

COMPUTATIONAL GEOMETRY

Review for Midterm

1 Art Gallery Problems

1.1 Basic definitions

Simple polygon, visibility, clear visibility, convex vertex, reflex vertex, monotone polygon, star-shaped polygon.

1.2 Art Gallery Theorem

$\lfloor n/3 \rfloor$ guards are always sufficient and sometimes necessary. (For polygons with h holes, the number is $\lfloor (n+h)/3 \rfloor$; this is very tricky to show – I did not give the proof.)

Understand Fisk's proof and how to use it to find a set of at most $\lfloor n/3 \rfloor$ guards (as on HW1).

Understand Chvatal's comb example, which is a "generic class of examples" (i.e., it is a family of examples that includes n -gons for arbitrarily large n), showing that $\lfloor n/3 \rfloor$ guards are sometimes necessary)

1.3 Computing $g(P)$

Be able to compute the guard number, $g(P)$, of a simple polygon and justify your answer (e.g., using "witness points"). You should be able to do this both for point guards and for vertex guards. You should think also how you may do it for clear visibility problems.

Be able to reason about slight variations to the standard models and problems.

2 Geometric Primitives

You should understand all of the basics of Section 1.4, including the predicates `Area2`, `Left`, `LeftOn`, `Collinear`, `IntersectProp`, `Between`, and `Intersect`, `InCone`, `Diagonalie`, `Diagonal`. You should also understand how to perform variations on these or to use them as building blocks for simple other predicates.

3 Triangulation

3.1 Basics

Understand the definition of diagonal, the existence of triangulations, Meisters 2-Ears Theorem, number of diagonals $(n - 3)$, number of triangles $(n - 2)$ in a simple polygon triangulation.

3.2 Ear-Clipping

Be able to perform the ear clipping algorithm `Triangulate`, as you did on HW2. Understand the algorithm and its running time (worst-case $O(n^2)$).

3.3 Trapezoidalization by Plane Sweep

Understand the basics of a plane sweep algorithm: events (in an event queue), and the “sweep line status” (SLS) data structure. Understand how it applies to compute a trapezoidalization of a polygon (even if it has holes), in time $O(n \log n)$.

3.4 Monotone Polygons and Monotone Mountains

Know the definitions and how to tell if a polygon is monotone, and if so, with respect to what directions.

Monotone polygons can be triangulated in $O(n)$ time (as can any simple polygon), using a very simple algorithm (not true for arbitrary simple polygons – arbitrary simple polygons have a linear-time algorithm that is VERY complicated!). We did not see the details of how to do it. Instead, we did show the details of how to triangulate a monotone mountain and you should be able to do that, using the rule I gave to make it specific: clip off the highest strictly convex vertex that is not a base endpoint (as in HW3).

3.5 Efficient Algorithms

Bottom line: Chazelle’s algorithm triangulates a simple polygon in time $O(n)$.

The algorithm we covered, based on the plane sweep trapezoidalization, runs in time $O(n \log n)$: convert the trapezoidalization to a decomposition (using only diagonals) of P into monotone polygons (or into monotone mountains) in time $O(n)$, then triangulate each such piece using the simple linear-time algorithm for monotone polygons (or monotone mountains).

For polygons with h holes (and total of n vertices, including those on the hole boundaries), we can use our algorithm to get $O(n \log n)$ time. The best known is $O(n + h \log^{1+\epsilon} h)$ (Bar-Yehuda and Chazelle), but we suspect that the real answer will be $\Theta(n + h \log h)$.

4 Convex Hulls in the Plane

4.1 Basics

Understand the definitions of convexity.

4.2 Algorithms

Bottom line: the “ultimate” convex hull algorithm can find the convex hull of n points in $O(n \log h)$ time, where h is the output size. We did not see how this was done, but we saw several ways to achieve $O(n \log n)$: Graham scan (which does an $O(n \log n)$ sort, either by angle or by x -coordinate, etc, followed by the “scan” in which we push and pop on a stack), incremental insertion (using the fact that binary search can yield tangency points in $O(\log n)$ time), divide-and-conquer (which leads to the recurrence $T(n) \leq 2T(n/2) + O(n)$, based on being able to do a linear-time merge of two nonoverlapping convex hulls (by the “wobbling stick” to find each of the two outer common tangents), which solves to $O(n \log n)$).

Understand the basics of these algorithms: Graham scan, incremental insertion, divide-and-conquer (how to merge), QuickHull, gift-wrapping ($O(nh)$).

Understand Graham’s algorithm and be able to apply it (as on HW4).

4.3 Convex Hulls of Simple Polygons and Chains

If the points are given to us in an order that defines a simple (non-self-intersecting) chain or polygon, then $O(n)$ time is possible for computing a convex hull: We saw Melkman’s algorithm in detail. You should be able to execute it on a simple example (as on the homework and practice exam).