Geometric Algorithms – Beyond Theory

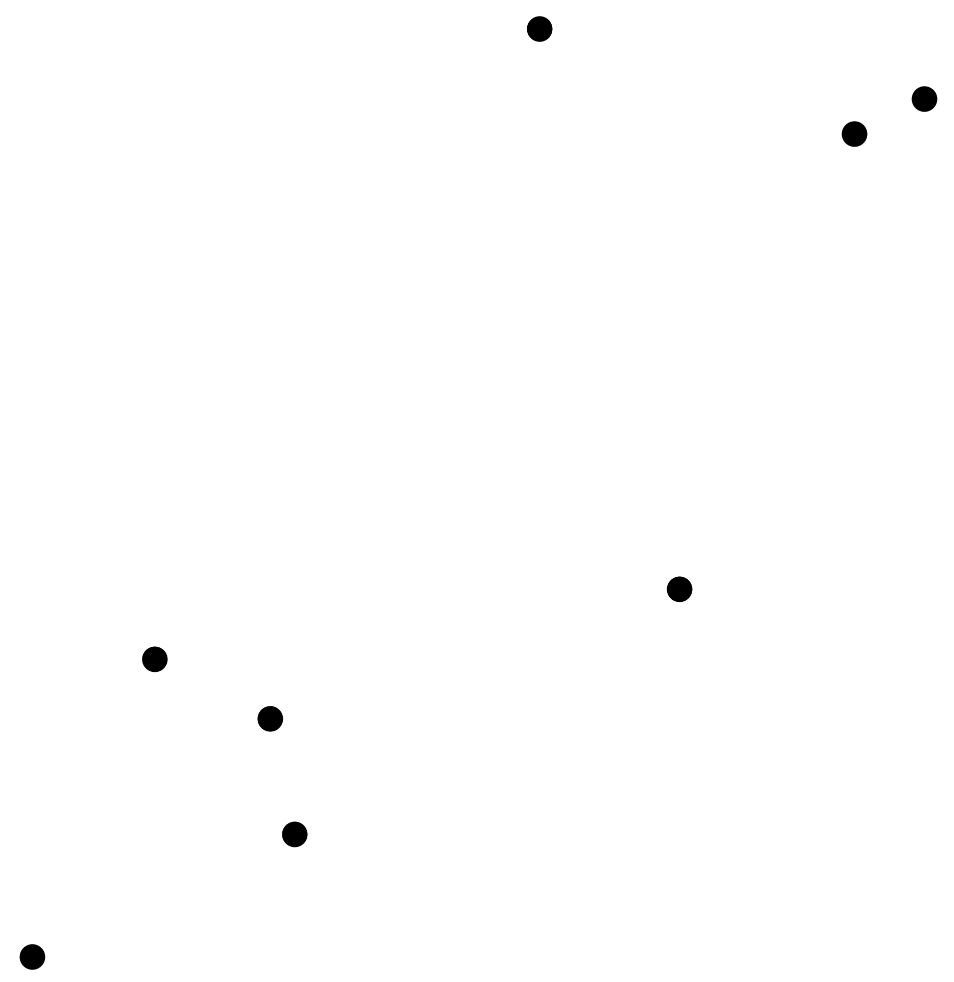
Algorithm Engineering

Robustness of Geometric Algorithms

Partial randomization

Computing convex hulls

We know how to compute convex hulls

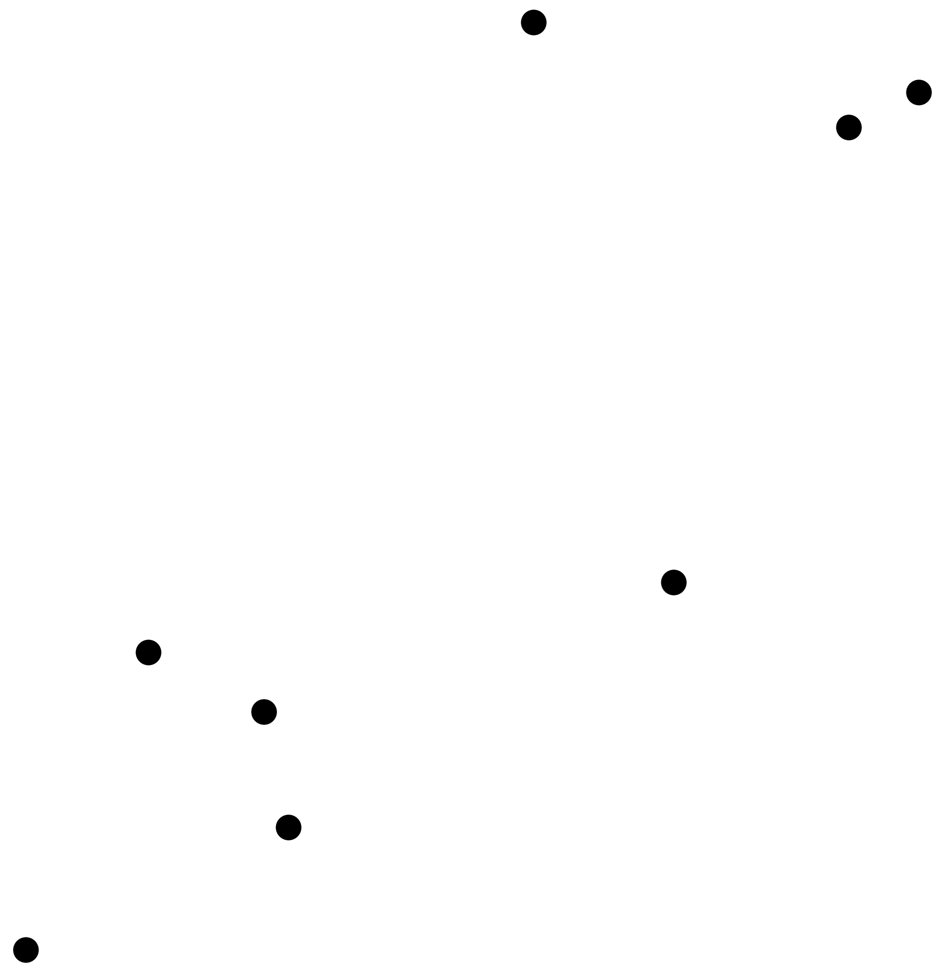


Computing convex hulls

We know how to compute convex hulls

Algorithm INCREMENTALCONVEXHULL(S)

$CH \leftarrow p_1, p_2, p_3$
$S \leftarrow S \backslash \{p_1, p_2, p_3\}$
for all $r \in S$ **do**
    if r is outside CH **then**
        replace CH edges visible from r with two tangent edges $\overline{v_i r}$, $\overline{r v_j}$

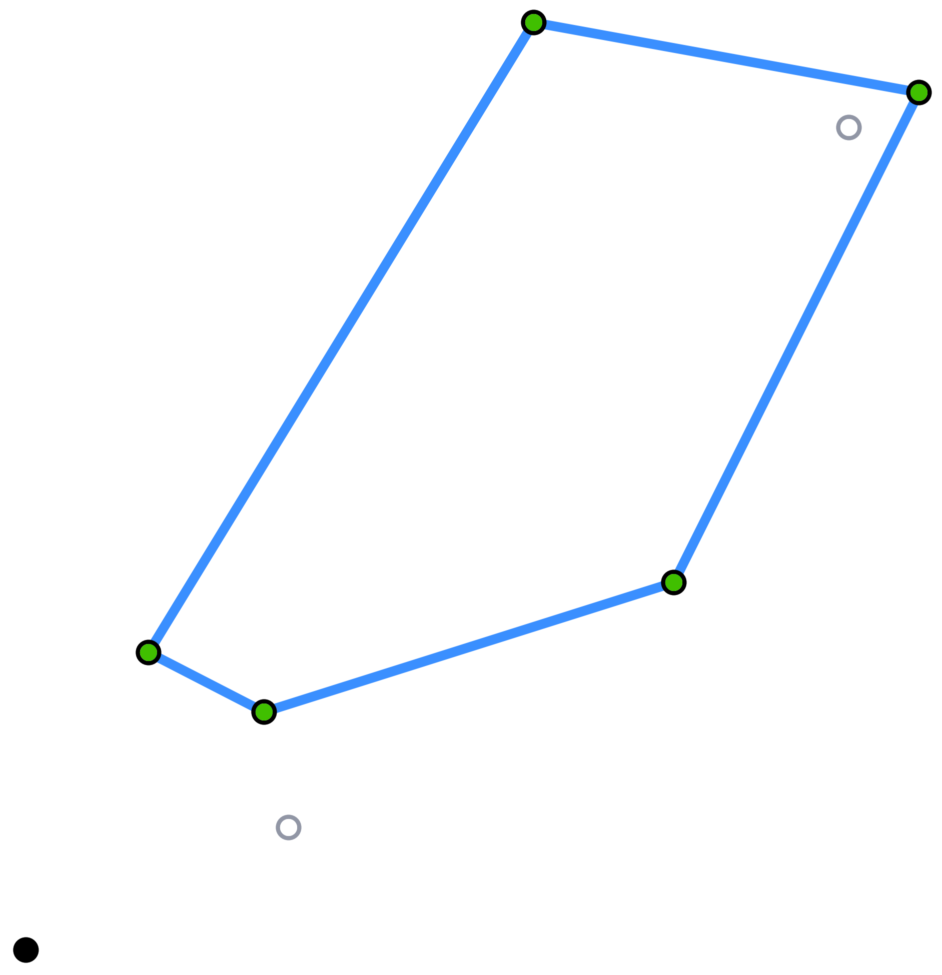


Computing convex hulls

We know how to compute convex hulls

Algorithm INCREMENTALCONVEXHULL(S)

1: $CH \leftarrow p_1, p_2, p_3$
2: $S \leftarrow S \backslash \{p_1, p_2, p_3\}$
3: **for all** $r \in S$ **do**
4: **if** r is outside CH **then**
5: replace CH edges visible from r with two tangent edges $\overline{v_i r}$, $\overline{r v_j}$

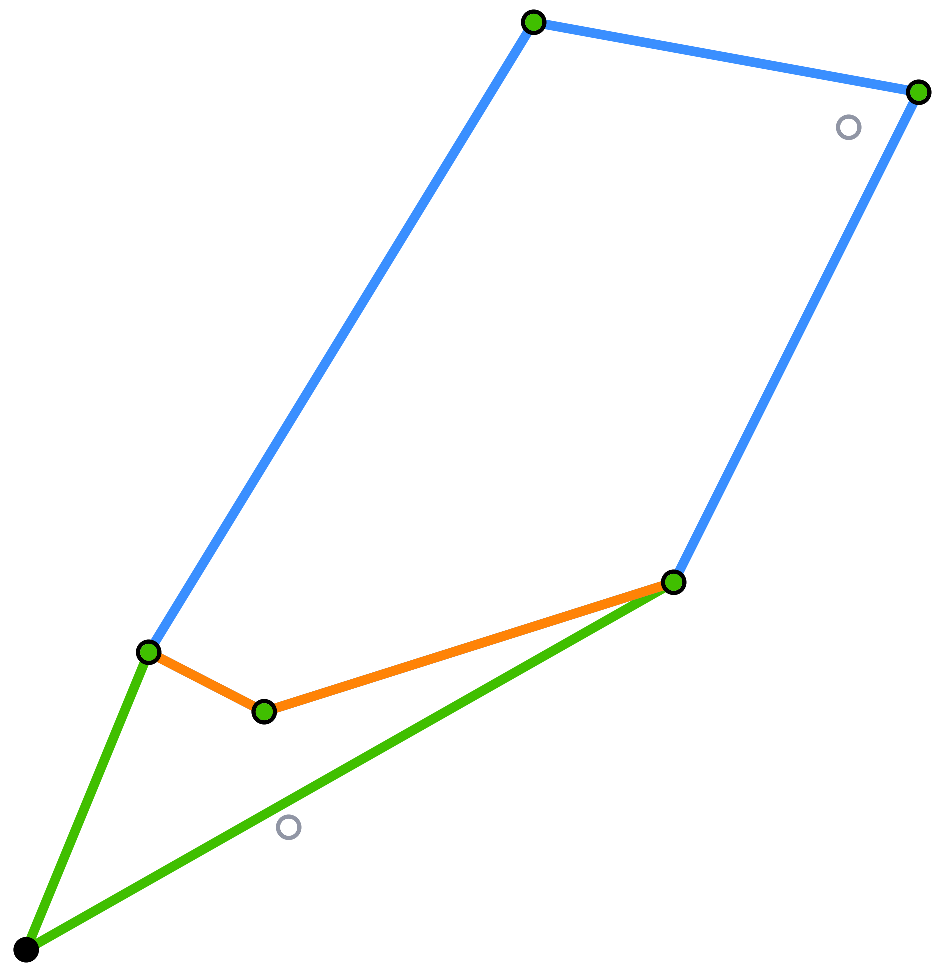


Computing convex hulls

We know how to compute convex hulls

Algorithm INCREMENTALCONVEXHULL(S)

$CH \leftarrow p_1, p_2, p_3$
$S \leftarrow S \backslash \{p_1, p_2, p_3\}$
for all $r \in S$ **do**
  if r is outside CH **then**
    replace CH edges visible from r with two tangent edges $\overline{v_i r}$, $\overline{r v_j}$

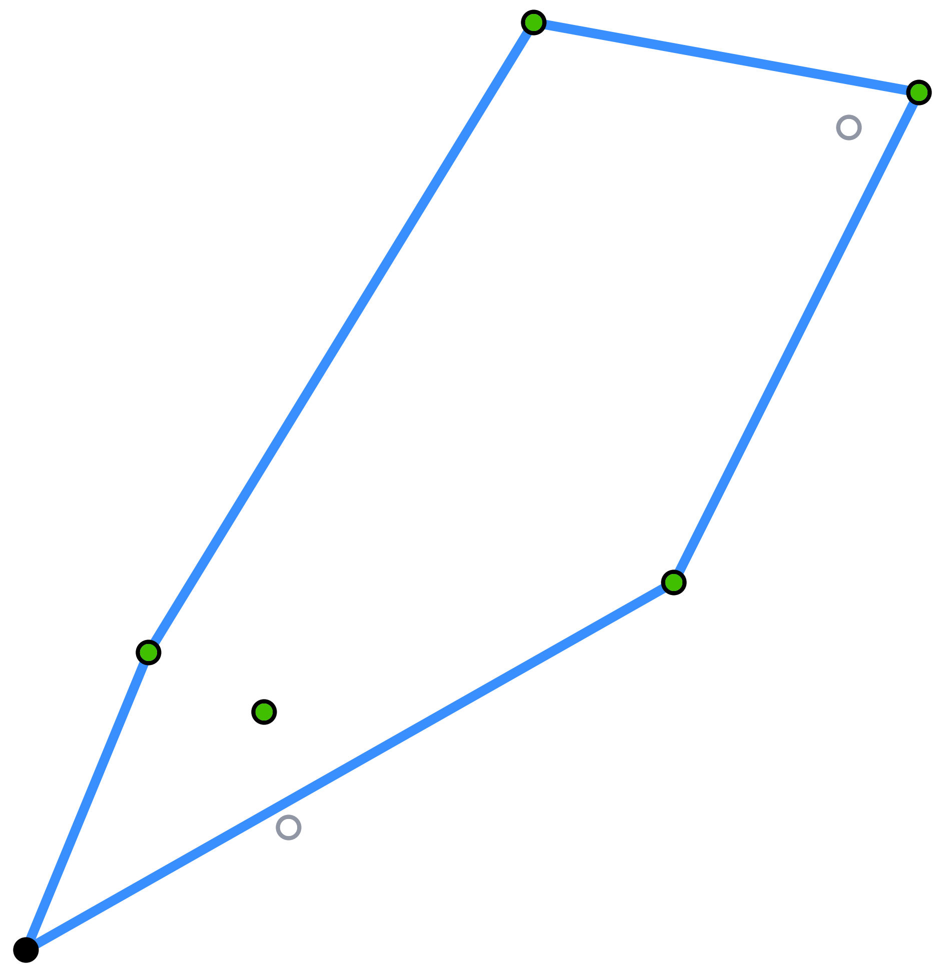


Computing convex hulls

We know how to compute convex hulls

Algorithm INCREMENTALCONVEXHULL(S)

1: $CH \leftarrow p_1, p_2, p_3$
2: $S \leftarrow S \backslash \{p_1, p_2, p_3\}$
3: **for all** $r \in S$ **do**
4: **if** r is outside CH **then**
5: replace CH edges visible from r with two tangent edges $\overline{v_i r}$, $\overline{r v_j}$

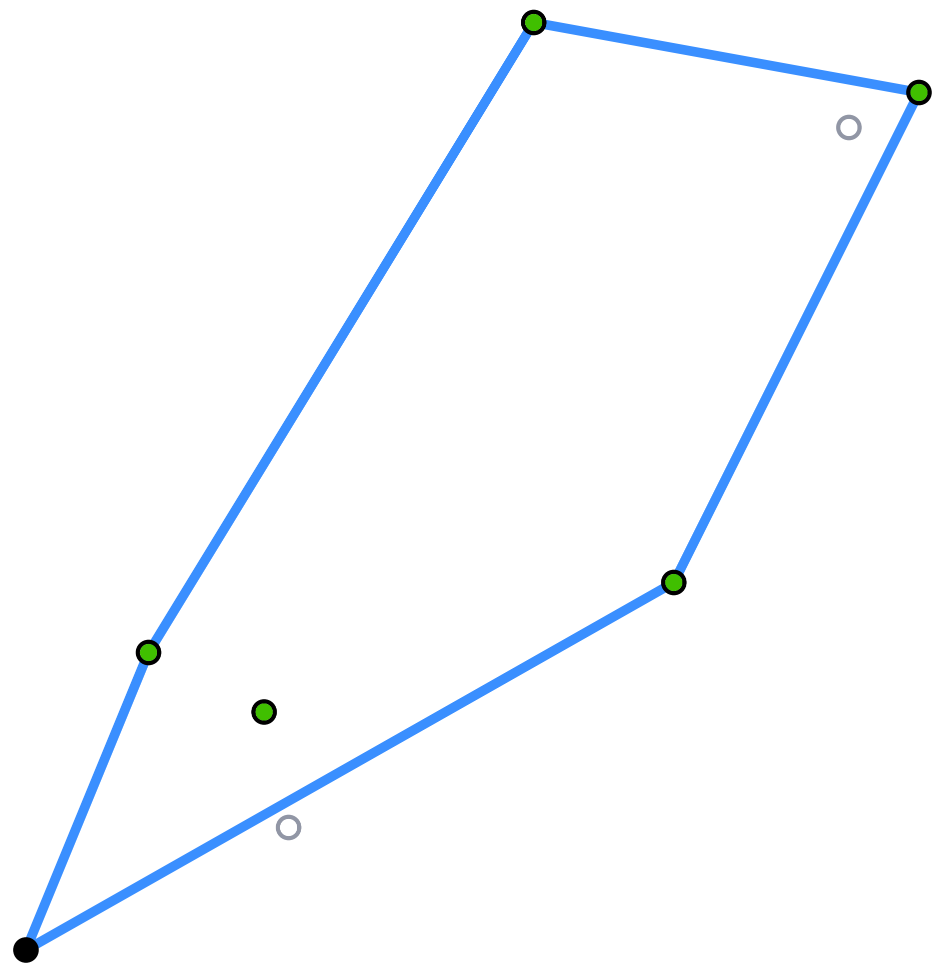


Computing convex hulls

We know how to compute convex hulls

Algorithm INCREMENTALCONVEXHULL(S)

$CH \leftarrow p_1, p_2, p_3$
$S \leftarrow S \backslash \{p_1, p_2, p_3\}$
for all $r \in S$ **do**
    if r is outside CH **then**
        replace CH edges visible from r with two tangent edges $\overline{v_i r}$, $\overline{r v_j}$



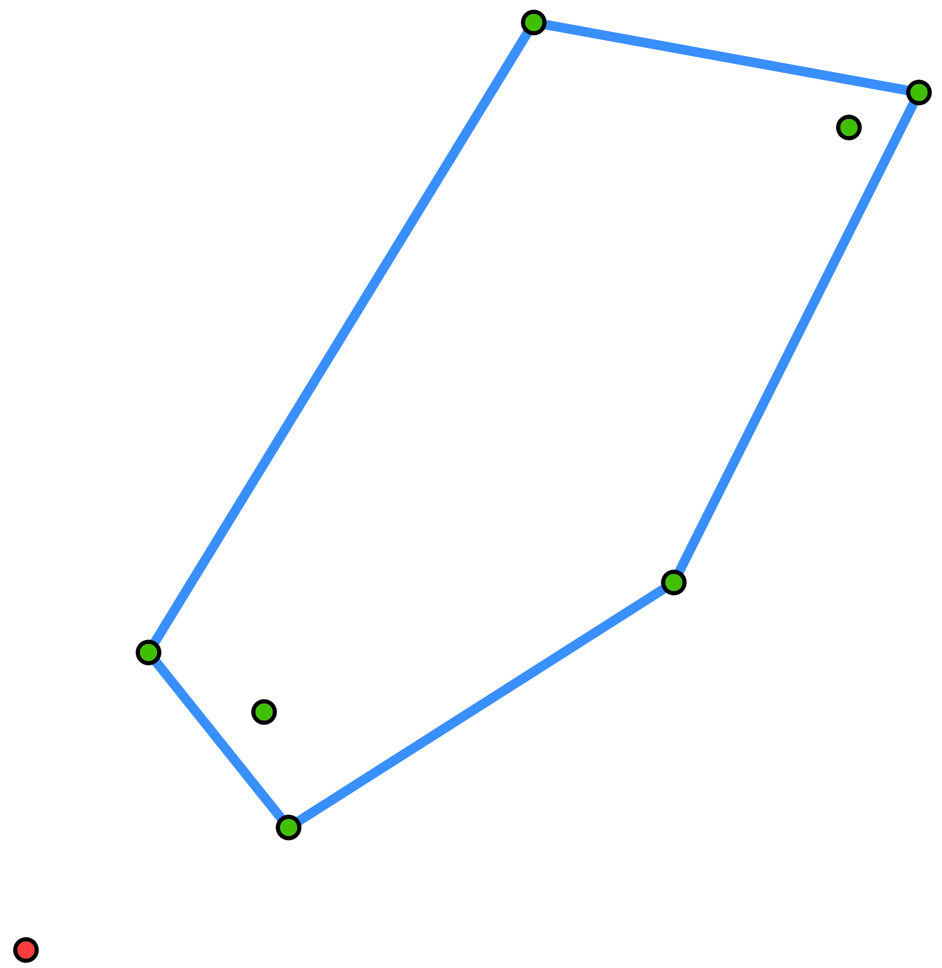
Question: Is this algorithm correct?

Computing convex hulls

We know how to compute convex hulls

Algorithm INCREMENTALCONVEXHULL(S)

1: $CH \leftarrow p_1, p_2, p_3$
2: $S \leftarrow S \backslash \{p_1, p_2, p_3\}$
3: **for all** $r \in S$ **do**
4: **if** r is outside CH **then**
5: replace CH edges visible from r with two tangent edges $\overline{v_i r}$, $\overline{r v_j}$

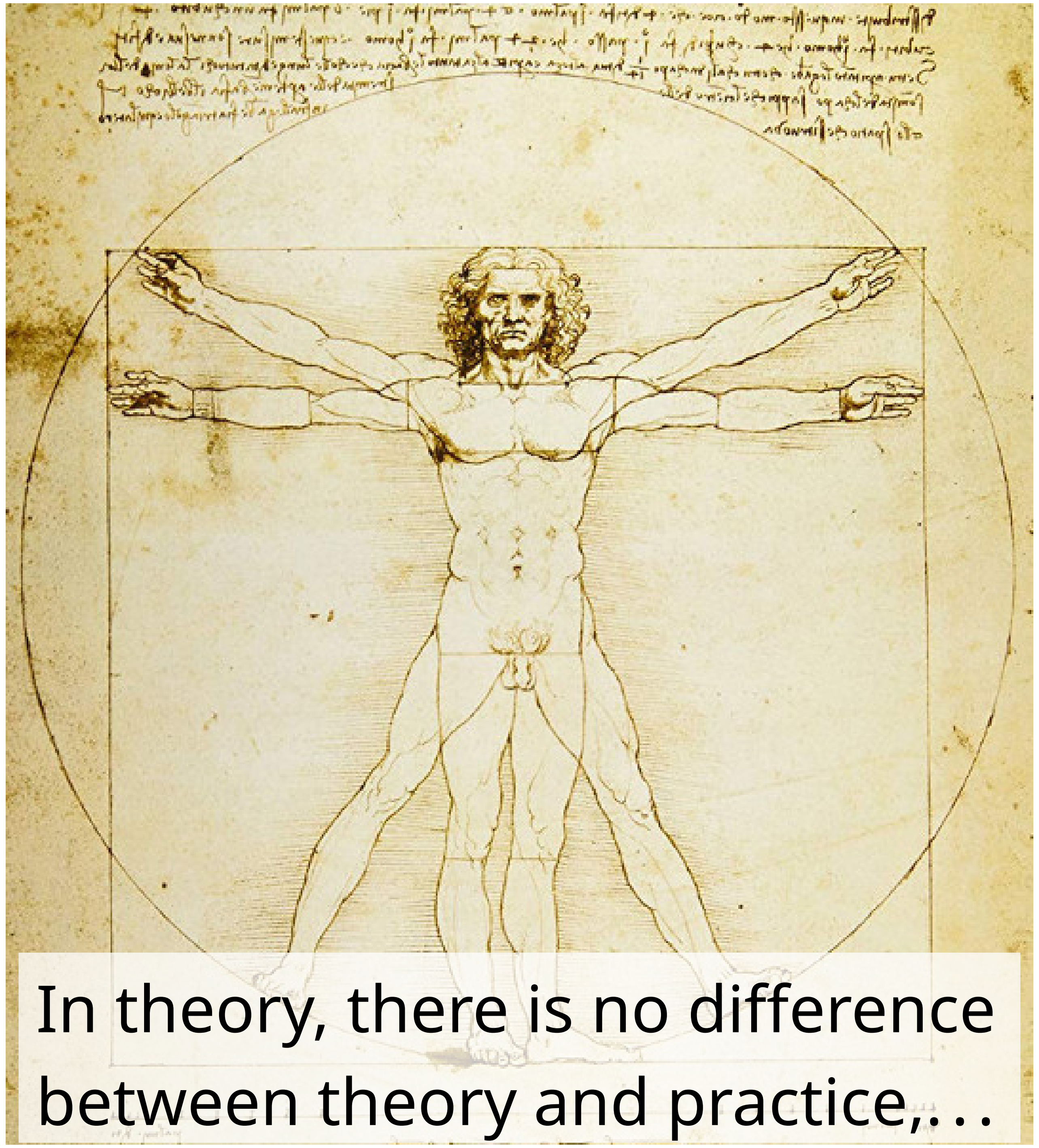


Question: Is this algorithm correct?

This algorithm can sometimes fail!

Why engineering algorithms?

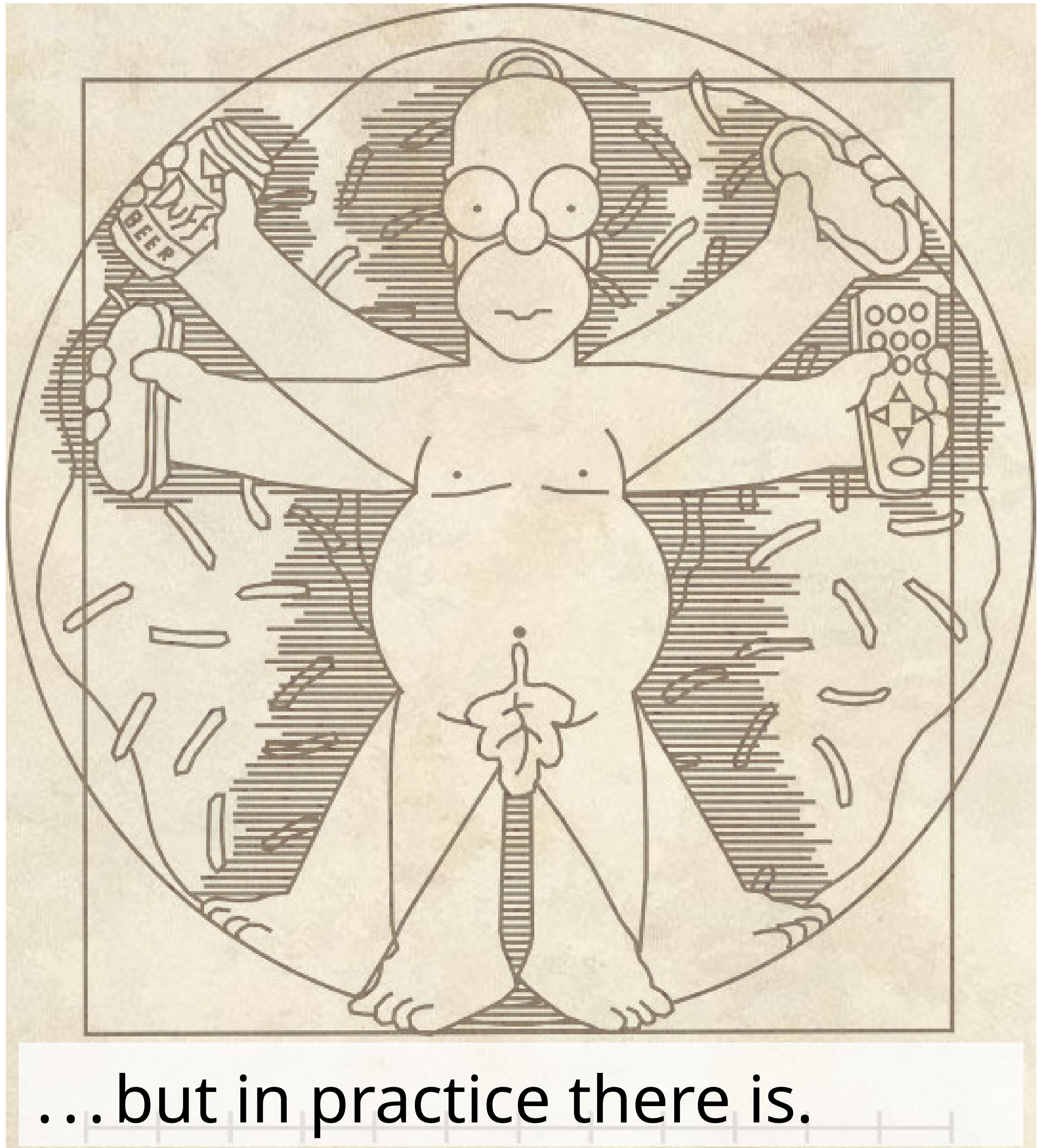
Theory



In theory, there is no difference between theory and practice,. . .

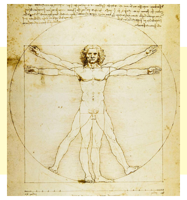
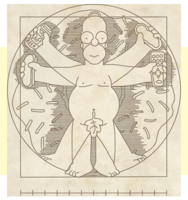
Why engineering algorithms?

The real world out there . . .

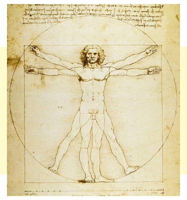
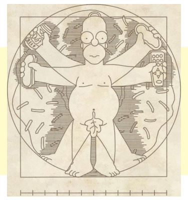


. . . but in practice there is.

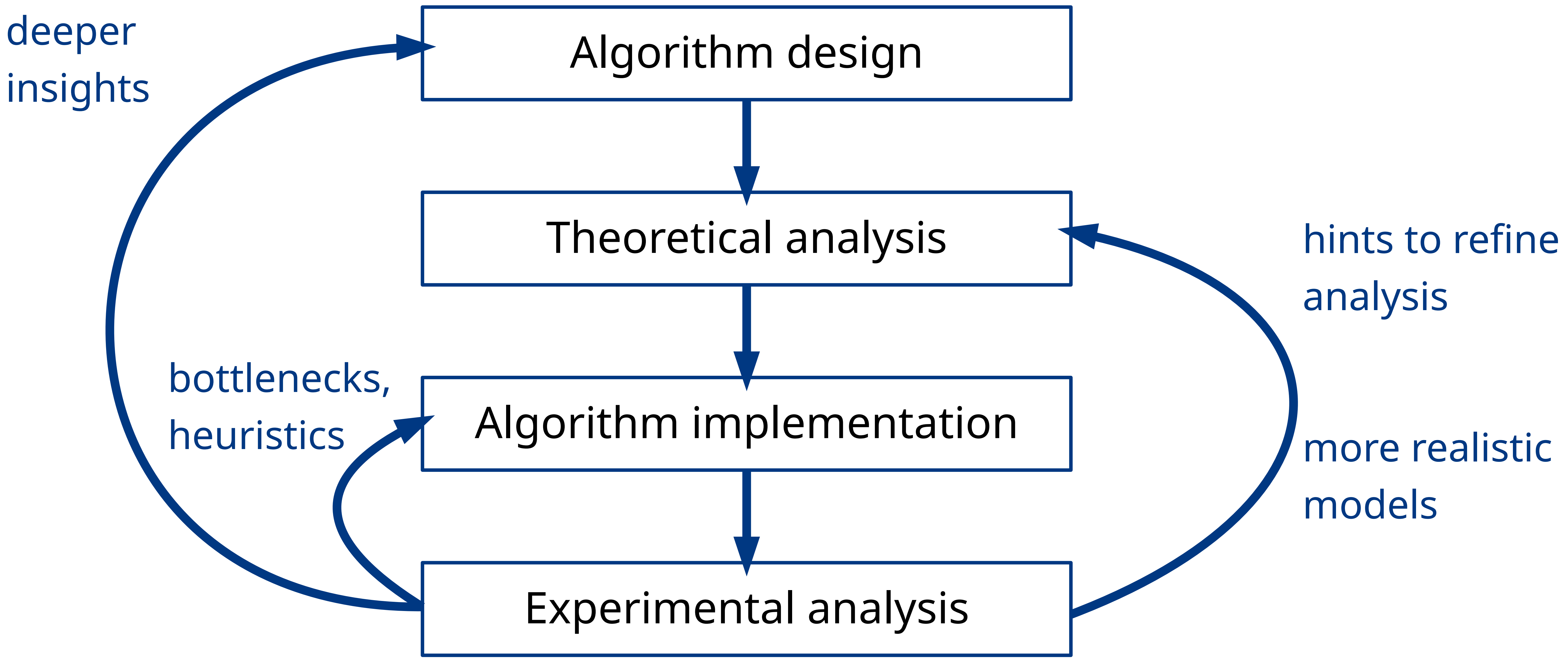
Why engineering algorithms?

Theory 	Practice 
• number types: $\mathbb{N}, \mathbb{R}$	• number types: int, float, double
• only asymptotics matter	• seconds do matter
• abstract algorithm description, often assuming general position	• non-trivial implementation decisions, error-prone
• unbounded memory, unit access cost	• memory hierarchy / bandwidth
• elementary operations take constant time	• instruction pipelining, . . .

Why engineering algorithms?

Theory 	Practice 
• number types: $\mathbb{N}, \mathbb{R}$	• number types: int, float, double
• only asymptotics matter	• seconds do matter
• abstract algorithm description, often assuming general position	• non-trivial implementation decisions, error-prone
• unbounded memory, unit access cost	• memory hierarchy / bandwidth
• elementary operations take constant time	• instruction pipelining, . . .

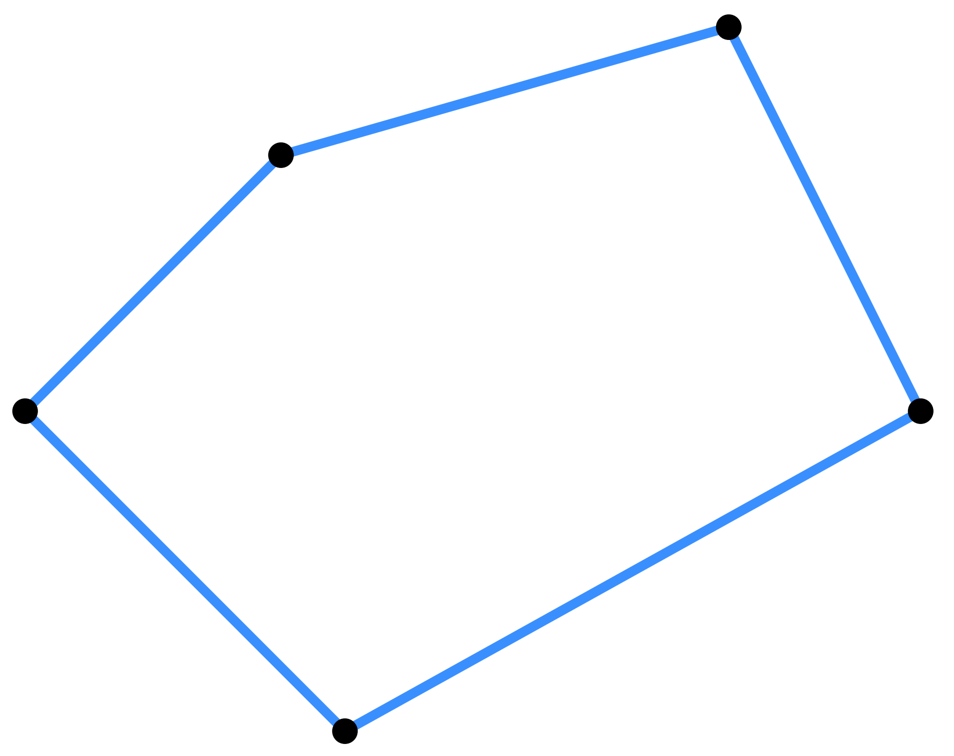
Algorithm engineering cycle



Investigating IncrementalConvexHull

Algorithm IncrementalConvexHull(S)

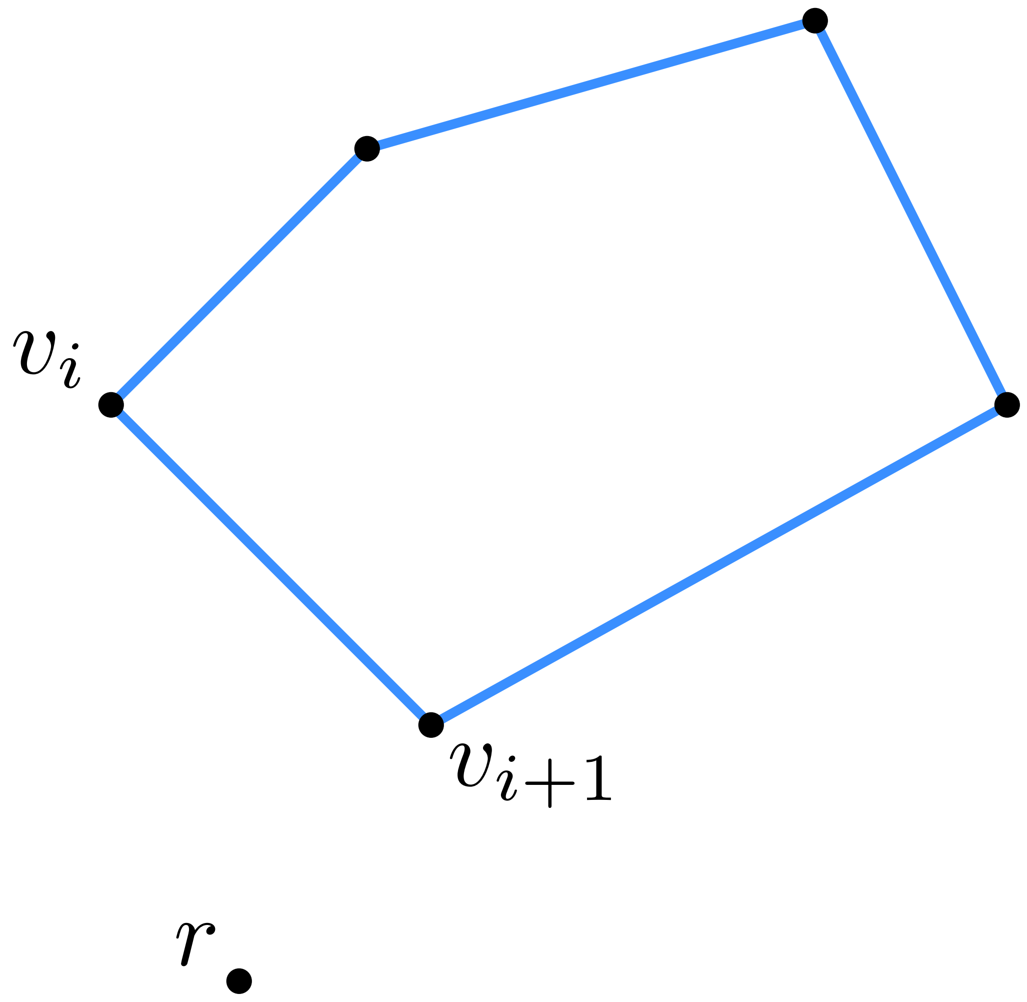
$CH \leftarrow p_1, p_2, p_3$
$S \leftarrow S \setminus \{p_1, p_2, p_3\}$
for all $r \in S$ **do**
    if r is outside CH **then**
        replace CH edges visible from r with two tangent edges $\overline{v_i r}$, $\overline{r v_j}$



Investigating IncrementalConvexHull

Algorithm IncrementalConvexHull(S)

1: $CH \leftarrow p_1, p_2, p_3$
2: $S \leftarrow S \backslash \{p_1, p_2, p_3\}$
3: **for all** $r \in S$ **do**
4:     **if** r is outside CH **then**
5:         replace CH edges visible from r with two tangent edges $\overline{v_i r}$, $\overline{r v_j}$

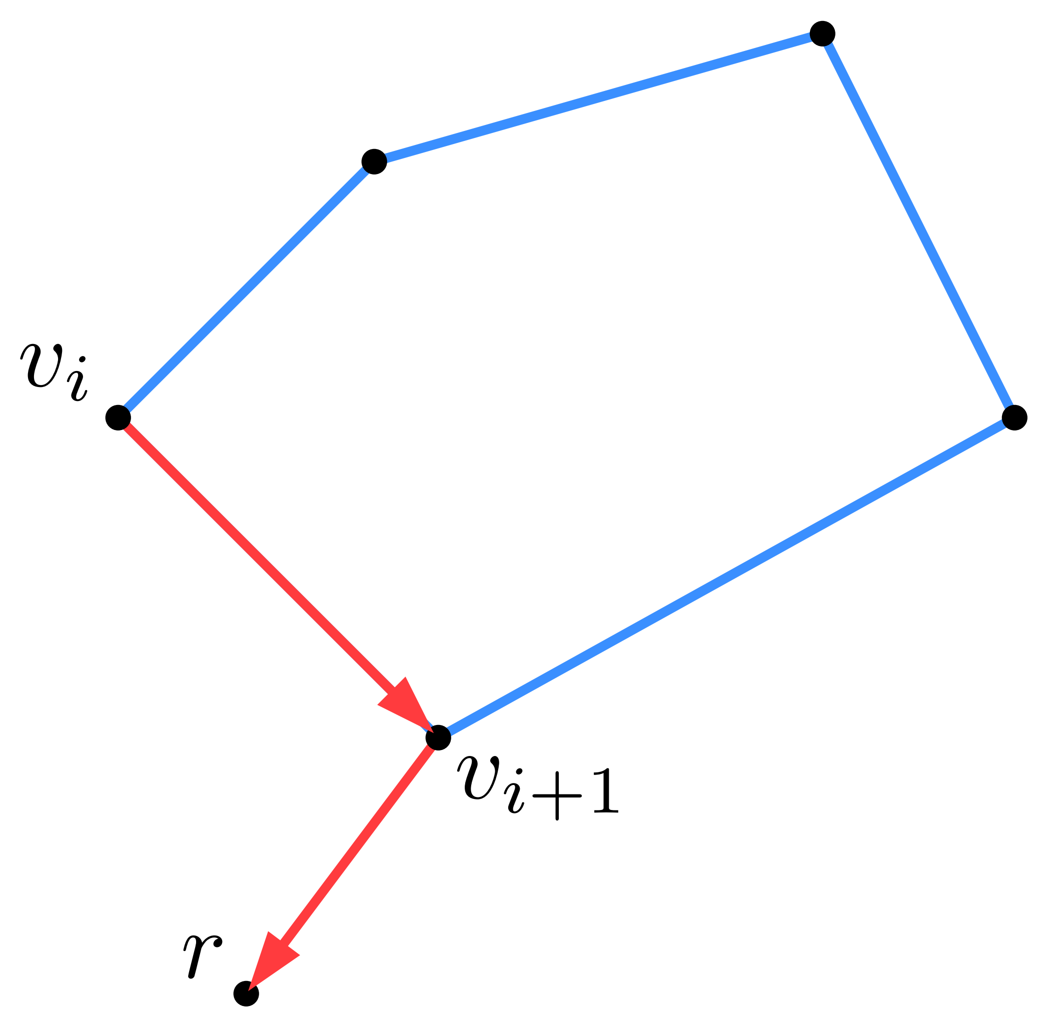


Question: when does r see an edge $\overline{v_i v_{i+1}}$ of CH?

Investigating IncrementalConvexHull

Algorithm IncrementalConvexHull(S)

1: $CH \leftarrow p_1, p_2, p_3$
2: $S \leftarrow S \backslash \{p_1, p_2, p_3\}$
3: **for all** $r \in S$ **do**
4:     **if** r is outside CH **then**
5:         replace CH edges visible from r with two tangent edges $\overline{v_i r}$, $\overline{r v_j}$

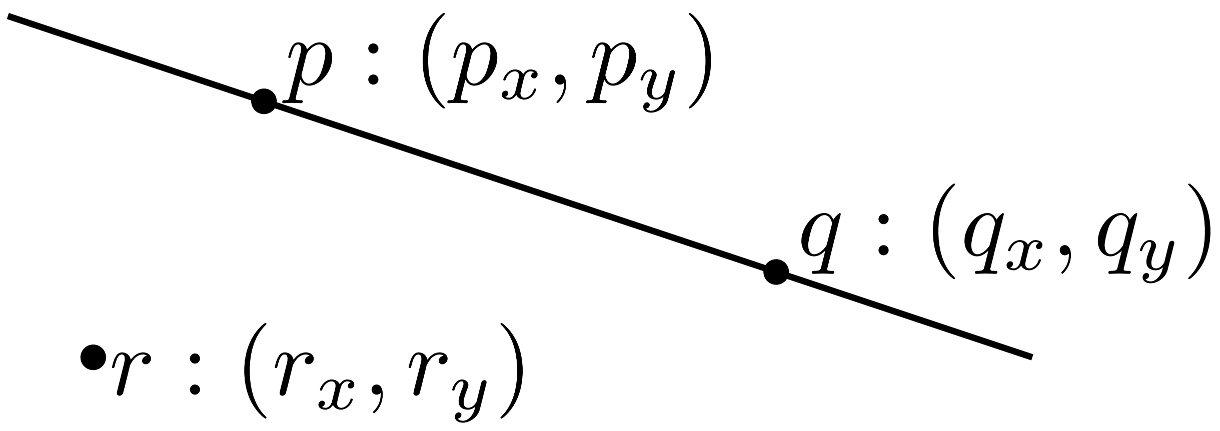


Question: when does r see an edge $\overline{v_i v_{i+1}}$ of CH?

r sees $\overline{v_i v_{i+1}}$ $\Leftrightarrow$ points (v_i, v_{i+1}, r) make a right turn

Orientation

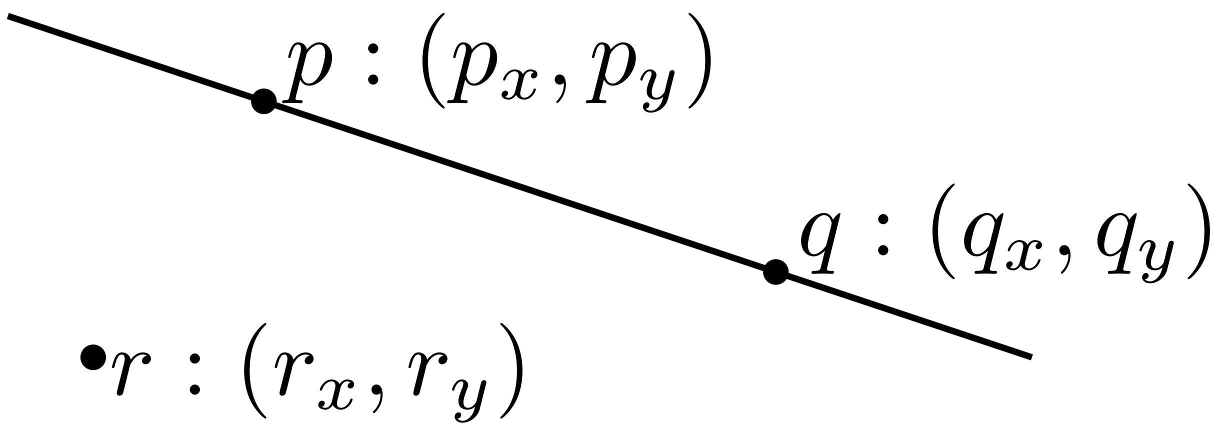
$$orientation(p, q, r) = \begin{cases} +1 & \text{, when } r \text{ lies to the left of } \vec{pq} \\ -1 & \text{, when } r \text{ lies to the right of } \vec{pq} \\ 0 & \text{, when } p, q, \text{ and } r \text{ are collinear} \end{cases}$$



Orientation

$$orientation(p,q,r) = \begin{cases} +1 & \text{, when } r \text{ lies to the left of } \vec{pq} \\ -1 & \text{, when } r \text{ lies to the right of } \vec{pq} \\ 0 & \text{, when } p, q, \text{ and } r \text{ are collinear} \end{cases}$$

$$= sign\left(\text{det}\begin{bmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{bmatrix}\right)$$

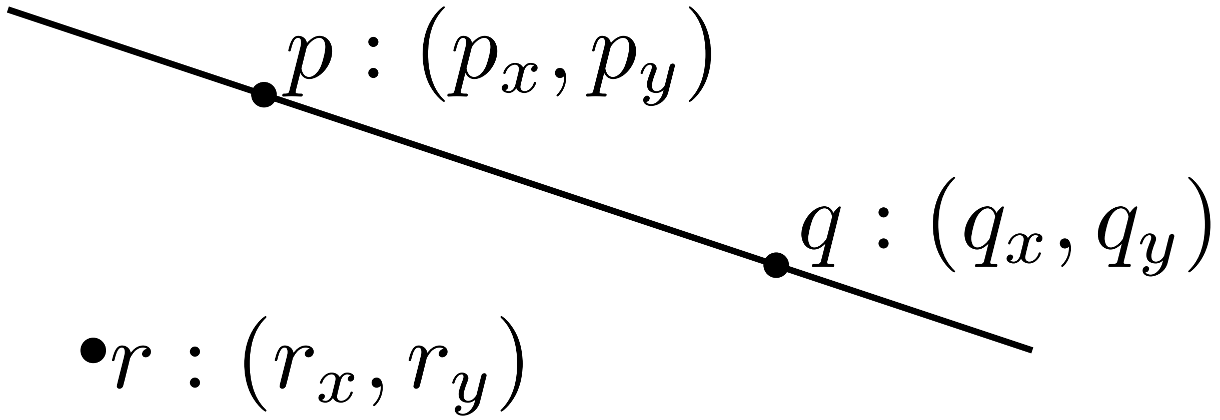


Orientation

$$orientation(p,q,r) = \begin{cases} +1 & \text{, when } r \text{ lies to the left of } \vec{pq} \\ -1 & \text{, when } r \text{ lies to the right of } \vec{pq} \\ 0 & \text{, when } p, q, \text{ and } r \text{ are collinear} \end{cases}$$

$$= sign\left(\det \begin{bmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{bmatrix}\right)$$

$$= sign\left((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x)\right)$$



Floating point numbers

Machine floating-point numbers:

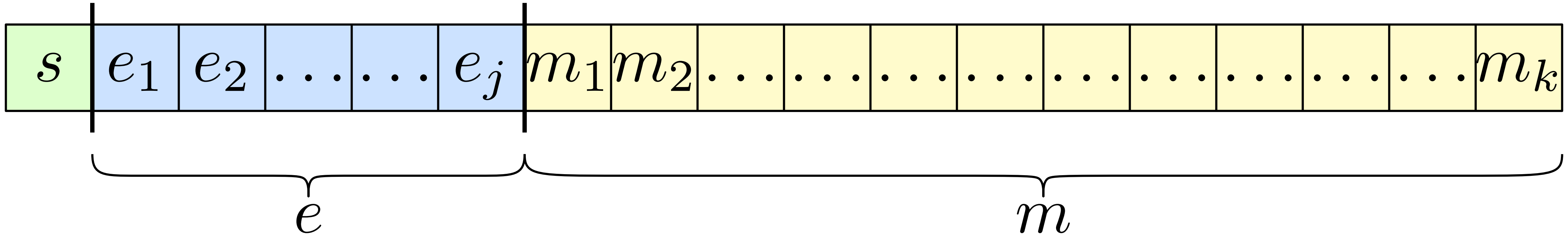
$$\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline s & e_1 & e_2 & \ldots & \ldots & e_j & m_1 & m_2 & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & m_k \\ \hline \end{array}$$

$$s\ \underbrace{e_1\ e_2\ \ldots\ \ldots\ e_j}_{e}\ \underbrace{m_1\ m_2\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ m_k}_{m}$$

Normalized:	$(-1)^s \cdot 1.m \cdot 2^{e-(2^{j-1}-1)}$	($e \neq 00\ldots0$, $e \neq 11\ldots1$)
Denormalized:	$(-1)^s \cdot 0.m \cdot 2^{-(2^{j-1}-2)}$	($e = 00\ldots0$)
Special numbers:	infinity and NaN	($e = 11\ldots1$)

Floating point numbers

Machine floating-point numbers:



Normalized:	$(-1)^s \cdot 1.m \cdot 2^{e-(2^{j-1}-1)}$	($e \neq 00\ldots0$, $e \neq 11\ldots1$)
Denormalized:	$(-1)^s \cdot 0.m \cdot 2^{-(2^{j-1}-2)}$	($e = 00\ldots0$)
Special numbers:	infinity and NaN	($e = 11\ldots1$)

	Normalized	Denormalized
`float`	$\pm 1.m_1 m_2 \ldots m_{23} \cdot 2^{e-127}$	$\pm 0.m_1 m_2 \ldots m_{23} \cdot 2^{-126}$
`double`	$\pm 1.m_1 m_2 \ldots m_{52} \cdot 2^{e-1023}$	$\pm 0.m_1 m_2 \ldots m_{52} \cdot 2^{-1022}$

Floating point numbers

Consider number type:

$$\texttt{very short float} = \begin{cases} \pm 1.m_1m_2m_3 \cdot 2^{e-3}, & \text{if } 0 < e < 7 \\ \pm 0.m_1m_2m_3 \cdot 2^{-2}, & \text{if } e = 0 \end{cases}$$

Floating point numbers

Consider number type:

$$\texttt{very short float} = \begin{cases} \pm 1.m_1m_2m_3 \cdot 2^{e-3}, & \text{if } 0 < e < 7 \\ \pm 0.m_1m_2m_3 \cdot 2^{-2}, & \text{if } e = 0 \end{cases}$$

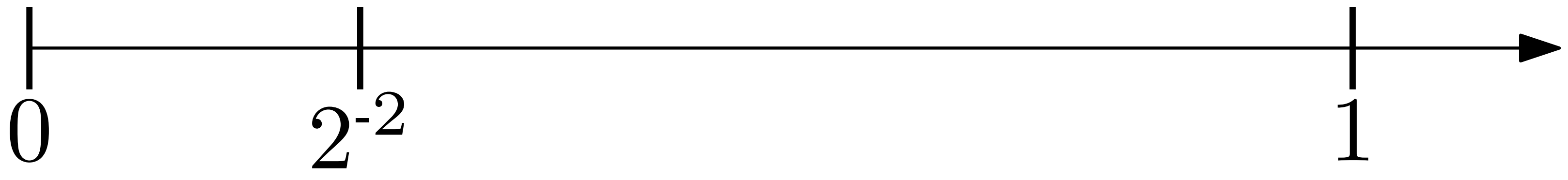
Question: What is the smallest strictly positive normalized number?

Floating point numbers

Consider number type:

$$\texttt{very short float} = \begin{cases} \pm 1.m_1m_2m_3 \cdot 2^{e-3}, & \text{if } 0 < e < 7 \\ \pm 0.m_1m_2m_3 \cdot 2^{-2}, & \text{if } e = 0 \end{cases}$$

- $1.000_2 \cdot 2^{-2} = 0.25$ smallest strictly positive normalized number



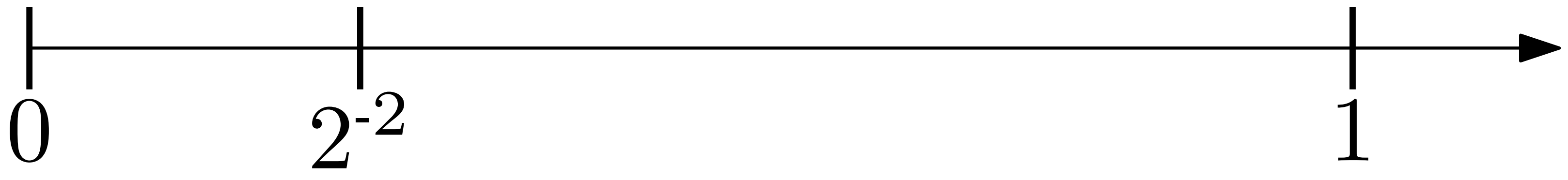
Floating point numbers

Consider number type:

$$\texttt{very short float} = \begin{cases} \pm 1.m_1m_2m_3 \cdot 2^{e-3}, & \text{if } 0 < e < 7 \\ \pm 0.m_1m_2m_3 \cdot 2^{-2}, & \text{if } e = 0 \end{cases}$$

- $1.000_2 \cdot 2^{-2} = 0.25$ smallest strictly positive normalized number

Question: What is the next smallest normalized number?

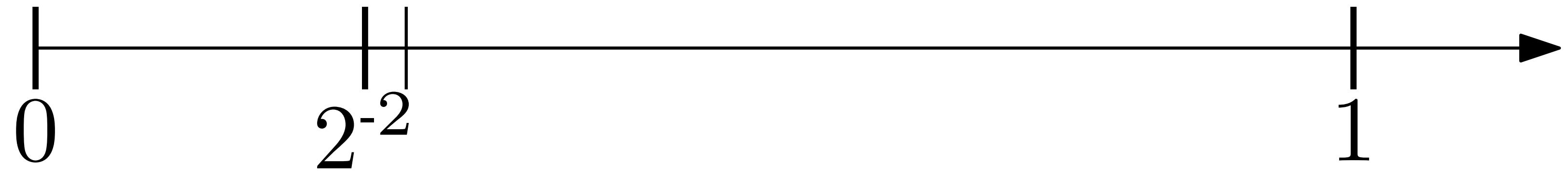


Floating point numbers

Consider number type:

$$\texttt{very short float} = \begin{cases} \pm 1.m_1 m_2 m_3 \cdot 2^{e-3}, & \text{if } 0 < e < 7 \\ \pm 0.m_1 m_2 m_3 \cdot 2^{-2}, & \text{if } e = 0 \end{cases}$$

- $1.000_2 \cdot 2^{-2} = 0.25$ smallest strictly positive normalized number
- $1.001_2 \cdot 2^{-2} = 0.28125$ is the next smallest normalized number

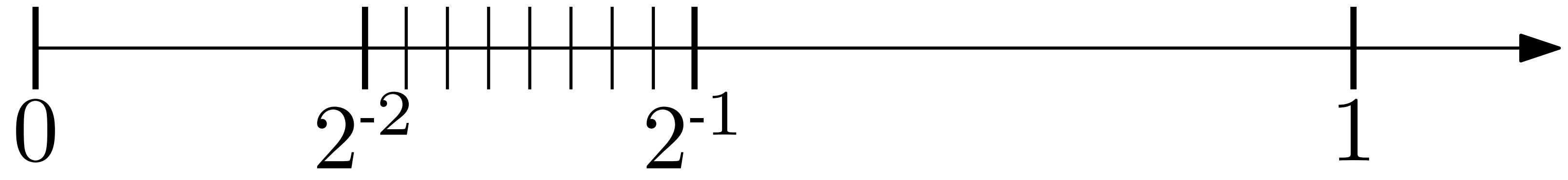


Floating point numbers

Consider number type:

$$\texttt{very short float} = \begin{cases} \pm 1.m_1 m_2 m_3 \cdot 2^{e-3}, & \text{if } 0 < e < 7 \\ \pm 0.m_1 m_2 m_3 \cdot 2^{-2}, & \text{if } e = 0 \end{cases}$$

- $1.000_2 \cdot 2^{-2} = 0.25$ smallest strictly positive normalized number
- $1.001_2 \cdot 2^{-2} = 0.28125$ is the next smallest normalized number
- the increment is 2^{-5}

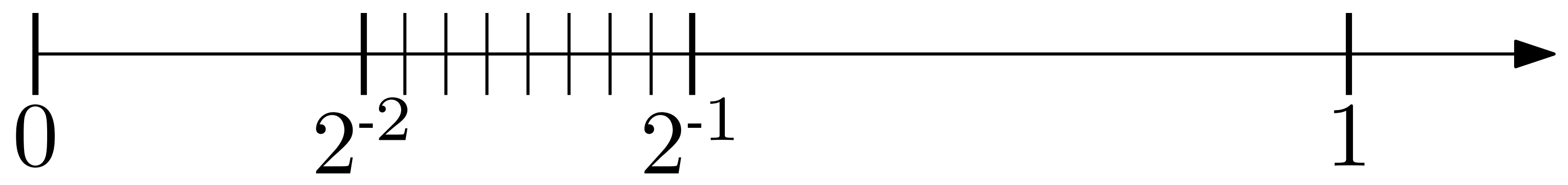


Floating point numbers

Consider number type:

$$\texttt{very short float} = \begin{cases} \pm 1.m_1m_2m_3 \cdot 2^{e-3}, & \text{if } 0 < e < 7 \\ \pm 0.m_1m_2m_3 \cdot 2^{-2}, & \text{if } e = 0 \end{cases}$$

- $1.000_2 \cdot 2^{-2} = 0.25$ smallest strictly positive normalized number
- $1.001_2 \cdot 2^{-2} = 0.28125$ is the next smallest normalized number
- the increment is 2^{-5}
- the increment between $1.000_2 \cdot 2^{-1}$ and the next number is 2^{-4}

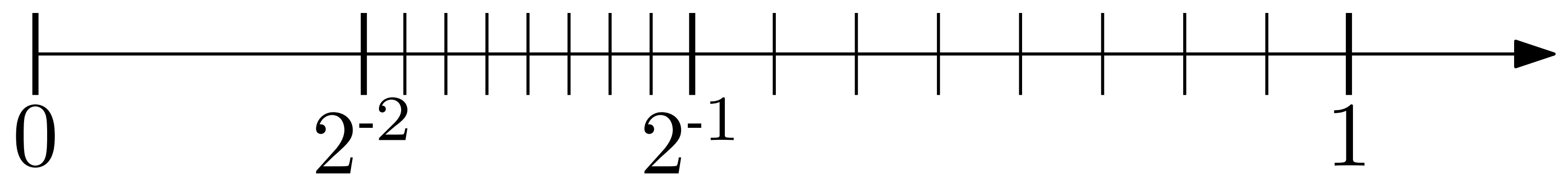


Floating point numbers

Consider number type:

$$\texttt{very short float} = \begin{cases} \pm 1.m_1 m_2 m_3 \cdot 2^{e-3}, & \text{if } 0 < e < 7 \\ \pm 0.m_1 m_2 m_3 \cdot 2^{-2}, & \text{if } e = 0 \end{cases}$$

- $1.000_2 \cdot 2^{-2} = 0.25$ smallest strictly positive normalized number
- $1.001_2 \cdot 2^{-2} = 0.28125$ is the next smallest normalized number
- the increment is 2^{-5}
- the increment between $1.000_2 \cdot 2^{-1}$ and the next number is 2^{-4}

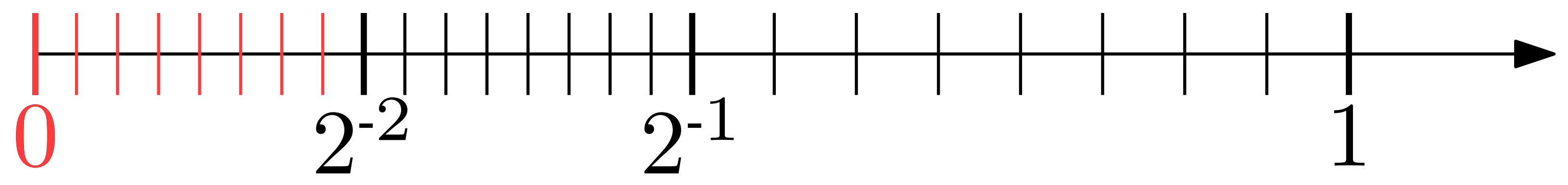


Floating point numbers

Consider number type:

$$\texttt{very short float} = \begin{cases} \pm 1.m_1m_2m_3 \cdot 2^{e-3}, & \text{if } 0 < e < 7 \\ \pm 0.m_1m_2m_3 \cdot 2^{-2}, & \text{if } e = 0 \end{cases}$$

- $1.000_2 \cdot 2^{-2} = 0.25$ smallest strictly positive normalized number
- $1.001_2 \cdot 2^{-2} = 0.28125$ is the next smallest normalized number
- the increment is 2^{-5}
- the increment between $1.000_2 \cdot 2^{-1}$ and the next number is 2^{-4}
- denormalized numbers lie in $(-2^{-2}, 2^{-2})$ with increment 2^{-5}



Quiz

Consider number type:

$$\texttt{very short float} = \pm 1.m_1 m_2 m_3 \cdot 2^{e-3} \qquad (0 < e < 7)$$

Quiz

Consider number type:

$$\texttt{very short float} = \pm 1.m_1 m_2 m_3 \cdot 2^{e-3} \qquad (0 < e < 7)$$

What is $4 + 0.25$?

Quiz

Consider number type:

$$\texttt{very short float} = \pm 1.m_1m_2m_3 \cdot 2^{e-3} \quad (0 < e < 7)$$

What is $4 + 0.25$?

A: 4

B: 4.25

C: 4.3

Quiz

Consider number type:

$$\texttt{very short float} = \pm 1.m_1m_2m_3 \cdot 2^{e-3} \quad (0 < e < 7)$$

What is $4 + 0.25$?

A: 4

B: 4.25

C: 4.3

Orientation in doubles

Consider three points p, q, r

Orientation in doubles

Consider three points p, q, r

Plot $\mathit{float\_orient}(p', q, r)$

- $p' = p + (u\Delta x, v\Delta y)$,
  whereΔx and Δy are increments between adjacent doubles
- u and v are in $[0, 255]$

Orientation in doubles

Consider three points p, q, r

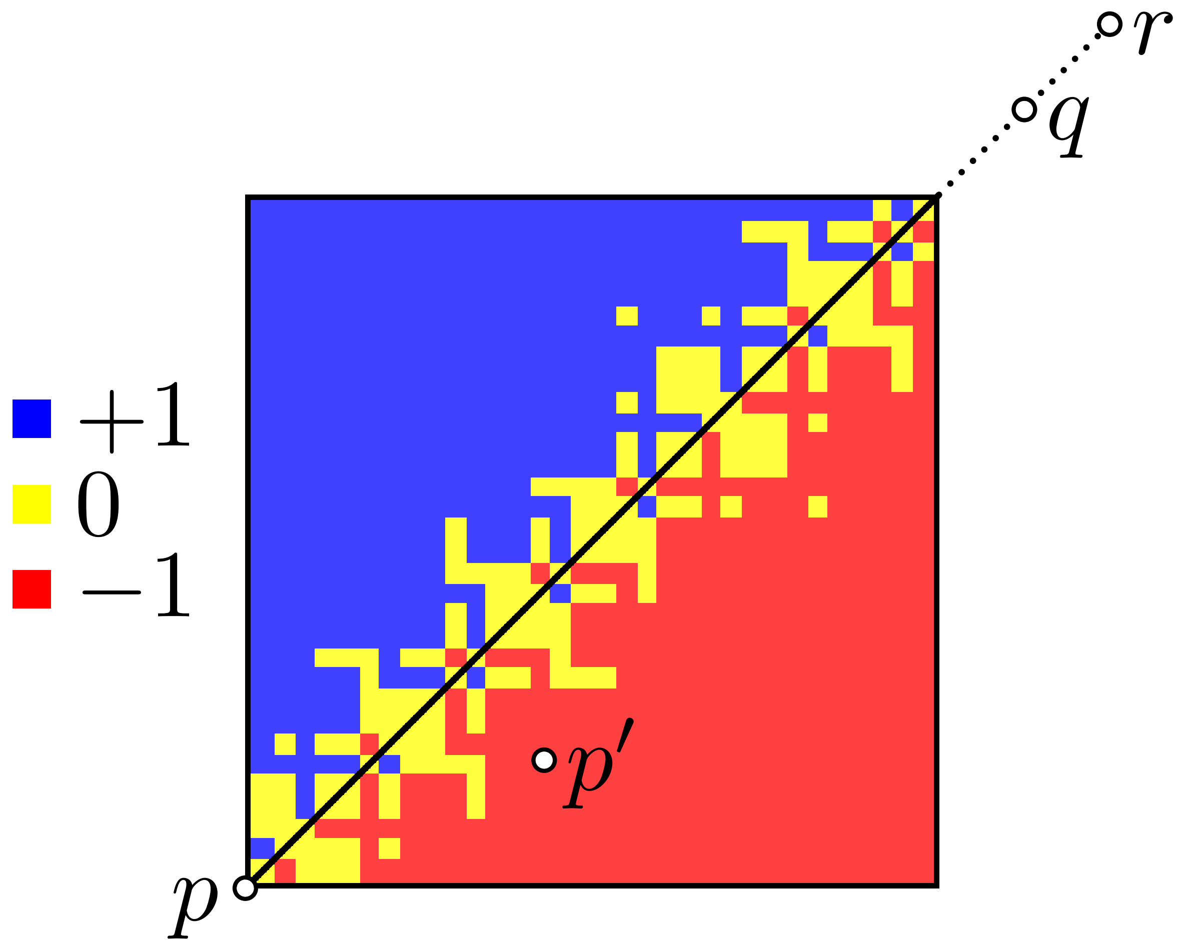
Plot $\mathit{float\_orient}(p', q, r)$

- $p' = p + (u\Delta x, v\Delta y)$,
  whereΔx and Δy are increments between adjacent doubles
- u and v are in $[0, 255]$

$p = (0.5, 0.5)$

$q = (12, 12)$

$r = (24, 24)$



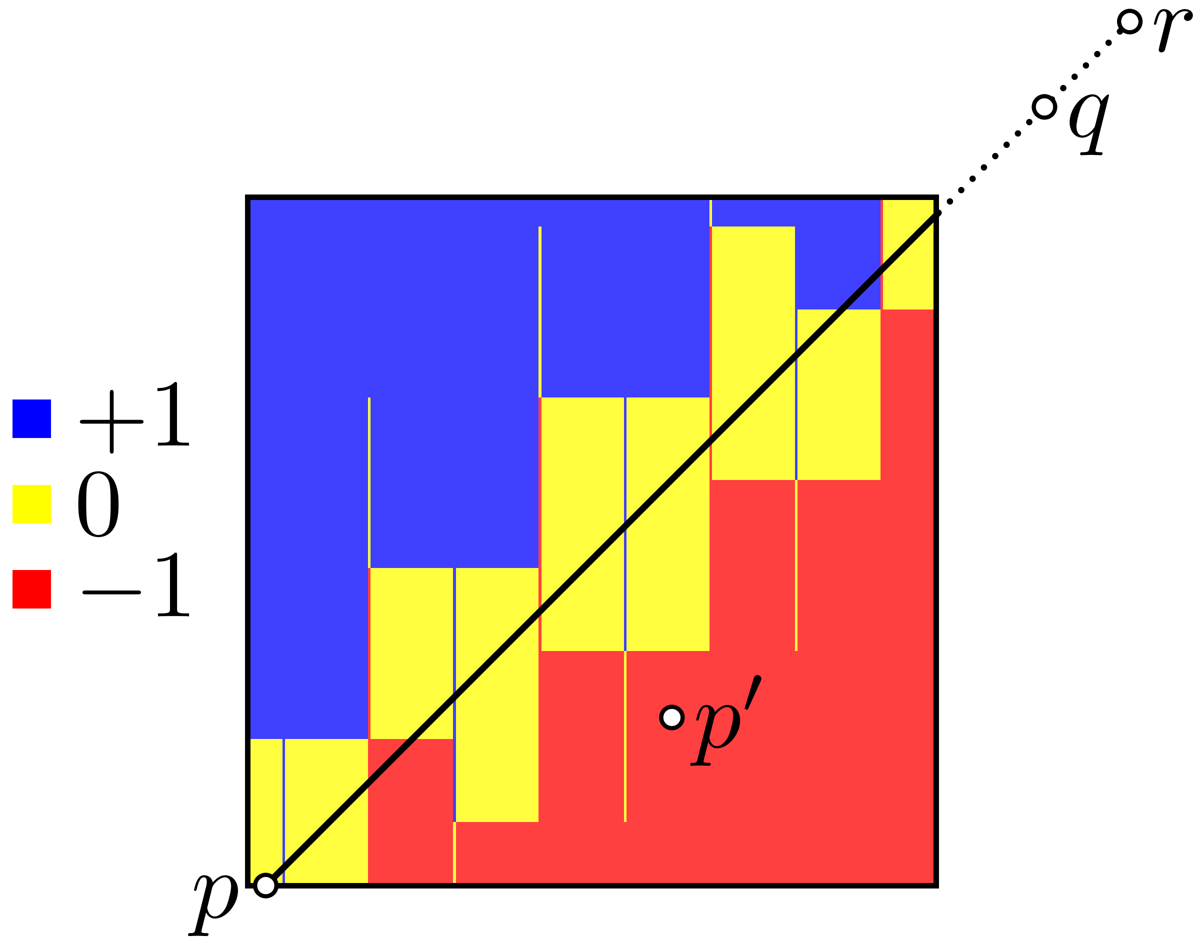
Orientation in doubles

Consider three points p, q, r

Plot $\mathit{float\_orient}(p', q, r)$

- $p' = p + (u\Delta x, v\Delta y)$,
  where Δx and Δy are increments between adjacent doubles
- u and v are in $[0, 255]$

$$p = (0.5000..02531,\ 0.5000..0171)$$
$$q = (17.3000..001,\ 17.3000..001)$$
$$r = (24.000..005,\ 24.000..005)$$



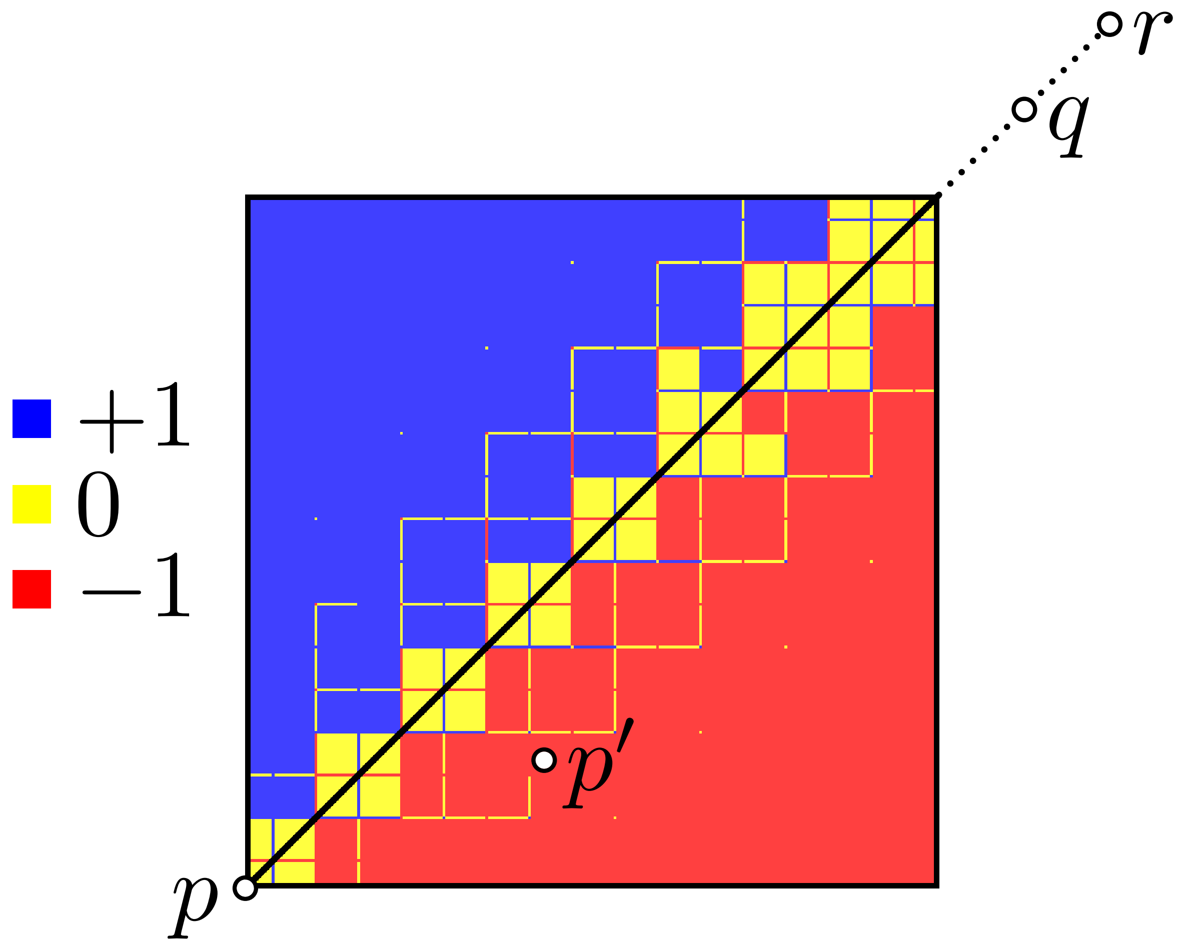
Orientation in doubles

Consider three points p, q, r

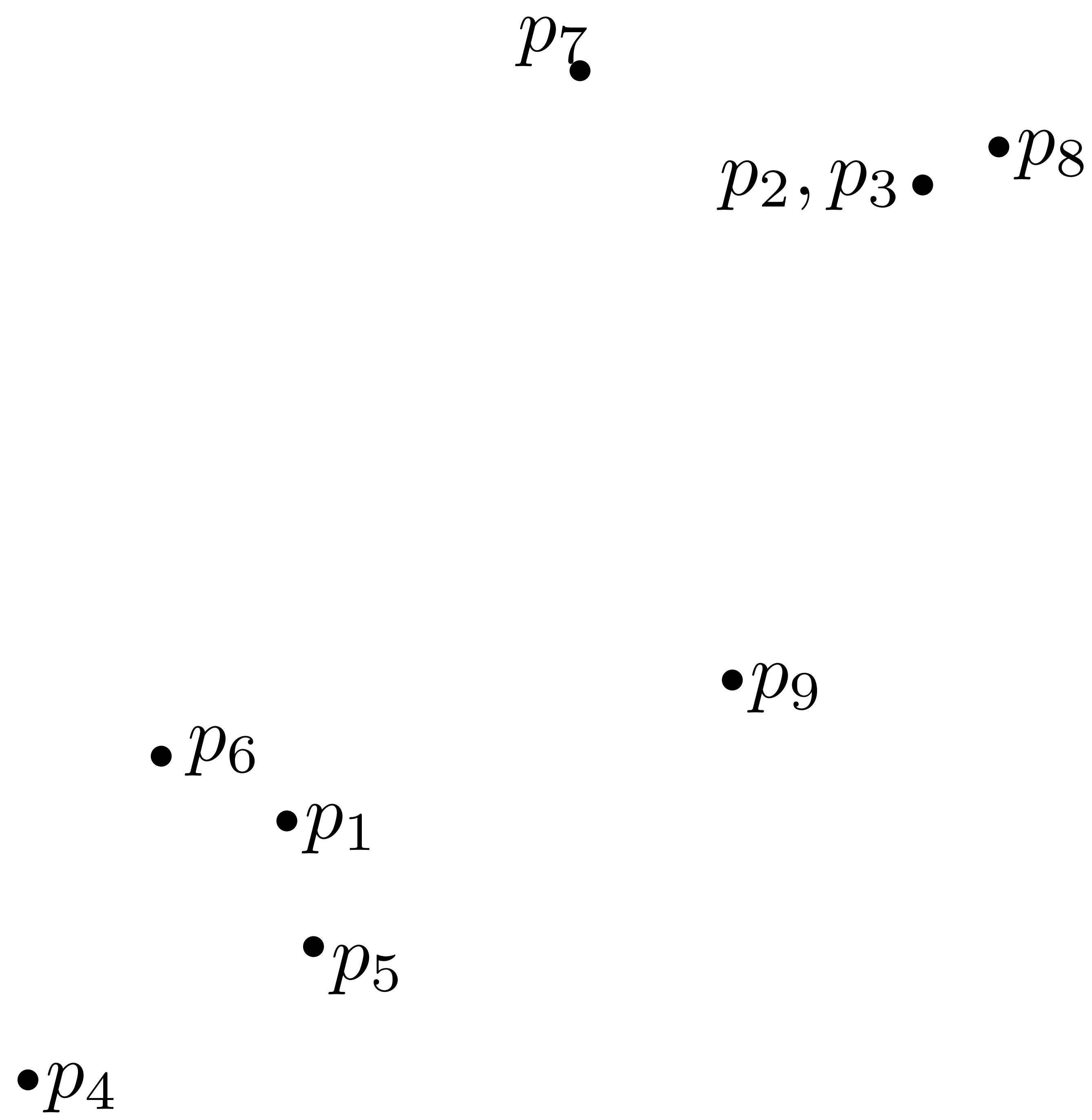
Plot $\mathit{float\_orient}(p', q, r)$

- $p' = p + (u\Delta x, v\Delta y)$,
  where Δx and Δy are increments between adjacent doubles
- u and v are in $[0, 255]$

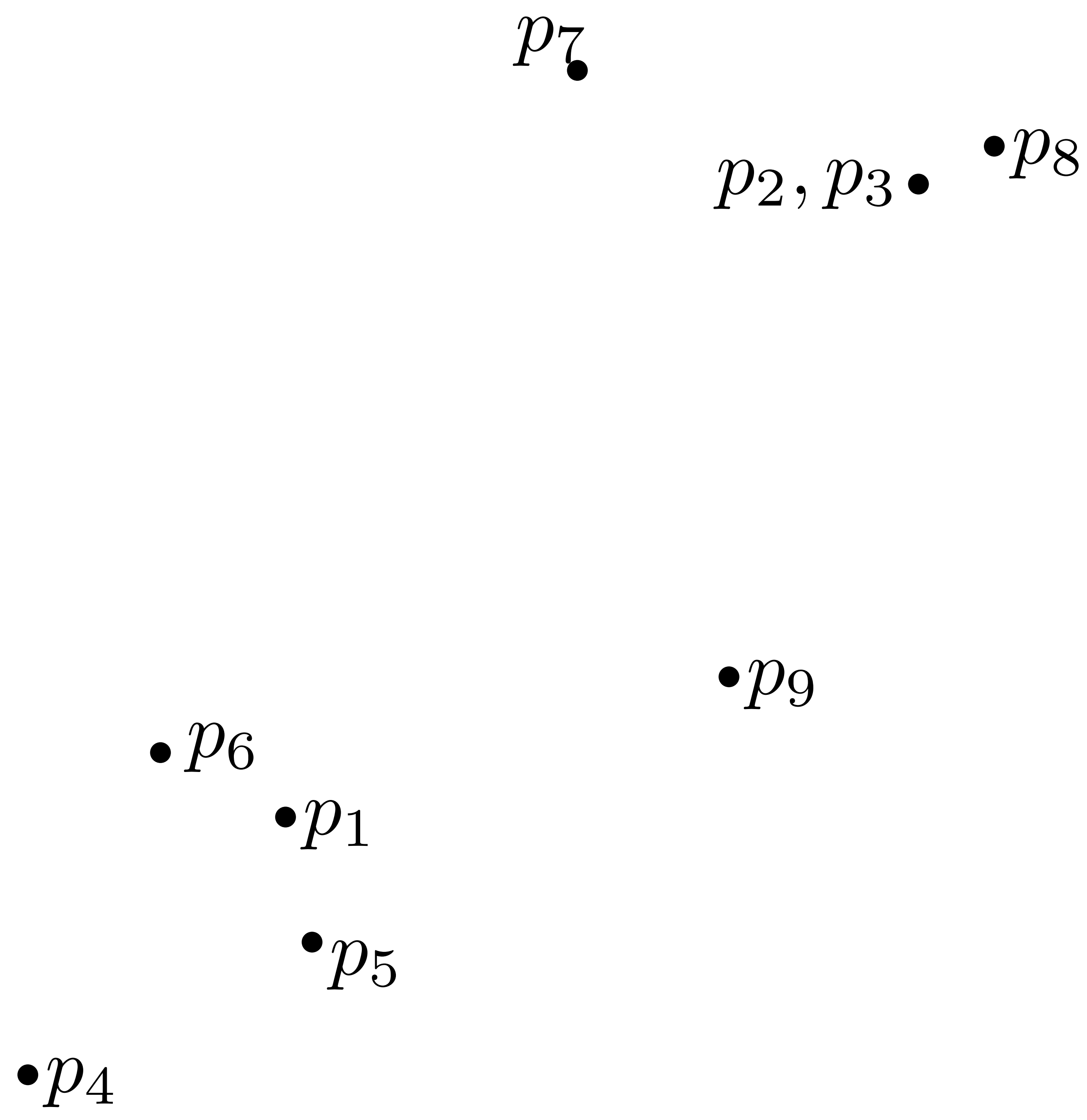
$$p = (0.5, 0.5)$$
$$q = (8.8000..007, 8.8000..007)$$
$$r = (12.1, 12.1)$$



Investigating INCREMENTALCONVEXHULL

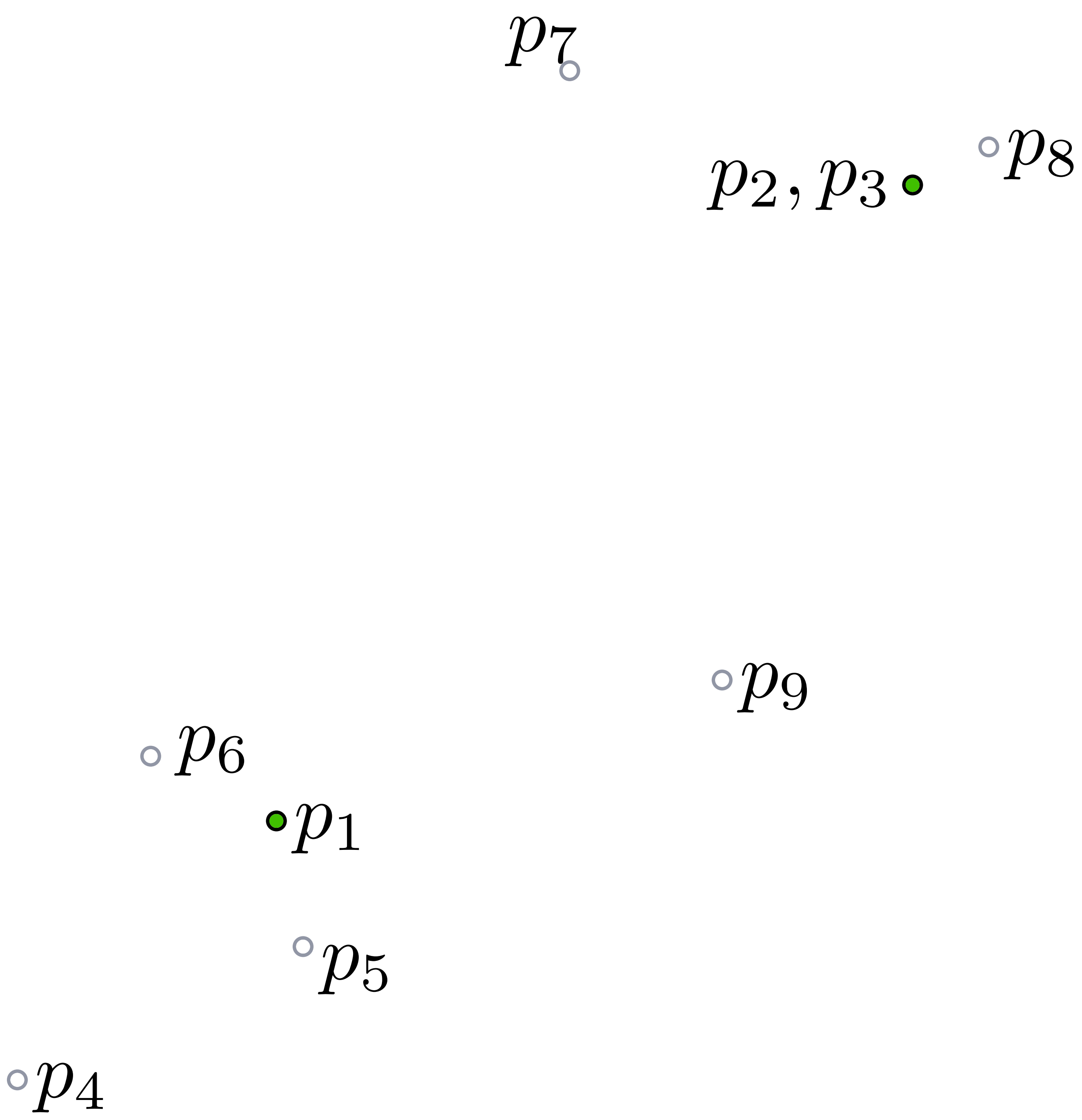


Investigating INCREMENTALCONVEXHULL



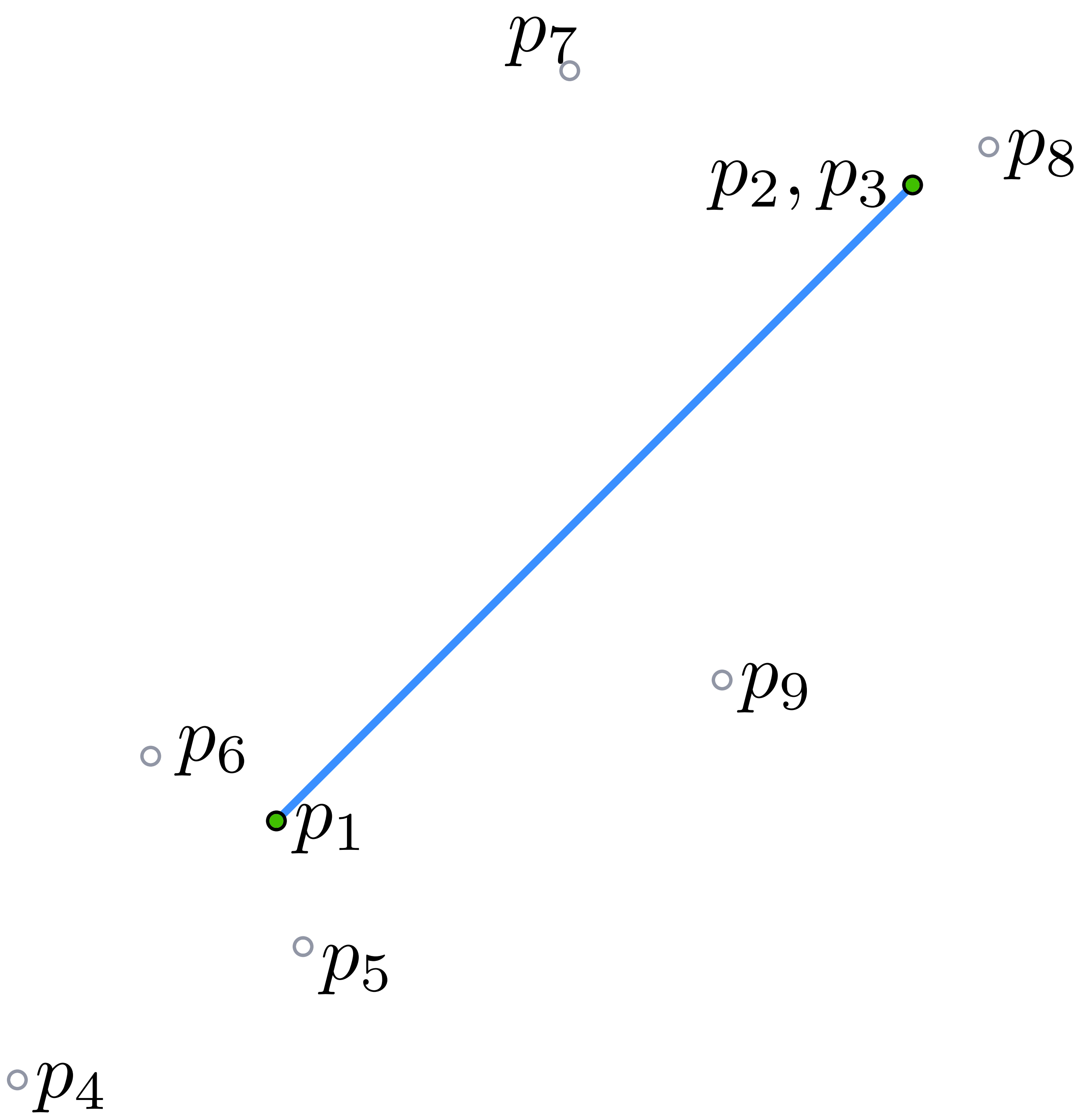
* refer to [Kettner et al.] for exact coordinates

Investigating IncrementalConvexHull



* refer to [Kettner et al.] for exact coordinates

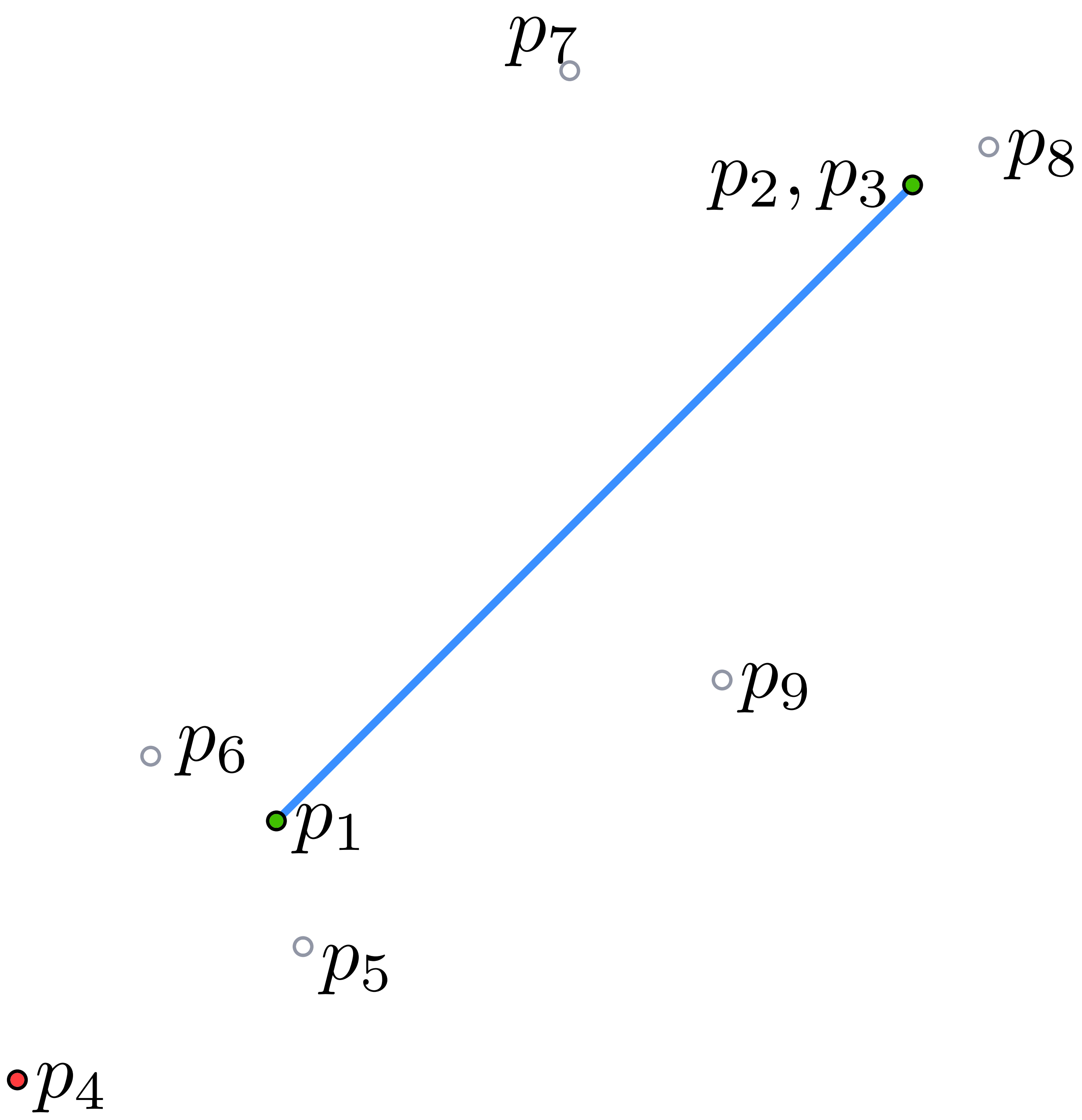
Investigating IncrementalConvexHull



* refer to [Kettner et al.] for exact coordinates

Investigating IncrementalConvexHull

$$\mathit{float\_orient}(p_1, p_2, p_3) > 0$$
$$\mathit{float\_orient}(p_1, p_2, p_4) > 0$$
$$\mathit{float\_orient}(p_2, p_3, p_4) > 0$$
$$\mathit{float\_orient}(p_3, p_1, p_4) > 0$$

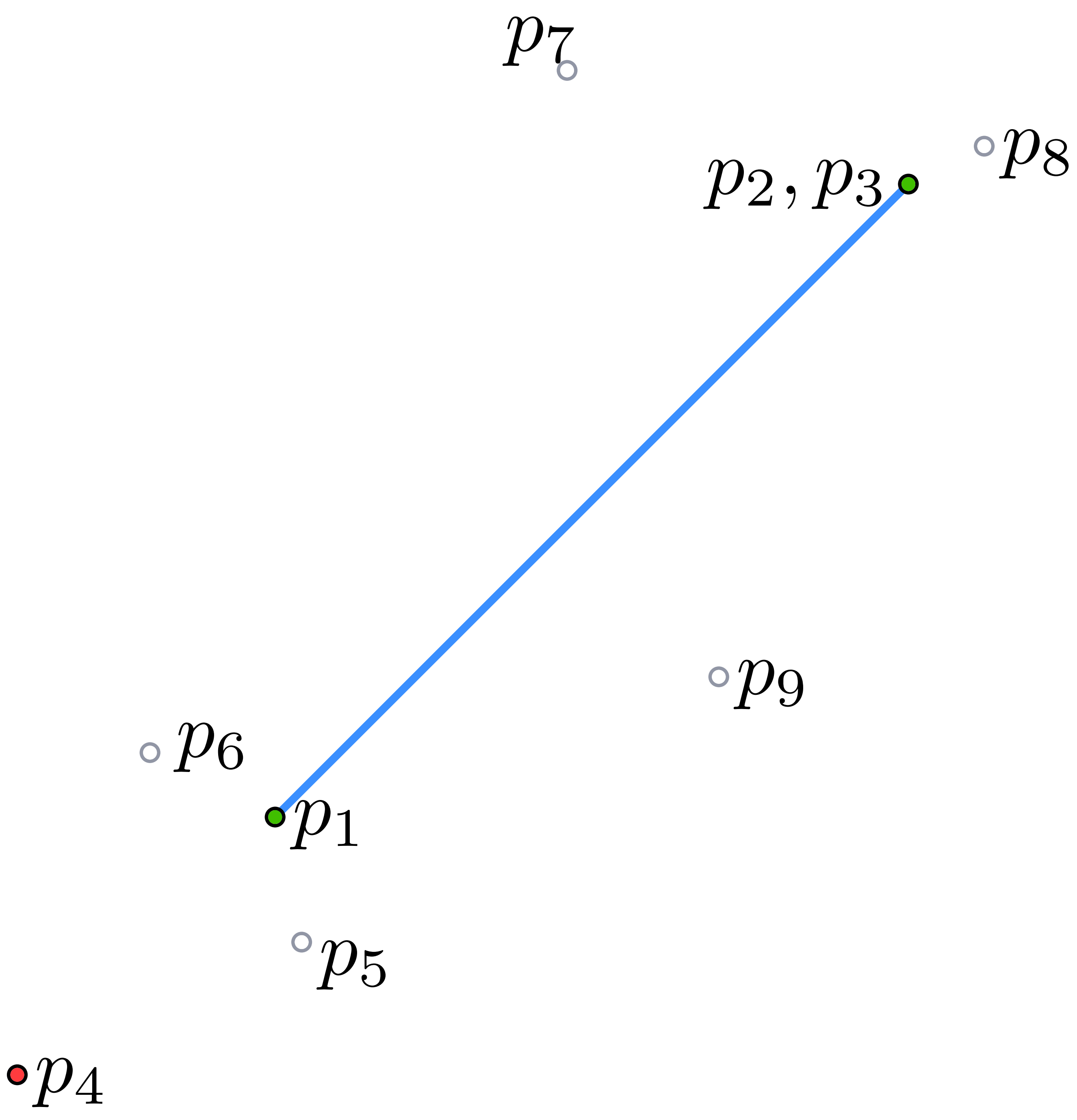


* refer to [Kettner et al.] for exact coordinates

Investigating IncrementalConvexHull

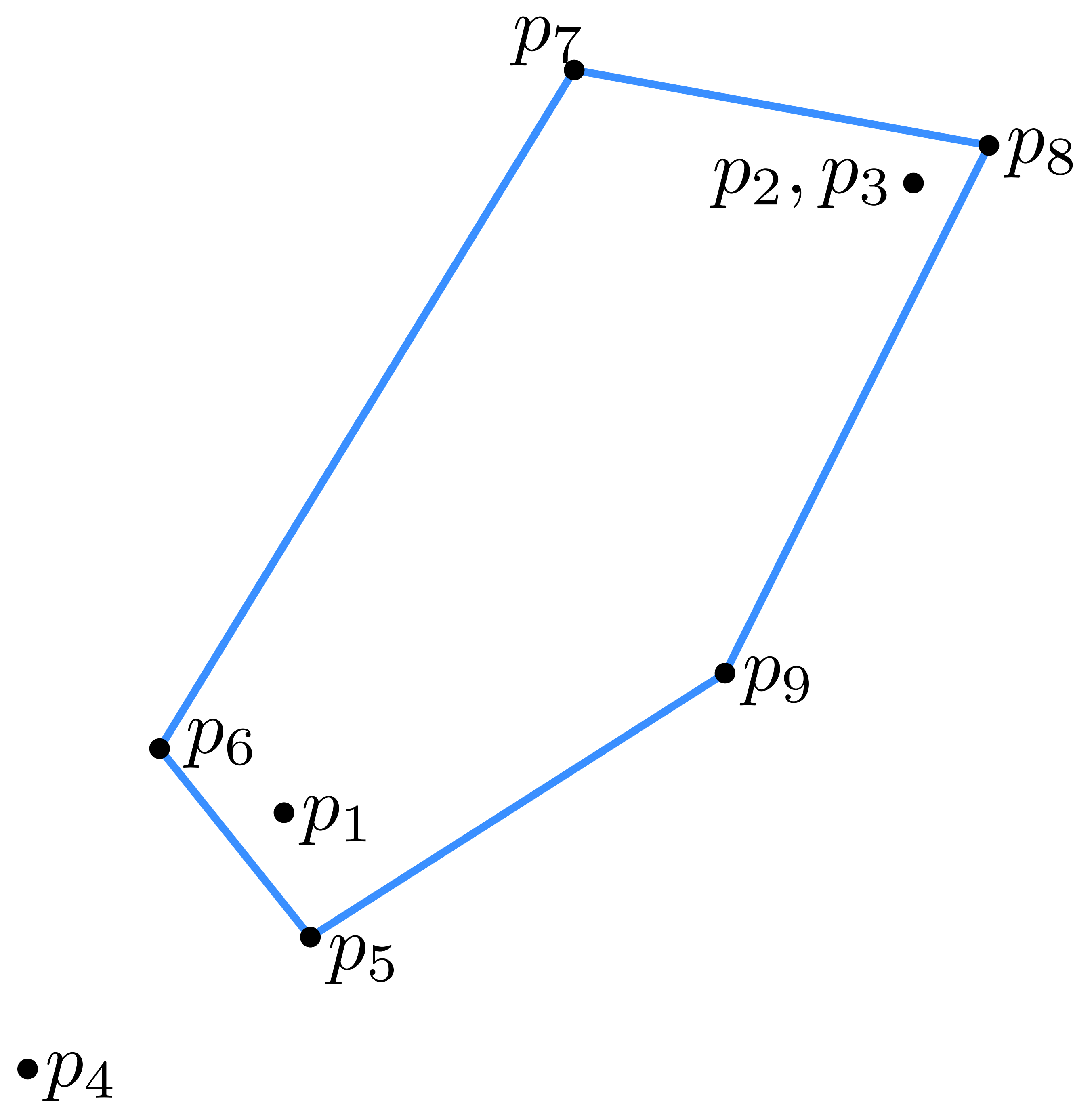
$$float\_orient(p_1, p_2, p_3) > 0$$
$$float\_orient(p_1, p_2, p_4) > 0$$
$$float\_orient(p_2, p_3, p_4) > 0$$
$$float\_orient(p_3, p_1, p_4) > 0$$

$\Rightarrow$ p_4 does not see any edge!



* refer to [Kettner et al.] for exact coordinates

Investigating IncrementalConvexHull



* refer to [Kettner et al.] for exact coordinates

Investigating INCREMENTALCONVEXHULL

Failure:

- point outside convex hull doesn't see an edge $\Rightarrow$ incorrect solution

Investigating INCREMENTALCONVEXHULL

Failure:

- point outside convex hull doesn't see an edge $\Rightarrow$ incorrect solution

Other failures include:

- point inside convex hull sees an edge $\Rightarrow$ non-convex solution

Investigating INCREMENTALCONVEXHULL

Failure:

- point outside convex hull doesn’t see an edge $\Rightarrow$ incorrect solution

Other failures include:

- point inside convex hull sees an edge $\Rightarrow$ non-convex solution
- point outside convex hull sees all edges $\Rightarrow$ infinite loop? crash?

Investigating IncrementalConvexHull

Failure:

- point outside convex hull doesn't see an edge $\Rightarrow$ incorrect solution

Other failures include:

- point inside convex hull sees an edge $\Rightarrow$ non-convex solution
- point outside convex hull sees all edges $\Rightarrow$ infinite loop? crash?
- point outside convex hull sees a non-contiguous set of edges $\Rightarrow$ non-convex solution, self-intersecting chain

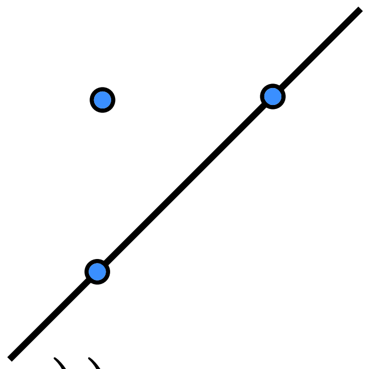
Geometric predicates

Definition: geometric predicate is a sign of a polynomial evaluated with the coordinates of the input.

Geometric predicates

Definition: geometric predicate is a sign of a polynomial evaluated with the coordinates of the input.

Examples:



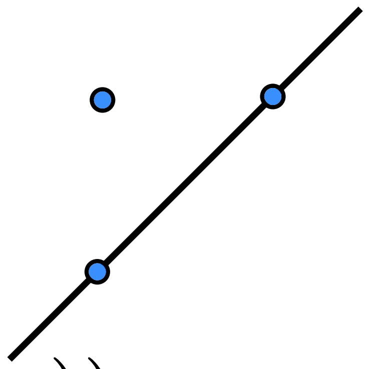
Orientation test (in convex hull):

$$orientation(p, q, r) = sign\left((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x)\right)$$

Geometric predicates

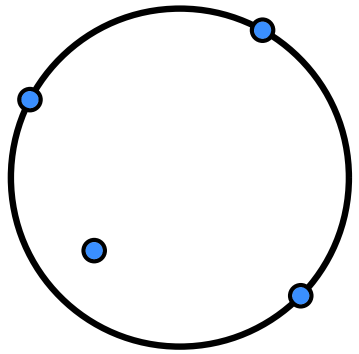
Definition: geometric predicate is a sign of a polynomial evaluated with the coordinates of the input.

Examples:



Orientation test (in convex hull):

$$orientation(p, q, r) = sign\left((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x)\right)$$



In-circle test (in Delaunay triangulation):

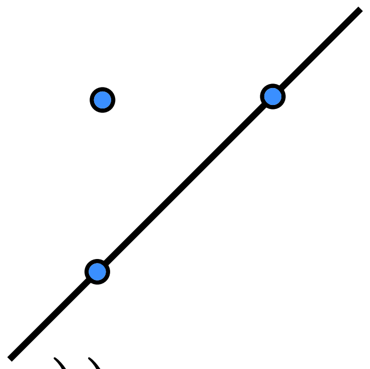
$$in\_circle(a, b, c, d) = \begin{cases} +1 & d \text{ lies inside circle through } a, b, c \\ -1 & d \text{ lies outside circle through } a, b, c \\ 0 & d \text{ lies on circle through } a, b, c \end{cases}$$

Geometric predicates

Definition: geometric predicate is a sign of a polynomial evaluated with the coordinates of the input.

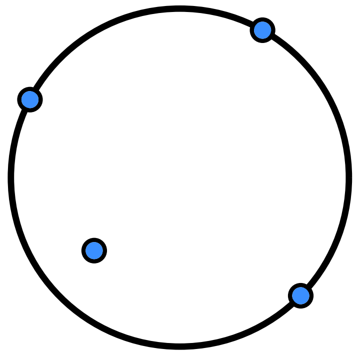
Examples:

Orientation test (in convex hull):



$$orientation(p, q, r) = sign\left((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x)\right)$$

In-circle test (in Delaunay triangulation):



$$in\_circle(a, b, c, d) = \begin{cases} +1 & d \text{ lies inside circle through } a, b, c \\ -1 & d \text{ lies outside circle through } a, b, c \\ 0 & d \text{ lies on circle through } a, b, c \end{cases}$$

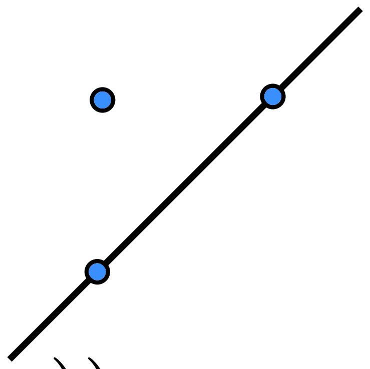
Question: other examples?

Geometric predicates

Definition: geometric predicate is a sign of a polynomial evaluated with the coordinates of the input.

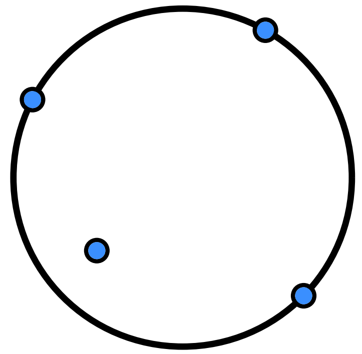
Examples:

Orientation test (in convex hull):



$$orientation(p, q, r) = sign\left((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x)\right)$$

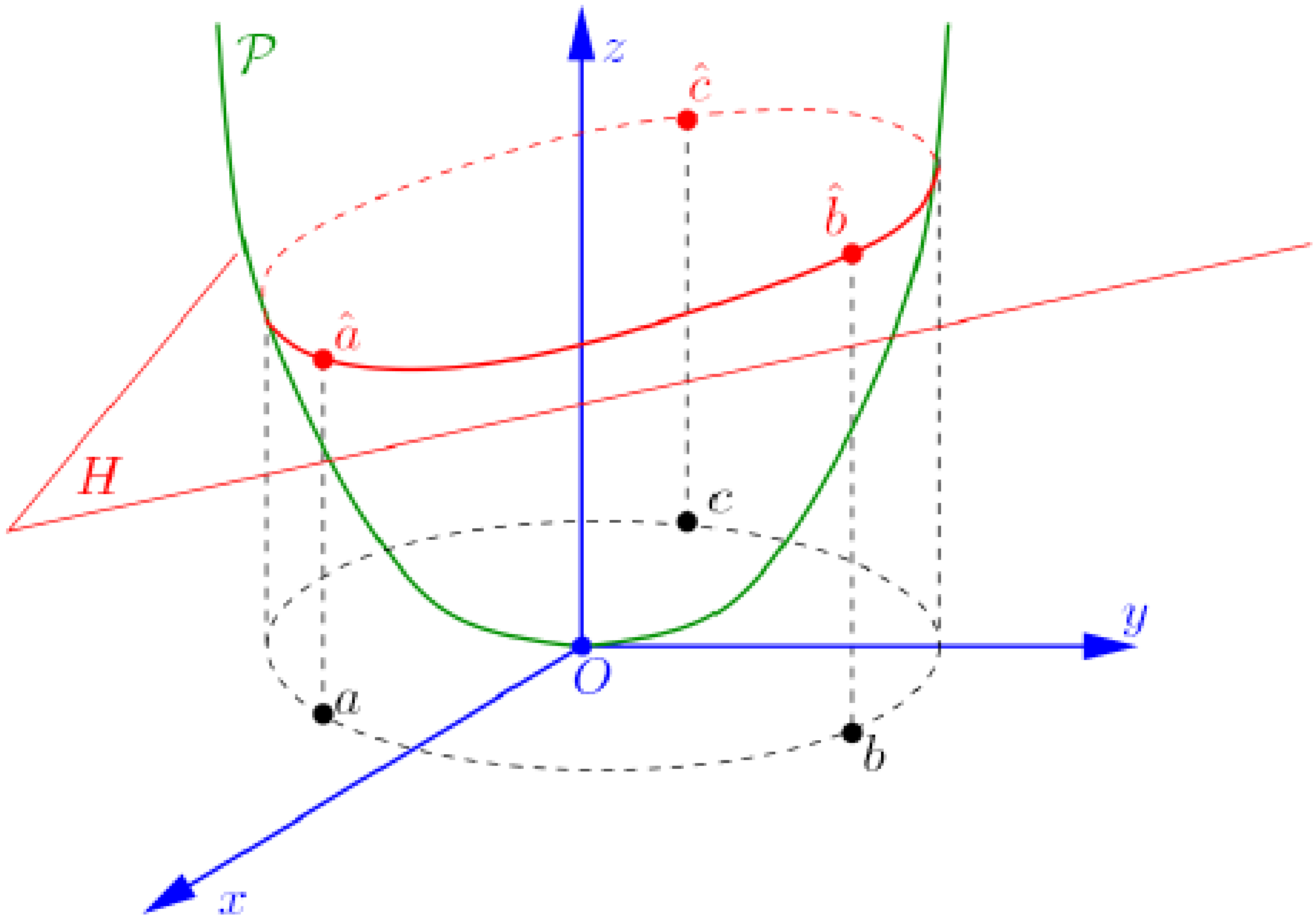
In-circle test (in Delaunay triangulation):



$$in\_circle(a, b, c, d) = \begin{cases} +1 & d \text{ lies inside circle through } a, b, c \\ -1 & d \text{ lies outside circle through } a, b, c \\ 0 & d \text{ lies on circle through } a, b, c \end{cases}$$

Question: other examples? Many predicates reduce to orientation test!

In-circle test



Observation: in-circle test = orientation test after lifting the point set

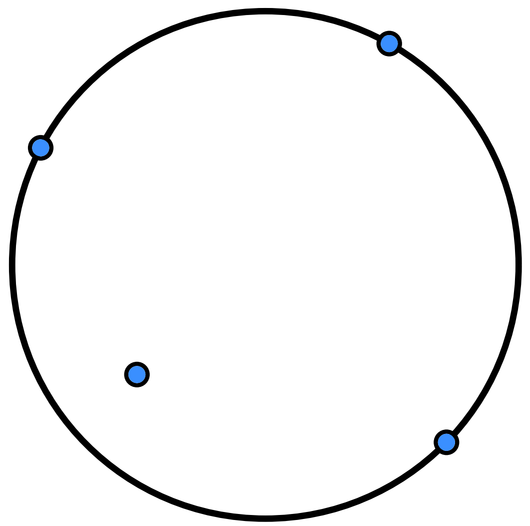
$$(a_x, a_y) \mapsto (a_x, a_y, a_x^2 + a_y^2)$$

In-circle test

$$in\_circle(a, b, c, d) =$$

$$= sign\left(\text{det}\begin{bmatrix} a_x & a_y & a_x^2 + a_y^2 & 1 \\ b_x & b_y & b_x^2 + b_y^2 & 1 \\ c_x & c_y & c_x^2 + c_y^2 & 1 \\ d_x & d_y & d_x^2 + d_y^2 & 1 \end{bmatrix}\right)$$

$$= sign\left(\text{det}\begin{bmatrix} a_x - d_x & a_y - d_y & a_z - d_z \\ b_x - d_x & b_y - d_y & b_z - d_z \\ c_x - d_x & c_y - d_y & c_z - d_z \end{bmatrix}\right)$$



Solutions

To avoid rounding errors:

Solutions

To avoid rounding errors:

- evaluate predicates exactly
- use predicate filtering

Solutions

To avoid rounding errors:

- evaluate predicates exactly (constructions do not have to be exact)
- use predicate filtering

Floating-point filters

Get correct sign (-1, 0 or 1) of an exact expression E using floating-point!

"filters out" the easy cases

1: Let $F = E(X)$ in **floating point**
2: **if** $F >$ error bound **then**
3: **return** 1
4: **else if** $-F >$ error bound **then**
5: **return** -1
6: **else**
7: increase precision and repeat, or switch to exact arithmetic

Floating-point filters

Get correct sign (-1, 0 or 1) of an exact expression E using floating-point!

"filters out" the easy cases

```
Let F = E(X) in floating point
if F > error bound then
    return 1
else if -F > error bound then
    return -1
else
    increase precision and repeat, or
    switch to exact arithmetic
```

If the correct result is 0, must go to exact phase

Floating-point filters

Get correct sign (-1, 0 or 1) of an exact expression E using floating-point!

“filters out” the easy cases

Let $F = E(X)$ in **floating point**
if $F >$ error bound **then**
    return 1
else if $-F >$ error bound **then**
    return -1
else
    increase precision and repeat, or switch to exact arithmetic

If the correct result is 0, must go to exact phase

Question: how exactly do we evaluate predicates exactly?

Floating-point filters

Get correct sign (-1, 0 or 1) of an exact expression E using floating-point!

“filters out” the easy cases

Let $F = E(X)$ in **floating point**
if $F >$ error bound **then**
    return 1
else if $-F >$ error bound **then**
    return -1
else
    increase precision and repeat, or
    switch to exact arithmetic

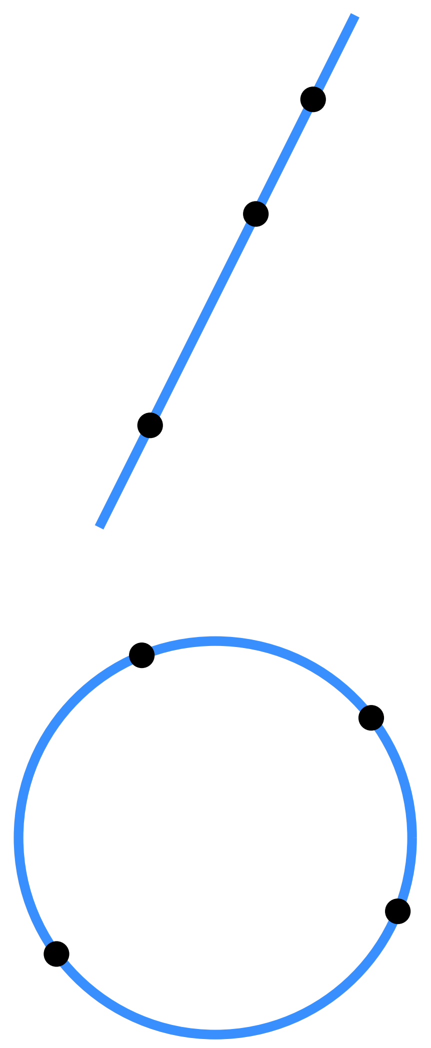
If the correct result is 0, must go to exact phase

Question: how exactly do we evaluate predicates exactly?

use exact arithmetic $\Rightarrow$ do not limit space to store numbers

Dealing with degeneracies

Option 1: carefully design your algorithm around degenerate cases

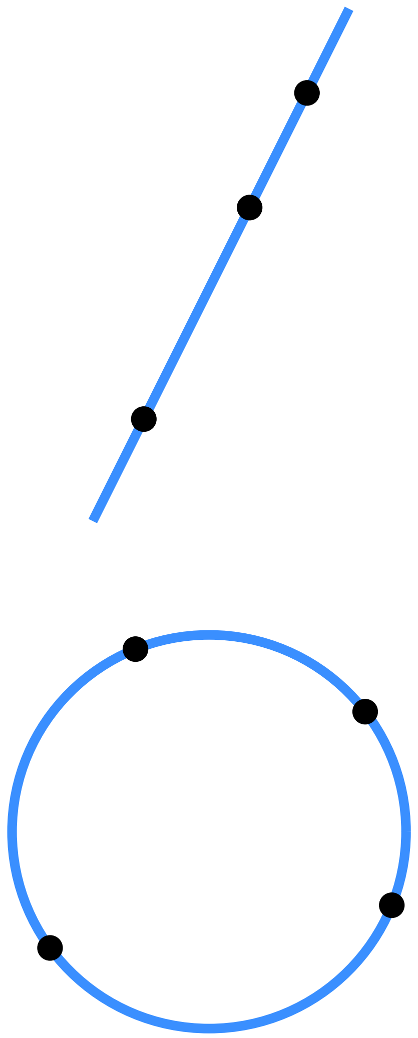


Dealing with degeneracies

Option 1: carefully design your algorithm around degenerate cases

Option 2: randomly perturb your input and solve your problem approximately

- input is precise
- perturbation is “close” to the input
- geometric traits are preserved



Robustness wrap-up

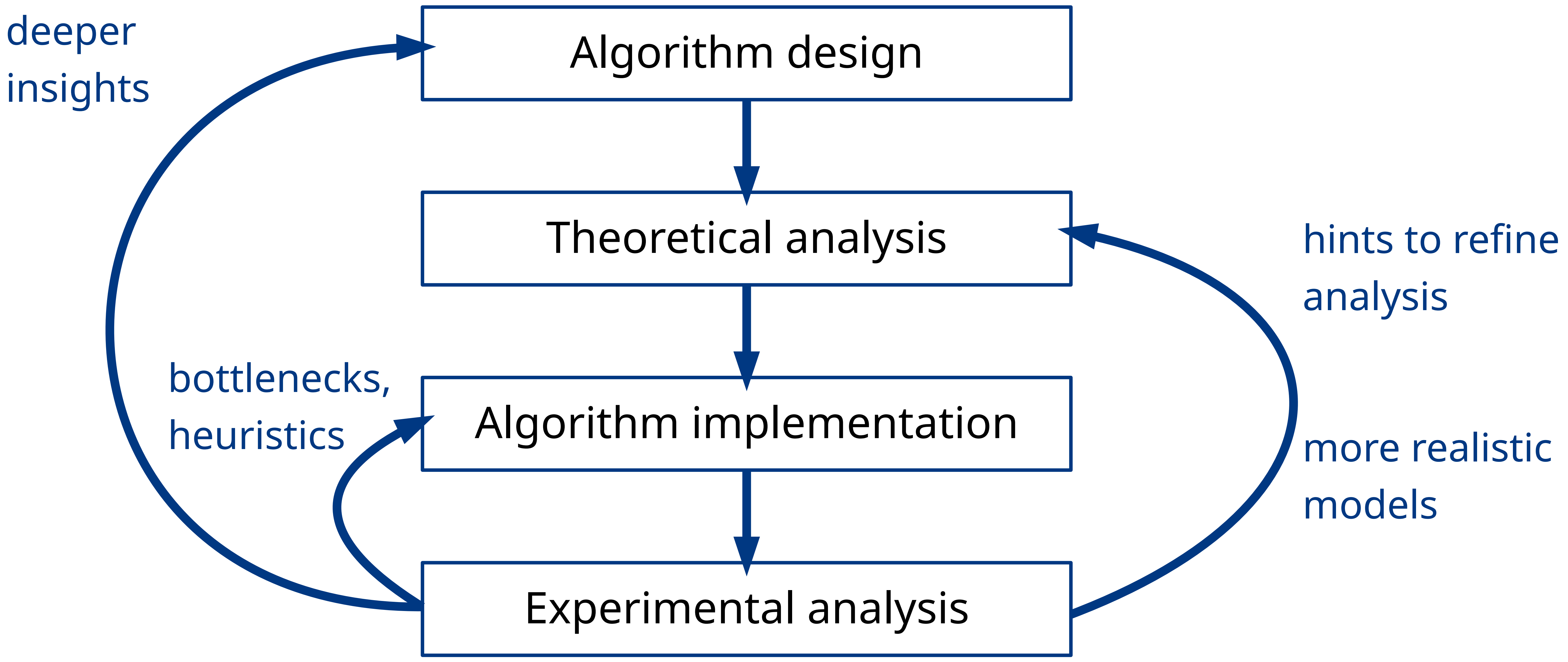
To make a robust implementation of your algorithm:

- trade-off: exact arithmetic vs speed
- often sufficient: answer predicates exactly, but construction may be inexact
- robustness is difficult to achieve
- good news: robust implementations exist

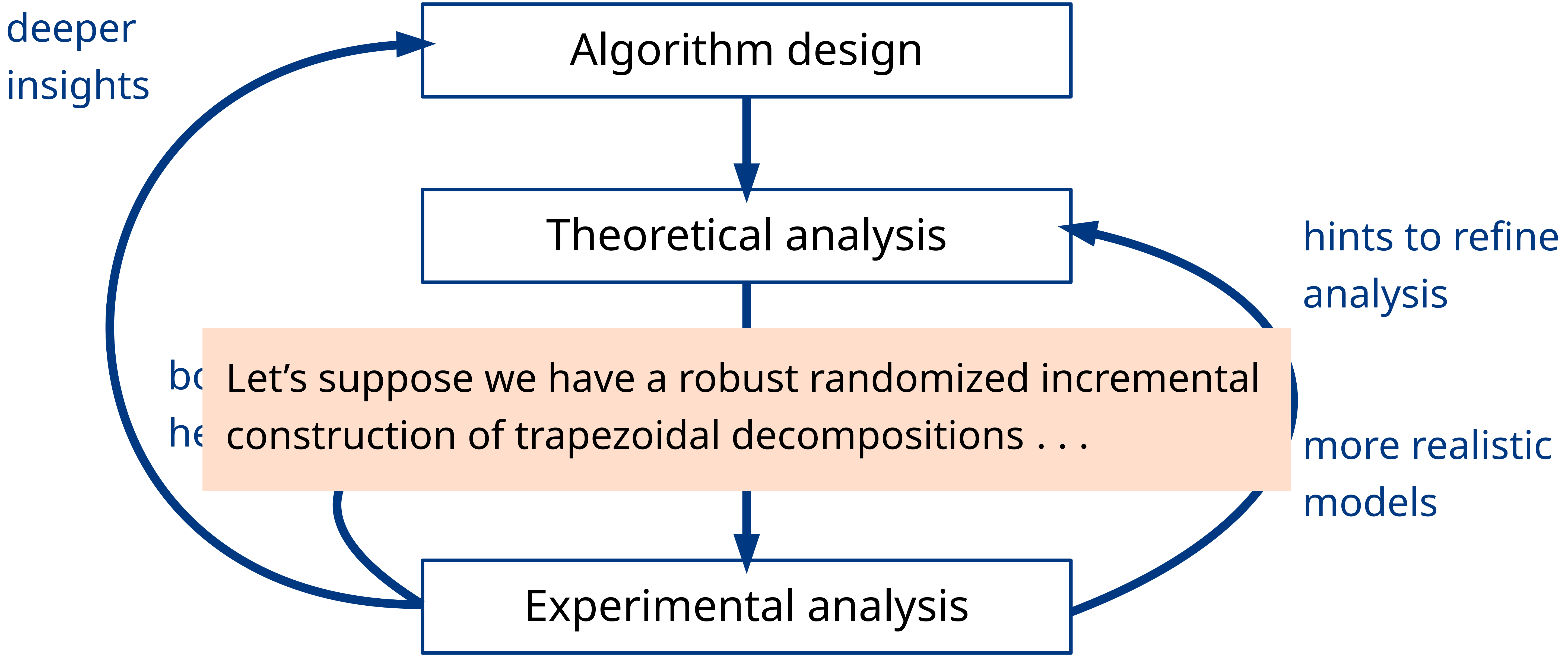
CGAL
Computational Geometry Algorithms Library

Triangle
A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator.

Algorithm engineering cycle



Algorithm engineering cycle

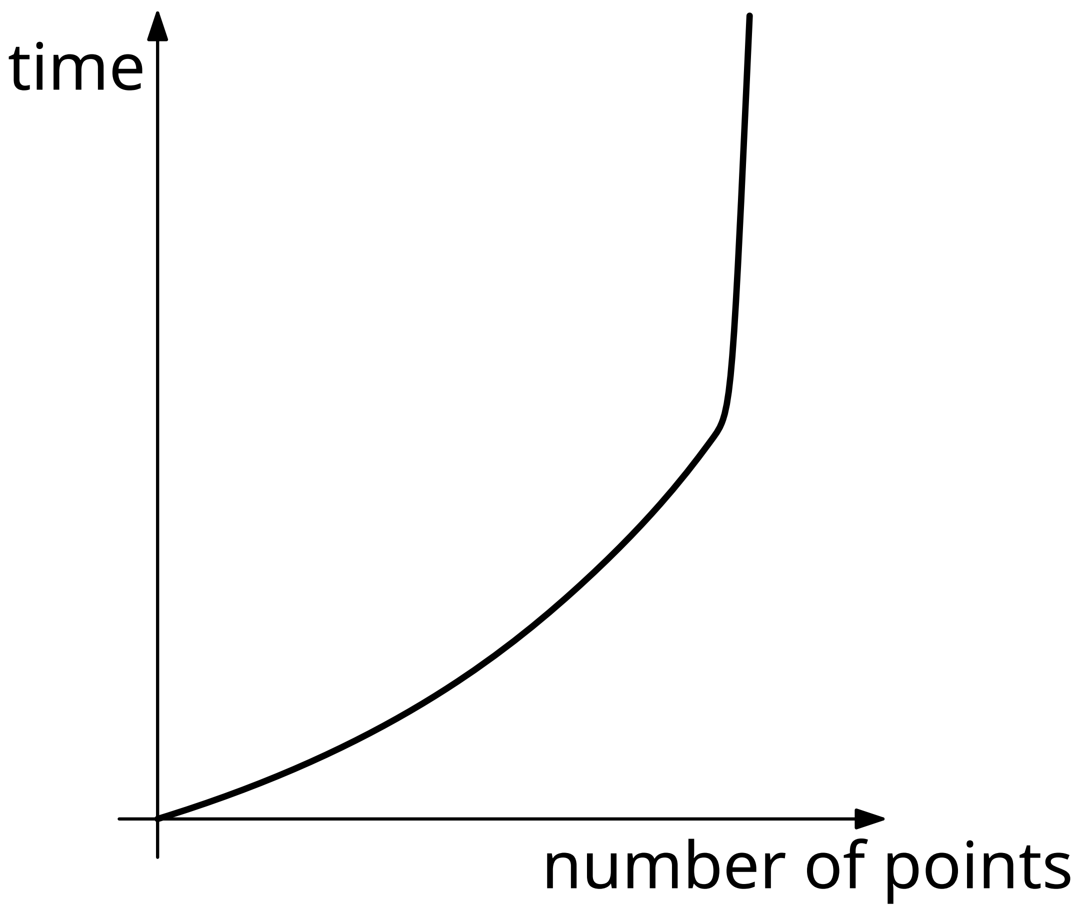


Let’s suppose we have a robust randomized incremental construction of trapezoidal decompositions . . .

Standard RIC in experiments

Experimental results of randomized incremental construction (RIC) of trapezoidal decomposition:

- What went wrong?

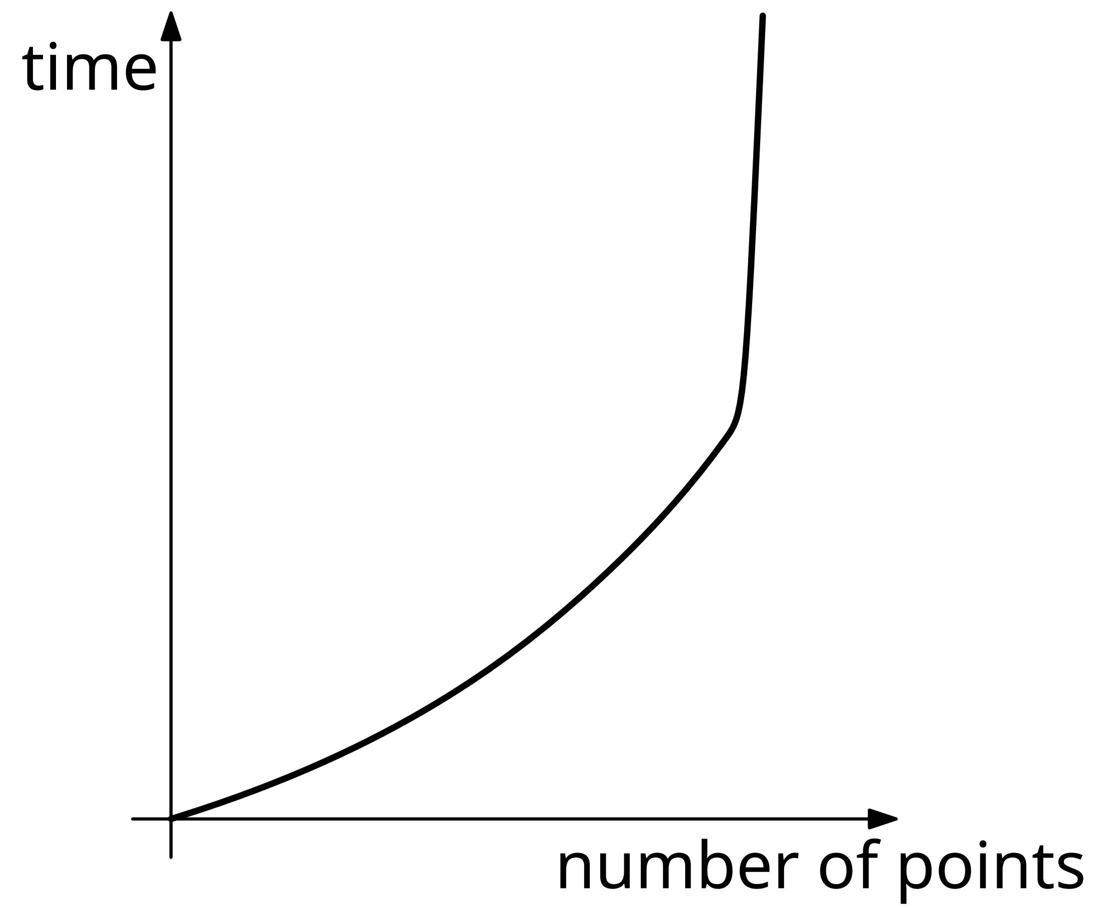


* refer to [Choi, Amenta ’02]

Standard RIC in experiments

Experimental results of randomized incremental construction (RIC) of trapezoidal decomposition:

- What went wrong?
- Thrashing due to random memory access

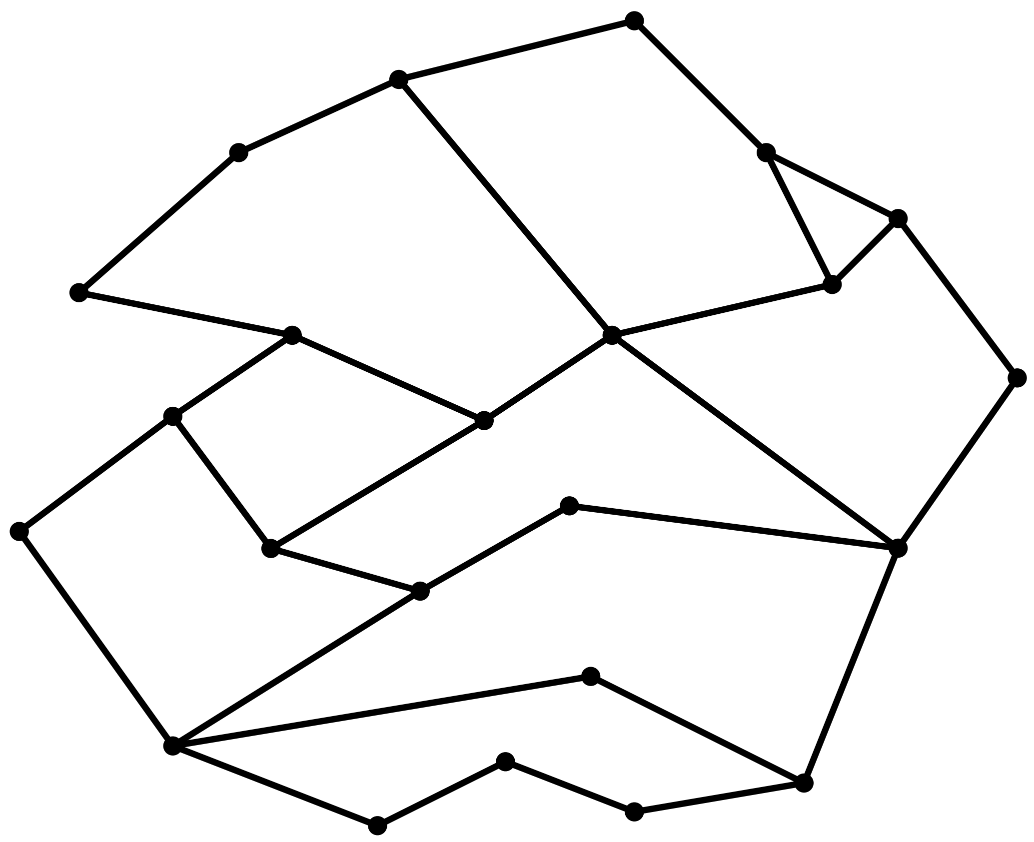


* refer to [Choi, Amenta ’02]

Partial randomization

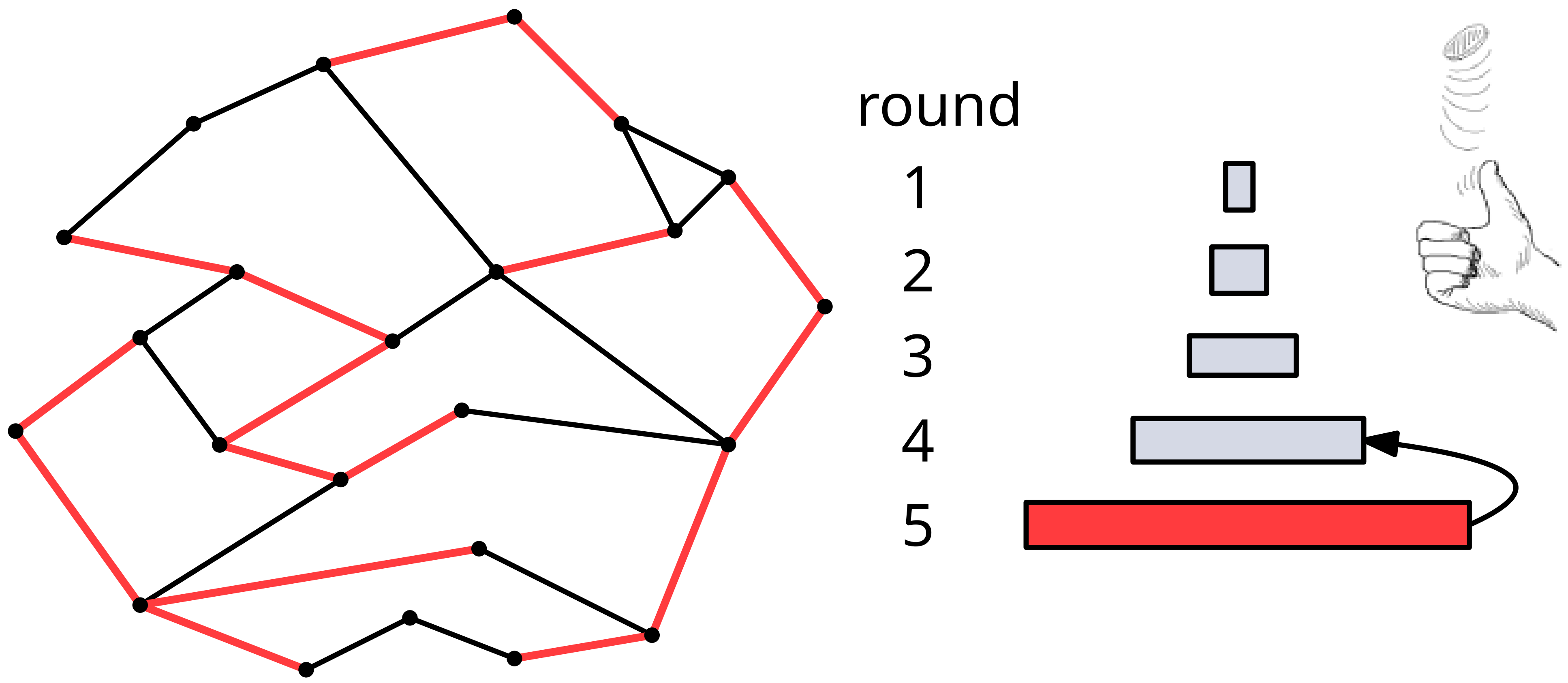
- random access to large data: a bad idea
- don’t randomize? really bad in theory and also causes overhead in experiments
- partially randomized insertion order
  - increase locality of reference, especially as data structure gets large
  - retain enough randomness to guarantee optimality

Partial randomization



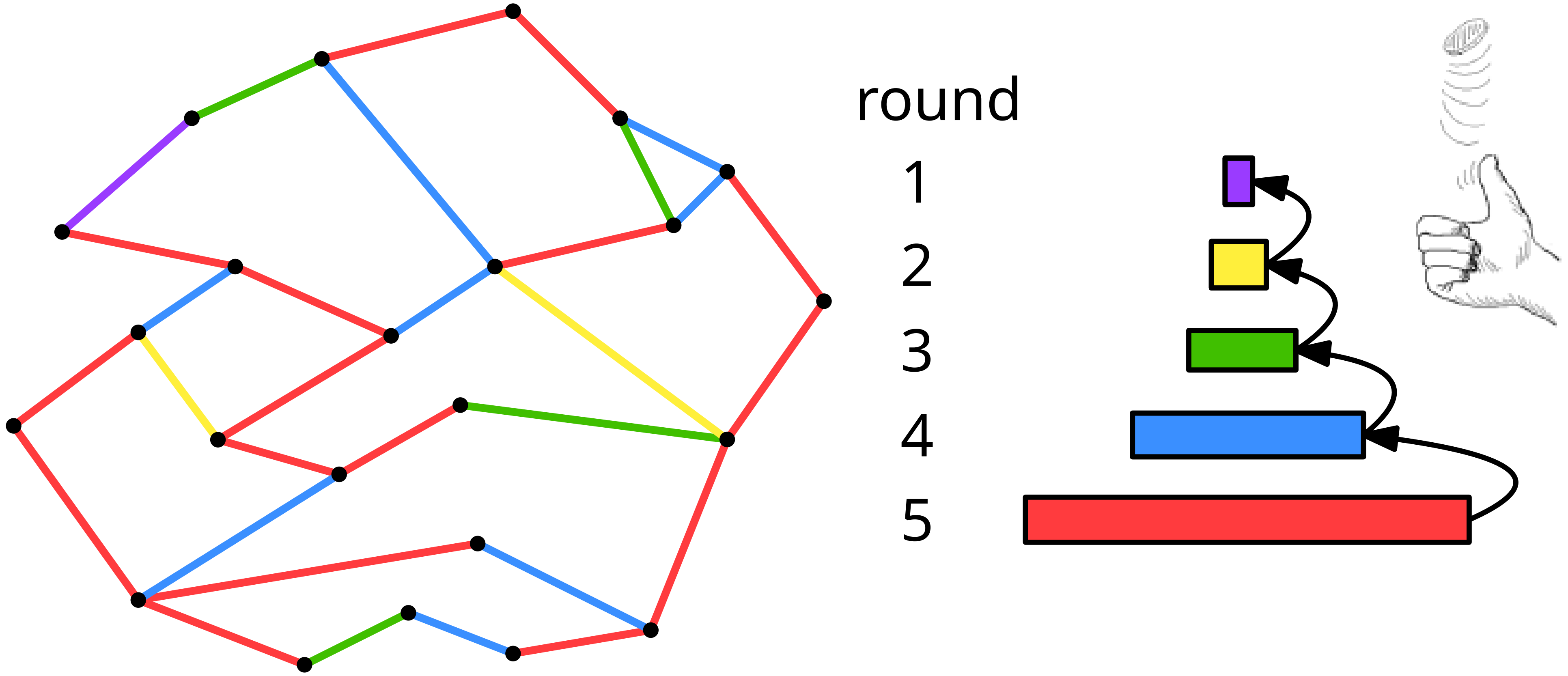
- in each round (from last to first) choose each segment with probability $1/2$
- order chosen segments in each round to benefit locality (not random!)

Partial randomization



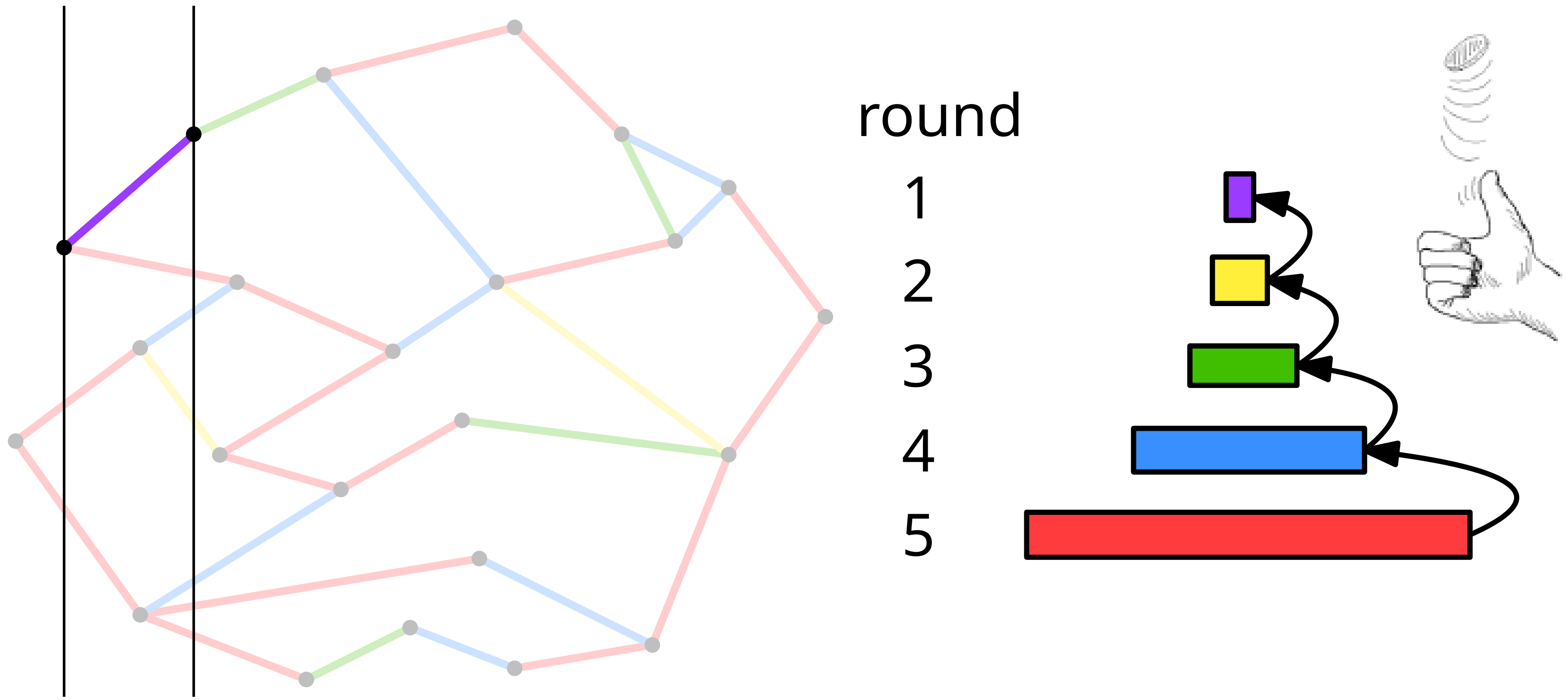
- in each round (from last to first) choose each segment with probability $1/2$
- order chosen segments in each round to benefit locality (not random!)

Partial randomization



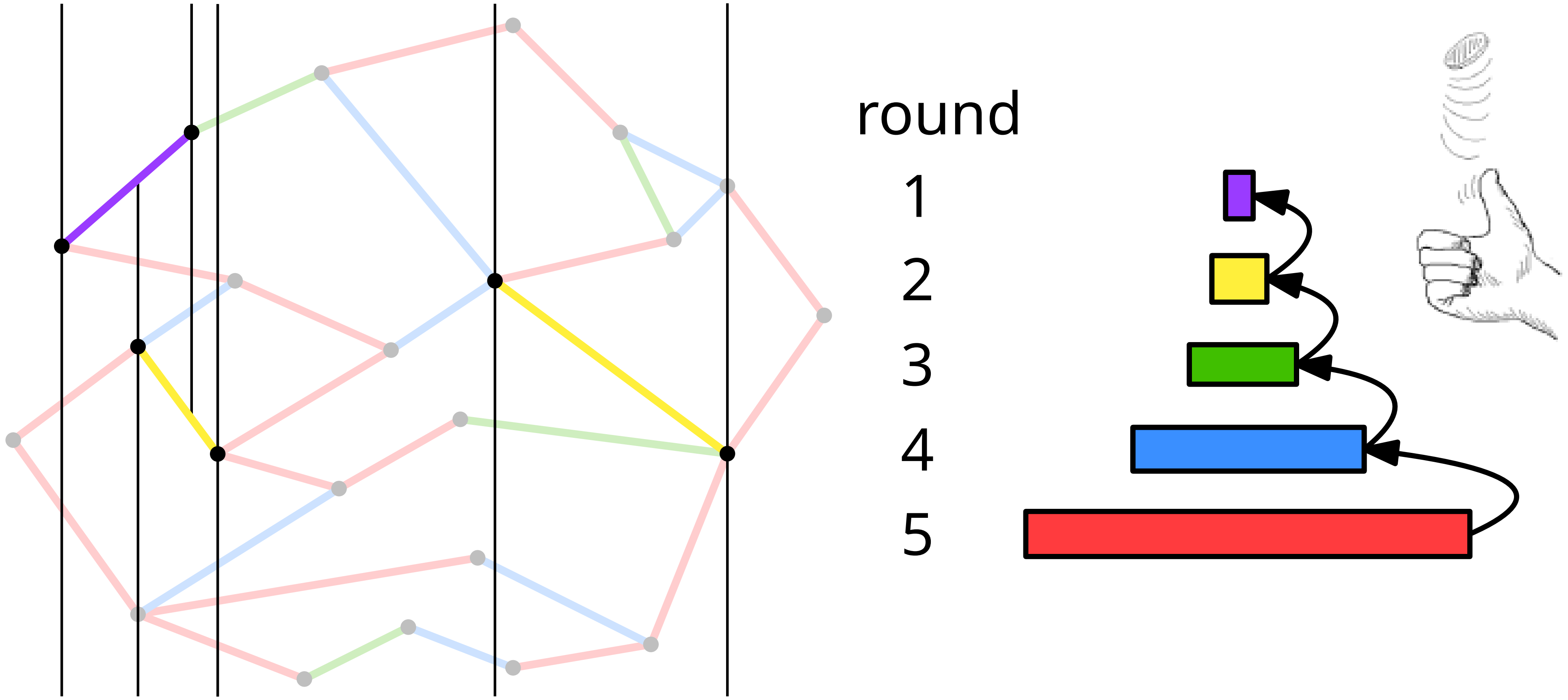
- in each round (from last to first) choose each segment with probability $1/2$
- order chosen segments in each round to benefit locality (not random!)

Partial randomization



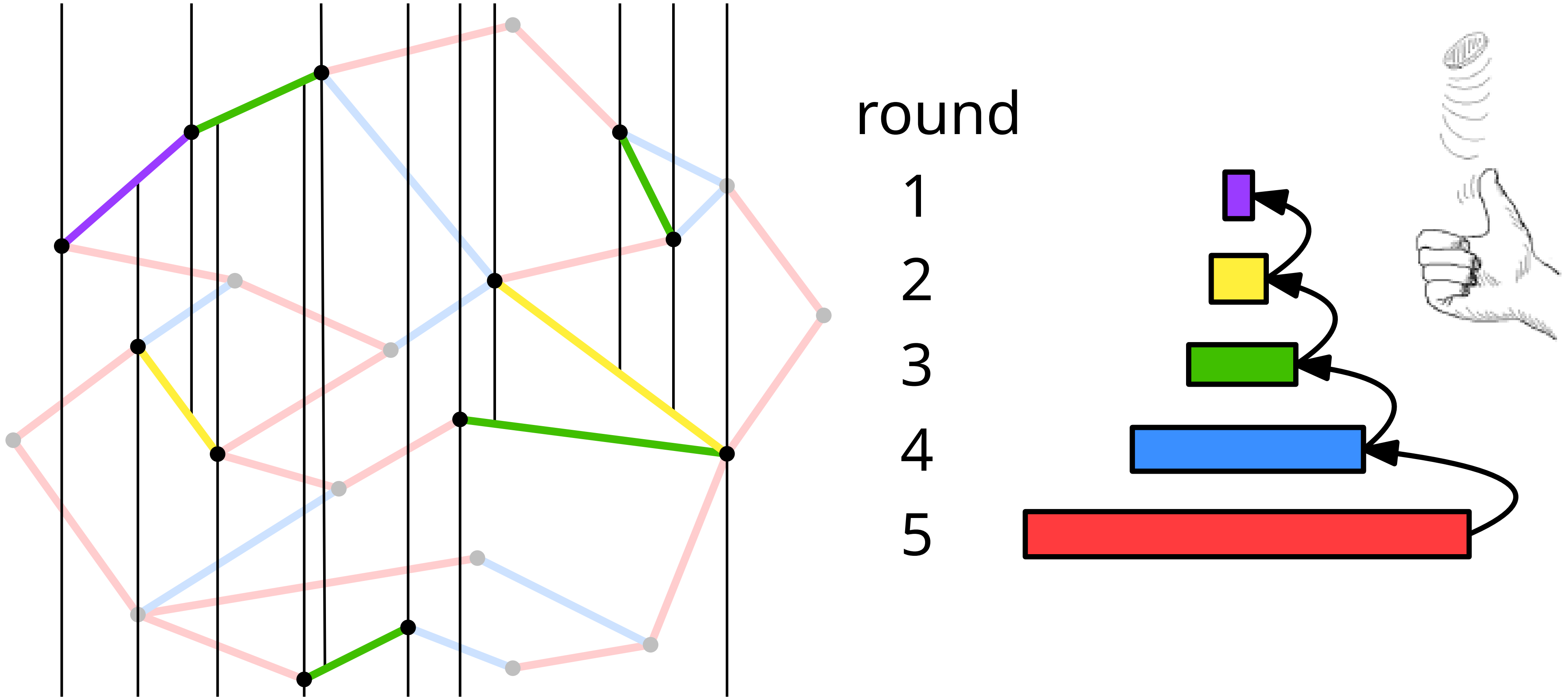
- in each round (from last to first) choose each segment with probability $1/2$
- order chosen segments in each round to benefit locality (not random!)

Partial randomization



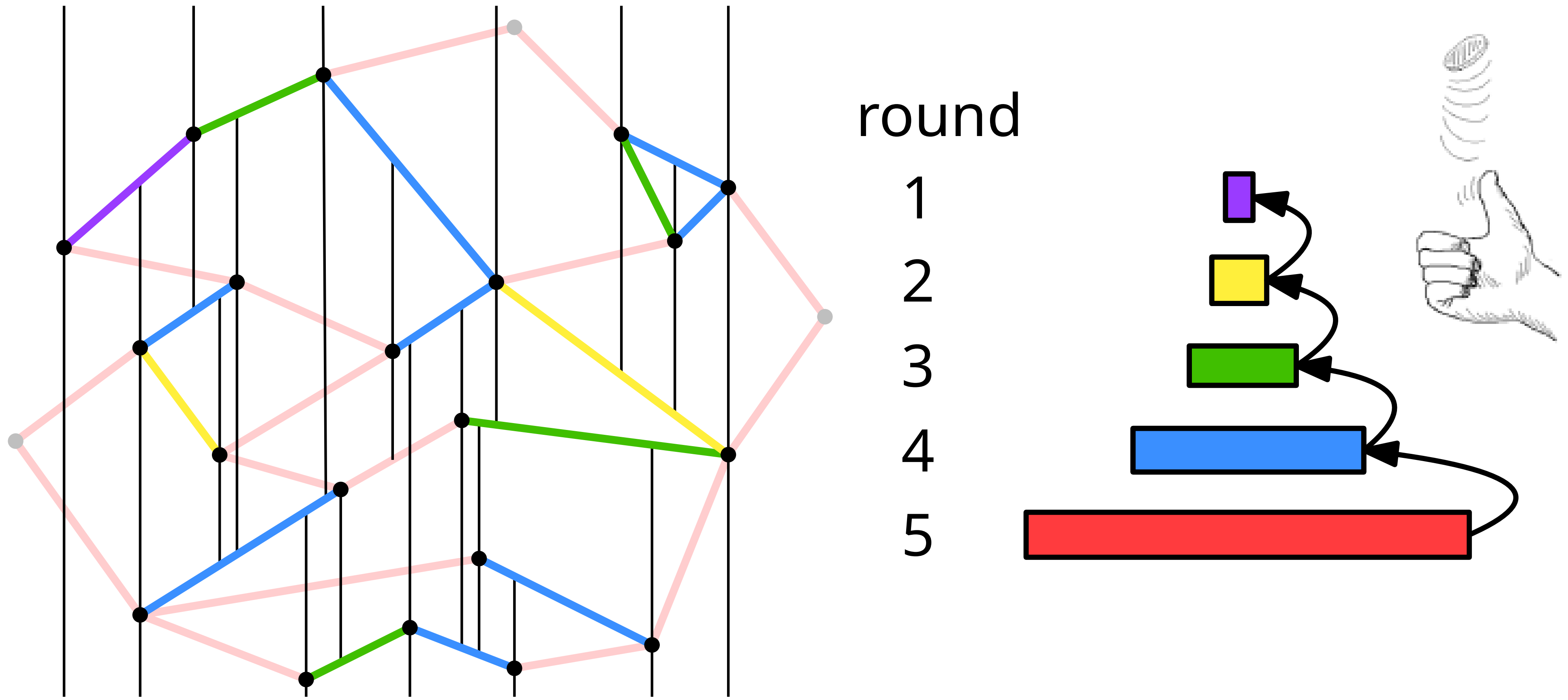
- in each round (from last to first) choose each segment with probability $1/2$
- order chosen segments in each round to benefit locality (not random!)

Partial randomization



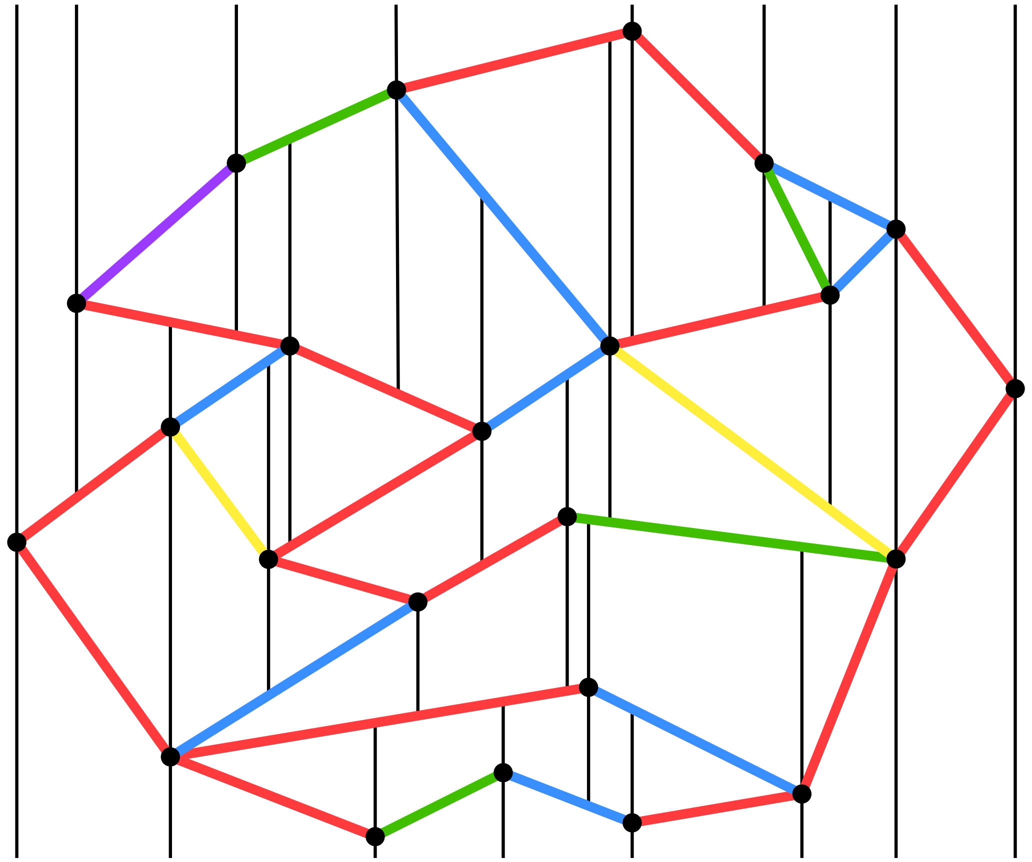
- in each round (from last to first) choose each segment with probability $1/2$
- order chosen segments in each round to benefit locality (not random!)

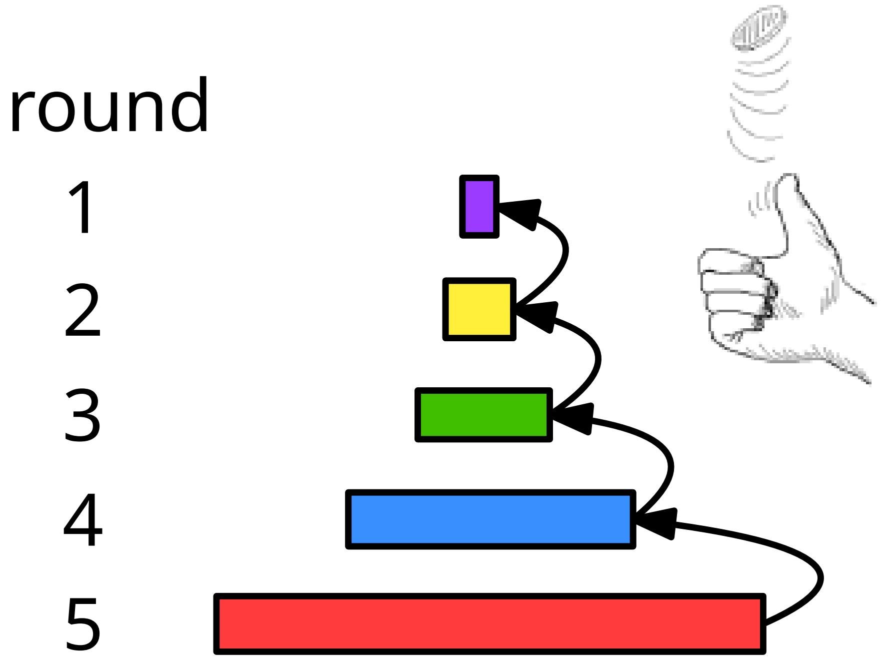
Partial randomization



- in each round (from last to first) choose each segment with probability $1/2$
- order chosen segments in each round to benefit locality (not random!)

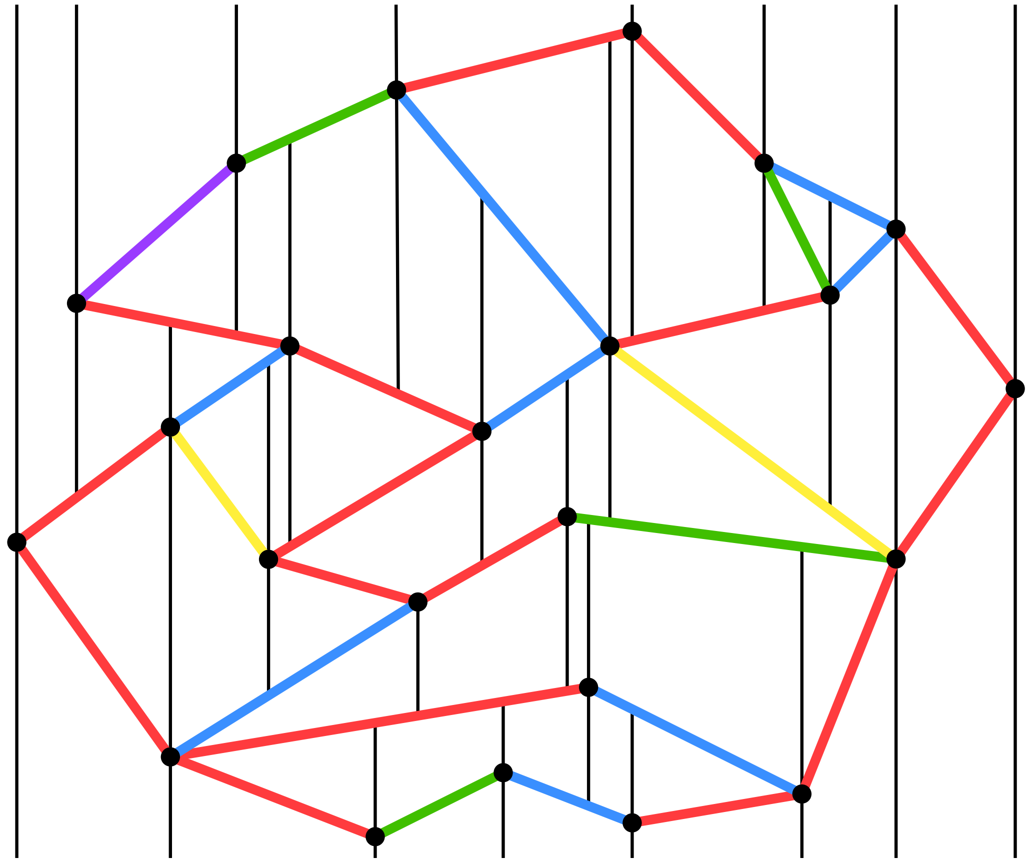
Partial randomization

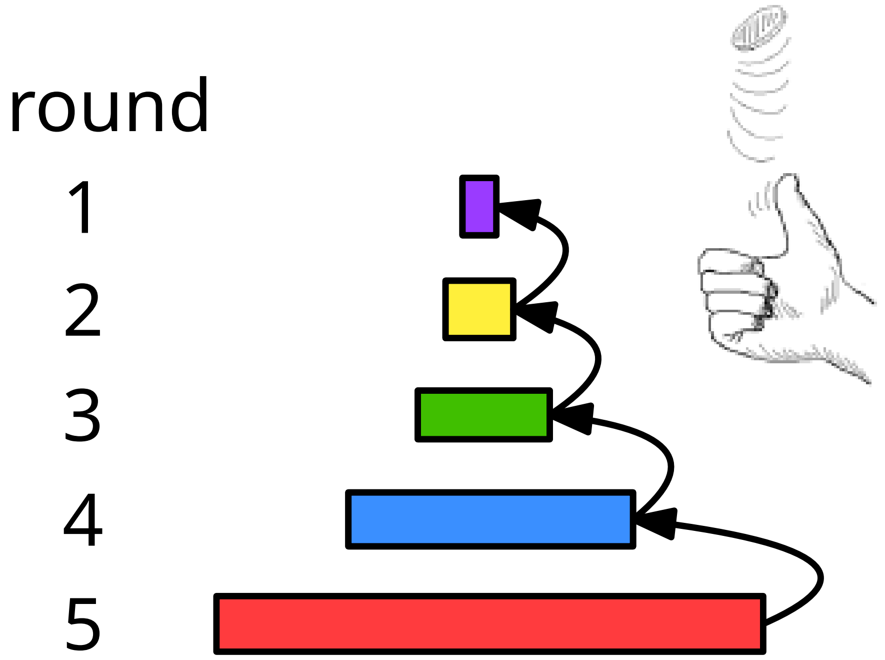




- in each round (from last to first) choose each segment with probability $1/2$
- order chosen segments in each round to benefit locality (not random!)

Partial randomization





Theorem: The trapezoidal decomposition of n non-intersecting segments in the plane can be constructed in $O(n \log n)$ expected time using partial randomization.

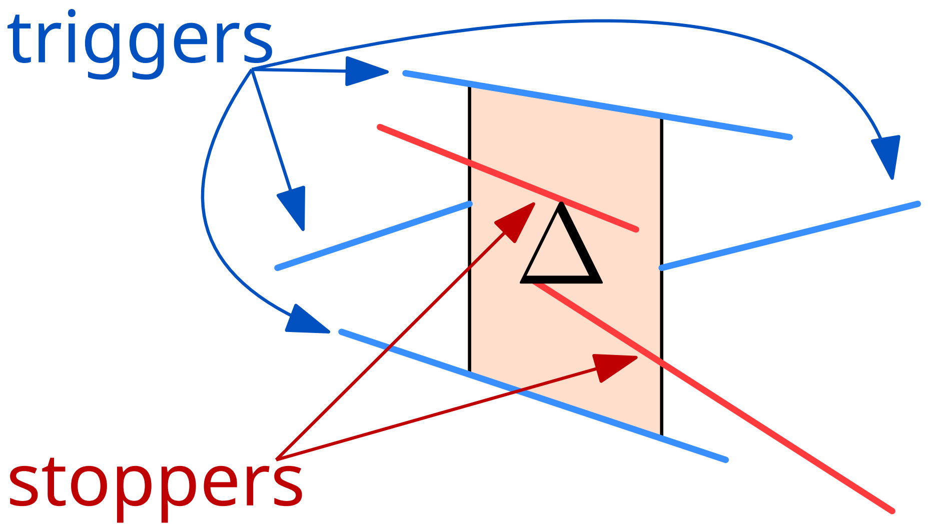
Proof idea: compare probabilities to standard randomized algorithm

Partial randomization

Lemma: For a given trapezoid Δ the probability p_{PRIC} of occurring in a partial RIC is at most 16 times the probability p_{RIC} of occurring in a (standard) RIC.

Partial randomization

Lemma: For a given trapezoid Δ the probability p_{PRIC} of occurring in a partial RIC is at most 16 times the probability p_{RIC} of occurring in a (standard) RIC.

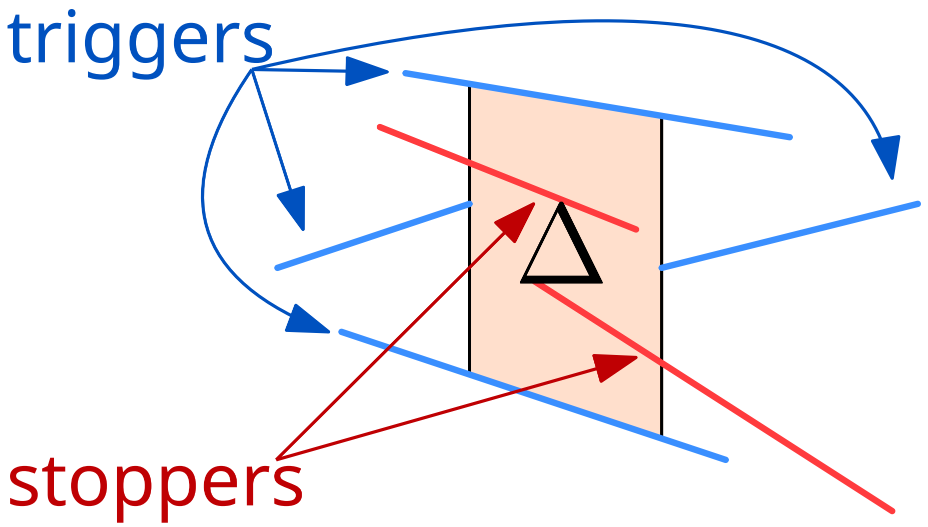


Partial randomization

Lemma: For a given trapezoid Δ the probability p_{PRIC} of occurring in a partial RIC is at most 16 times the probability p_{RIC} of occurring in a (standard) RIC.

$$p_{PRIC} \le P[\ \boxed{1\,2\,\ldots\,i\text{-}1\ i\ i{+}1\,\ldots\,k}\ ]$$

triggers stoppers

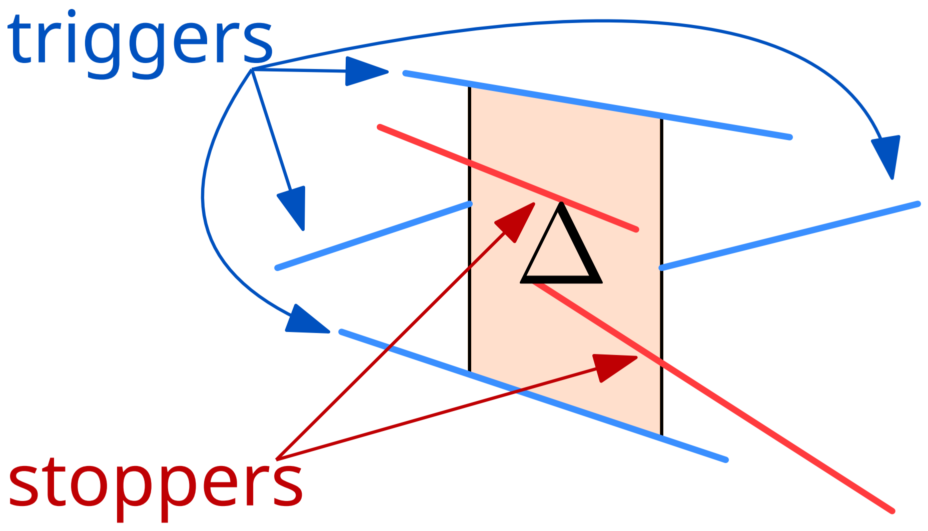


Partial randomization

Lemma: For a given trapezoid Δ the probability p_{PRIC} of occurring in a partial RIC is at most 16 times the probability p_{RIC} of occurring in a (standard) RIC.

$$p_{PRIC} \leq P[\ \boxed{1\ 2\ \ldots\ i\text{-}1\ i\ i\text{+}1\ \ldots\ k}\ ]$$

triggers stoppers



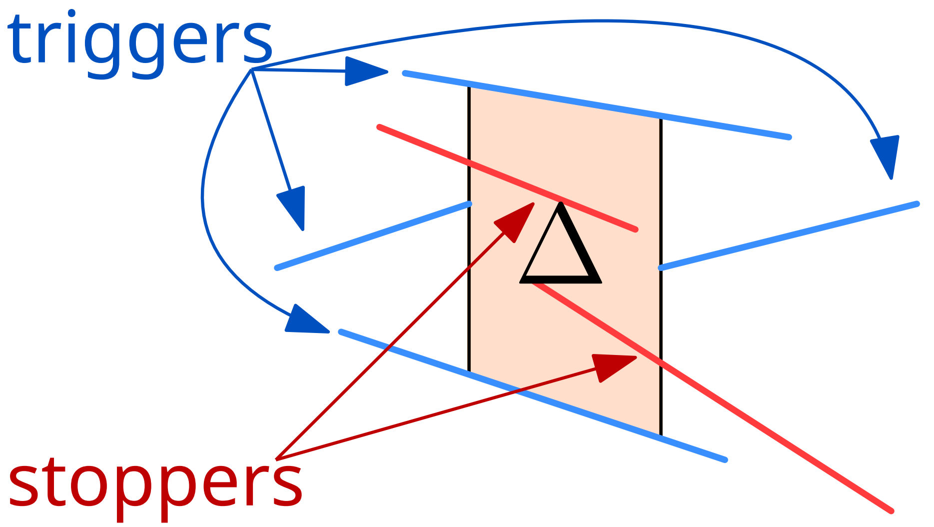
$T_i =$ All triggers of Δ appear in the i^{th} round or before

$S_i =$ The first stopper of Δ appears in the i^{th} round

Partial randomization

Lemma: For a given trapezoid Δ the probability p_{PRIC} of occurring in a partial RIC is at most 16 times the probability p_{RIC} of occurring in a (standard) RIC.

$$p_{PRIC} \leq P[\ \boxed{\underbrace{1\ 2\ \ldots\ i\text{-}1}_{\text{triggers}}\ i\ \underbrace{i{+}1\ \ldots\ k}_{\text{stoppers}}}\ ] = P[\bigcup_{i=1}^{k}(S_i \cap T_i)]$$



$T_i =$ All triggers of Δ appear in the i^{th} round or before

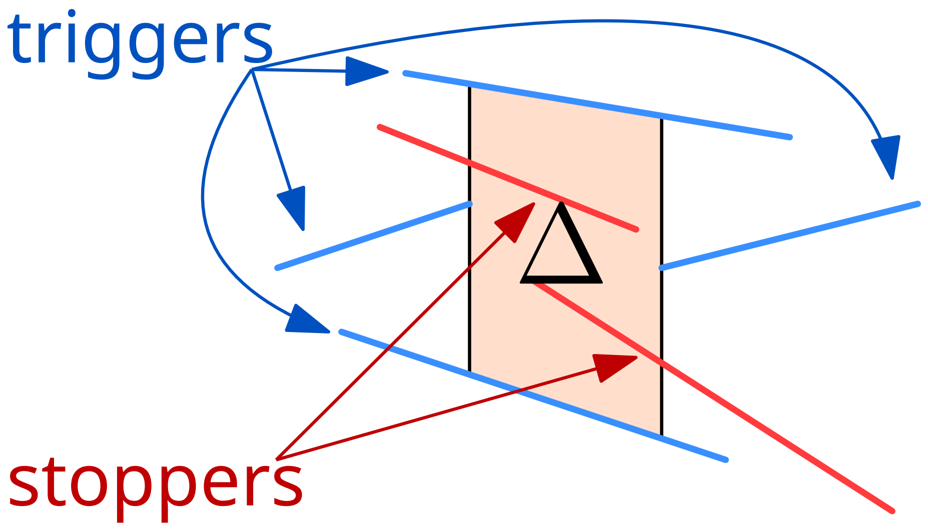
$S_i =$ The first stopper of Δ appears in the i^{th} round

Partial randomization

Lemma: For a given trapezoid Δ the probability p_{PRIC} of occurring in a partial RIC is at most 16 times the probability p_{RIC} of occurring in a (standard) RIC.

$$p_{PRIC} \leq P[\ \boxed{1\ 2\ \ldots\ i\text{-}1\ i\ i\text{+}1\ \ldots\ k}\ ] = P[\bigcup_{i=1}^{k}(S_i \cap T_i)]$$

triggers stoppers



$T_i =$ All triggers of Δ appear in the i^{th} round or before

$S_i =$ The first stopper of Δ appears in the i^{th} round

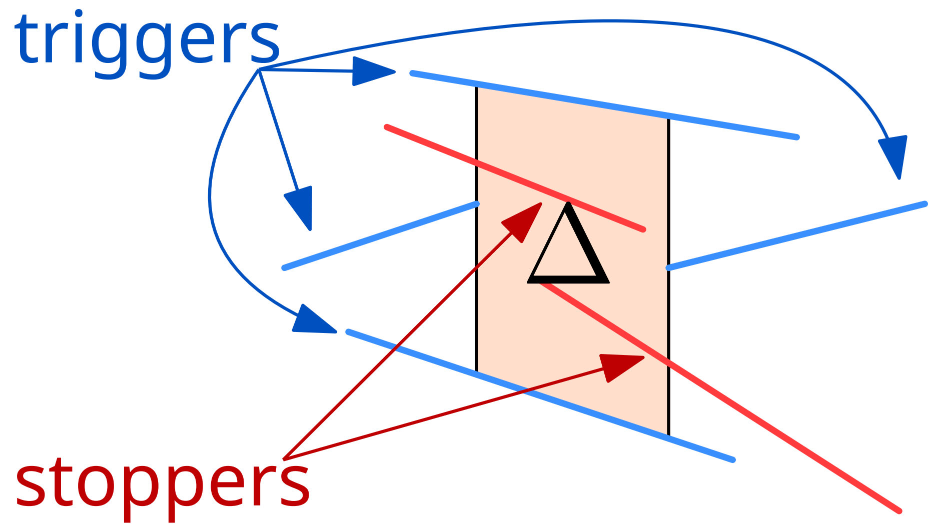
$$= \sum_{i=1}^{k} P[S_i \cap T_i]$$

Partial randomization

Lemma: For a given trapezoid Δ the probability p_{PRIC} of occurring in a partial RIC is at most 16 times the probability p_{RIC} of occurring in a (standard) RIC.

$$p_{PRIC} \leq P[\ \boxed{1\ 2\ \ldots\ i\text{-}1\ i\ i{+}1\ \ldots\ k}\ ] = P[\bigcup_{i=1}^{k}(S_i \cap T_i)]$$

triggers stoppers



$T_i =$ All triggers of Δ appear in the i^{th} round or before

$S_i =$ The first stopper of Δ appears in the i^{th} round

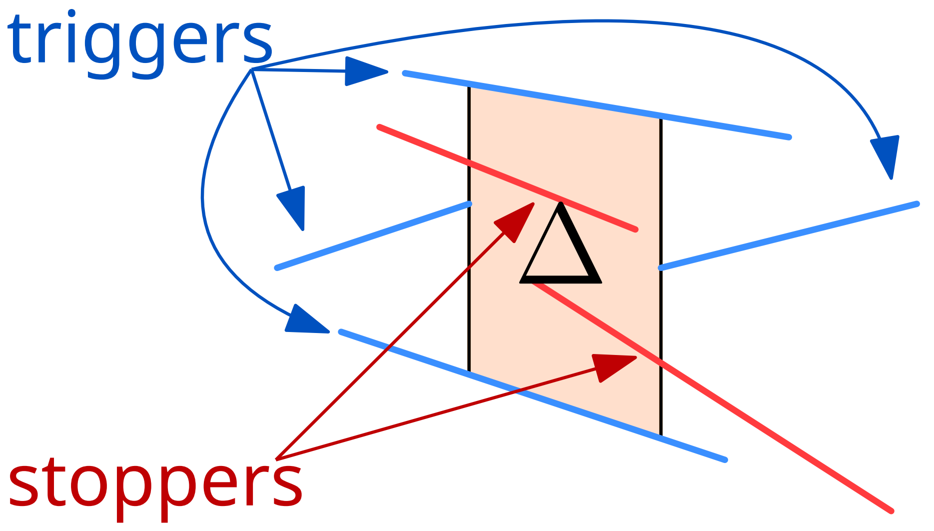
$$= \sum_{i=1}^{k} P[S_i \cap T_i]$$

$\{S_i \cap T_i\}$ are disjoint

Partial randomization

Lemma: For a given trapezoid Δ the probability p_{PRIC} of occurring in a partial RIC is at most 16 times the probability p_{RIC} of occurring in a (standard) RIC.

$$p_{PRIC} \leq P[\ \boxed{\underbrace{1\,2\ \ldots\ i\text{-}1}_{\text{triggers}}\ i\ \underbrace{i\text{+}1\ \ldots\ k}_{\text{stoppers}}}\ ] = P[\bigcup_{i=1}^{k}(S_i \cap T_i)]$$



$T_i =$ All triggers of Δ appear in the i^{th} round or before

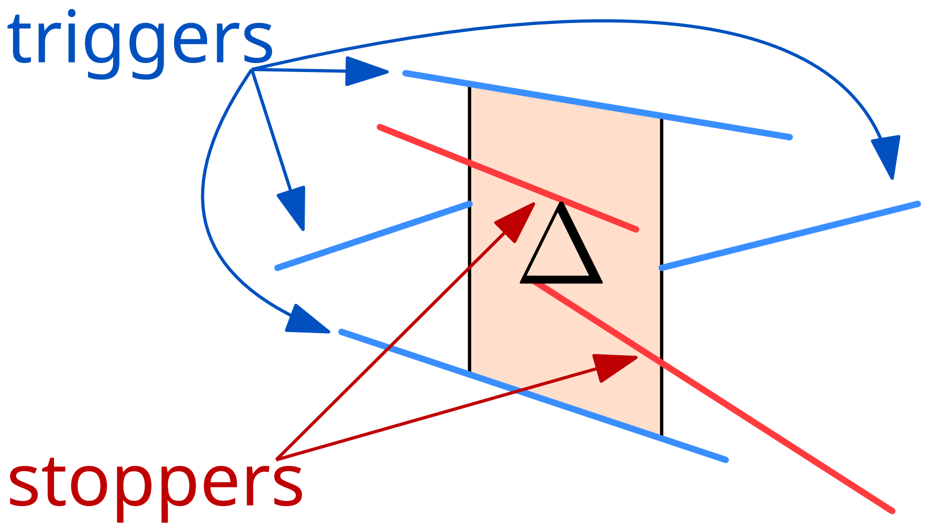
$S_i =$ The first stopper of Δ appears in the i^{th} round

$$= \sum_{i=1}^{k} P[S_i \cap T_i] = \sum_{i=1}^{k} P[S_i] \cdot P[T_i]$$

$\{S_i \cap T_i\}$ are disjoint

Partial randomization

Lemma: For a given trapezoid Δ the probability p_{PRIC} of occurring in a partial RIC is at most 16 times the probability p_{RIC} of occurring in a (standard) RIC.



$$p_{PRIC} \leq P[\ \boxed{\underbrace{1\;2\;\ldots\;i\text{-}1\;i}_{\text{triggers}}\;\underbrace{i{+}1\;\ldots\;k}_{\text{stoppers}}}\ ] = P[\bigcup_{i=1}^{k} (S_i \cap T_i)]$$

$T_i =$ All triggers of Δ appear in the i^{th} round or before

$S_i =$ The first stopper of Δ appears in the i^{th} round

$$= \sum_{i=1}^{k} P[S_i \cap T_i] = \sum_{i=1}^{k} P[S_i] \cdot P[T_i]$$

$\{S_i \cap T_i\}$ are disjoint

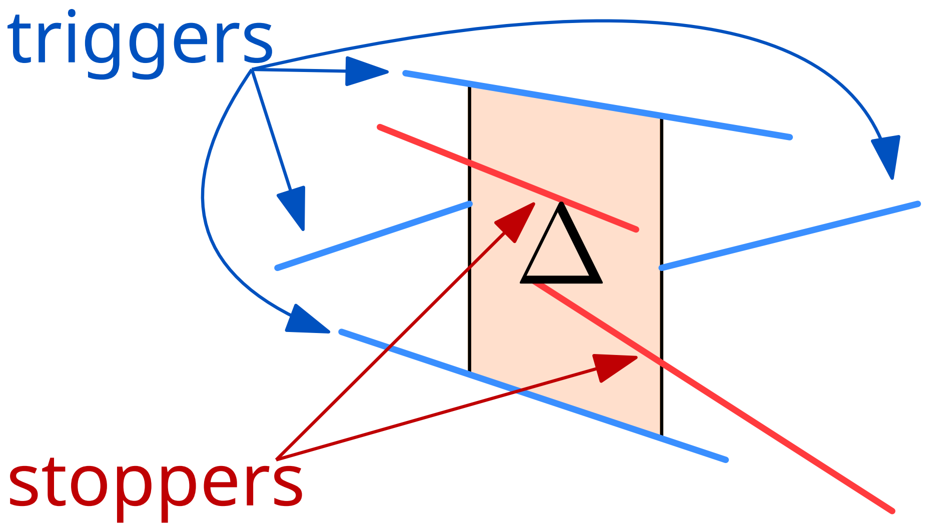
S_i and T_i are independent

Partial randomization

Lemma: For a given trapezoid Δ the probability p_{PRIC} of occurring in a partial RIC is at most 16 times the probability p_{RIC} of occurring in a (standard) RIC.

$$p_{PRIC} \leq P[\ \boxed{1\ 2\ \ldots\ i\text{-}1\ i\ i\text{+}1\ \ldots\ k}\ ] = P[\bigcup_{i=1}^{k} (S_i \cap T_i)]$$

triggers stoppers



$T_i =$ All triggers of Δ appear in the i^{th} round or before

$S_i =$ The first stopper of Δ appears in the i^{th} round

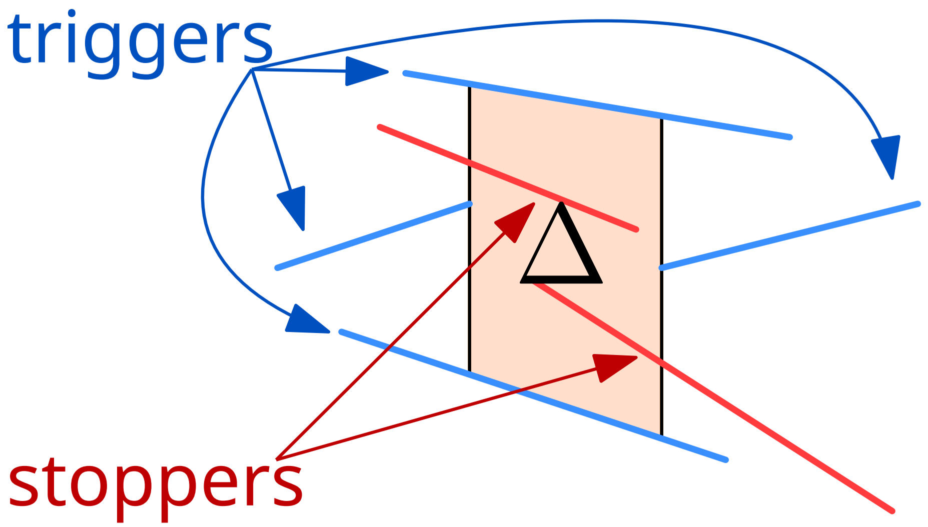
$$= \sum_{i=1}^{k} P[S_i \cap T_i] = \sum_{i=1}^{k} P[S_i] \cdot P[T_i]$$

Partial randomization

Lemma: For a given trapezoid Δ the probability p_{PRIC} of occurring in a partial RIC is at most 16 times the probability p_{RIC} of occurring in a (standard) RIC.

$$p_{PRIC} \leq P[\ \boxed{1\ 2\ \ldots\ i\text{-}1\ i\ i\text{+}1\ \ldots\ k}\ ] = P[\bigcup_{i=1}^{k}(S_i \cap T_i)]$$

triggers stoppers



$T_i =$ All triggers of Δ appear in the i^{th} round or before

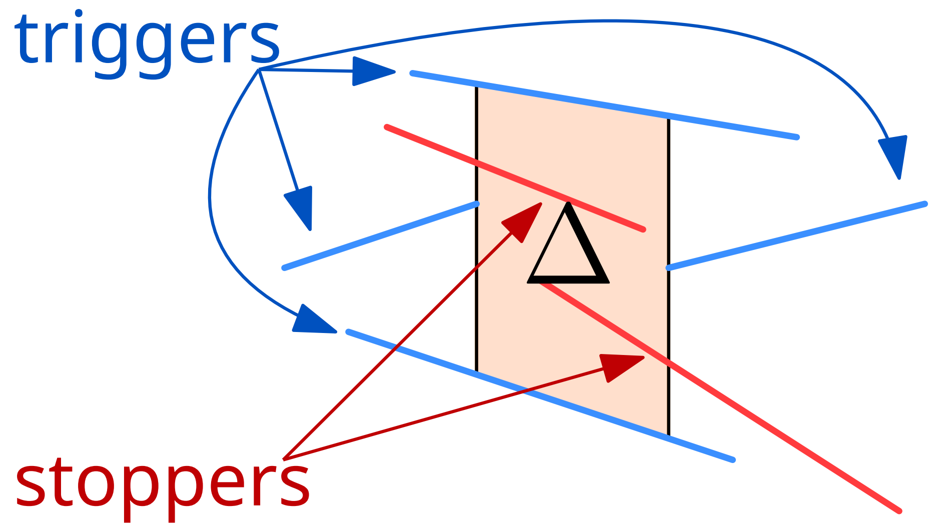
$S_i =$ The first stopper of Δ appears in the i^{th} round

$$= \sum_{i=1}^{k} P[S_i \cap T_i] = \sum_{i=1}^{k} P[S_i] \cdot P[T_i] \leq 16 \sum_{i=1}^{k} P[S_i] \cdot P[T_{i-1}]$$

Partial randomization

Lemma: For a given trapezoid Δ the probability p_{PRIC} of occurring in a partial RIC is at most 16 times the probability p_{RIC} of occurring in a (standard) RIC.

$$p_{PRIC} \leq P[\ \boxed{\underbrace{1\ 2\ \dots\ i\text{-}1}_{\text{triggers}}\ i\ \underbrace{i{+}1\ \dots\ k}_{\text{stoppers}}}\ ] = P[\bigcup_{i=1}^{k}(S_i \cap T_i)]$$



$T_i =$ All triggers of Δ appear in the i^{th} round or before

$S_i =$ The first stopper of Δ appears in the i^{th} round

$$= \sum_{i=1}^{k} P[S_i \cap T_i] = \sum_{i=1}^{k} P[S_i] \cdot P[T_i] \leq 16 \sum_{i=1}^{k} P[S_i] \cdot P[T_{i-1}]$$

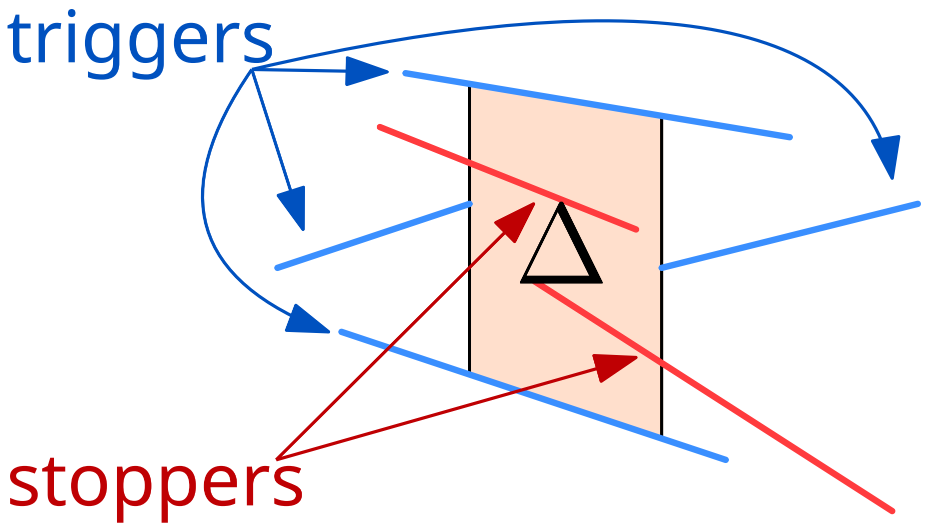
$$\begin{aligned} P[T_{i-1}] &= P[T_{i-1} \cap T_i] \\ &= P[T_{i-1} | T_i] P[T_i] \\ &\geq 1/2^4 P[T_i] \end{aligned}$$

Partial randomization

Lemma: For a given trapezoid Δ the probability p_{PRIC} of occurring in a partial RIC is at most 16 times the probability p_{RIC} of occurring in a (standard) RIC.

$$p_{PRIC} \leq P[\ \boxed{1\,2\ \ldots\ i\text{-}1\ i\ i{+}1\ \ldots\ k}\ ] = P[\bigcup_{i=1}^{k}(S_i \cap T_i)]$$

triggers stoppers



$T_i =$ All triggers of Δ appear in the i^{th} round or before

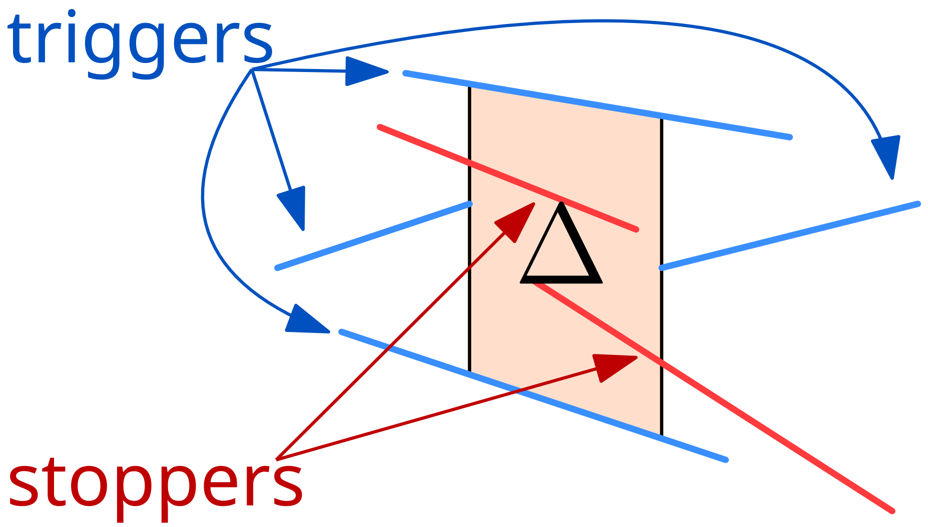
$S_i =$ The first stopper of Δ appears in the i^{th} round

$$= \sum_{i=1}^{k} P[S_i \cap T_i] = \sum_{i=1}^{k} P[S_i] \cdot P[T_i] \leq 16 \sum_{i=1}^{k} P[S_i] \cdot P[T_{i-1}]$$

Partial randomization

Lemma: For a given trapezoid Δ the probability p_{PRIC} of occurring in a partial RIC is at most 16 times the probability p_{RIC} of occurring in a (standard) RIC.

$$p_{PRIC} \leq P[\ \boxed{\underbrace{1\,2\,\ldots\,i\text{-}1\;i}_{\text{triggers}}\;\underbrace{i{+}1\,\ldots\,k}_{\text{stoppers}}}\ ] = P[\bigcup_{i=1}^{k}(S_i \cap T_i)]$$



$T_i =$ All triggers of Δ appear in the i^{th} round or before

$S_i =$ The first stopper of Δ appears in the i^{th} round

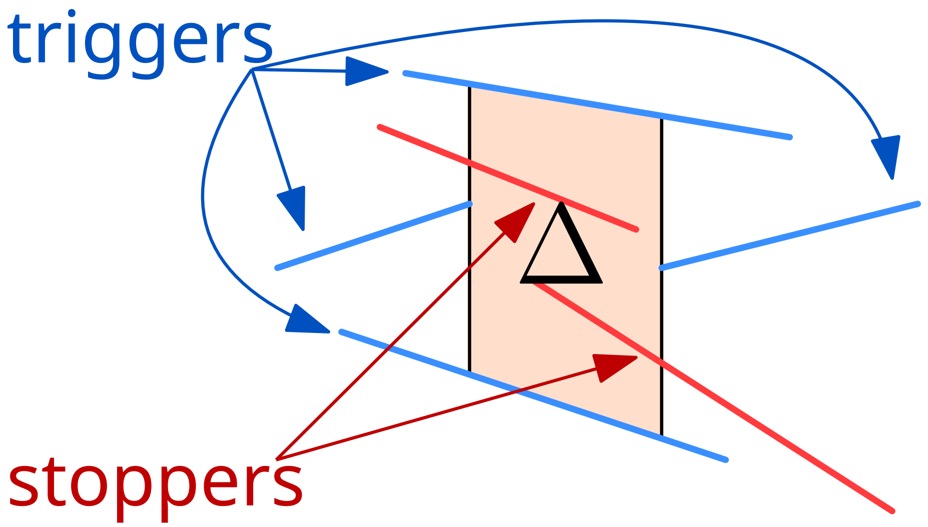
$$= \sum_{i=1}^{k} P[S_i \cap T_i] = \sum_{i=1}^{k} P[S_i] \cdot P[T_i] \leq 16 \sum_{i=1}^{k} P[S_i] \cdot P[T_{i-1}]$$

$$= 16P[\bigcup_{i=1}^{k}(S_i \cap T_{i-1})] = 16P[\ \boxed{\underbrace{1\,2\,\ldots\,i\text{-}1}\;\underbrace{i\;i{+}1\,\ldots\,k}}\ ]$$

Partial randomization

Lemma: For a given trapezoid Δ the probability p_{PRIC} of occurring in a partial RIC is at most 16 times the probability p_{RIC} of occurring in a (standard) RIC.

$$p_{PRIC} \leq P[\ \boxed{\underbrace{1\ 2\ \dots\ i\text{-}1}_{\text{triggers}}\ i\ \underbrace{i{+}1\ \dots\ k}_{\text{stoppers}}}\ ] = P[\bigcup_{i=1}^{k} (S_i \cap T_i)]$$



$T_i =$ All triggers of Δ appear in the i^{th} round or before

$S_i =$ The first stopper of Δ appears in the i^{th} round

$$= \sum_{i=1}^{k} P[S_i \cap T_i] = \sum_{i=1}^{k} P[S_i] \cdot P[T_i] \leq 16 \sum_{i=1}^{k} P[S_i] \cdot P[T_{i-1}]$$

$$= 16P[\bigcup_{i=1}^{k} (S_i \cap T_{i-1})] = 16P[\boxed{\underbrace{1\ 2\ \dots\ i\text{-}1}_{}\ \underbrace{i\ i{+}1\ \dots\ k}_{}}] \leq 16p_{RIC}$$

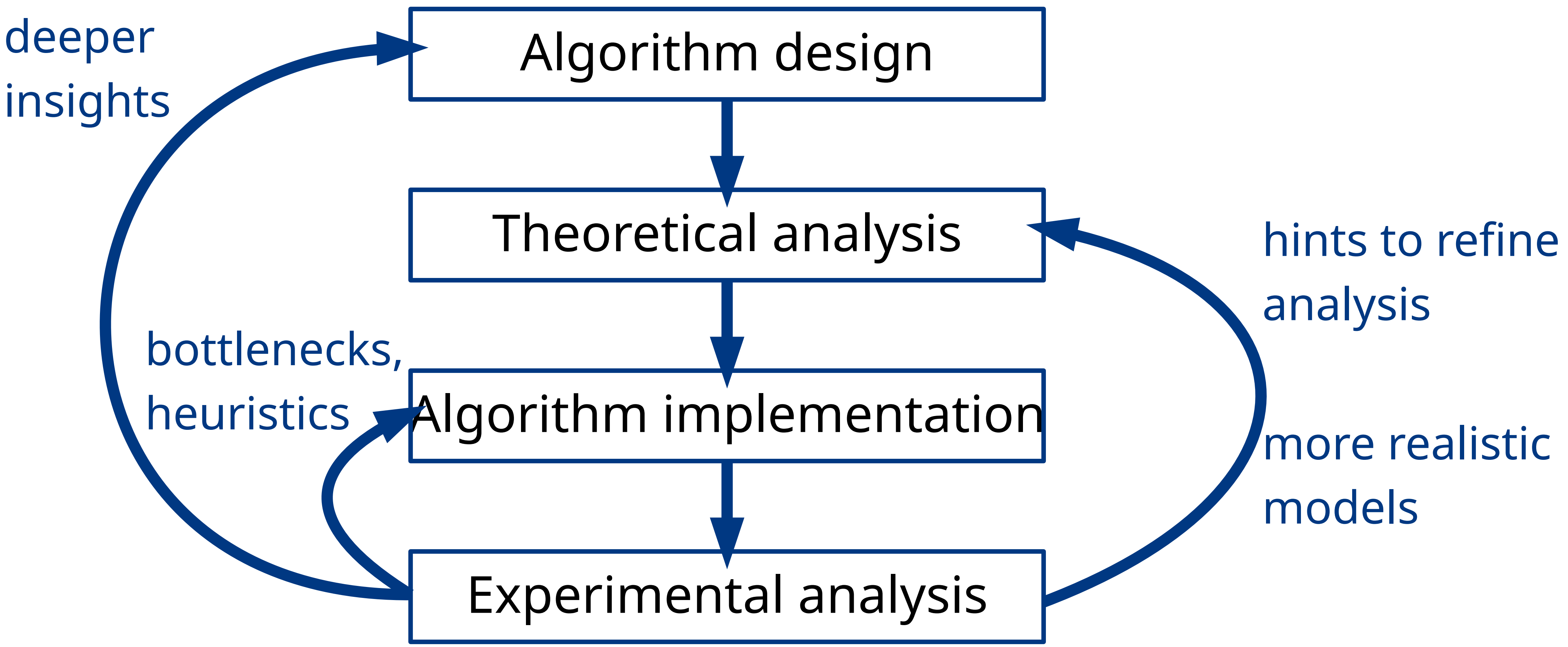
Partial randomization

Lemma: For a given trapezoid Δ the probability p_{PRIC} of occurring in a partial RIC is at most 16 times the probability p_{RIC} of occurring in a (standard) RIC.

Theorem: The trapezoidal decomposition of n non-intersecting segments in the plane can be constructed in $O(n \log n)$ expected time using partial randomization.

Algorithm engineering cycle

- implementations, experiments, and theory go well together
- robust implementations are challenging
- strong experimental analysis is crucial



Experimental analysis

A Theoretician's Guide to the Experimental Analysis of Algorithms [Johnson]:

- Perform "newsworthy" experiments
- Place work in context
- Use reasonably efficient implementations
- Use testbeds that support general conclusions
- Provide explanations and back them up with experiment
- Ensure reproducibility
- Ensure comparability (and give the full picture)