Presentation writer: Tigran Hayrapetyan

Lecturer | Programmer | Researcher

www.linkedin.com/in/tigran-hayrapetyan-cs/

Convex hull

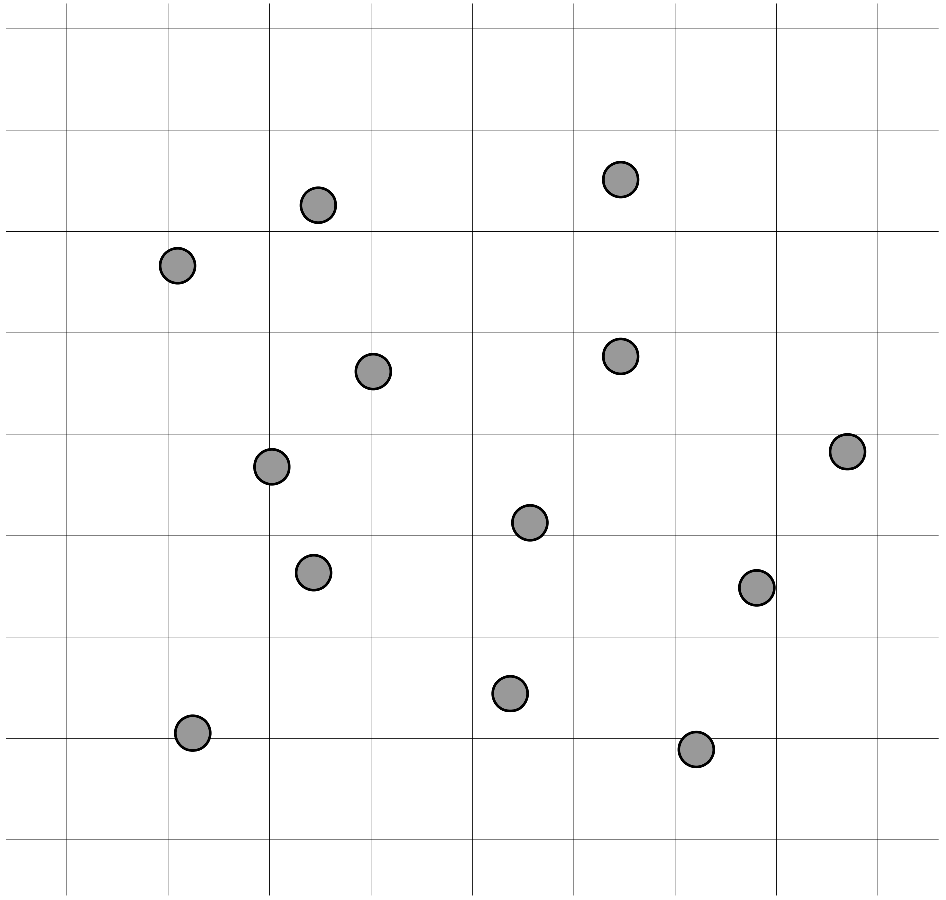
(Graham scan)

Prerequisites:

- *<none>*

Convex hull

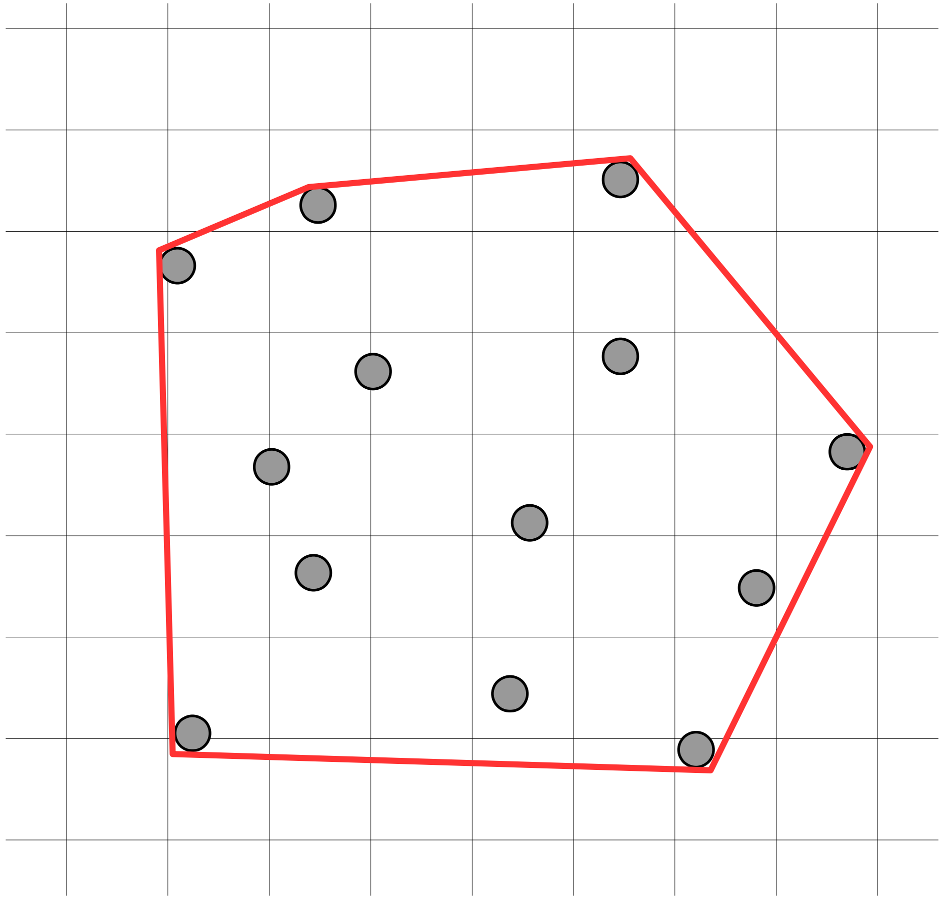
For given set of points, convex hull is such a smallest convex polygon, which contains them all.



Convex hull

For given set of points, convex hull is such a smallest convex polygon, which contains them all.

... so we can say that convex hull can be presented as a subset of set of those points.

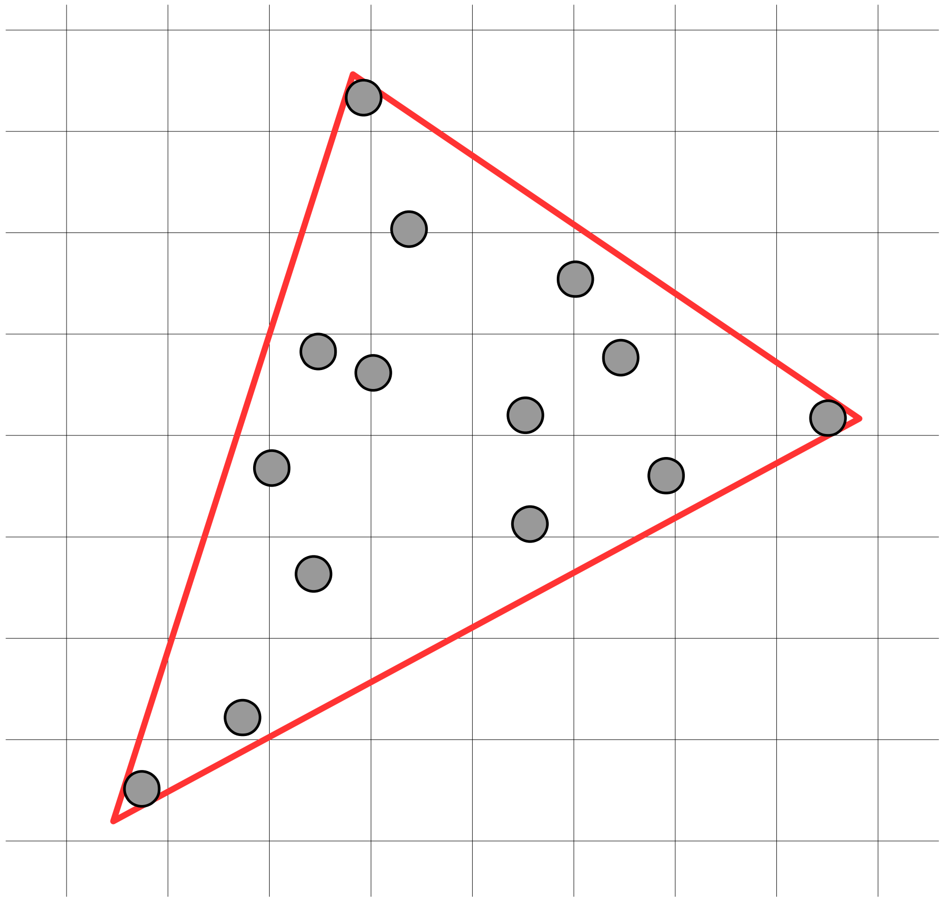


Convex hull

For some cases convex hull can have only **3** points:

n = 14,

h = 3.

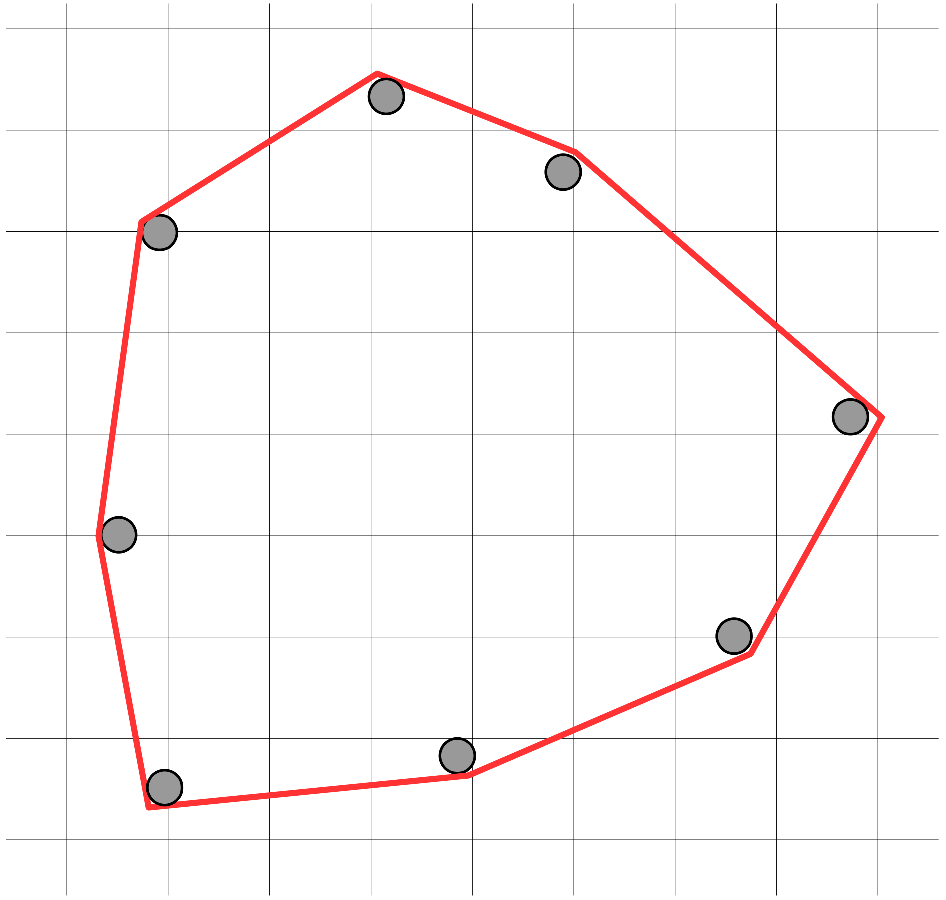


Convex hull

While in other cases all the **n** points might participate in the hull:

n = 8,

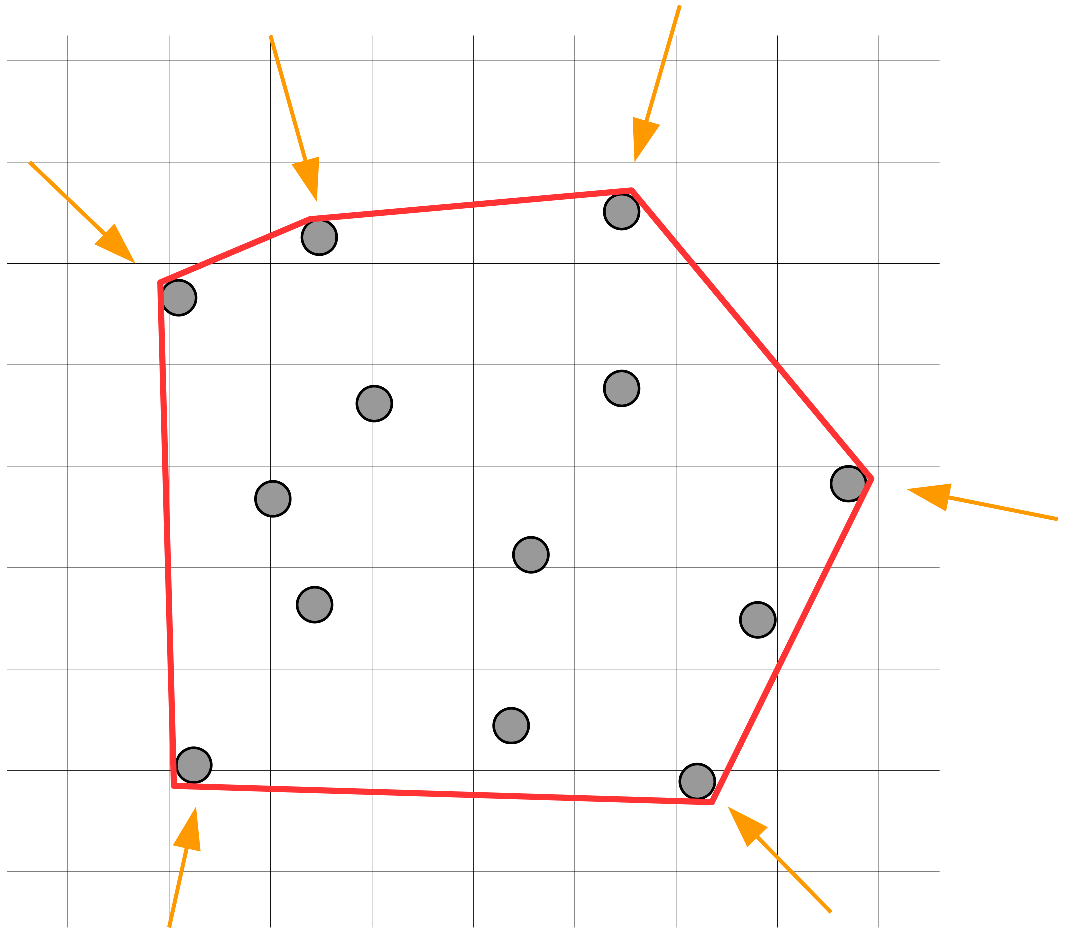
h = 8.



Convex hull

So finding the convex hull is:

... <u>finding those points</u> from the given **n** ones, which form the convex hull.



Convex hull

```
(5, 12),      (3, 12),
(3, 14),      (17, 1),
(15, 9),      (18, 2),
(14, 6),      (20, 6),
(3, 2),       (11, 12),
(1, 19),      (8, 8),
(18, 4),      (7, 9),
(3, 7),       (16, 10),
(15, 14),     (9, 3).
(6, 20),
```

It might be easy to do when all the points are drawn on a plane,

... but in computers points are represented by pairs of **(X,Y)**.

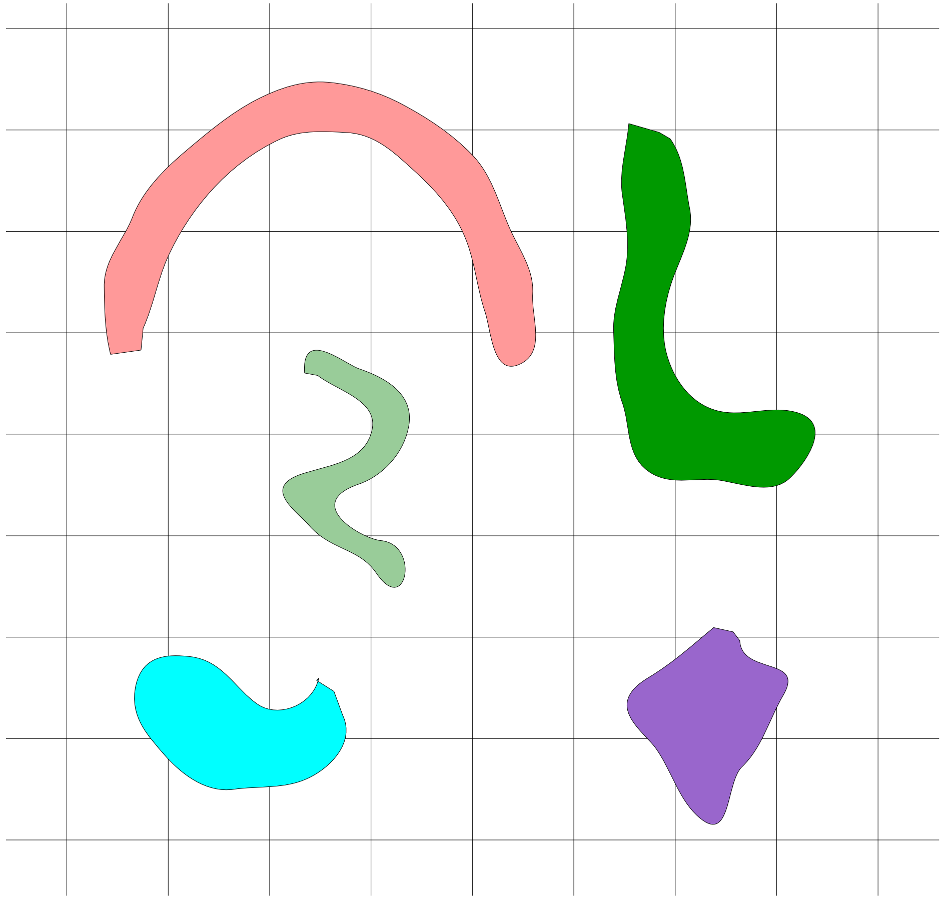
Usage

1) Simulation software:

... objects can be of various shapes,

... and <u>might collide</u> during move,

... identifying <u>collisions of concave objects</u> is not easy



Usage

1) Simulation software:

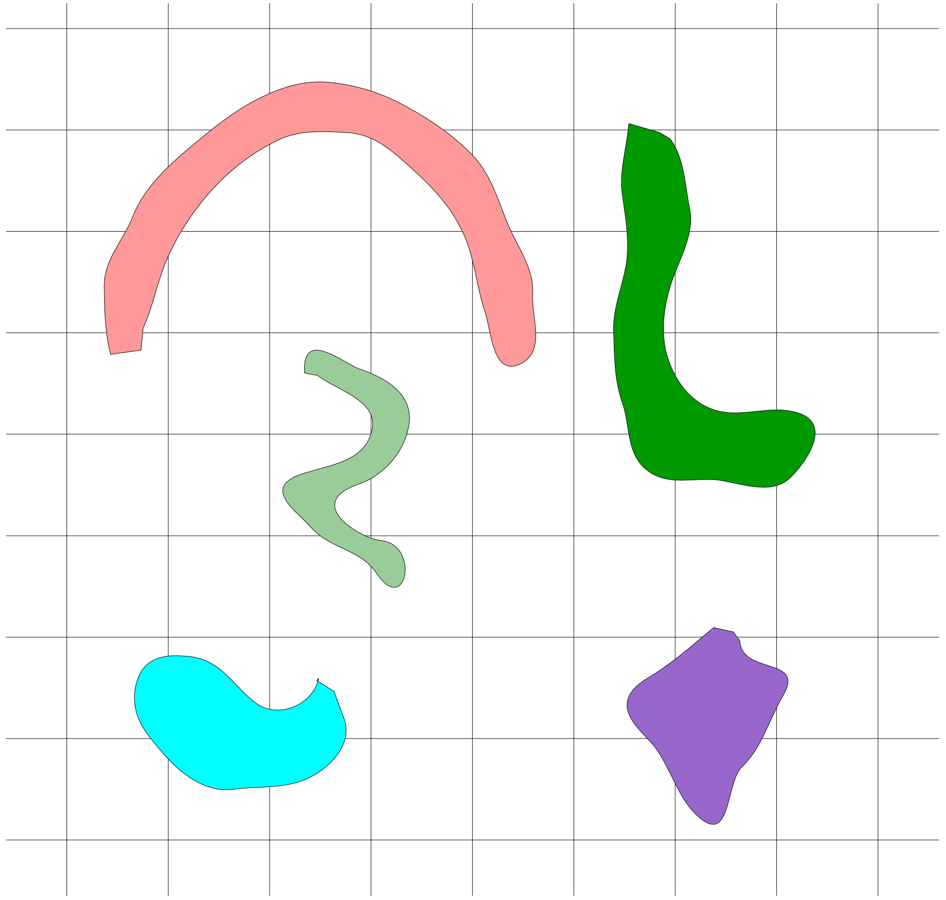
...

...

... that's why at first <u>convex hull of all objects is calculated</u>,

... and it is checked <u>if convex hulls do collide</u>,

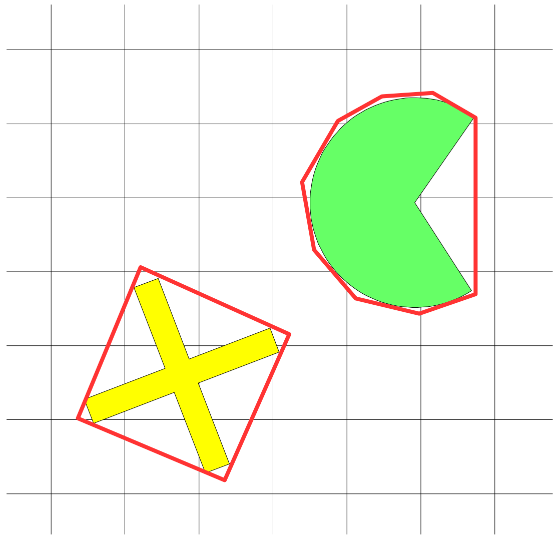
... only if they do, actual intersection between objets is being checked.

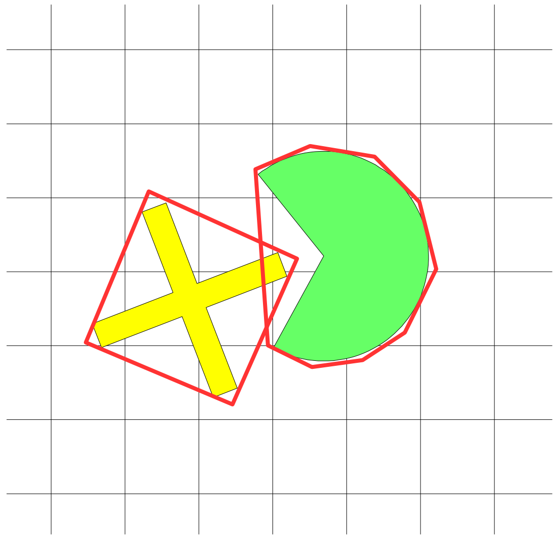


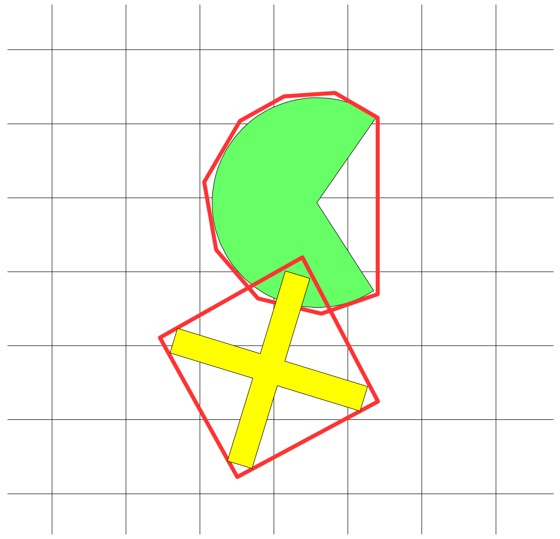
Usage

1) Simulation software:

... note, there can be **3** cases:





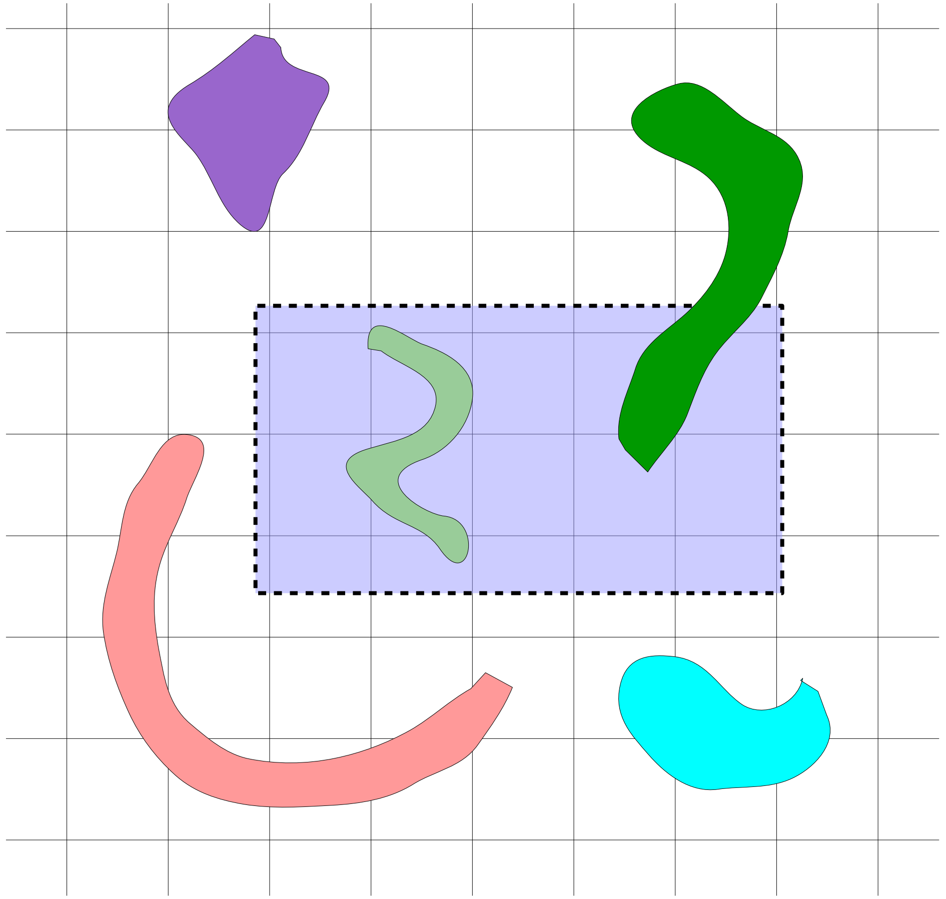


Usage

2) Computer graphics:

... rendering <u>only those objects</u>, which fall on screen area,

... checking if a concave object falls (either entirely or partially) on the screen area is not easy,



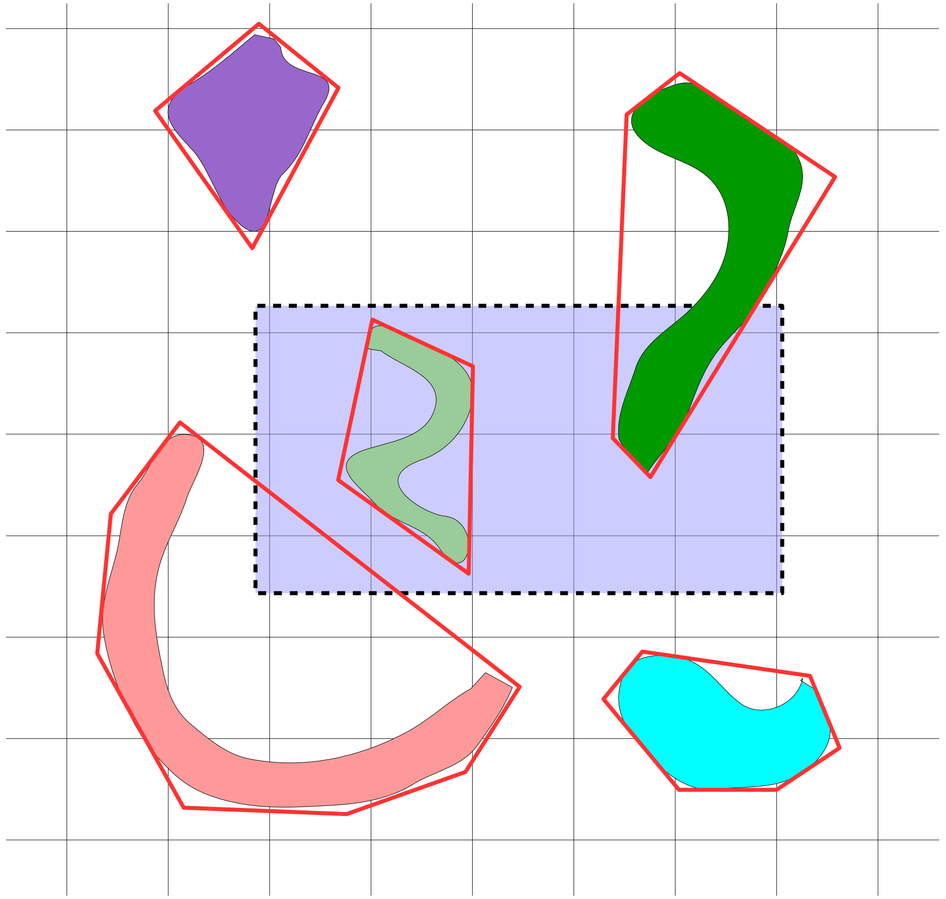
Usage

2) Computer graphics:

...

... so at first we check for intersections between screen area and convex hulls,

... only then check more precisely (or draw).



Graham scan

Graham scan is a convex hull algorithm, which is:

- fast,
- easy to implement,
- simple to understand.

Ronald Graham,

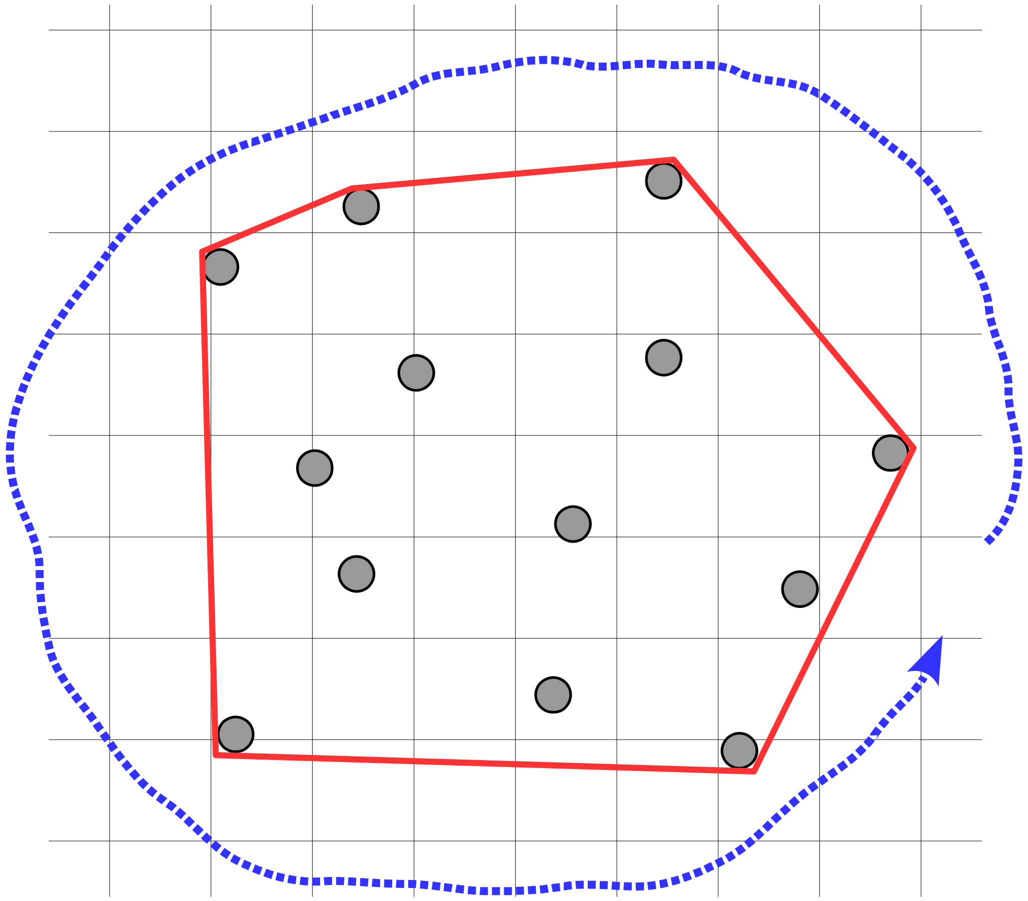
published in 1972.

Graham scan

Key idea:

... moving by convex hull is like traversing the polygon in clockwise direction.

... so perhaps we can figure out the convex hull also by a similar traversal?



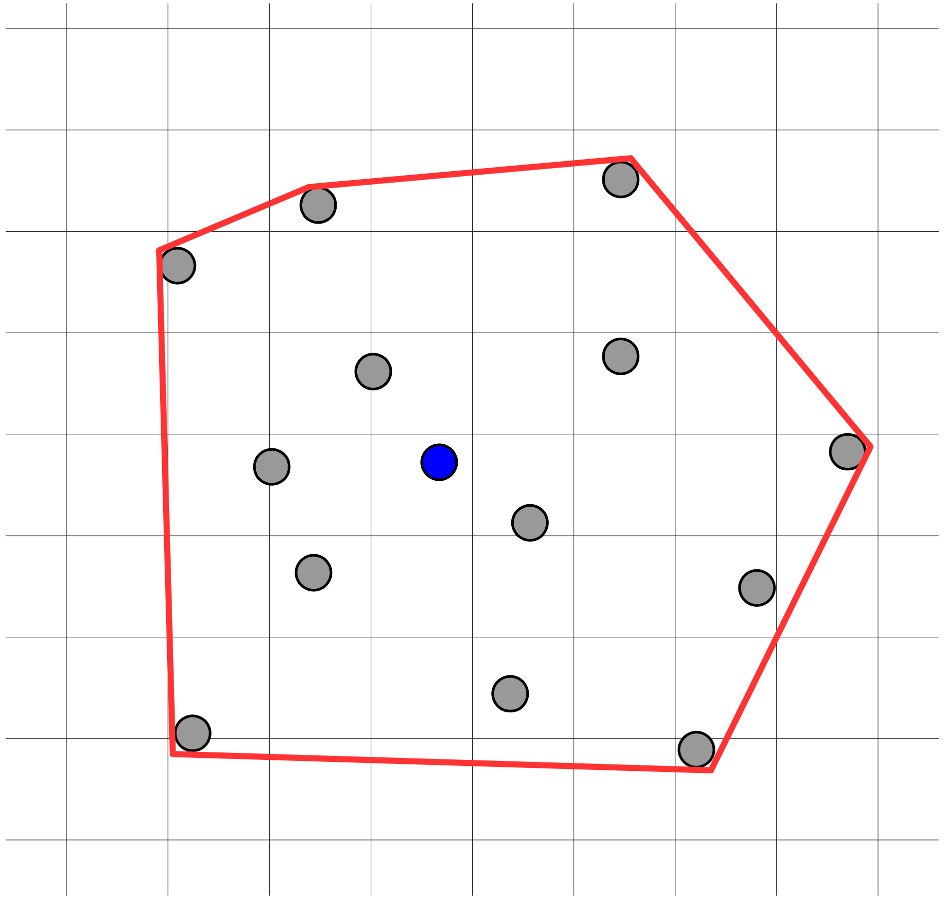
Graham scan

Step 1: Find a point inside, around which we can rotate.

... this can be algebraic mean of all the points:

$$c = \frac{\sum P_i}{N}$$

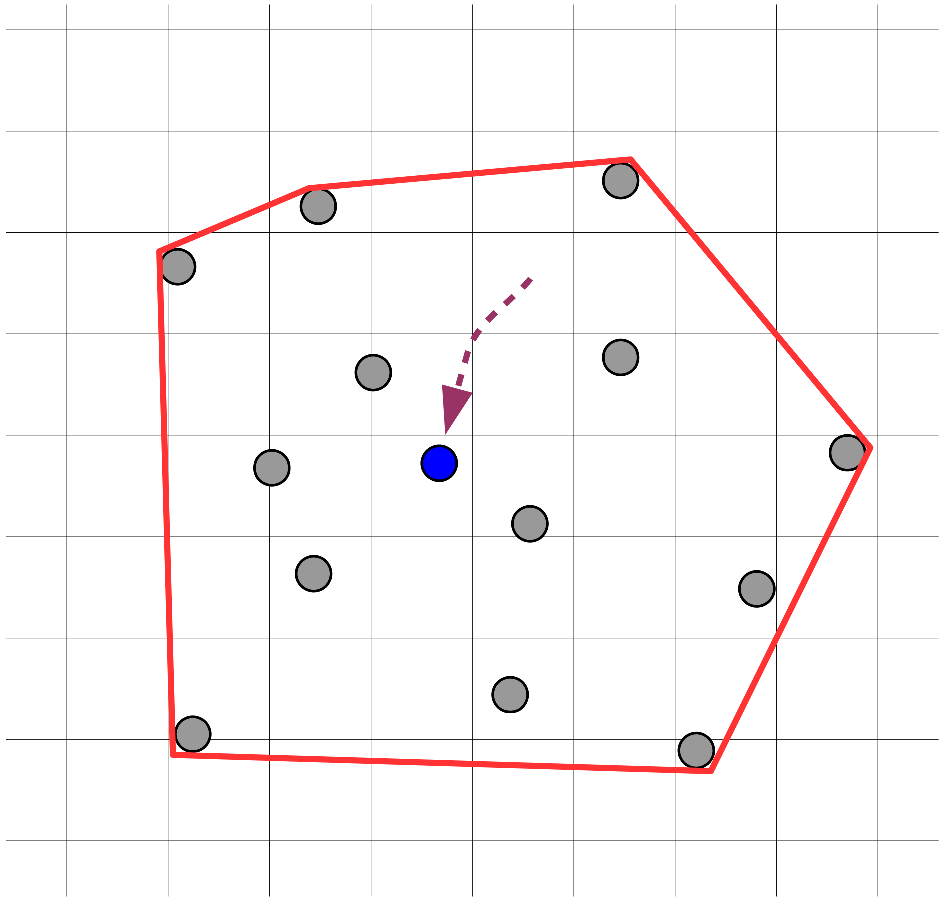
$$X_C = \frac{\sum X_i}{N} \qquad Y_C = \frac{\sum Y_i}{N}$$



Graham scan

Question: can we prove that the mean point will always be inside the convex hull?

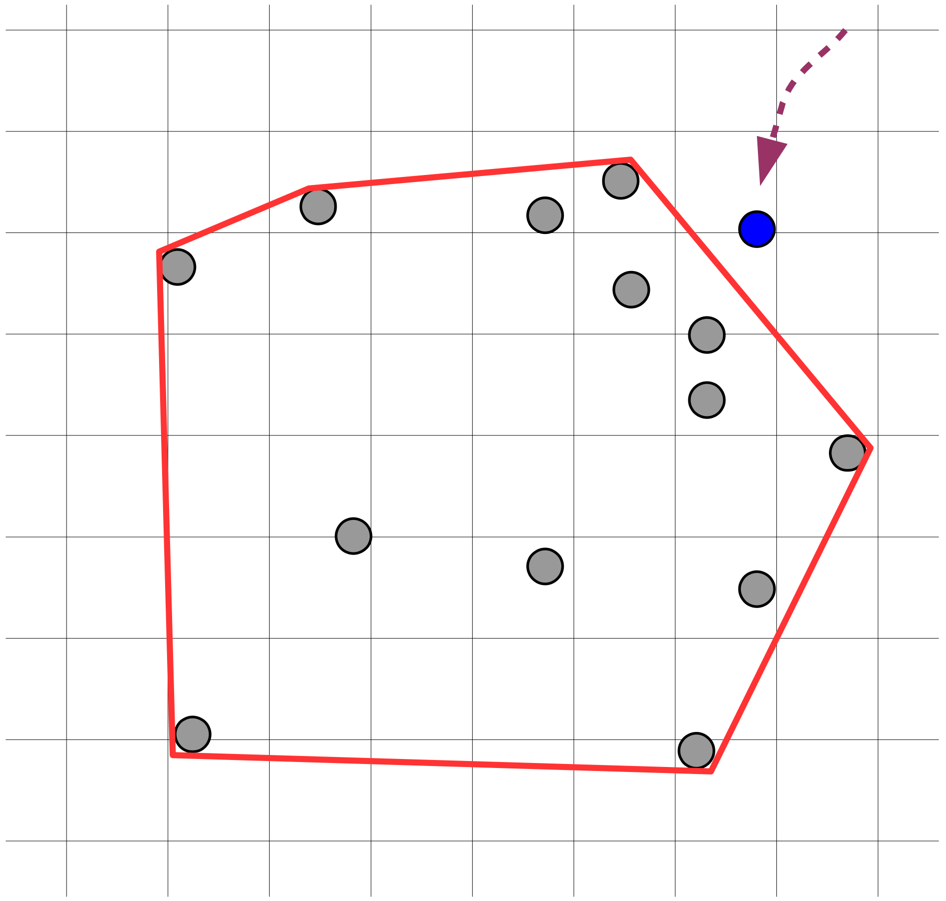
$$c = \frac{\sum P_i}{N}$$



Graham scan

Answer: Yes. Assume for some input arrangement, center appears outside of the hull.

$$c = \frac{\sum P_i}{N}$$



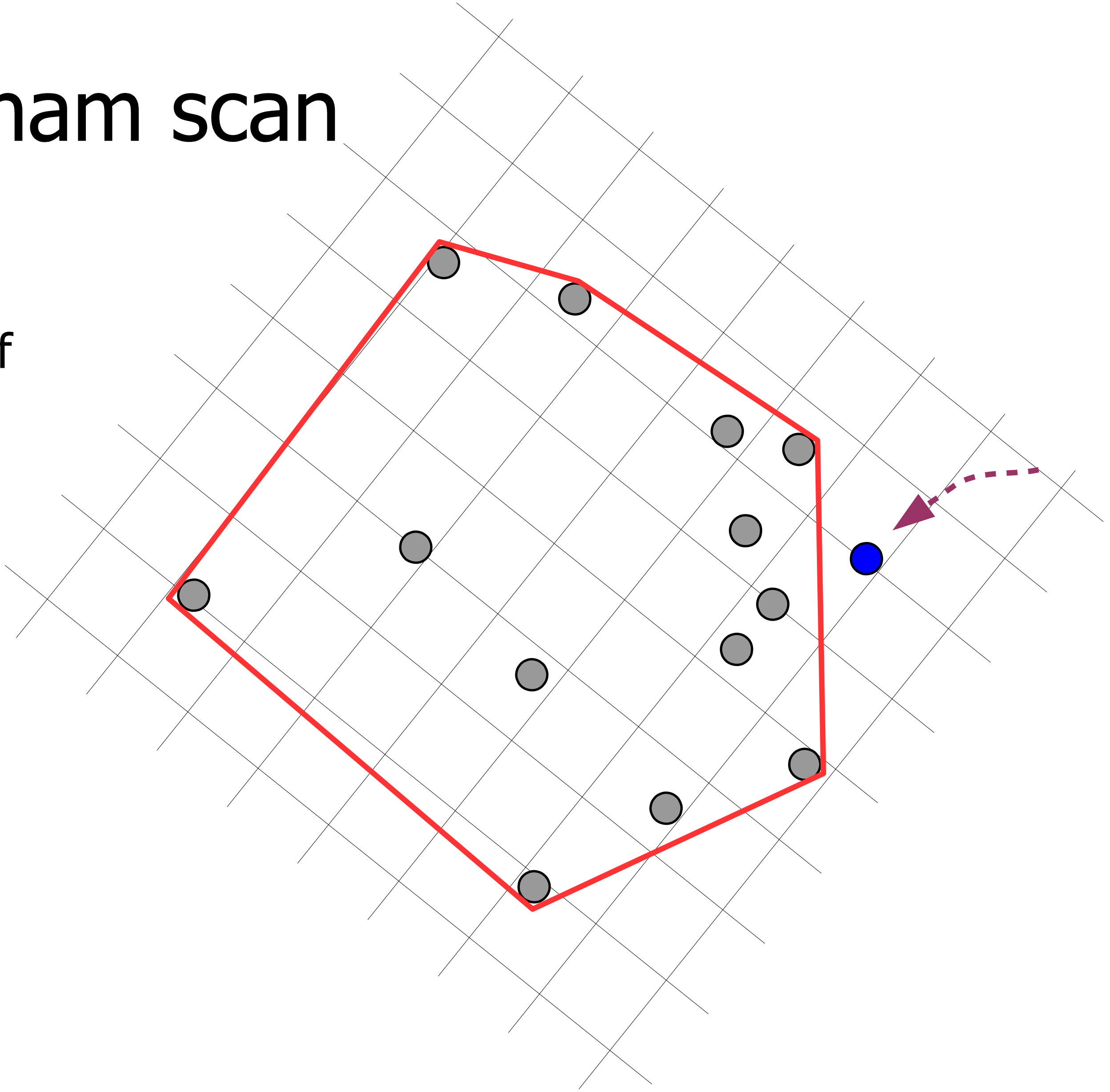
Graham scan

Answer: Yes. Assume for some input arrangement, center appears outside of the hull.

$$c = \frac{\sum P_i}{N}$$

Then let's rotate entire picture,

... center point will remain on place, relatively to the input points.



Graham scan

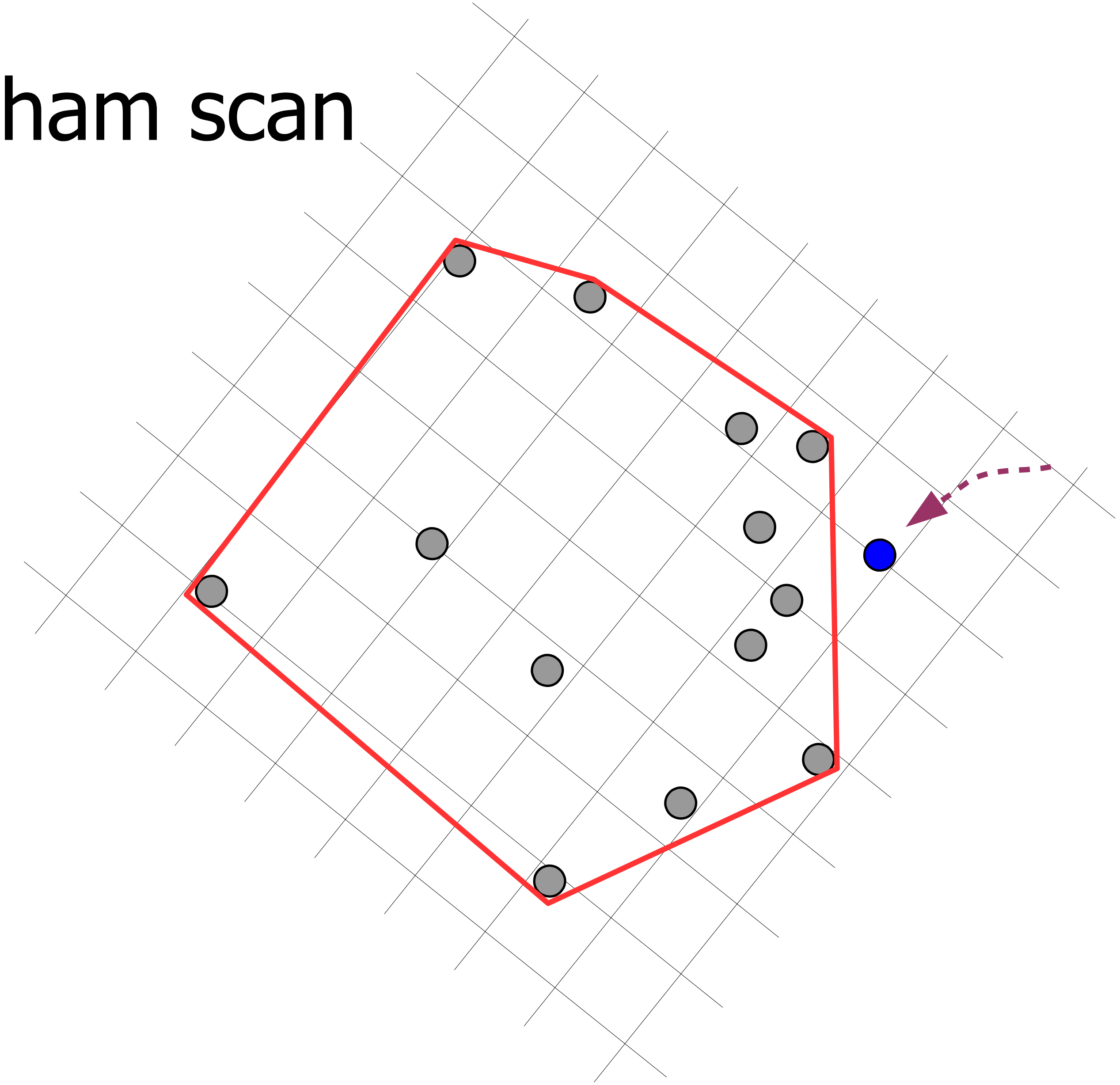
Answer:

…

… so it will result that the "center" is <u>more to right</u> than all other points.

… but that can't happen as mean of **X** can't be greater than all other $\mathbf{X}_i$ -s.

$$X_C = \frac{\sum X_i}{N}$$



Graham scan

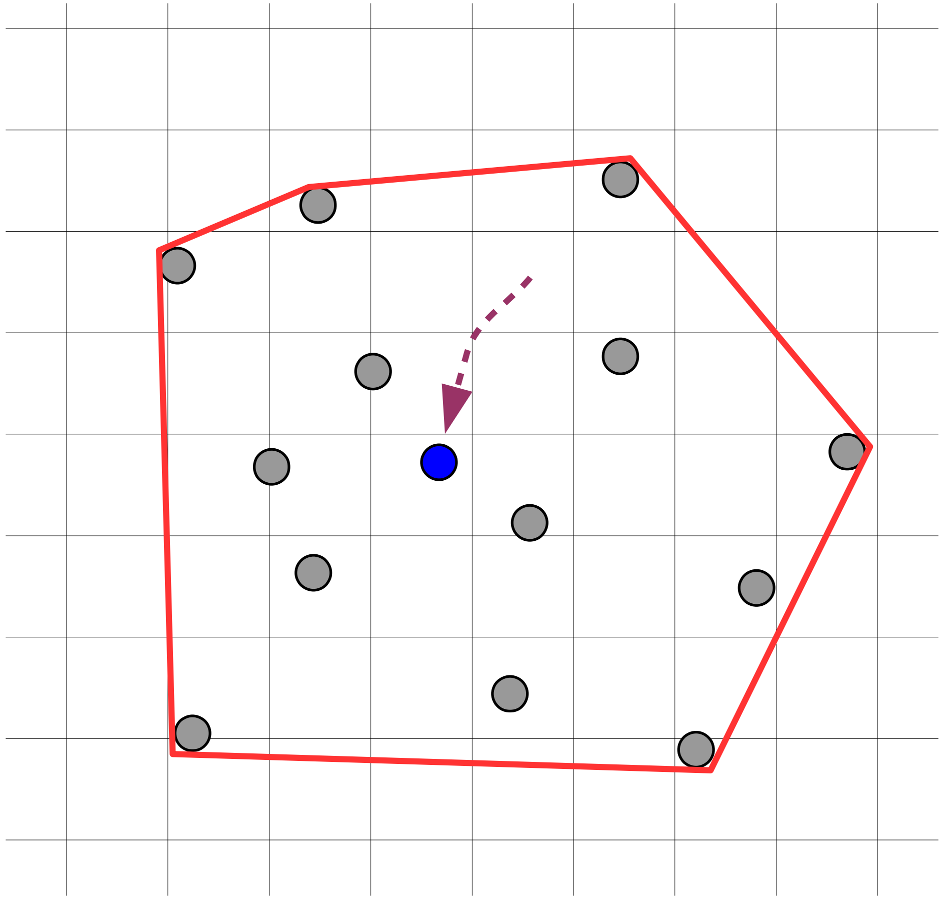
Answer:

…

…

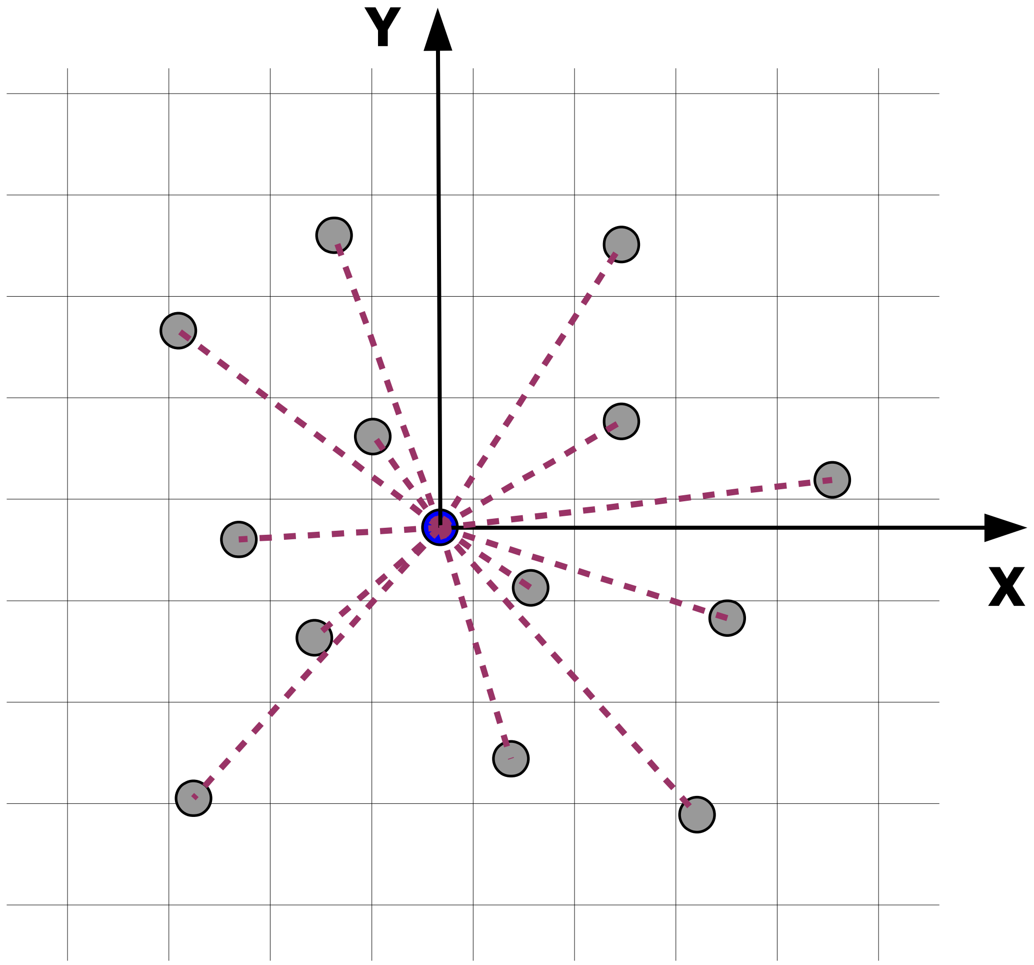
… so the presented formula <u>will always work</u>.

$$c = \frac{\sum P_i}{N}$$



Graham scan

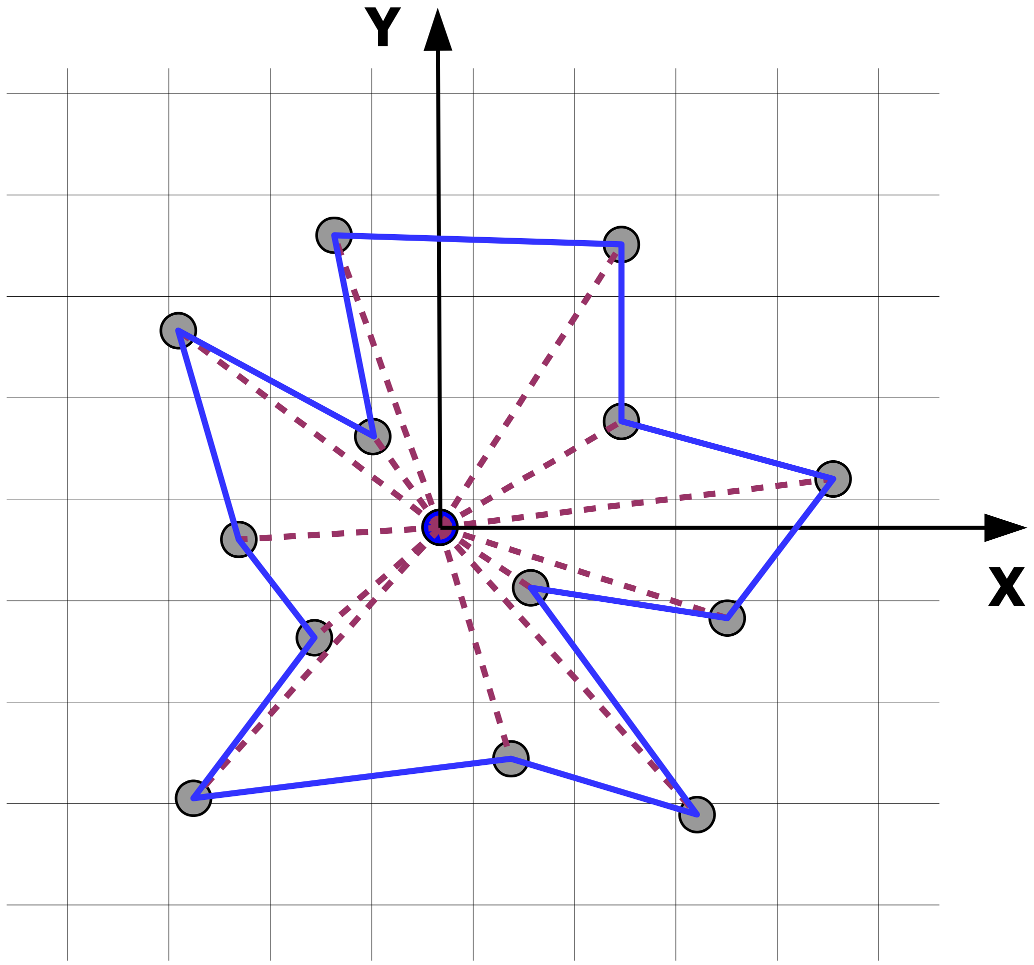
Step 2: Sort all the **N** points by increase of angle, that they have looking from the "center" point.



Graham scan

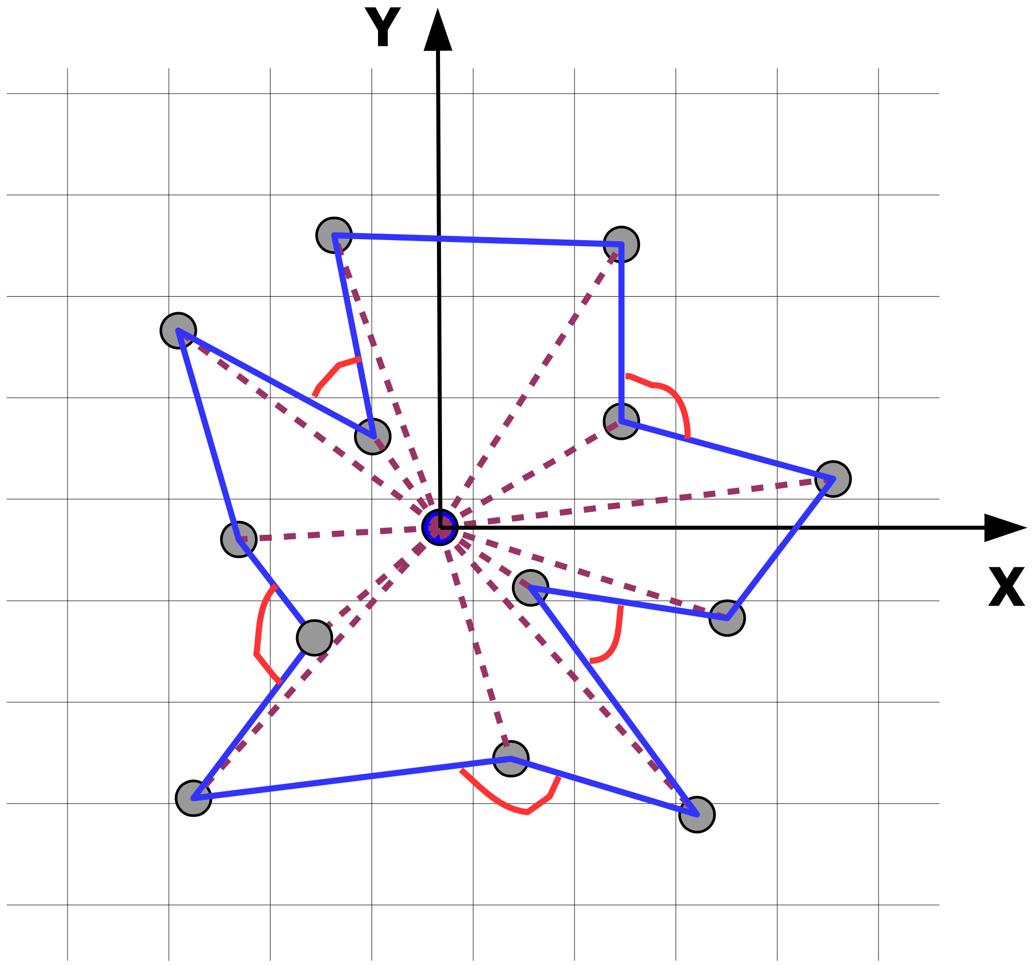
Step 2: Sort all the **N** points by increase of angle, that they have looking from the "center" point.

... so after sorting we can kind of traverse the following concave polygon.



Graham scan

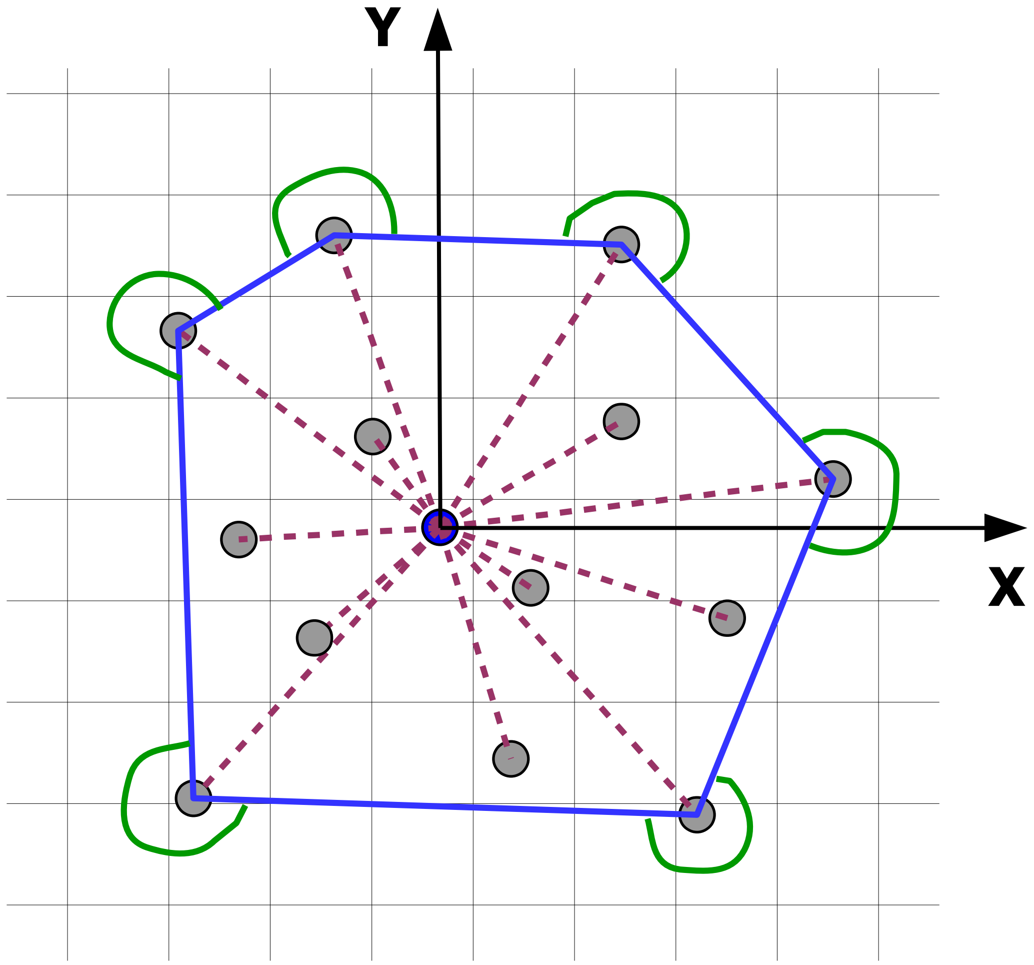
Step 3: When traversing clockwise, repeatedly remove all "right turns",



Graham scan

Step 3: When traversing clockwise, repeatedly remove all "right turns",

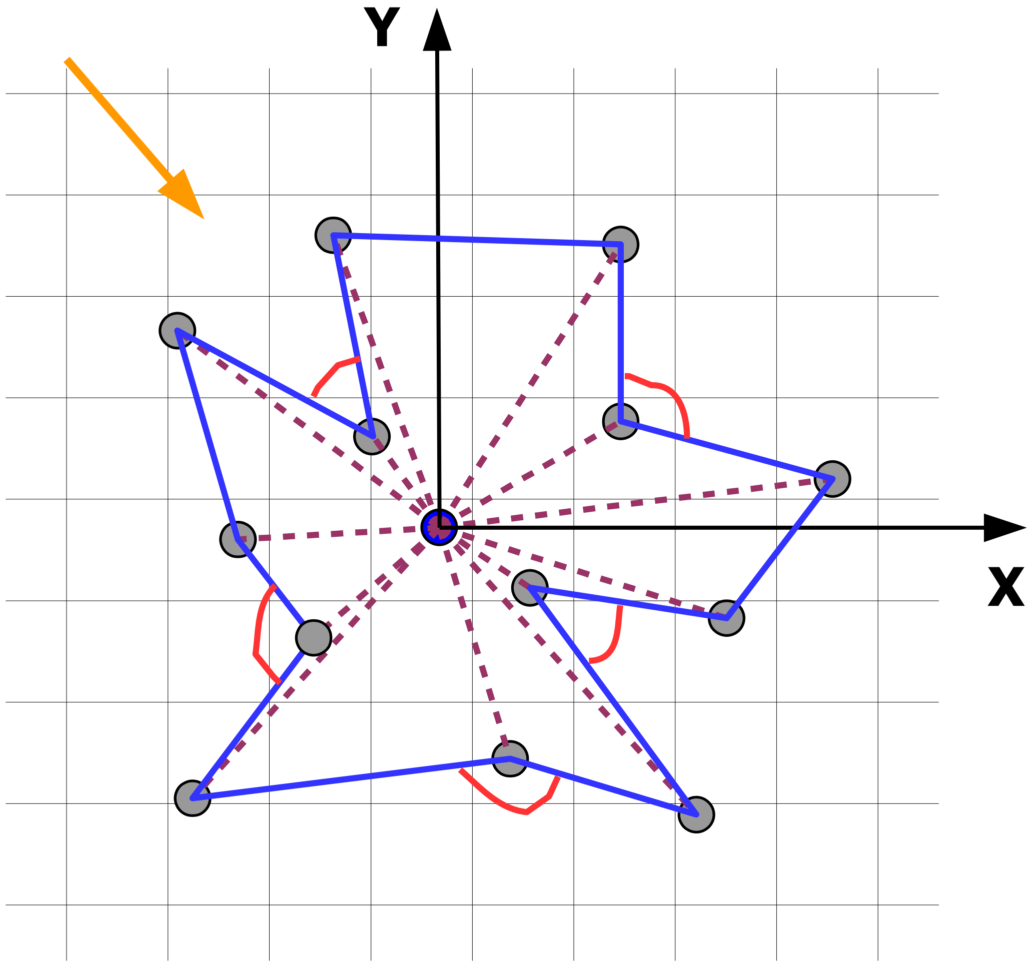
... because the convex hull should have only "left turns".



Graham scan

<u>How we are going</u> to do that?

... one way is to repeatedly search for right turns,

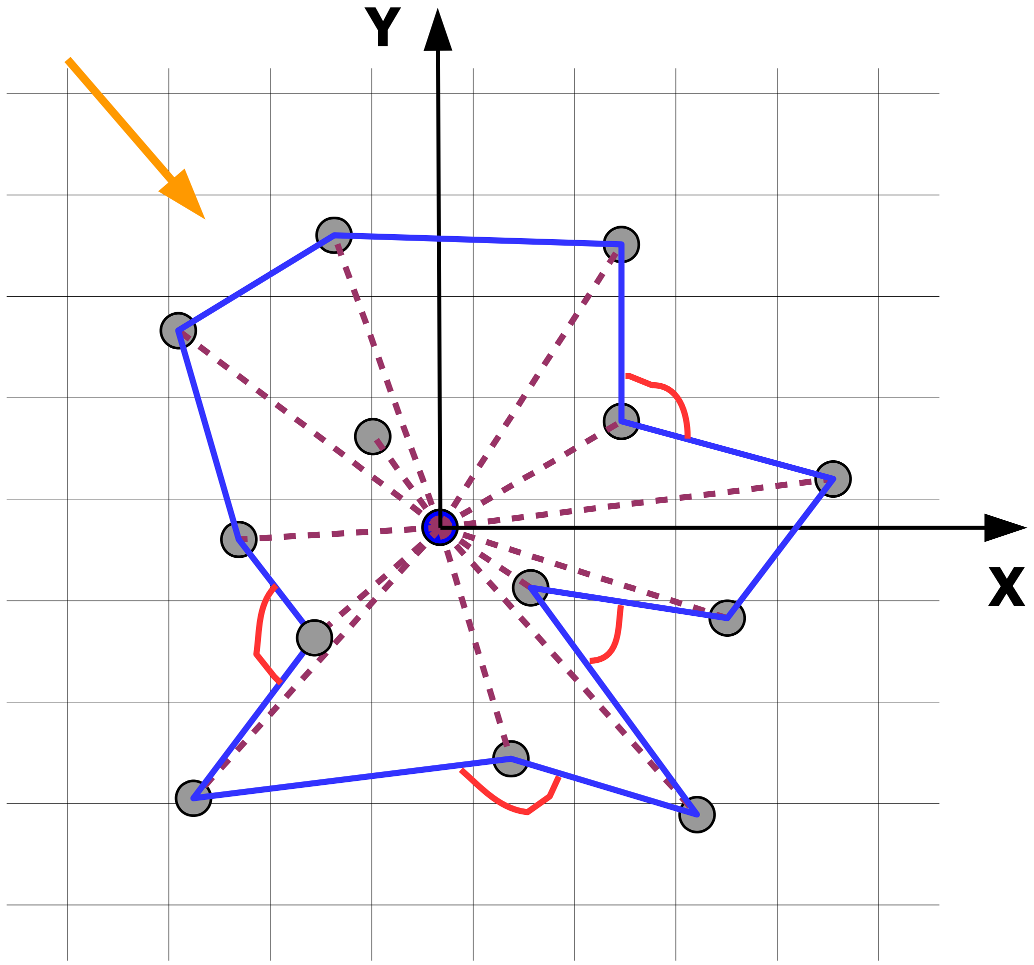


Graham scan

How we are going to do that?

... one way is to repeatedly search for right turns,

... remove them,



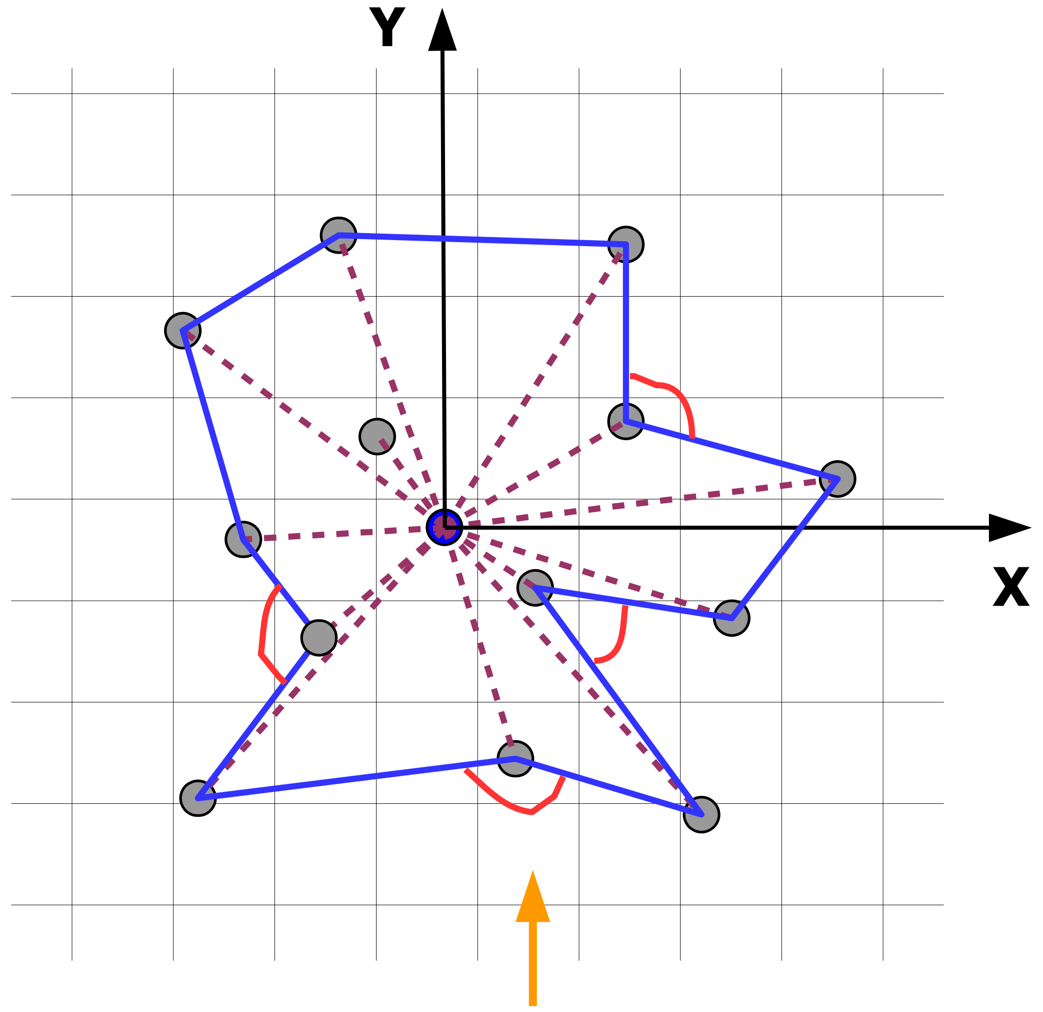
Graham scan

<u>How we are going</u> to do that?

... one way is to repeatedly search for right turns,

... remove them,

... then search again.



Graham scan

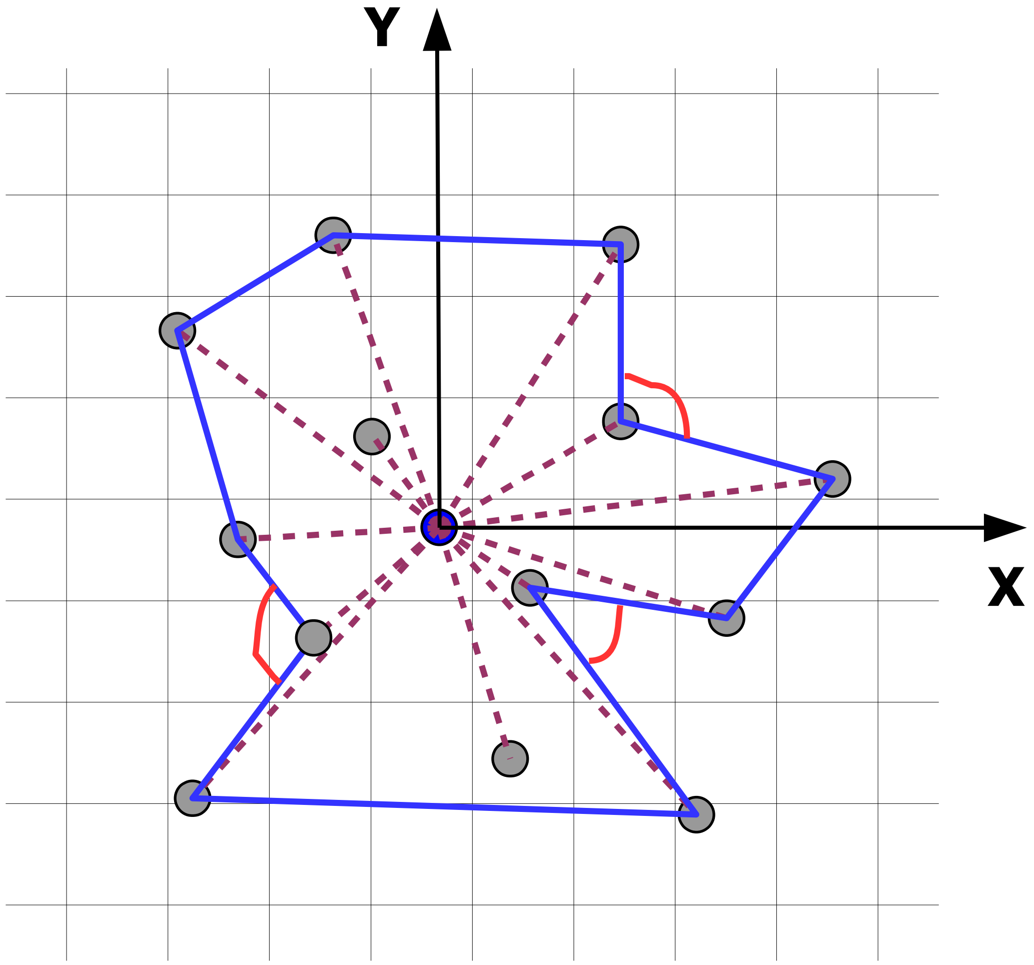
<u>How we are going</u> to do that?

... one way is to repeatedly search for right turns,

... remove them,

... then search again.

Such implementation can take up to **O(n^2)** time.



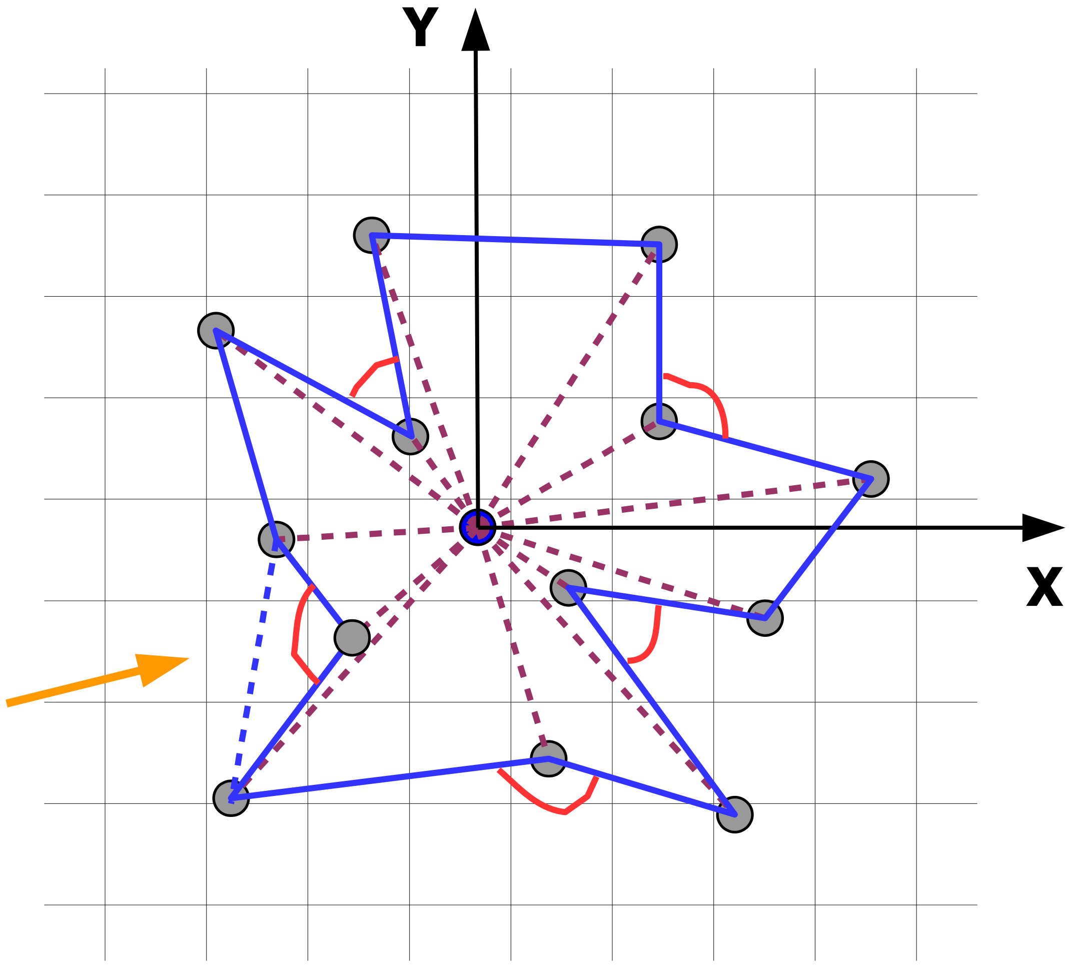
Graham scan

How to do it better?

Note, once a "right turn" is removed, only its adjacent angles are being affected.

... no sense to re-check entire polygon again,

... so we can walk over the polygon only by adjacent vertices.

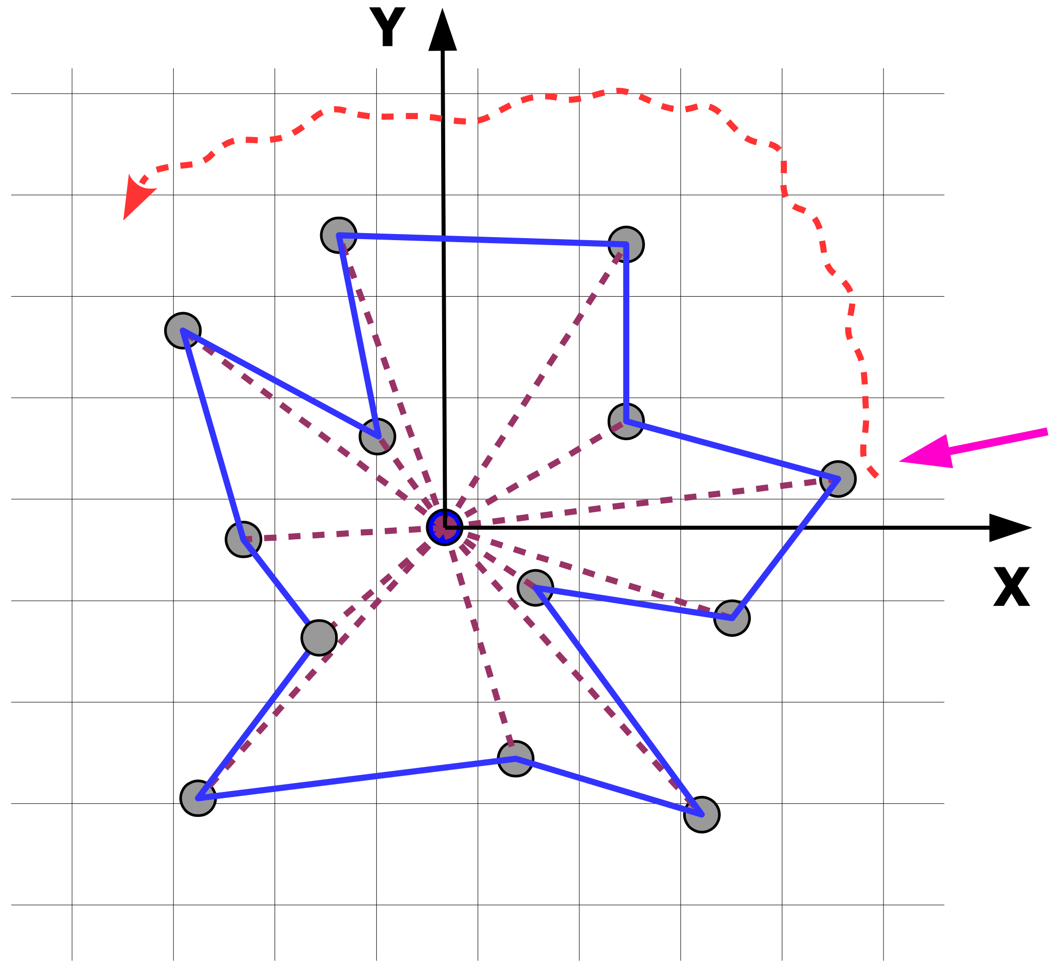


Graham scan

Step 3:

Start from the <u>right-most point</u>,

... it definitely participates in the hull.



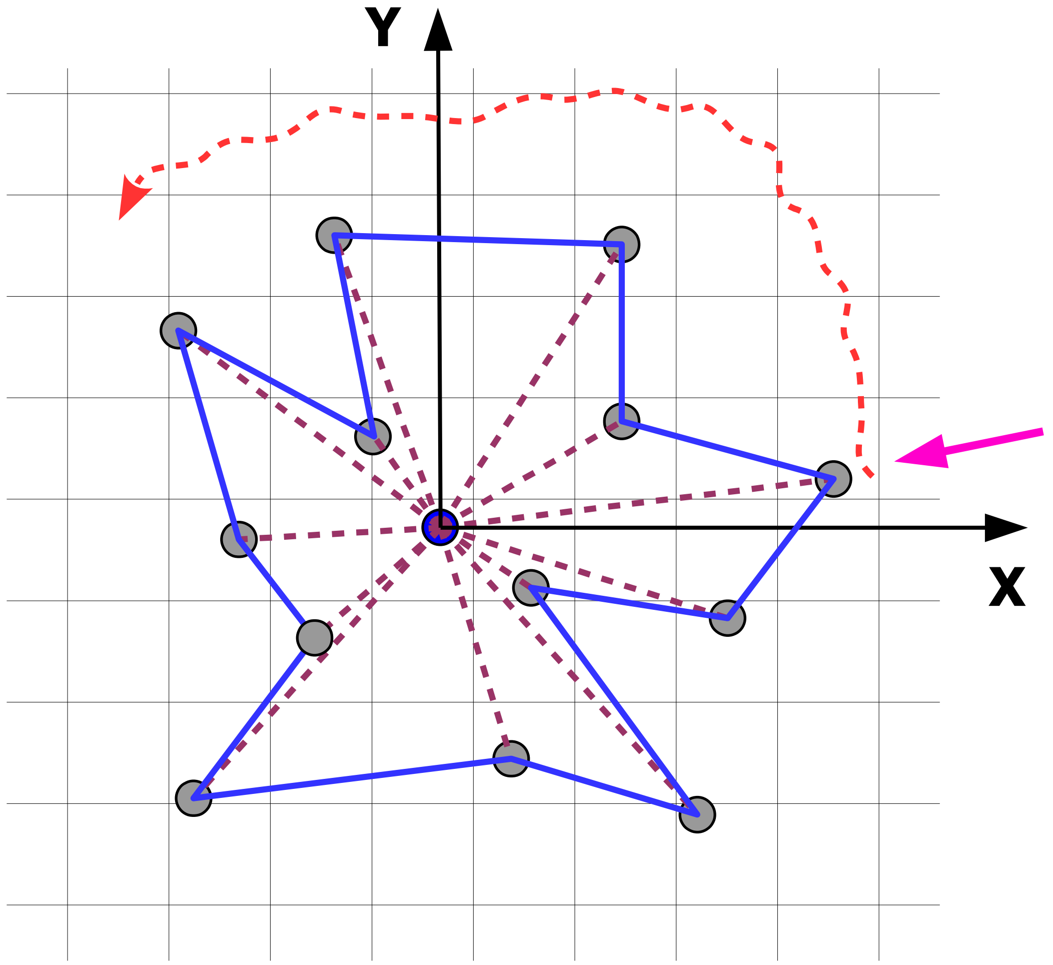
Graham scan

Step 3:

Start from the right-most point,

... it definitely participates in the hull.

If meeting "left turn", continue.



Graham scan

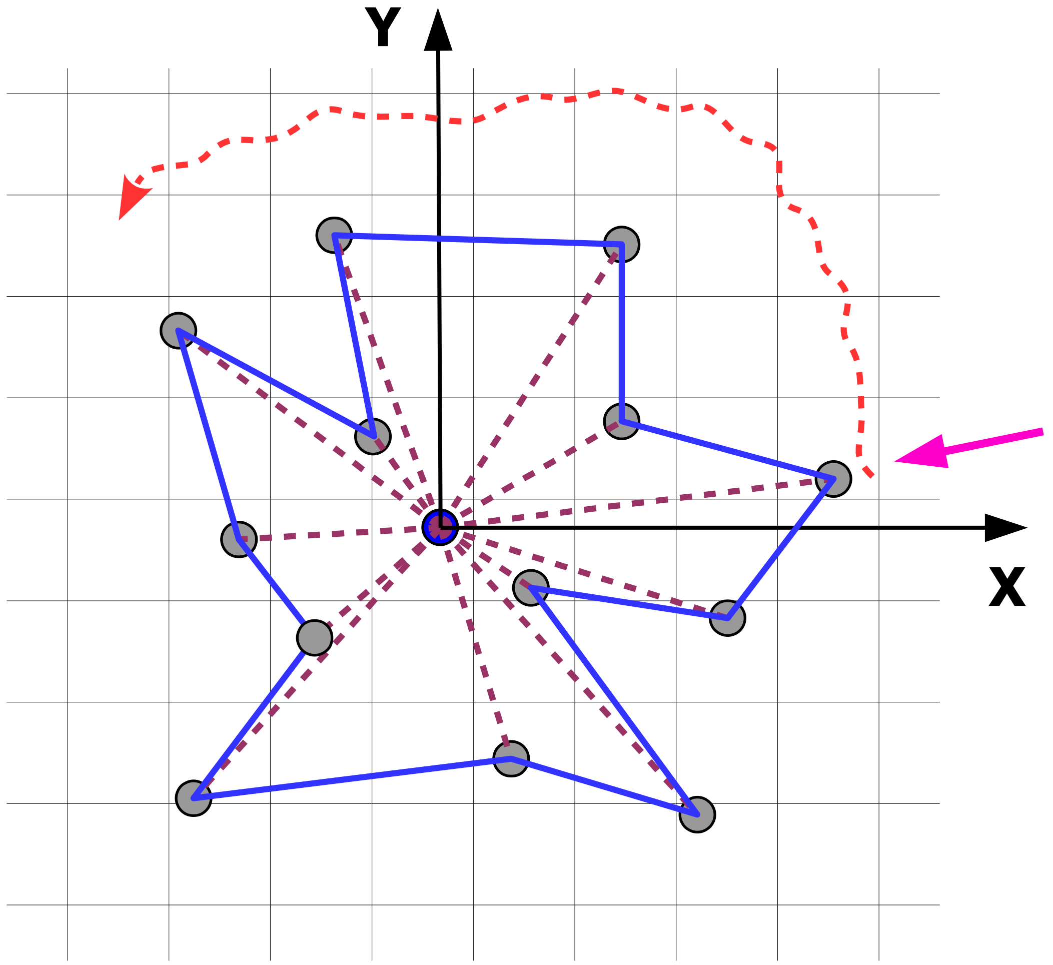
Step 3:

Start from the right-most point,

... it definitely participates in the hull.

If meeting "left turn", continue.

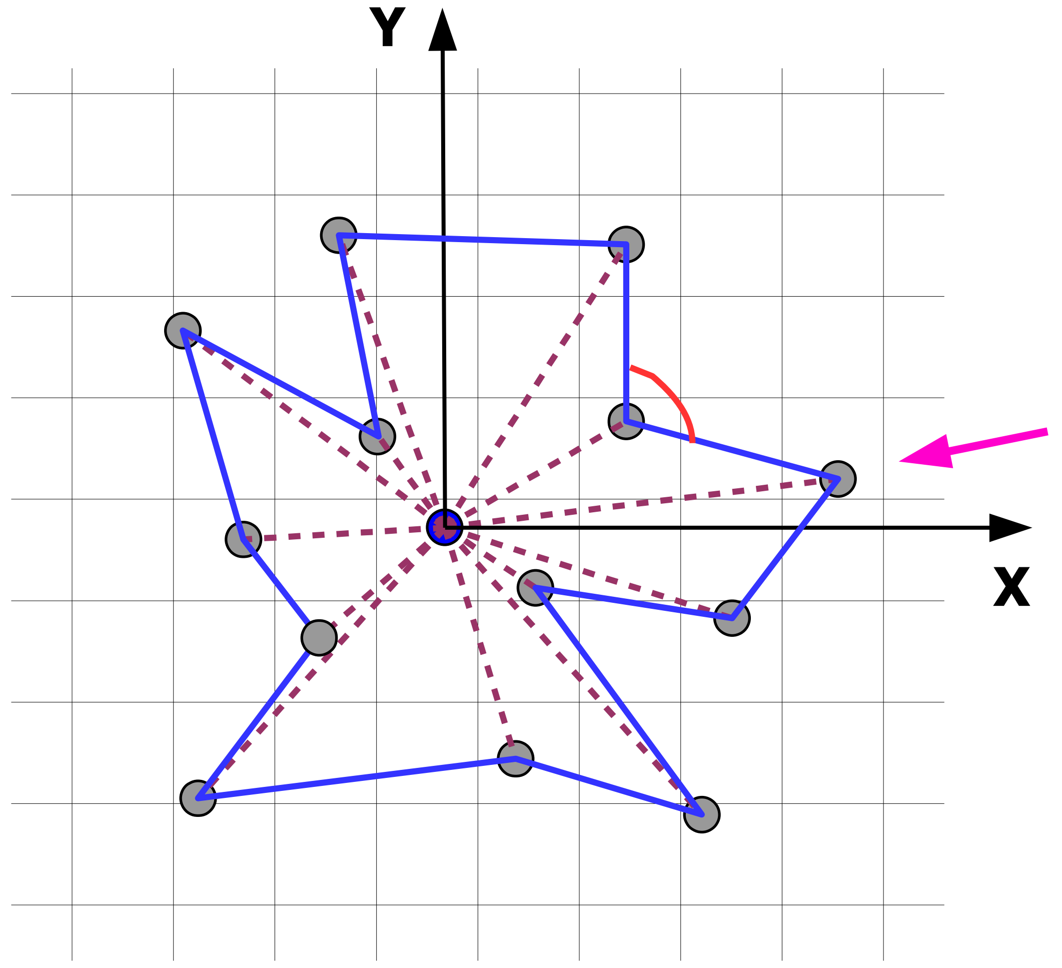
If meeting "right turn", remove the previous points, as many for the "right turn" to eliminate.



Graham scan

Complete invocation:

... 1,

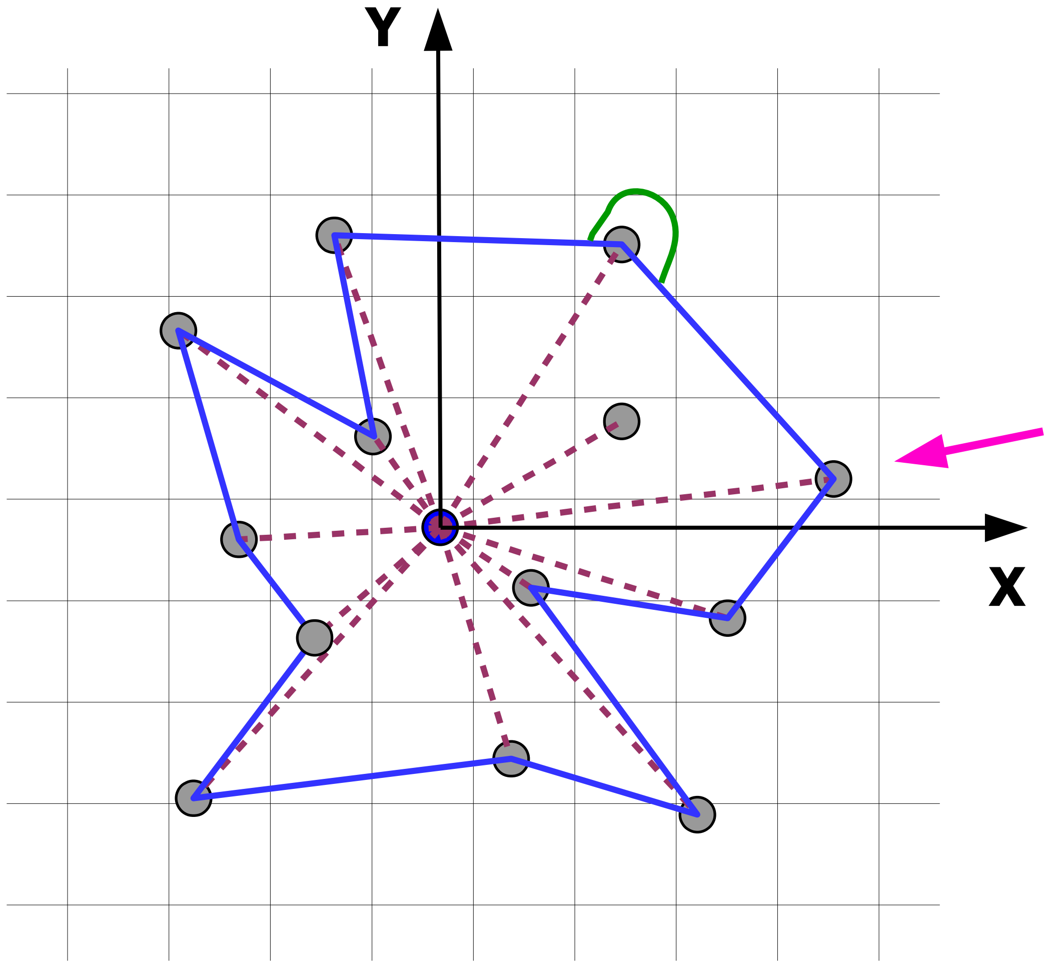


Graham scan

Complete invocation:

... 1,

... 2,



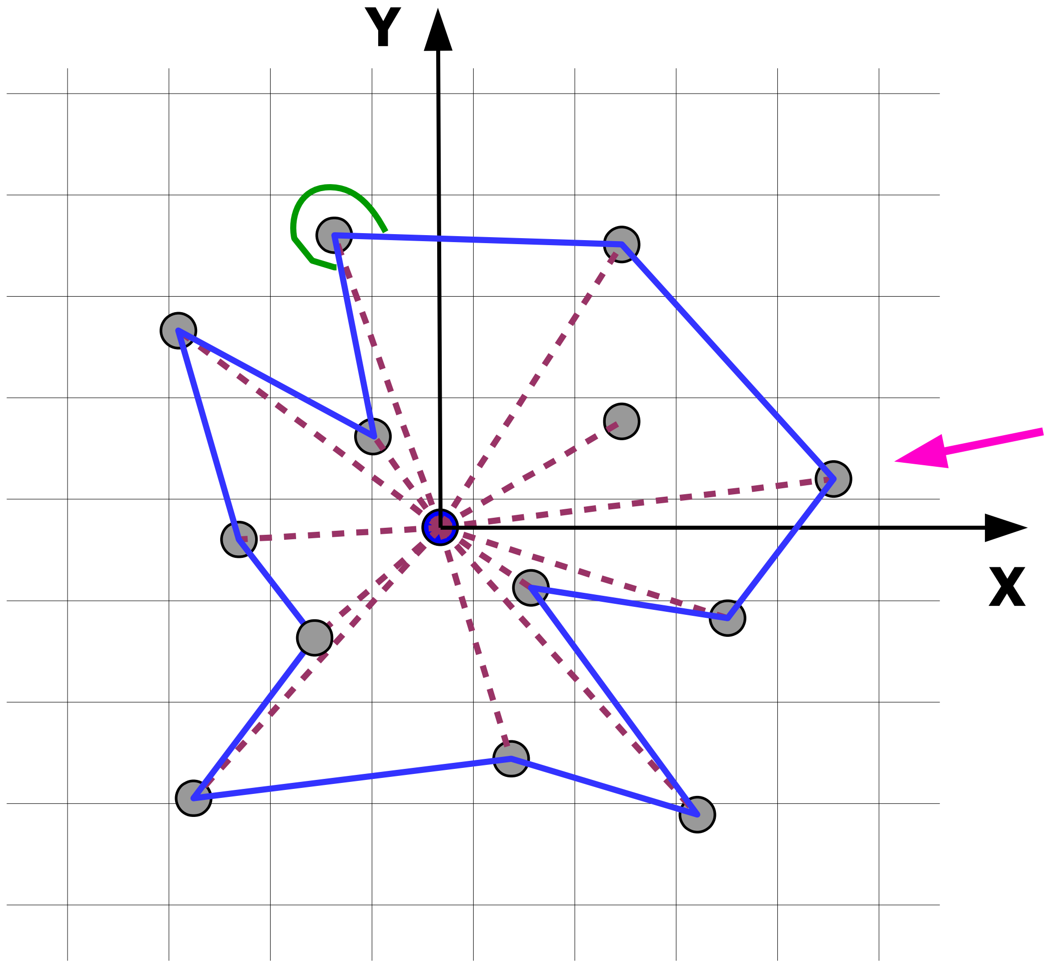
Graham scan

Complete invocation:

... 1,

... 2,

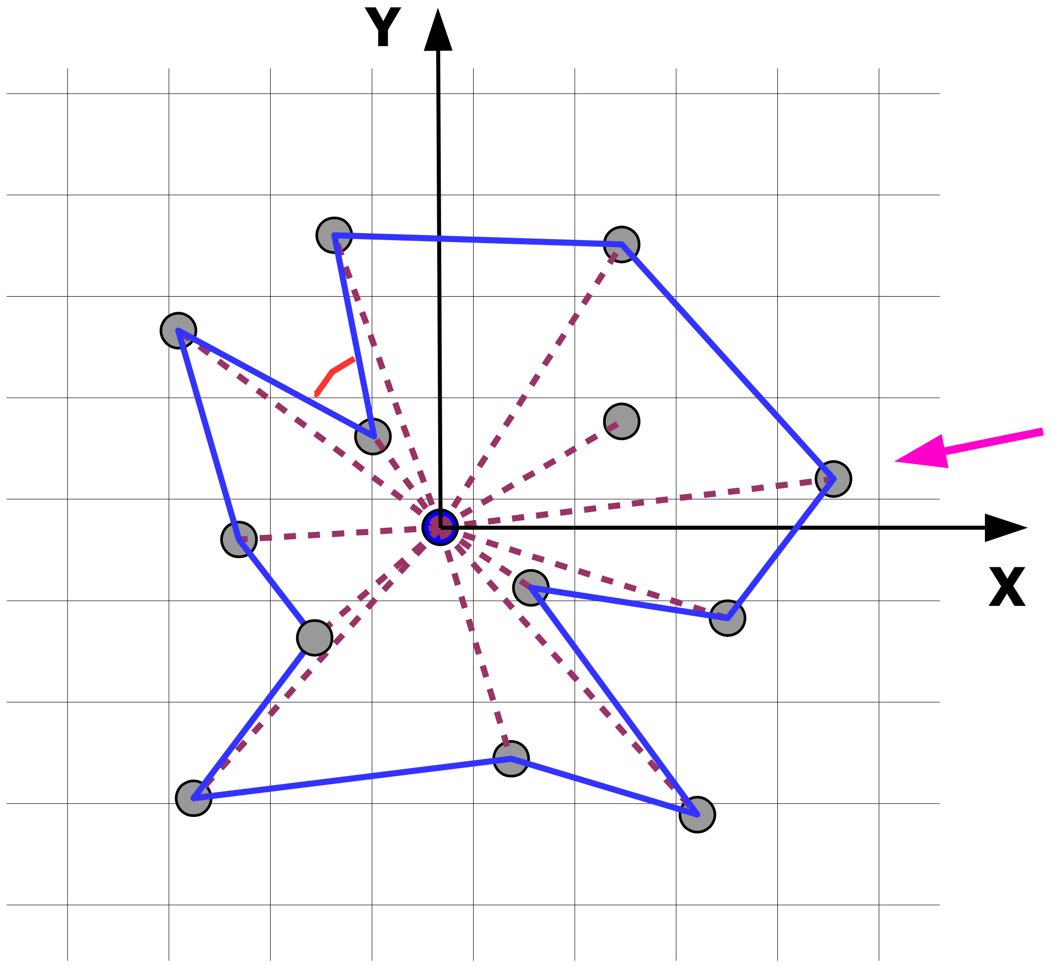
... 3,



Graham scan

Complete invocation:

- ... 1,
- ... 2,
- ... 3,
- ... 4,



Graham scan

Complete invocation:

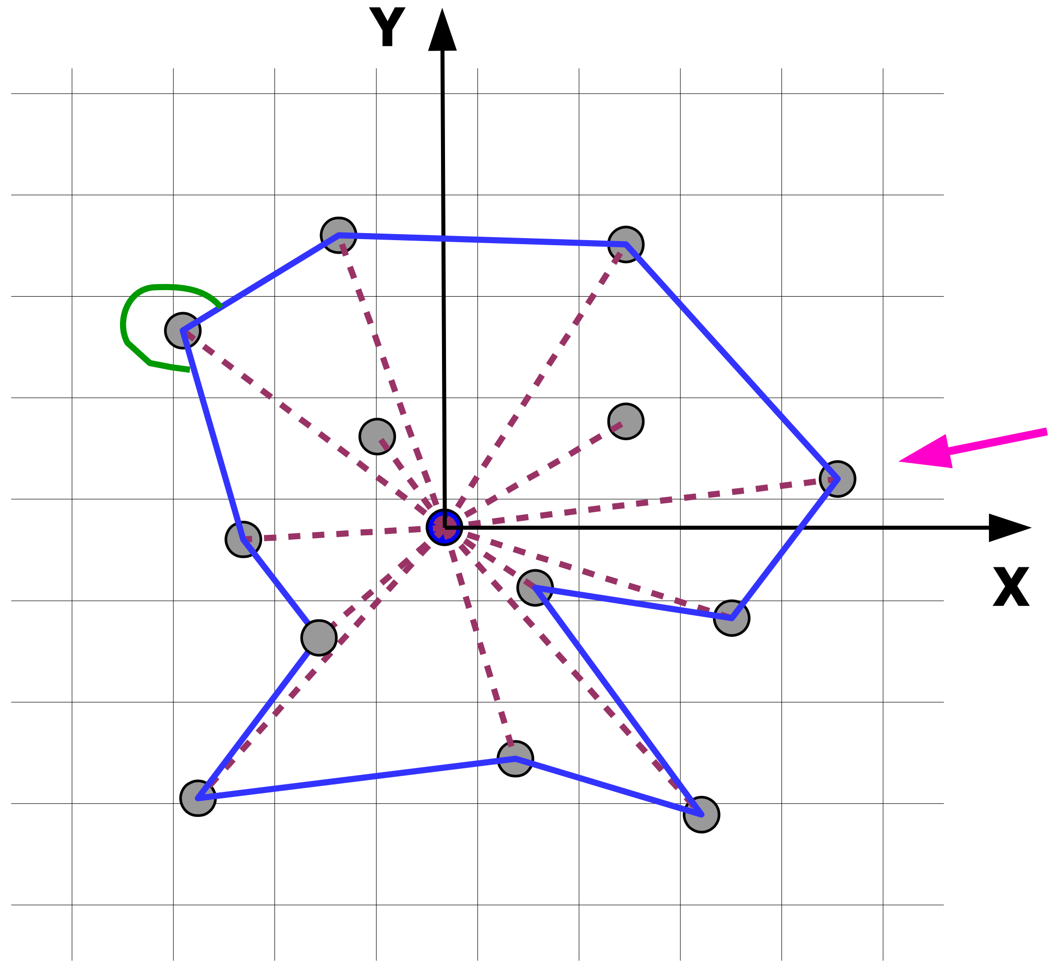
... 1,

... 2,

... 3,

... 4,

... 5,



Graham scan

Complete invocation:

... 1,

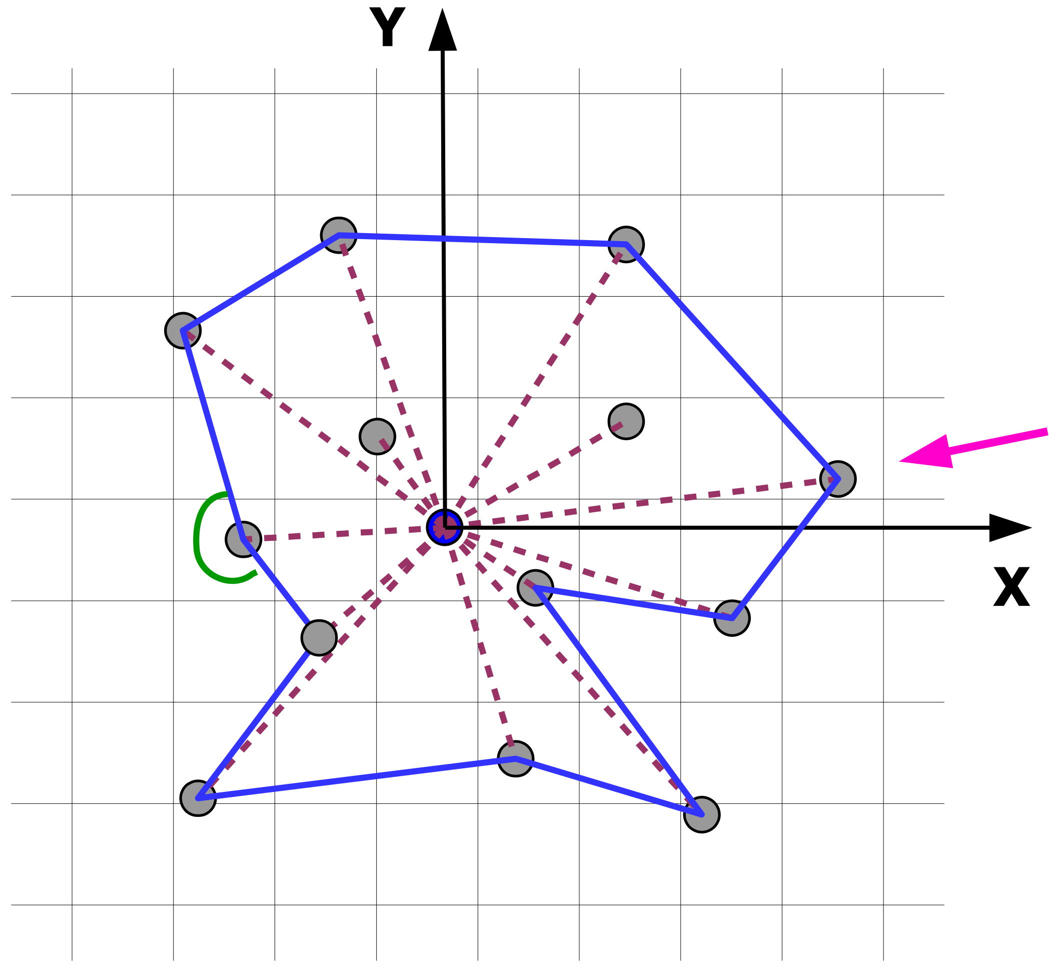
... 2,

... 3,

... 4,

... 5,

... 6,



Graham scan

Complete invocation:

... 1,

... 2,

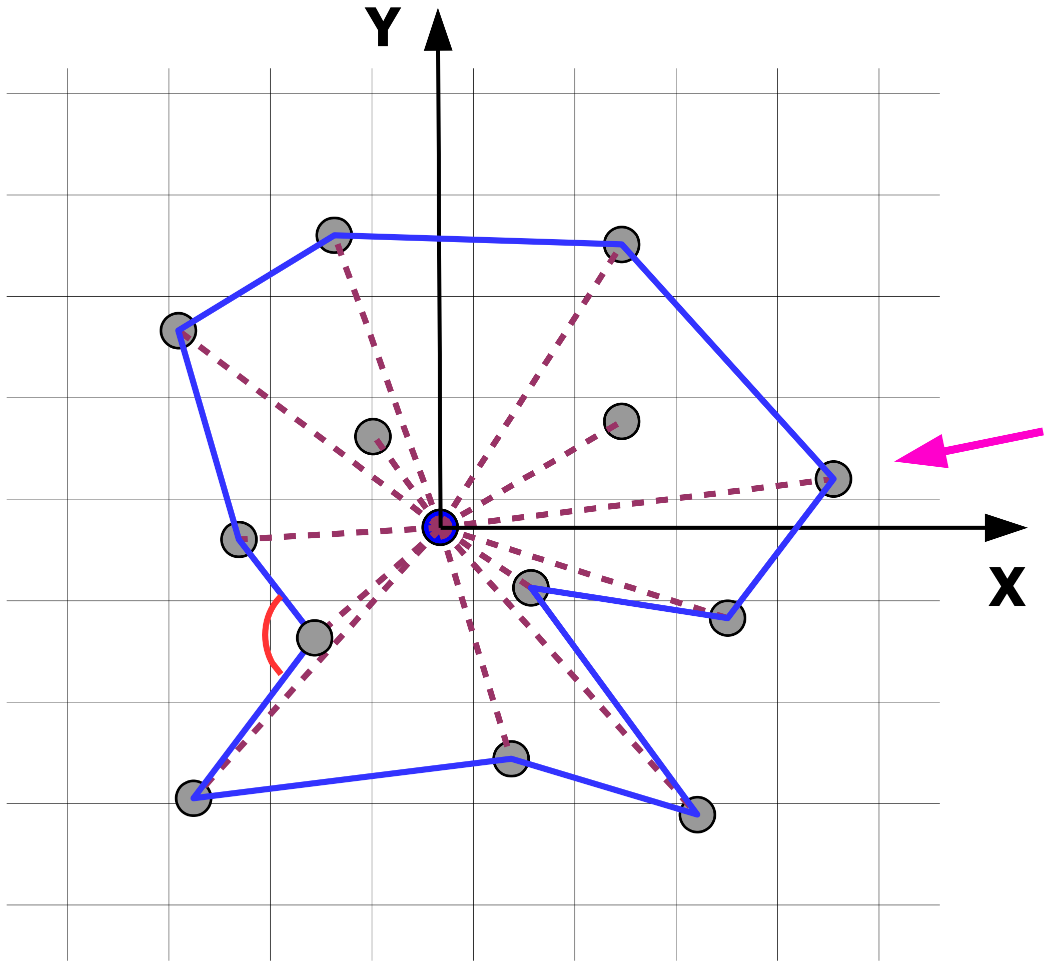
... 3,

... 4,

... 5,

... 6,

... 7,



Graham scan

Complete invocation:

... 1,

... 2,

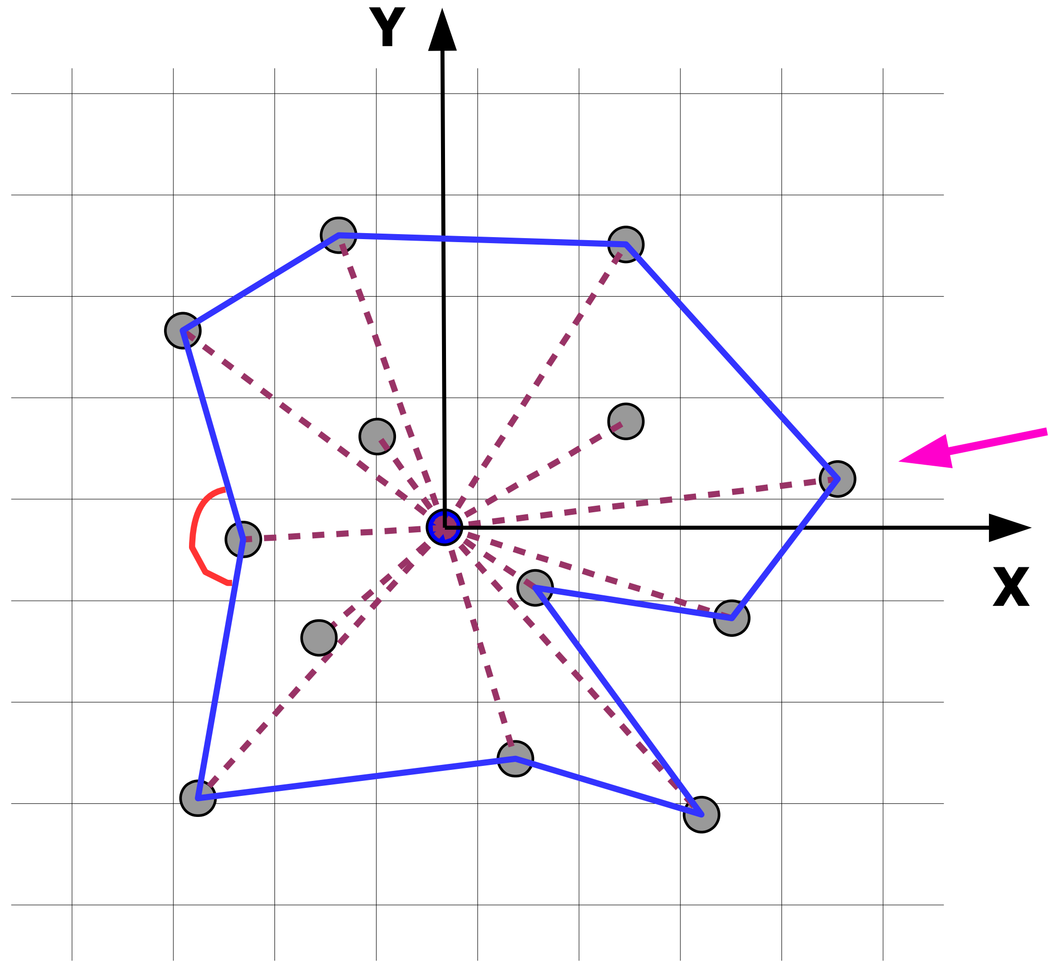
... 3,

... 4,

... 5,

... 6,

... 7,



Graham scan

Complete invocation:

... 1,

... 2,

... 3,

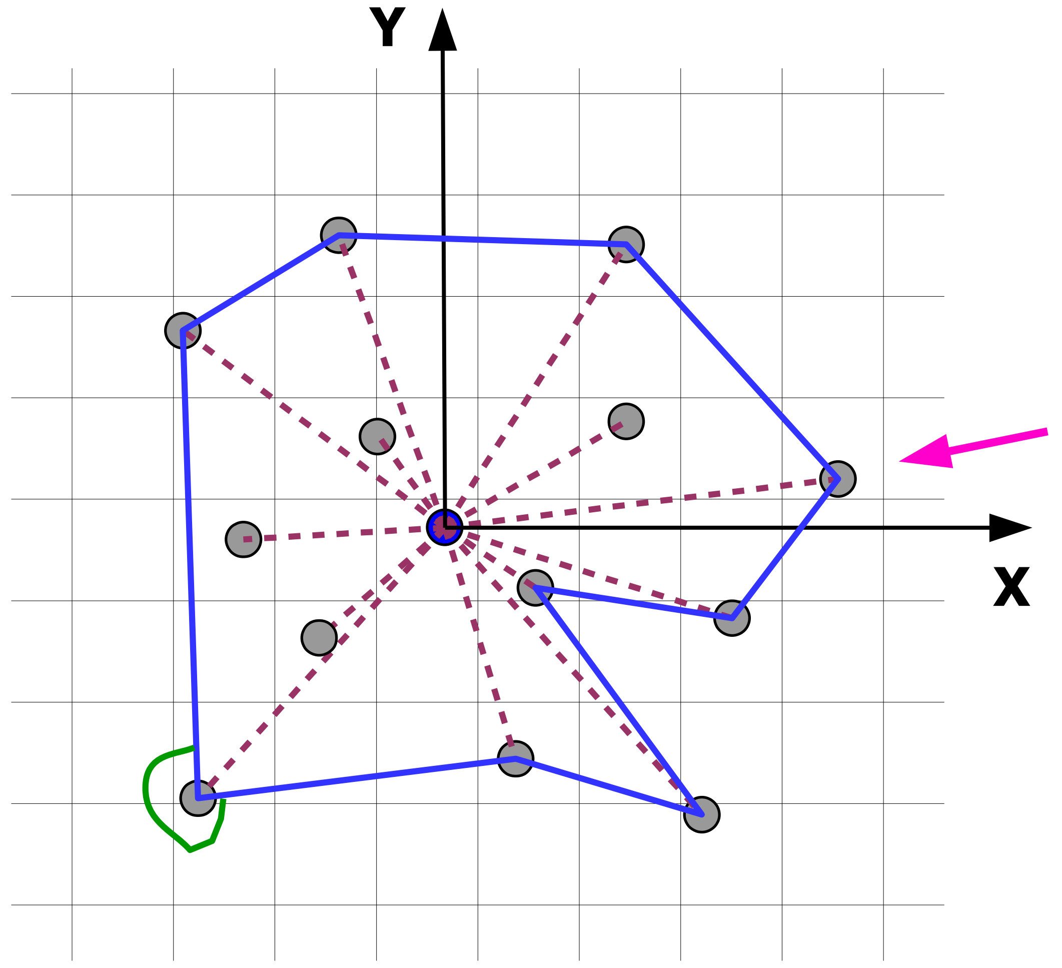
... 4,

... 5,

... 6,

... 7,

... 8,

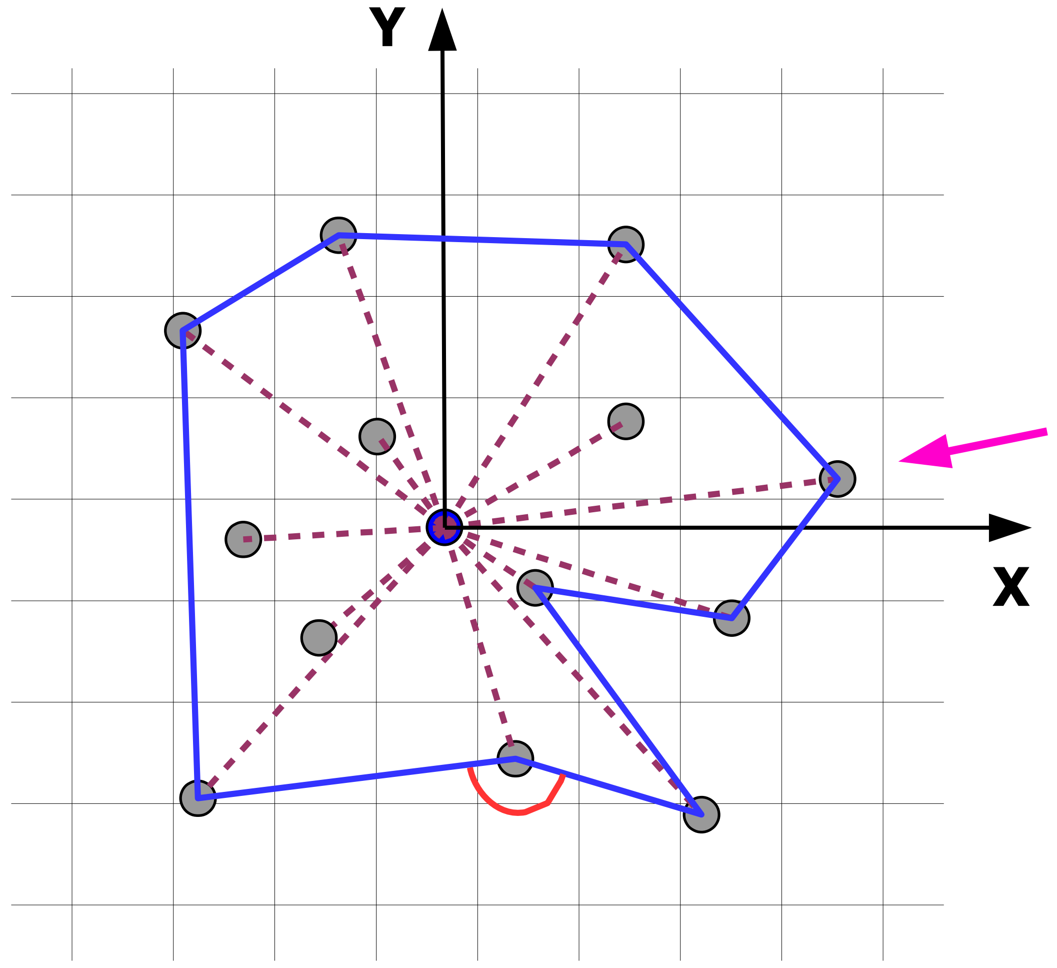


Graham scan

Complete invocation:

…

… 9,



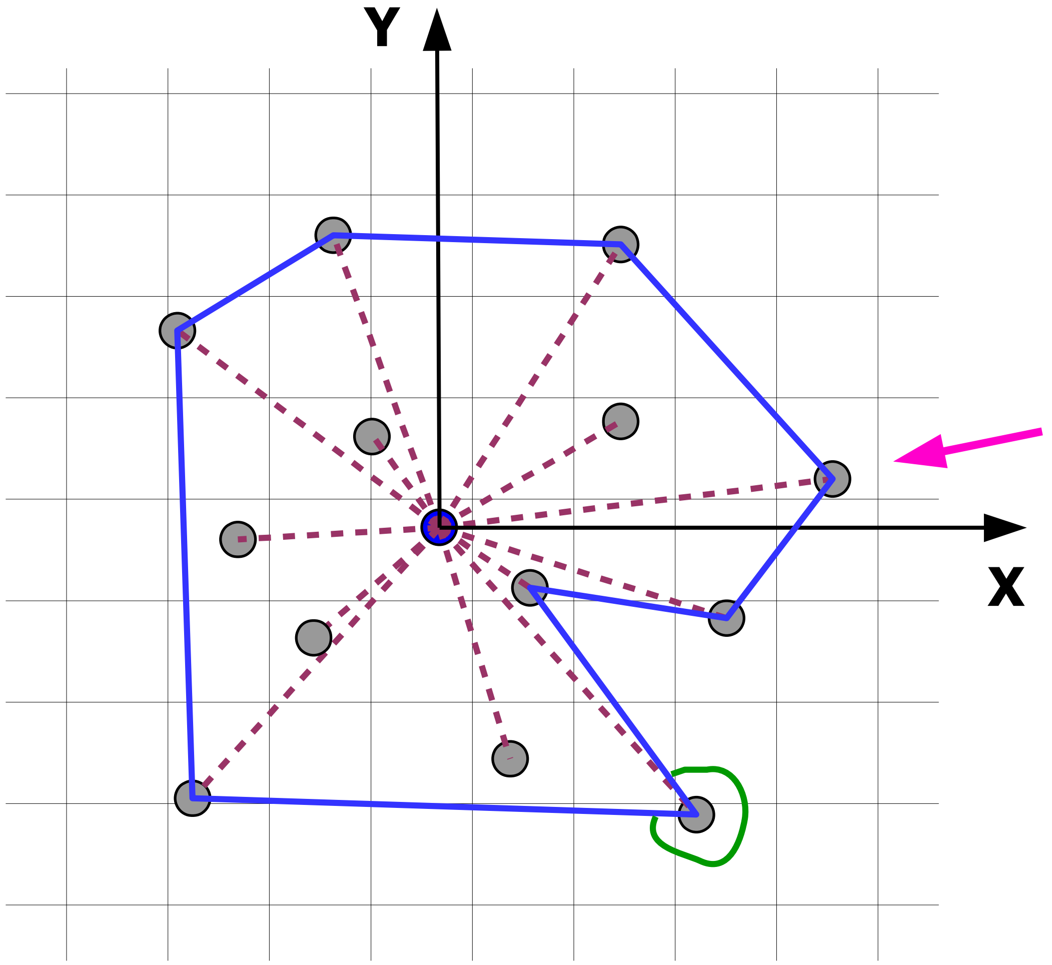
Graham scan

Complete invocation:

…

… 9,

… 10,



Graham scan

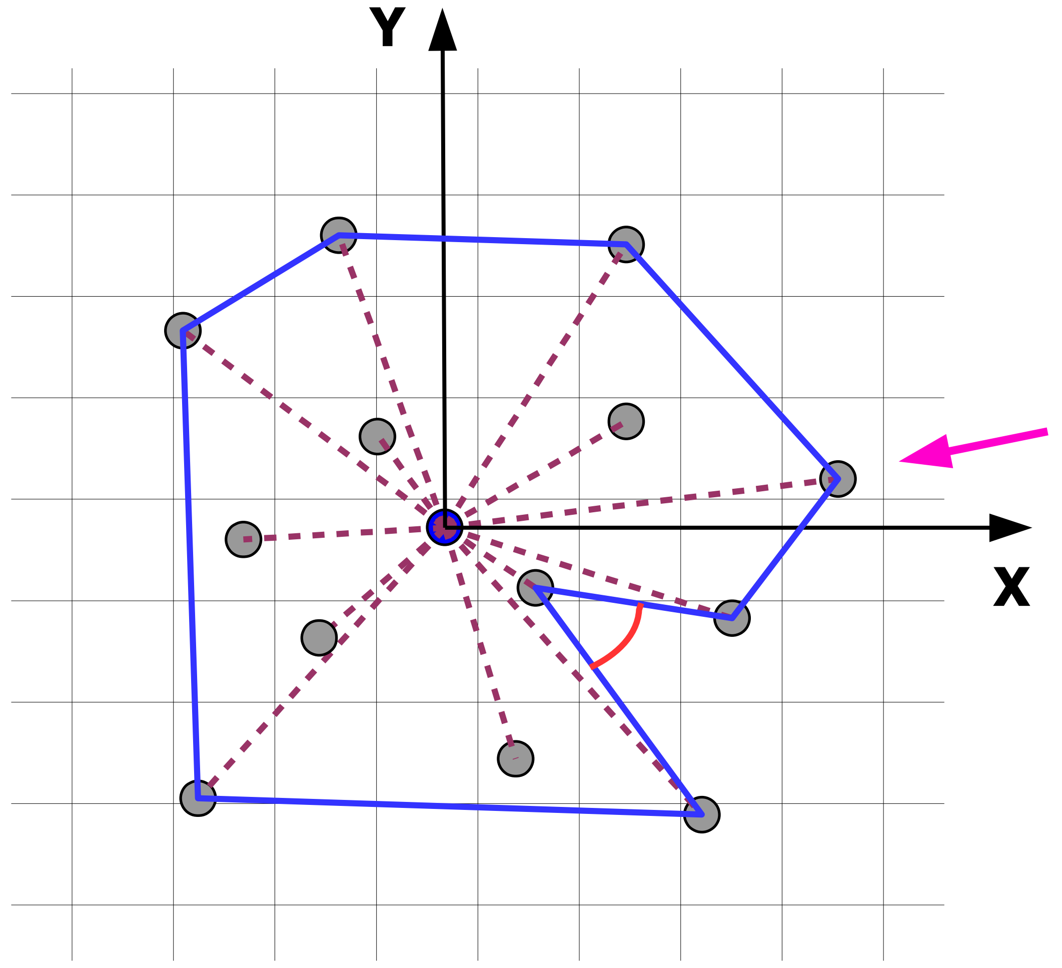
Complete invocation:

...

... 9,

... 10,

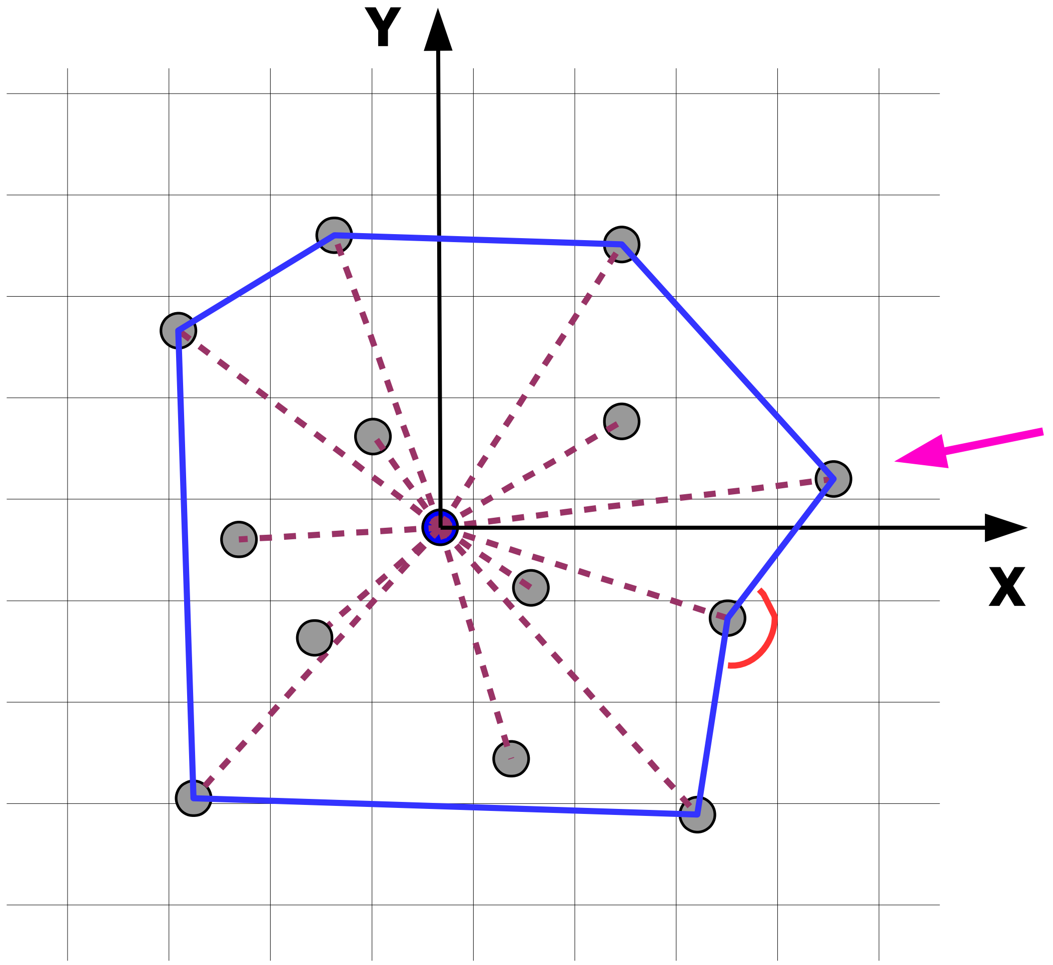
... 11,



Graham scan

Complete invocation:

- ...
- ... 9,
- ... 10,
- ... 11,
- ... 12,



Graham scan

Complete invocation:

...

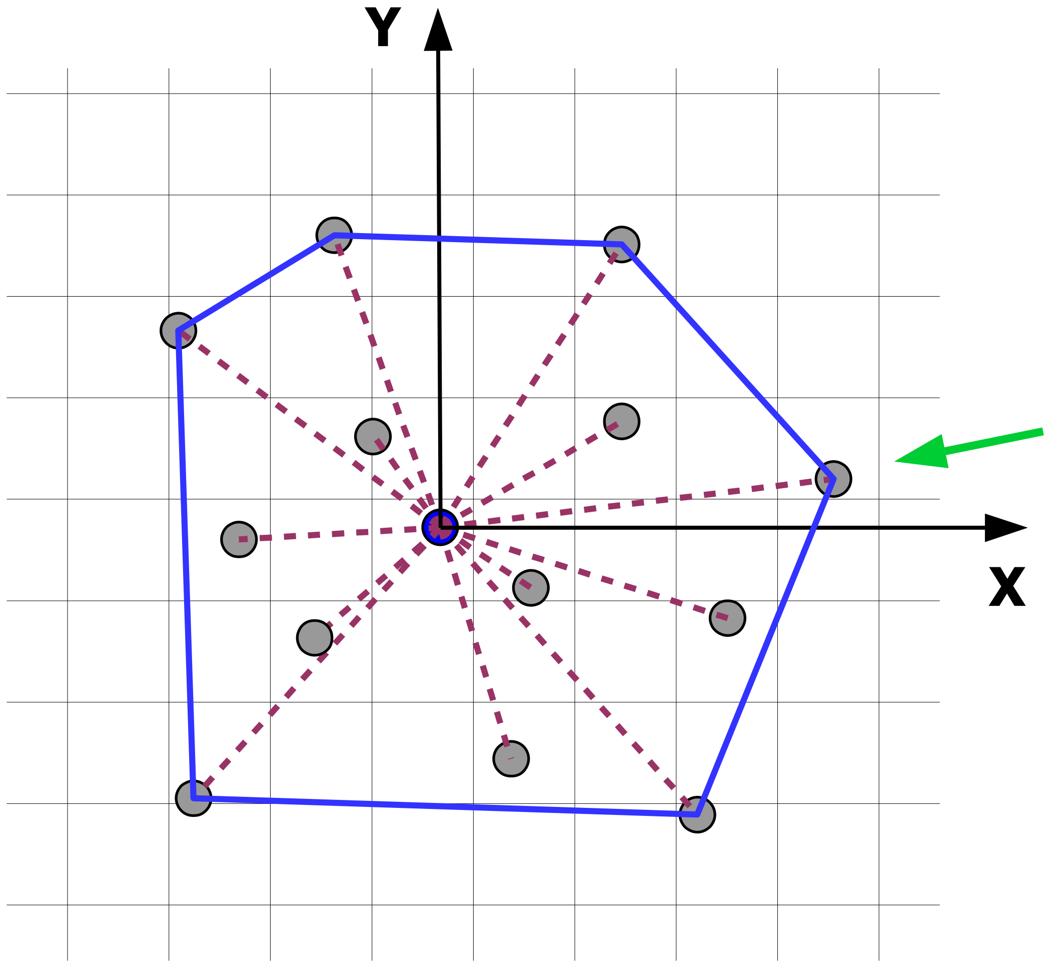
... 9,

... 10,

... 11,

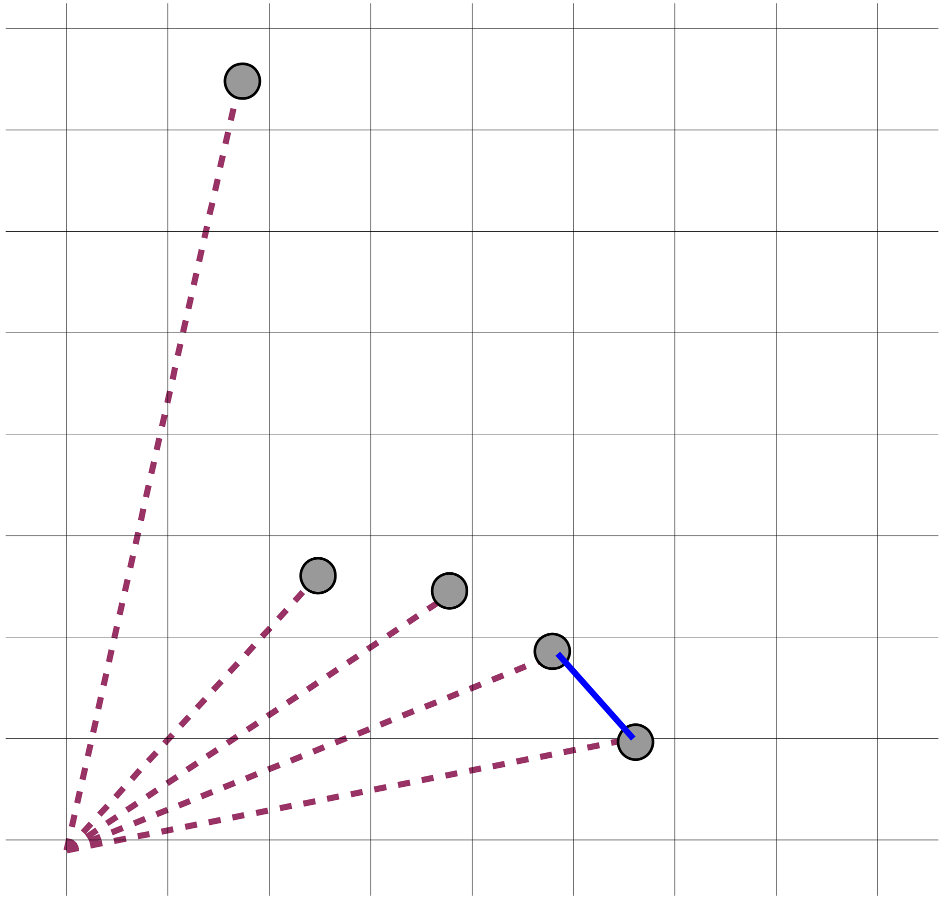
... 12,

... 13.



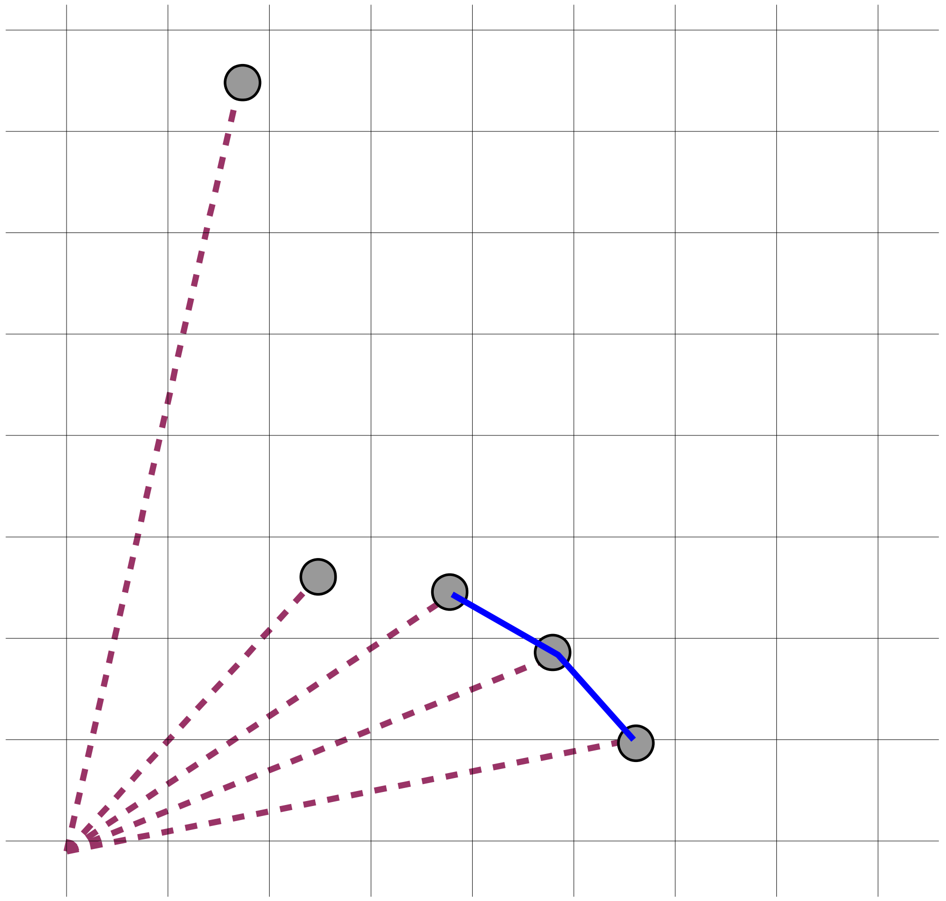
Graham scan

Note, there can be cases when during consideration of **1** next point, several points will be removed.



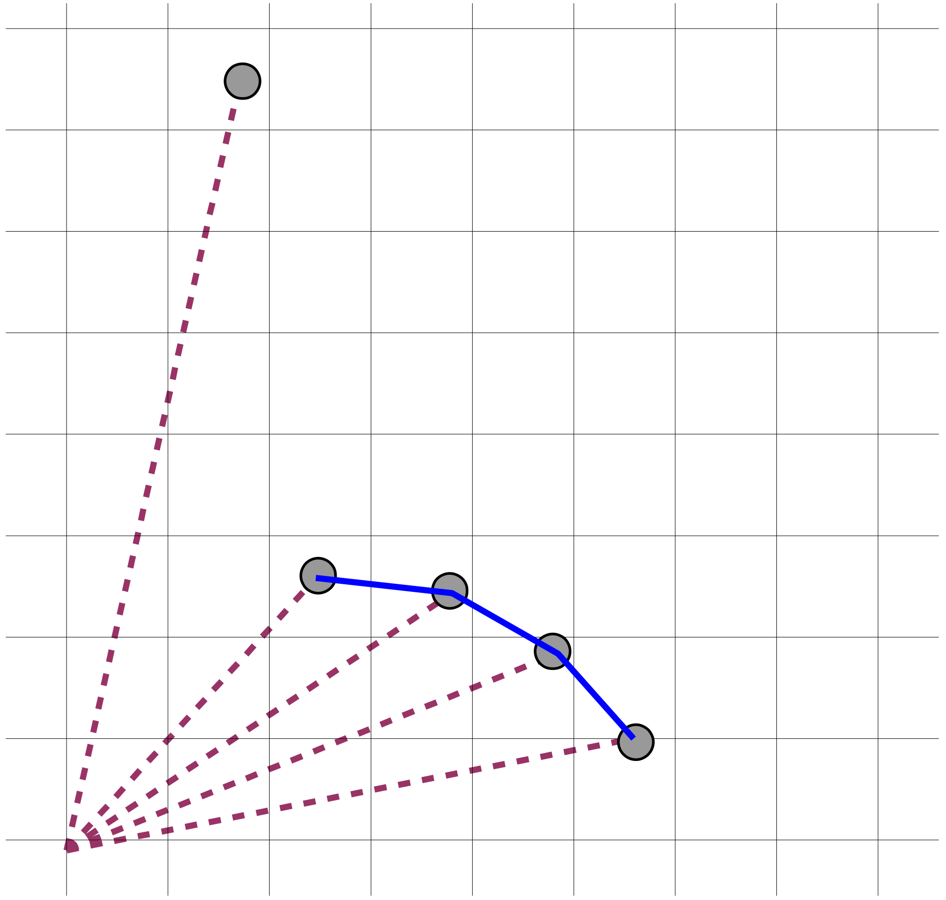
Graham scan

Note, there can be cases when during consideration of **1** next point, <u>several points will be removed</u>.



Graham scan

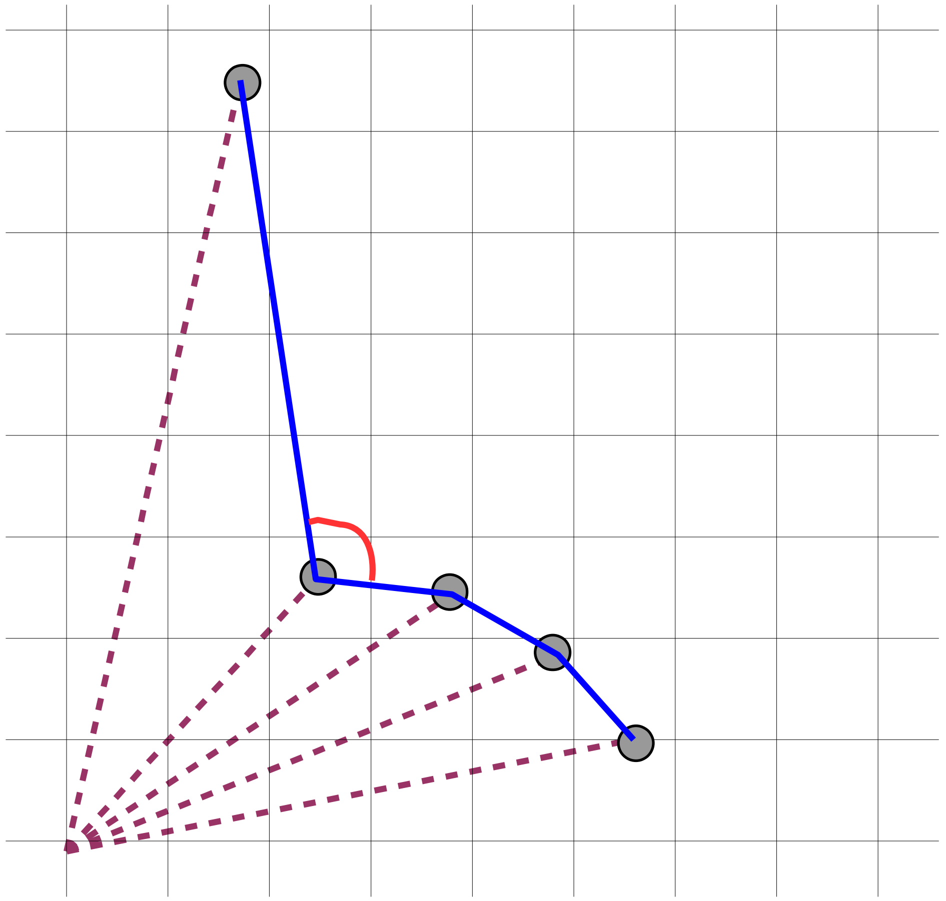
Note, there can be cases when during consideration of **1** next point, several points will be removed.



Graham scan

Note, there can be cases when during consideration of **1** next point, <u>several points will be removed</u>.

... one right turn,

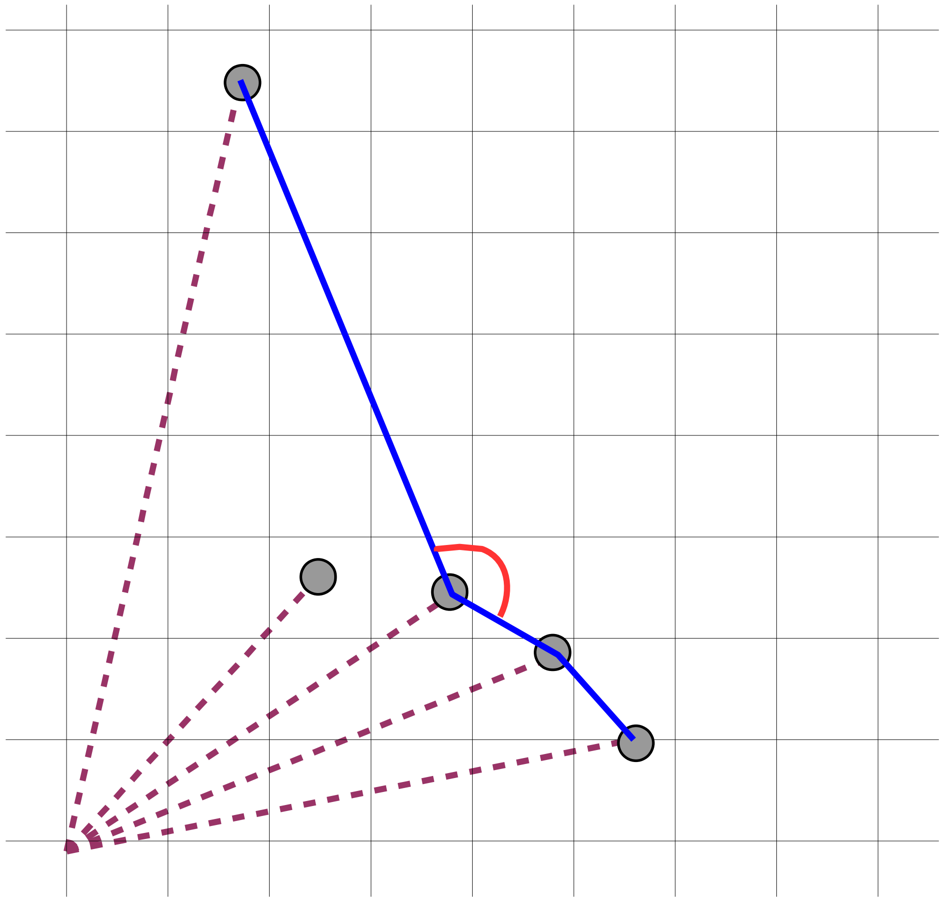


Graham scan

Note, there can be cases when during consideration of **1** next point, several points will be removed.

... one right turn,

... second right turn,



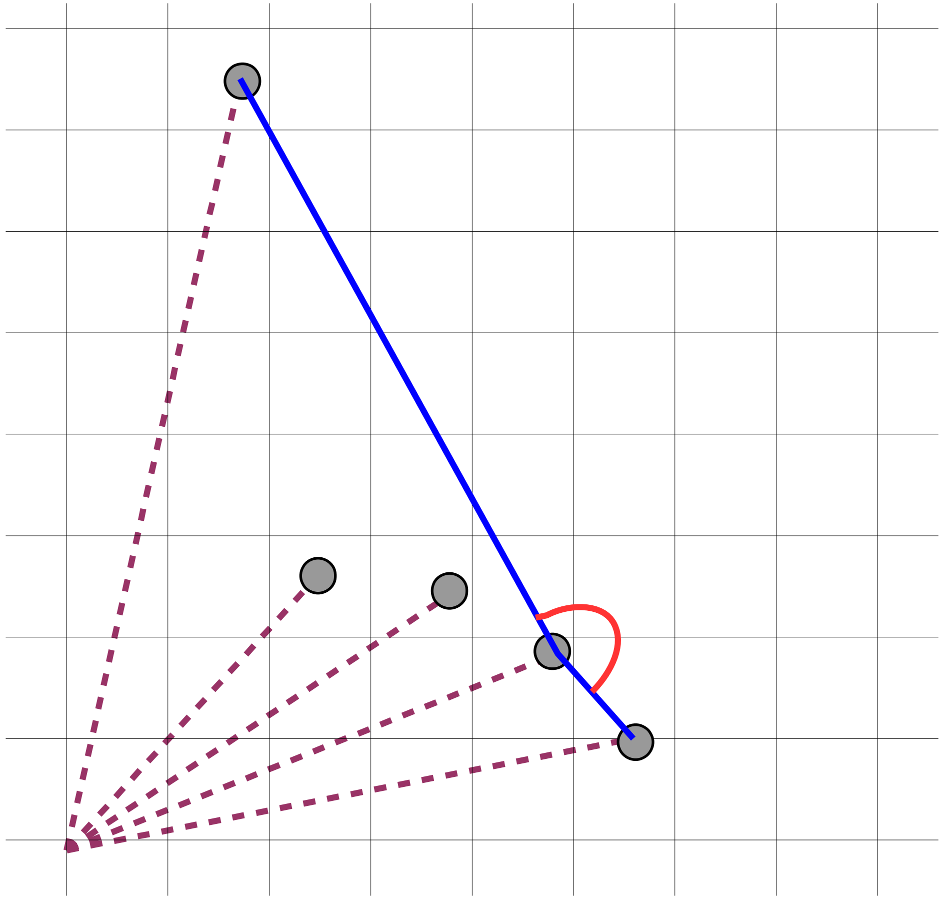
Graham scan

Note, there can be cases when during consideration of **1** next point, <u>several points will be removed</u>.

... one right turn,

... second right turn,

... third right turn.



Graham scan

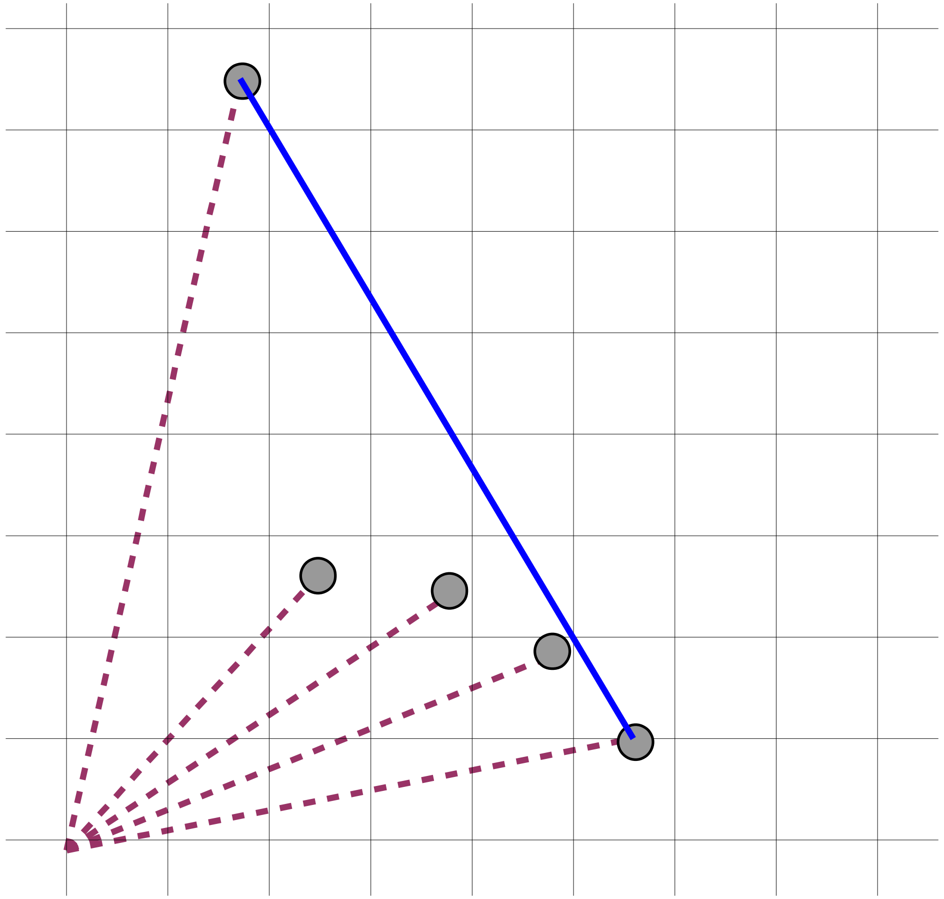
Note, there can be cases when during consideration of **1** next point, <u>several points will be removed</u>.

... one right turn,

... second right turn,

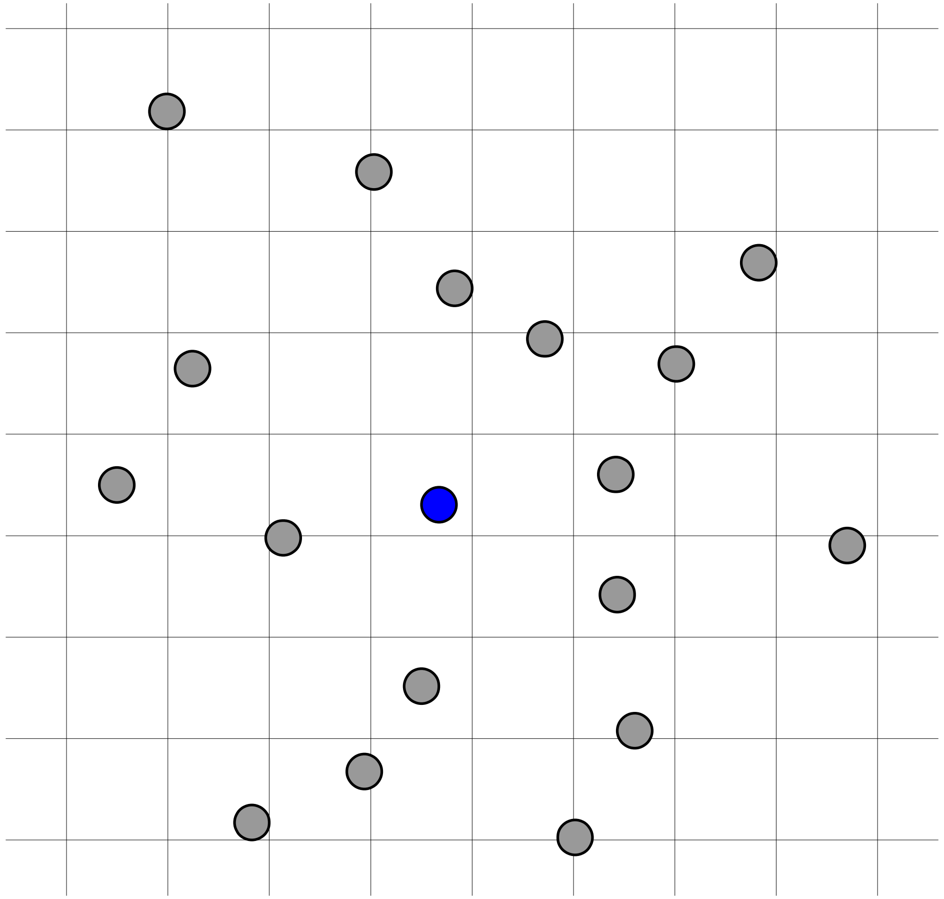
... third right turn.

... resolved.



Exercise

Perform Graham scan on the following set of points.

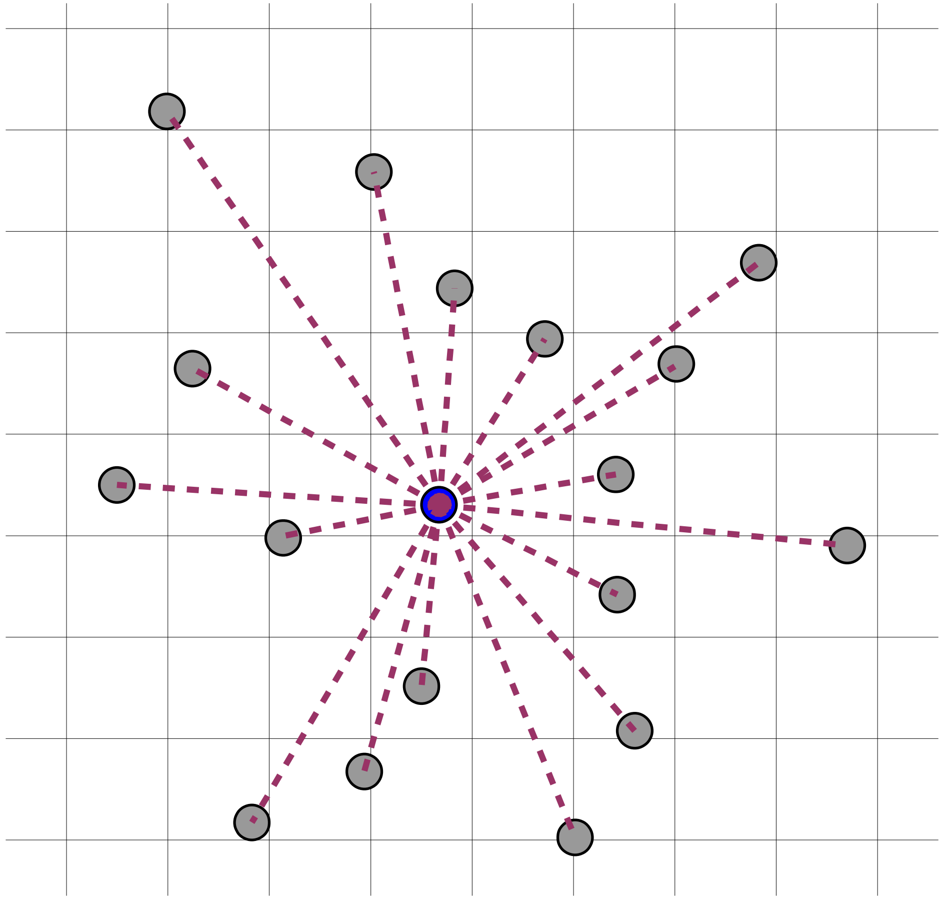


Exercise

(solution)

Step 1,

Step 2,

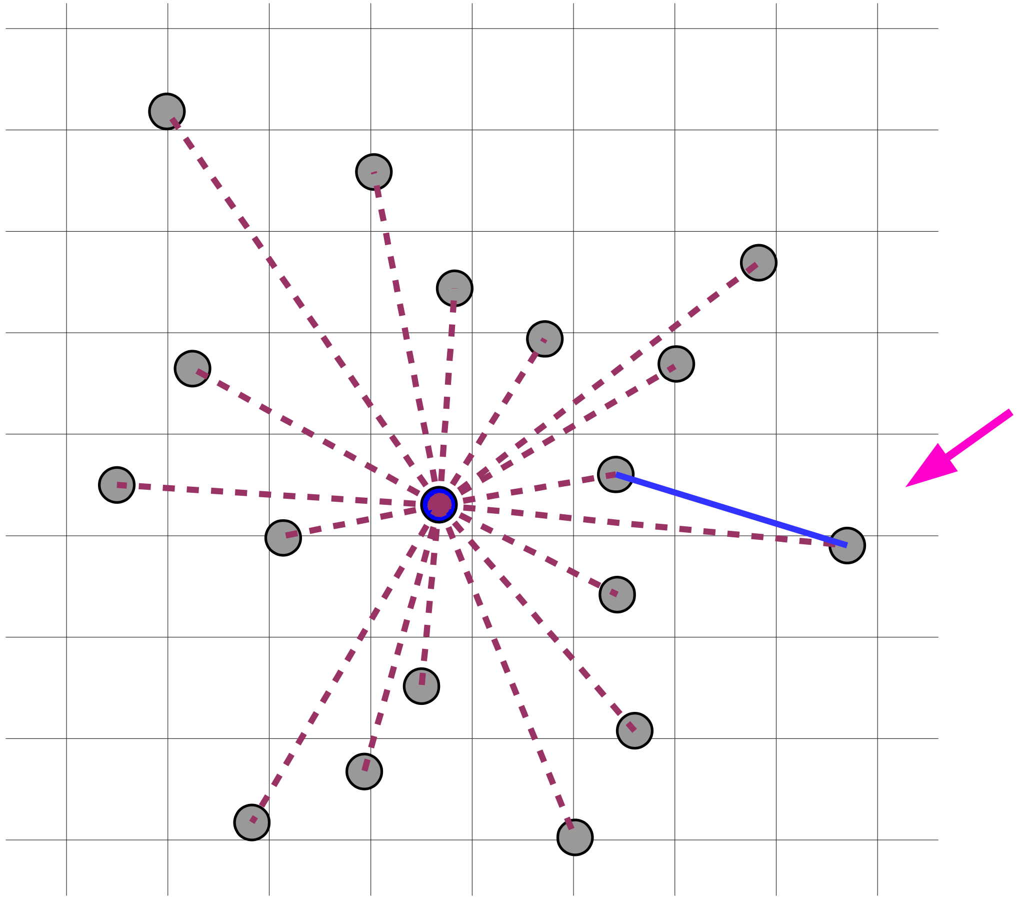


Exercise
(solution)

Step 1,

Step 2,

Step 3 ...

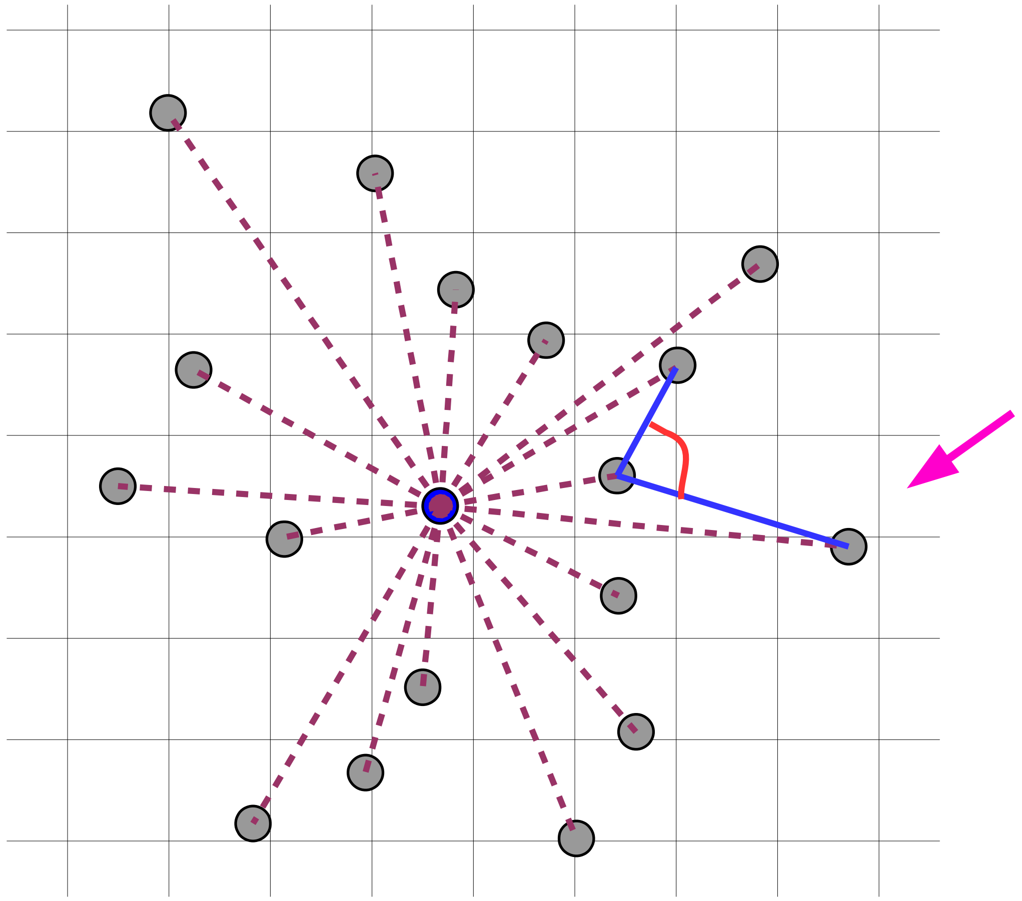


Exercise
(solution)

Step 1,

Step 2,

Step 3 ...

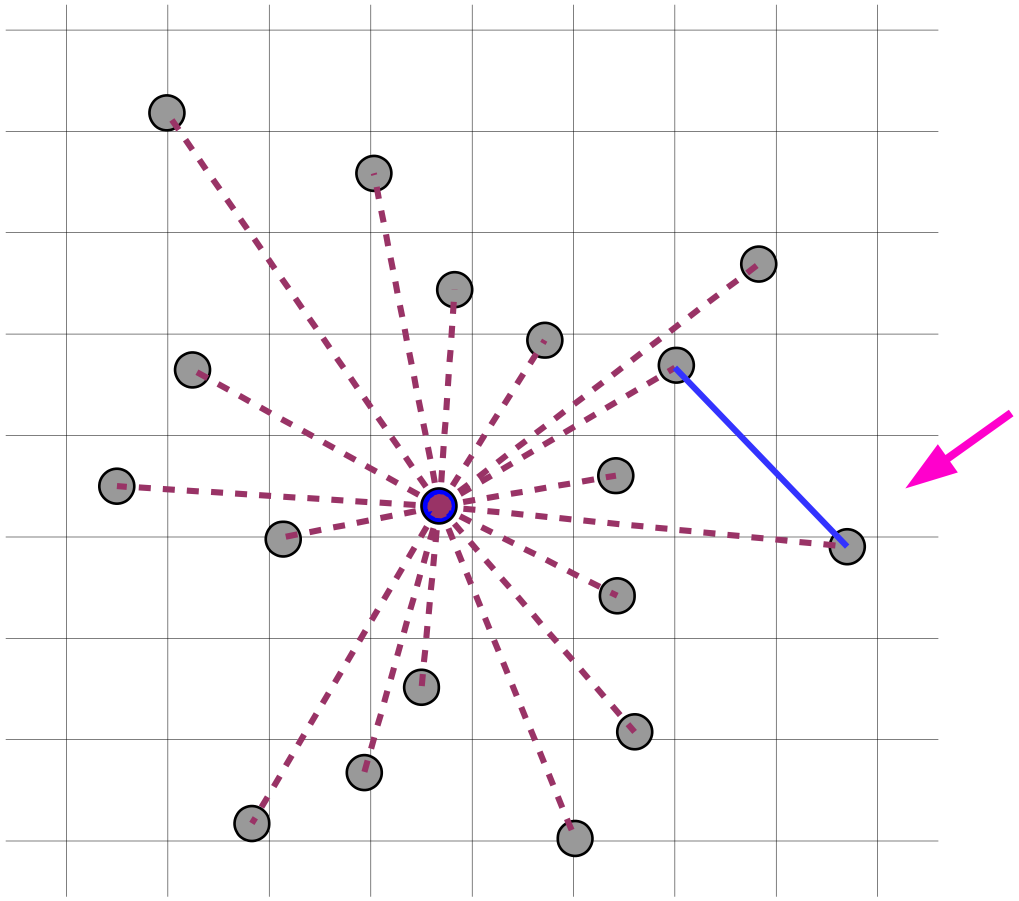


Exercise
(solution)

Step 1,

Step 2,

Step 3 ...

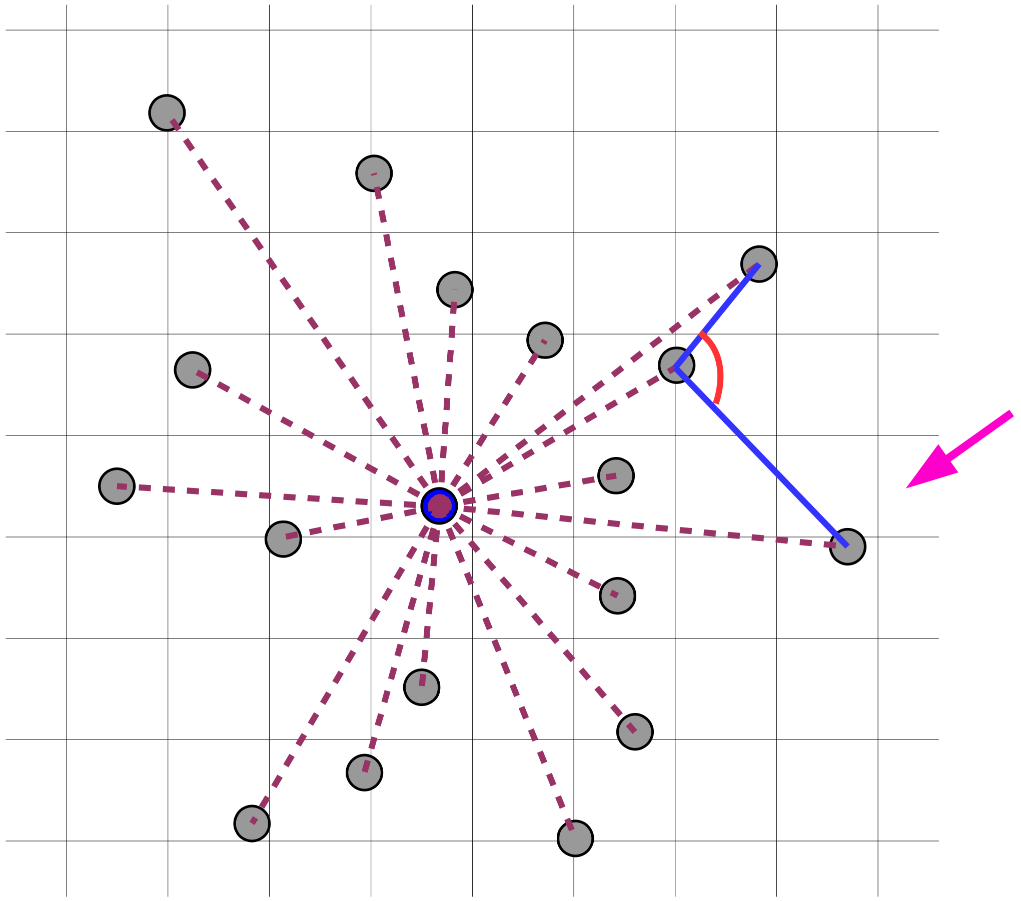


Exercise
(solution)

Step 1,

Step 2,

Step 3 ...



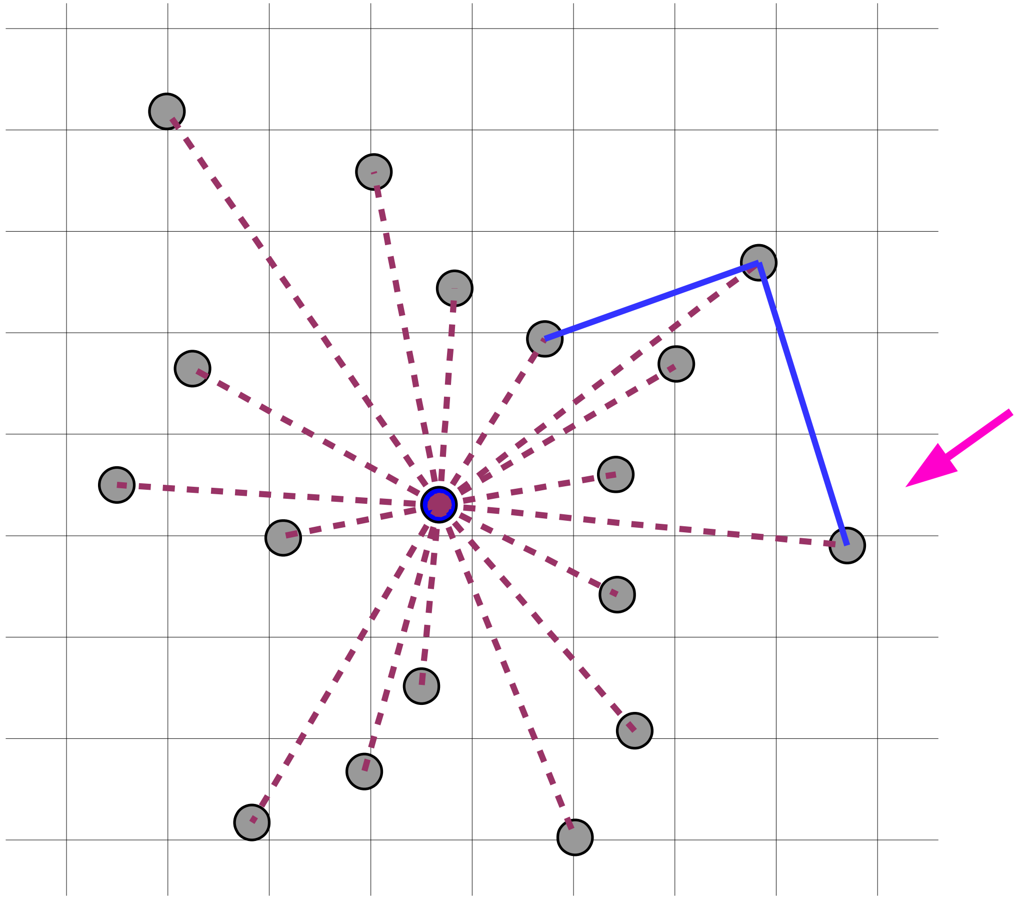
Exercise

(solution)

Step 1,

Step 2,

Step 3 ...

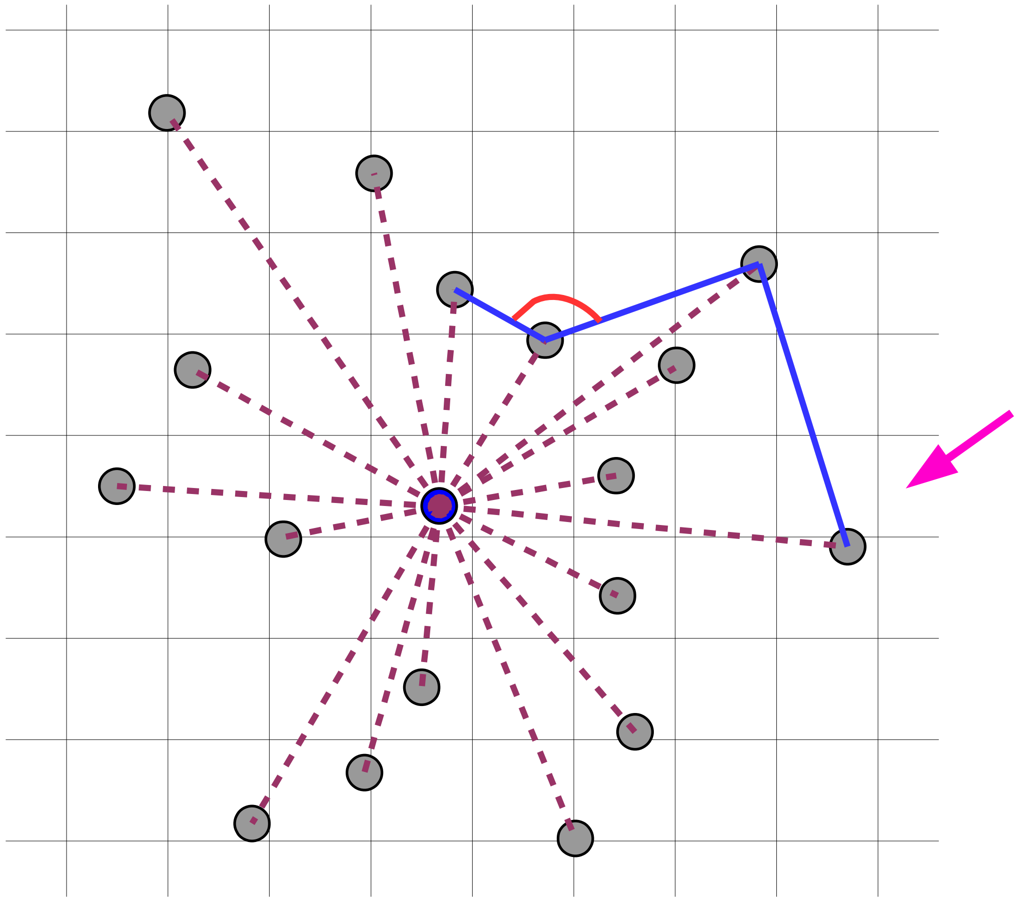


Exercise
(solution)

Step 1,

Step 2,

Step 3 ...

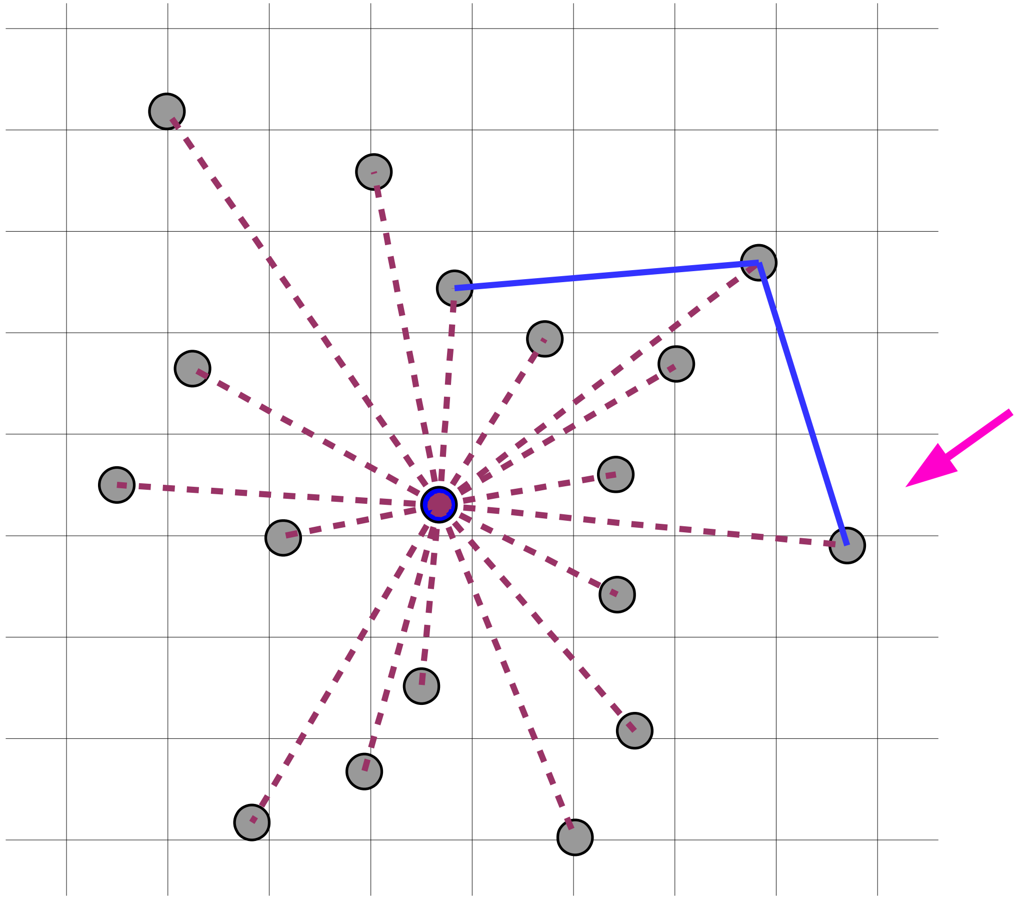


Exercise
(solution)

Step 1,

Step 2,

Step 3 ...

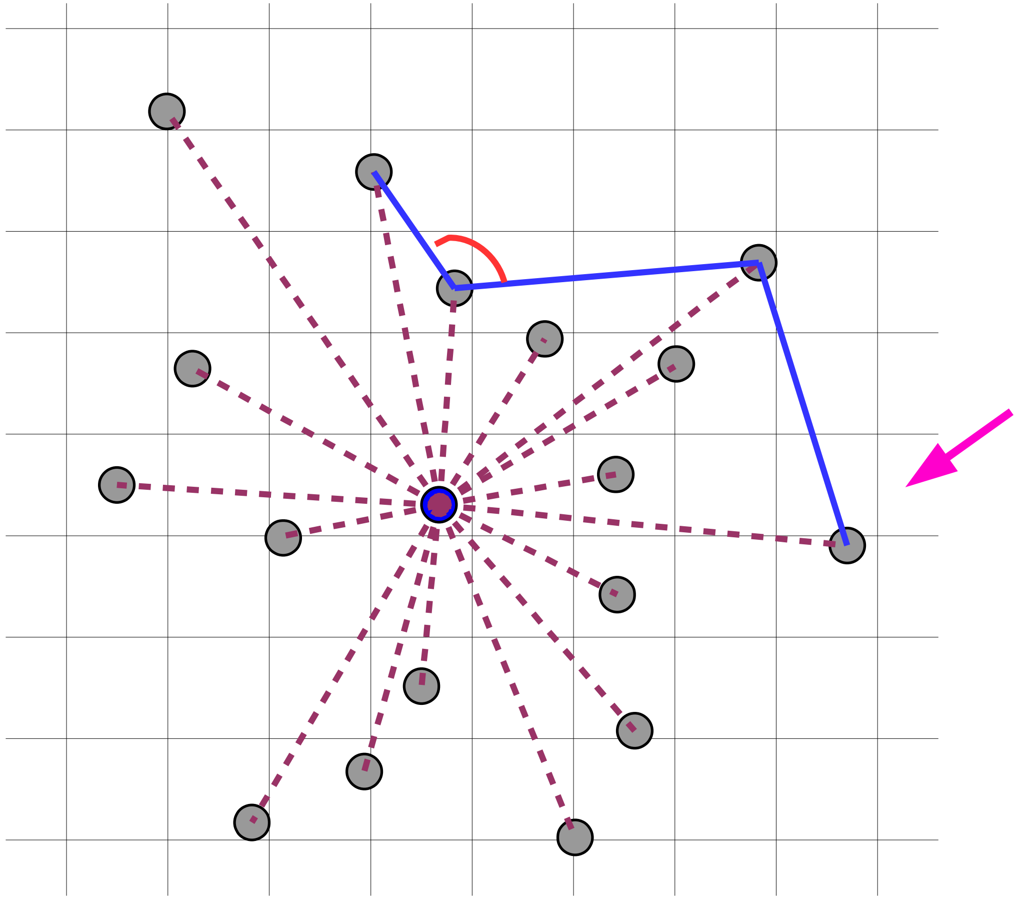


Exercise
(solution)

Step 1,

Step 2,

Step 3 ...

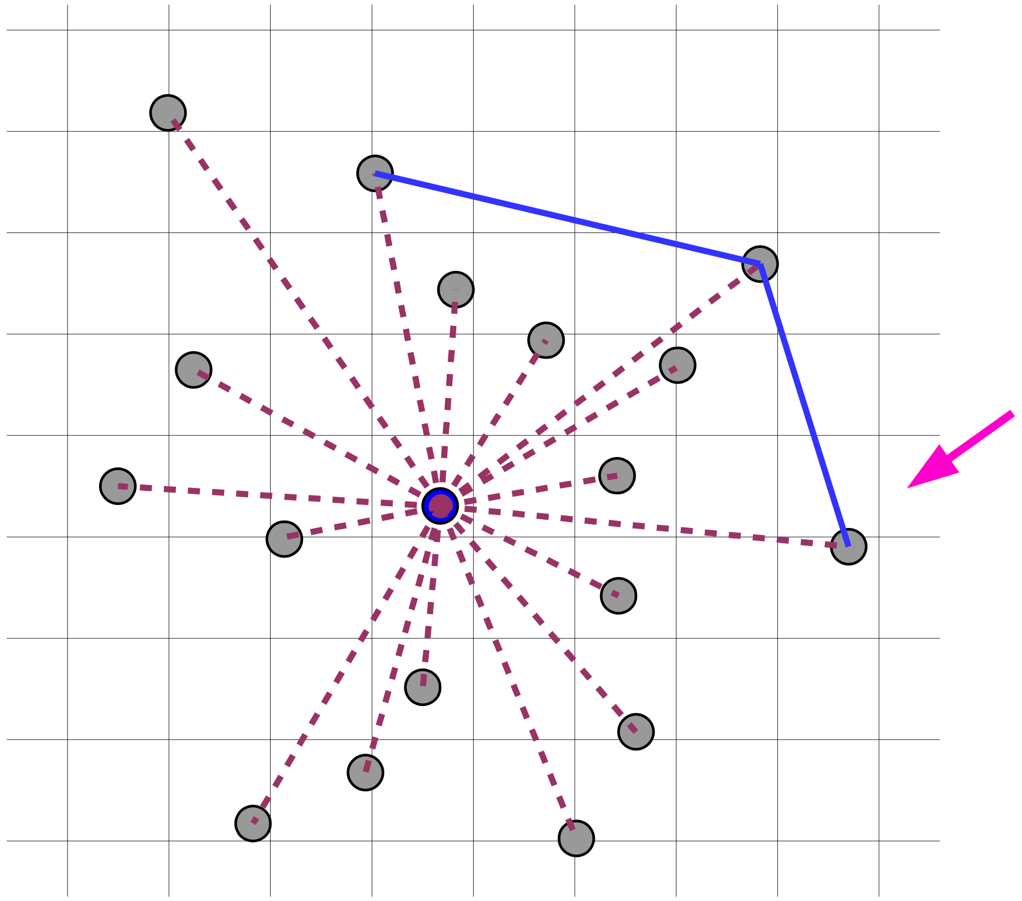


Exercise
(solution)

Step 1,

Step 2,

Step 3 ...



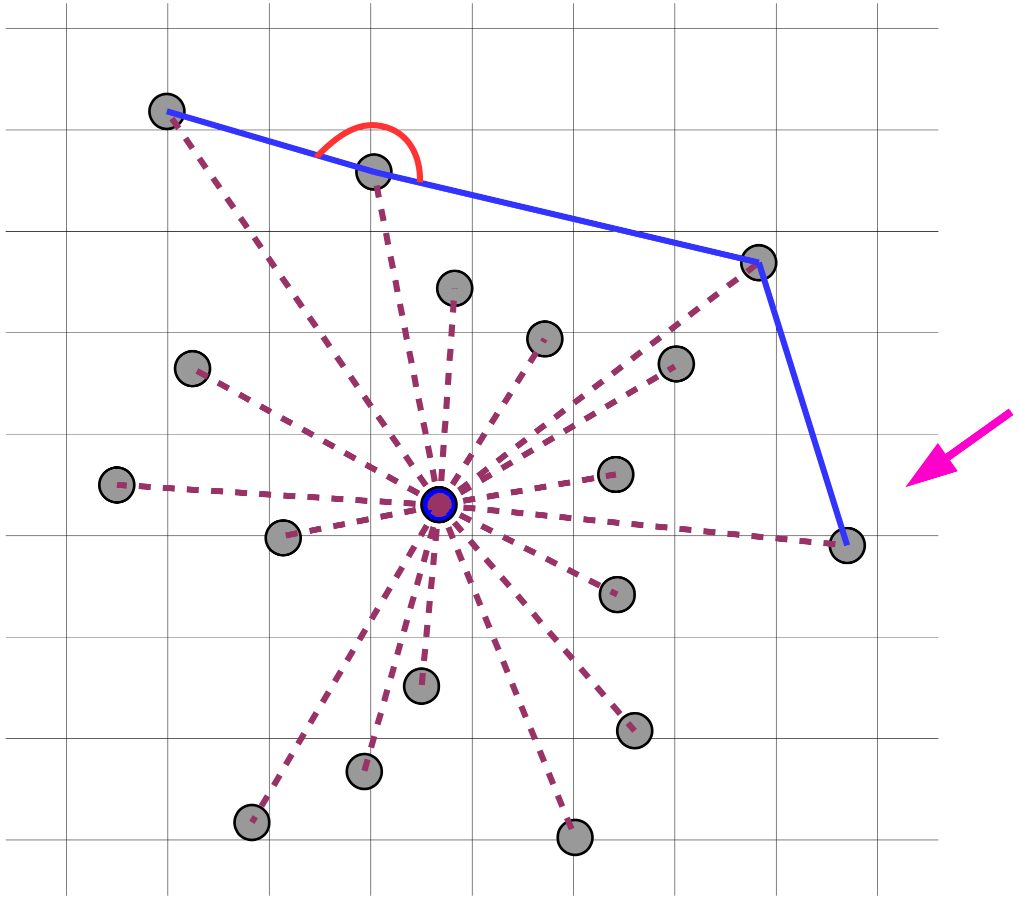
Exercise

(solution)

Step 1,

Step 2,

Step 3 ...

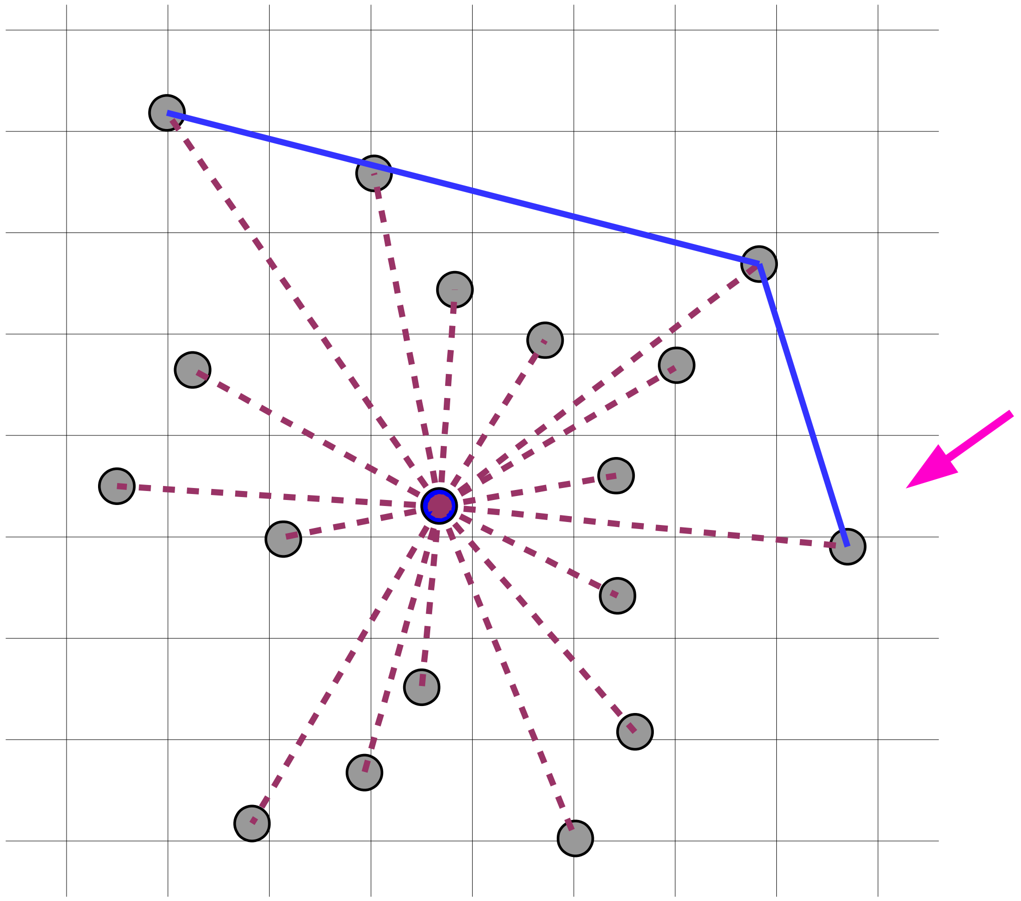


Exercise
(solution)

Step 1,

Step 2,

Step 3 ...



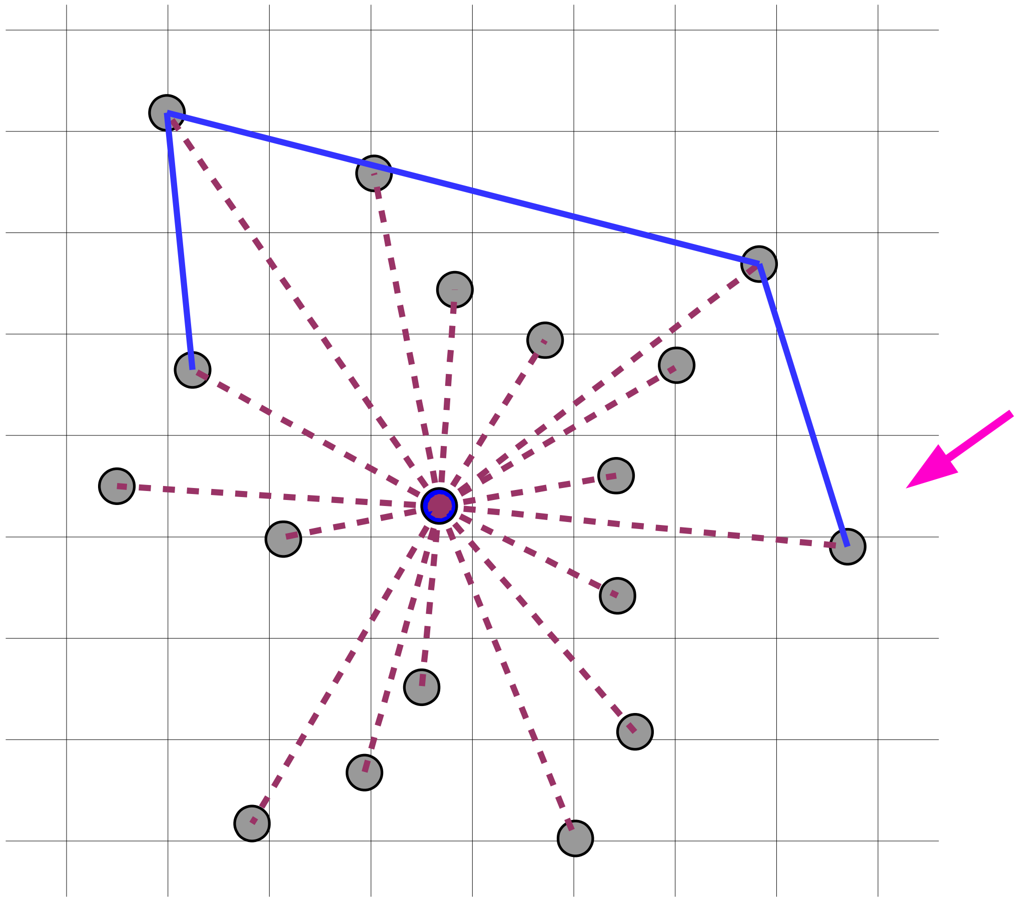
Exercise

(solution)

Step 1,

Step 2,

Step 3 ...



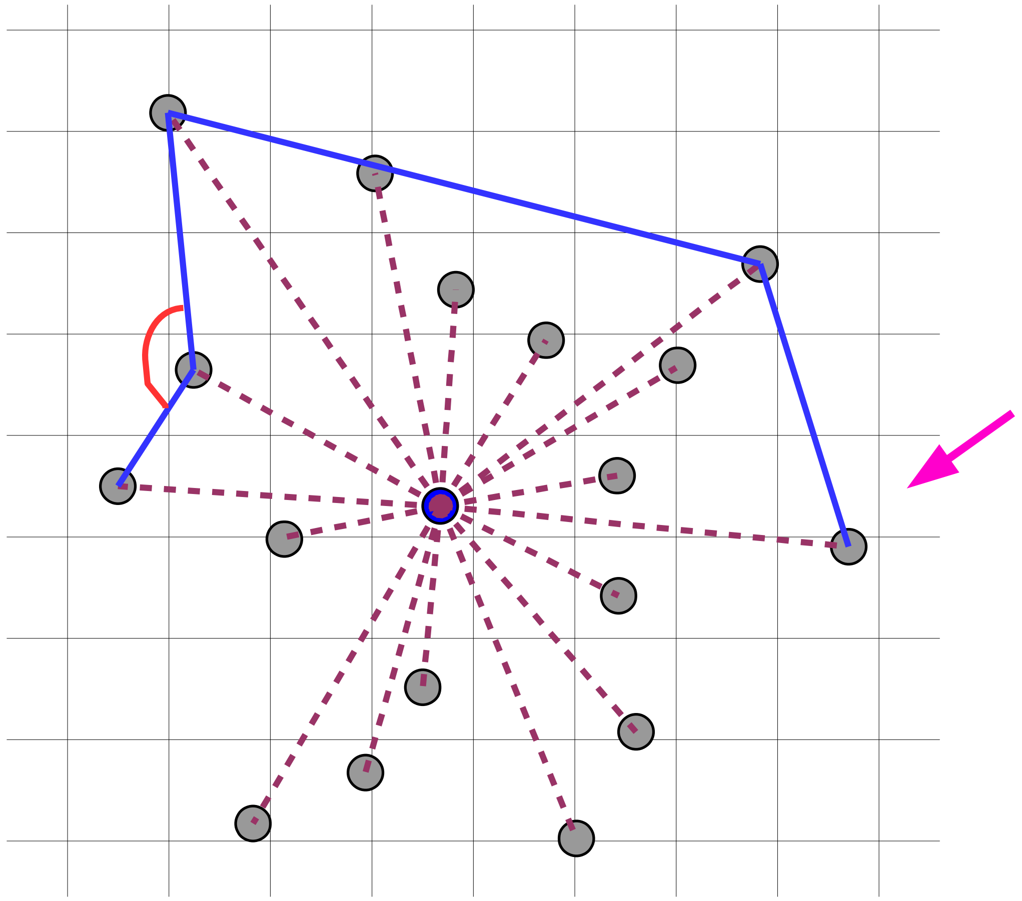
Exercise

(solution)

Step 1,

Step 2,

Step 3 ...



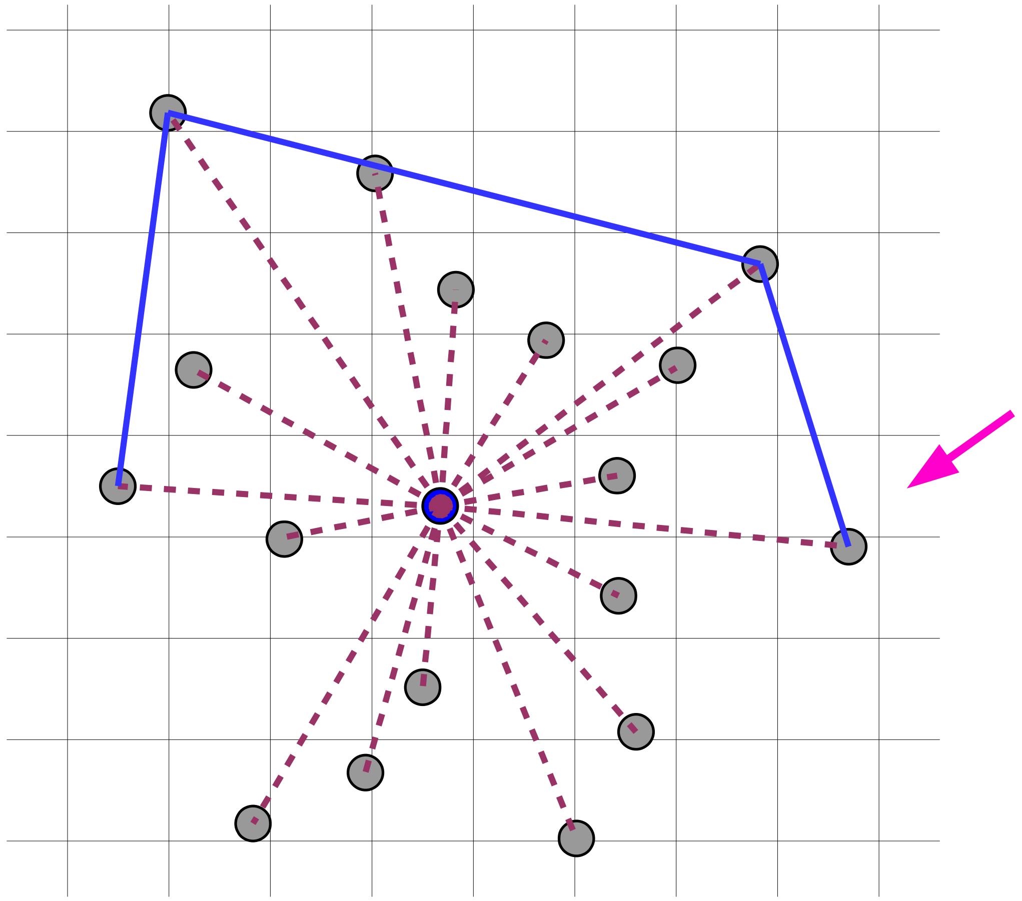
Exercise

(solution)

Step 1,

Step 2,

Step 3 ...



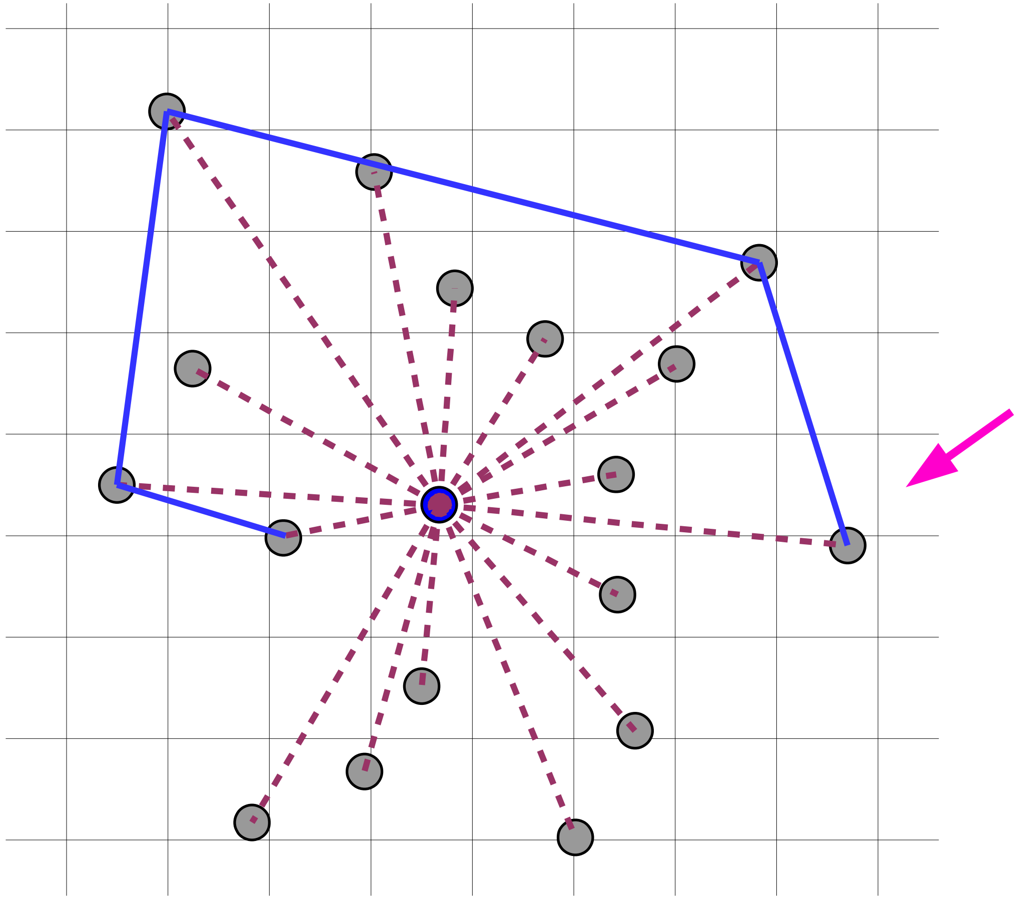
Exercise

(solution)

Step 1,

Step 2,

Step 3 ...

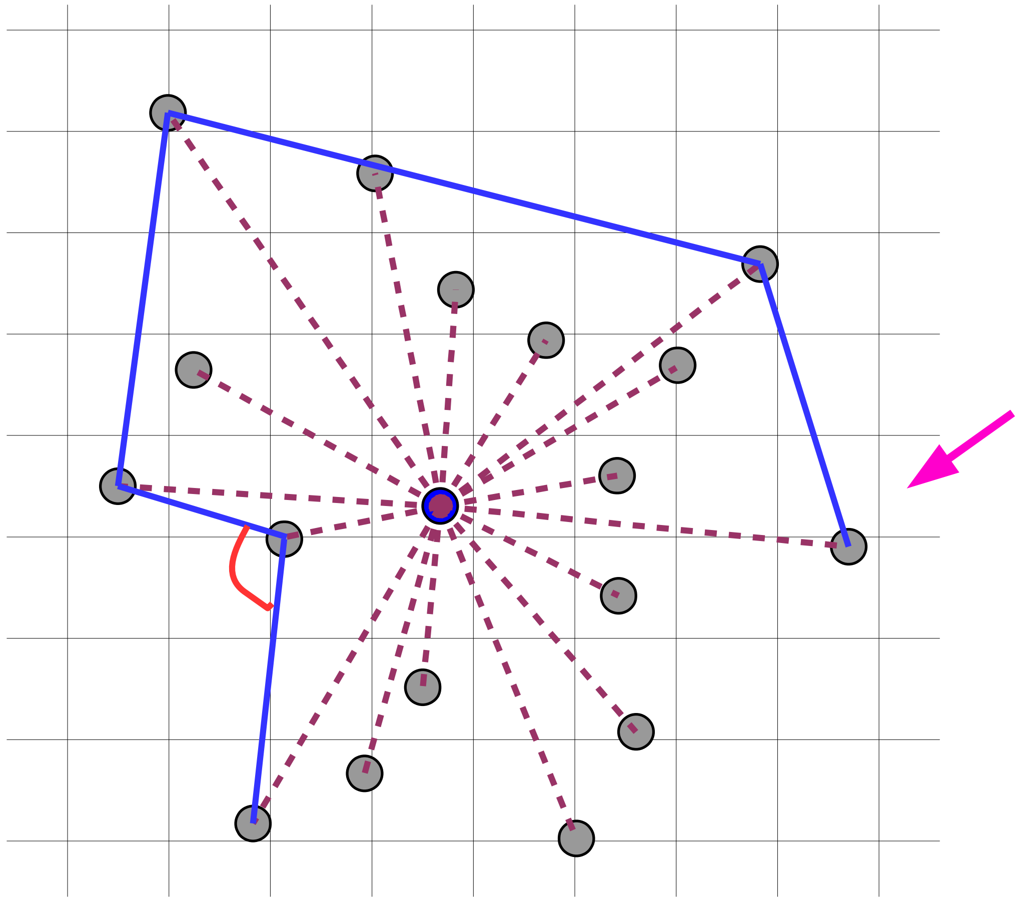


Exercise
(solution)

Step 1,

Step 2,

Step 3 ...

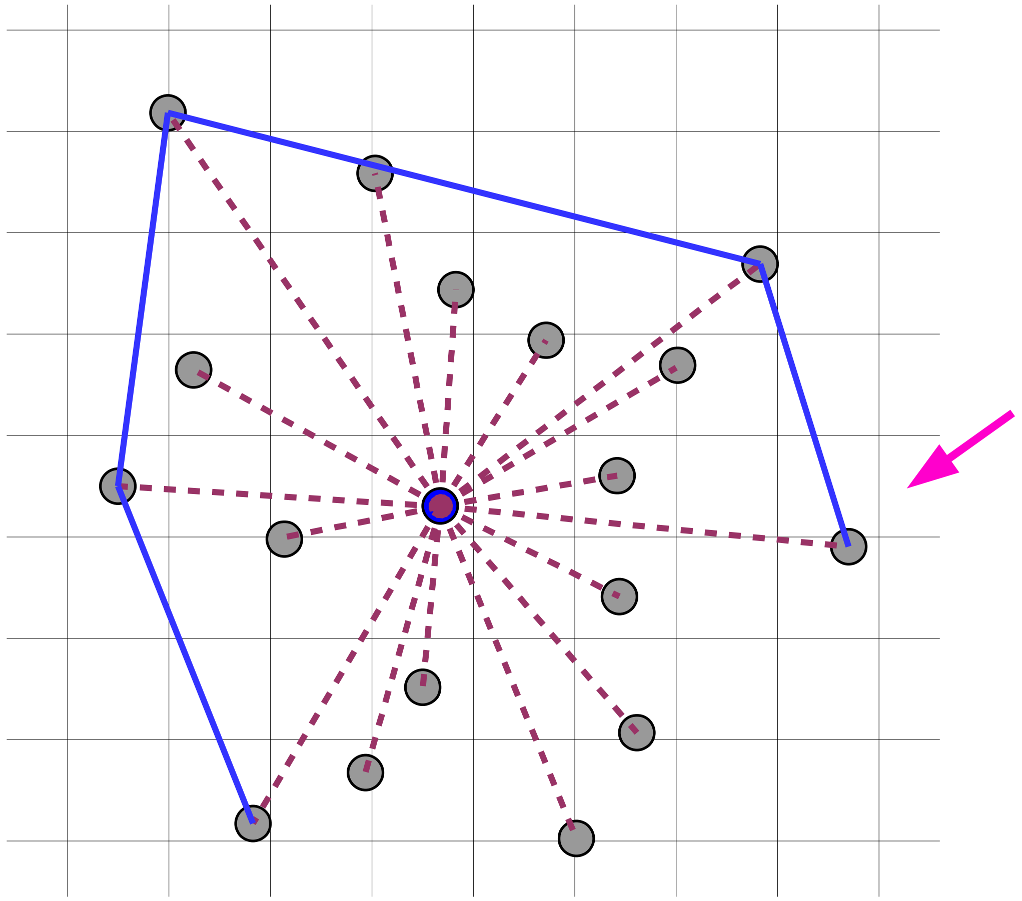


Exercise
(solution)

Step 1,

Step 2,

Step 3 ...

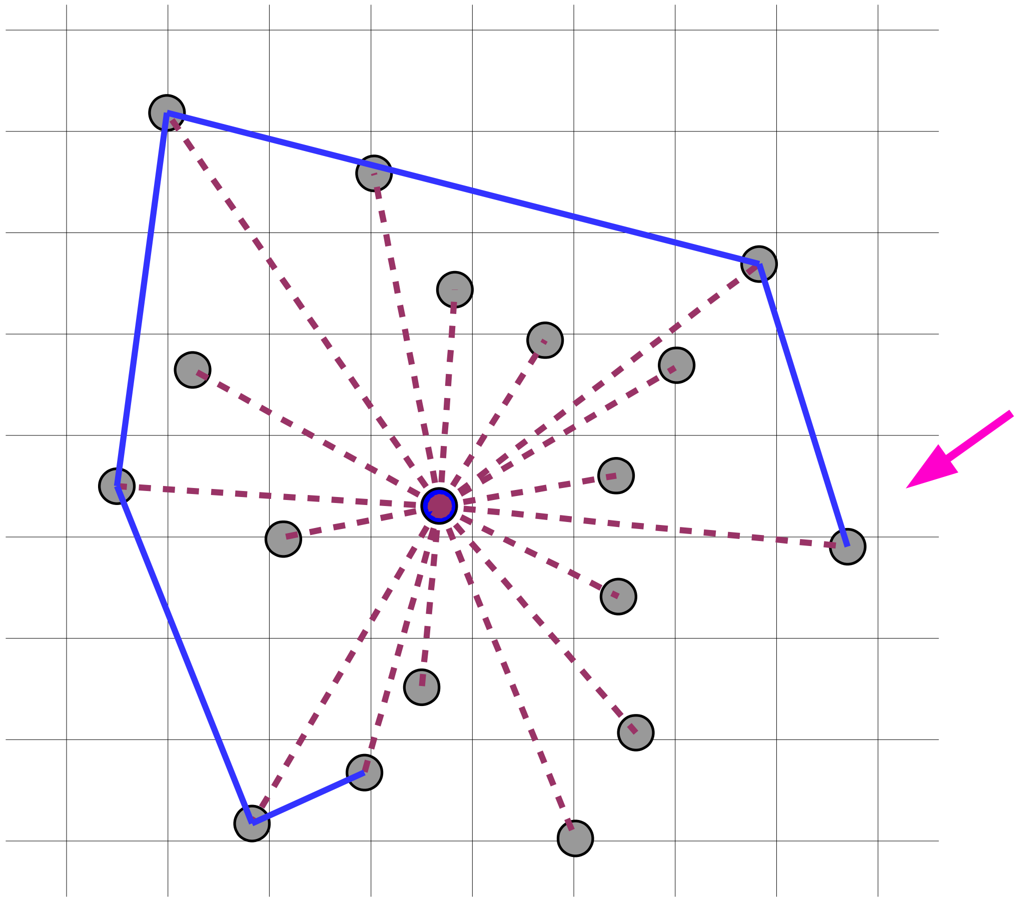


Exercise
(solution)

Step 1,

Step 2,

Step 3 ...



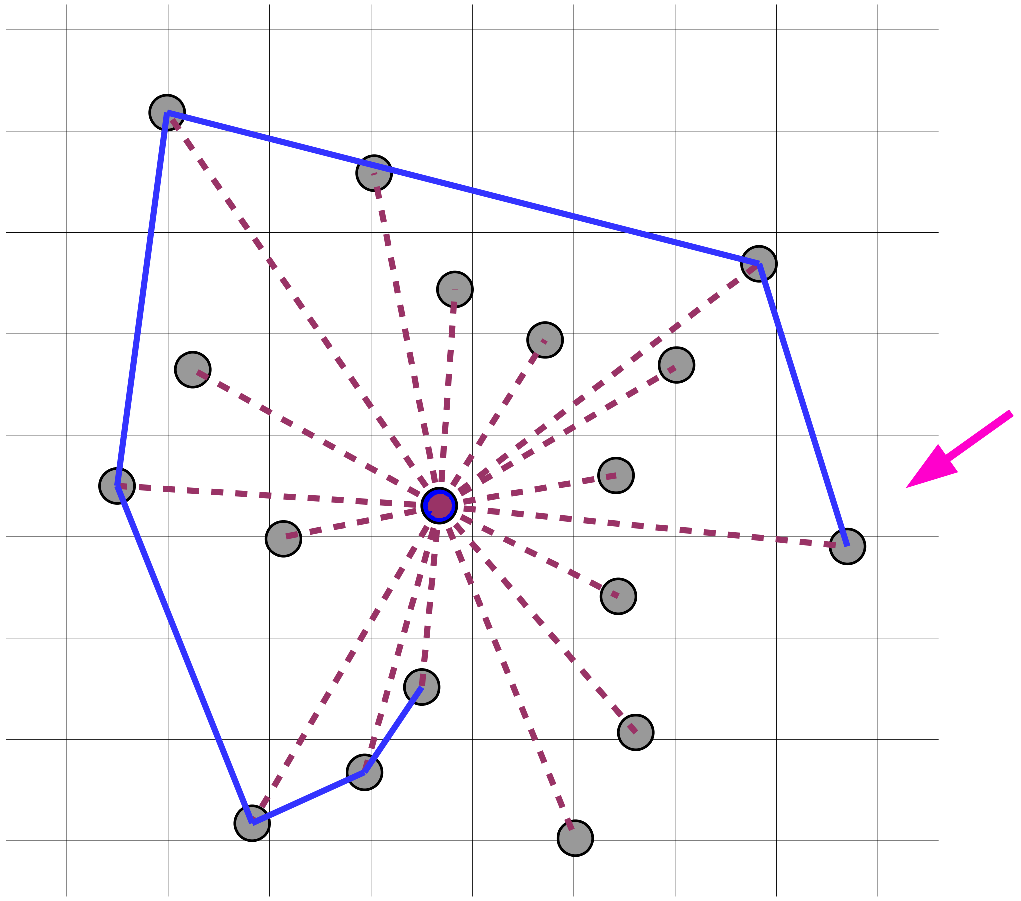
Exercise

(solution)

Step 1,

Step 2,

Step 3 ...

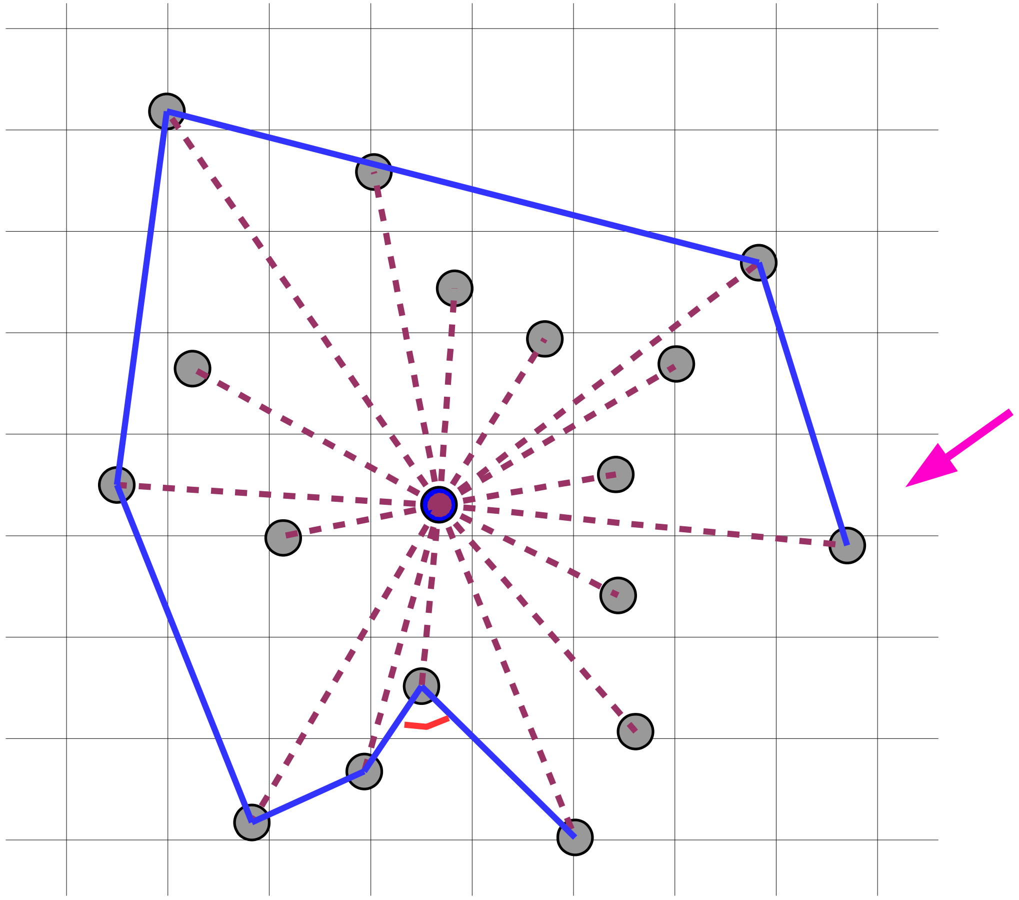


Exercise
(solution)

Step 1,

Step 2,

Step 3 ...



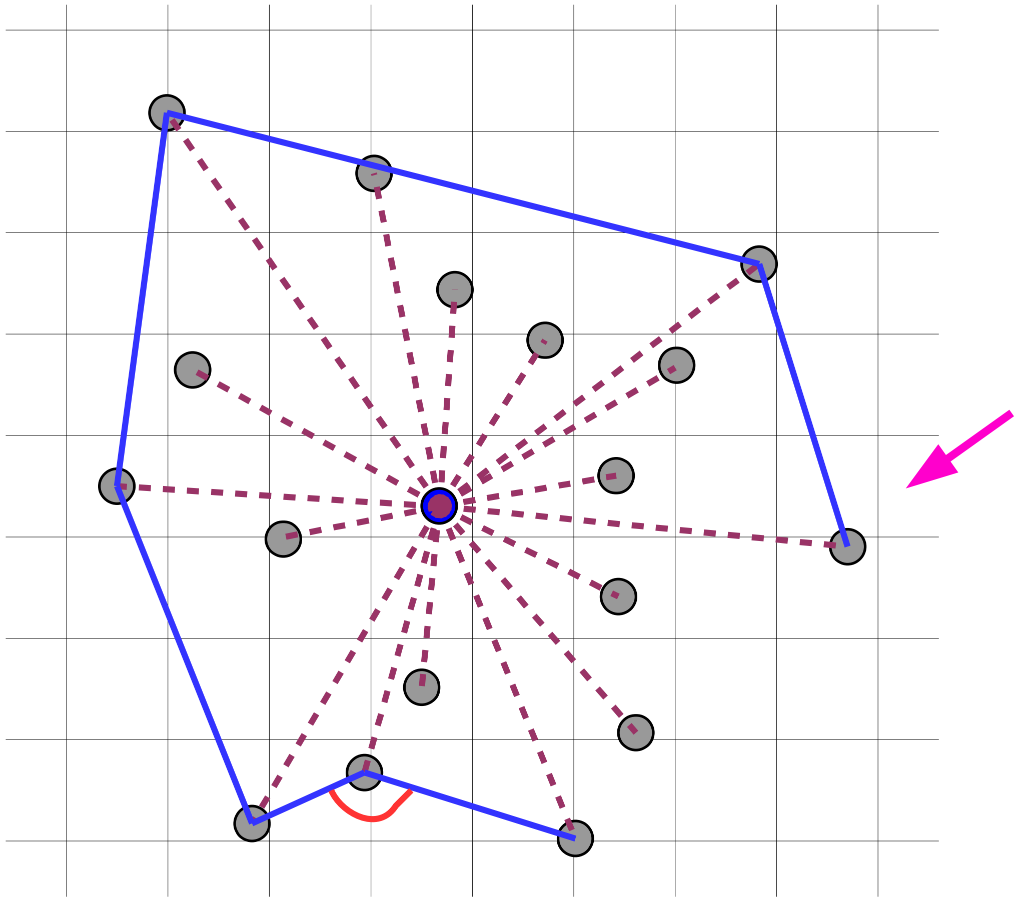
Exercise

(solution)

Step 1,

Step 2,

Step 3 ...



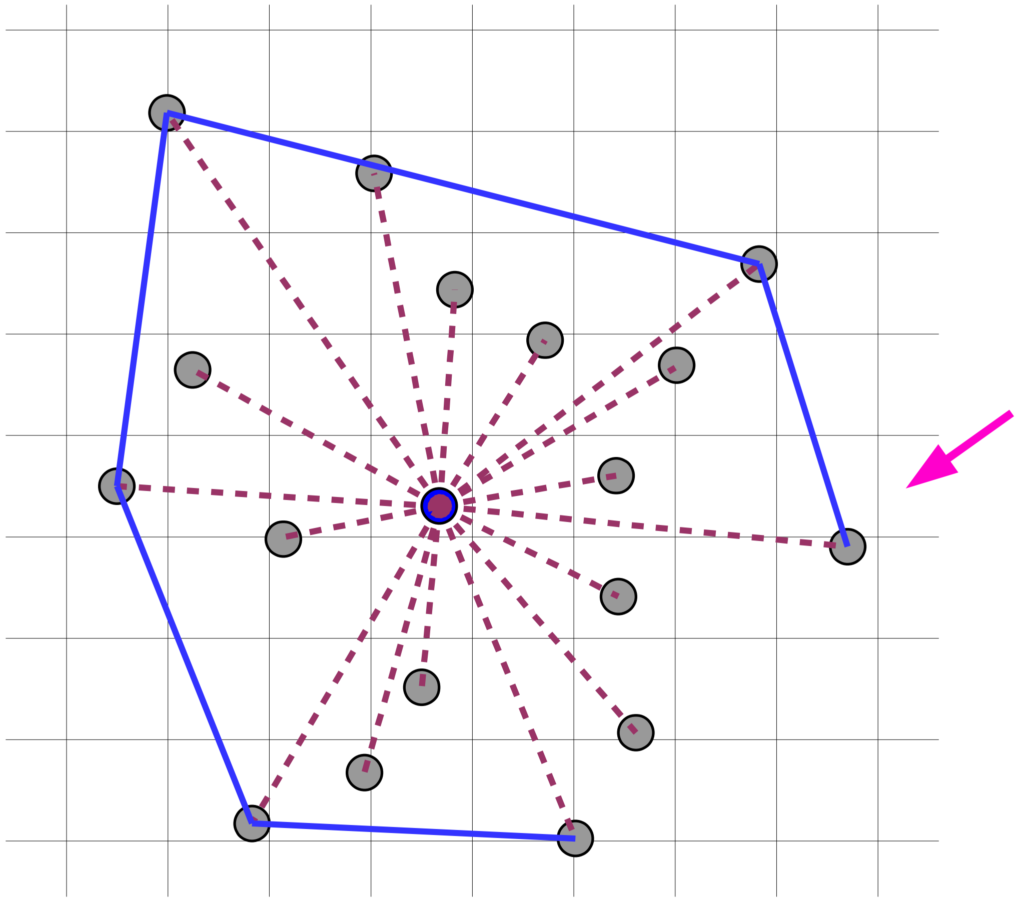
Exercise

(solution)

Step 1,

Step 2,

Step 3 ...



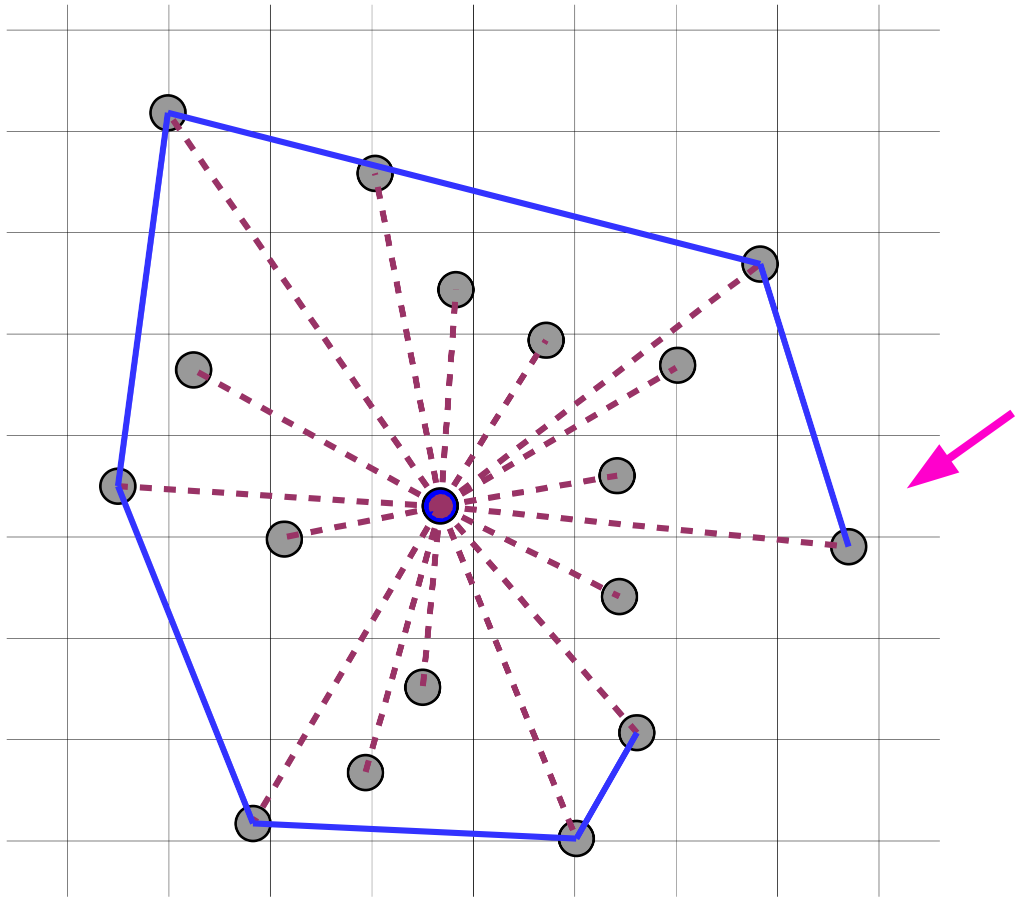
Exercise

(solution)

Step 1,

Step 2,

Step 3 ...



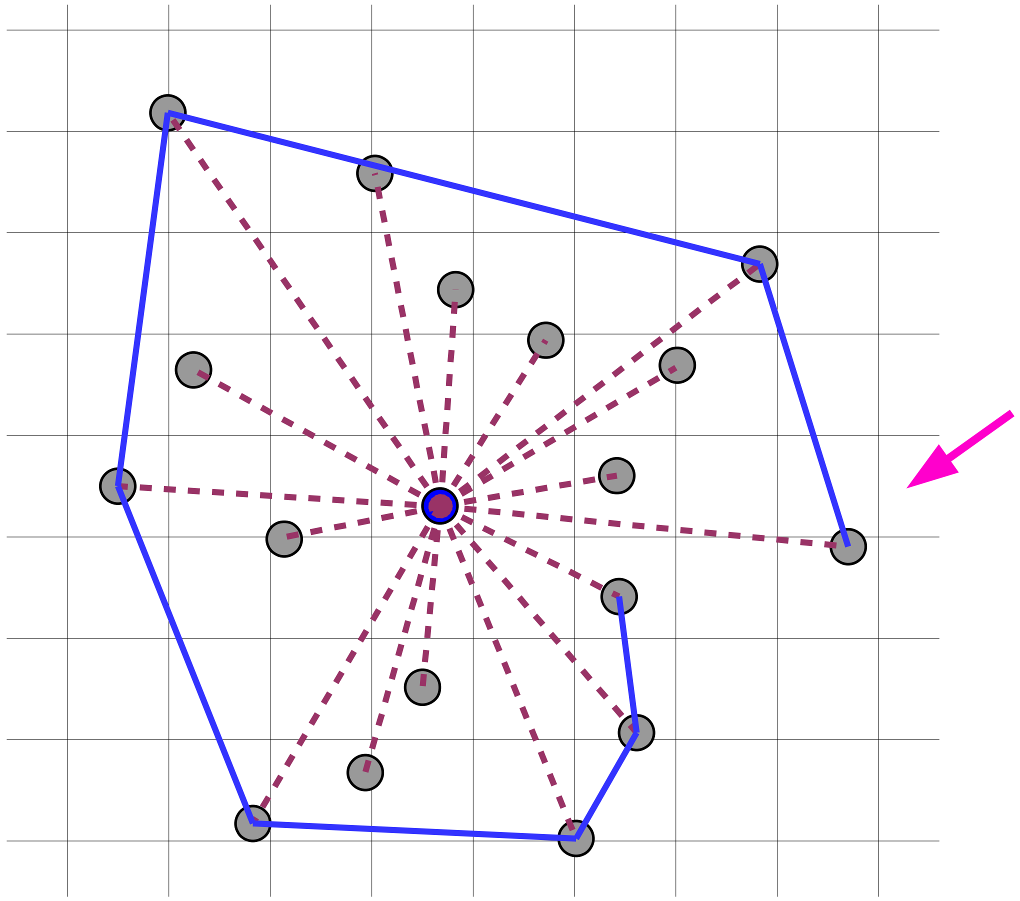
Exercise

(solution)

Step 1,

Step 2,

Step 3 ...



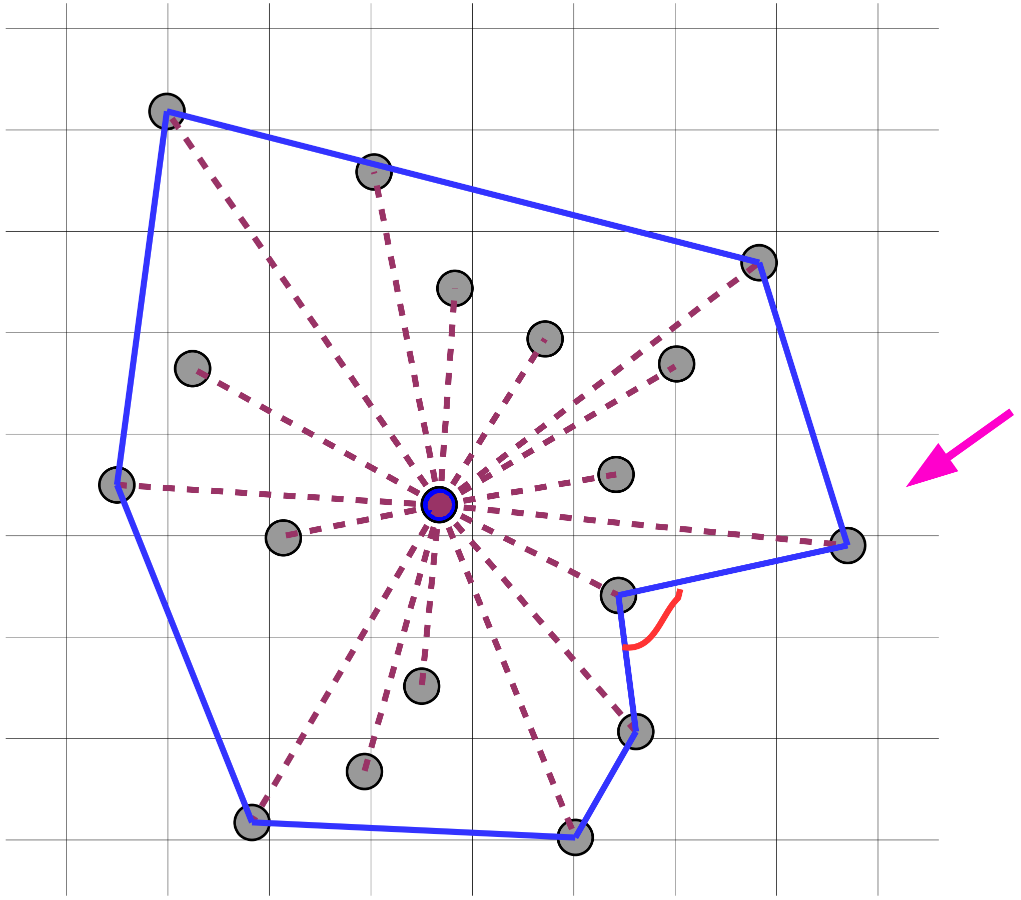
Exercise

(solution)

Step 1,

Step 2,

Step 3 ...



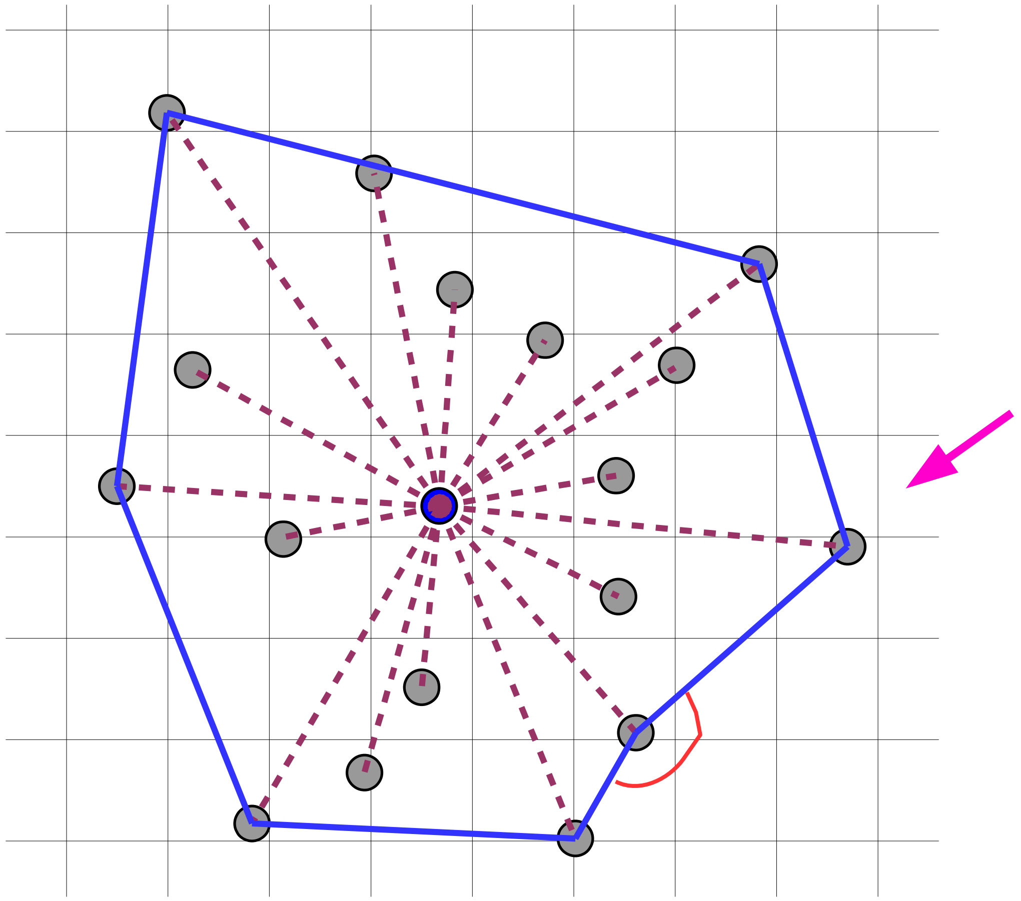
Exercise

(solution)

Step 1,

Step 2,

Step 3 ...



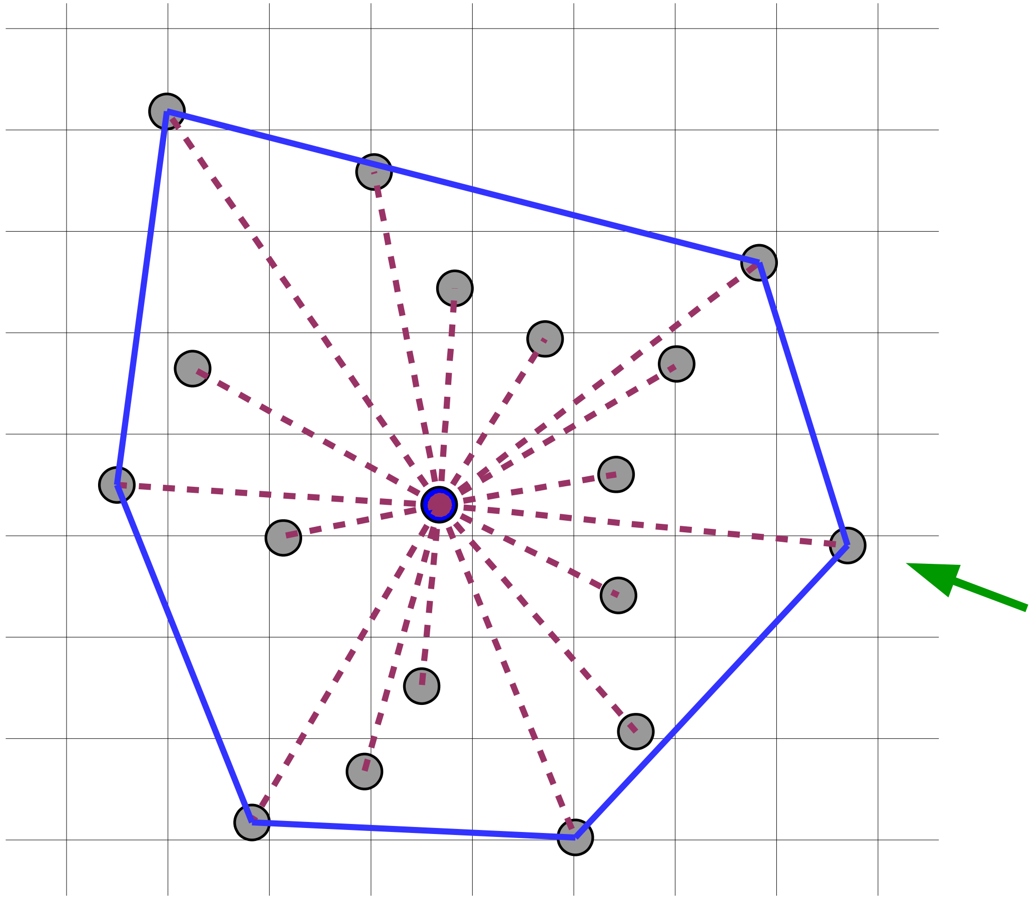
Exercise
(solution)

Step 1,

Step 2,

Step 3

... (completed).



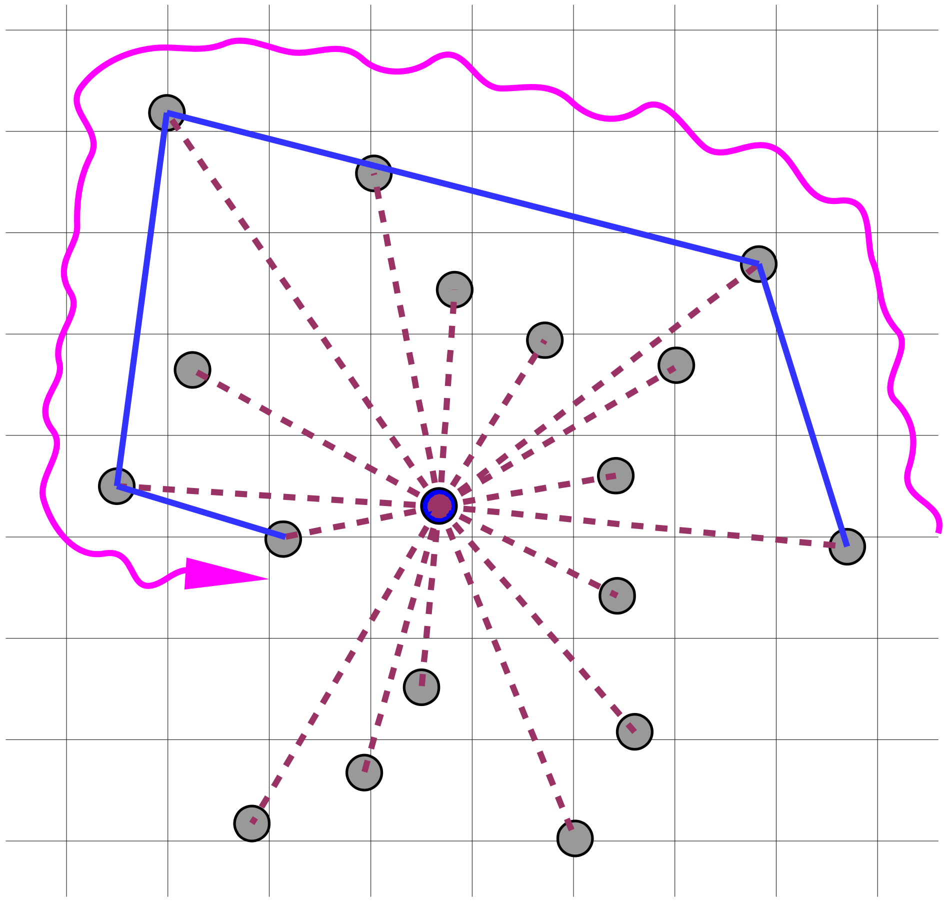
Graham scan

```
function grahamScanCH( A: Seq. of Points ) : Seq. of Points
    // Find center point
    center := mean( A[0..N) )
    // Sort by angle
    sort A[0..N) by increase of angle 'A[i] - center'
    // Traverse
    start := index in [0..N) with maximal X coordinate
    ch := []
    for i := start cyclic to start do
        // Remove last points (if needed)
        while ch.size >= 2
                and rightTurn( ch.top(2), ch.top(), A[i] )
            ch.pop()
        // Add current point
        ch.push( A[i] )
    ch.pop()
    return ch
```

Graham scan

Note, during the algorithm we:

- either add point at the end,
- or remove several points from the same end.

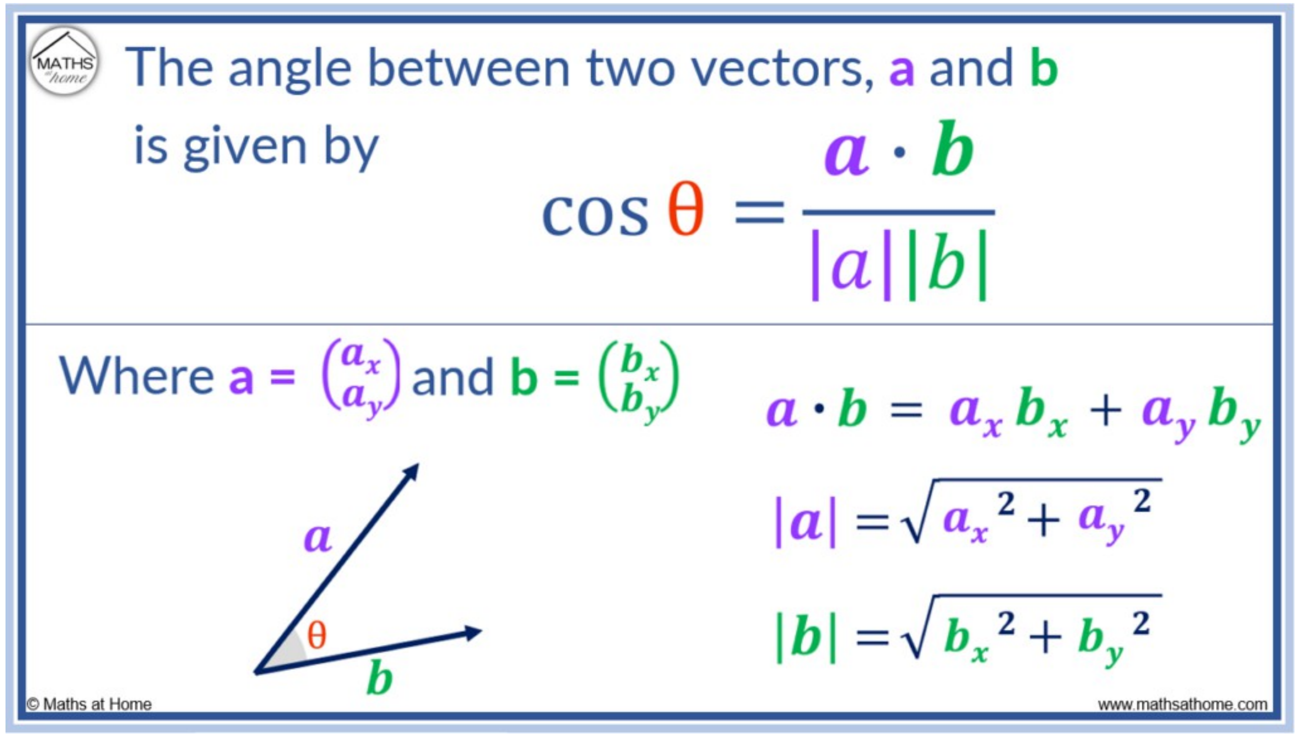


This means that we can keep all points of the current hull in a stack.

... content of the stack is the blue polyline.

Graham scan

Calculating exact angle formed by **2** vectors might require time:



The angle between two vectors, a and b is given by

$$\cos\theta = \frac{a \cdot b}{|a||b|}$$

Where $a = \begin{pmatrix} a_x \\ a_y \end{pmatrix}$ and $b = \begin{pmatrix} b_x \\ b_y \end{pmatrix}$

$$a \cdot b = a_x b_x + a_y b_y$$

$$|a| = \sqrt{a_x^2 + a_y^2}$$

$$|b| = \sqrt{b_x^2 + b_y^2}$$

© Maths at Home

www.mathsathome.com

Graham scan

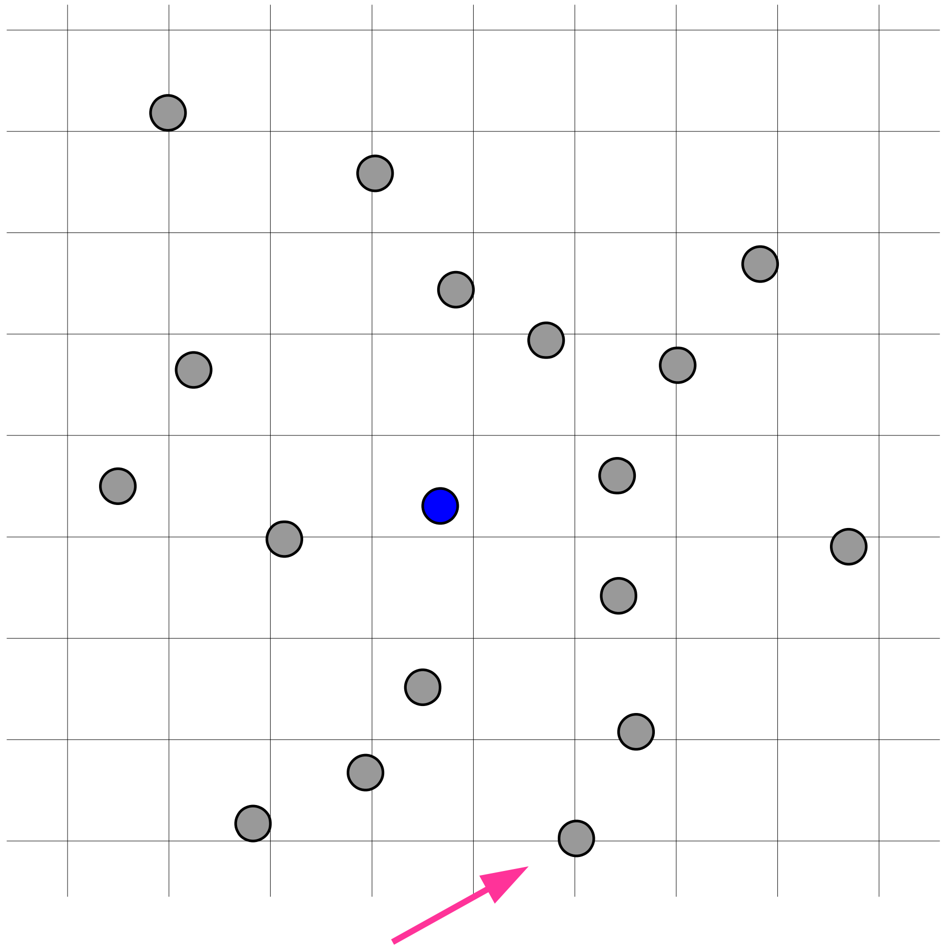
But we are interested only in whether it is left turn or right turn.

$$P_1 = (x_1, y_1),$$
$$P_2 = (x_2, y_2)$$
$$P_3 = (x_3, y_3),$$

$$(x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)$$

Graham scan

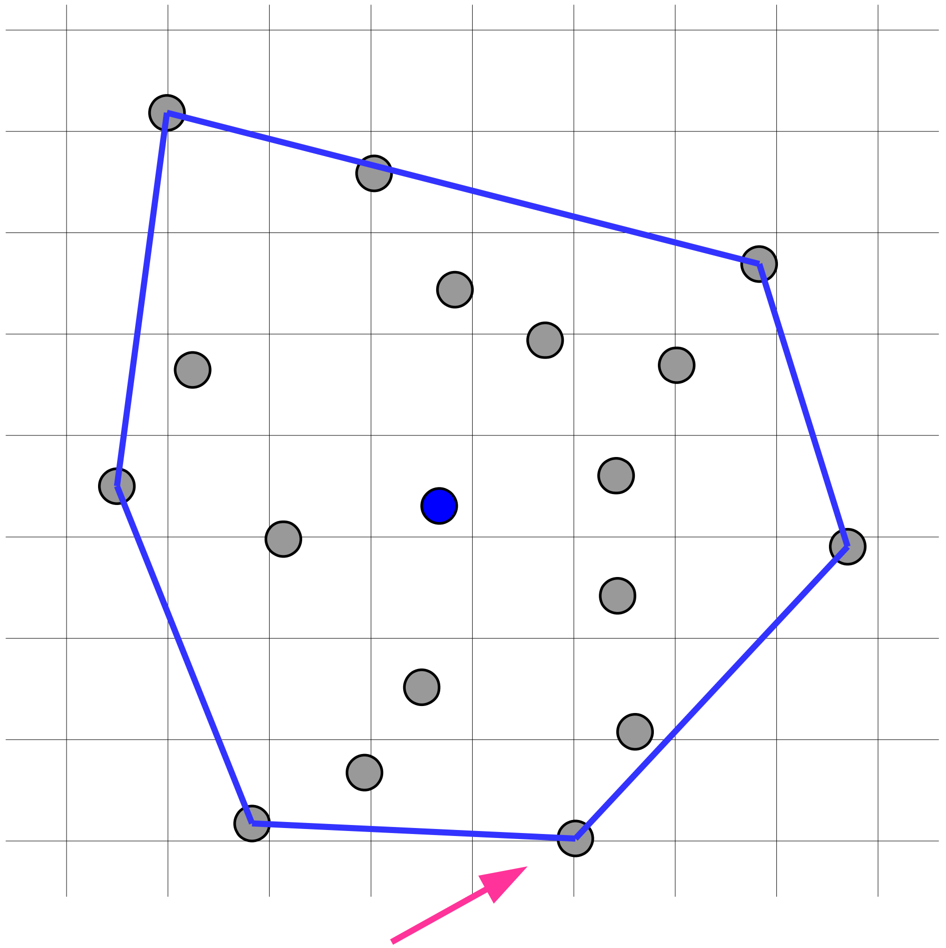
Question: Can we start the algorithm not from the right-most, but from bottom-most point?



Graham scan

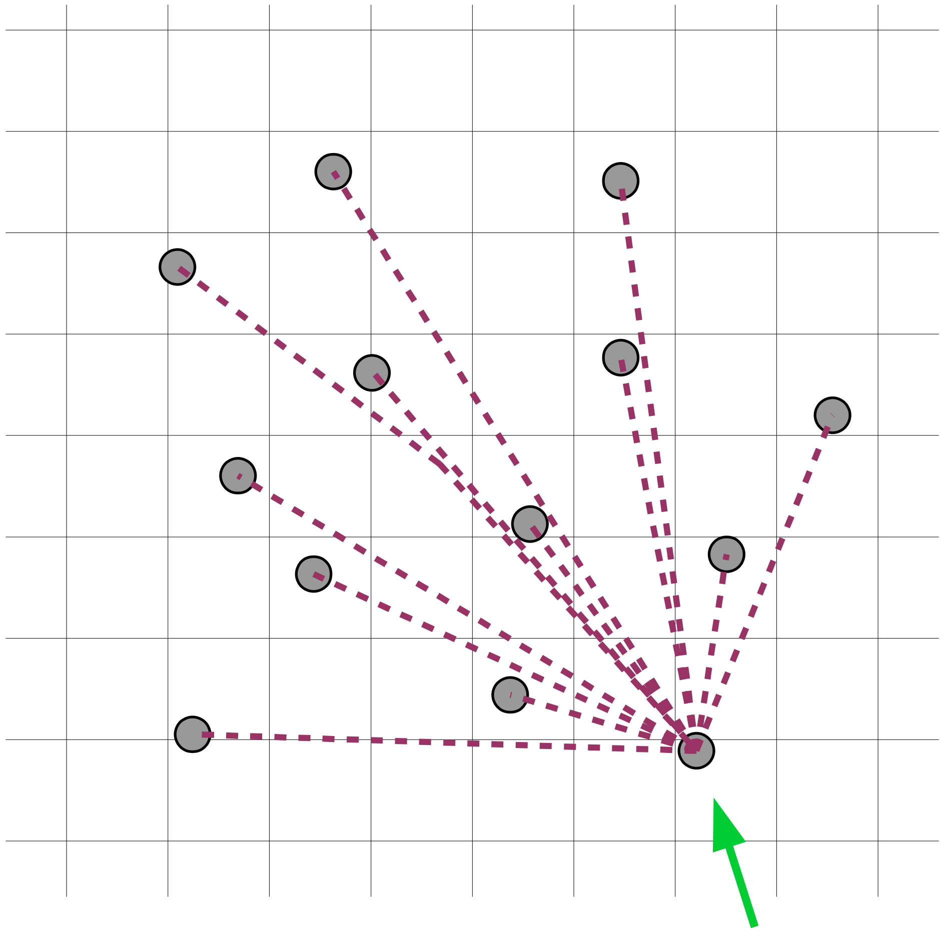
Question: Can we start the algorithm not from the right-most, but from bottom-most point?

Answer: Yes, as it also participates in the hull.



Graham scan

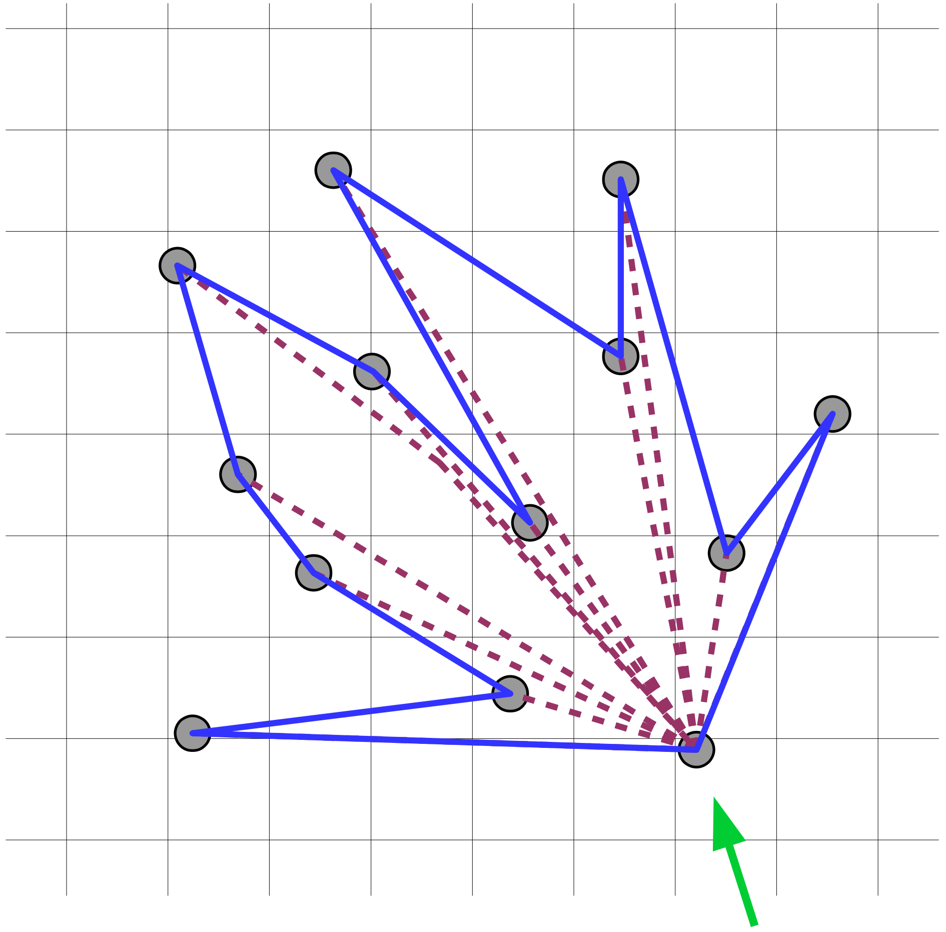
Question: Can we sort all the points not by angle formed with center point, but by angle formed with the bottom-most point?



Graham scan

Question: Can we sort all the points not by angle formed with center point, but by angle formed with the bottom-most point?

Answer: Yes, as eventually we will still need to eliminate all "right turns".



Time complexity

What is time complexity of the algorithm?

- We have **2** nested loops,
- Each can run up to **N** iterations.

But...

Time complexity

What is time complexity of the algorithm?

- We have **2** nested loops,
- Each can run up to **N** iterations.

But...

- Outer loop only adds points '**A[i]**' to the stack,
- Inner loop only extracts unnecessary points,
- Every point can be extracted from the stack at most once.

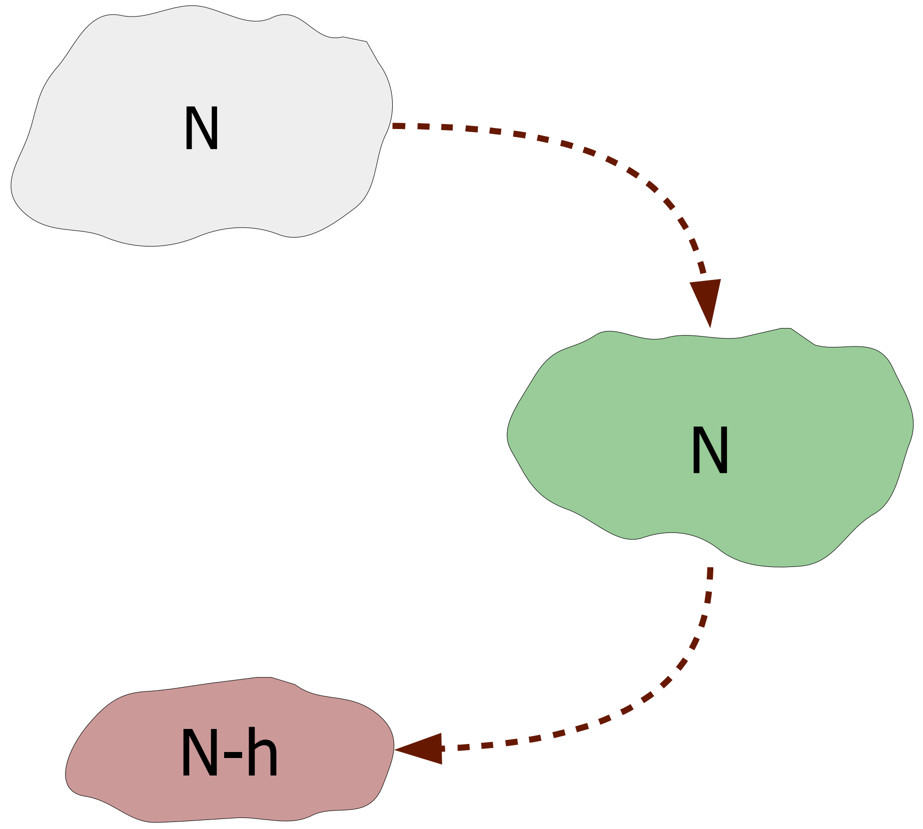
So the time complexity of the main step is **O(N)**.

Time complexity

So during the algorithm:

... all '**N**' points are added once into the stack,

... extra '**N-h**' points are extracted from the stack.



Time complexity

Overall work is distributed like:

$$\underbrace{\mathbf{O(N*logN)}}_{\text{sorting}} \;\; \mathbf{+} \;\; \underbrace{\mathbf{O(N)}}_{\text{CH}}$$

If points are already sorted by angle, the algorithm will work in **O(N)**.

Presentation writer: Tigran Hayrapetyan

Lecturer | Programmer | Researcher

www.linkedin.com/in/tigran-hayrapetyan-cs/

Thank you!

Convex hull
(Graham scan)