Plane Geometry Algorithms
Geometry Algorithms

Show up every now and then

• 0-1 questions per contest

Almost always 2D (plane) geometry
Geometry Algorithms

Show up every now and then
• 0-1 questions per contest

Almost always 2D (plane) geometry
• 3D algorithms “too hard”/specialized
• test cases easier to create
• diagrams easier to draw…
Geometry Algorithms

Key challenges:

1. Remembering (or deriving) formulas
Blast Zone

Given:

- list of points \((x_i, y_i)\)
- epicenter \((cx, cy)\)
- radius \(r\)

Count number of points distance \(<= r\) from epicenter

All inputs are ints between \(-2^{30}\) and \(2^{30}\)
Vectors and Distance

\[(x_2, y_2)\]

\[(x_1, y_1)\]
Vectors and Distance

(vector from $(x_1, y_1)$ to $(x_2, y_2)$:)

$$(x_2 - x_1, y_2 - y_1)$$
Vectors and Distance

The vector from \((x_1, y_1)\) to \((x_2, y_2)\):

\[(x_2 - x_1, y_2 - y_1)\]

Length of vectors:

\[\|(a, b)\| = \sqrt{a^2 + b^2}\]
Vectors and Distance

Let $(x_1, y_1)$ and $(x_2, y_2)$ be two points in a plane.

- **Vector from $(x_1, y_1)$ to $(x_2, y_2)$:**
  \[(x_2 - x_1, y_2 - y_1)\]

- **Length of vectors:**
  \[\| (a, b) \| = \sqrt{a^2 + b^2} \]

- **Distance:**
  \[d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \]
Circle-Point Queries

\[(cx, cy)\]
Circle-Point Queries

\[(cx, cy)\]

point inside circle if

\[\|(x, y) - (cx, cy)\| \leq r\]

\[\sqrt{(x - cx)^2 + (y - cy)^2} \leq r\]
Blast Zone?

Given:

- list of points \((x_i, y_i)\)
- epicenter \((cx, cy)\)
- radius \(r\)

All inputs are ints between \(-2^{30}\) and \(2^{30}\)

O(points) solution: loop over all points, and check if

\[
\sqrt{(x_i - cx)^2 + (y_i - cy)^2} \leq r
\]
Geometry Algorithms

Key challenges:

1. Remembering (or deriving) formulas
2. Dealing with precision issues
Two Kinds of Geometry Problems

<table>
<thead>
<tr>
<th>Exact Arithmetic</th>
<th>Floating Point</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Inputs are doubles</td>
</tr>
<tr>
<td></td>
<td>• Problem statement says “answer is accepted if it is within [tolerance]”</td>
</tr>
<tr>
<td></td>
<td>• Problem statement says “answer will not change if inputs are slightly perturbed”</td>
</tr>
</tbody>
</table>
### Two Kinds of Geometry Problems

<table>
<thead>
<tr>
<th><strong>Exact Arithmetic</strong></th>
<th><strong>Floating Point</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Pretty much <strong>any</strong> other situation</td>
<td>Inputs are doubles</td>
</tr>
<tr>
<td>Problem statements with “round the answer to <strong>n</strong> digits” are <strong>not</strong> safe for floating point!</td>
<td>Problem statement says “answer is accepted if it is within [tolerance]”</td>
</tr>
<tr>
<td></td>
<td>Problem statement says “answer will not change if inputs are slightly perturbed”</td>
</tr>
</tbody>
</table>
Two Kinds of Geometry Problems

Exact Arithmetic

• Pretty much any other situation

• Problem statements with “round the answer to n digits” are not safe for floating point!

Floating Point

• Inputs are doubles

• Problem statement says “answer is accepted if it is within [tolerance]”

• Problem statement says “answer will not change if inputs are slightly perturbed”

Incorrectly classifying the type of problem is a fatal noob trap!!
Circle-Point Queries

Point inside circle if

\[ \|(x, y) - (cx, cy)\| \leq r \]

\[ \sqrt{(x - cx)^2 + (y - cy)^2} \leq r \]

\[ (x - cx)^2 + (y - cy)^2 \leq r^2 \]
Blast Zone? Take II

Given:
- list of points \((x_i, y_i)\)
- epicenter \((cx, cy)\)
- radius \(r\)

All inputs are ints between \(-2^{30}\) and \(2^{30}\)

O(points) solution: loop over all points, and check if

\[
(x_i - cx)^2 + (y_i - cy)^2 \leq r^2
\]
Geometry Algorithms

Key challenges:

1. Remembering (or deriving) formulas
2. Dealing with precision issues
3. Dealing with potential overflow
Line Classification

Given two lines, encoded as two pairs of endpoints \((x_s^1, y_s^1), (x_e^1, y_e^1); (x_s^2, y_s^2), (x_e^2, y_e^2)\)

Determine if the lines are parallel.

All inputs are ints between \(-2^{30}\) and \(2^{30}\)
Triangle Area

Signed area:

\[
\frac{1}{2} \det \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{bmatrix}
\]

\[
= \frac{(x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0)}{2}
\]
Triangle Area

Signed area:

\[ \frac{1}{2} \det \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{bmatrix} \]

Why signed?
Triangle Area

Signed area:

\[
\frac{1}{2} \text{det} \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{bmatrix}
\]

Why signed?
Triangle Area

Signed area:

$$\frac{1}{2} \det \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{bmatrix}$$

Why signed?
Polygon Area

Given a list of integer points in the plane, calculate area of polygon

All input ints between $-2^{15}$ and $2^{15}$
Area is an integer
Polygon Area

The hard way: divide and conquer “ear cutting”
Polygon Area

The easy way:
Polygon Area

The easy way:

\[(0, 0) \rightarrow (x_0, y_0) \rightarrow (x_1, y_1)\]
Polygon Area

The easy way:
Polygon Area

The easy way:

(0, 0)
Polygon Area

The easy way:
Polygon Area

The easy way:
Polygon Area

The easy way: “shoelace formula”

\[
\frac{1}{2} \sum [(x_i - 0)(y_{i+1} - 0) - (x_{i+1} - 0)(y_i - 0)]
\]
Convex Hull

“Envelope” of set of points

- rubber band analogy
Convex Hull

“Envelope” of set of points
- rubber band analogy
Computing Convex Hull

1. Find one point on boundary

2. Until we have complete polygon, walk counterclockwise around boundary
Computing Convex Hull

1. Find one point on boundary
Computing Convex Hull

1. Find one point on boundary
   • e.g. point with min x coord
Computing Convex Hull

How to find next boundary point?

can it be this one?
Computing Convex Hull

How to find next boundary point?

can it be this one?
Computing Convex Hull

How to find next boundary point?

sweep through pts; if we find better one, switch to it
Computing Convex Hull

How to find next boundary point?

sweep through pts;
if we find better one,
switch to it
Computing Convex Hull

How to find next boundary point?

sweep through pts;
if we find better one, switch to it
Jarvis Marching

Pros:
• Intuitive
• Easy to code

Cons:
• Doesn’t work in 3D+
• $O(n^2)$
Graham Scan

Idea: sort points by angle
Graham Scan

Idea: sort points by angle
- Start adding edges
Graham Scan

Idea: sort points by angle

- Start adding edges
Graham Scan

Idea: sort points by angle
- Start adding edges
Graham Scan

Idea: sort points by angle
- Start adding edges
- Detect “right turns”
Graham Scan

Idea: sort points by angle
- Start adding edges
- Detect “right turns” and delete prev pt
- Check again for a right turn
Graham Scan

Idea: sort points by angle
- Start adding edges
- Detect “right turns” and delete prev pt
- Check again for a right turn

Now $O(n \log n)$
Graham Scan

Pros:
• Fast ($O(n \log n)$)

Cons:
• Still doesn’t work in 3D+
• Numerical robustness issues if points are close to colinear