Computational Geometry

Subhash Suri

Computer Science Department
UC Santa Barbara

Feb. 2, 2001

What is CG?

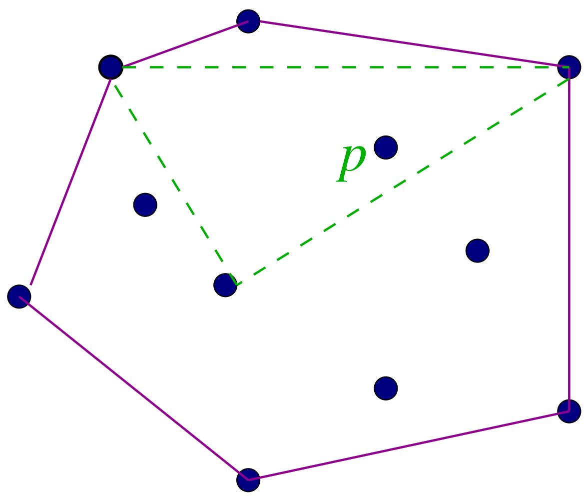
- **Algorithmic study of geometric problems.**
- **Finite sets.**
- **Discrete and combinatorial methods.**
- **Optimal Solutions.**
- **Worst-case time complexity.**

Outline

- **Convex Hulls**
  1. **Algorithms**
  2. **Applications**
- **Triangulations**
  1. **Existence**
  2. **Applications**
  3. **Special kind of triangulations**
- **Research directions.**

Convex Hull

- **Smallest convex set containing S.**
- **Archimedes, Fourier, Newton.**
- **Classical methods concerned with finiteness, not efficiency.**



- $p \notin CH(S)$ **iff** $\exists a, b, c \in S$ **such that** $p \in \triangle abc$.
- **Leads to an** $O(N^4)$ **time algorithm.**

CG Approach

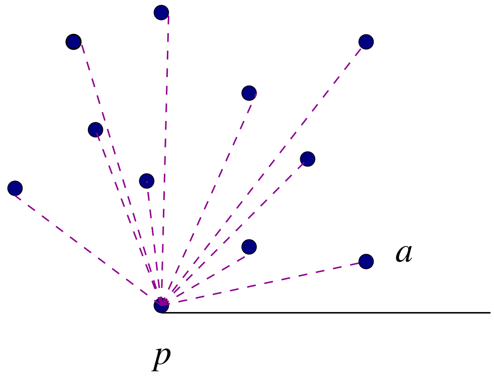
1. **Efficient algorithms require new characterizations.**

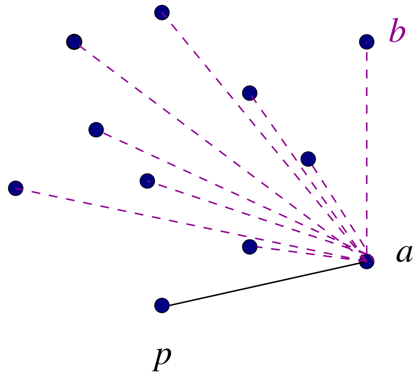
2. **Start with the bottom point p.**

3. **Angularly sort remaining points, around p.**

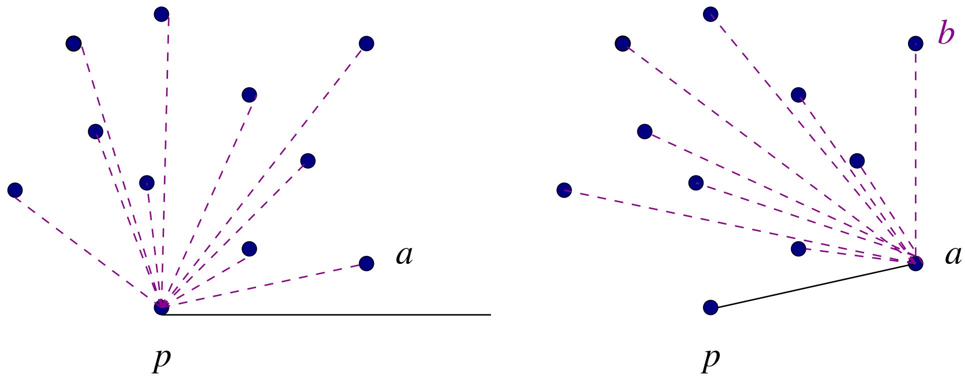
4. **Point a with smallest angle is on CH.**

5. **Repeat algorithm at a.**





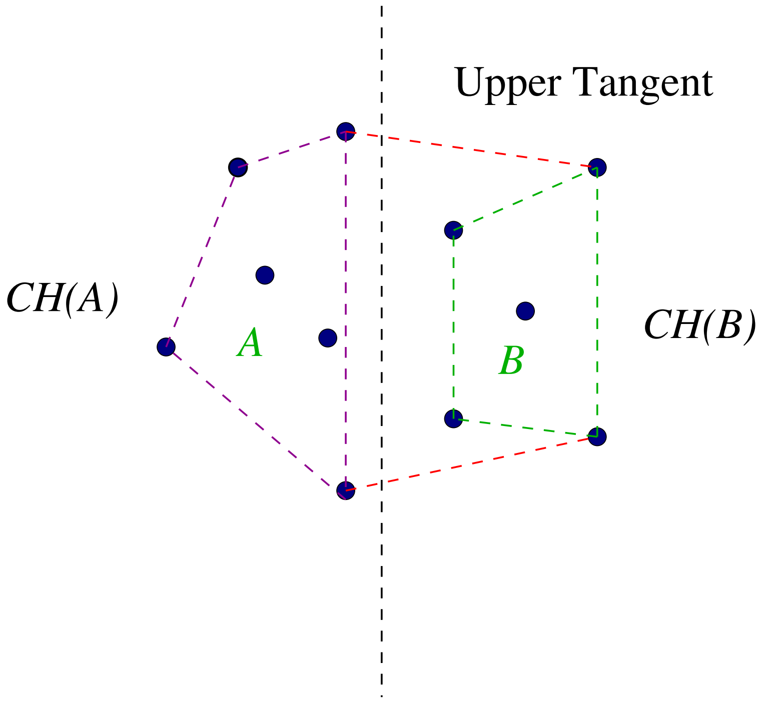
Gift Wrapping Algorithm



- **This algorithm is called Gift-Wrapping.**
- **Discovering each new CH vertex takes $O(N)$ time.**
- **Algorithm runs in worst-case $O(Nh)$ time; $3 \leq h \leq N$.**

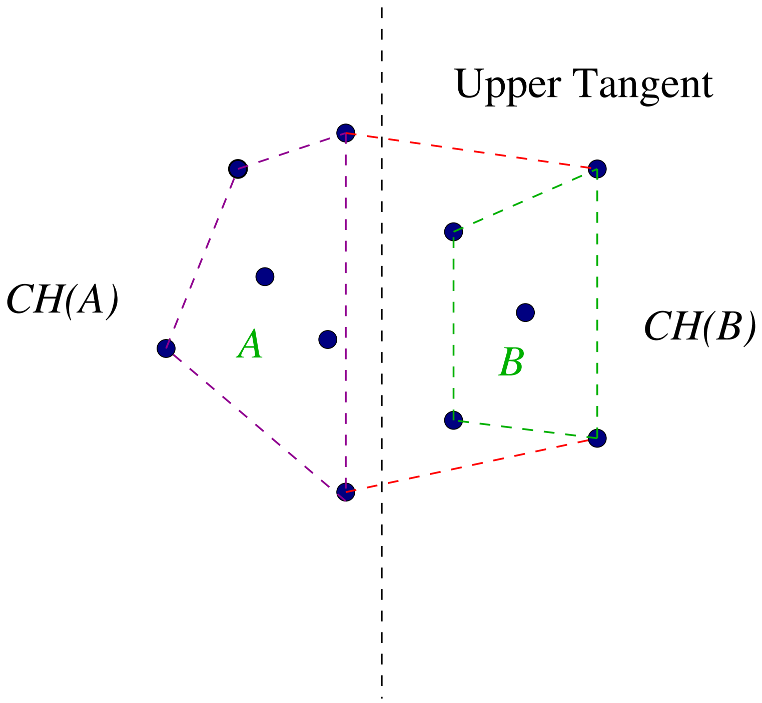
Divide and Conquer

1. **Gift-Wrap is not optimal.**
2. **Divide and Conquer works in $O(N \log N)$ time.**



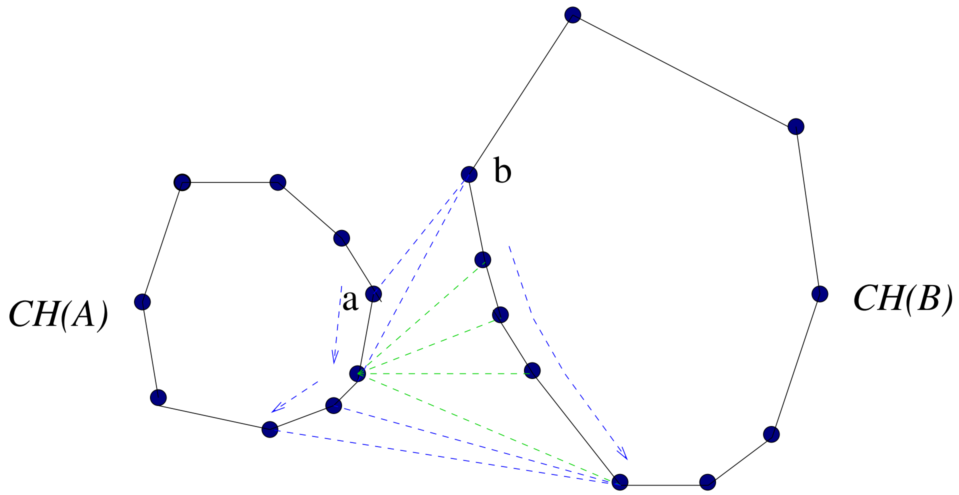
3. **Divide the problem into two halves; Recursively solve the subproblems; Merge the solutions.**
4. **Merging involves computing Upper and Lower tangent lines.**

Divide and Conquer



- **Sort points by X-coordinates.**
- **Let A be the set of $n/2$ leftmost points, and B the set of $n/2$ rightmost points.**
- **Recursively compute $CH(A)$ and $CH(B)$.**
- **Merge $CH(A)$ and $CH(B)$ to obtain $CH(S)$.**

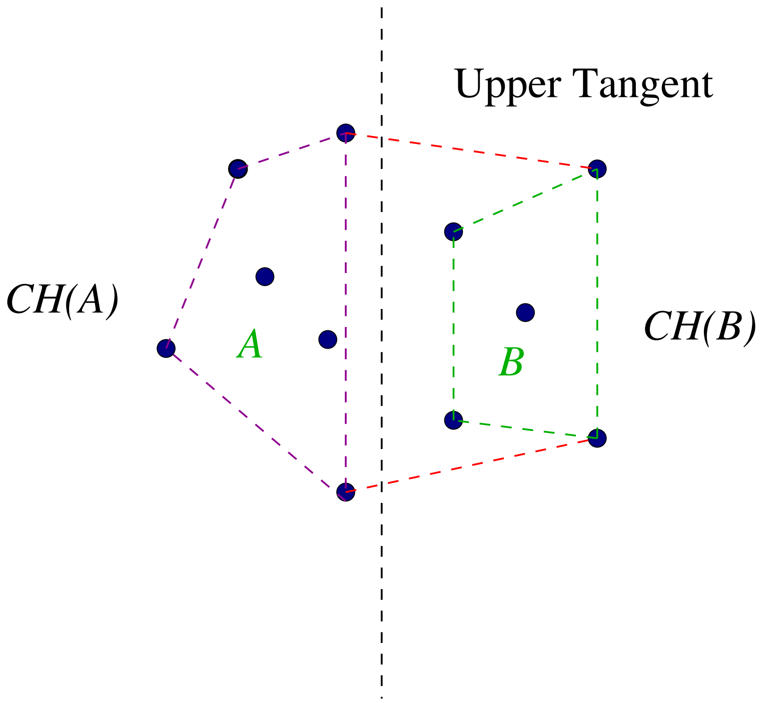
Merging Convex Hulls



Lower Tangent

- $a =$ **rightmost point of** $CH(A)$.
- $b =$ **leftmost point of** $CH(B)$.
- **while** ab **not lower tangent of** $CH(A)$ **and** $CH(B)$ **do**
  1. **while** ab **not lower tangent to** $CH(A)$ **set** $a = a - 1$ **(move** a **CW);**
  2. **while** ab **not lower tangent to** $CH(B)$ **set** $b = b + 1$ **(move** b **CCW);**
- **Return** ab.

Analysis of Divide and Conquer



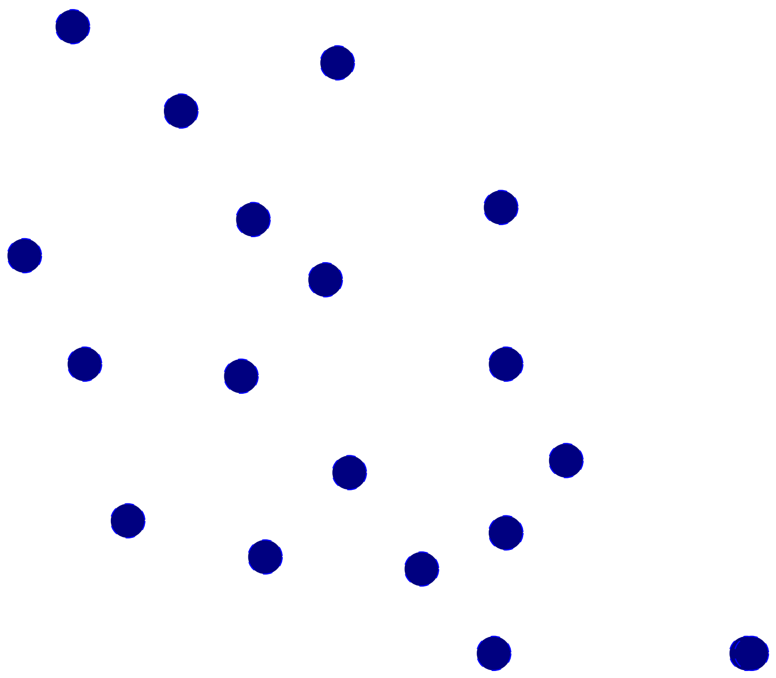
- **Initial sorting takes** $O(N \log N)$ **time.**
- **Recurrence for divide and conquer** $T(N) = 2T(N/2) + O(N)$
- $O(N)$ **for merging–computing tangents.**
- **Recurrence solves to** $T(N) = O(N \log N)$**.**
- $\Omega(N \log N)$ **also a lower bound.**

Convex Hulls in 3D and Beyond

- **The convex hull of N points in $3D$ is a planar graph.**
- **By Euler's Formula, this face lattice has $O(N)$ faces and edges.**
- **An $O(N \log N)$ time divide and conquer algorithms also works in 3D.**
- **In higher dimensions, complexity of convex hull face lattice grows.**
- **In d dimensions, Upper Bound Theorem of McMullen-Shepard gives $\binom{N}{k}$ faces of dimension $(k-1)$, where $0 \leq k \leq d/2$.**

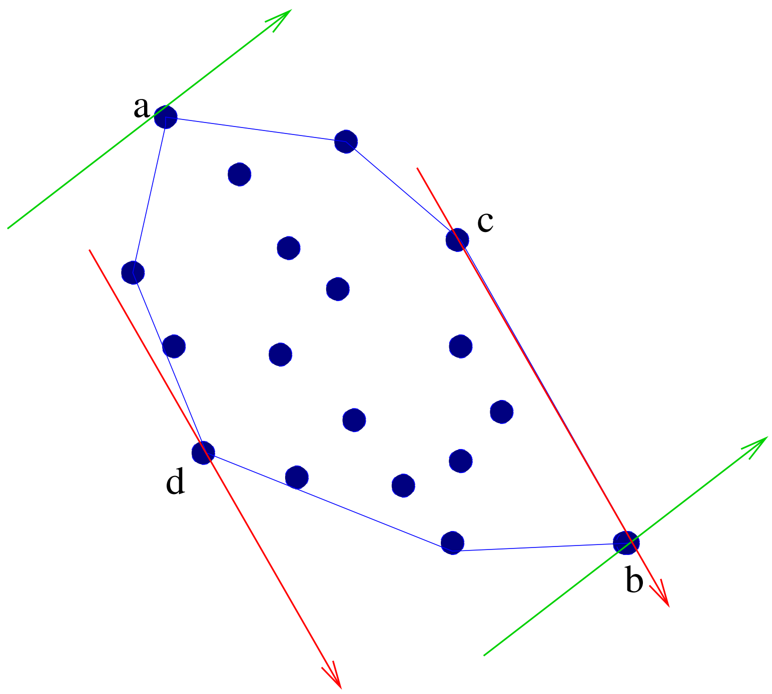
Applications of CH

A problem in statistics



- **Given a set of N data points in R^2, fit a line that minimizes the maximum error.**
- **A data point's error is its L_2 norm distance to the line.**

Line Fitting

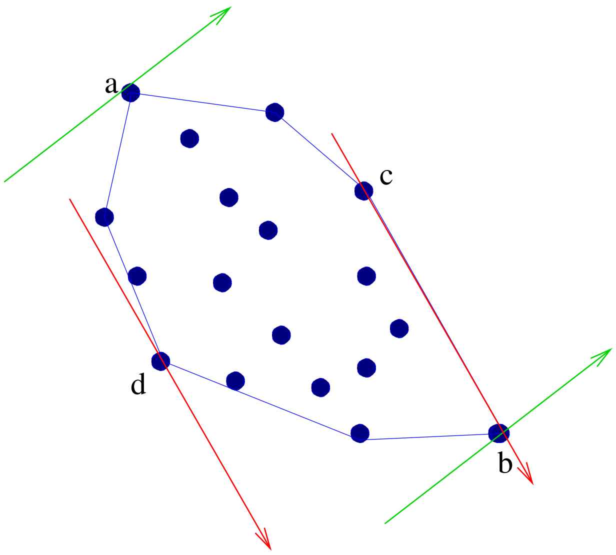


- **Minimizing max error = parallel lines of support with Min separation.**

- **Max error D implies parallel lines of support with separation $2D$, and vice versa.**

- **Min separation between parallel support lines is also called width of S.**

Algorithm for Width



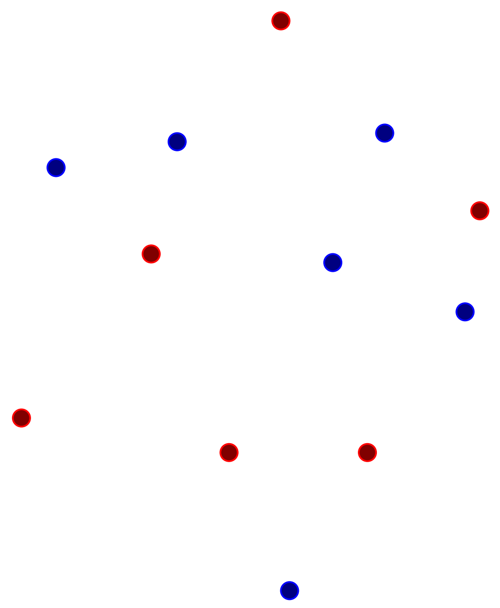
- **Call a, b antipodal pair if they admit parallel lines of support.**

- **In R^2, only $O(N)$ antipodal pairs.**

- **If L_1, L_2 are parallel support lines, with minimum separation, then at least one of the lines contains an edge of $CH(S)$.**

- **We can enumerate all antipodal pairs by a linear march around CH.**

Noncrossing Matching

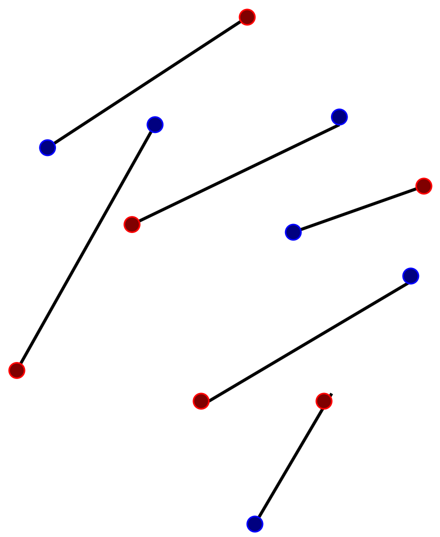


- **Given N red and N blue points in the plane (no three collinear), compute a red-blue non-crossing matching.**

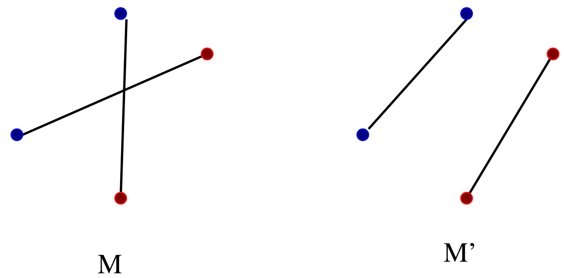
- **Does such a matching always exist?**

- **Find if one exists.**

Noncrossing Matching

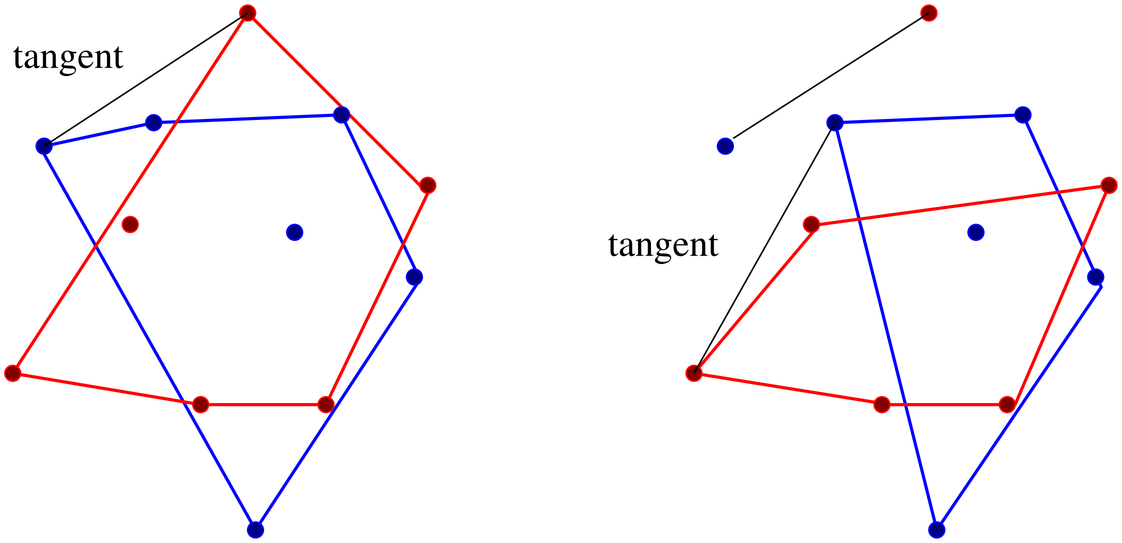


- **A non-crossing matching always exists.**
- **(Non-constructive Proof:) Matching of minimum total length must be non-crossing.**



- **But how about an algorithm?**

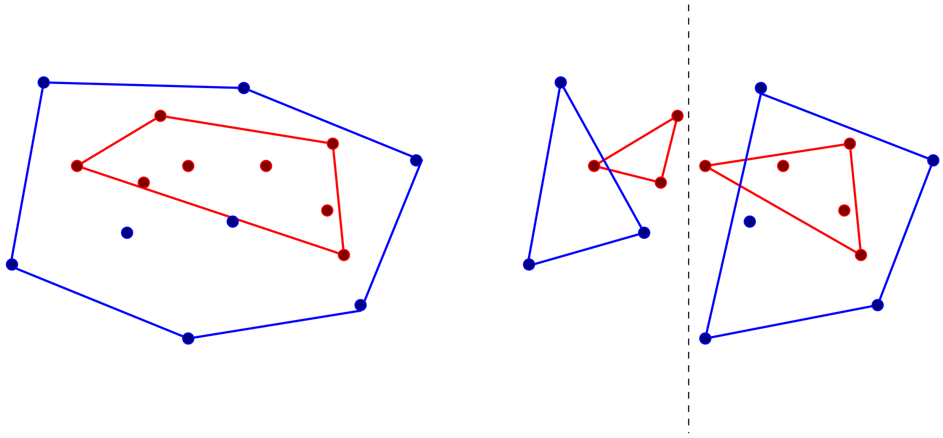
Algorithm



- **Compute $CH(R)$ and $CH(B)$.**
- **Compute a common tangent, say, rb.**
- **Output rb as a matching edge; remove r, b, update convex hulls and iterate.**

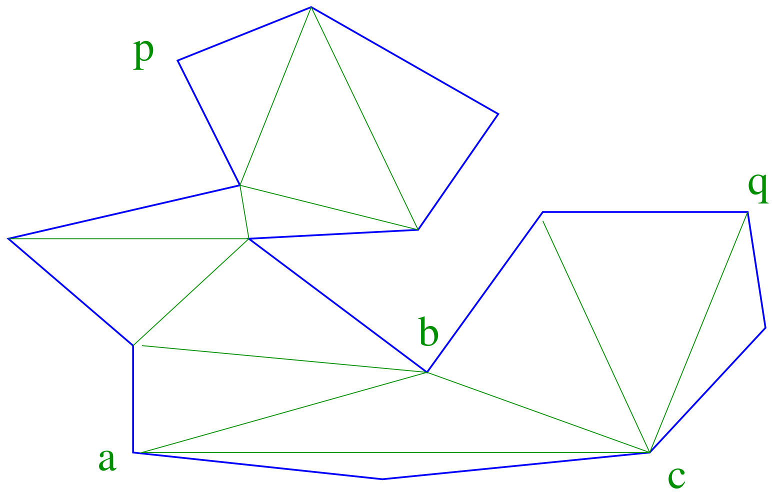
When CH Nest?

- **Algorithm fails if $CH(R)$ and $CH(B)$ nest.**



- **Split by a vertical line, creating two smaller, hull-intersecting problems.**
- **[Hershberger-Suri '92] gives optimal $O(N \log N)$ solution.**

Triangulation



- **Given a polygon, decompose the interior into triangles.**

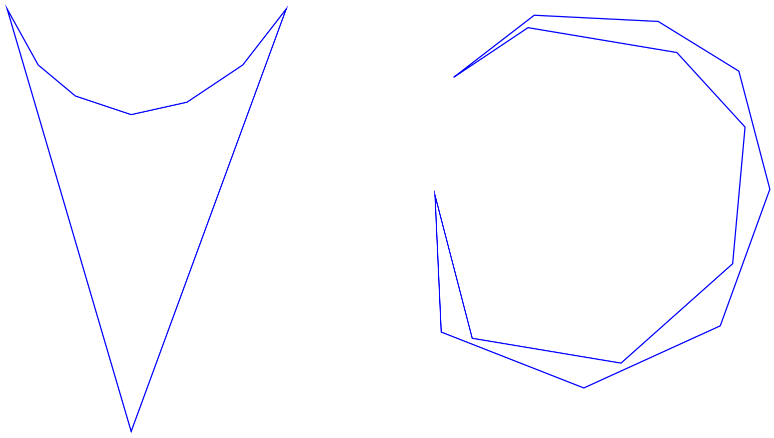
- **Only use internal diagonals, and introduce no new vertices.**

- **Is it always possible?**

- **Does the number of triangles depend on the triangulation method?**

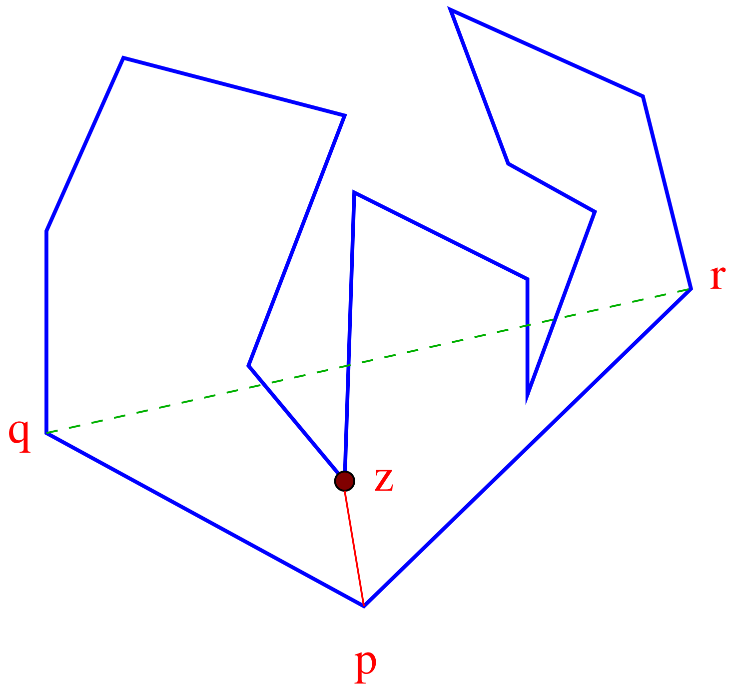
Triangulating Polygons

- **Every simple polygon admits a triangulation.**
- **Any triangulation of N-vertex polygon has $N-2$ triangles.**
- **Figure shows some weird polygons.**



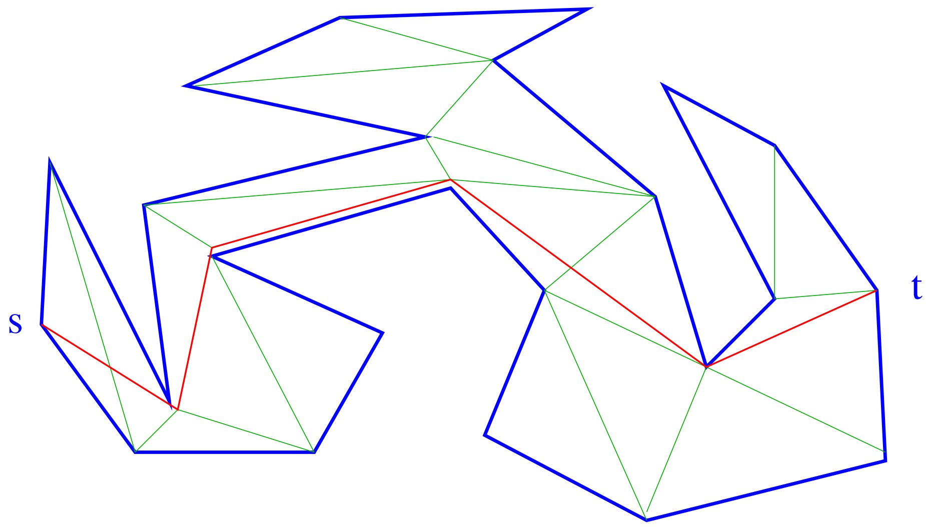
Triangulating Polygons

- **Inductive Proof.**
- **Enough to show one diagonal exists.**



- **Induction: diagonal divides** P **into** P_1**,** P_2**.**
- $t(P) = t(P_1) + t(P_2) = n_1 - 2 + n_2 - 2$.
- $n_1 + n_2 = n + 2$, **so** $t(P) = n - 2$.

Applications: Shortest Paths

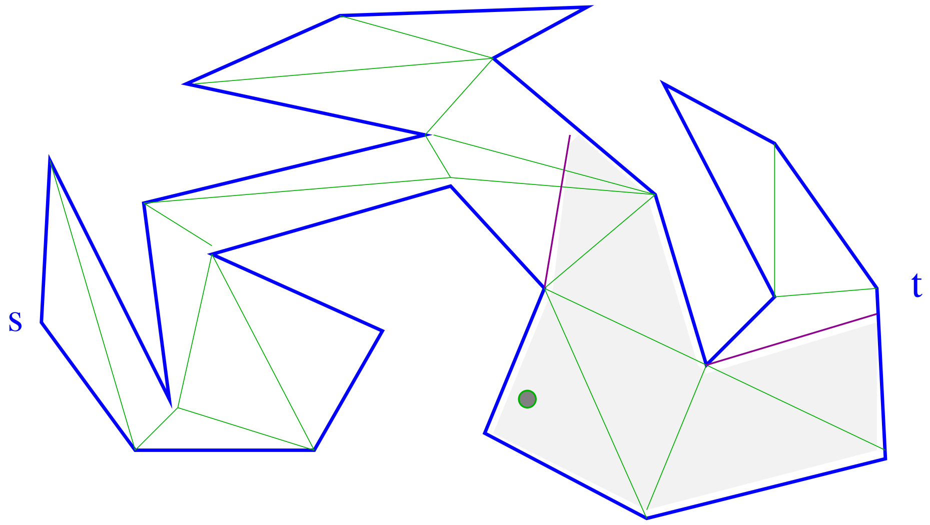


- **Given two points s, t, find shortest path between them.**

- **Use triangulation to advance shortest path, one triangle at a time.**

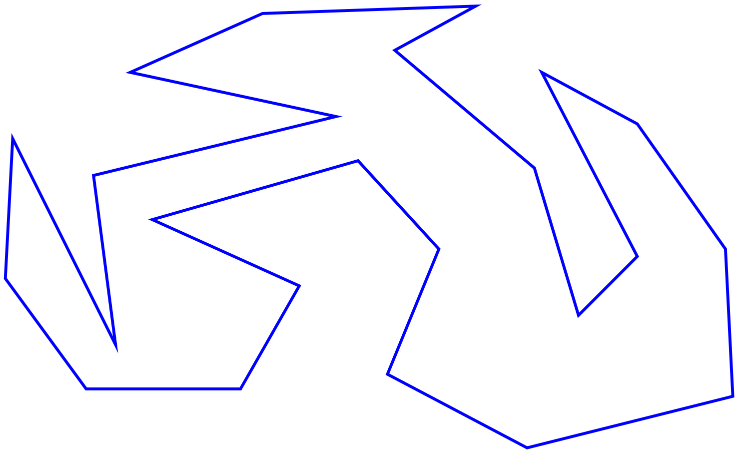
- **Java demo at my web site.**

Applications: Visibility



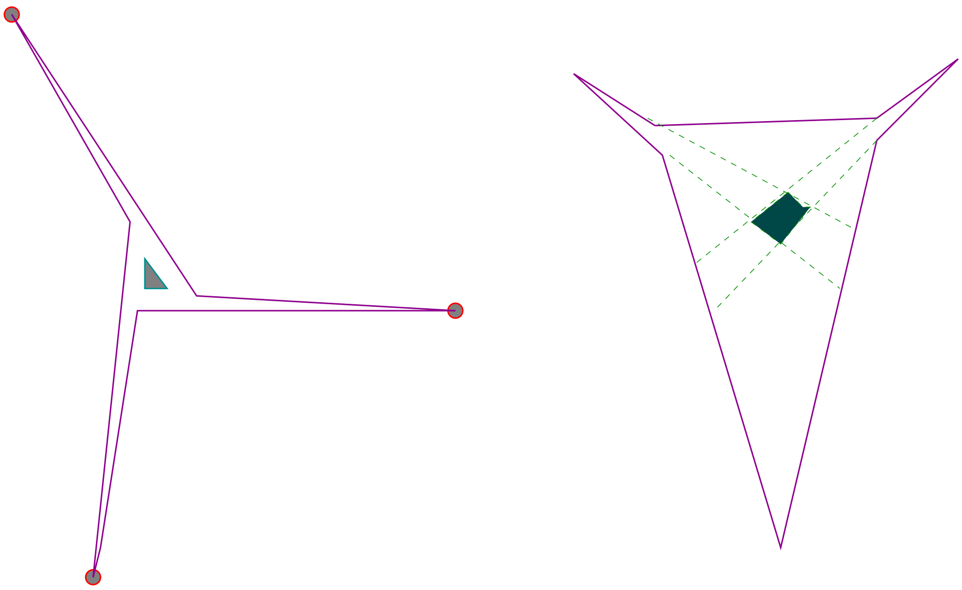
- **Given a point s, determine the region visible to s.**
- **Step through triangulation advancing visibility front.**

Applications: Art Gallery Theorem



- **Think of P as the floorplan of an art gallery.**
- **How many stationary guards necessary and sufficient to guard P?**
- **Determine the best possible function $G(n)$—any polygon with n vertices guardable by n stationary guards.**

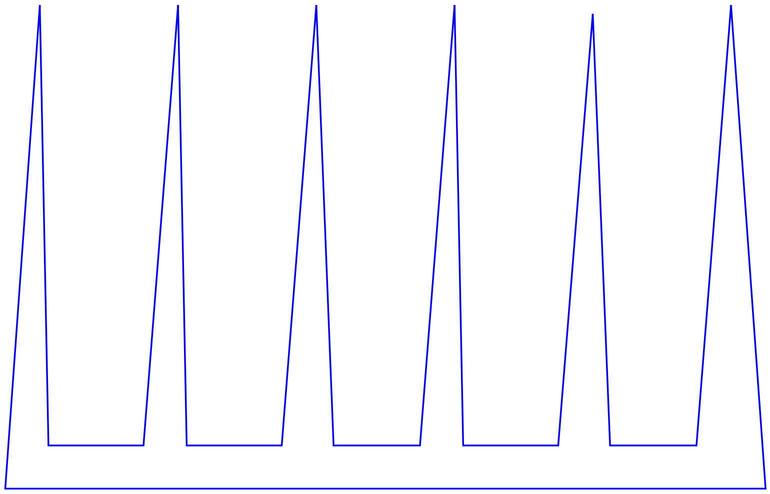
Art Gallery Theorem



- **Fig. (1): Being able to see boundary $\neq$ seeing interior.**
- **Even putting guards at every other vertex is not enough.**
- **Fig. (2): Putting guards at vertices is more restrictive.**

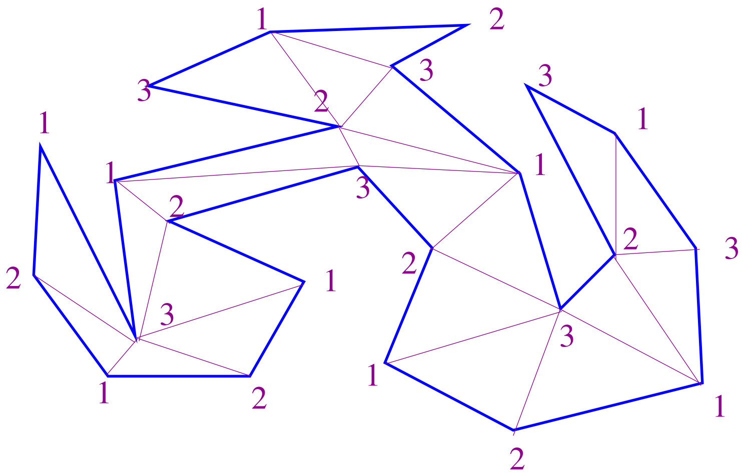
The Theorem

- $\lfloor n/3 \rfloor$ **Vertex guards both necessary and sufficient.**



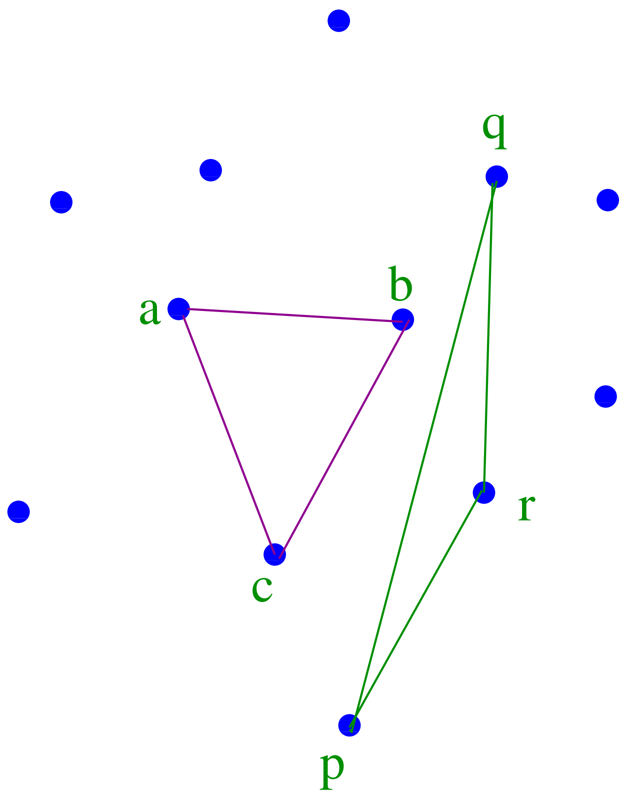
- **Figure above shows the necessity.**

Sufficiency



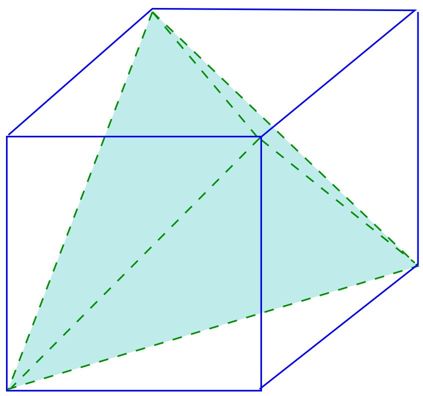
- **Triangulate P. 3-color it.**
- **The least frequent color does not appear more than $\lfloor n/3 \rfloor$ times.**
- **Place guards at this color positions—each triangle has all 3 colors, so seen by a guard.**
- **In example: Colors 1, 2, 3 appear 9, 8 and 7 times, resp.**
- **So, color 3 works.**

Special Triangulations

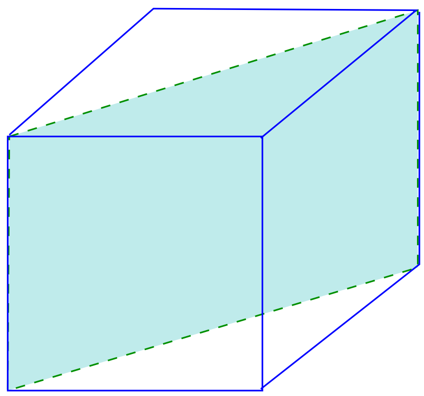


- [Delaunay:] Triangulations that maximize angles.
- [Constrained:] Some edges enforced.
- [Min Length:] Minimize total length.

Triangulation in 3D



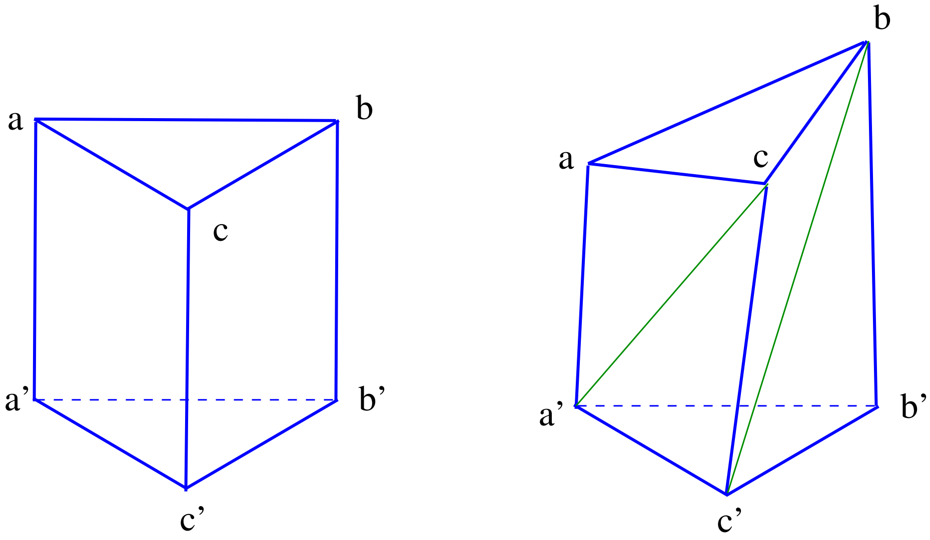
5 Tetrahedra



6 Tetrahedra

- **Different triangulations can have different number of tetrahedra (3D triangles).**

Untriangulable Polyhedron



- **Smallest example of a polyhedron that cannot be triangulated without adding new vertices. (Schoenhardt [1928]).**

- **It is NP-Complete to determine if a polyhedron requires Steiner vertices for triangulation.**

- **Every 3D polyhedron with N vertices can be triangulated with $O(N^2)$ tetrahedra.**

Research Directions

1. **[3D Triangulation:] Algorithm for triangulating 3D polyhedron in near-optimal number of tetrahedra.**

2. **[Shape Sensitive Analysis:] Worst-case forcing examples are often contrived. Algorithms and analysis for non-pathological input.**
   Example: Bounding box heuristic in collision detection *[Suri-Zhou].*

3. **[Kinetic Data Structures:] Develop data structures for objects in continuous motion to maintain attributes that change discretely.**