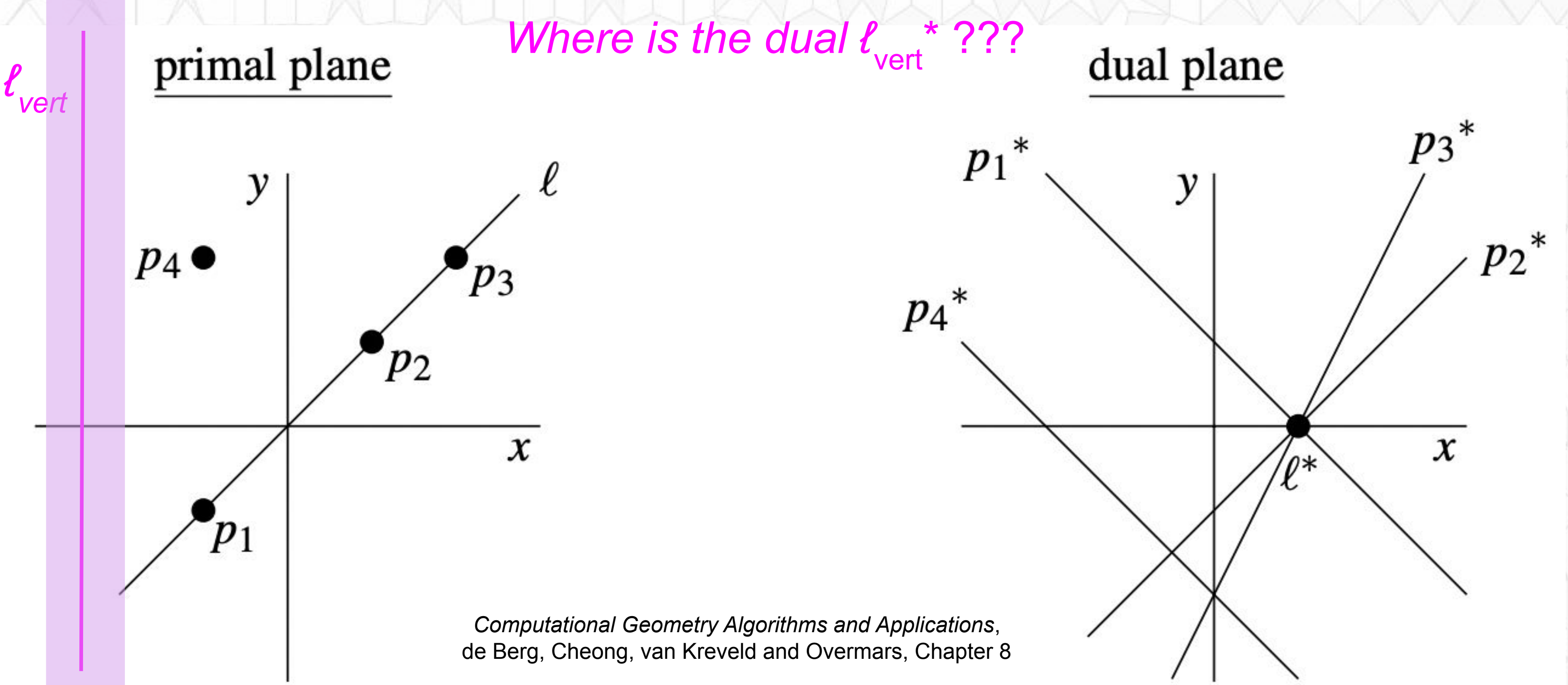
The intersection point $p_{\ell s}$ of segment s and line ℓ in primal plane,
Is line $p_{\ell s}{}^*$ that lies inside double wedge s^* *and crosses* ℓ^* *and* $\ell_{pq}{}^*$ *in the dual plane*



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 8

Duality: Assumptions / Special Cases

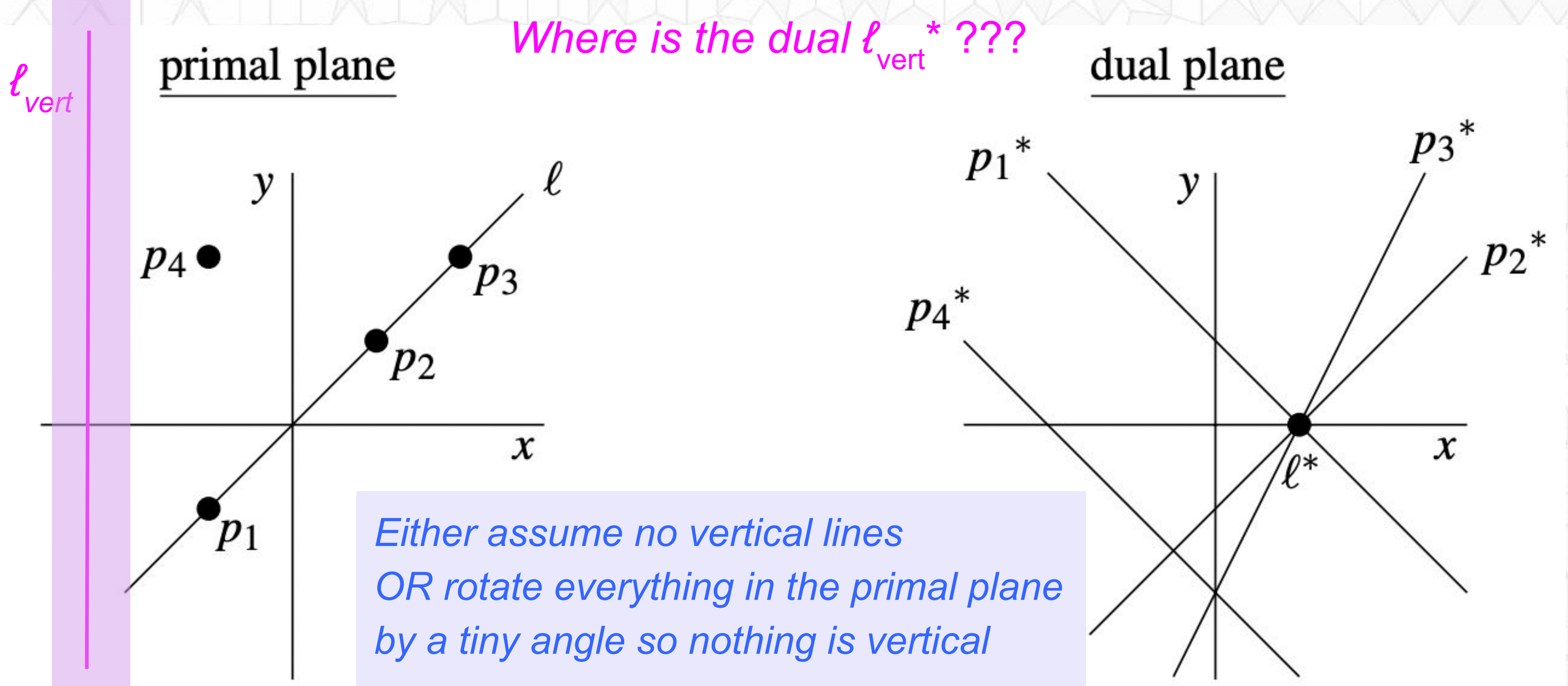
A vertical line segment ℓ_{vert} in the primal plane has slope *m* = ∞*, and b = undefined*



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 8

Duality: Assumptions / Special Cases

A vertical line segment ℓ_{vert} in the primal plane has slope m = ∞, *and b = undefined*



Where is the dual ℓ_{vert} ???*

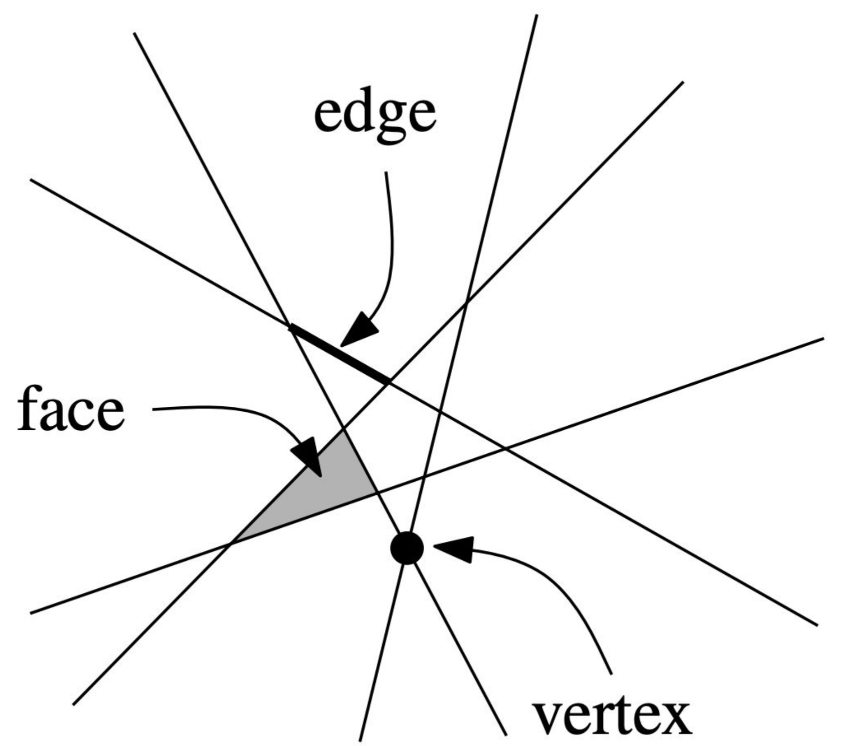
Either assume no vertical lines
OR rotate everything in the primal plane
by a tiny angle so nothing is vertical

Outline for Today

- Last Lecture: Problems that reduce to Voronoi Diagrams
- Duality: Points $\leftrightarrow$ Lines
- Arrangement of Lines
- Complexity of an Arrangement of Lines
- Algorithm to Construct Arrangement of Lines
- Arrangement Application: Ray Tracing Supersampling
- Next Time: Arrangement Application: Architectural Sketching
- Next Time: Delaunay Triangulations

Arrangement of Lines

- A collection of *n* lines in the plane
- Creates a subdivision of the plane into vertices, edges, and faces

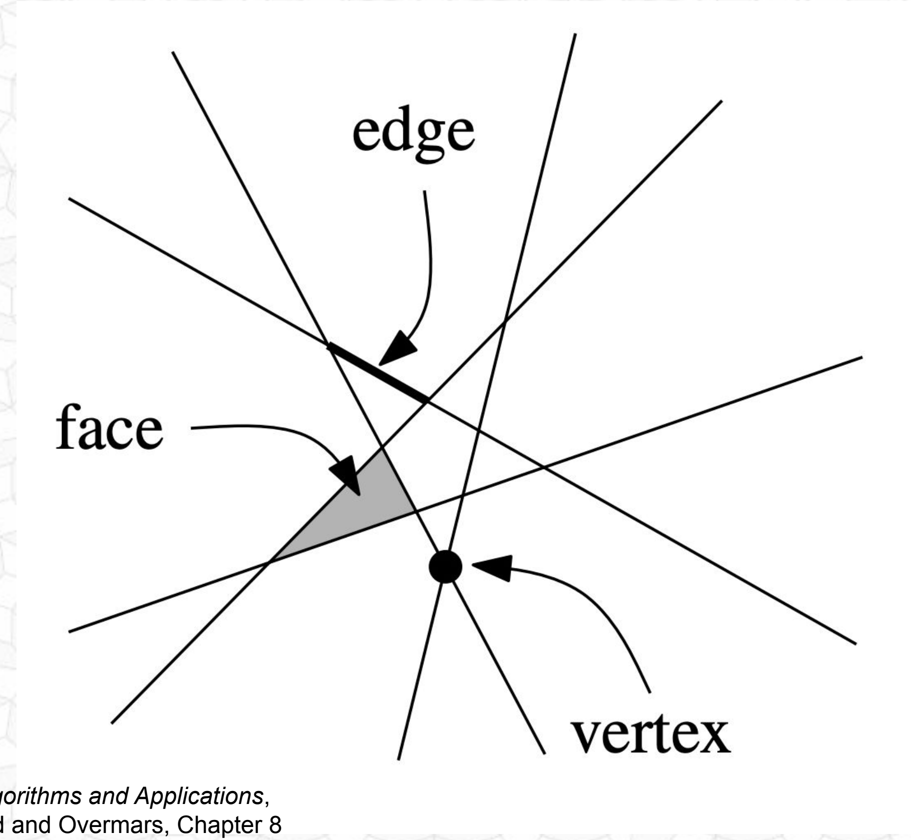


Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 8

Arrangement of Lines

- A collection of *n* lines in the plane
- Creates a subdivision of the plane into vertices, edges, and faces

- Definition:
  A *simple arrangement* of lines
  - No three lines pass through the same point
  - No two lines are parallel



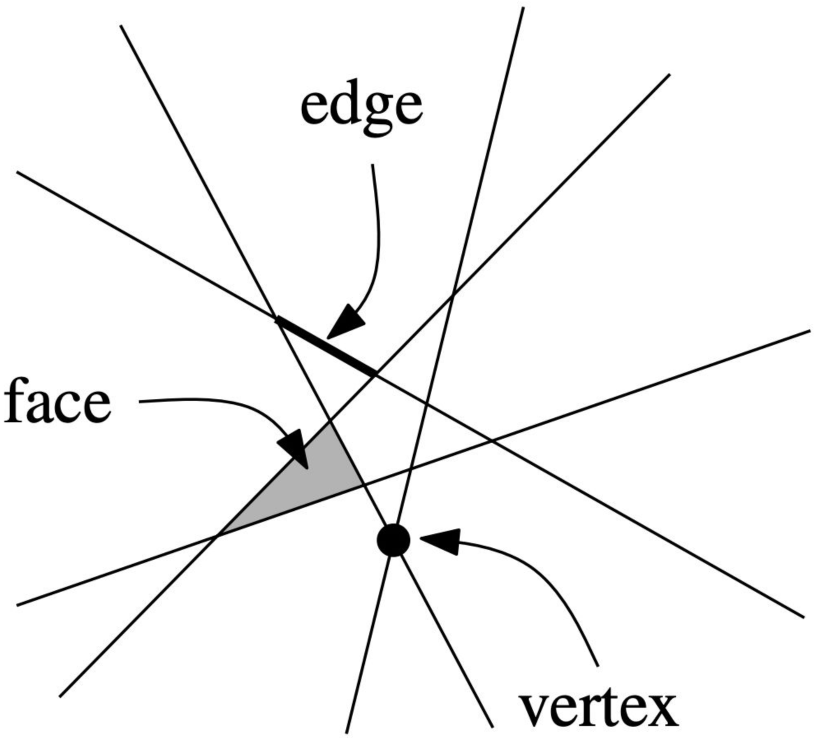
Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 8

Outline for Today

- Last Lecture: Problems that reduce to Voronoi Diagrams
- Duality: Points ↔ Lines
- Arrangement of Lines
- Complexity of an Arrangement of Lines
- Algorithm to Construct Arrangement of Lines
- Arrangement Application: Ray Tracing Supersampling
- Next Time: Arrangement Application: Architectural Sketching
- Next Time: Delaunay Triangulations

Complexity of an Arrangement of Lines

- A collection of *n* lines in the plane
- How many vertices?
  -
- How many edges?
  -
- How many faces?
  -



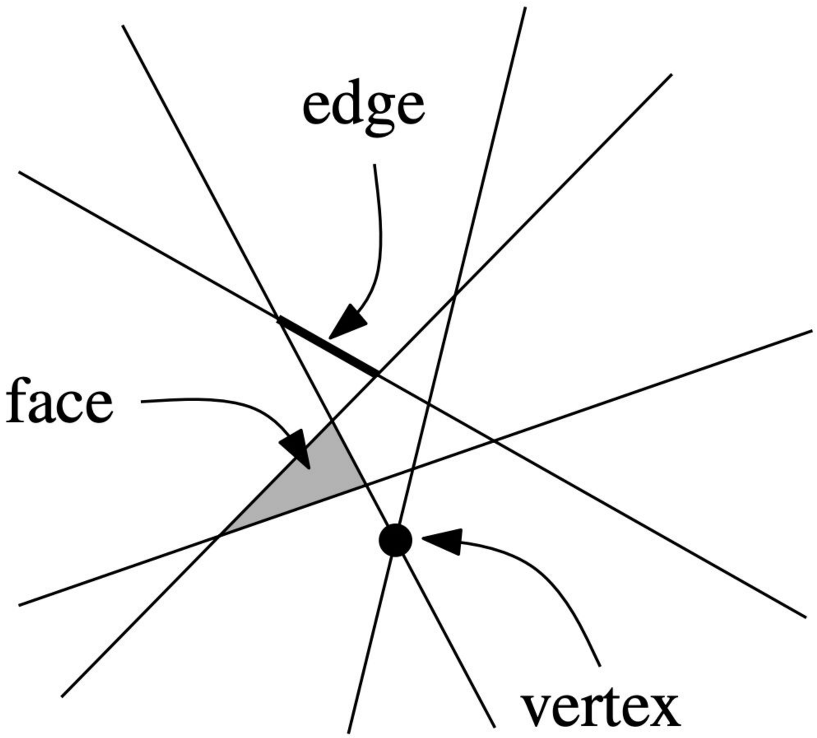
Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 8

Complexity of an Arrangement of Lines

- A collection of *n* lines in the plane
- How many vertices?
  - $n * (n-1) / 2$
- How many edges?
  - n^2
- How many faces?
  - $n^2/2 + n/2 + 1$

Or fewer if not a *simple arrangement*

- 3 or more lines intersect at a point, or
- 2 or more lines are parallel



Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 8

Outline for Today

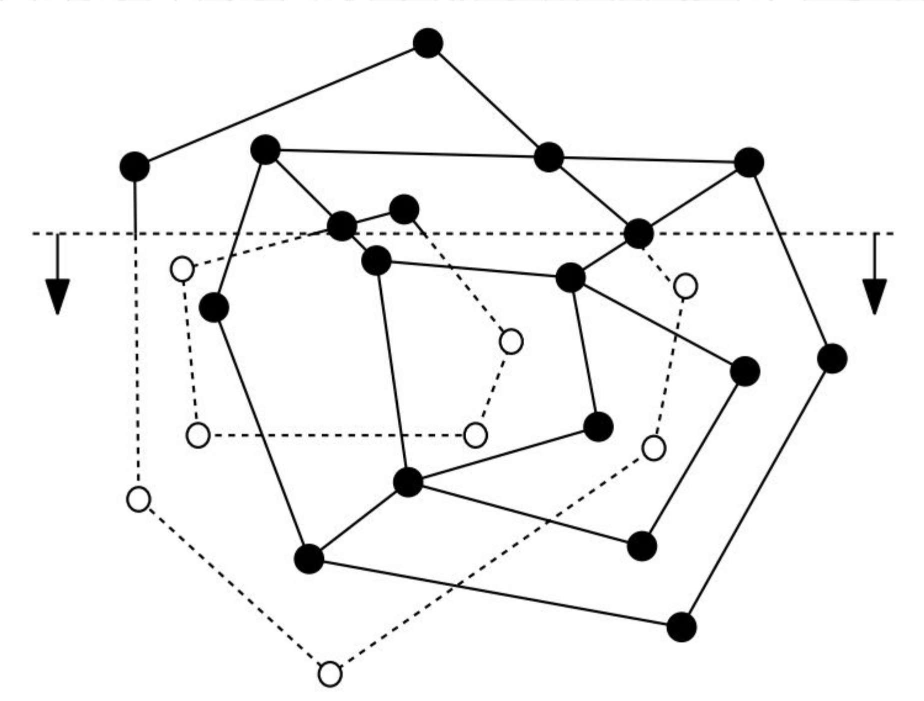
- Last Lecture: Problems that reduce to Voronoi Diagrams
- Duality: Points ↔ Lines
- Arrangement of Lines
- Complexity of an Arrangement of Lines
- Algorithm to Construct Arrangement of Lines
- Arrangement Application: Ray Tracing Supersampling
- Next Time: Arrangement Application: Architectural Sketching
- Next Time: Delaunay Triangulations

Map Overlay & Line Segment Intersection

- Line Sweep Algorithm covered in Lecture 3

- For *n* line segments
- With *k* overlay complexity (# of elements in output)

- Runtime Analysis: *O(n log n + k log n)*



Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 2

Applied to (Unbounded) Lines…

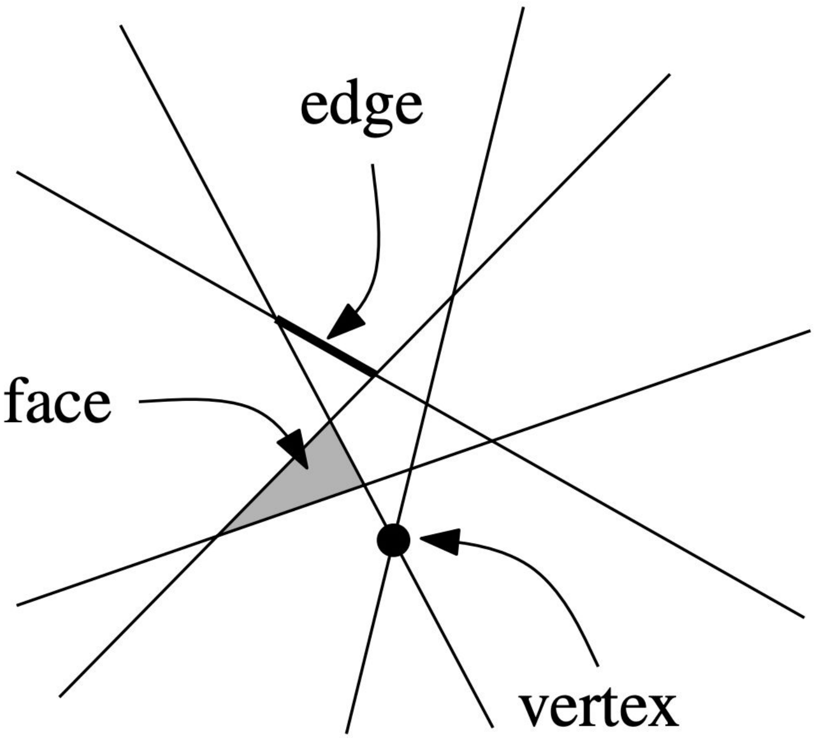
- For n ~~line segments~~ **lines**
- With k overlay complexity (# of elements in output)

  → $\boldsymbol{k = O(n^2)}$

- Runtime Analysis: $O(n \log n + k \log n)$

  → $\boldsymbol{O(n^2 \log n)}$

Can we do better?



Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 8

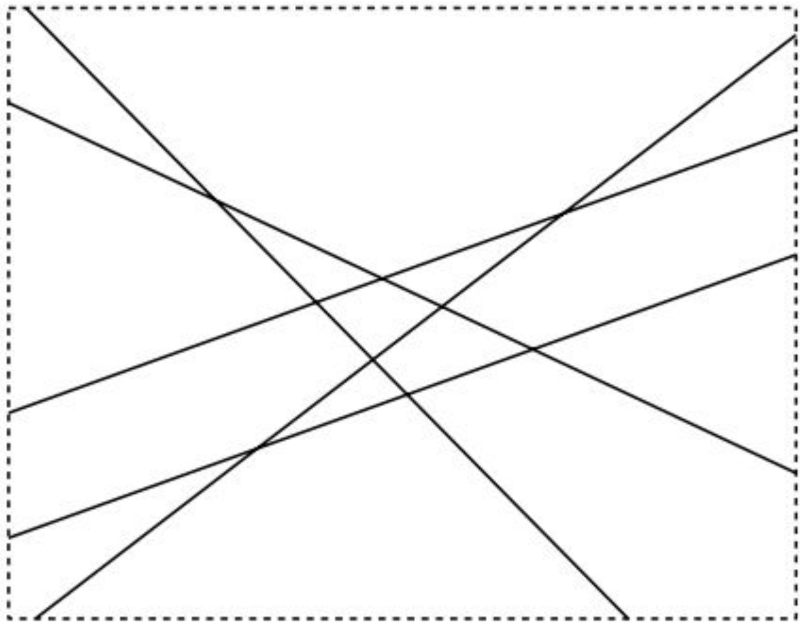
Construct an Arrangement

- Dealing with unbounded cells in a half-edge structure is impractical.

- Compute the bounding box
  for the arrangement.

- Find all n * (n-1) vertices
  (pairwise intersect all of the lines)

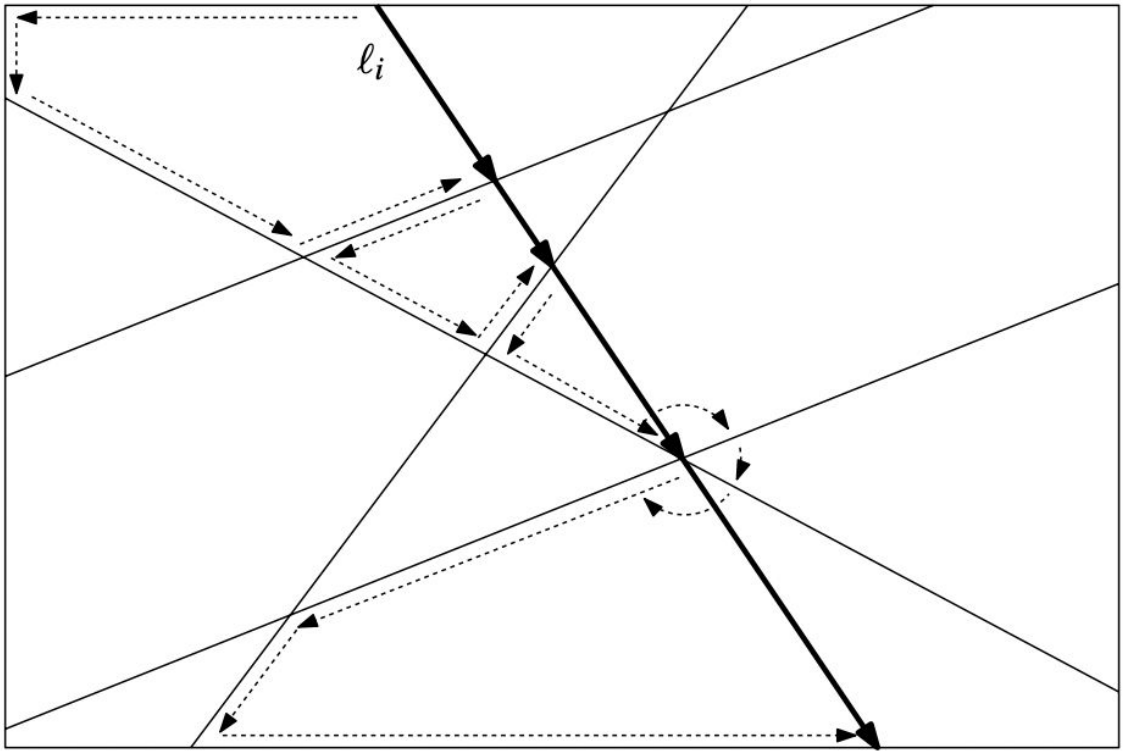
- Find the maximum and minimum
  x and y coordinates



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 8

Construct an Arrangement

- Insert the lines one at a time
- Intersect the line with the bounding box
- Cut edge into two new edges
- Cut face into two new faces
- Walk the edges of the face to find the next face



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 8

Construct an Arrangement

Runtime Analysis:

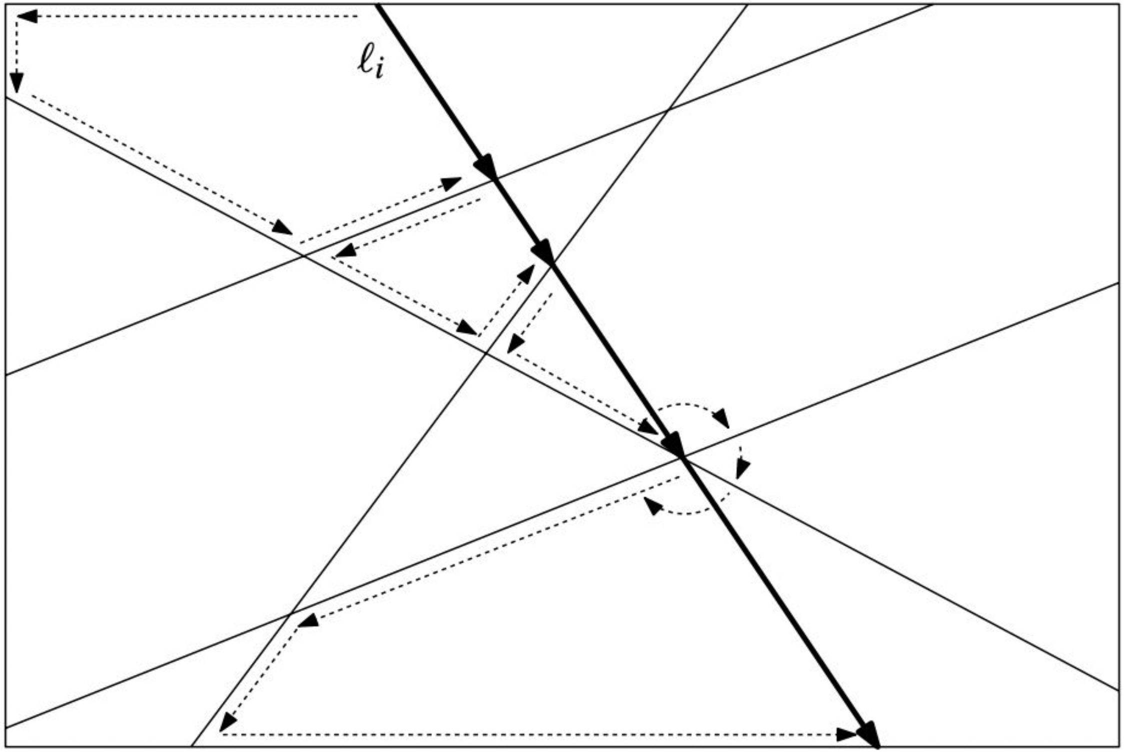
- linear cost to insert each line

  → ***O(n)***

- Overall:

  → ***O(n²)***

Line arrangements (& their computation) are quadratic…



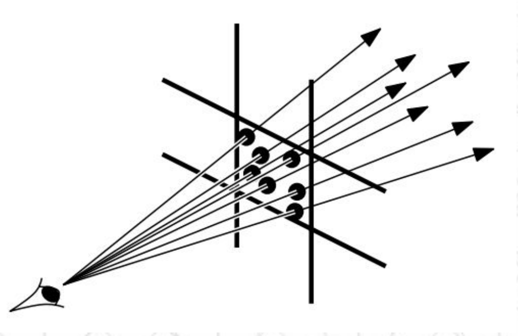
Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 8

Outline for Today

- Last Lecture: Problems that reduce to Voronoi Diagrams
- Duality: Points $\leftrightarrow$ Lines
- Arrangement of Lines
- Complexity of an Arrangement of Lines
- Algorithm to Construct Arrangement of Lines
- Arrangement Application: Ray Tracing Supersampling
- Next Time: Arrangement Application: Architectural Sketching
- Next Time: Delaunay Triangulations

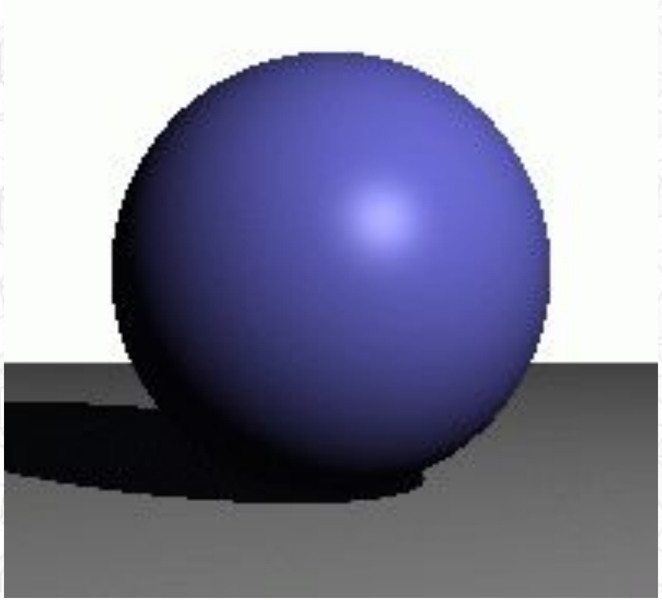
Ray Tracing Antialiasing – Supersampling

● Trace multiple rays per pixel

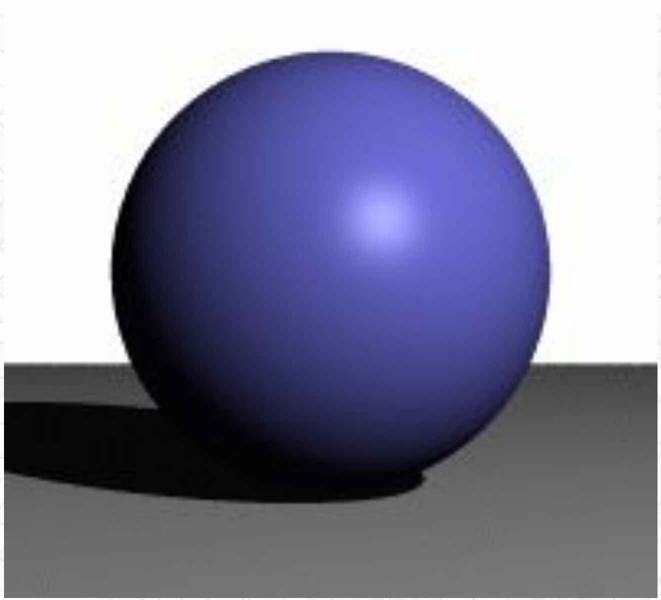


Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld
and Overmars, Chapter 8

jaggies



w/ antialiasing



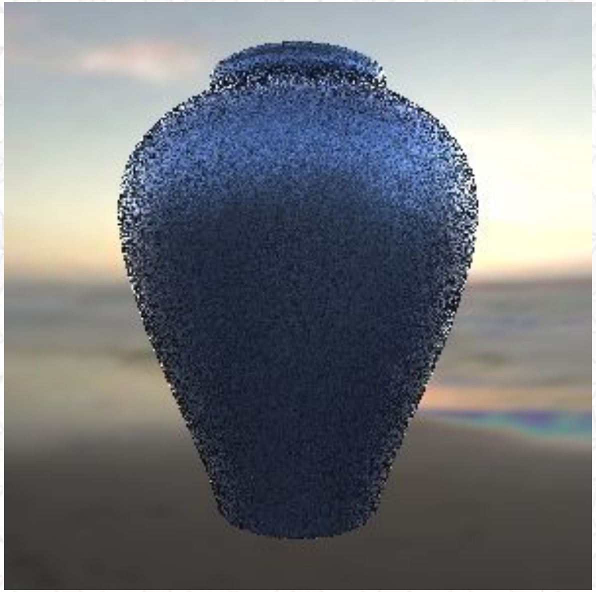
Noise from Insufficient Sampling

Can be very noticeable and distracting!

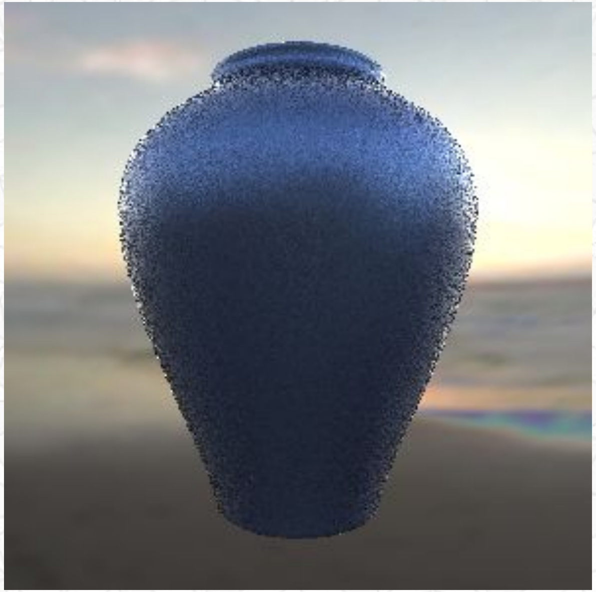
Henrik Wann Jensen

Noise from Insufficient Sampling

5 Samples/Pixel



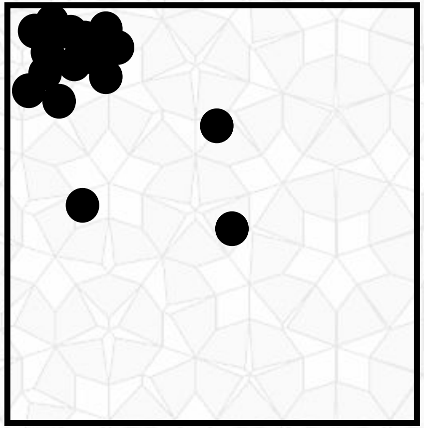
25 Samples/Pixel



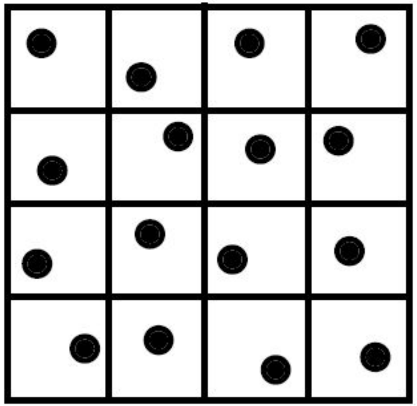
75 Samples/Pixel

Noise also comes from Poor Sampling

- With uniform random sampling,
  we can get unlucky…
  e.g. all samples in a corner



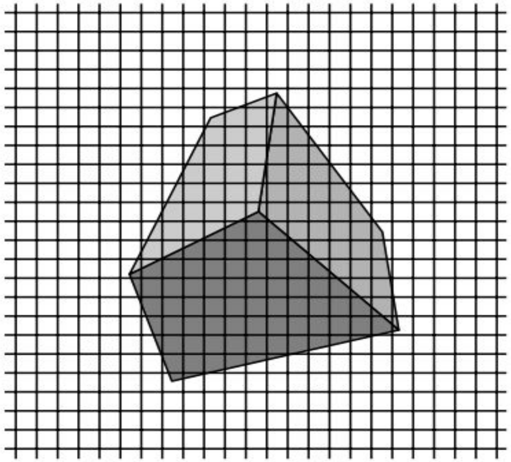
- *Stratified Sampling* can prevent it
  - Subdivide domain Ω into non-overlapping regions Ω_i
  - Each region is called a stratum
  - Take one random samples per Ω_i

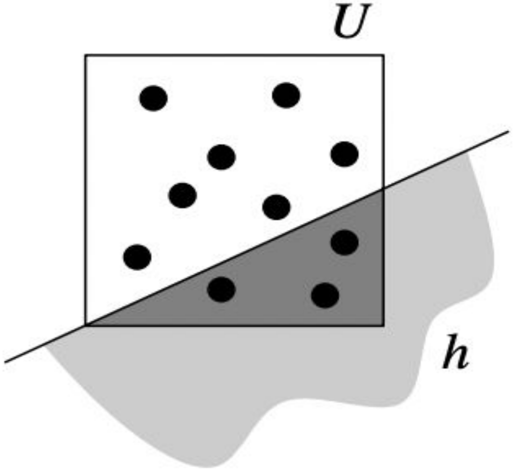


Compute the Discrepancy of a Specific Pixel Sampling

- Primarily we'll be ray tracing / sampling straight-edged geometric objects

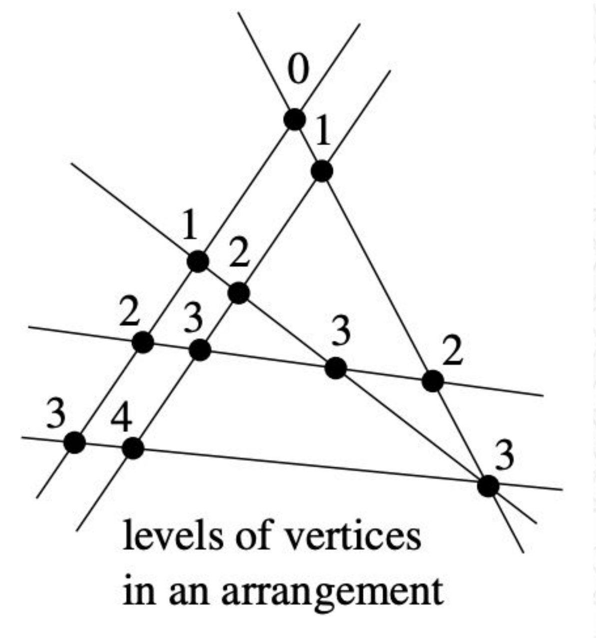
- So our primary concern:
  Is the number of samples in the half space above the line proportional to the area of the square pixel above the line?

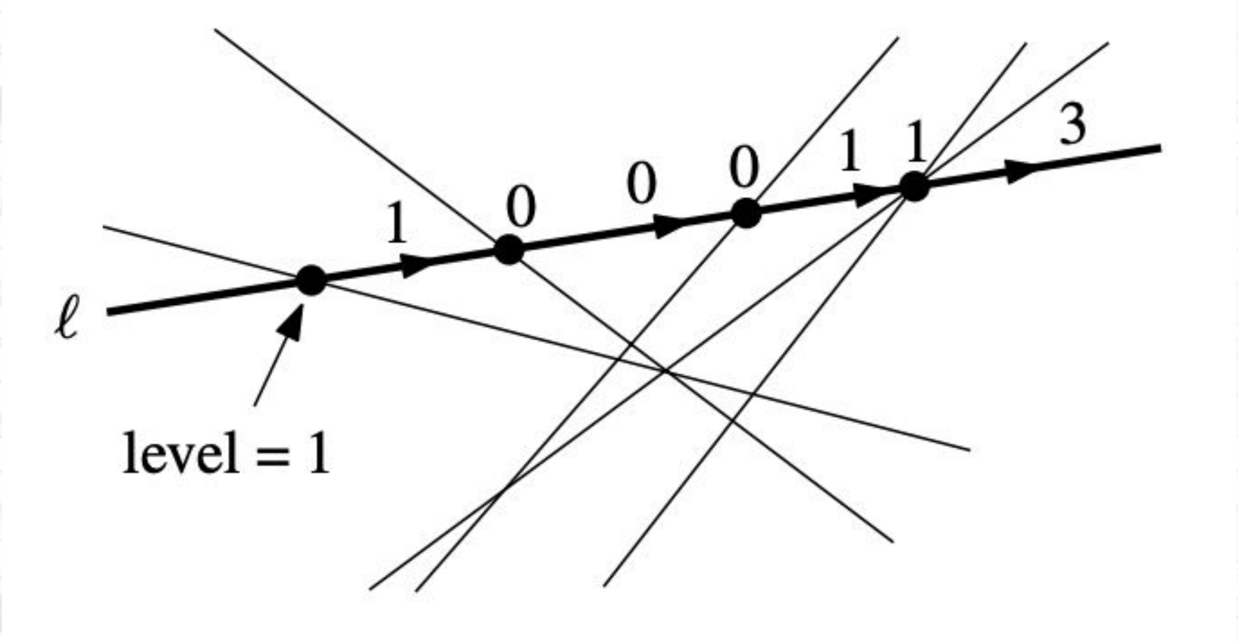




Compute the Discrepancy of a Sampling

Arrangements allow us to compute in $O(n^2)$ the level, or number of line segments above any point



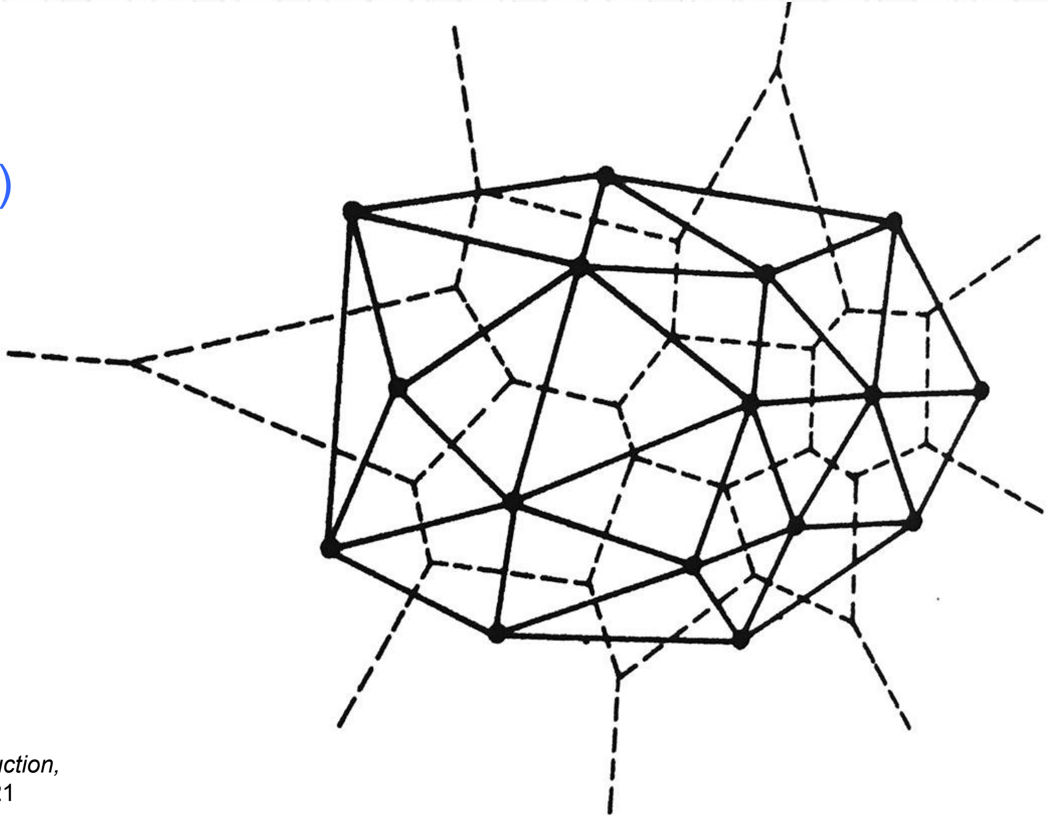


Outline for Today

- Last Lecture: Problems that reduce to Voronoi Diagrams
- Duality: Points $\leftrightarrow$ Lines
- Arrangement of Lines
- Complexity of an Arrangement of Lines
- Algorithm to Construct Arrangement of Lines
- Arrangement Application: Ray Tracing Supersampling
- Next Time: Arrangement Application: Architectural Sketching
- Next Time: Delaunay Triangulations

Next Time: Delaunay Triangulation!

- The Voronoi Diagram (VD) *is the dual of the* Delaunay Triangulation (DT)
- Every Voronoi Site is a face in Voronoi Diagram and a vertex in the DT
- Every Voronoi Edge is an edge in the DT
- Every Voronoi Vertex is a triangle in the DT



Computational Geometry: An Introduction,
Preparata & Shamos, Figure 5.21