Computational Geometry (CS 437/537)

Alon Efrat

http://www.cs.arizona.edu/~alon

Chapter 1 – Introduction

Slides are gratitude of

Craig Gotsman

11.

Bibliography

- **Computational Geometry**
  de Berg, van Kreveld, Overmars, Schwarzkopf,
  2nd edition, Springer Verlag, 2000.

- **Computational Geometry in C**
  O'Rourke,
  2nd edition, Cambridge Univ. Press, 2000.

- **Course notes,** *D. Mount*
- **Course slides,** *C. Gotsman*

21.

Assessment

- **6 Homework Assignments** (65%). Primarily theoretical problems.
  - (7 homework, only the 6 better ones are counted)

- **Final Exam (10%)**
- **Midterm (10%)**
- **Max(Final, Midterm) (10%)**

- **Class Participation (5%)**.

Grads have one more question in each hw.

31.

Syllabus

- Introduction
- Basic techniques
- Basic data structures
- Polygon triangulation
- Linear programming
- Range searching
- Point location
- Voronoi diagrams
- Duality and Arrangements
- Delaunay triangulations
- Computer graphics applications

41.

Questions ?

51.

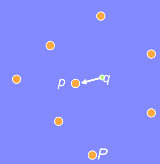
Lecture Topics

- Sample problems
- Basic concepts
- Convex hull algorithms

61.

Nearest Neighbor

- Problem definition:
  - Input: a set of points (*sites*) *P* in the plane and a query point *q*.
  - Output: The point $p \in P$ closest to *q* among all points in *P*.

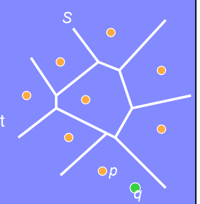


- Rules of the game:
  - One point set, multiple queries
- Applications:
  - Store Locator
  - Cellphones

71.

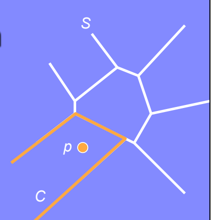
The Voronoi Diagram

- Problem definition:
  - Input: a set of points (*sites*) *P* in the plane.
  - Output: A planar subdivision *S* into cells. One cell per site. A a point *q* lies in the cell corresponding to a site $p \in P$ iff *p* is the nearest site to *q*.

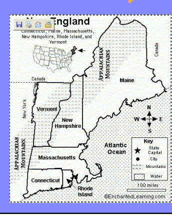


81.

Point Location



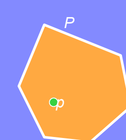
- Problem definition:
  - Input: A partition *S* of the plane into cells and a query point *p*.
  - Output: The cell $C \in S$ containing *p*.
- Rules of the game:
  - One partition, multiple queries
- Applications:
  - Nearest neighbor
  - State locator.



91.

Point in Polygon

- Problem definition:
  - Input: a polygon *P* in the plane and a query point *p*.
  - Output: *true* if $p \in P$, else *false*.

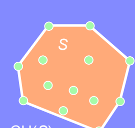


- Rules of the game:
  - One polygon, multiple queries

101.

Convex Hull

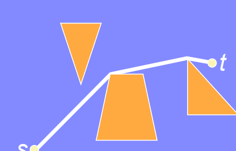
- Problem definition:
  - Input: a set of points *S* in the plane.
  - Output: Minimal convex polygon containing *S*.



CH(*S*)

111.

Shortest Path

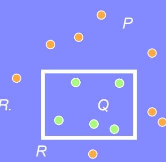


- Problem definition:
  - Input: Obstacles locations and *query* endpoints *s* and *t*.
  - Output: the shortest path between *s* and *t* that avoids all obstacles.
- Application: Robotics.

121.

Range Searching and Counting

- Problem definition:
  - Input: A set of points *P* in the plane and a query rectangle *R*
  - Output: (report) The subset $Q \subseteq P$ contained in *R*.
    (count) The size of *Q*.

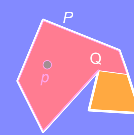


- Rules of the game:
  - One point set, multiple queries.
- Application: Urban planning, data-bases

131.

Visibility

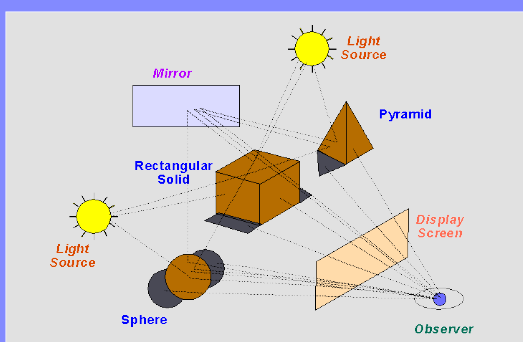
- Problem definition:
  - Input: a polygon *P* in the plane and a query point *p*.
  - Output: Polygon Q ⊆ P, visible to p.



- Rules of the game:
  - One polygon, multiple queries
- Applications: Security

141.

Ray Tracing



151.

Questions ?

171.

Basic Concepts

181.

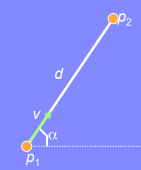
Complexity (reminder)

Symbol	Definition	"Nickname"
$f(n) = O(g(n))$	$\exists N,C\ \forall n>N\ f(n)/g(n) \le C$	"≤"
$f(n) = \Omega(g(n))$	$g(n) = O(f(n))$	"≥"

191.

Representing Geometric Elements

- Representation of a line segment by four real numbers:
  - Two endpoints (p_1 and p_2)
  - One endpoint (p_1), a slope (α), and length (d)
  - One endpoint (p_1), vector direction (v) and parameter interval length (t)
  - Parametric form



$$p(t) = p_1 + t(p_2 - p_1) = (1-t)p_1 + tp_2, \quad t \in [0,1]$$

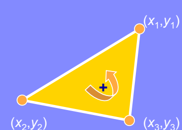
- Different representations may affect the numeric accuracy of algorithms…

201.

Orientation

$$\text{Area} = \frac{1}{2}\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

$=1/2\ (\ x_1(y_2-y_3) - x_2(y_1-y_3) + x_3(y_1-y_2)\ )$



- The sign of the area indicates the orientation of the points.
- Positive area ≡ counterclockwise orientation ≡ left turn.
- Negative area ≡ clockwise orientation ≡ right turn.

- **Question:** How can this be used to determine whether a given point is "above" or "below" or "on" a given line segment ? Is this numerically stable ?

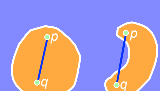
211.

Convex Hull Algorithms

221.

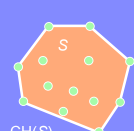
Convexity and Convex Hull

- A set S is *convex* if any pair of points $p,q \in S$ satisfy $pq \subseteq S$.



- The *convex hull* of a set S is:
  - The minimal convex set that contains S, i.e. any convex set C such that $S \subseteq C$ satisfies $CH(S) \subseteq C$.
  - The intersection of all convex sets that contain S.
  - The set of all convex combinations of $p_i \in S$, i.e. all points of the form:

$$\sum_{i=1}^{n} \alpha_i p_i\ , \qquad \alpha_i \ge 0,\ \sum_{i=1}^{n} \alpha_i = 1$$

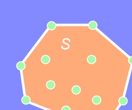


CH(S)

231.

Convex Hulls – Some Facts

- The convex hull of a set is unique.
- The boundary of the convex hull of a point set is a polygon on a subset of the points.

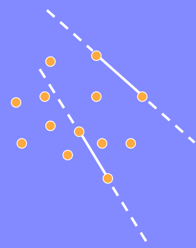


CH(S)

241.

Convex Hull – Naive Algorithm

- Description:
  - For each pair of points construct its connecting segment and *supporting line*.
  - Find all the segments whose supporting lines divide the plane into two halves, such that one half plane contains *all* the other points.
  - Construct the convex hull out of these segments.

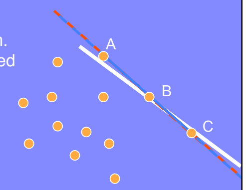


- Time complexity:
  - All pairs: $O\left(\binom{n}{2}\right) = O\left(\frac{n(n-1)}{2}\right) = O(n^2)$
  - Check all points for each pair: O(n)
  - Total: O(n^3)

251.

Possible Pitfalls

- Degenerate cases – e.g. 3 collinear points. Might harm the correctness of the algorithm. Segments AB, BC and AC will *all* be included in the convex hull.

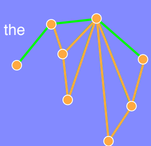


- Numerical problems – We might conclude that *none* of the three segments belongs to the convex hull.

261.

Convex Hull – Graham's Scan

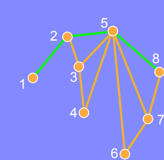
- Ideas: Sort the points according to their *x* coordinates. First we construct only the upper CH.
- Process the points from the leftmost to rightmost.
- Maintain the upper CH of all points from the leftmost one to the currently processed scanned point.
- Develop the left-turn critiria for the last 3 processed points:
  - if we need to turn left when traveling along these points, the middle one is NOT on the upper CH, and we delete it.
  - Note: After deletion, we have new 3 points to consider.



271.

The Algorithm

- Sort the points in increasing order of *x*-coord:
  $p_1, \ldots, p_n$.
  /* Note – this is the only part not done in O(n) time */
- Push(S,p_1); Push(S,p_2);
- For i = 3 to n do
  - While Size(S) ≥ 2 and
    - Orient(p_i, top(S),second(S)) ≤ 0 /* left turn */
      - do Pop(S);
  - Push(S,p_i);
- Print(S);



281.

Graham's Scan – Time Complexity

- Sorting – O(n log n)
- If D_i is number of points popped on processing p_i,

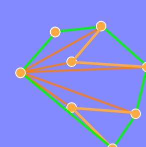
$$\text{time} = \sum_{i=1}^{n}(D_i + 1) = n + \sum_{i=1}^{n} D_i$$

- Each point is pushed on the stack only once.
- Once a point is popped – it cannot be popped again.
- Hence $\sum_{i=1}^{n} D_i \le n$
- Question: What is actually $\sum_{i=1}^{n} D_i \le n$?

291.

Graham's Scan– a Variant

- Assume the points are given in increasing x-coord order.
- Time Complexity: O(n log n)

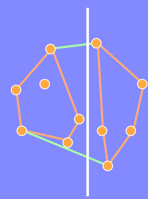


- **Question:** What are the pros and cons of this algorithm relative to the previous ?

301.

Convex Hull - Divide and Conquer

- Algorithm:
  - Find a point with a median *x* coordinate (time: O(*n*))
  - Compute the convex hull of each half (recursive execution)
  - Combine the two convex hulls by finding common *tangents*.
    This can be done in O(*n*).



- Complexity: O(*n* log *n*)

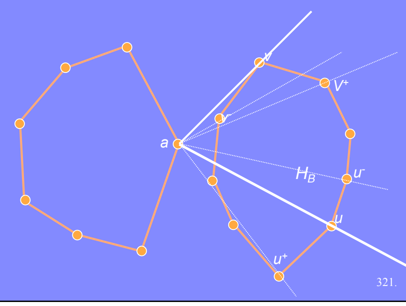
$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

311.

Finding Common Tangents

A tangent line – a line cutting the CH at a single point

Consider a line passing through a vertex *v* of H_B. How can we determine if *v* is a tangent to H_B.



321.

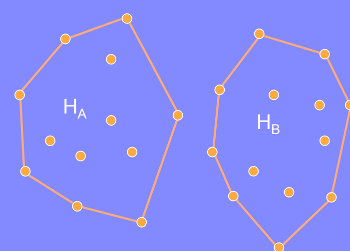
Finding Common Tangents

To find **lower** tangent:

- Find *a* - the rightmost point of H_A } O(n)
- Find *b* – the leftmost point of H_B
- While *ab* is not a lower tangent for H_A and H_B, do:
  - If *ab* is not a lower tangent to H_A do *a = a*-1
    - /* Move one point **clockwise** */
  - If *ab* is not a lower tangent to H_B do *b = b*-1
    - /* Move one point **counterclockwise** */

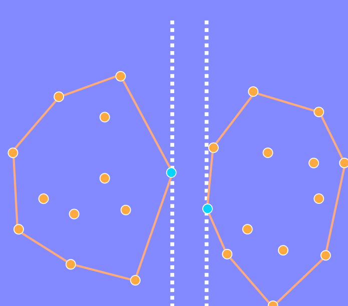
331.

Finding Common Tangents



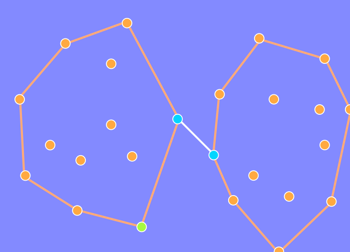
341.

Finding Common Tangents

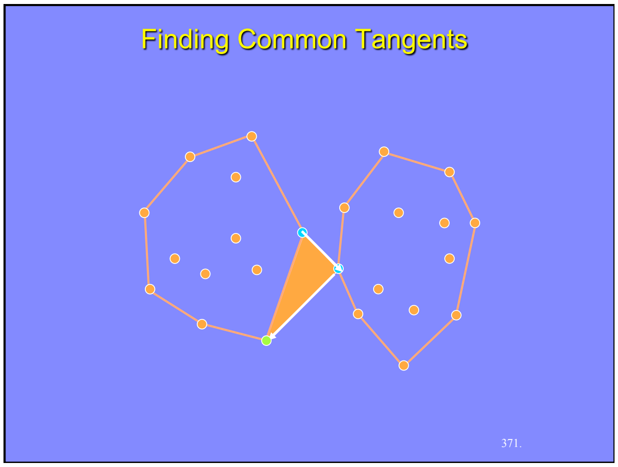


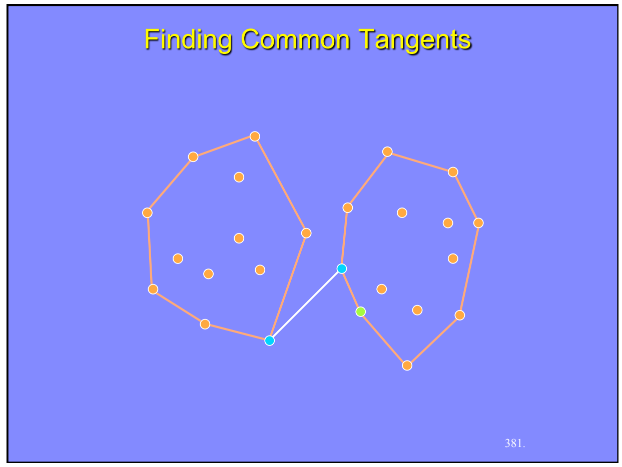
351.

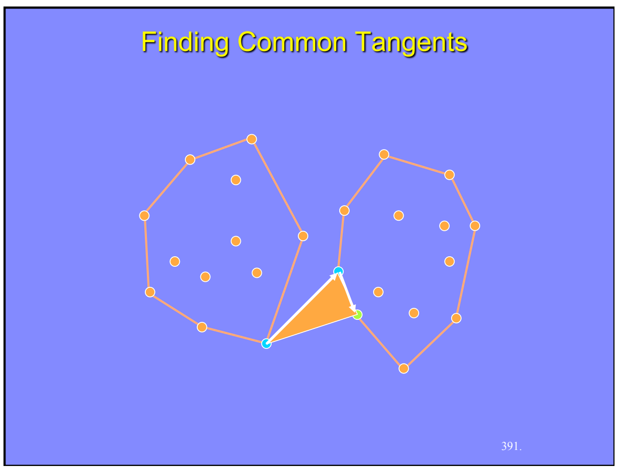
Finding Common Tangents

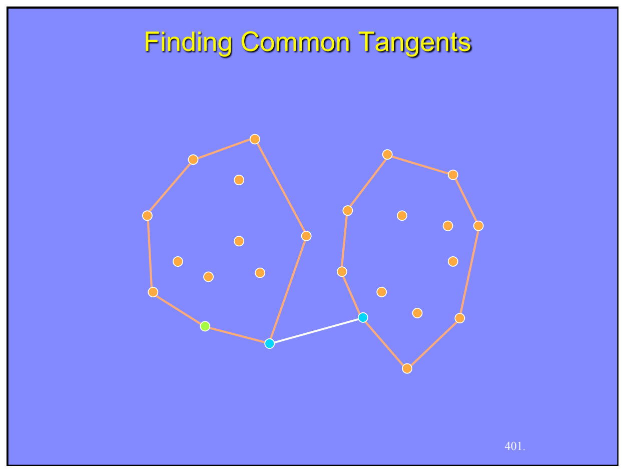


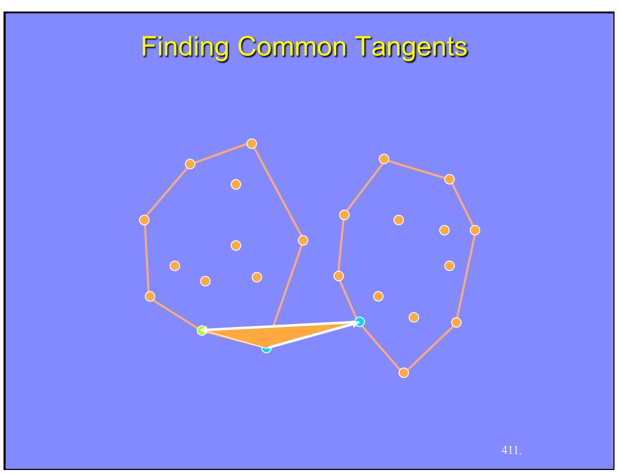
361.

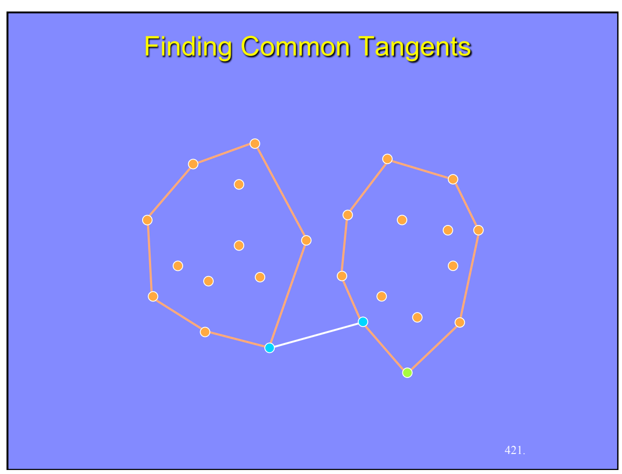

Finding Common Tangents
371.


Finding Common Tangents
381.

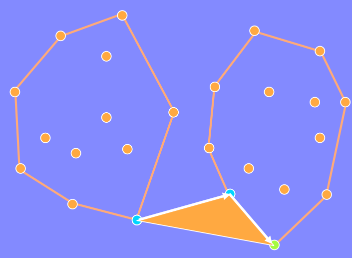

Finding Common Tangents
391.


Finding Common Tangents
401.


Finding Common Tangents
411.

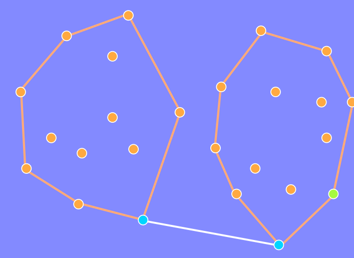

Finding Common Tangents
421.

Finding Common Tangents



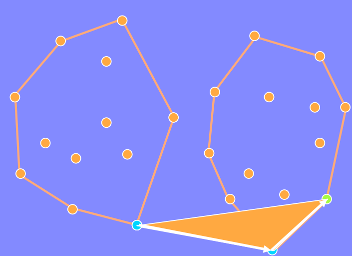
431.

Finding Common Tangents



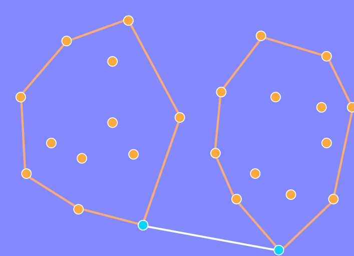
441.

Finding Common Tangents



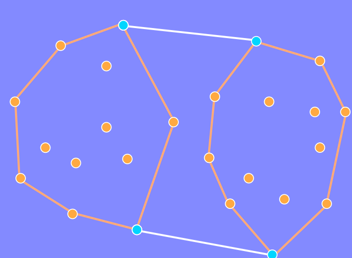
451.

Finding Common Tangents



461.

Finding Common Tangents

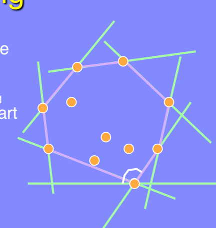


471.

Output-Sensitive Convex Hull Gift Wrapping

- Algorithm:
  - Find a point p_1 on the convex hull (e.g. the lowest point).
  - Rotate counterclockwise a line through p_1 until it touches one of the other points (start from a horizontal orientation).

    Question: How is this done ?
  - Repeat the last step for the new point.
  - Stop when p_1 is reached again.



- Time Complexity: O(nh), where n is the input size and h is the output (hull) size.
- Best alg in 2D: O(n log h)

481.

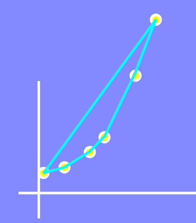
General Position

- When designing a geometric algorithm, we first make some simplifying assumptions, e.g:
  - No 3 colinear points.
  - No two points with the same *x* coordinate.
  - etc.
- Later, we consider the general case:
  - How should the algorithm react to degenerate cases ?
  - Will the correctness be preserved ?
  - Will the runtime remain the same ?

491.

Lower Bound for Convex Hull

- A reduction from sorting to convex hull is:
  - Given *n* real values x_i, generate *n* 2D points on the graph of a convex function, e.g. (x_i,x_i^2).
  - Compute the (ordered) convex hull of the points.
  - The order of the convex hull points is the numerical order of the x_i.



- So CH=$\Omega(n\lg n)$

501.