Computational Geometry

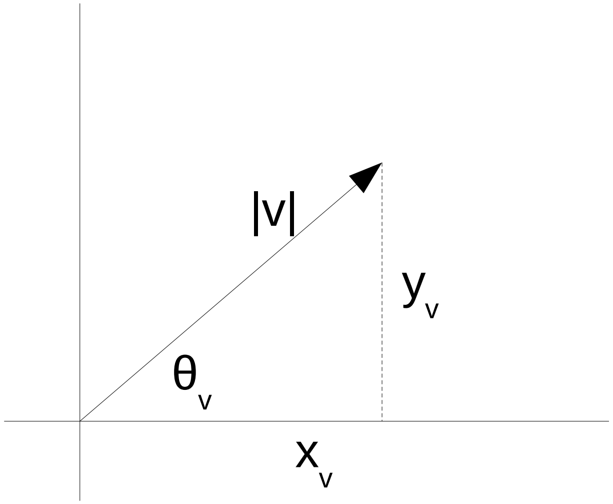
Ulrik de Muelenaere

IOI Camp 2
April 2015

Vectors

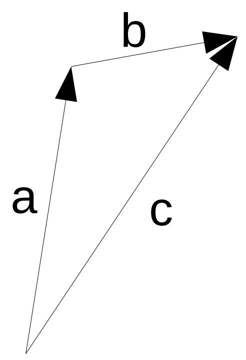
- Will be working in 2D
- Vector **v** has magnitude |**v**| and direction $\boldsymbol{\theta}_\mathbf{v}$
- Components: $x_v = |v| \cos\theta_v,\ y_v = |v| \sin\theta_v$
- Typically stored as ($\mathbf{x_v}$, $\mathbf{y_v}$)
- Calculate magnitude:

$$|v| = \sqrt{x_v^2 + y_v^2}$$



Vectors

$$c = a + b$$



- Place start of **b** at end of **a**
- Connect start of **a** to end of **b**
- Also works with components:

$$a + b = (x_a + x_b, y_a + y_b)$$

$$a - b = (x_a - x_b, y_a - y_b)$$

- With points **A** and **B**, **B** – **A** is vector from **A** to **B**

Vectors

- Dot product: $a \cdot b = |a||b| \cos\theta = x_a x_b + y_a y_b$
- Dot product is scalar, not vector
- Cross product: $a \times b = |a||b| \sin\theta = x_a y_b - x_b y_a$
- 3D vector in **z** direction, so use as scalar
- Find properties of angle **θ** between vectors:

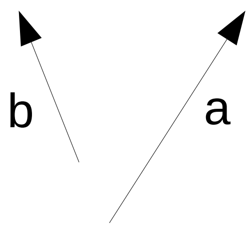
$$a \cdot b = |a||b|, a \times b = 0 \qquad a \cdot b = 0, a \times b = \pm |a||b|$$

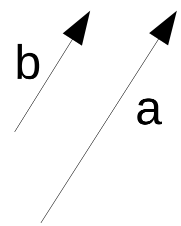
Vectors

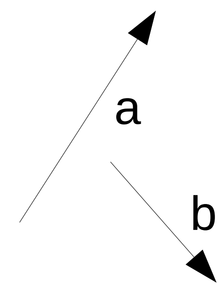
- $\boldsymbol{\theta}$ is angle from **a** to **b**

$$a\times b=|a||b|\sin(\theta)=-|b||a|\sin(-\theta)=-b\times a$$

- Find direction of $\boldsymbol{\theta}$ by looking at sign of **a** x **b**

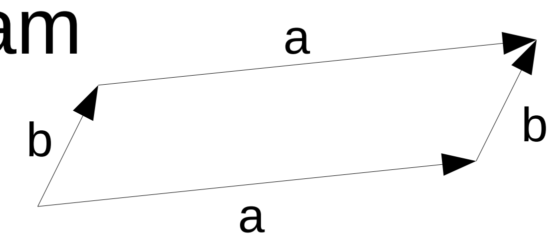






$$a\times b>0 \qquad a\times b=0 \qquad a\times b<0$$

- Cross product is area of parallelogram



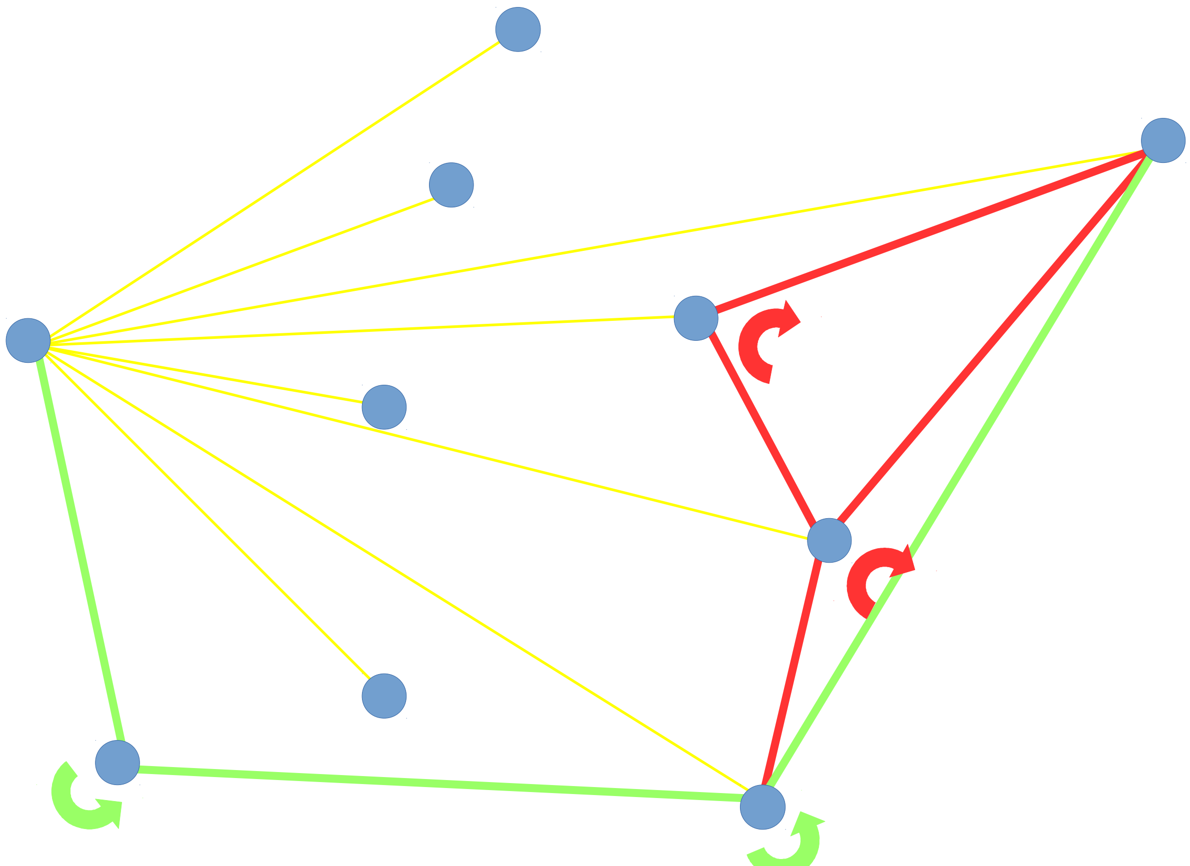
Convex hull

- Smallest convex polygon enclosing set of points
- Graham scan runs in O(**n** log **n**) or O(**n**) if already sorted
- Pick extreme, e.g. leftmost, point $\mathbf{p}_0$
- Sort other points based on angle of line from $\mathbf{p}_0$ to point
- Add each point then check if previous must be removed

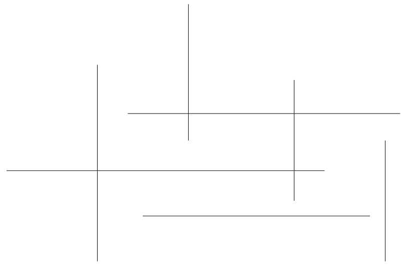
Convex hull

- Use integers instead of floating point
- If $\mathbf{p_0}$ is leftmost point, can sort by slope instead of angle
- Slope of line from $\mathbf{p_0}$ to $\mathbf{p_i}$ is: $\frac{y_i - y_0}{x_i - x_0} = \frac{n_i}{d_i}, d_i > 0$
- Compare slopes: $\frac{n_i}{d_i} < \frac{n_j}{d_j}$
- Since $\mathbf{d_i}$, $\mathbf{d_j}$ > 0, equivalent to: $n_i d_j < n_j d_i$

Convex hull



Line sweep



- Find all intersections of line segments
- Compare each line to every other – O(**n**^{2})
- Sweep vertical line from left to right, looking for intersections – O(**n** log **n**)
- Define event as x-value where something happens
- Move line from one event to next

Line sweep

- Keep track of horizontal segments intersecting sweep line in set ordered by y-value
- 3 types of events:
  - Start of horizontal segment – add to set
  - End of horizontal segment – remove from set
  - Vertical segment – find intersecting segments in set

Lines

- Various forms of equations:

$$a x + b y = c \qquad y = m x + k \qquad y = -\frac{a}{b} x + \frac{c}{b}$$

- To find equation given 2 points, solve:

$$y_1 = m x_1 + k\,,\; y_2 = m x_2 + k$$

- To find intersection of lines (if $\mathbf{m_1} \neq \mathbf{m_2}$), solve:

$$y = m_1 x + k_{1,}\; y = m_2 x + k_2$$

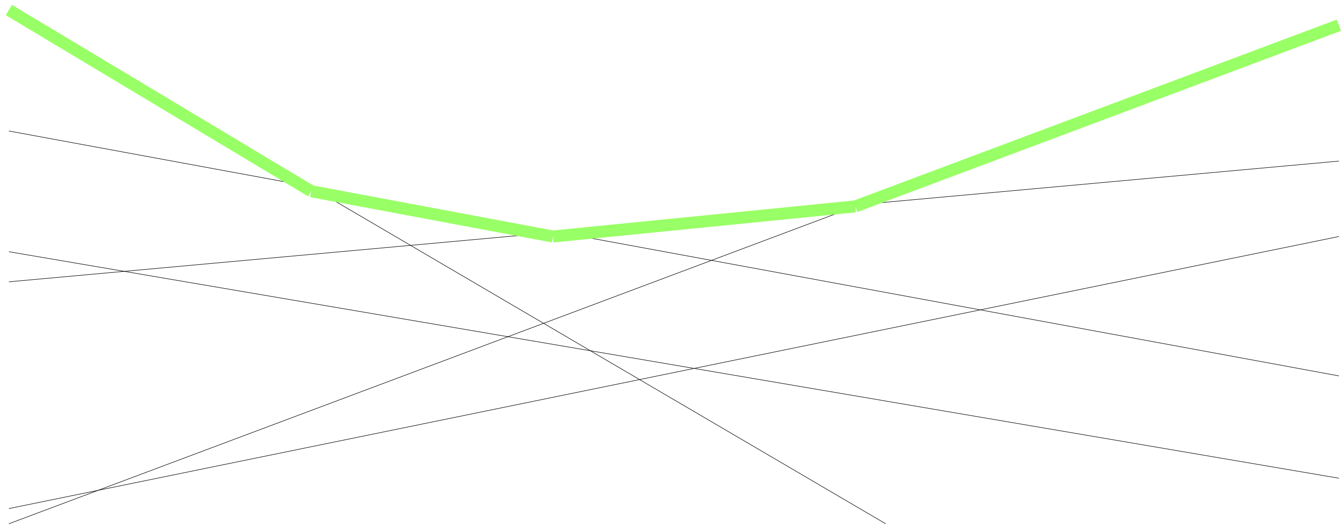
$$x = \frac{k_1 - k_2}{m_2 - m_1} = \frac{k_2 - k_1}{m_1 - m_2}$$

Convex hull trick

- Optimisation for dynamic programming algorithms
- Consider set of lines $y = m_i x + k_i$
- Queries to find maximum (or minimum) **y** of any line at specific **x**
- Could try all lines – O(**n**)
- Trick allows adding line in O(1) or O(log **n**) and queries take O(log **n**)

Convex hull trick

- Only keep track of lines that are maximum over some interval
- Order lines by this interval
- Query performs binary search



Convex hull trick

- Note that slope increases with increasing **x**
- If each line has greater slope than previous lines, can add to end (similar if slope decreasing)
- If intersection with previous line before start of previous line's interval, remove previous line and check new previous line
- If slopes not increasing, add line to set and check if new, previous or next line should be removed

Convex hull trick

- Consider sequence of rectangles in order of increasing height $\mathbf{h_i}$ and decreasing width $\mathbf{w_i}$
- Subsequence [**i**, **j**] has cost $\mathbf{w_i}\ \mathbf{h_j}$
- Partition sequence into subsequences to minimize total cost

Convex hull trick

- Let $\mathbf{c_i}$ be minimum cost of first $\mathbf{i}$ rectangles, with

  $c_0 = 0$

- For $\mathbf{c_i}$, last subsequence must be [$\mathbf{j}$ + 1, $\mathbf{i}$] with

  $0 \le j < i$

- Therefore $c_i = \min_{0 \le j < i} \{c_j + w_{j+1} h_i\}$
- This is the minimum at $\mathbf{h_i}$ of a set of lines with y-intercept $\mathbf{c_j}$ and slope $\mathbf{w_{j+1}}$, which is decreasing