

CS 5114

- Manber's notation: -p q-.
 A line segment is also represented by a pair of distinct
- A line segment is also represented by a pair of distinct points: the endpoints.
 - Notation: p q.
- A <u>path</u> *P* is a sequence of points *p*₁, *p*₂, ..., *p_n* and the line segments *p*₁ − *p*₂, *p*₂ − *p*₃, ..., *p_{n-1}* − *p_n* connecting them.
- A closed path has $p_1 = p_n$. This is also called a polygon.
 - Points \equiv vertices.
 - A polygon is a sequence of points, not a set.

Spring 2014 3 / 3

Definitions (cont)

- Simple Polygon: The corresponding path does not intersect itself.
 - A simple polygon encloses a region of the plane INSIDE the polygon.
- Basic operations, assumed to be computed in constant time:
 - Determine intersection point of two line segments.
 - Determine which side of a line that a point lies on.
 - Determine the distance between two points.

CS 5114: Theory of Algorithms

no notes

Definitions (cont)

CS 5114 CS 5114

triangle are not the same even if they occupy the same space.

Point in Polygon

Problem: Given a simple polygon P and a point q, determine whether q is inside or outside P.

Basic approach:

- Cast a ray from q to outside P. Call this L.
- Count the number of intersections between *L* and the edges of *P*.
- If count is even, then q is outside. Else, q is inside.

Problems:

- How to find intersections?
- Accuracy of calculations.

Special cases.

CS 5114: Theory of Algorithms

Spring 2014 5 / 39

Spring 2014 6 / 39

Point in Polygon Analysis (1)

Time complexity:

- Compare the ray to each edge.
- Each intersection takes constant time.
- Running time is O(n).

Improving efficiency:

- O(n) is best possible for problem as stated.
- Many lines are "obviously" not intersected.

CS 5114: Theory of Algorithms

Point in Polygon Analysis (2)

Two general principles for geometrical and graphical algorithms:

- Operational (constant time) improvements:
 - Only do full calculation for 'good' candidates
 - Perform 'fast checks' to eliminate edges.
 - ► Ex: If *p*₁.*y* > *q*.*y* and *p*₂.*y* > *q*.*y* then don't bother to do full intersection calculation.
- When doing many point-in-polygon operations, preprocessing may be worthwhile.
 - Ex: Sort edges by min and max y values.
 Only check for edges covering y value of point q.

CS 5114: Theory of Algorithms

Constructing Simple Polygons

Problem: Given a set of points, connect them with a simple closed path.

Approaches:

CS 5114: Theory of Algorithms

- Randomly select points.
- Use a scan line:
 - Sort points by y value.
 - Connect in sorted order.
- Sort points, but instead of by y value, sort by angle with respect to the vertical line passing through some point.
 - Simplifying assumption: The scan line hits one point at a time.
 - Do a rotating scan through points, connecting as you go.

Spring 2014 8 / 39

Spring 2014





Special cases:

- · Line intersects polygon at a vertex, goes in to out.
- Line intersects poly. at inflection point (stays in or stays out).
- Line intersects polygon through a line.

Simplify calculations by making line horizontal.

Accuracy of calculations is not a problem with integer coordinates for points and a horizontal line. But think about representing the intersection point for two arbitrary line segements (from a polygon intersection operation). Cascading intersections can lead to ever-increasing demand for precision in coordinate representation.

듓 CS 5114	Point in Polygon Analysis (1)
Point in Polygon Analysis (1)	Time complexity: - Compares have to such edge. - Calarity intermediant tables constant if max. - Exhibit tables and to (r) (r). - Deposing efficiency: - K(r)(i) is built possible for problem as initial. - Many if maximize the "deboardy" for intermedial.

no notes



Spatial data structures can help.

"Fast checks" take time. When they "win" (they rule something out), they save time. When they "lose" (they fail to rule something out) they add extra time. So they have to "win" often enough so that the time savings outweighs the cost of the check.

Constructing Simple Polygons	Constructing Simple Polygons Packer: Const and practice.consect from with a strate description.constrate from the strate strate description.constrate strate strate strate description.constrate strate strate strate strate description.constrate strate strate strate strate strate strate strate strate description.constrate strate strate strate strate strate strate strate strate strate description.constrate strate strate strate strate strate strate strate strate strate description.constrate strate strate strate strate strate strate strate strate strate description.constrate strate str

(1) Could easily yield an intersection.

(2) The problem is connecting point p_n back to p_1 . This could yield an intersection.

Simplifying assumption is that the points are not colinear w.r.t. the scan line.

See Manber Figure 8.6.

Validation

Theorem: Connecting points in the order in which they are encountered by the rotating scan line creates a simple polygon.

Proof:

- Denote the points p_1, \dots, p_n by the order in which they are encountered by the scan line.
- For all *i*, 1 ≤ *i* < *n*, edge *p_i* − *p_{i+1}* is in a distinct slice of the circle formed by a rotation of the scan line.
- Thus, edge $p_i p_{i+1}$ does not intersect any other edge.
- Exception: If the angle between points *p_i* and *p_{i+1}* is greater than 180∘.

CS 5114: Theory of Algorithms

CS 5114: Theory of Algorithms

Spring 2014 9 / 39

Implementation

How do we find the point for the scanline center?

Actually, we don't care about angle - slope will do.

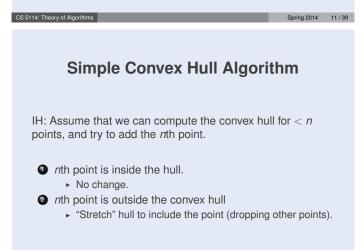
Select *z*; for (*i* = 2 to *n*) compute the slope of line $z - p_i$. Sort points p_i by slope; label points in sorted order;

Time complexity: Dominated by sort.

Spring 2014 10 / 39

Convex Hull

- A <u>convex hull</u> is a polygon such that any line segment connecting two points inside the polygon is itself entirely inside the polygon.
- A **convex path** is a path of points p_1, p_2, \dots, p_n such that connecting p_1 and p_n results in a convex polygon.
- The convex hull for a set of points is the smallest convex polygon enclosing all the points.
 - imagine placing a tight rubberband around the points.
- The point **belongs** to the hull if it is a vertex of the hull.
- Problem: Compute the convex hull of n points.



So, the key is to pick a point for the center of the rotating scan that guarentees that the angle never reachese $180\circ$.

E CS	S 5114	Implementation
-		How do we find the point for the scanline center?
-03	1	Actually, we don't care about angle - slope will do.
2014-		Solve x : the $(x - 2x - 1x - 1x)$ the compute the aloge of the $x - p$. Solve points, p_1 by larger: label points in section order;
		Time complexity: Dominated by sort

Pick as z the point with greatest x value (and least y value if there is a tie). See Manber Figure 8.7.

The next point is the next largest angle between $z - p_i$ and the vertical line through *z*. It is important to use the slope, because then our computation is a constant-time operation with no transendental functions.

z is the point with greatest x value (minimum y in case of tie)

So, time is $\Theta(n \log n)$

CS 5114 Convex Hull	Convex Hull 4. success the su
no notes	

က ^{CS 5114}		Simple Convex Hull Algo
Conversion Simple Conversion	ex Hull Algorithm	H: Assume that we can compute the convex to points, and try to add the rth point. In the point is inside the hult. No charge. In the point is custicle the convex hull "Stretch" hull to include the point (dropping)

See Manber Figure 8.9.

Spring 2014 12 / 39

Subproblems (1)

Potential problems as we process points:

- Determine if point is inside convex hull.
- Stretch a hull.

CS 5114: Theory of Algorithms

The straightforward induction approach is inefficient. (Why?)

Our standard induction alternative: Select a special point for the *n*th point – some sort of min or max point.

If we always pick the point with max *x*, what problem is eliminated? Stretch:

Find vertices to eliminate

Add new vertex between existing vertices.

Spring 2014 13 / 39

Spring 2014

Spring 2014

Spring 2014 16 / 39

15/39

14/39

Subproblems (2)

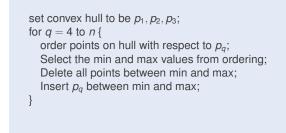
Supporting line of a convex polygon is a line intersecting the polygon at exactly one vertex.

Only two supporting lines between convex hull and max point *q*.

These supporting lines intersect at "min" and "max" points on the (current) convex hull.

CS 5114: Theory of Algorithms

Sorted-Order Algorithm



CS 5114: Theory of Algorithms

Time complexity

Sort by x value: $O(n \log n)$.

For *q*th point:

CS 5114: Theory of Algorithms

- Compute angles: O(q)
- Find max and min: O(q)
- Delete and insert points: *O*(*q*).

$$T(n) = T(n-1) + O(n) = O(n^2)$$

Why? Lots of points don't affect the hull, and stretching is expensive.

Subproblem 1 can be eliminated: the max is always outside the polygon.

က္ ^{CS 51}	14	Subproblems (2)
4-03.	Subproblems (2)	Supporting line of a convex polygon is a line intersecting the polygon at exactly one vertex. Only two supporting lines between convex hull and max
2014		point q. These supporting lines intersect at 'min' and 'max' points on the (current) convex hull.

"Min" and "max" with respect to the angle formed by the supporting lines.

φ CS 5114	Sorted-Order Algorithm
Sorted-Order Algorithm	set convex hull to be $p,p,p(:)$ for $q \to 0,n$ (if for $q \to 0,n$ (), and $q \to 0,n$ (), and $q \to 0,n$ (), the set of n as values from ordering the set of n as values from ordering the set of n and n and n (). Set of $p,$ between mix and max ()

Sort by x value.



no notes

Gift Wrapping Concept

- Straightforward algorithm has inefficiencies.
- Alternative: Consider the whole set and build hull directly.
- Approach:
 - Find an extreme point as start point.
 - ► Find a supporting line.
 - Use the vertex on the supporting line as the next start point and continue around the polygon.
- Corresponding Induction Hypothesis:
 - Given a set of *n* points, we can find a convex path of length *k* < *n* that is part of the convex hull.
- The induction step extends the PATH, not the hull.

CS 5114: Theory of Algorithms Spring 2014 17 / 39 Gift Wrapping Algorithm ALGORITHM GiftWrapping(Pointset S) { ConvexHull P; $P = \emptyset;$ Point p = the point in S with largest x coordinate; $P = \dot{P} \cup p;$ Line L = the vertical line containing p; while (P is not complete) do { Point q = the point in S such that angle between line -p - q - and *L* is minimal along all points; $P = P \cup q$; L = -p - q -;p = q;} } CS 5114: Theory of Algorith Spring 2014 18/39 **Gift Wrapping Analysis**

Complexity:

- To add *k*th point, find the min angle among n k lines.
- Do this *h* times (for *h* the number of points on hull).
- Often good in average case.
- Could be bad in worst case.

CS 5114: Theory of Algorithms

Spring 2014 19 / 39

Spring 2014 20 / 39

Graham's Scan

- Approach:
 - Start with the points ordered with respect to some maximal point.
 - Process these points in order, adding them to the set of processed points and its convex hull.
 - Like straightforward algorithm, but pick better order.
- Use the Simple Polygon algorithm to order the points by angle with respect to the point with max *x* value.
- Process points in this order, maintaining the convex hull of points seen so far.



CS 5114

2014-03-31



Straightforward algorithm spends time to build convex hull with points interior to final convex hull.

CS 5114 Gift Wrapping Algorithm Gift Wrapping Algorit

no notes



$O(n^2)$. Actually, O(hn) where h is the number of edges to hull.



See Manber Figure 8.11.

Graham's Scan (cont)

Induction Hypothesis:

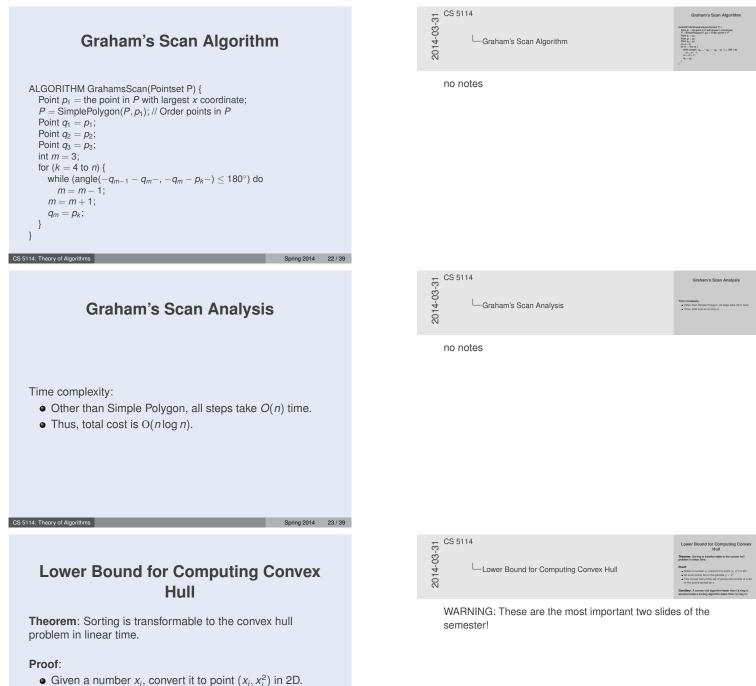
• Given a set of *n* points ordered according to algorithm Simple Polygon, we can find a convex path among the first n-1 points corresponding to the convex hull of the n-1 points.

Induction Step:

- Add the kth point to the set.
- Check the angle formed by p_k, p_{k-1}, p_{k-2} .
- If angle < 180° with respect to inside of the polygon, then delete p_{k-1} and repeat.

CS 5114: Theory of Algorithms

Spring 2014 21 / 39



• The convex hull of this set of points will consist of a list of the points sorted by *x*.

Corollary: A convex hull algorithm faster than O(*n* log *n*) would provide a sorting algorithm faster than $O(n \log n)$. CS 5114: Theory of Algorithms



no notes

"Black Box" Model

A Sorting Algorithm:

keys \rightarrow points: $O(n)$
Convex Hull
CH Polygon \rightarrow Sorted Keys: $O(n)$

CS 5114: Theory of Algorithms Spring 2014 25 / 39

<u>-</u>	CS 5114	
- 20-		i
ZU 14	└─-"Black Box" Model	I
V		

2

This is the fundamental concept of a reduction. We will use this constantly for the rest of the semester.

"Black Box" Model