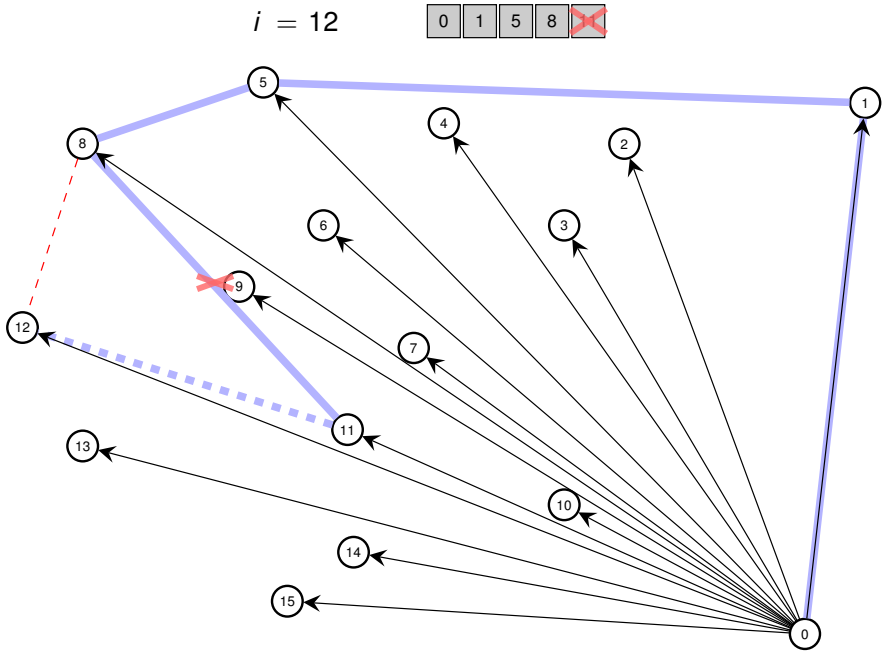


7: Geometric Algorithms

Frank Stajano

Thomas Sauerwald

Lent 2015

UNIVERSITY OF CAMBRIDGE

Outline

Introduction and Line Intersection

Convex Hull

Introduction

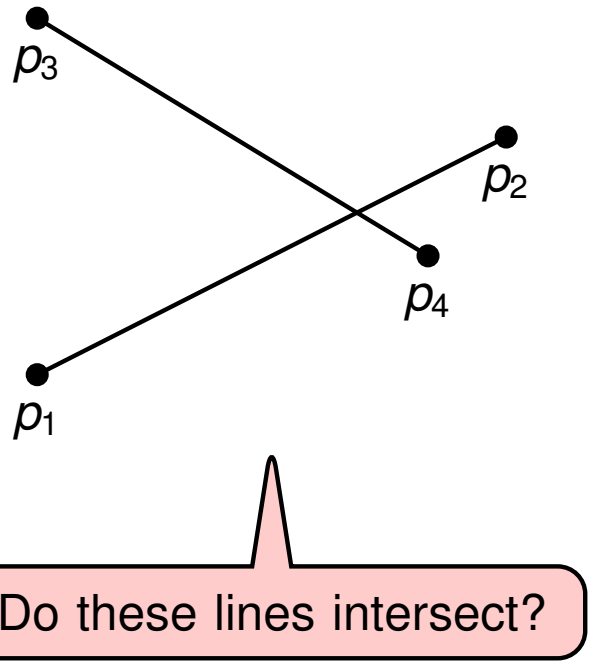
Computational Geometry

- Branch that studies algorithms for geometric problems
- typically, input is a set of points, line segments etc.

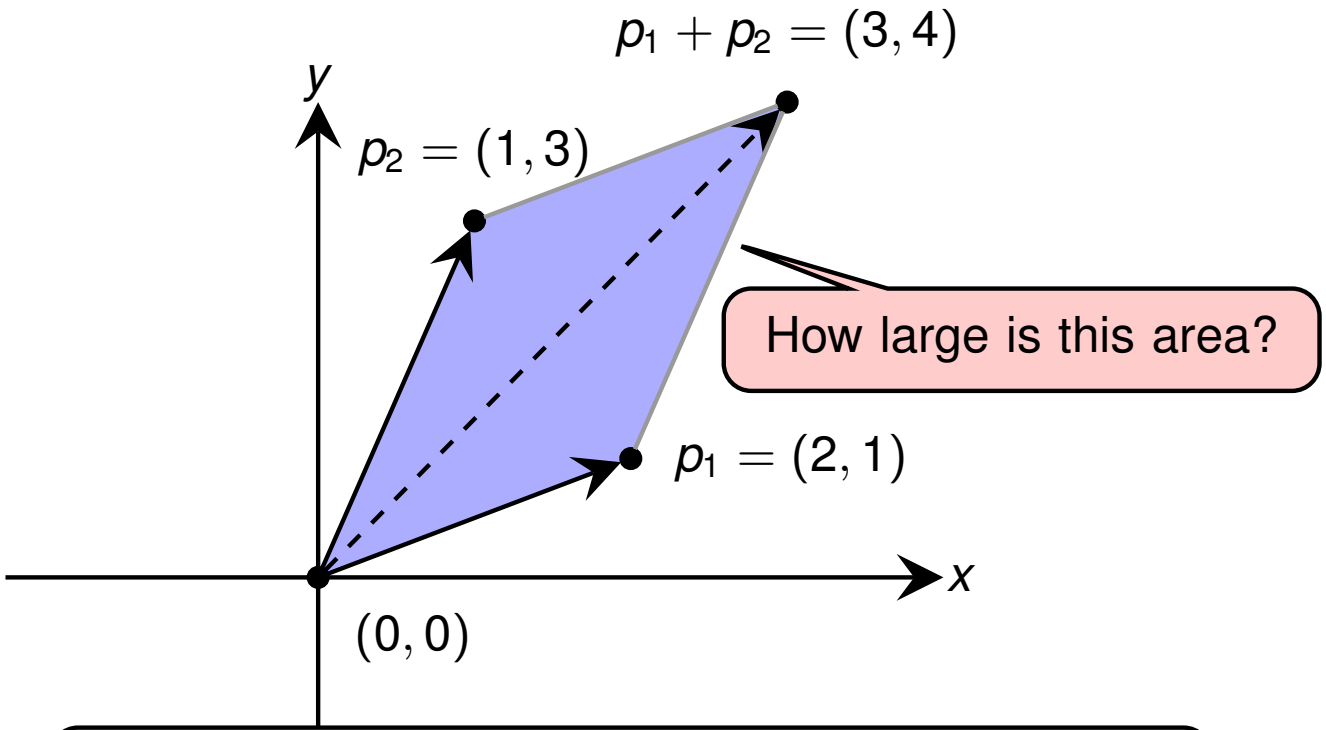
Applications

- computer graphics
- computer vision
- textile layout
- VLSI design

⋮



Cross Product (Area)

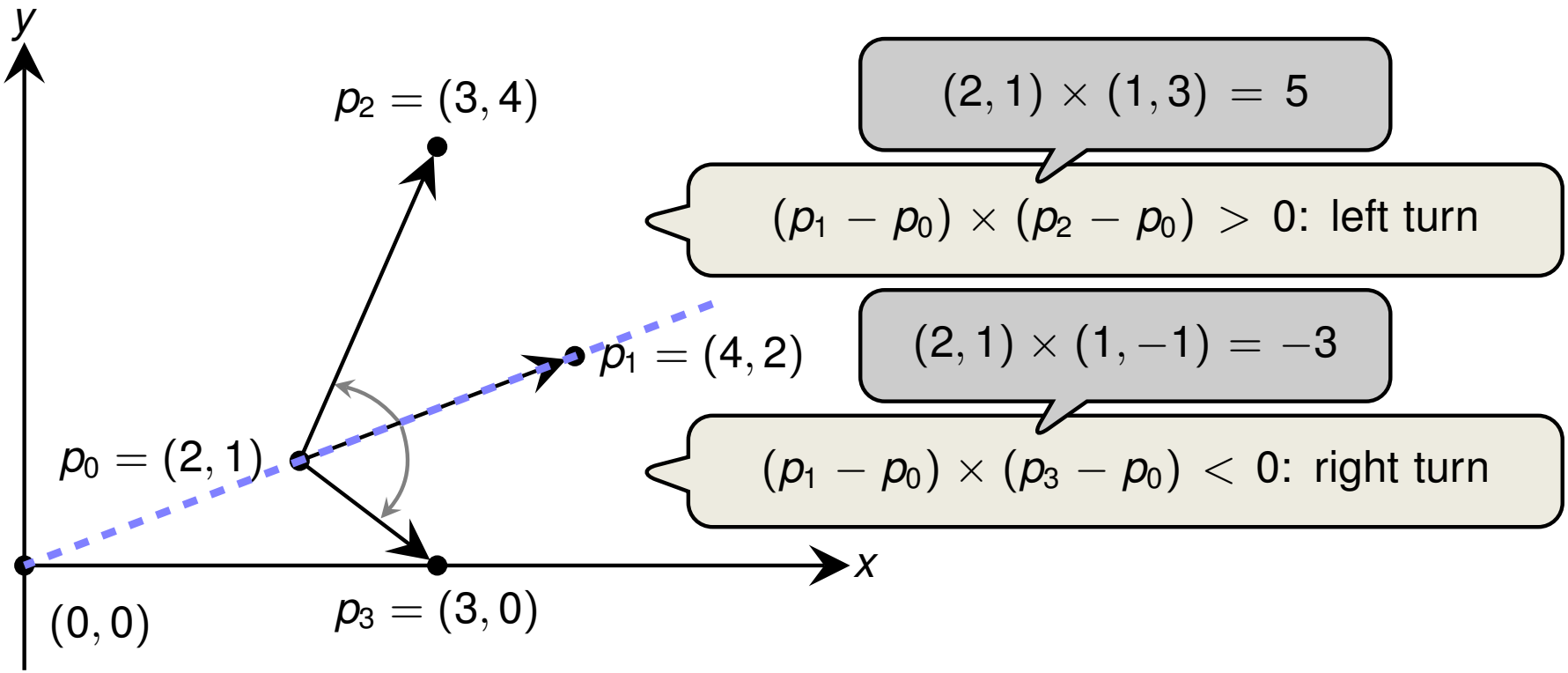


Alternatively, one could take the dot-product (but not used here): $p_1 \cdot p_2 = \|p_1\| \cdot \|p_2\| \cdot \cos(\phi)$.

$$p_1 \times p_2 = \det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} = x_1 y_2 - x_2 y_1 = 2 \cdot 3 - 1 \cdot 1 = 5$$

$$p_2 \times p_1 = y_1 x_2 - y_2 x_1 = -(p_1 \times p_2) = -5$$

Using Cross product to determine Turns (2/2)



Outline

Introduction and Line Intersection

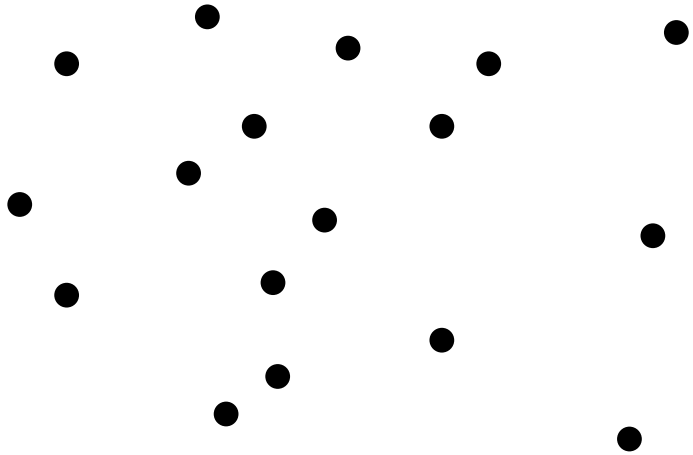
Convex Hull

Convex Hull

Definition

The convex hull of a set Q of points is the smallest convex polygon P for which each point in Q is either on the boundary of P or in its interior.

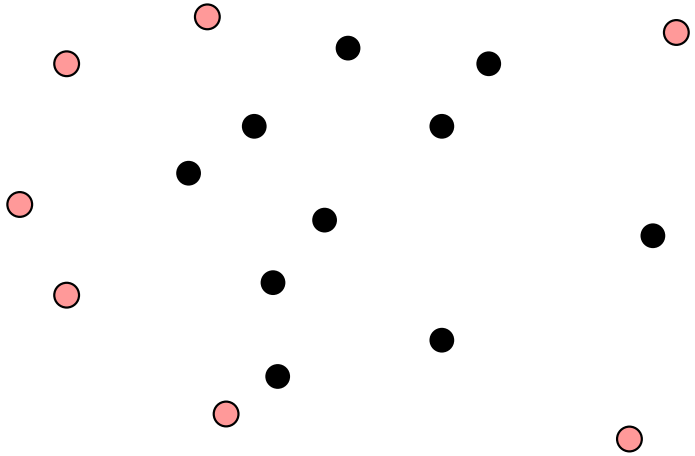
Convex Hull



Definition

The convex hull of a set Q of points is the smallest convex polygon P for which each point in Q is either on the boundary of P or in its interior.

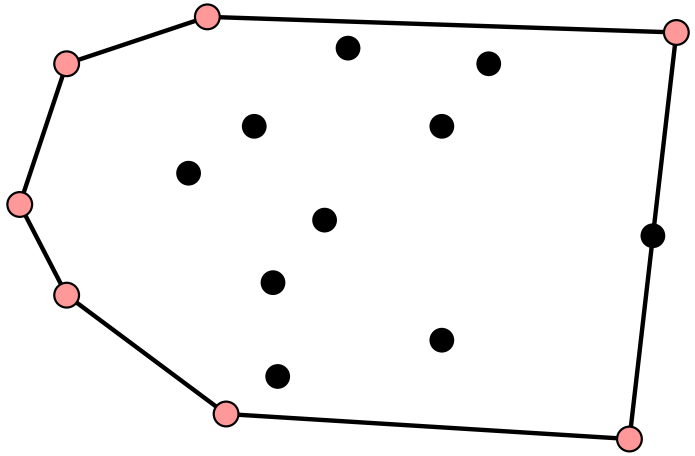
Convex Hull



Definition

The convex hull of a set Q of points is the smallest convex polygon P for which each point in Q is either on the boundary of P or in its interior.

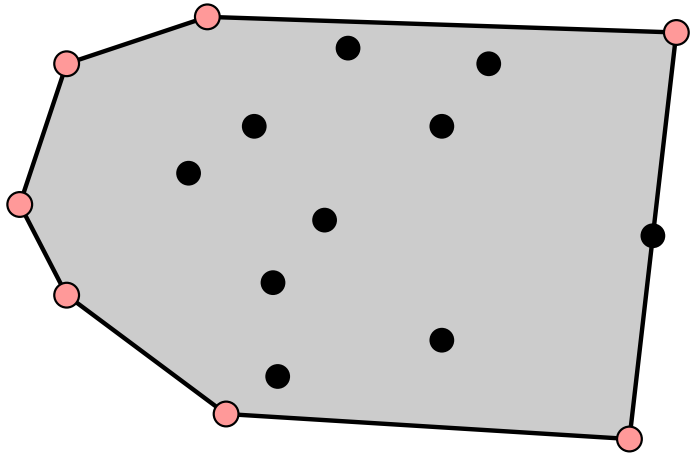
Convex Hull



Definition

The convex hull of a set Q of points is the smallest convex polygon P for which each point in Q is either on the boundary of P or in its interior.

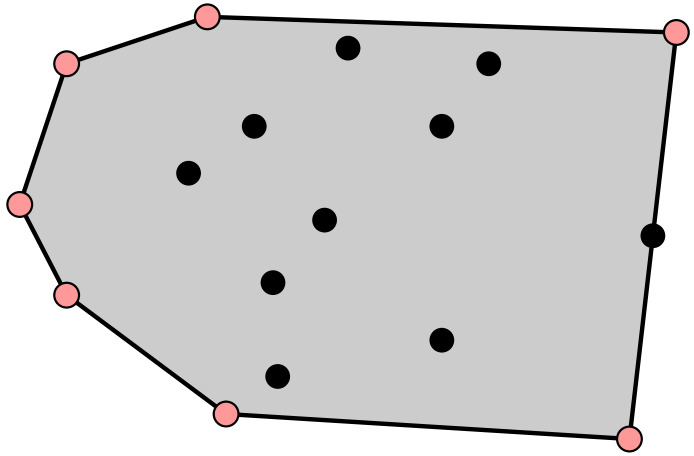
Convex Hull



Definition

The convex hull of a set Q of points is the smallest convex polygon P for which each point in Q is either on the boundary of P or in its interior.

Convex Hull

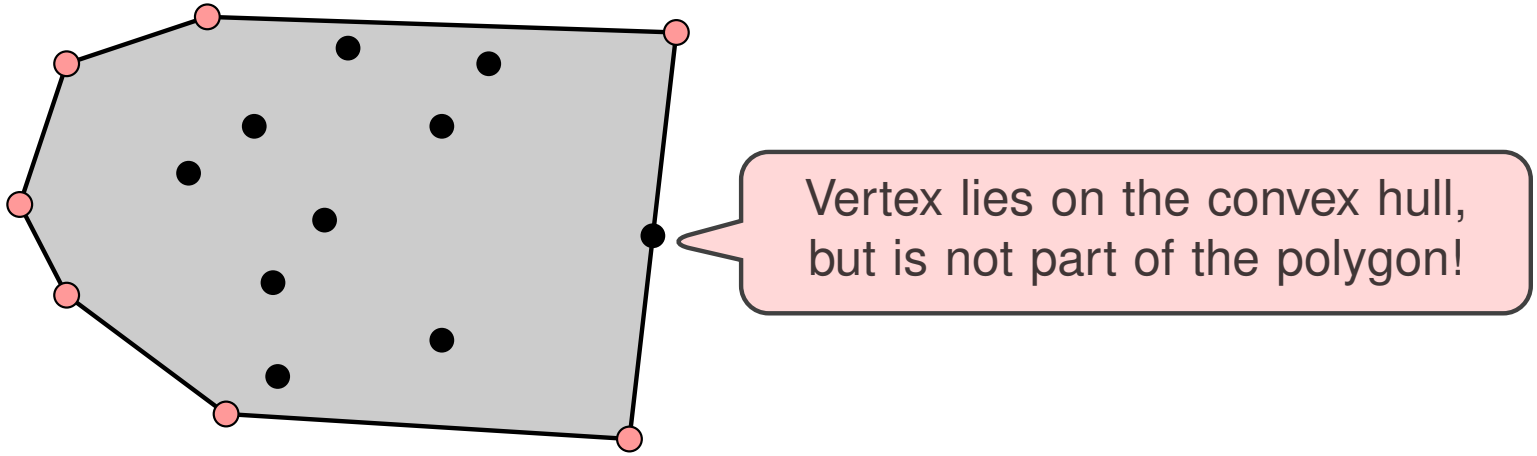


Definition

The convex hull of a set Q of points is the smallest convex polygon P for which each point in Q is either on the boundary of P or in its interior.

Smallest perimeter fence enclosing the points

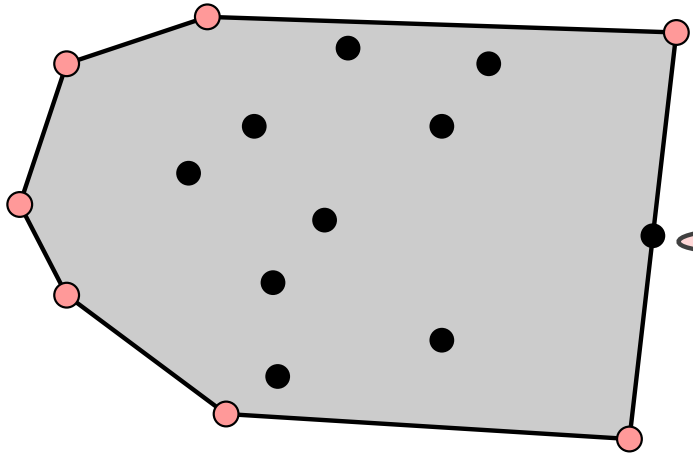
Convex Hull



Definition

The convex hull of a set Q of points is the smallest convex polygon P for which each point in Q is either on the boundary of P or in its interior.

Convex Hull



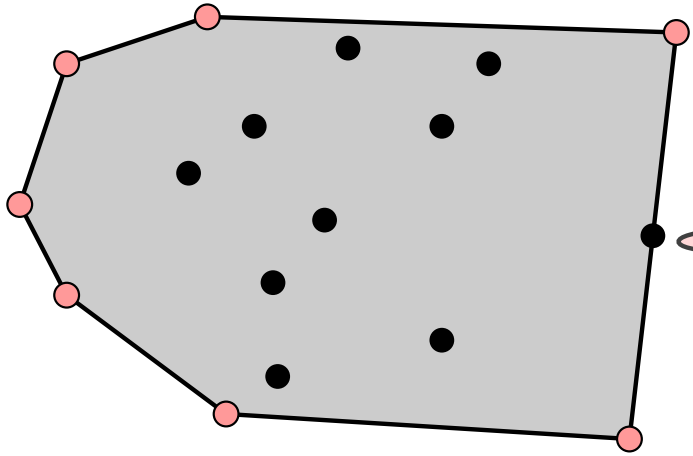
Vertex lies on the convex hull, but is not part of the polygon!

Definition

The convex hull of a set Q of points is the smallest convex polygon P for which each point in Q is either on the boundary of P or in its interior.

Convex Hull Problem

Convex Hull



Vertex lies on the convex hull, but is not part of the polygon!

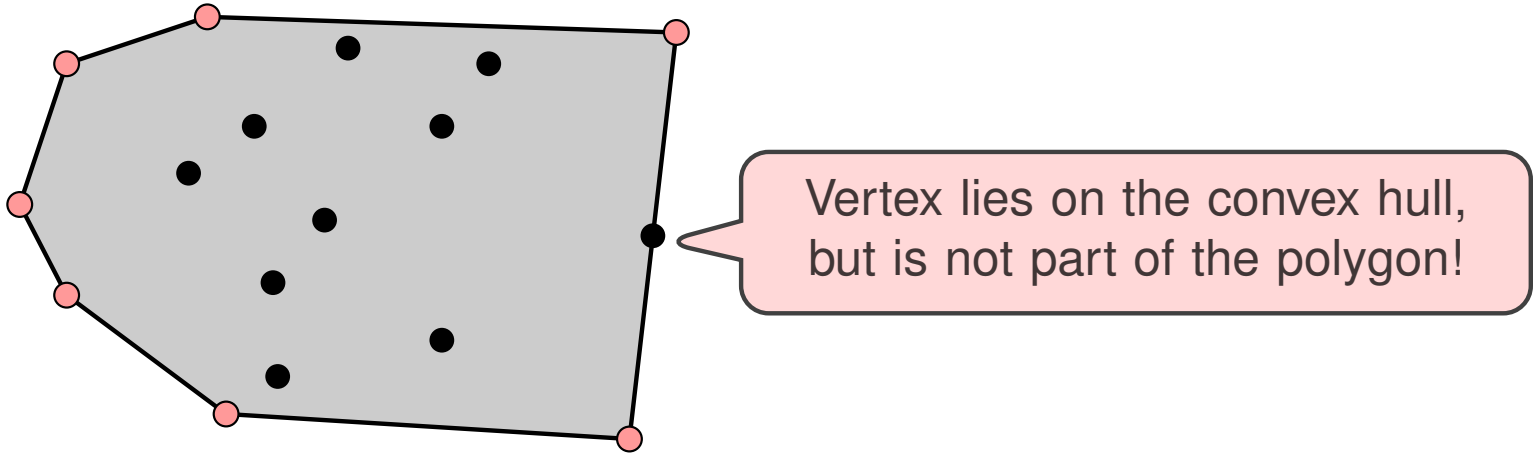
Definition

The convex hull of a set Q of points is the smallest convex polygon P for which each point in Q is either on the boundary of P or in its interior.

Convex Hull Problem

- Input: set of points Q in the Euclidean space

Convex Hull



Definition

The convex hull of a set Q of points is the smallest convex polygon P for which each point in Q is either on the boundary of P or in its interior.

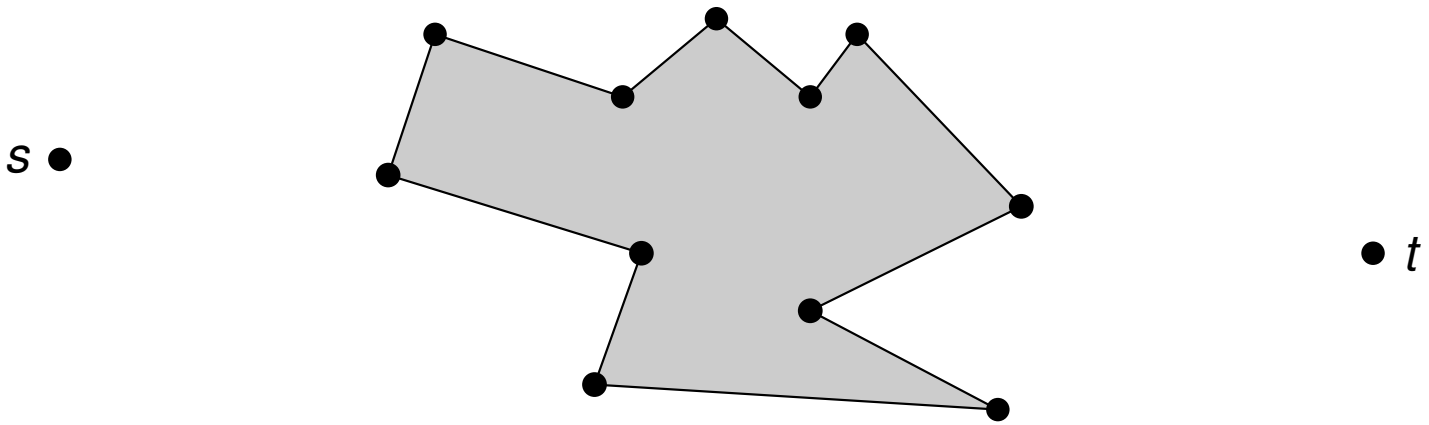
Convex Hull Problem

- Input: set of points Q in the Euclidean space
- Output: return points of the convex hull in counterclockwise order

Application of Convex Hull

Robot Motion Planning

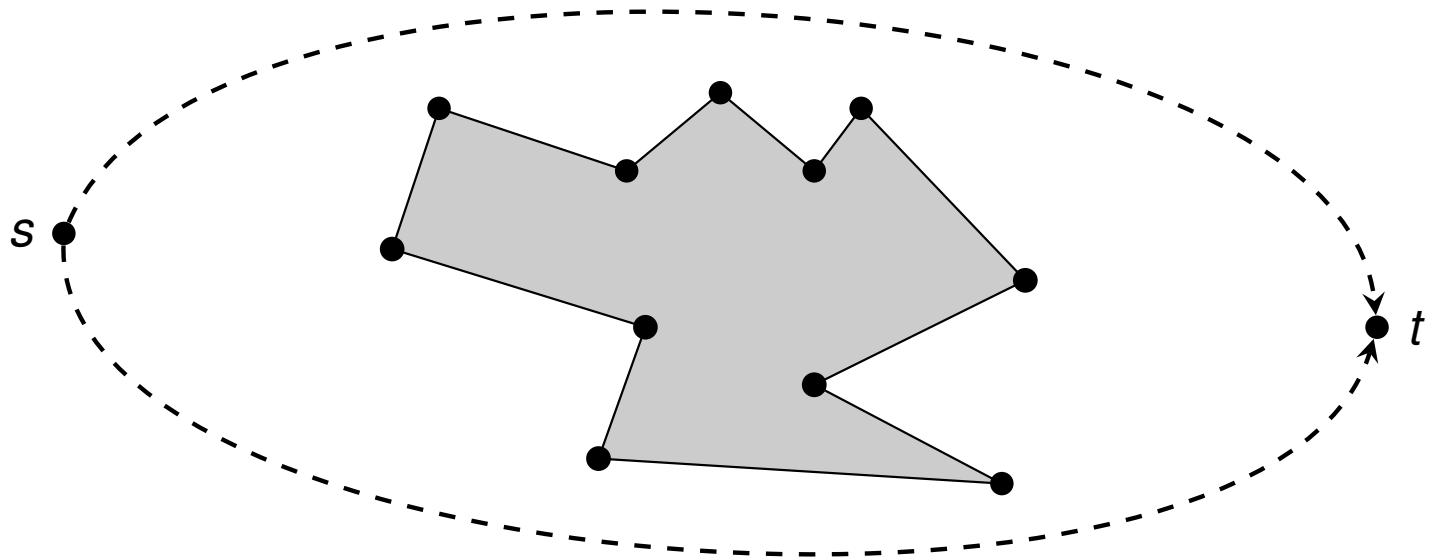
Find shortest path from s to t which avoids a polygonal obstacle.



Application of Convex Hull

Robot Motion Planning

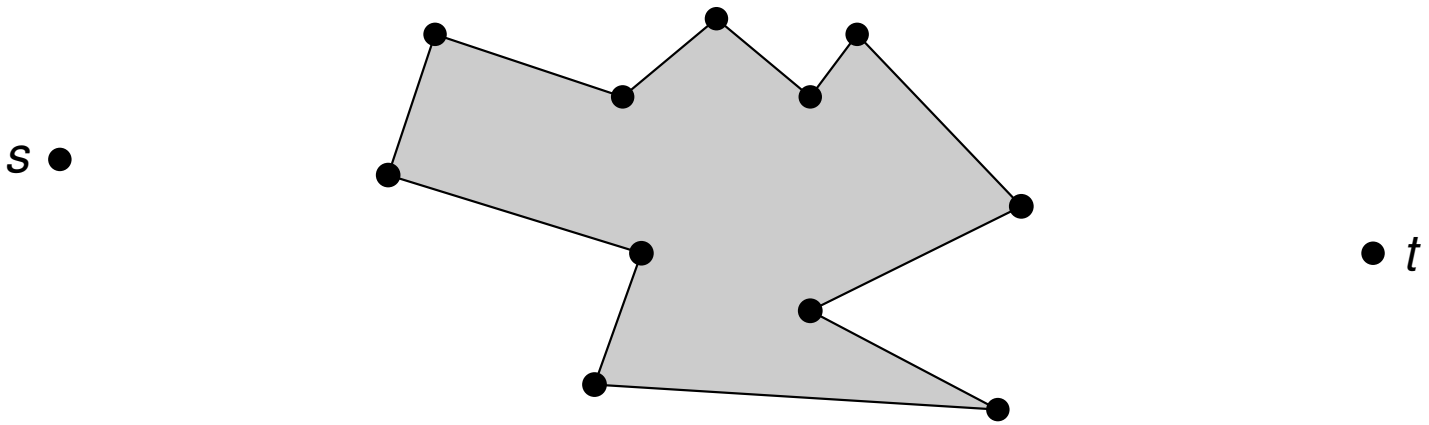
Find shortest path from s to t which avoids a polygonal obstacle.



Application of Convex Hull

Robot Motion Planning

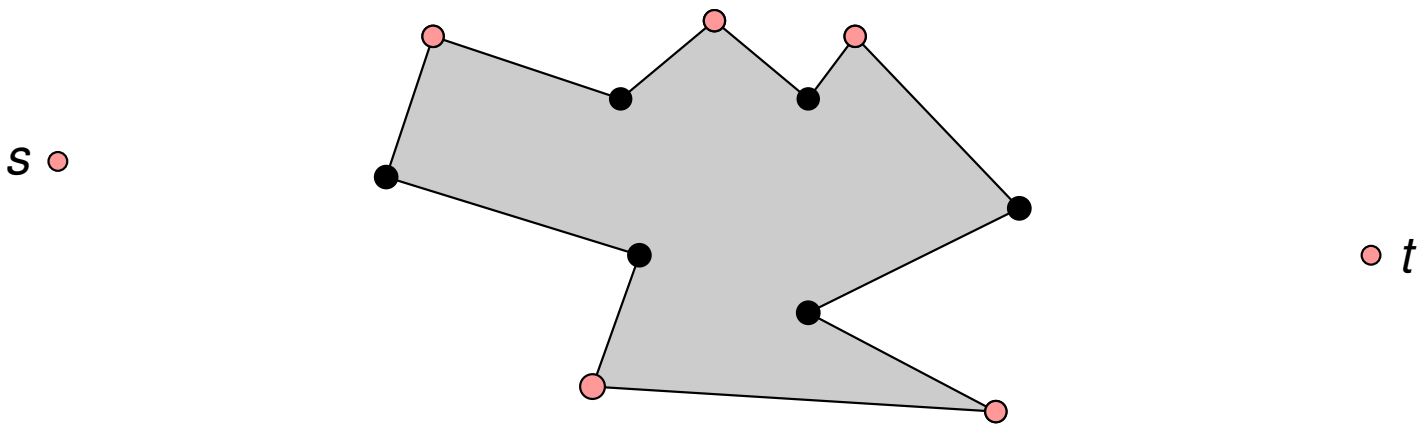
Find shortest path from s to t which avoids a polygonal obstacle.



Application of Convex Hull

Robot Motion Planning

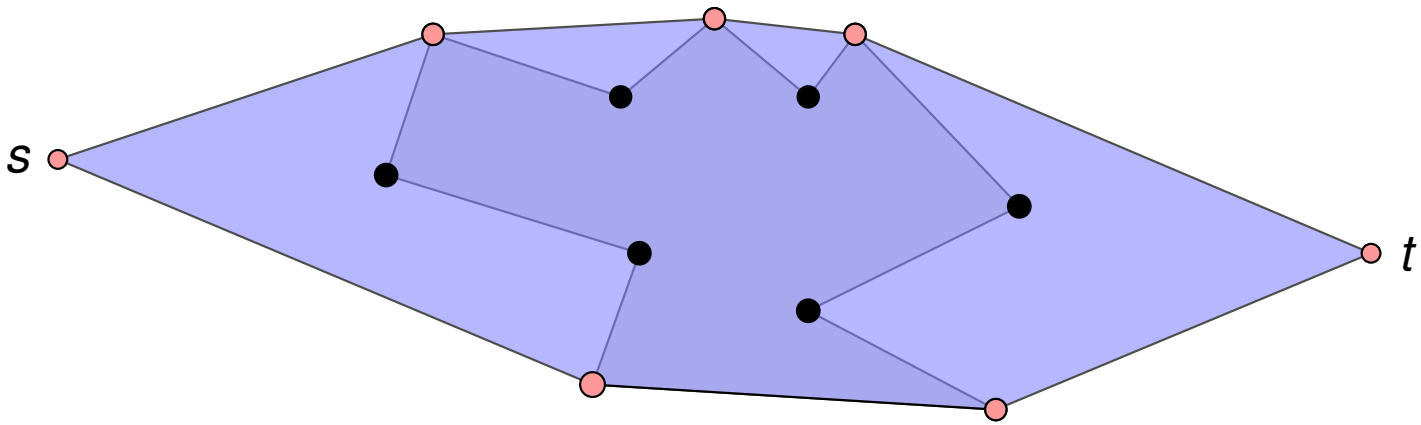
Find shortest path from s to t which avoids a polygonal obstacle.



Application of Convex Hull

Robot Motion Planning

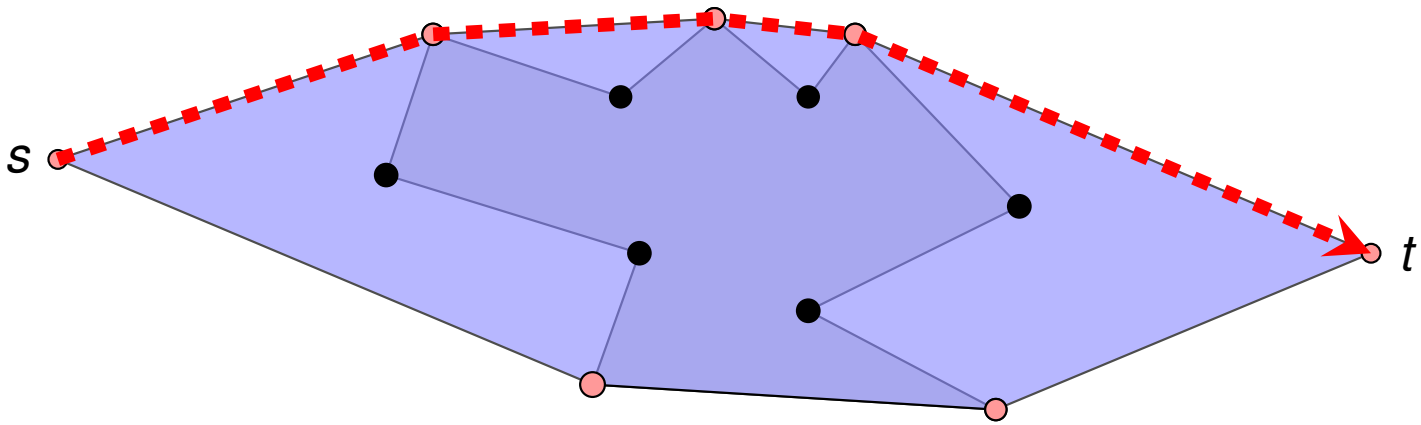
Find shortest path from s to t which avoids a polygonal obstacle.



Application of Convex Hull

Robot Motion Planning

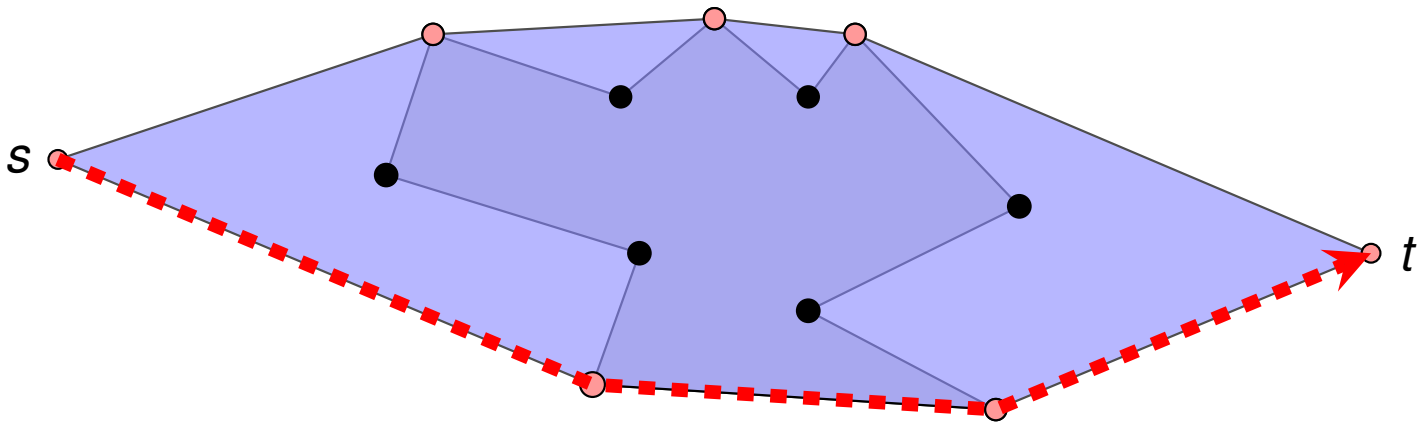
Find shortest path from s to t which avoids a polygonal obstacle.



Application of Convex Hull

Robot Motion Planning

Find shortest path from s to t which avoids a polygonal obstacle.

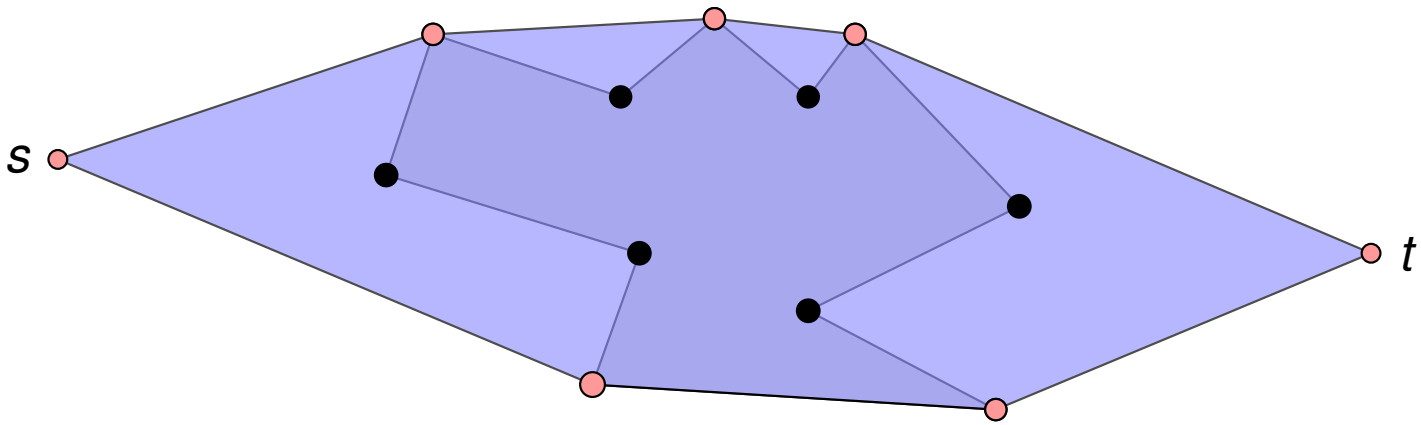


Application of Convex Hull

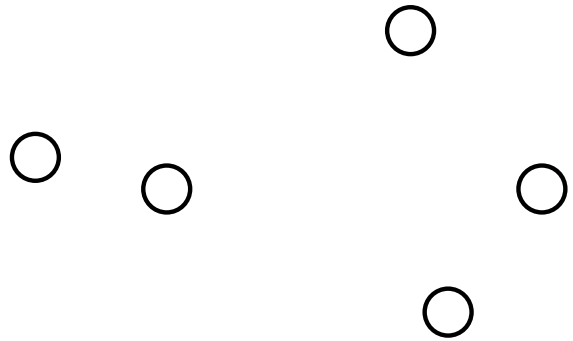
Robot Motion Planning

Find shortest path from *s* to *t* which avoids a polygonal obstacle.

can be solved by computing the Convex hull!



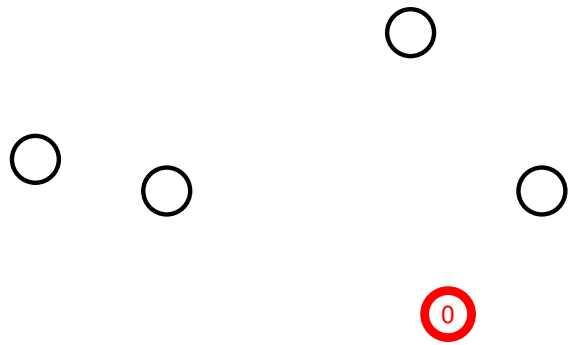
Graham's Scan



Basic Idea

- Start with the point with smallest y-coordinate

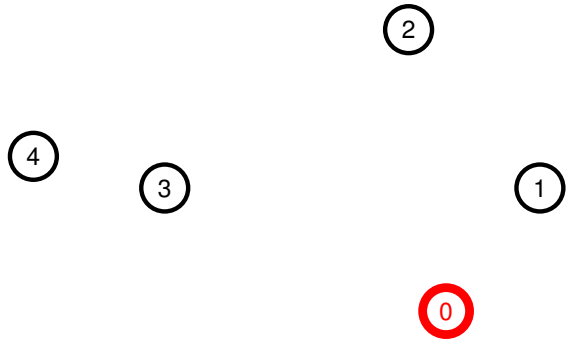
Graham's Scan



Basic Idea

- Start with the point with smallest y-coordinate

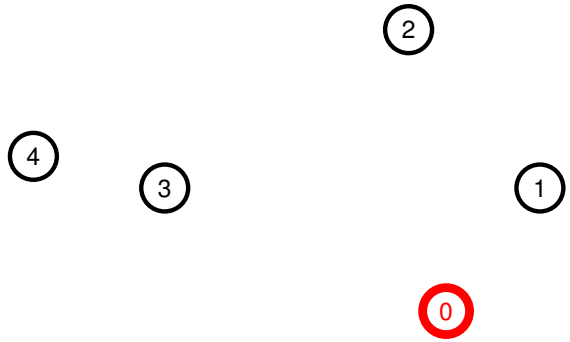
Graham's Scan



Basic Idea

- Start with the point with smallest y-coordinate
- Sort all points increasingly according to their polar angle

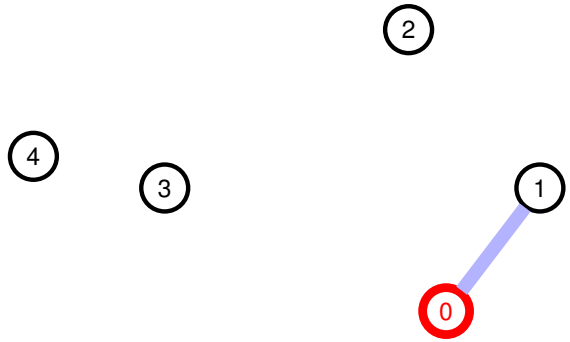
Graham's Scan



Basic Idea

- Start with the point with smallest y-coordinate
- Sort all points increasingly according to their polar angle
- Try to add next point to the convex hull

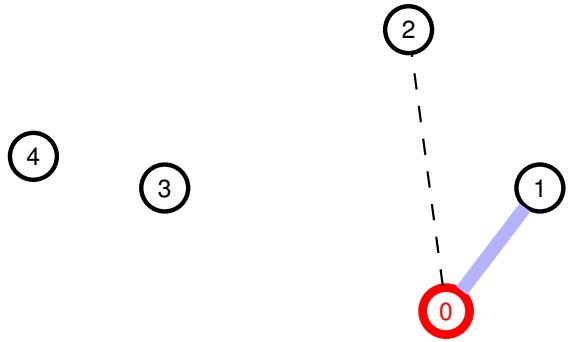
Graham's Scan



Basic Idea

- Start with the point with smallest y-coordinate
- Sort all points increasingly according to their polar angle
- Try to add next point to the convex hull

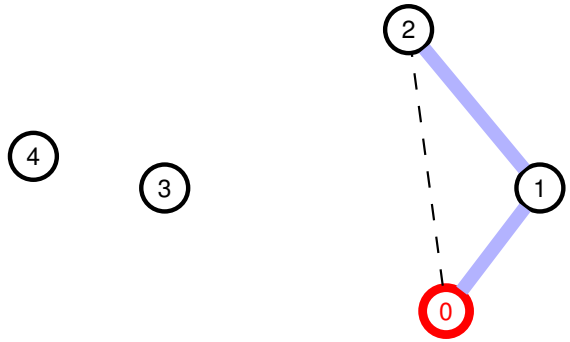
Graham's Scan



Basic Idea

- Start with the point with smallest y-coordinate
- Sort all points increasingly according to their polar angle
- Try to add next point to the convex hull
  - If it does not introduce non-left turn, then fine

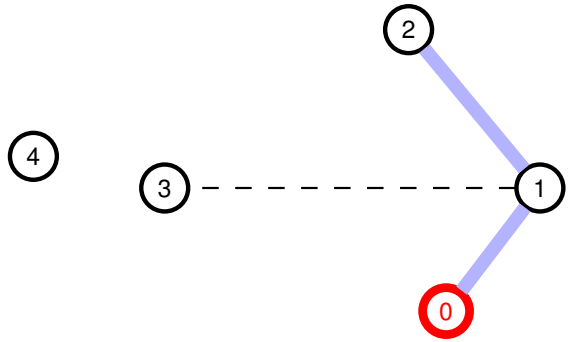
Graham's Scan



Basic Idea

- Start with the point with smallest y-coordinate
- Sort all points increasingly according to their polar angle
- Try to add next point to the convex hull
  - If it does not introduce non-left turn, then fine

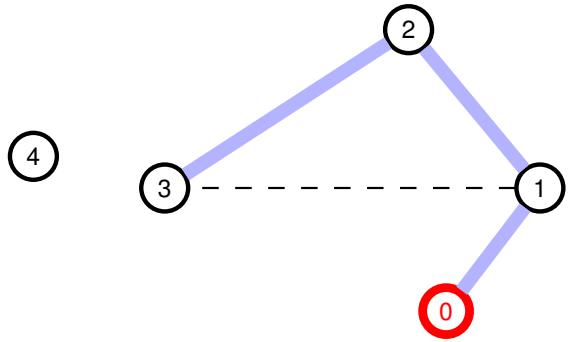
Graham's Scan



Basic Idea

- Start with the point with smallest y-coordinate
- Sort all points increasingly according to their polar angle
- Try to add next point to the convex hull
  - If it does not introduce non-left turn, then fine

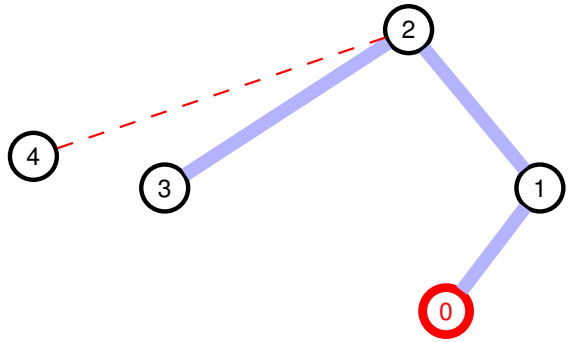
Graham's Scan



Basic Idea

- Start with the point with smallest y-coordinate
- Sort all points increasingly according to their polar angle
- Try to add next point to the convex hull
  - If it does not introduce non-left turn, then fine ✓

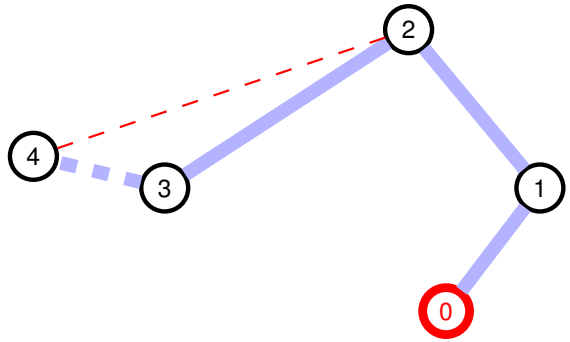
Graham's Scan



Basic Idea

- Start with the point with smallest y-coordinate
- Sort all points increasingly according to their polar angle
- Try to add next point to the convex hull
  - If it does not introduce non-left turn, then fine ✓
  - Otherwise,

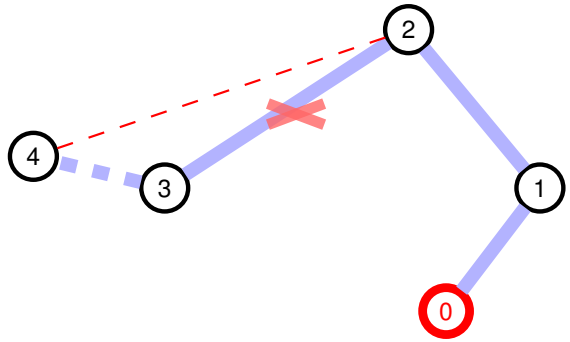
Graham's Scan



Basic Idea

- Start with the point with smallest y-coordinate
- Sort all points increasingly according to their polar angle
- Try to add next point to the convex hull
  - If it does not introduce non-left turn, then fine ✓
  - Otherwise, keep on removing recent points until point can be added

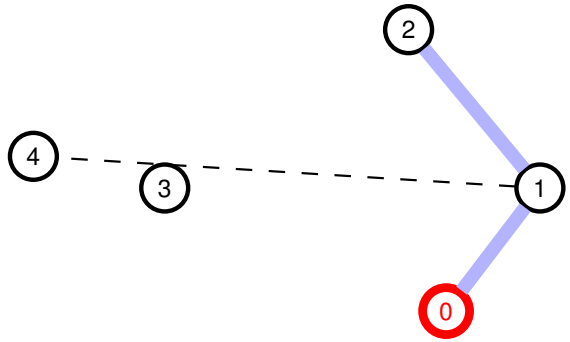
Graham's Scan



Basic Idea

- Start with the point with smallest y-coordinate
- Sort all points increasingly according to their polar angle
- Try to add next point to the convex hull
  - If it does not introduce non-left turn, then fine ✓
  - Otherwise, keep on removing recent points until point can be added

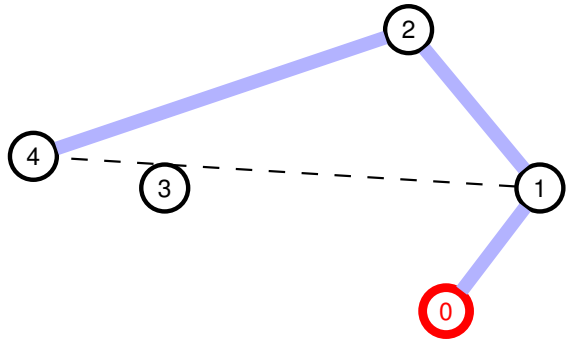
Graham's Scan



Basic Idea

- Start with the point with smallest y-coordinate
- Sort all points increasingly according to their polar angle
- Try to add next point to the convex hull
  - If it does not introduce non-left turn, then fine ✓
  - Otherwise, keep on removing recent points until point can be added

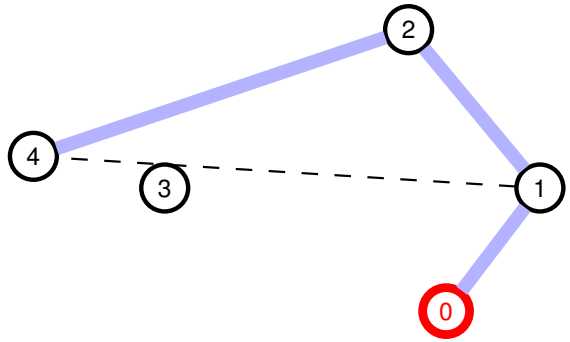
Graham's Scan



Basic Idea

- Start with the point with smallest y-coordinate
- Sort all points increasingly according to their polar angle
- Try to add next point to the convex hull
  - If it does not introduce non-left turn, then fine ✓
  - Otherwise, keep on removing recent points until point can be added

Graham's Scan

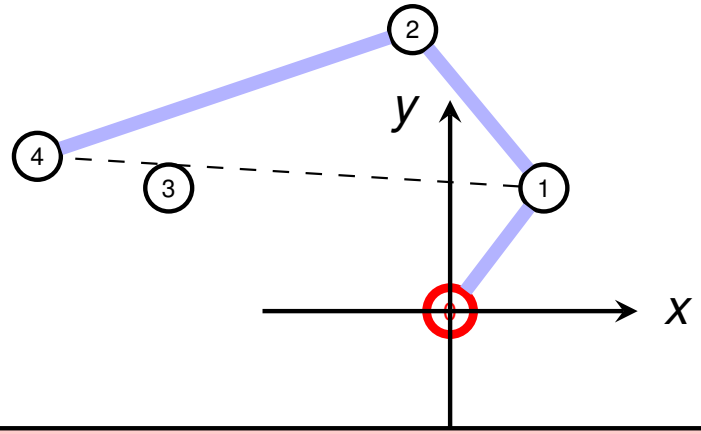


Efficient Sorting by comparing (not computing!) polar angles

Basic Idea

- Start with the point with smallest y-coordinate
- Sort all points increasingly according to their polar angle
- Try to add next point to the convex hull
  - If it does not introduce non-left turn, then fine ✓
  - Otherwise, keep on removing recent points until point can be added

Graham's Scan

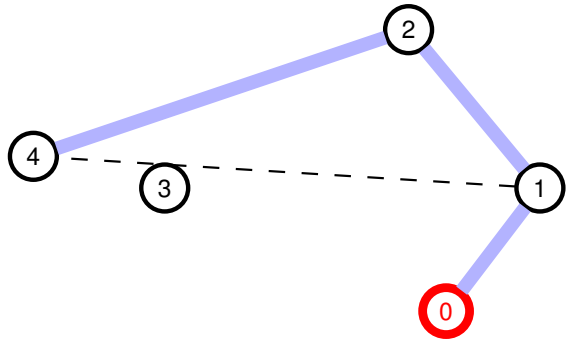


Efficient Sorting by comparing (not computing!) polar angles

Basic Idea

- Start with the point with smallest y-coordinate
- Sort all points increasingly according to their polar angle
- Try to add next point to the convex hull
  - If it does not introduce non-left turn, then fine ✓
  - Otherwise, keep on removing recent points until point can be added

Graham's Scan

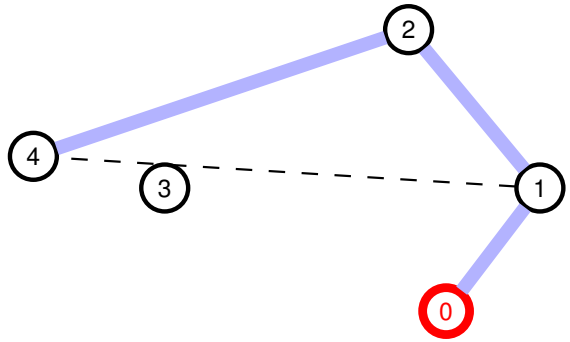


Efficient Sorting by comparing (not computing!) polar angles

Basic Idea

- Start with the point with smallest y-coordinate
- Sort all points increasingly according to their polar angle
- Try to add next point to the convex hull
  - If it does not introduce non-left turn, then fine ✓
  - Otherwise, keep on removing recent points until point can be added

Graham's Scan



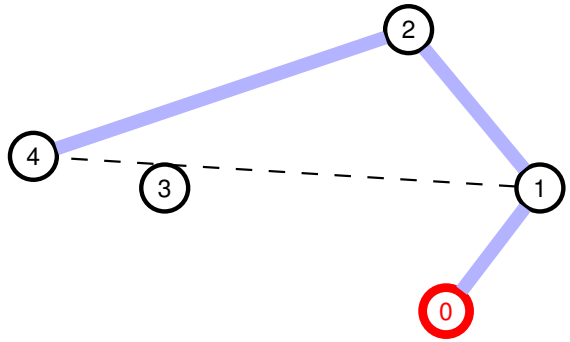
Use Cross Product!

Efficient Sorting by comparing (not computing!) polar angles

Basic Idea

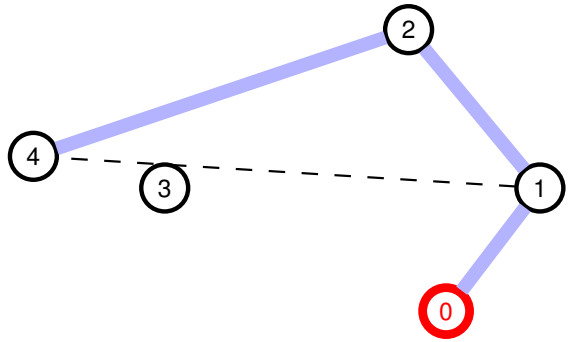
- Start with the point with smallest y-coordinate
- Sort all points increasingly according to their polar angle
- Try to add next point to the convex hull
  - If it does not introduce non-left turn, then fine ✓
  - Otherwise, keep on removing recent points until point can be added

Graham's Scan



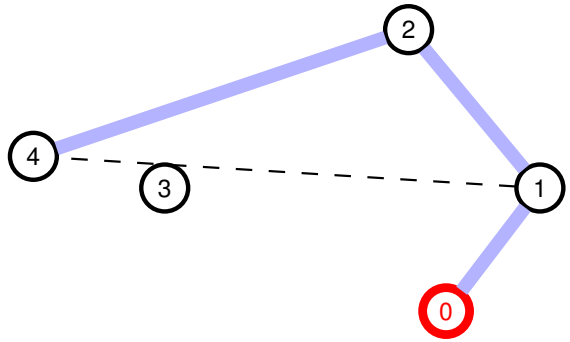

```
 0: GRAHAM-SCAN(Q)
 1:     Let p_0 be the point with minimum y-coordinate
 2:     Let (p_1, p_2, ..., p_n) be the other points sorted by polar angle w.r.t. p_0
 3:     If n < 2 return false
 4:     S = ∅
 5:     PUSH(p_0,S)
 6:     PUSH(p_1,S)
 7:     PUSH(p_2,S)
 8:     For i = 3 to n
 9:         While angle of NEXT-TO-TOP(S),TOP(S),p_i makes a non-left turn
10:             POP(S)
11:         End While
12:         PUSH(p_i,S)
13:     End For
14:     Return S
```

Graham's Scan




```
 0: GRAHAM-SCAN(Q)
 1:     Let p_0 be the point with minimum y-coordinate
 2:     Let (p_1, p_2, ..., p_n) be the other points sorted by polar angle w.r.t. p_0
 3:     If n < 2 return false
 4:     S = ∅
 5:     PUSH(p_0,S)
 6:     PUSH(p_1,S)
 7:     PUSH(p_2,S)
 8:     For i = 3 to n
 9:         While angle of NEXT-TO-TOP(S),TOP(S),p_i makes a non-left turn
10:             POP(S)
11:         End While
12:         PUSH(p_i,S)
13:     End For
14:     Return S
```

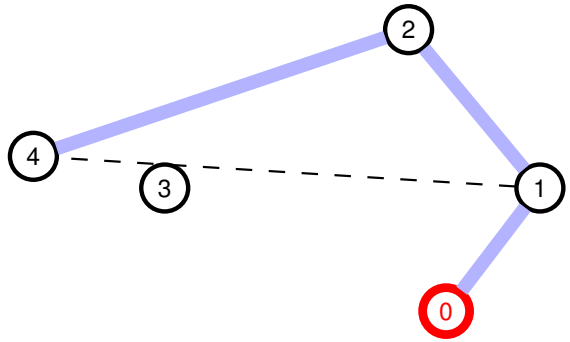
Graham's Scan




```
0:  GRAHAM-SCAN(Q)
1:      Let p_0 be the point with minimum y-coordinate
2:      Let (p_1, p_2, ..., p_n) be the other points sorted by polar angle w.r.t. p_0
3:      If n < 2 return false
4:      S = ∅
5:      PUSH(p_0,S)
6:      PUSH(p_1,S)
7:      PUSH(p_2,S)
8:      For i = 3 to n
9:          While angle of NEXT-TO-TOP(S),TOP(S),p_i makes a non-left turn
10:             POP(S)
11:         End While
12:         PUSH(p_i,S)
13:     End For
14:     Return S
```

Takes $O(n \log n)$ time

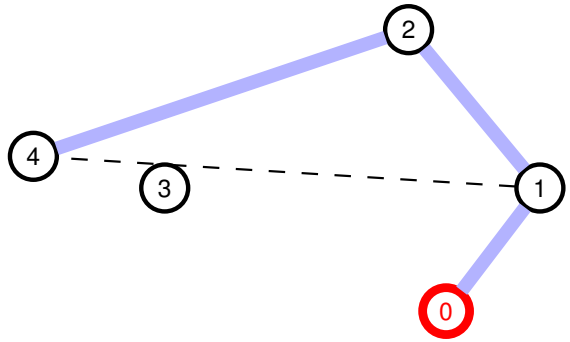
Graham's Scan




```
0:  GRAHAM-SCAN(Q)
1:      Let p_0 be the point with minimum y-coordinate
2:      Let (p_1, p_2, ..., p_n) be the other points sorted by polar angle w.r.t. p_0
3:      If n < 2 return false
4:      S = ∅
5:      PUSH(p_0,S)
6:      PUSH(p_1,S)
7:      PUSH(p_2,S)
8:      For i = 3 to n
9:          While angle of NEXT-TO-TOP(S),TOP(S),p_i makes a non-left turn
10:             POP(S)
11:         End While
12:         PUSH(p_i,S)
13:     End For
14:     Return S
```

Takes $O(n \log n)$ time

Graham's Scan

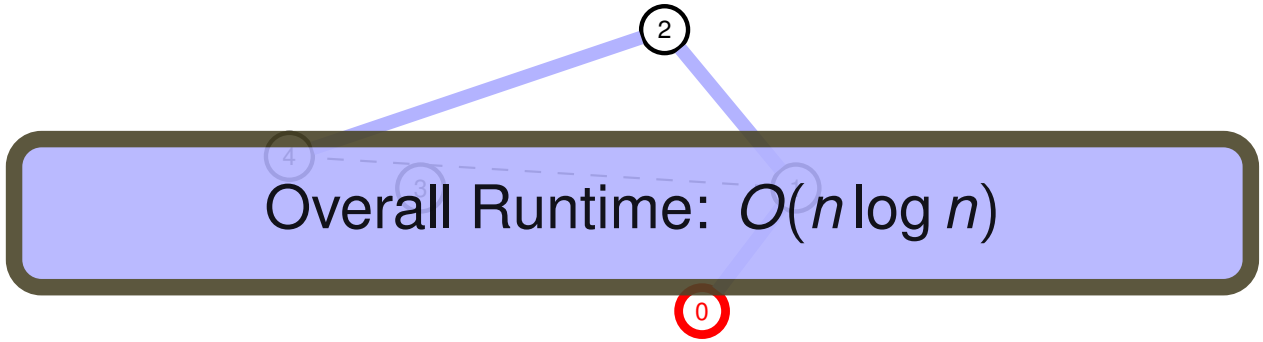



```
0:  GRAHAM-SCAN(Q)
1:      Let p_0 be the point with minimum y-coordinate
2:      Let (p_1, p_2, ..., p_n) be the other points sorted by polar angle w.r.t. p_0
3:      If n < 2 return false
4:      S = ∅
5:      PUSH(p_0,S)
6:      PUSH(p_1,S)
7:      PUSH(p_2,S)
8:      For i = 3 to n
9:          While angle of NEXT-TO-TOP(S),TOP(S),p_i makes a non-left turn
10:             POP(S)
11:         End While
12:         PUSH(p_i,S)
13:     End For
14:     Return S
```

Takes $O(n \log n)$ time

Takes $O(n)$ time, since every point is part of a PUSH or POP at most once.

Graham's Scan

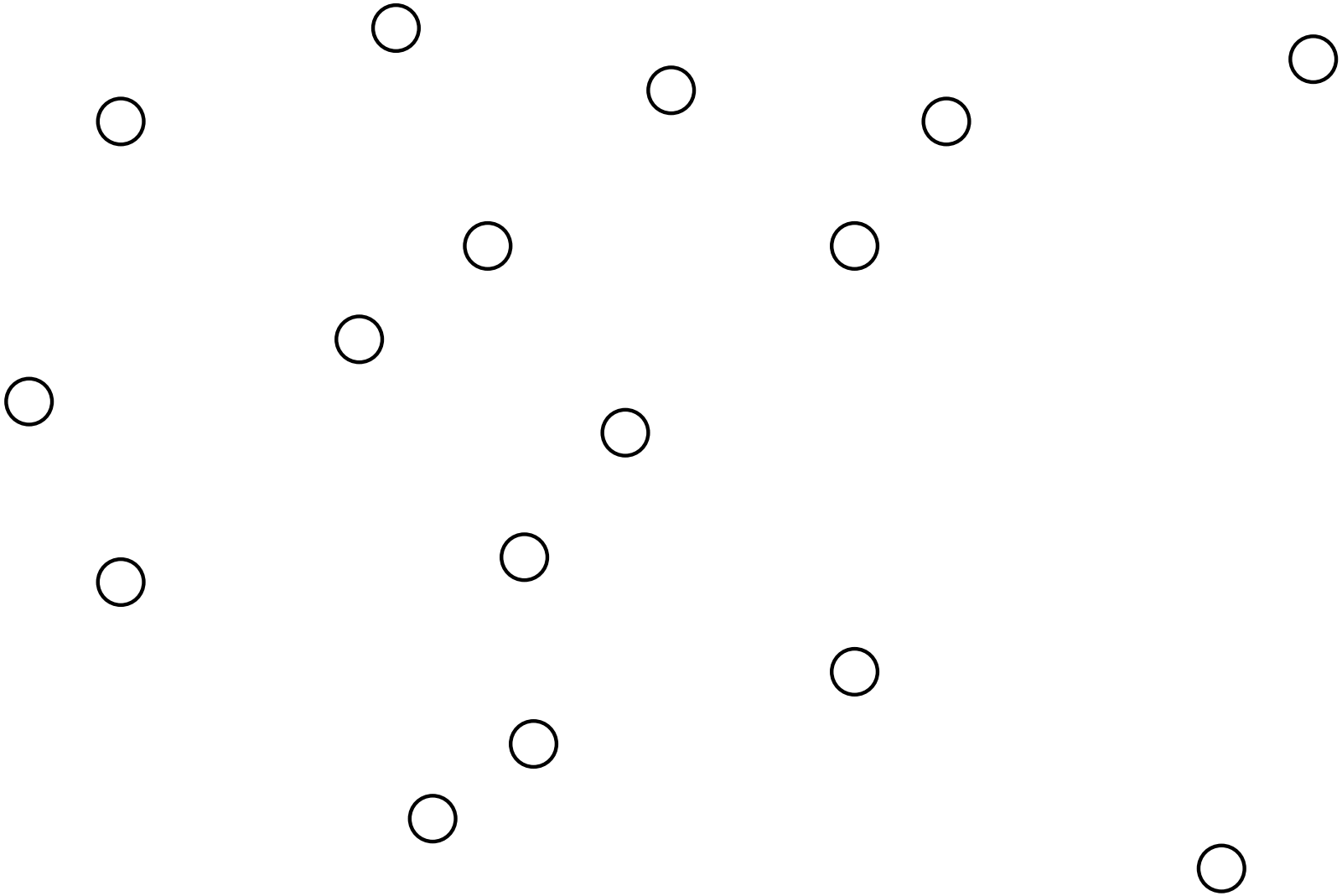



```
0: GRAHAM-SCAN(Q)
1:     Let p_0 be the point with minimum y-coordinate
2:     Let (p_1, p_2, ..., p_n) be the other points sorted by polar angle w.r.t. p_0
3:     If n < 2 return false
4:     S = ∅
5:     PUSH(p_0,S)
6:     PUSH(p_1,S)
7:     PUSH(p_2,S)
8:     For i = 3 to n
9:         While angle of NEXT-TO-TOP(S),TOP(S),p_i makes a non-left turn
10:            POP(S)
11:        End While
12:        PUSH(p_i,S)
13:    End For
14:    Return S
```

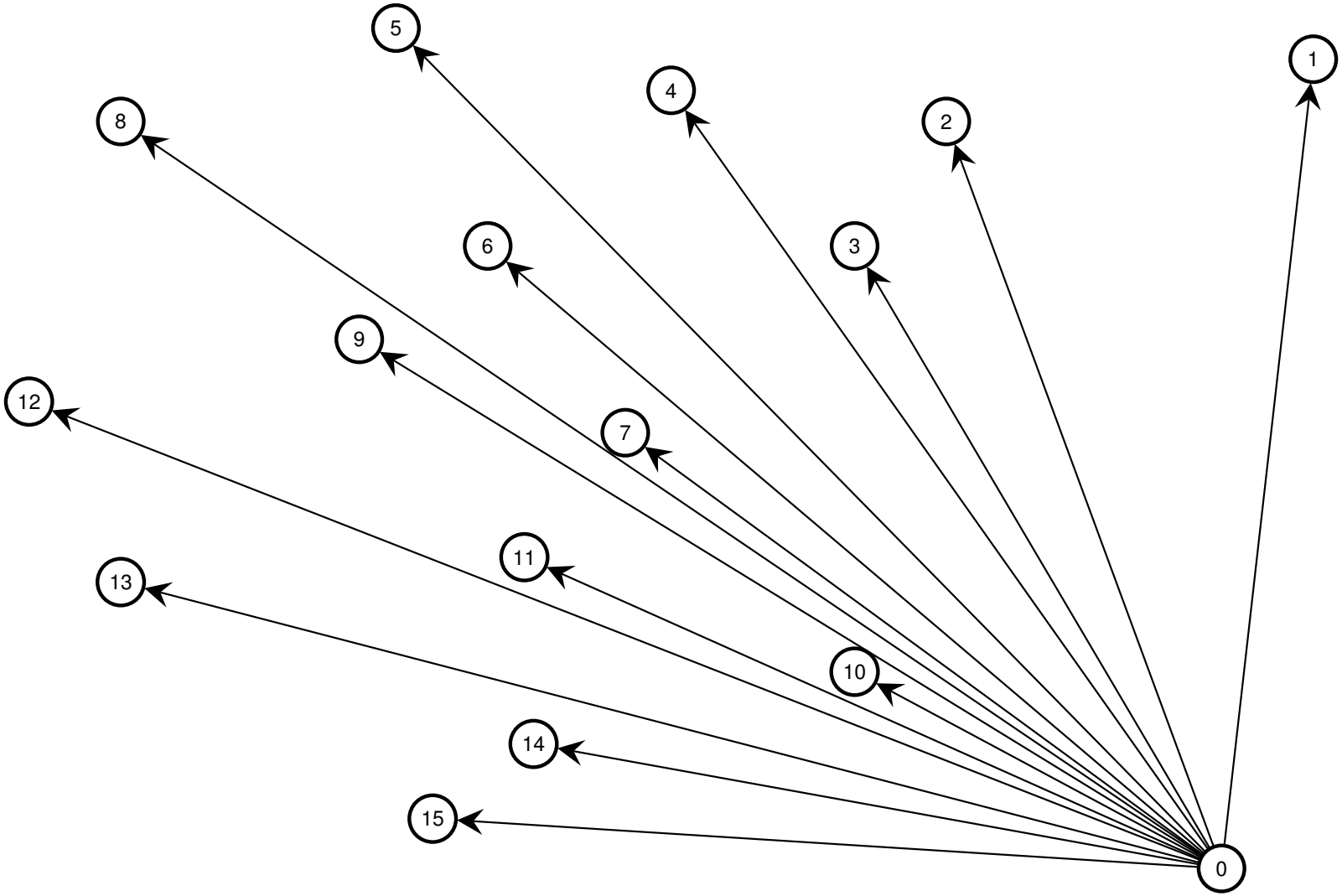
Takes $O(n \log n)$ time

Takes $O(n)$ time, since every point is part of a PUSH or POP at most once.

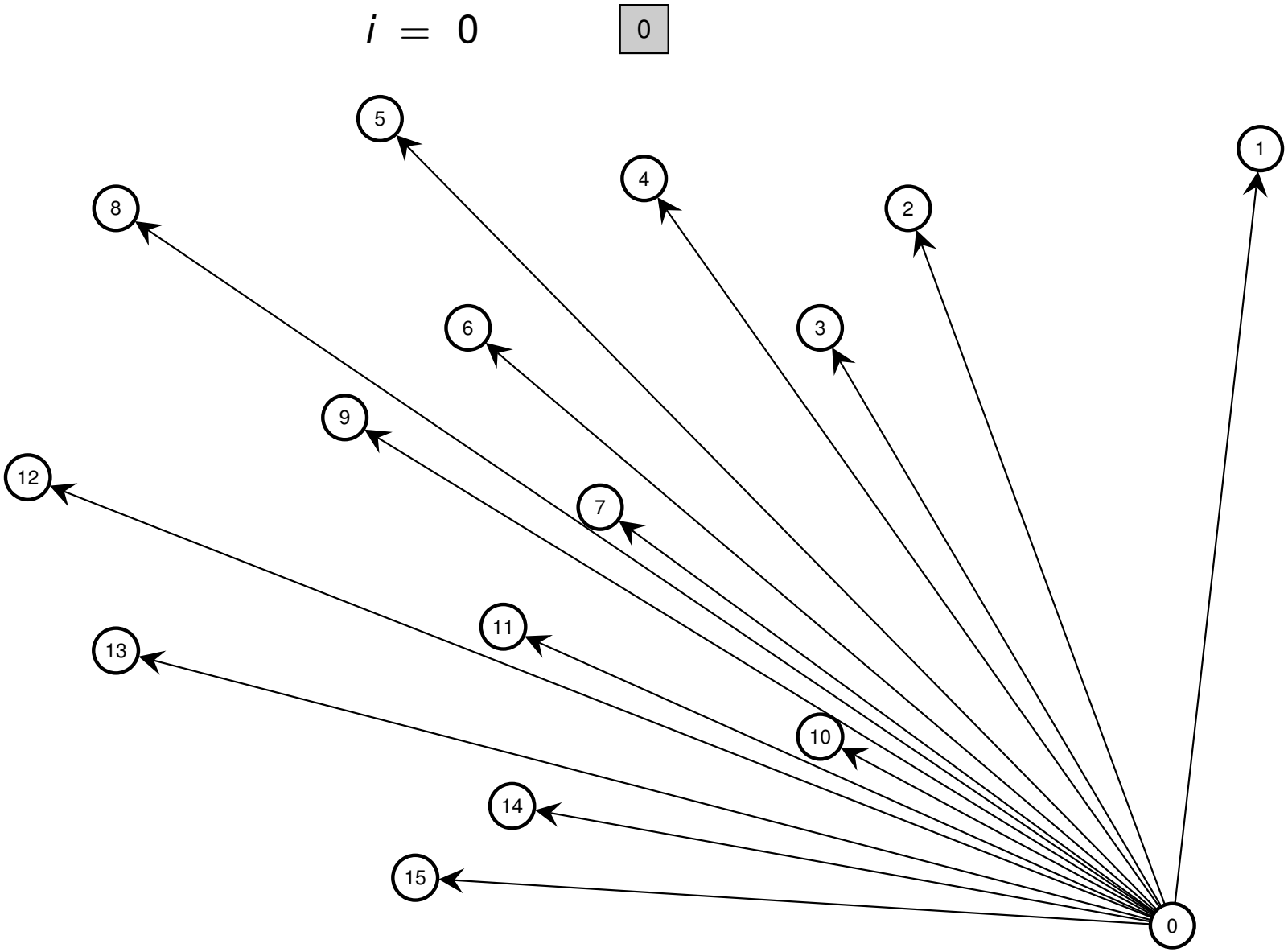
Complete Run of Graham's Scan



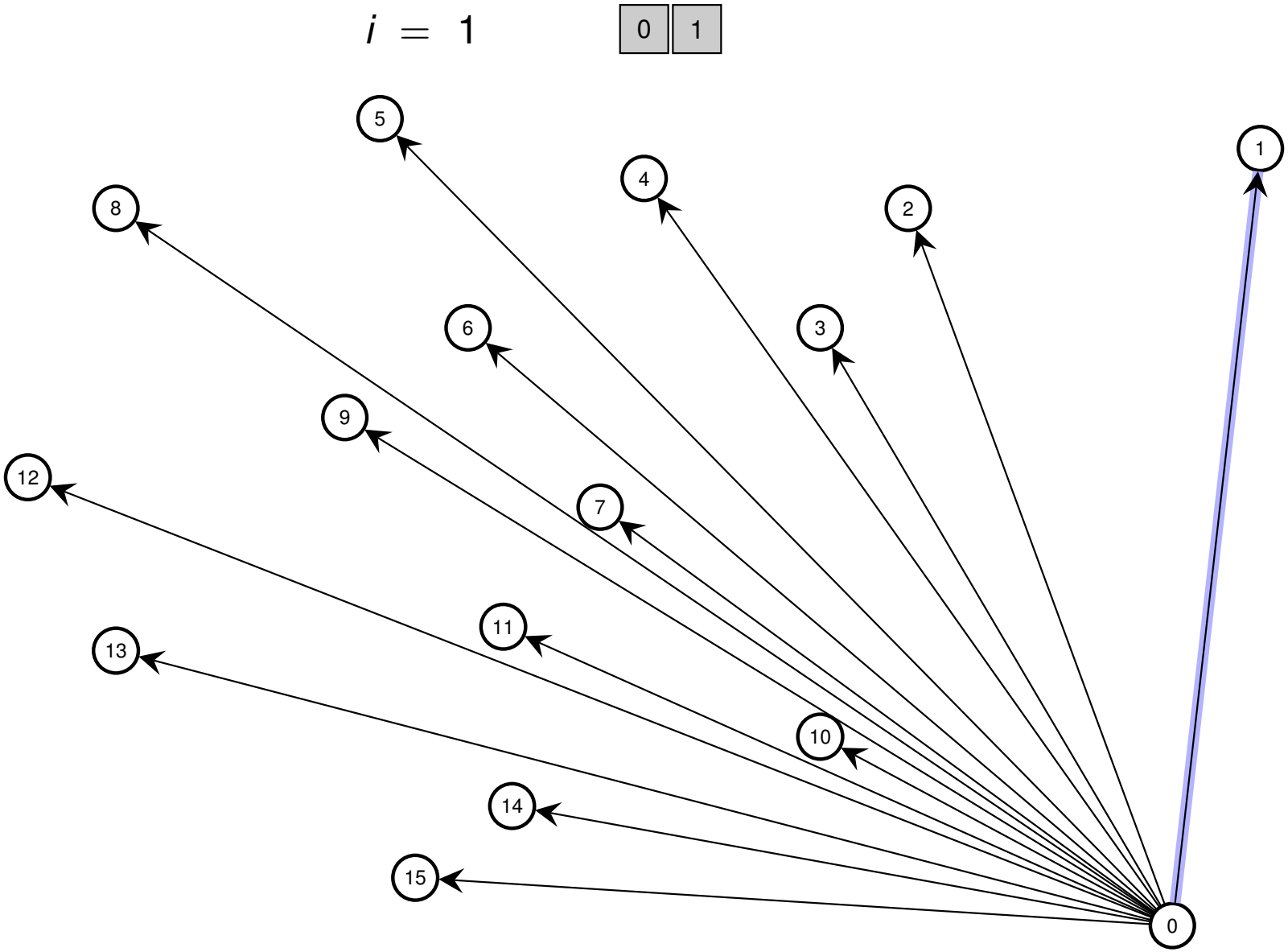
Complete Run of Graham's Scan



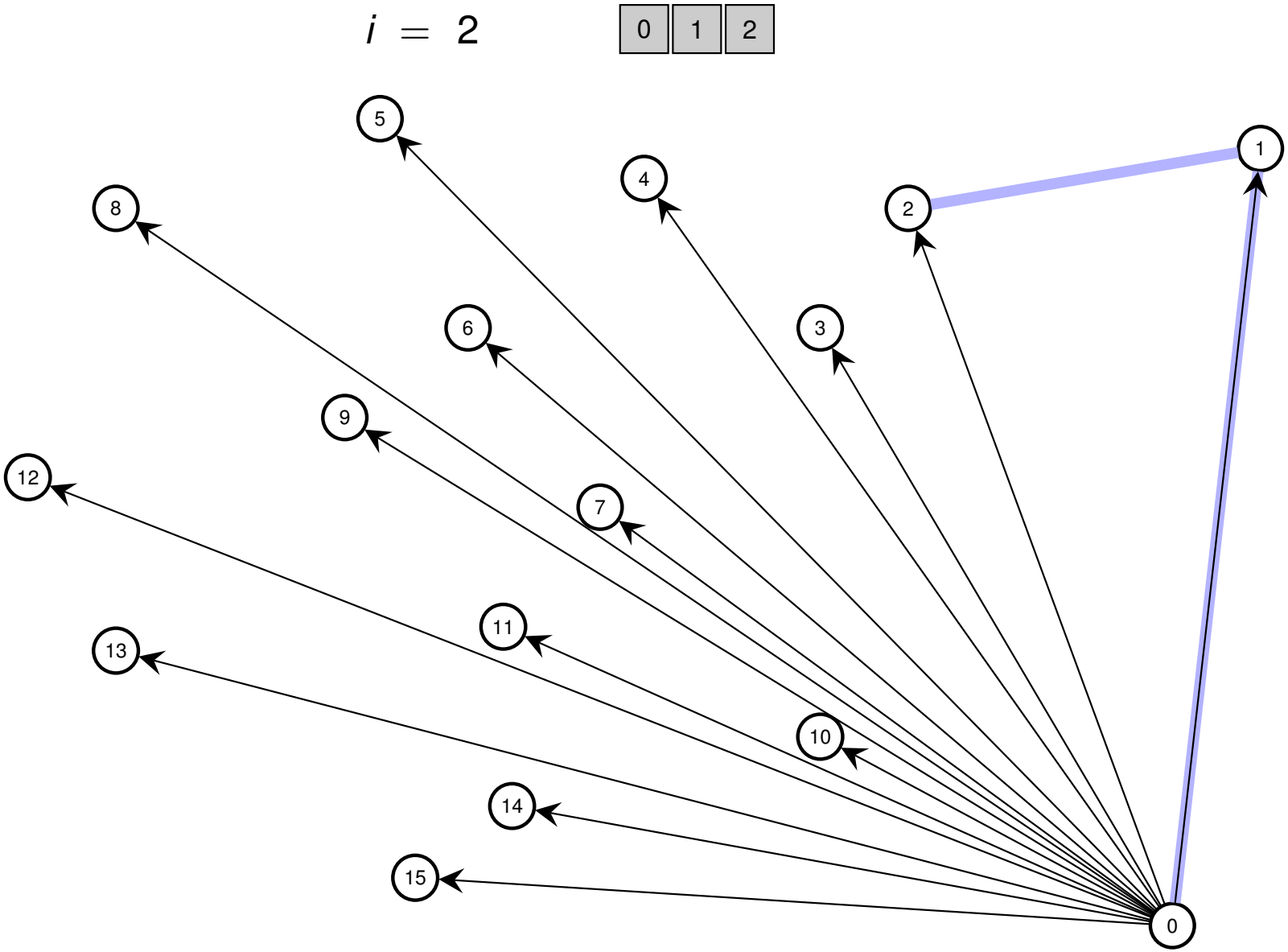
Complete Run of Graham's Scan



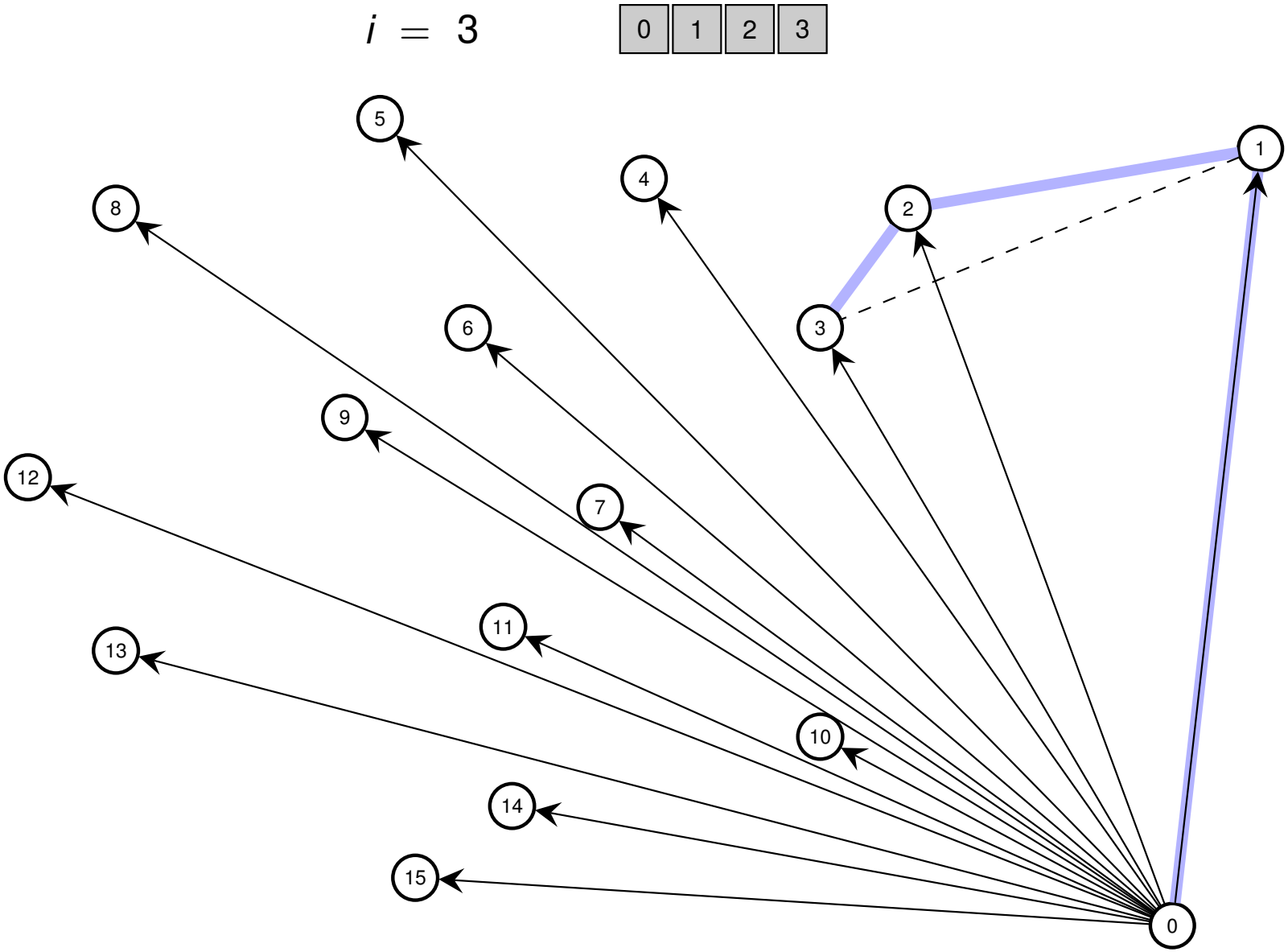
Complete Run of Graham's Scan



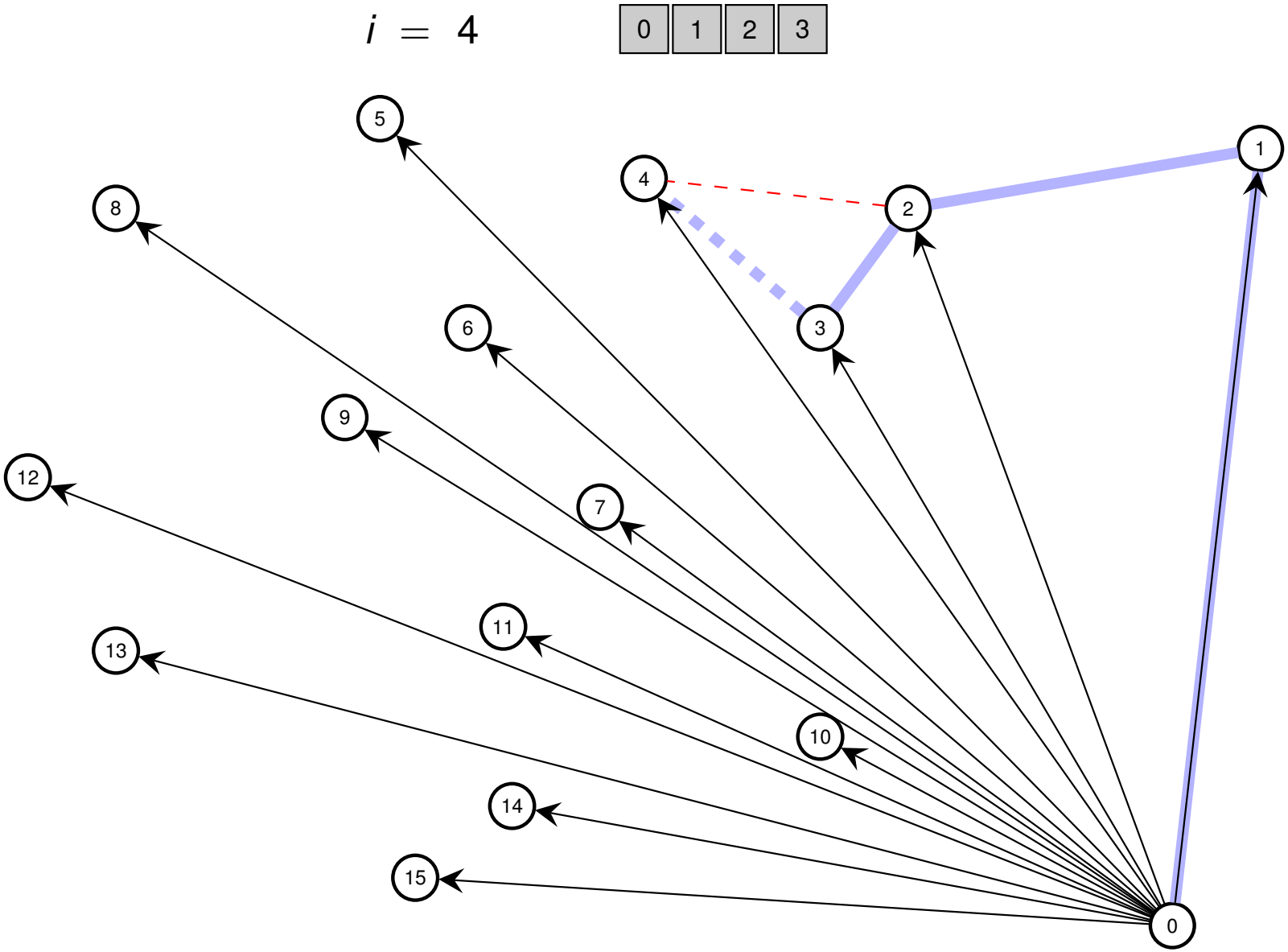
Complete Run of Graham's Scan



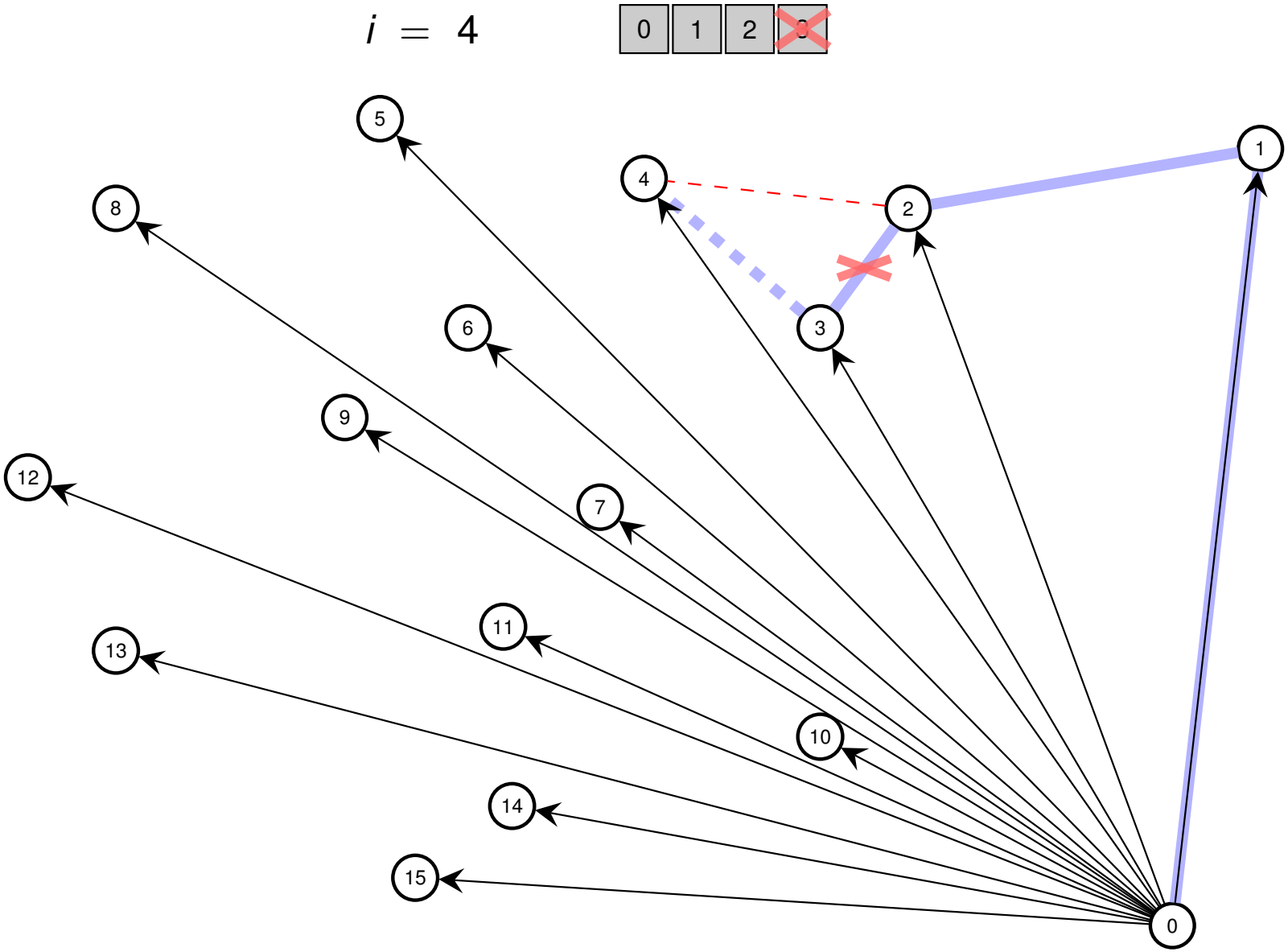
Complete Run of Graham's Scan



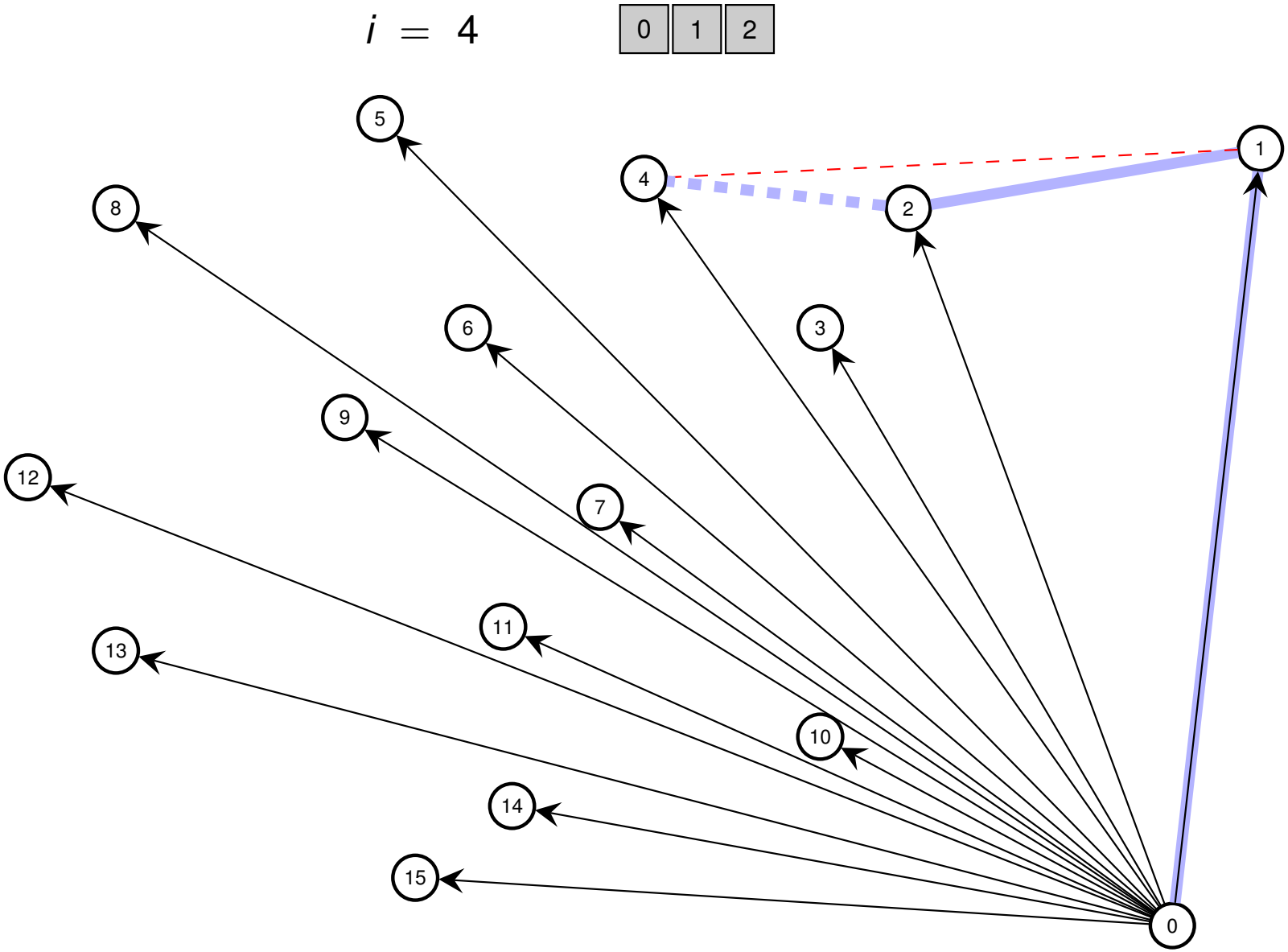
Complete Run of Graham's Scan



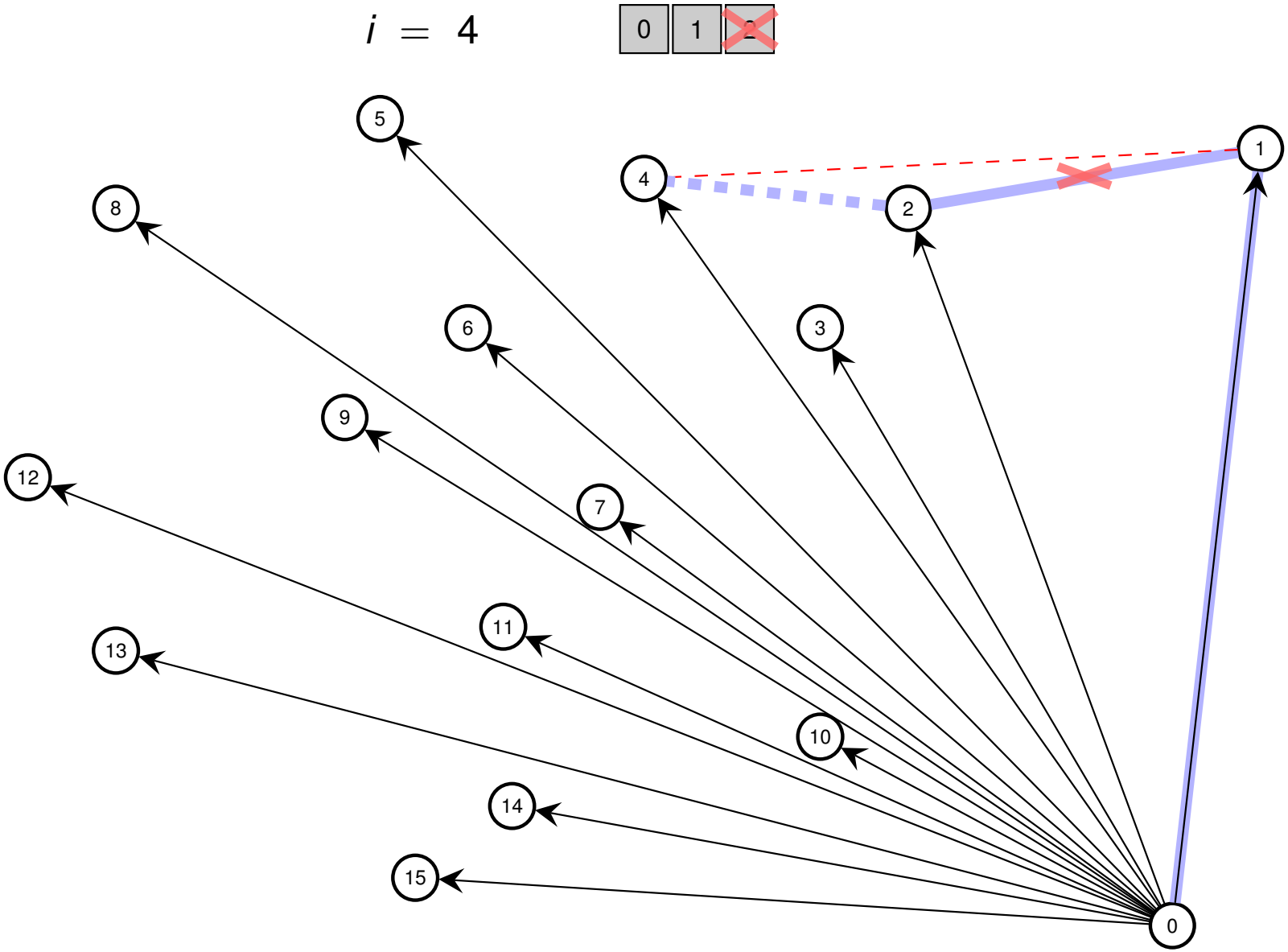
Complete Run of Graham's Scan



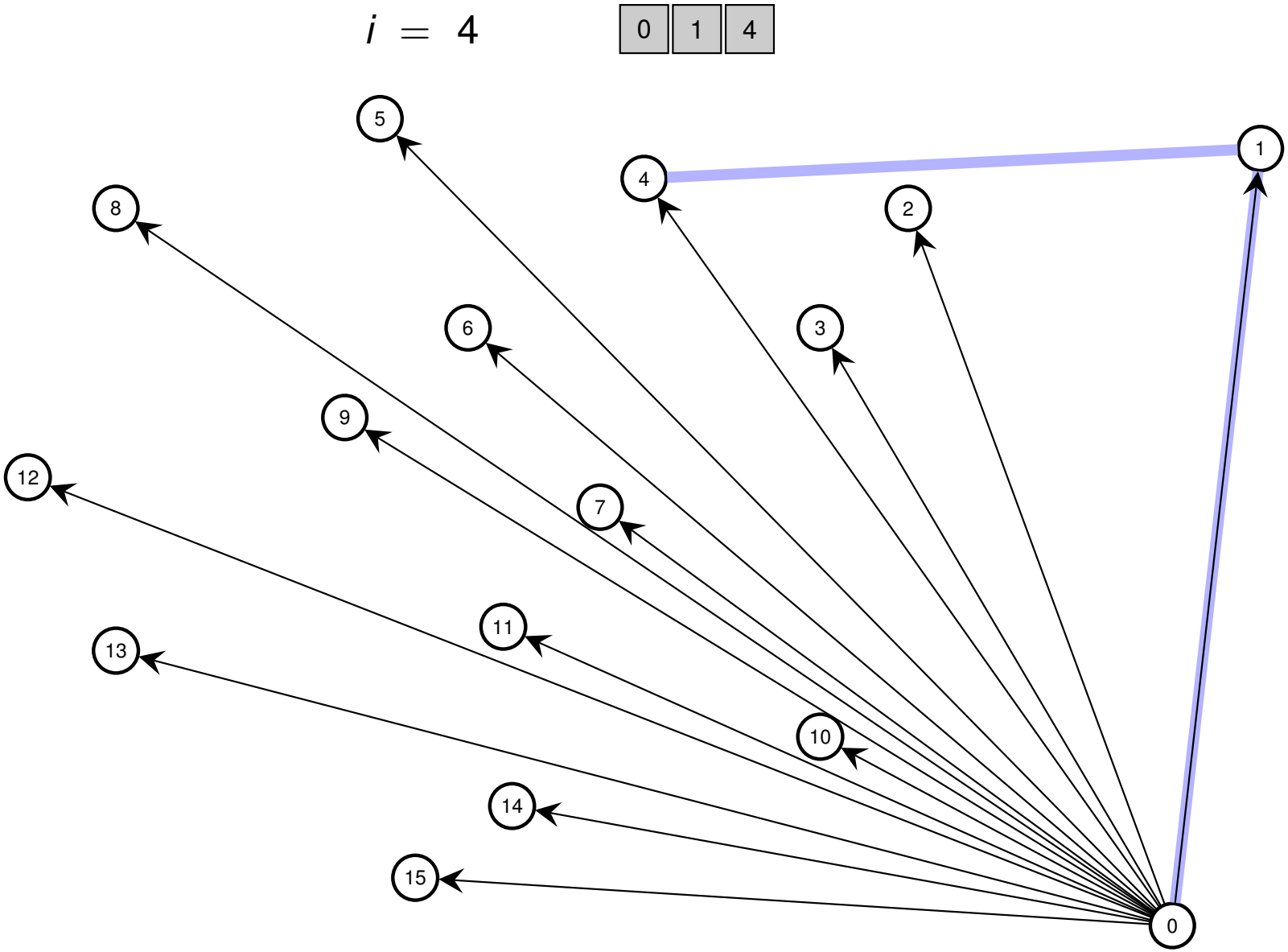
Complete Run of Graham's Scan



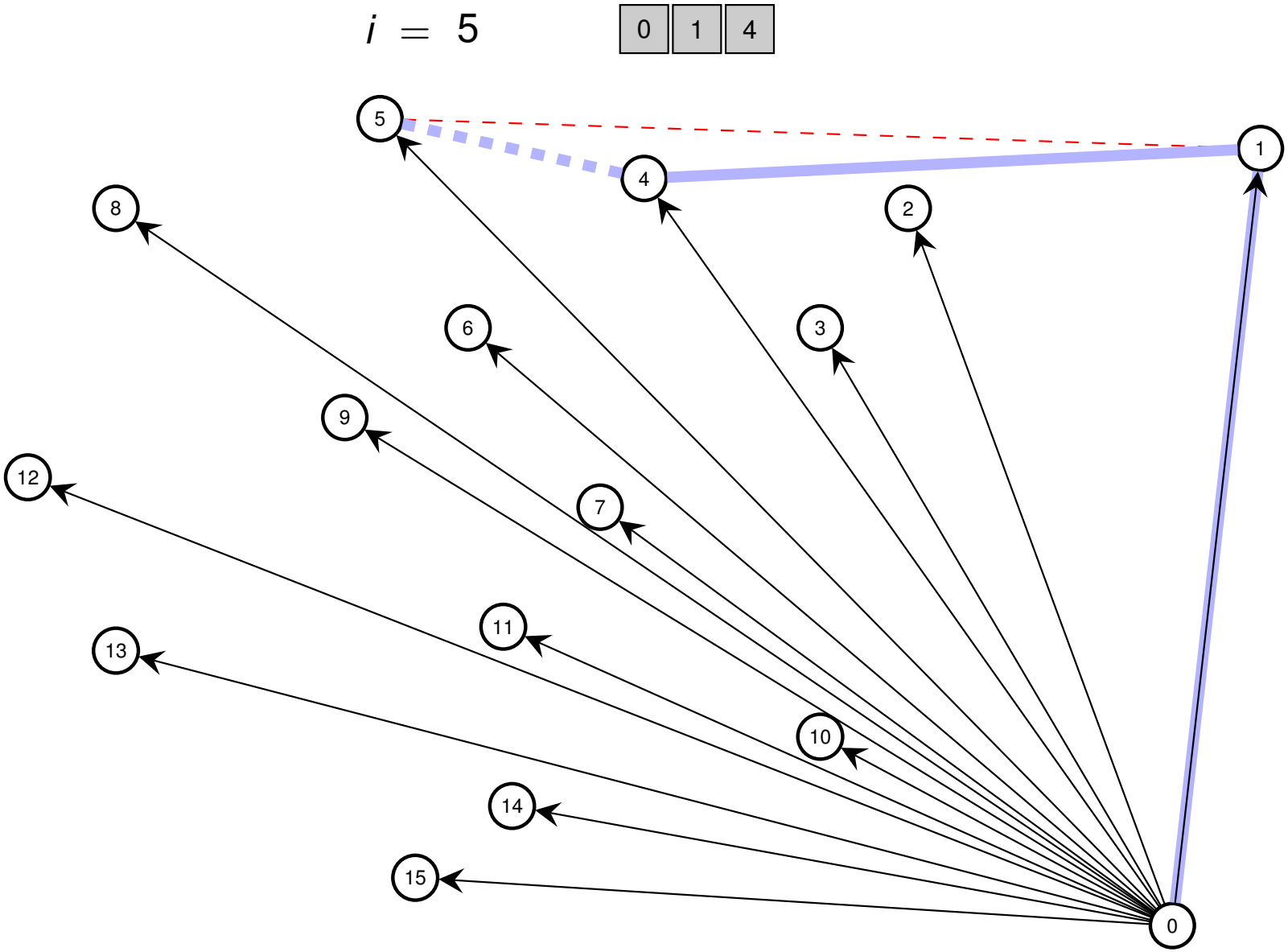
Complete Run of Graham's Scan



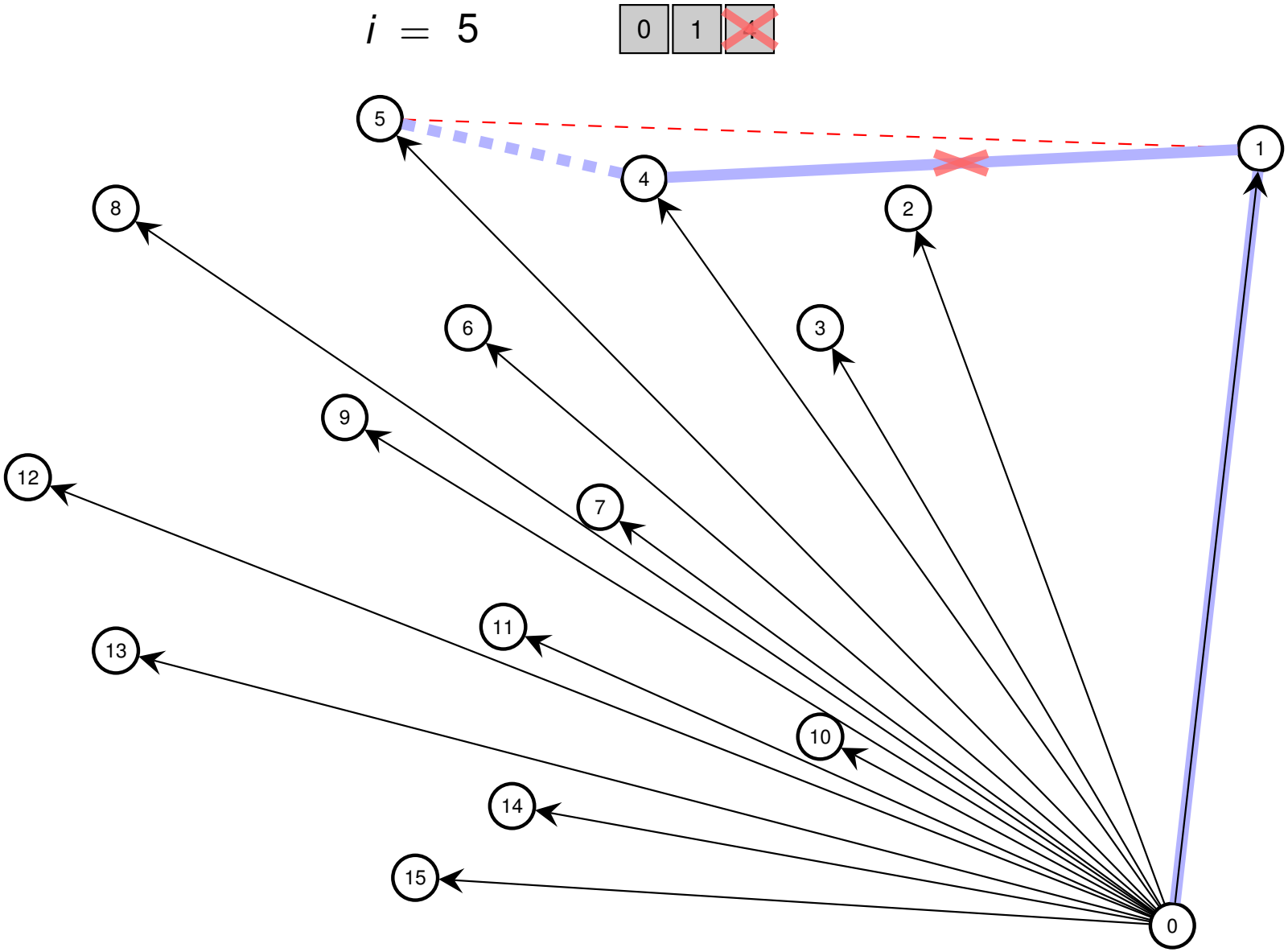
Complete Run of Graham's Scan



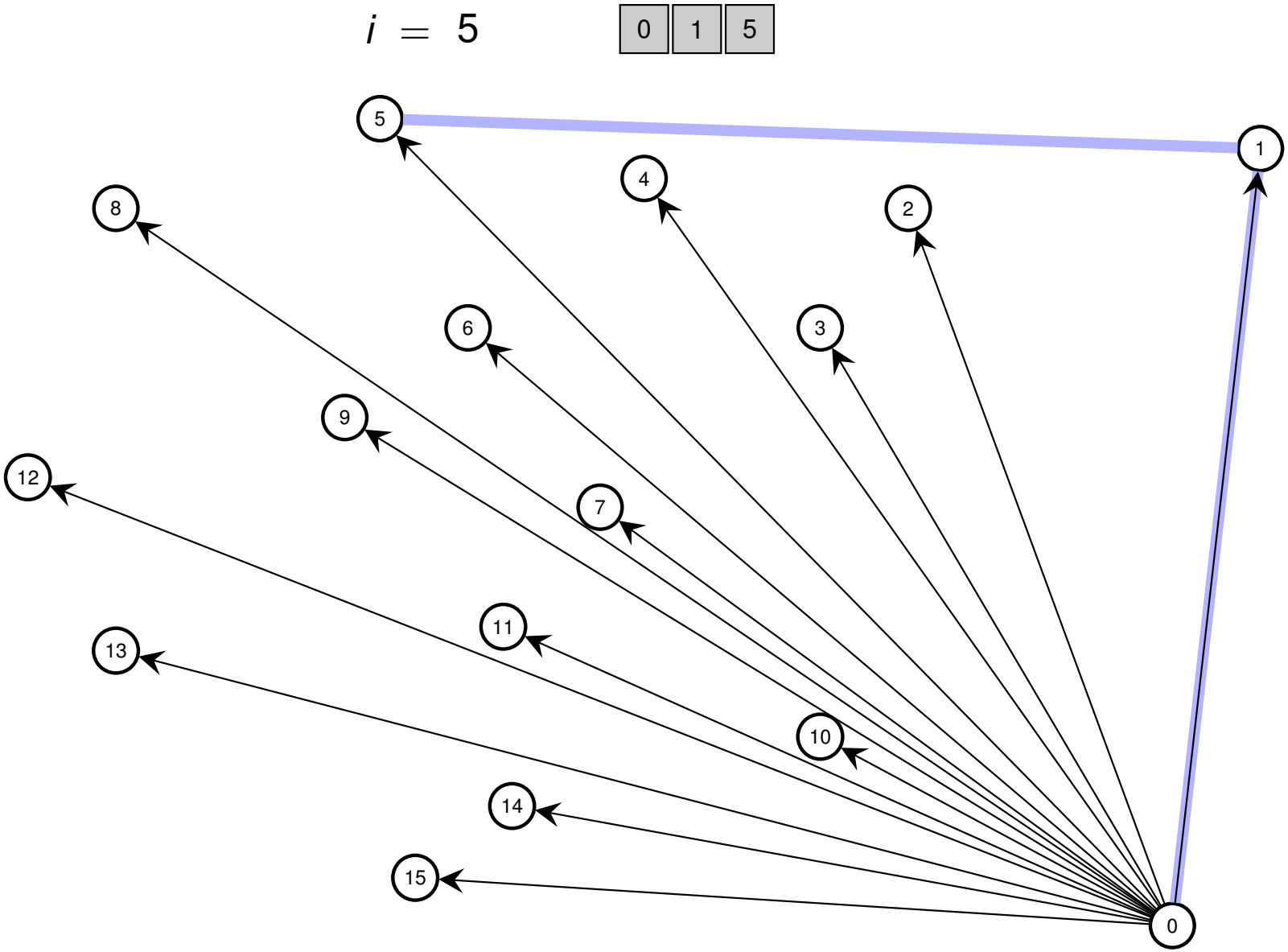
Complete Run of Graham's Scan



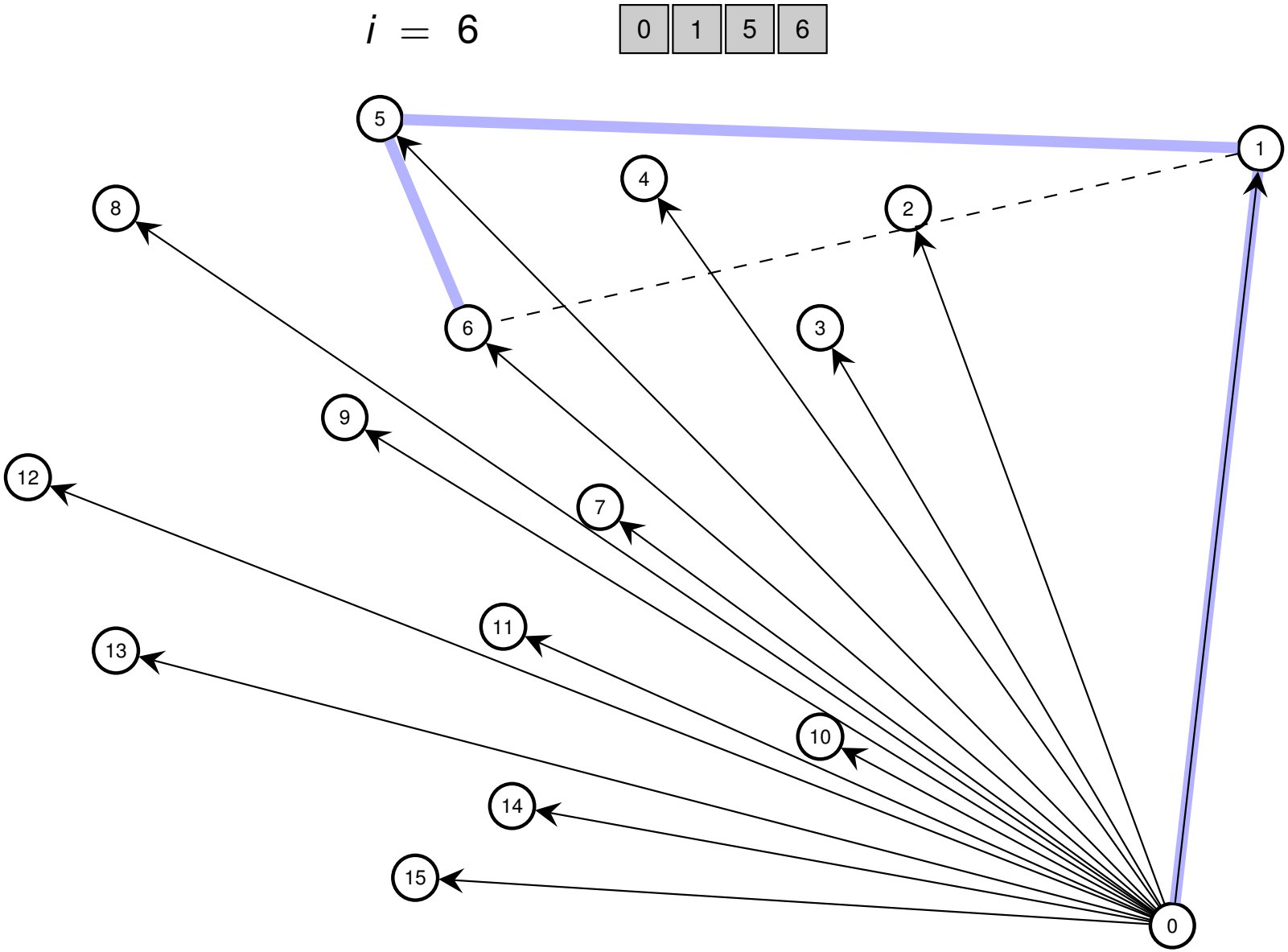
Complete Run of Graham's Scan



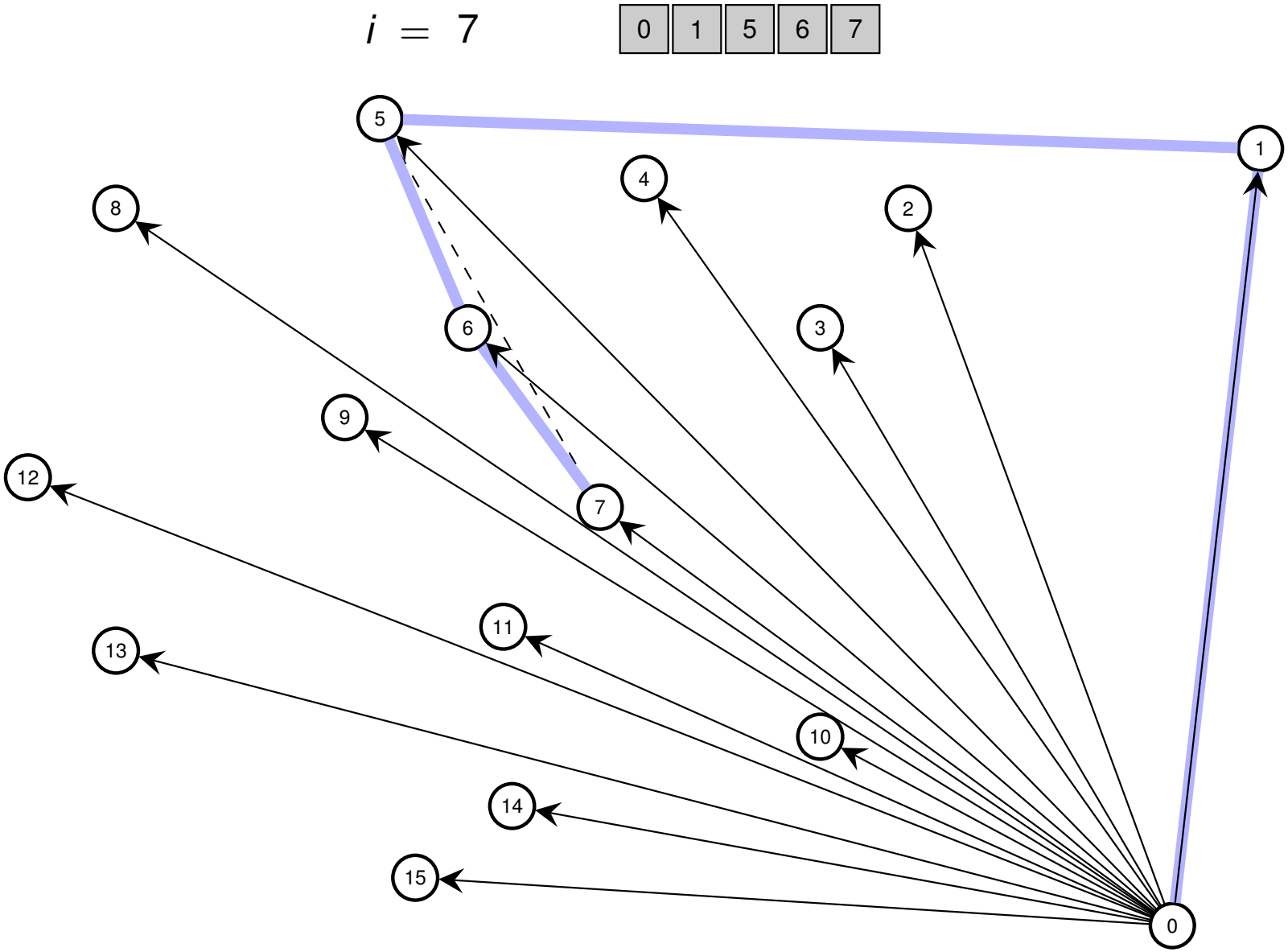
Complete Run of Graham's Scan



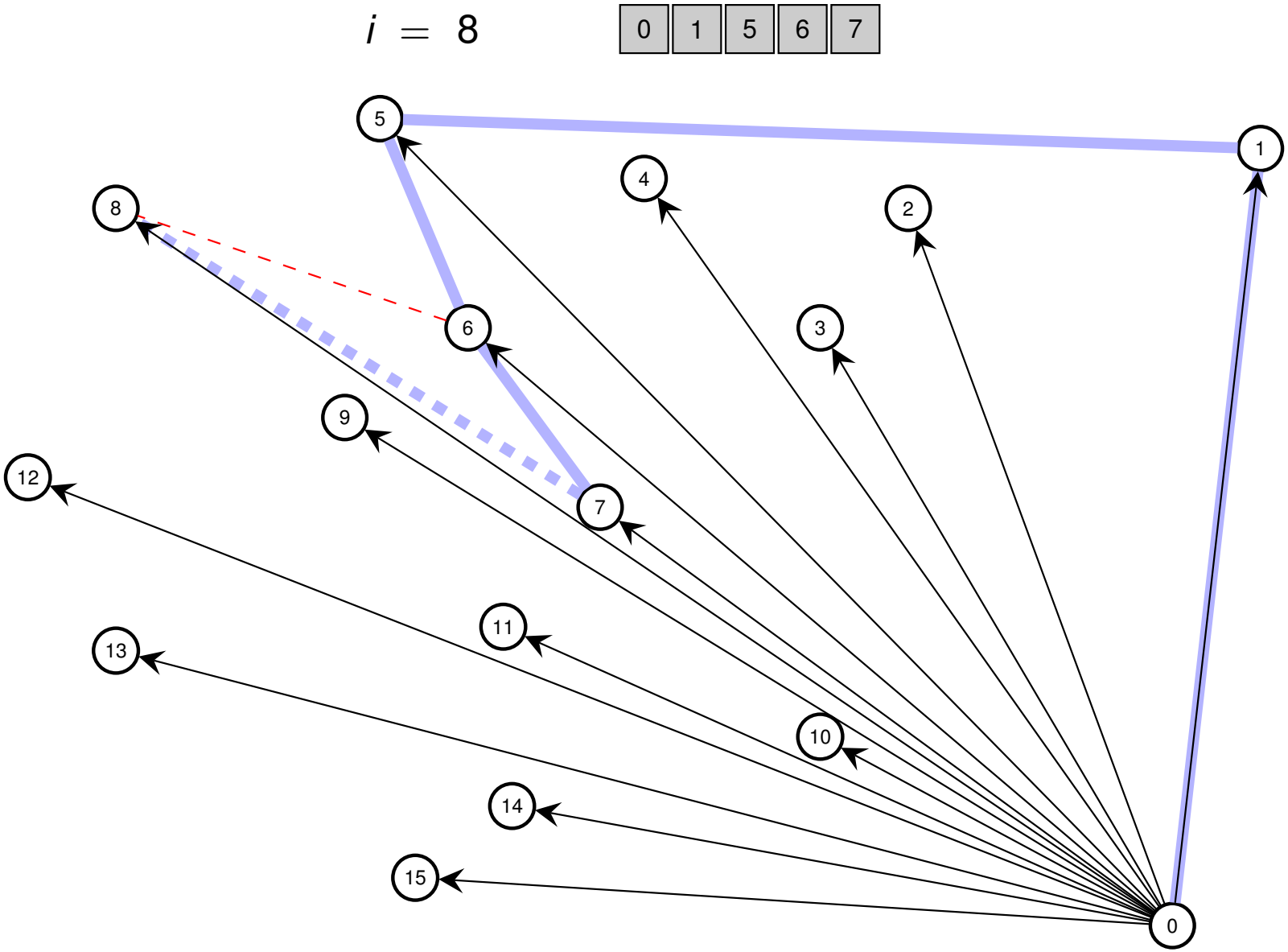
Complete Run of Graham's Scan



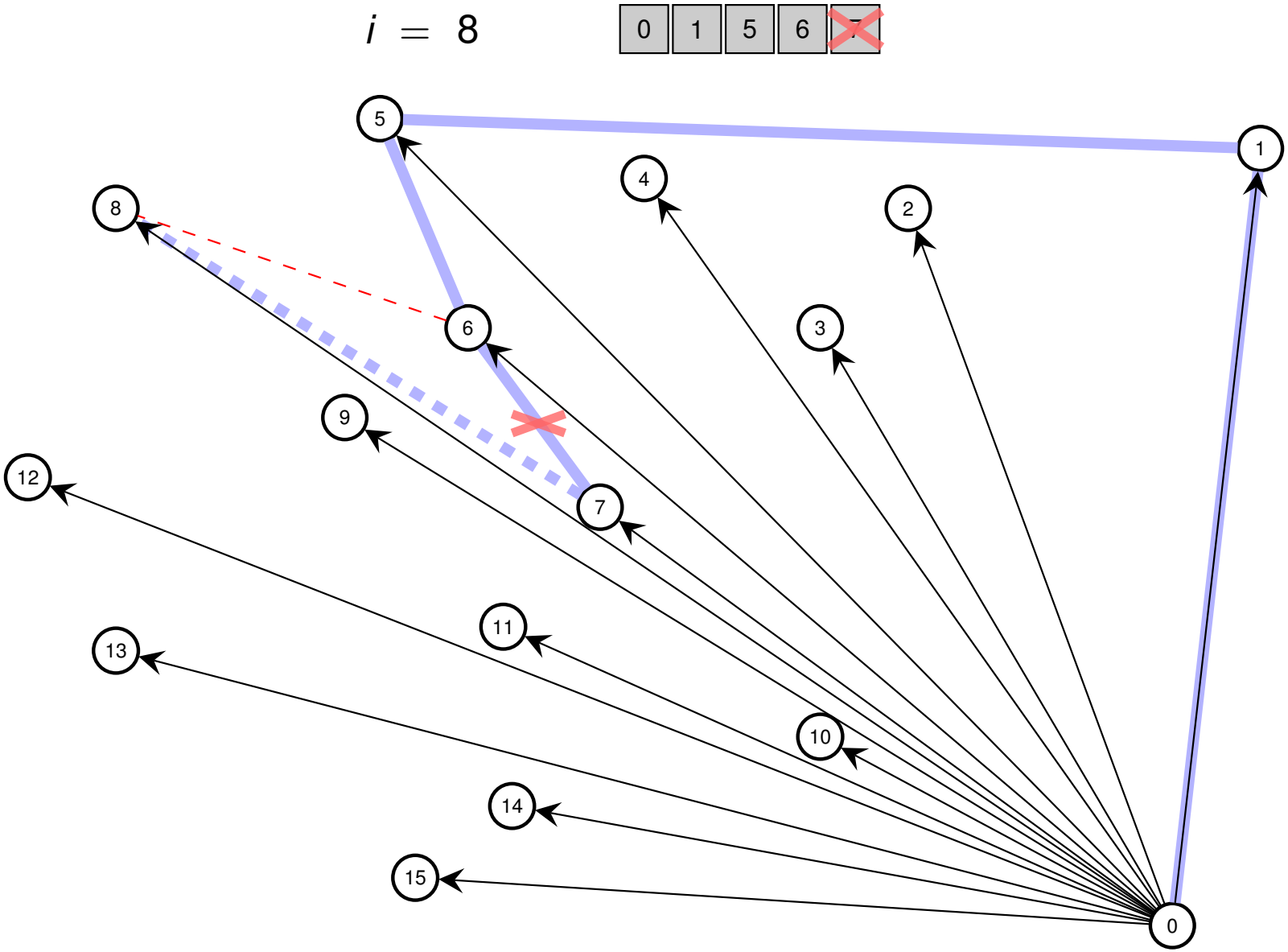
Complete Run of Graham's Scan



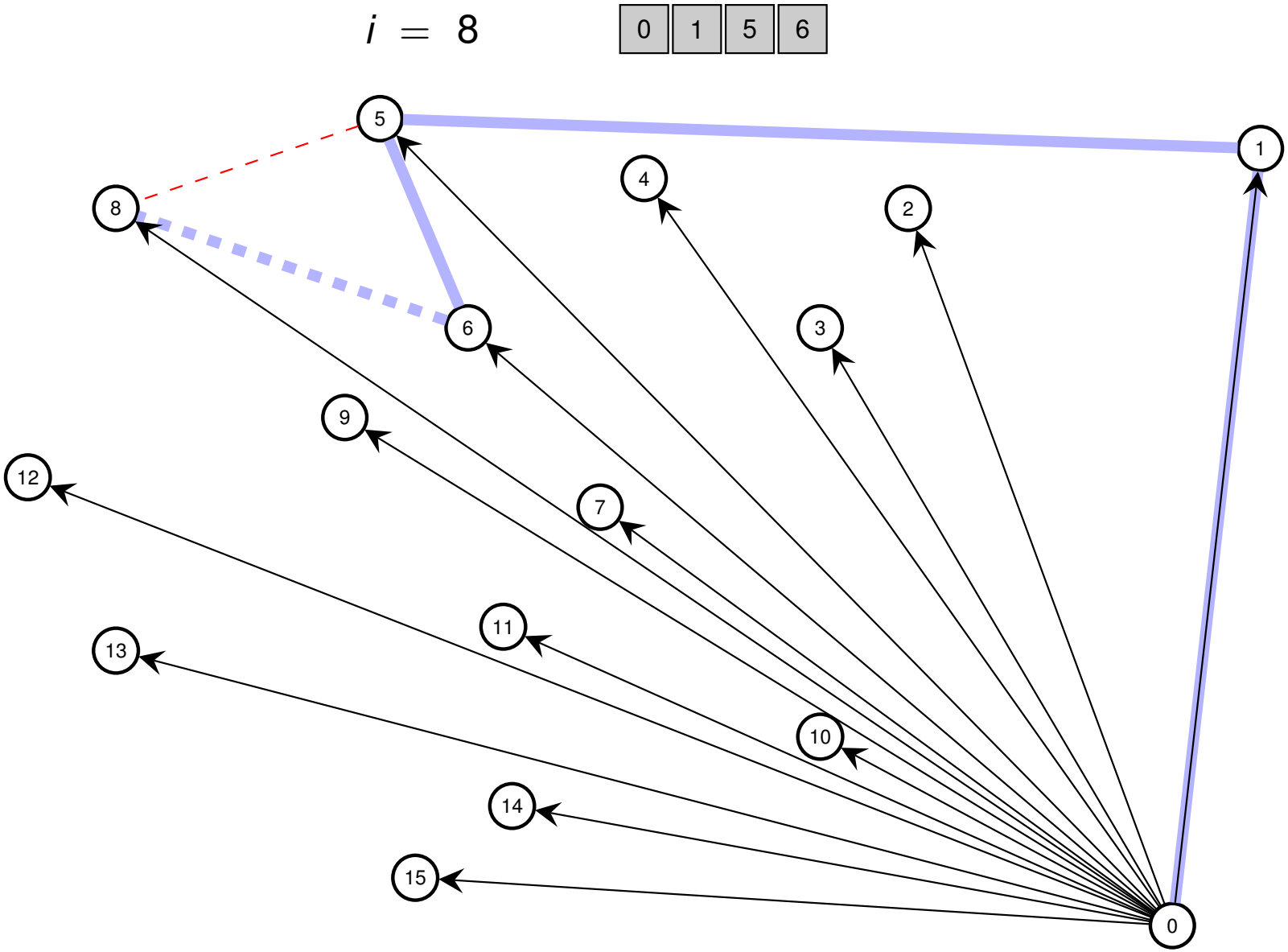
Complete Run of Graham's Scan



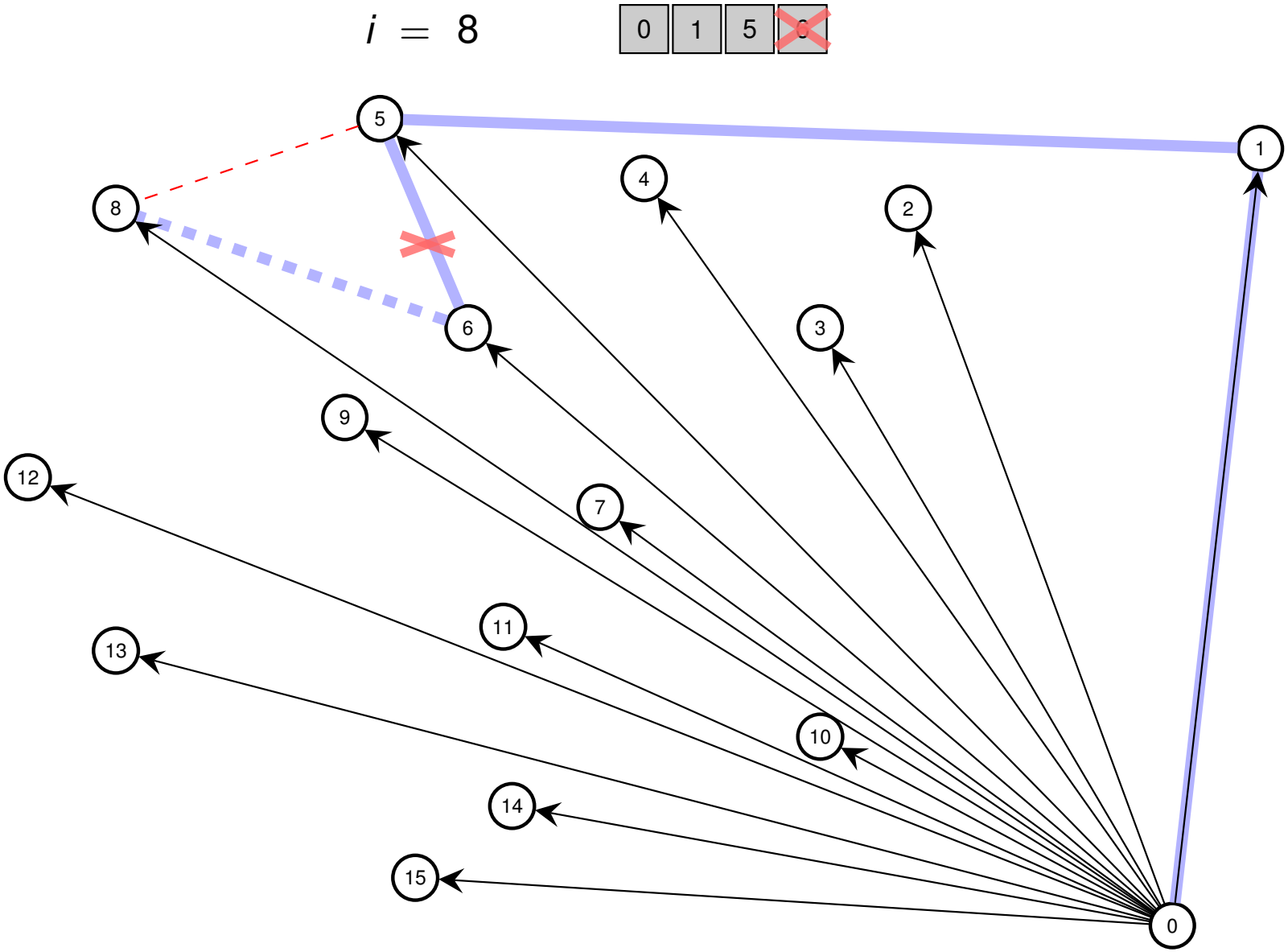
Complete Run of Graham's Scan



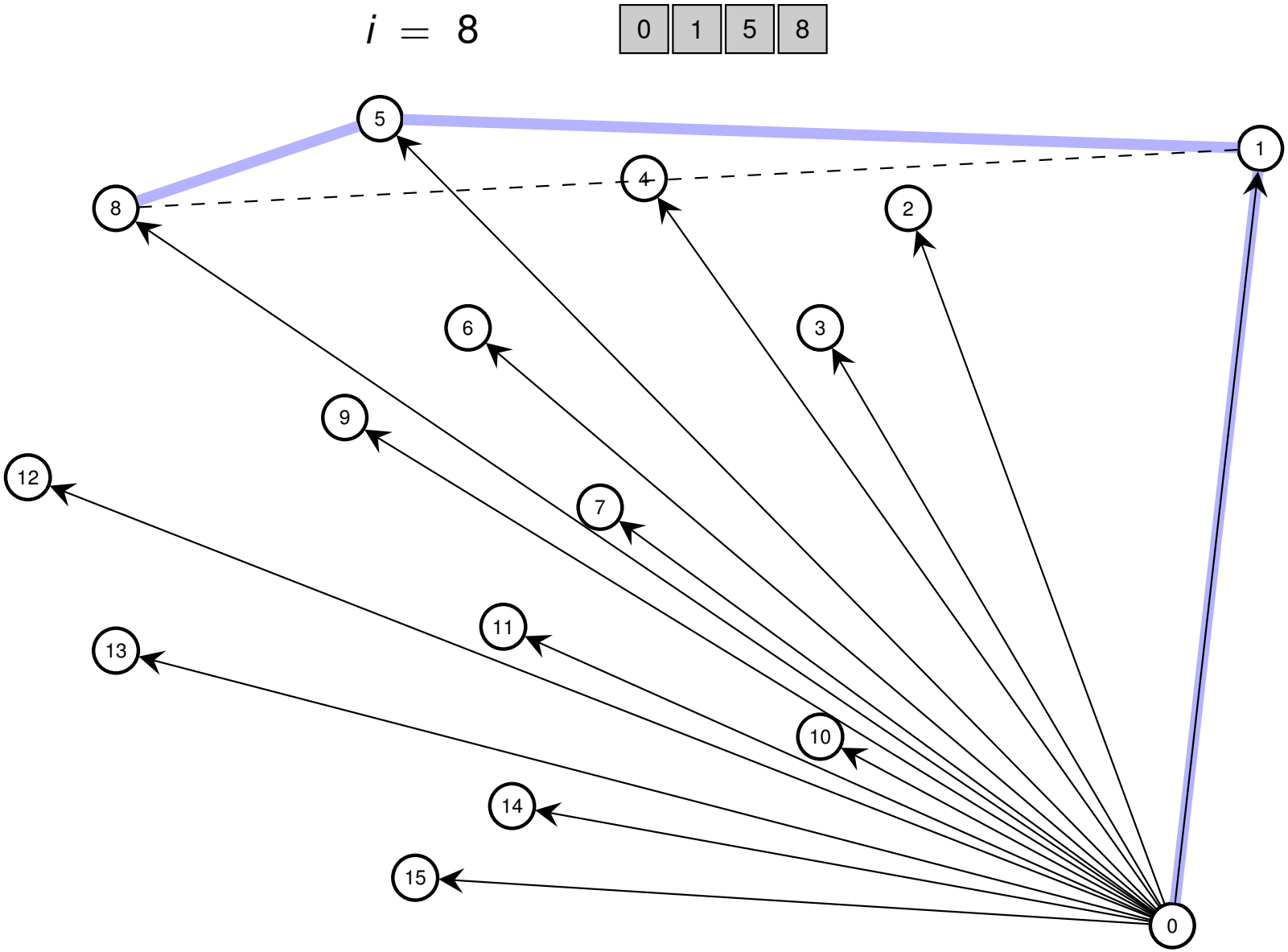
Complete Run of Graham's Scan



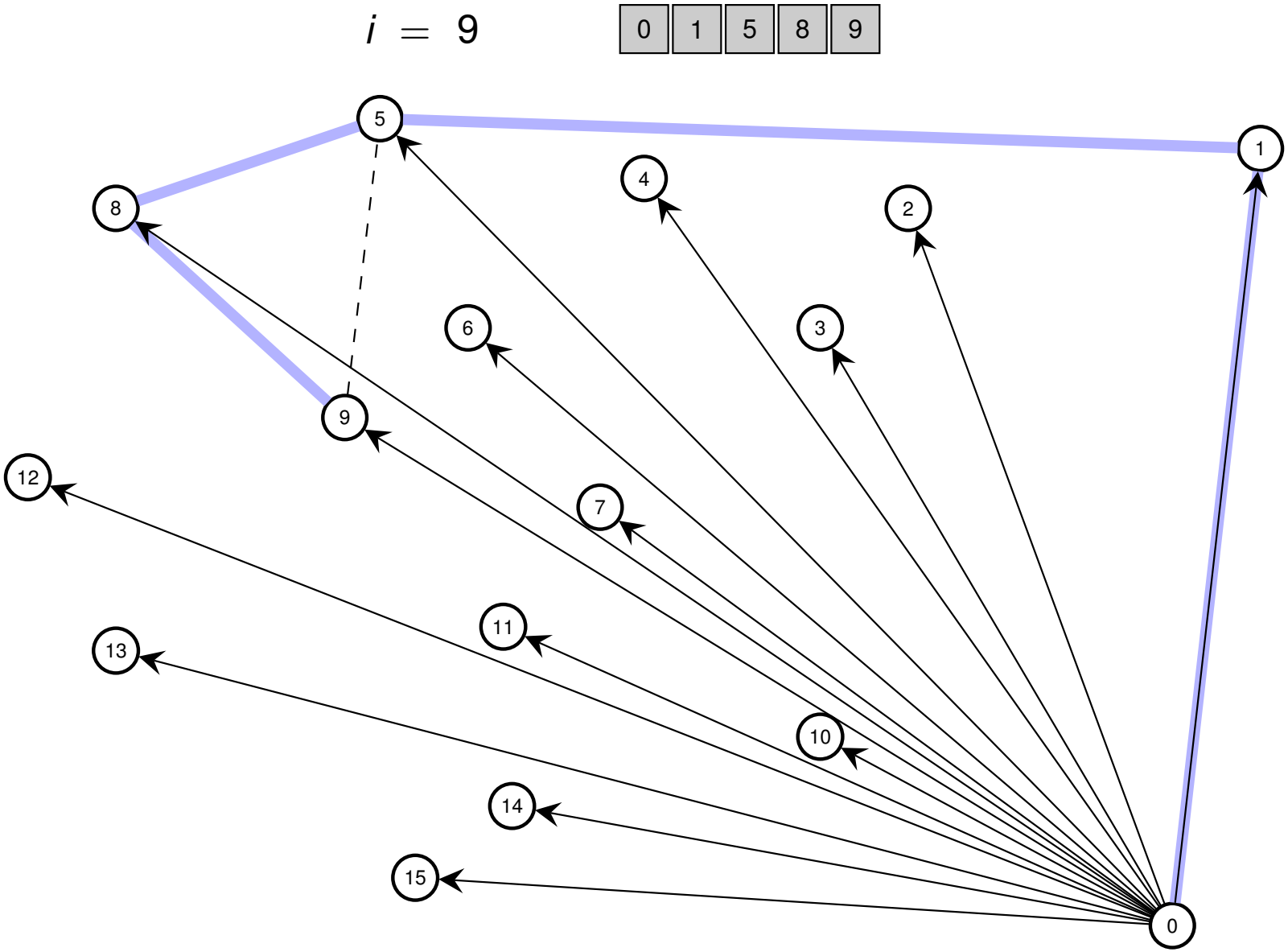
Complete Run of Graham's Scan



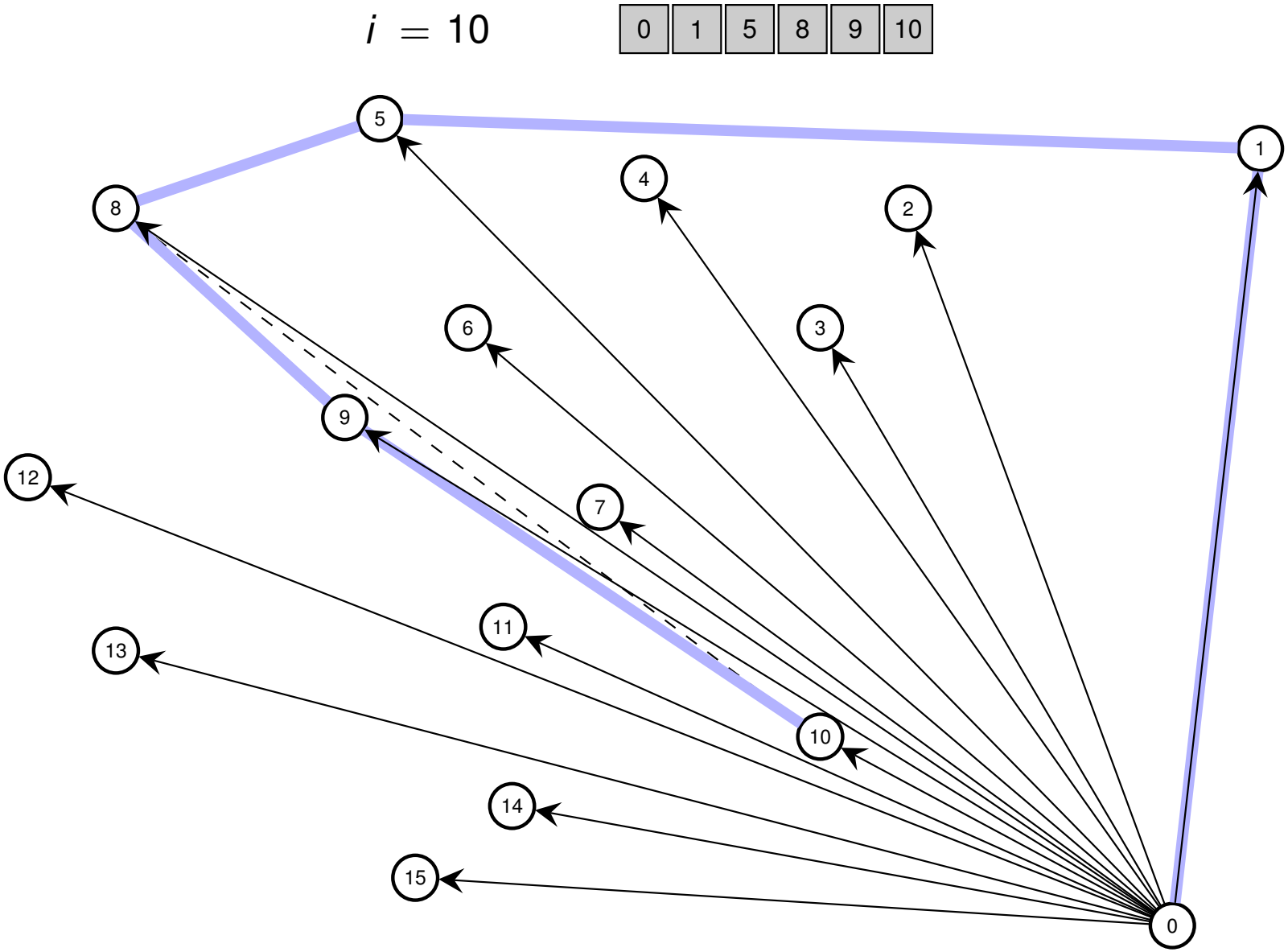
Complete Run of Graham's Scan



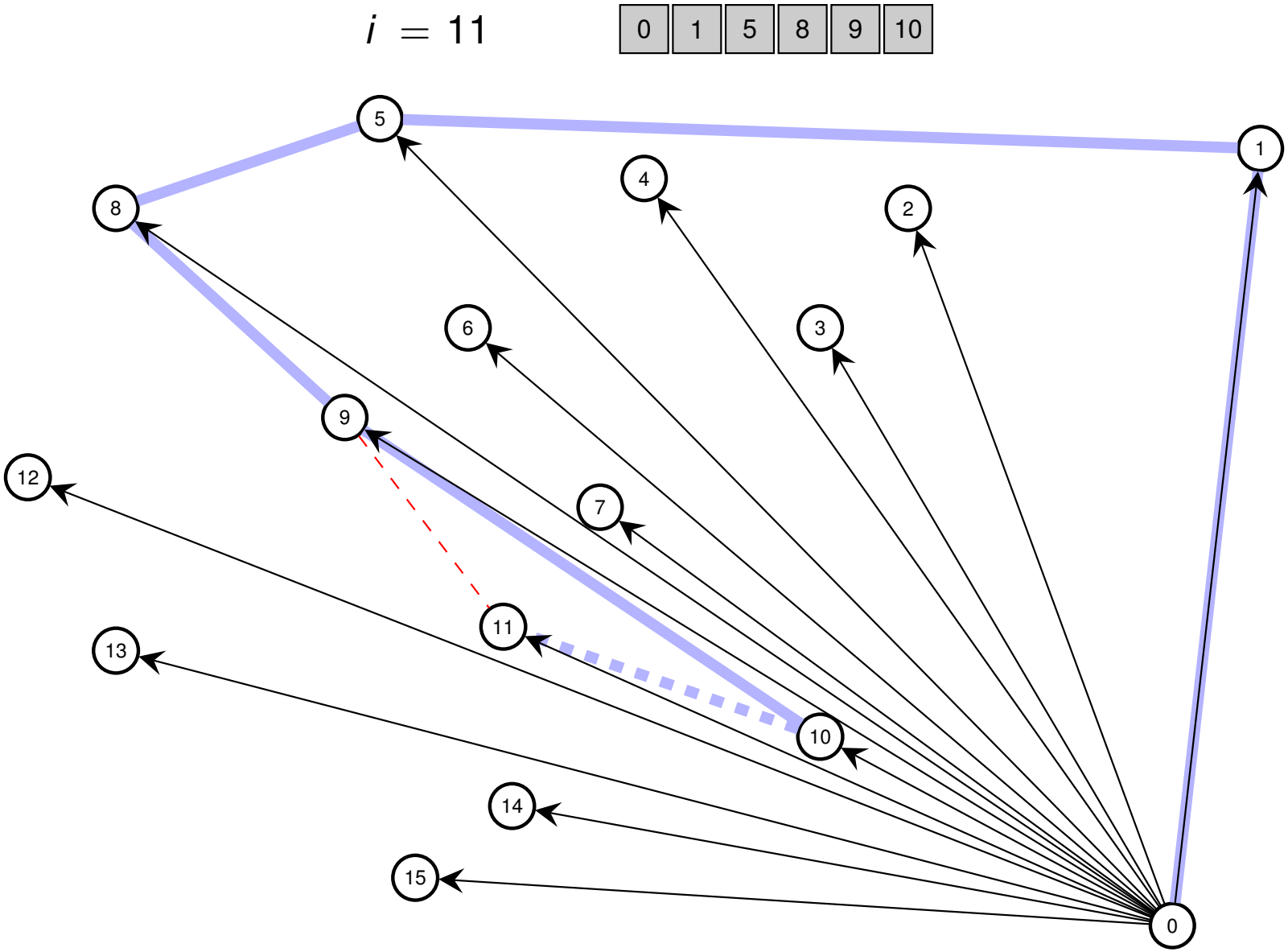
Complete Run of Graham's Scan



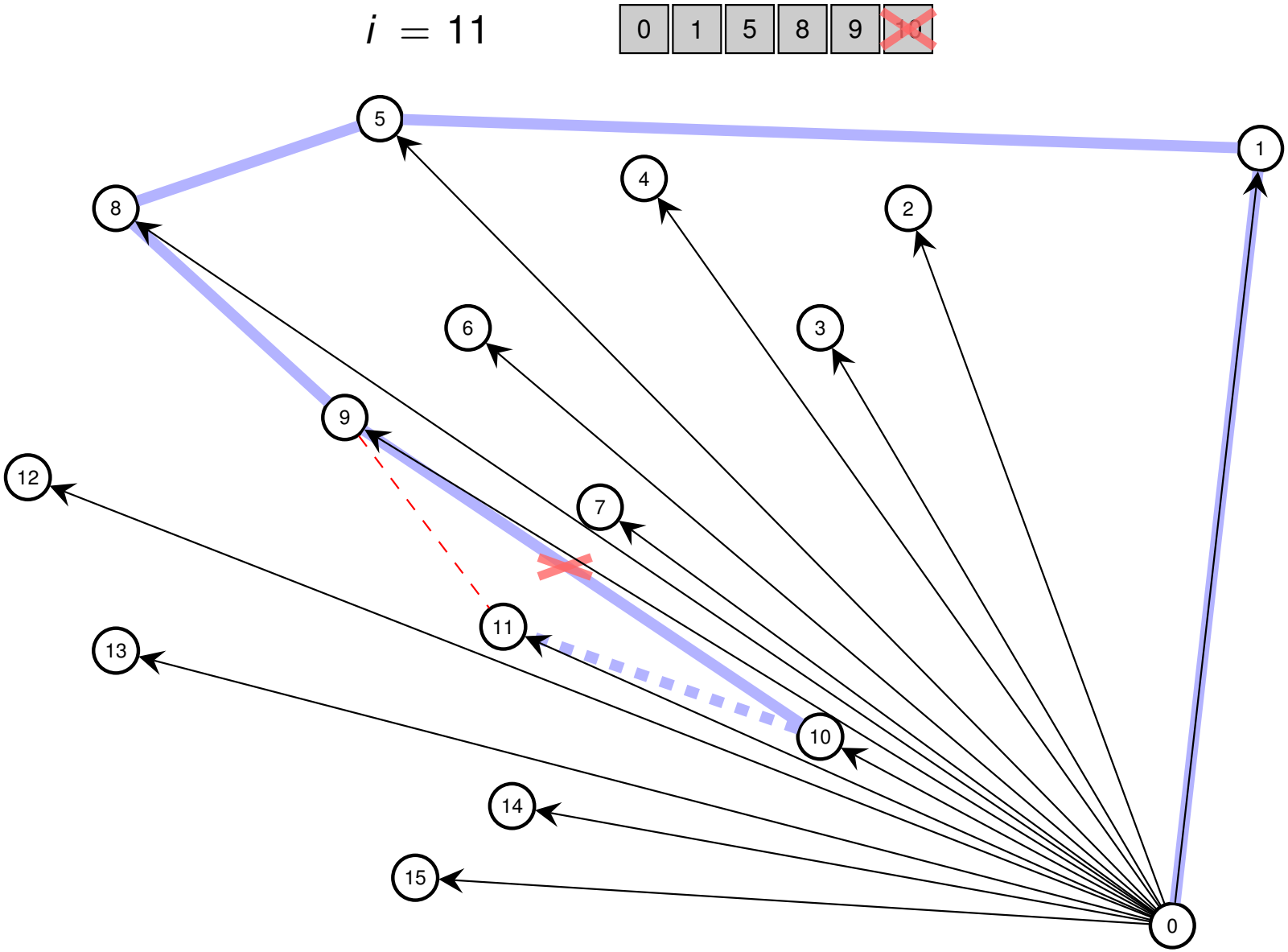
Complete Run of Graham's Scan



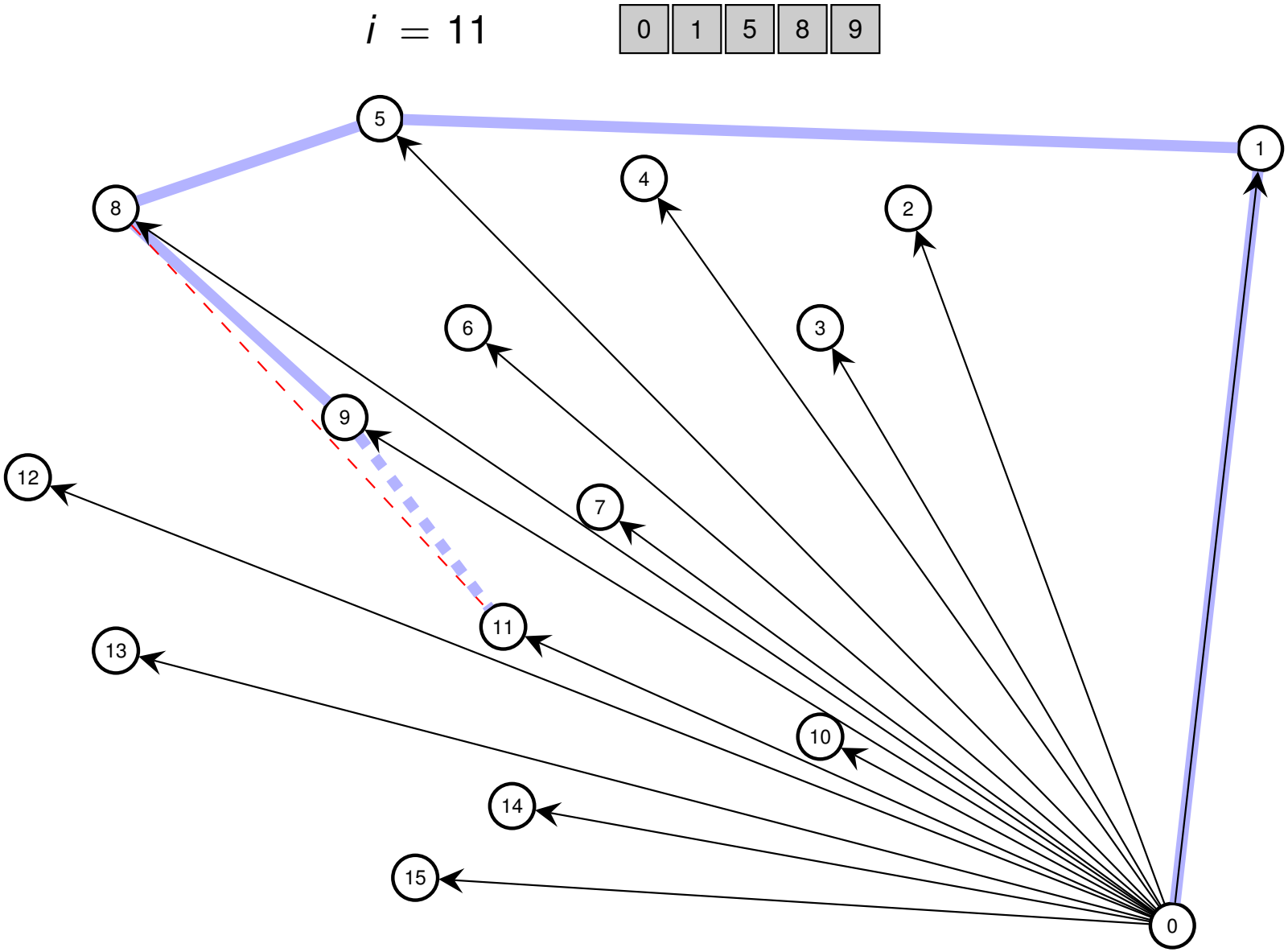
Complete Run of Graham's Scan



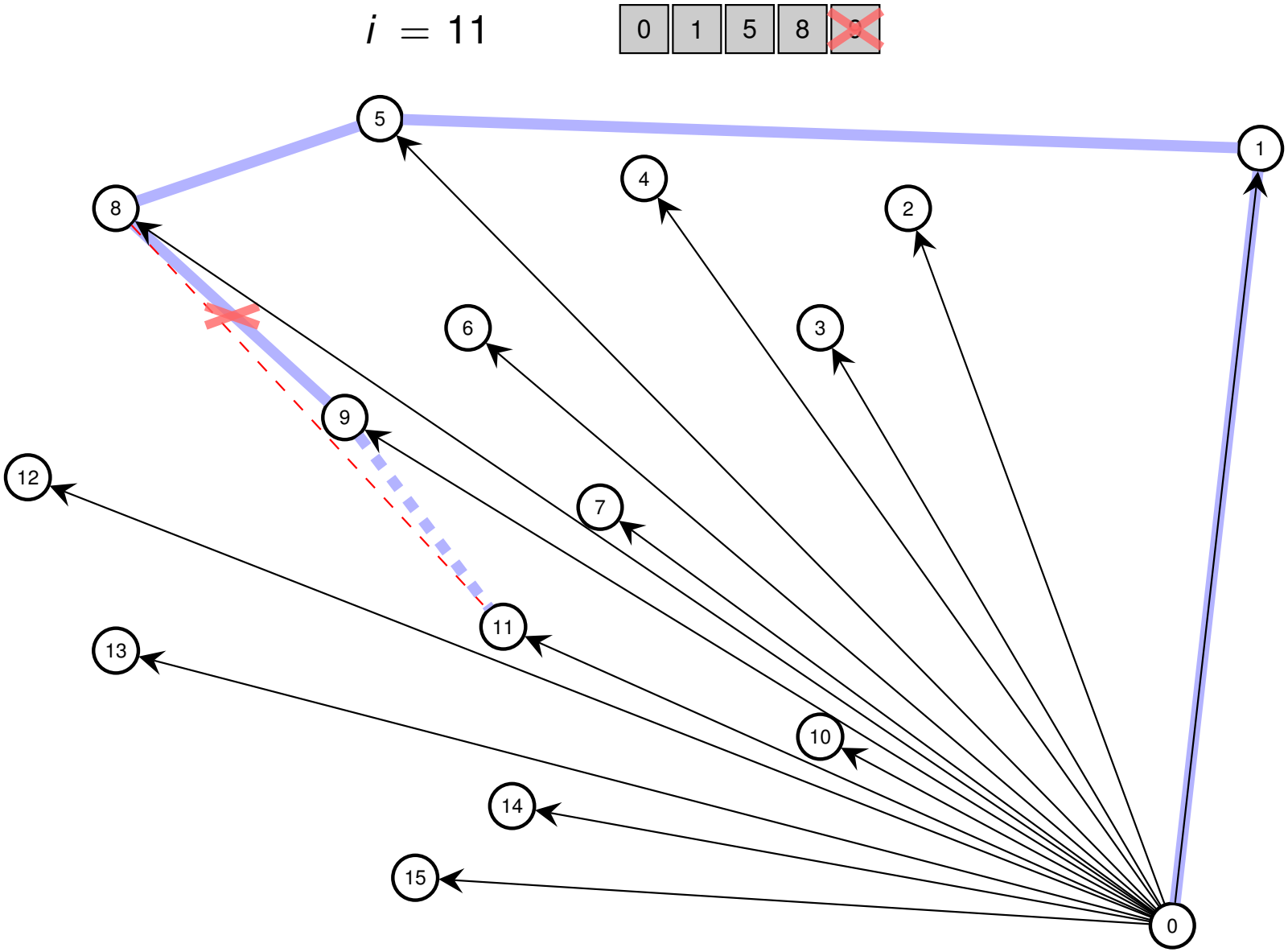
Complete Run of Graham's Scan



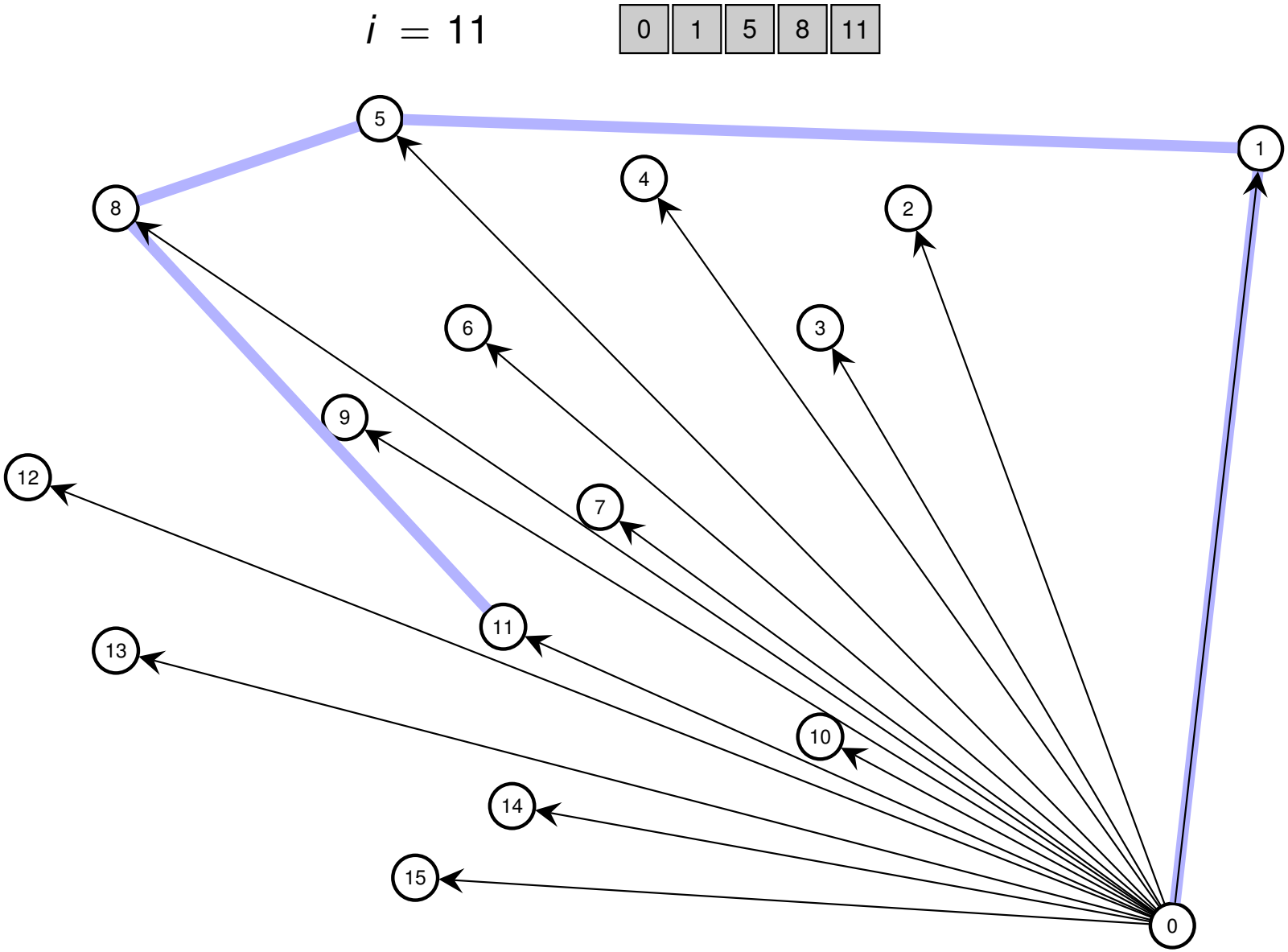
Complete Run of Graham's Scan



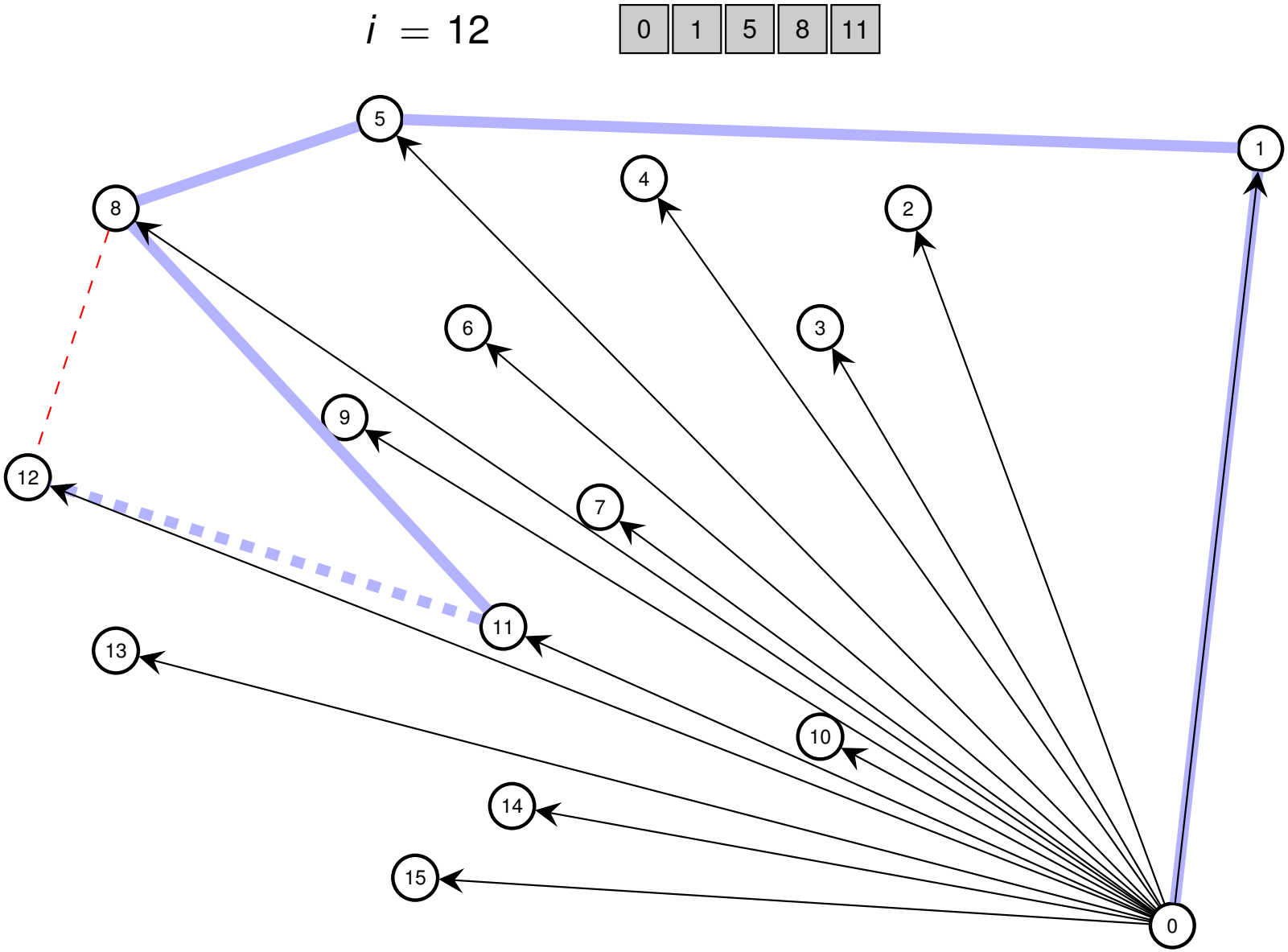
Complete Run of Graham's Scan



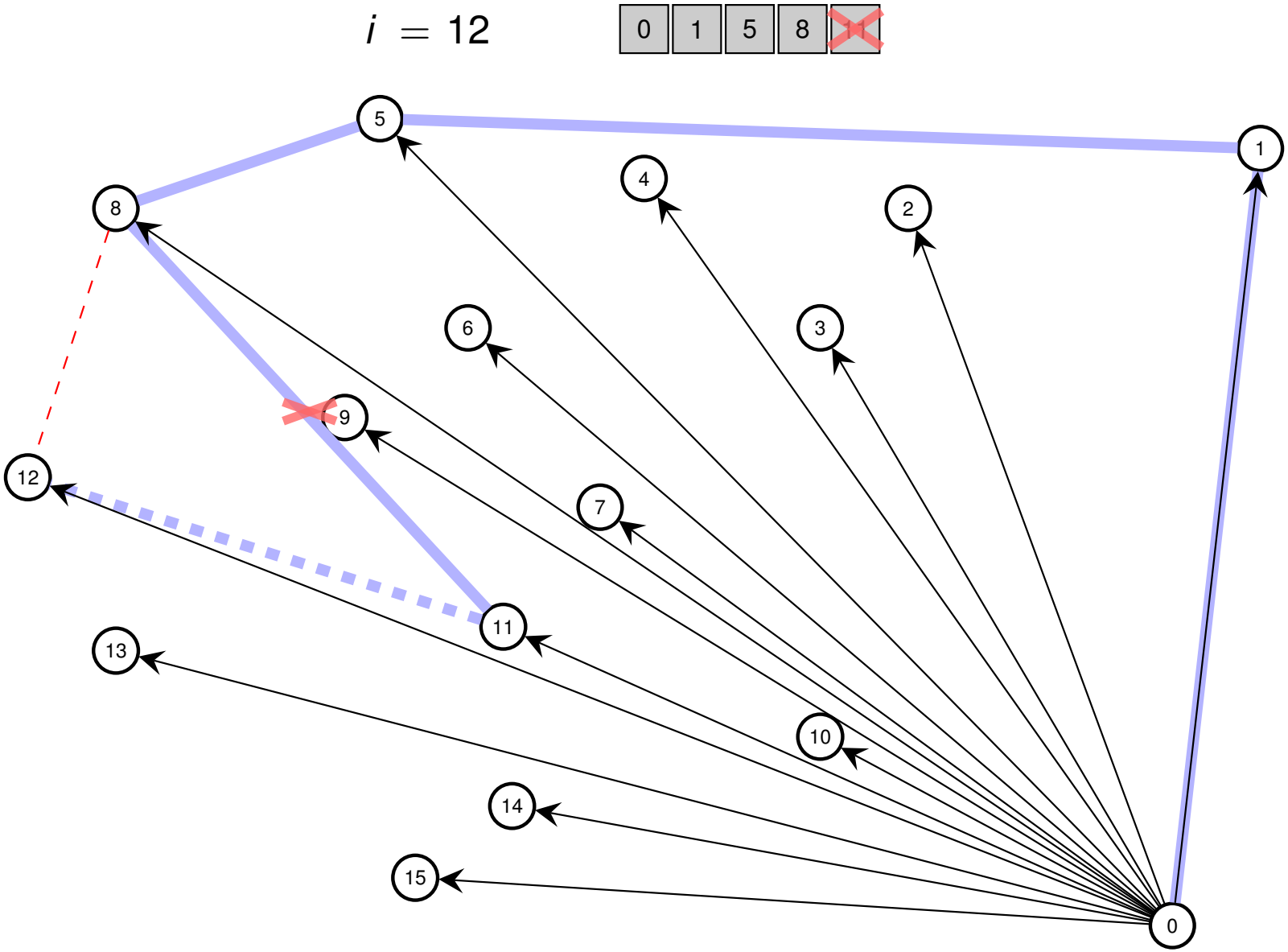
Complete Run of Graham's Scan



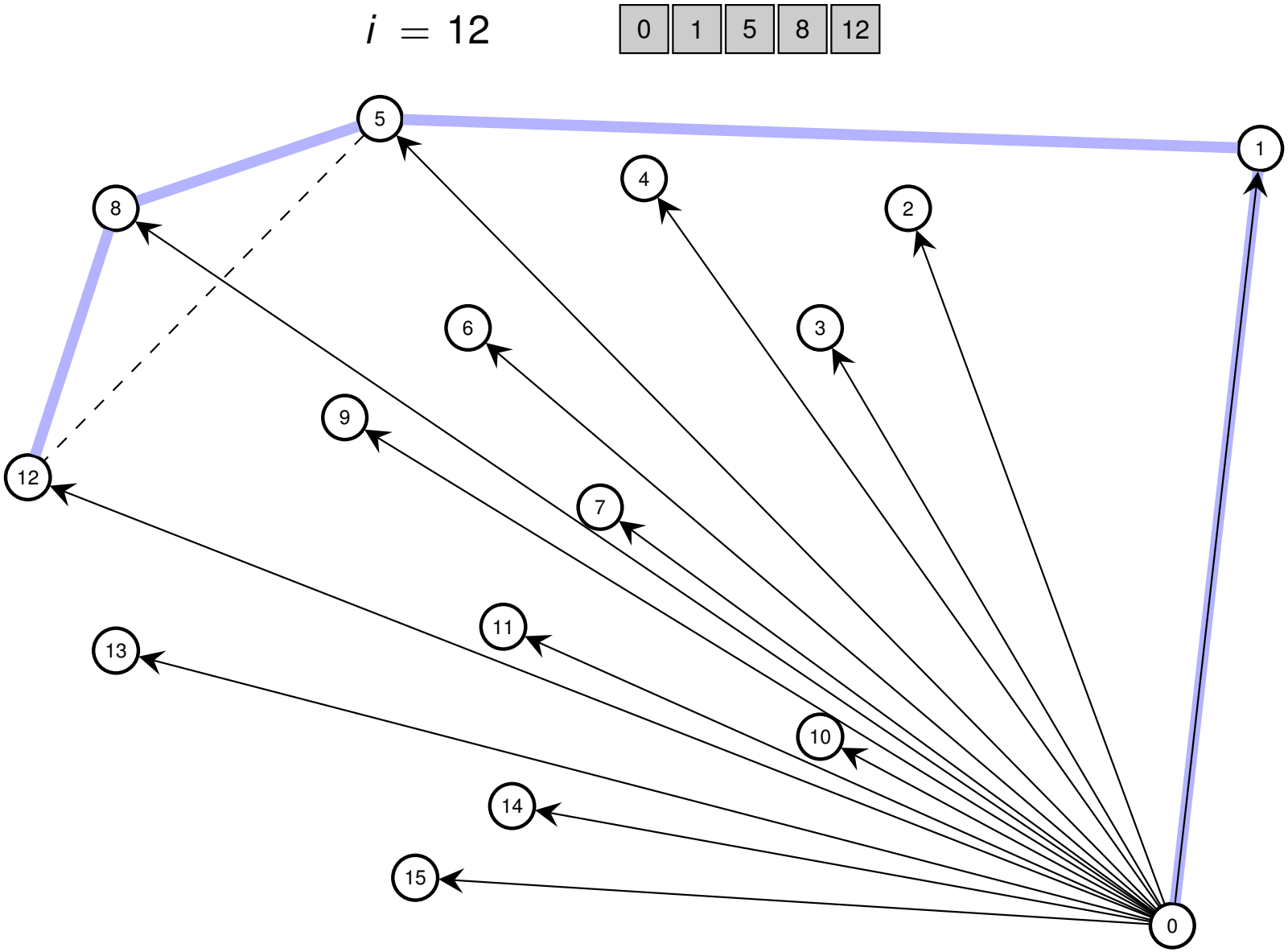
Complete Run of Graham's Scan



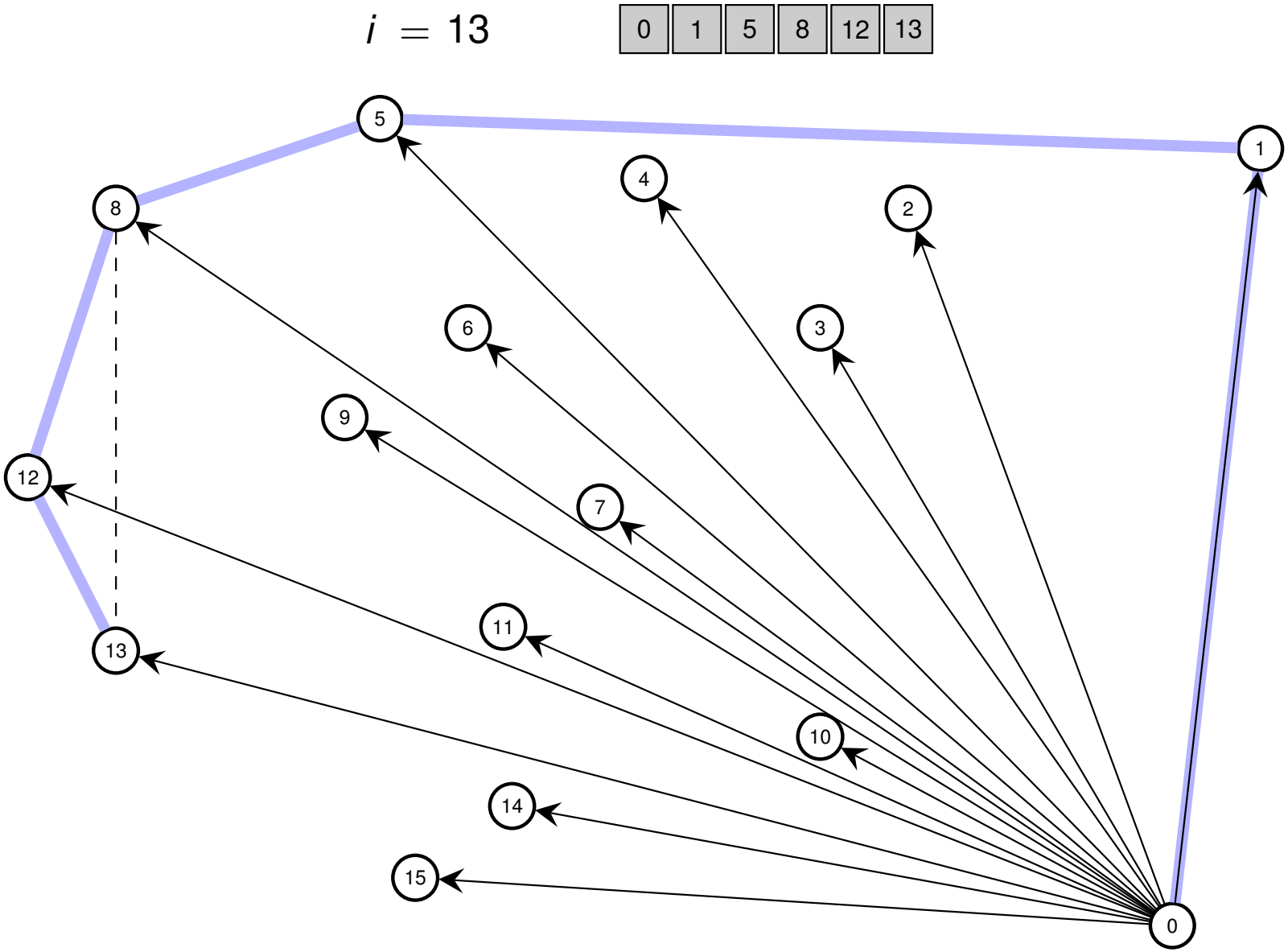
Complete Run of Graham's Scan



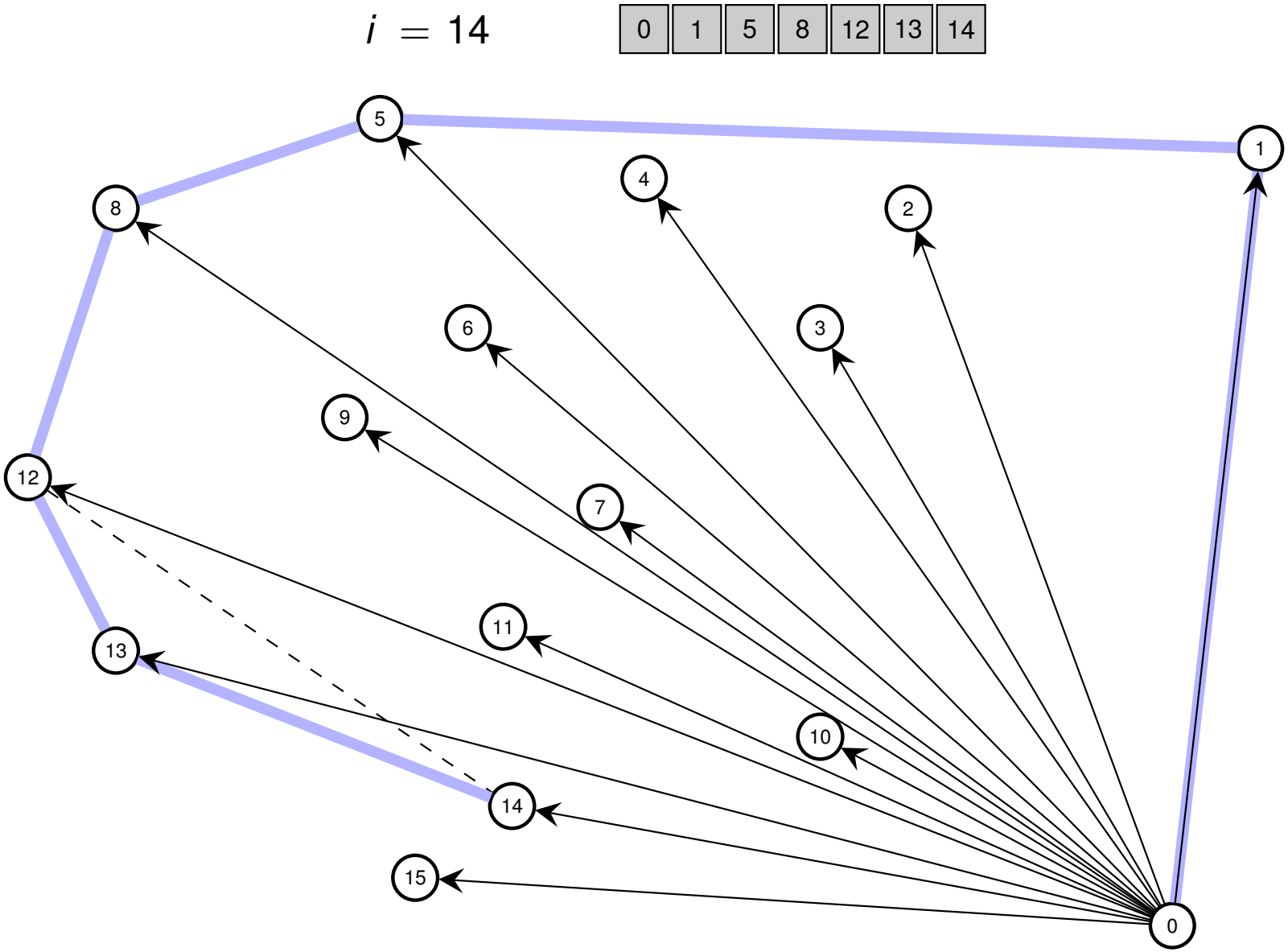
Complete Run of Graham's Scan



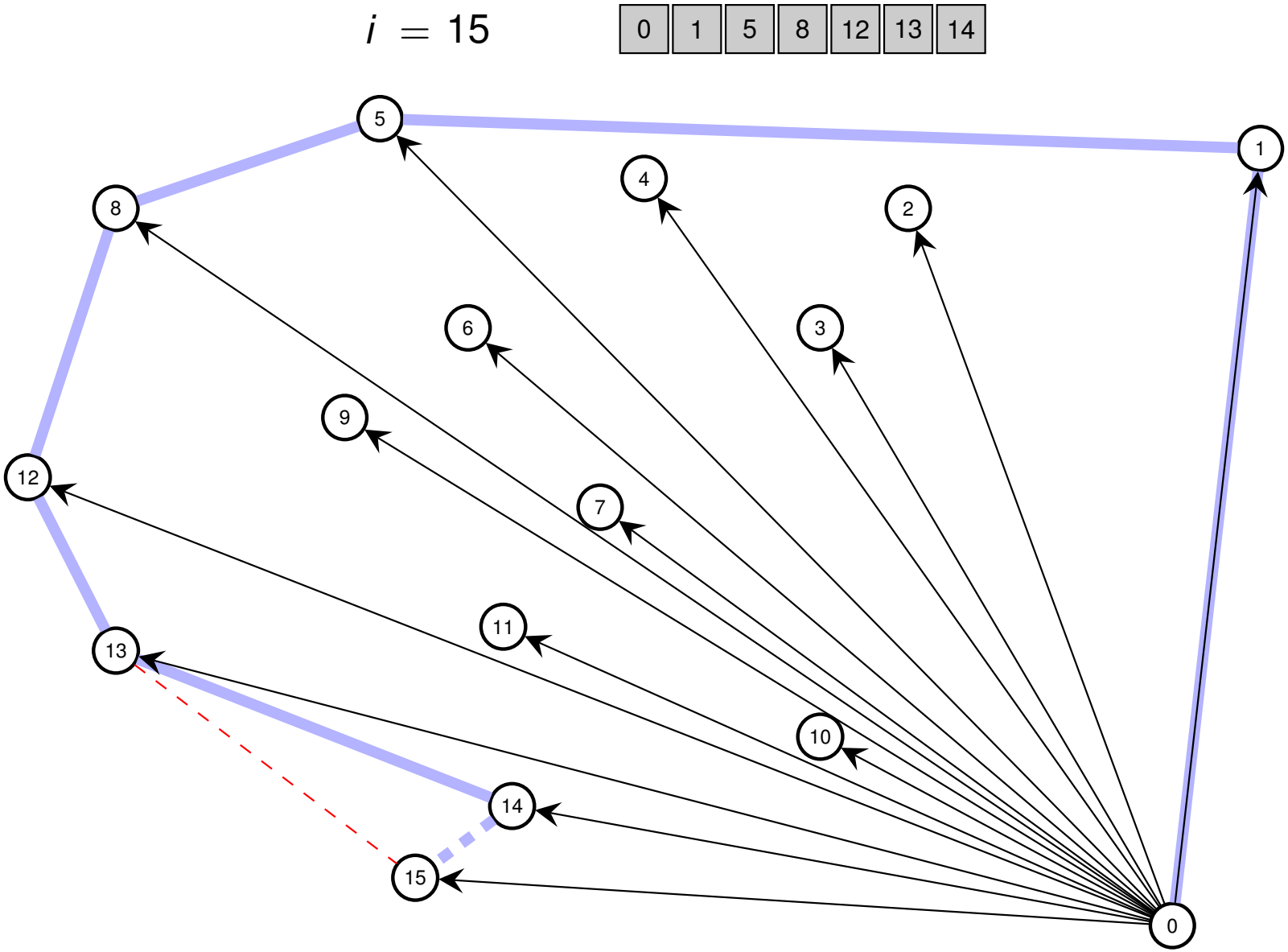
Complete Run of Graham's Scan



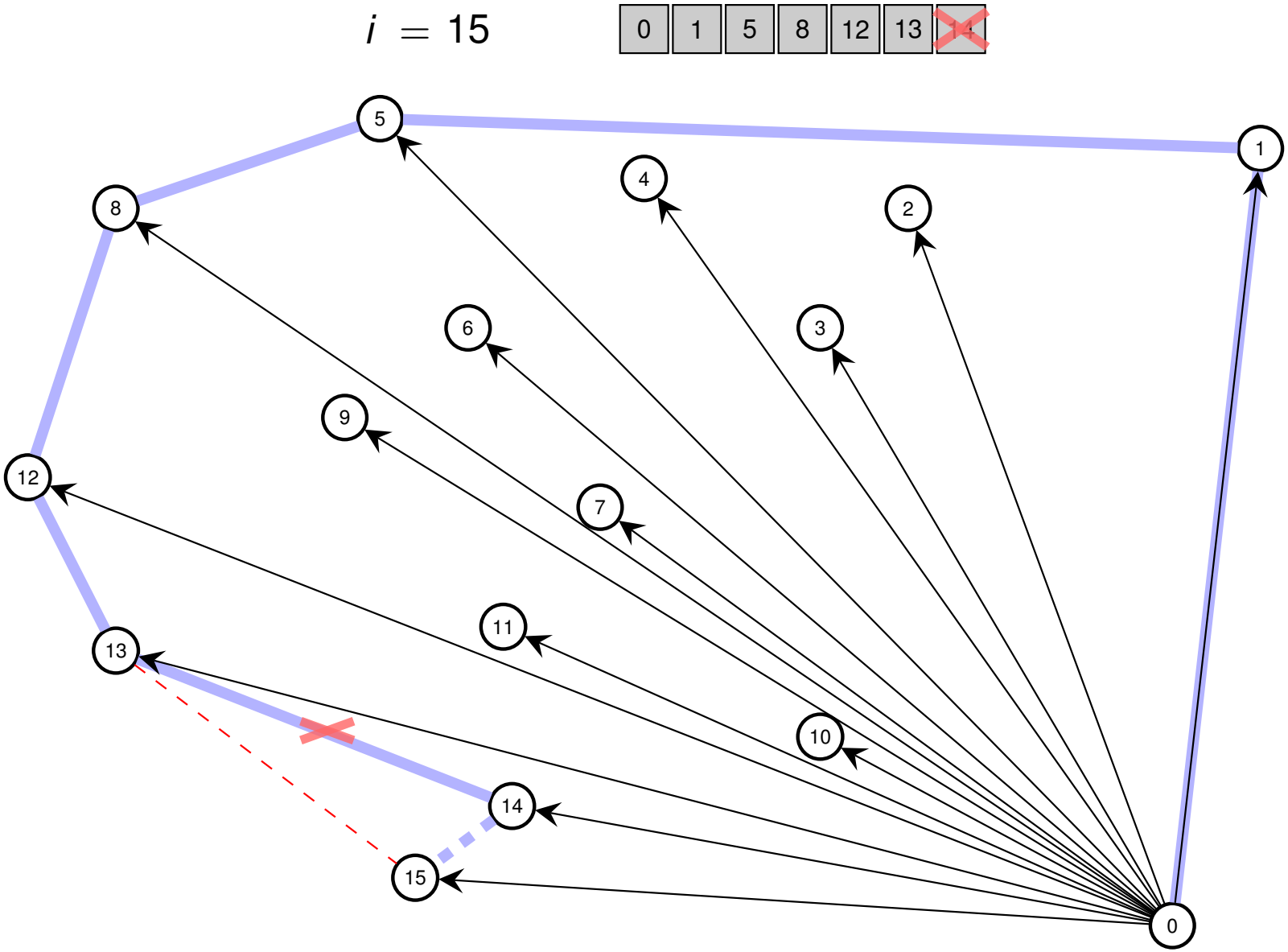
Complete Run of Graham's Scan



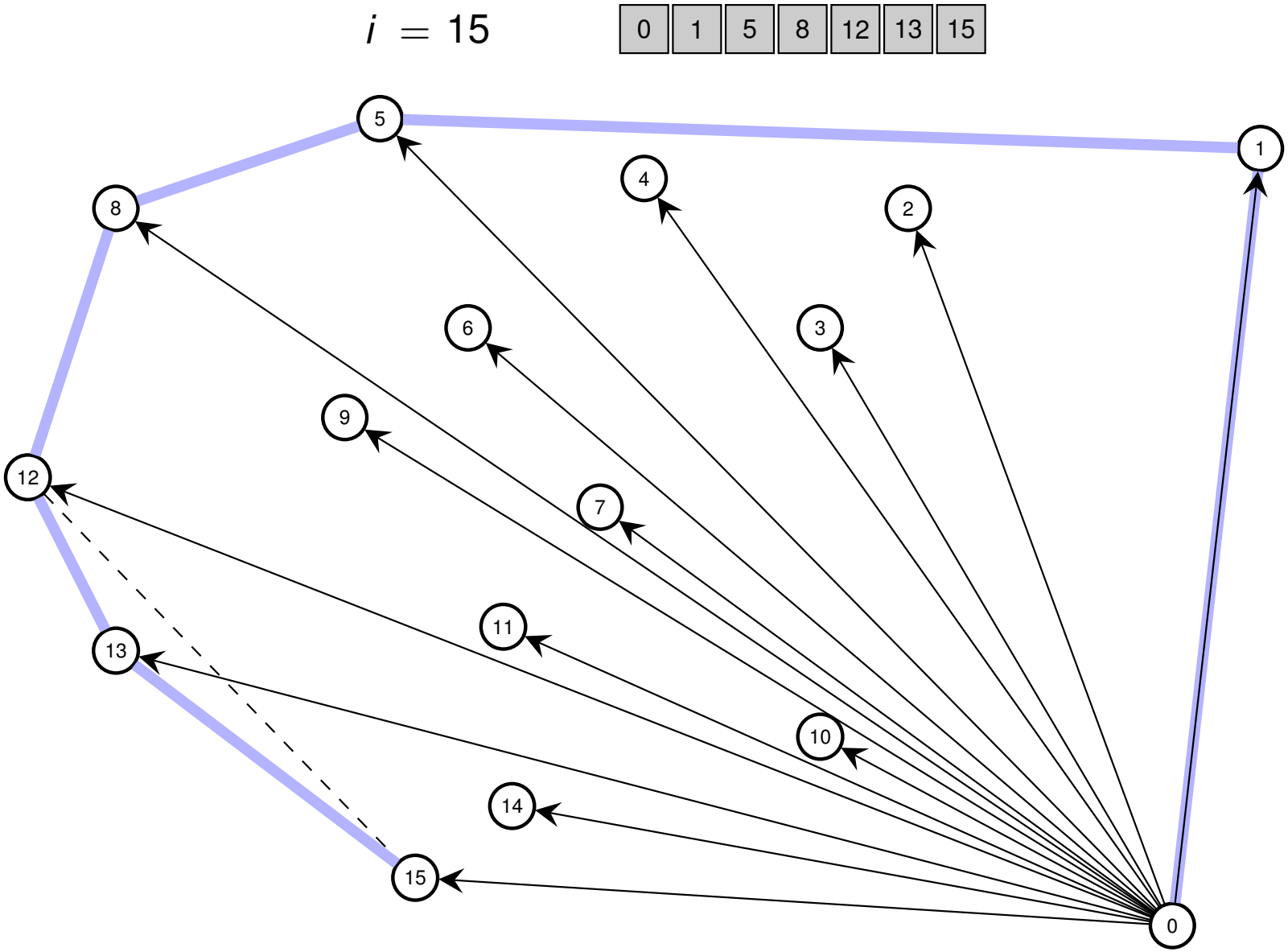
Complete Run of Graham's Scan



Complete Run of Graham's Scan



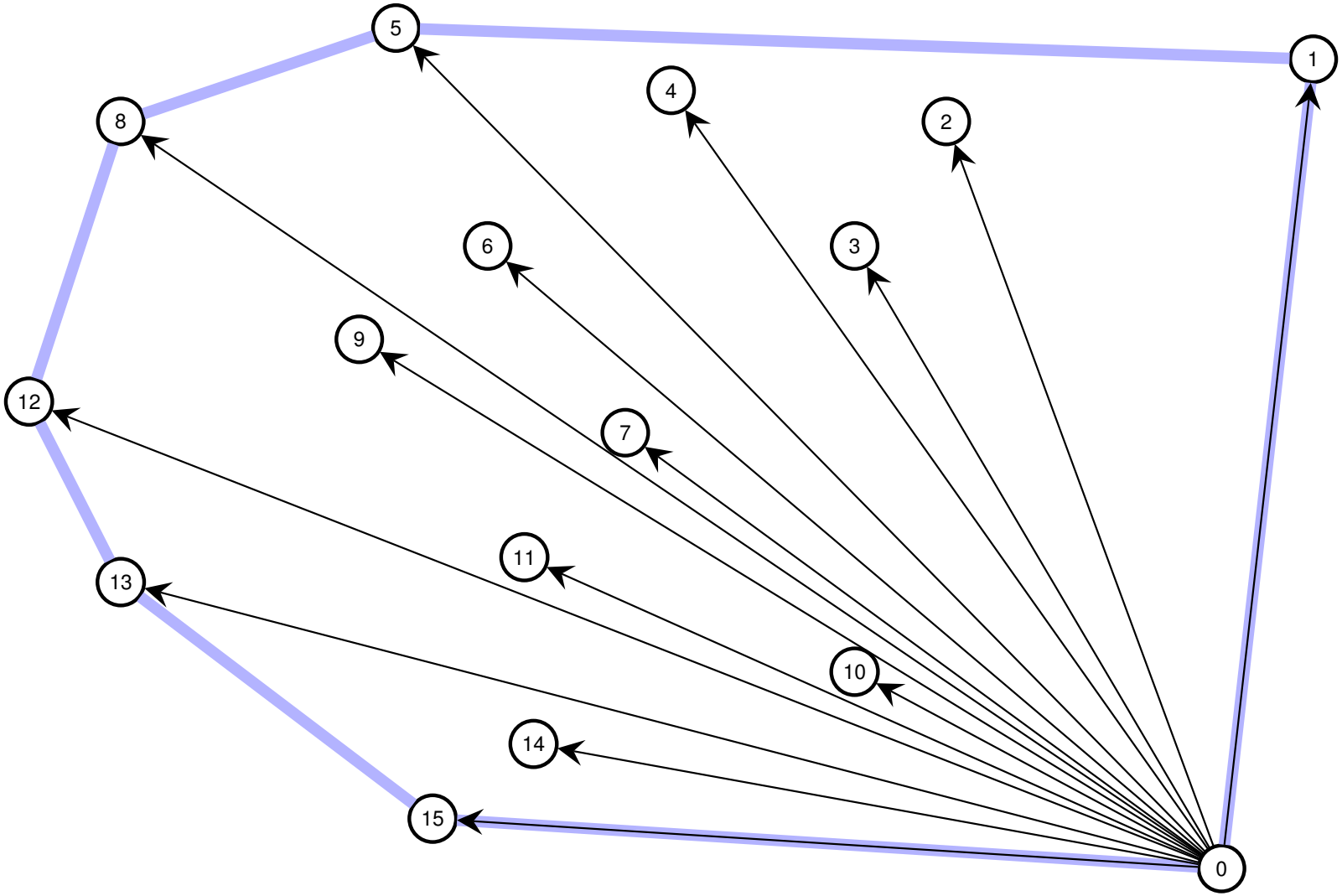
Complete Run of Graham's Scan



Complete Run of Graham's Scan

$i = 15$ ✓

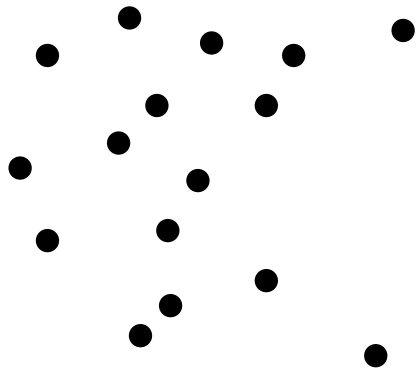
0	1	5	8	12	13	15



Jarvis' March (Gift wrapping)

Intuition

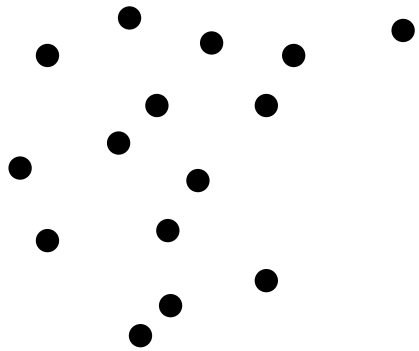
- Wrapping taut paper around the points



Jarvis' March (Gift wrapping)

Intuition

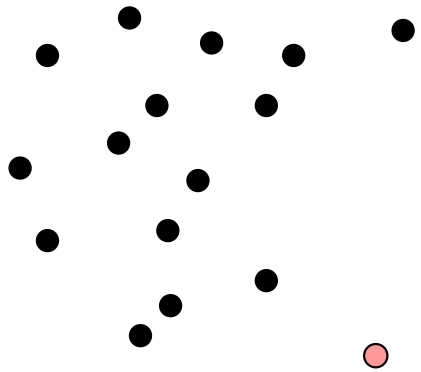
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point



Jarvis' March (Gift wrapping)

Intuition

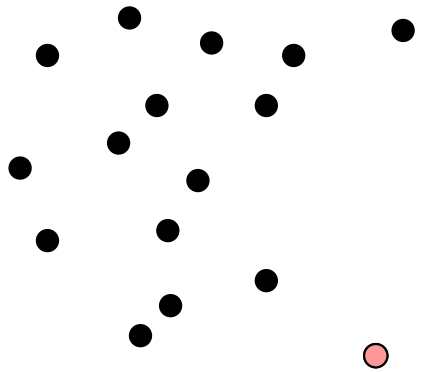
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point



Jarvis' March (Gift wrapping)

Intuition

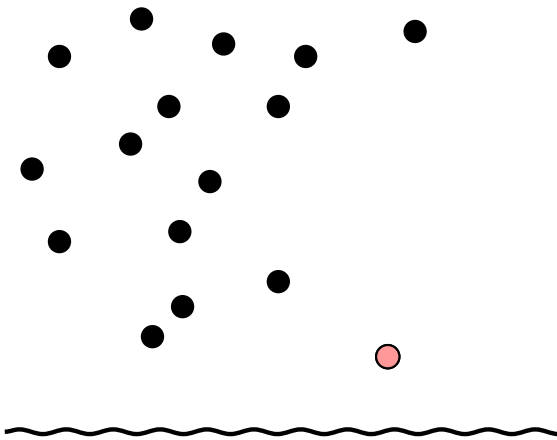
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point



Jarvis' March (Gift wrapping)

Intuition

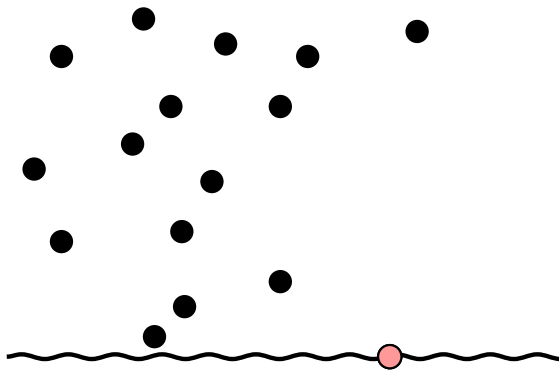
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point



Jarvis' March (Gift wrapping)

Intuition

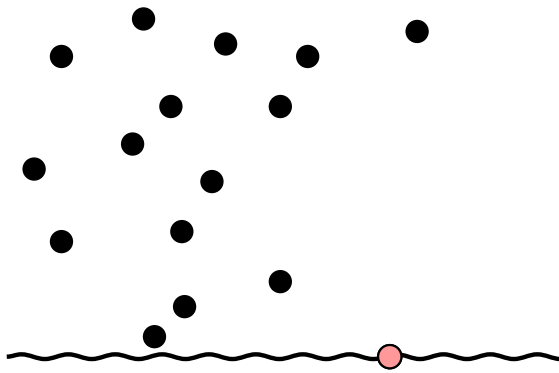
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point



Jarvis' March (Gift wrapping)

Intuition

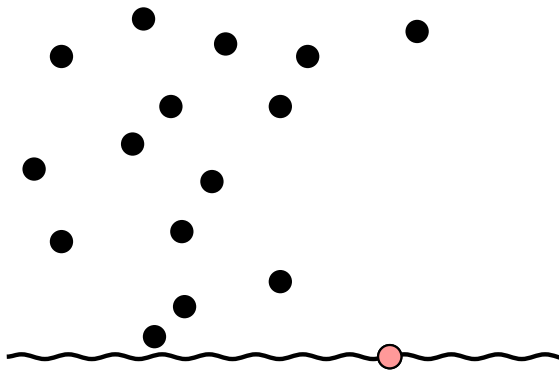
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

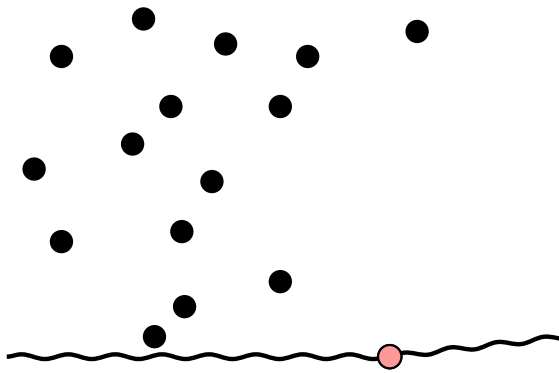
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

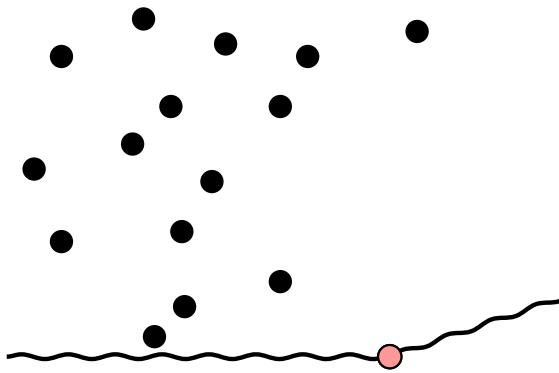
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

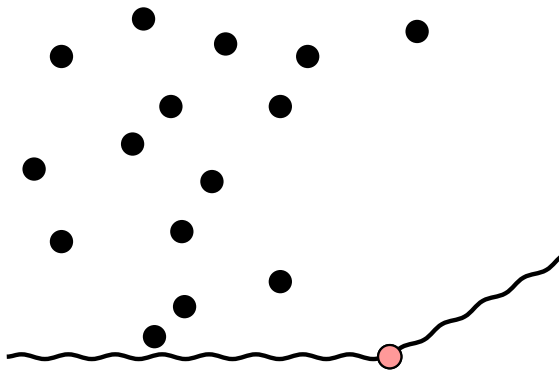
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

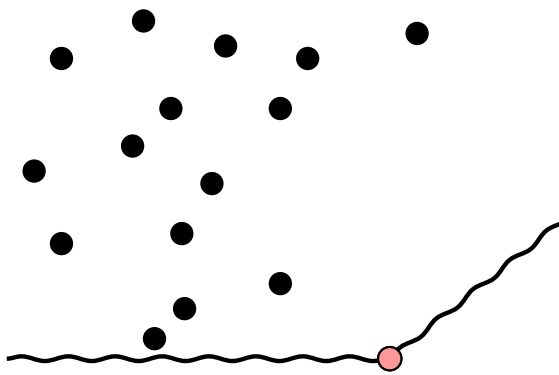
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

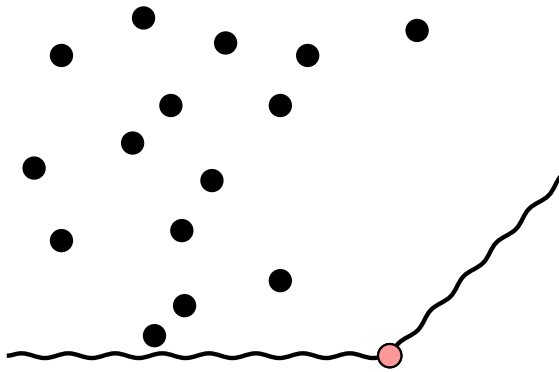
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

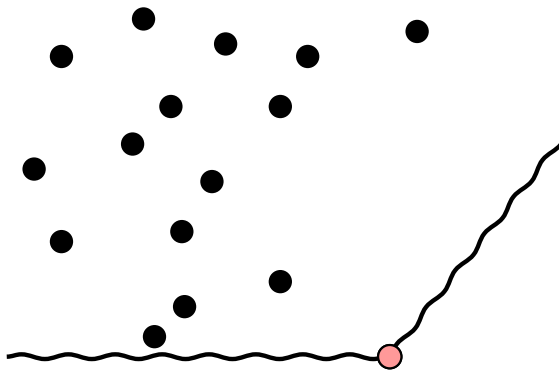
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

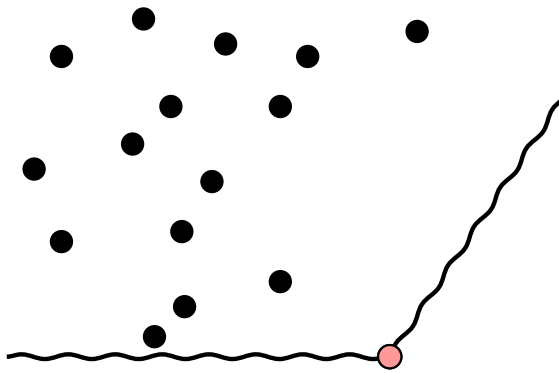
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

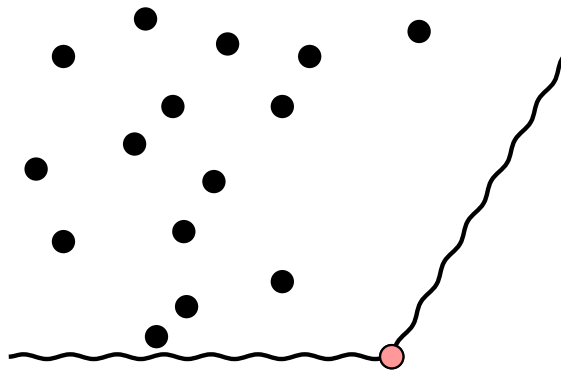
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

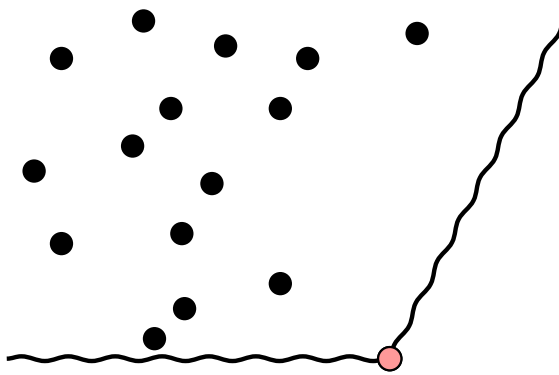
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

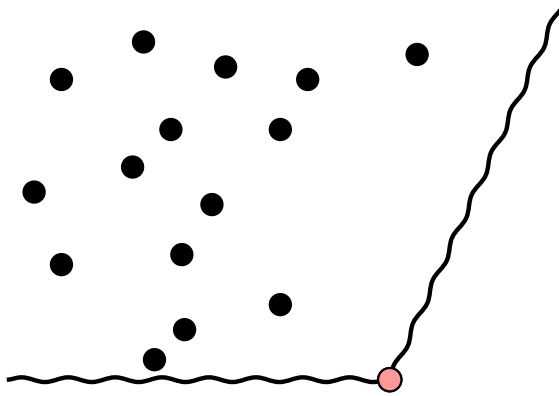
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

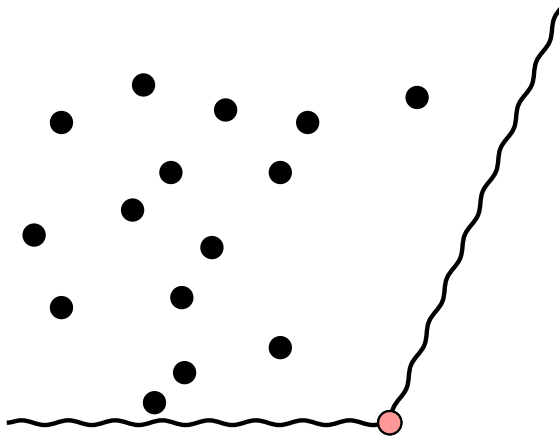
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

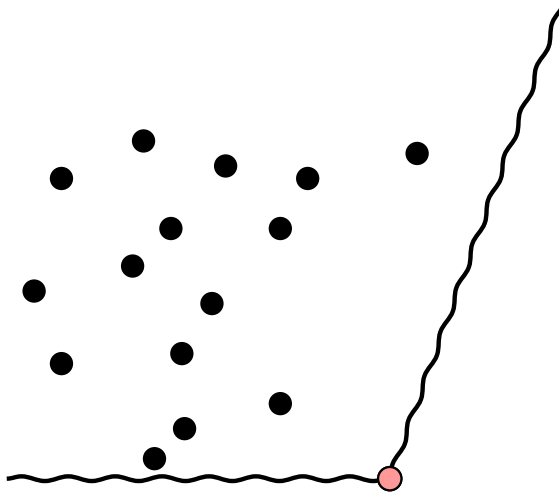
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

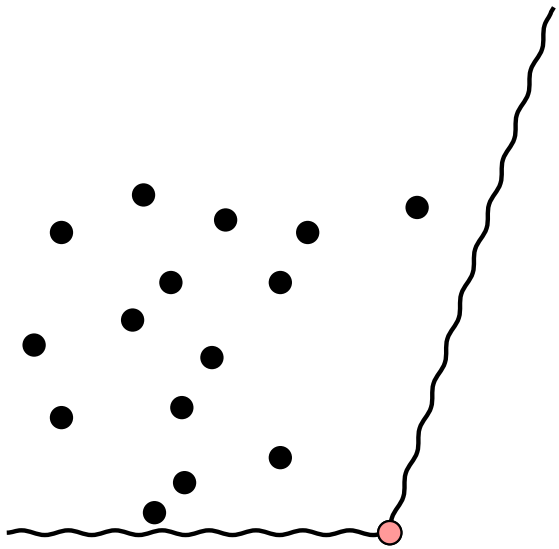
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

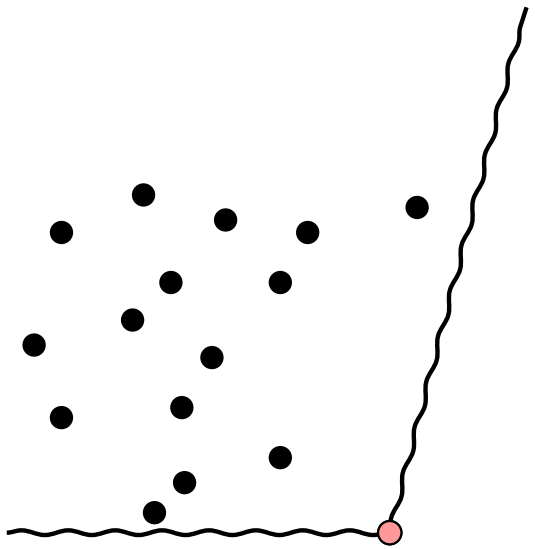
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

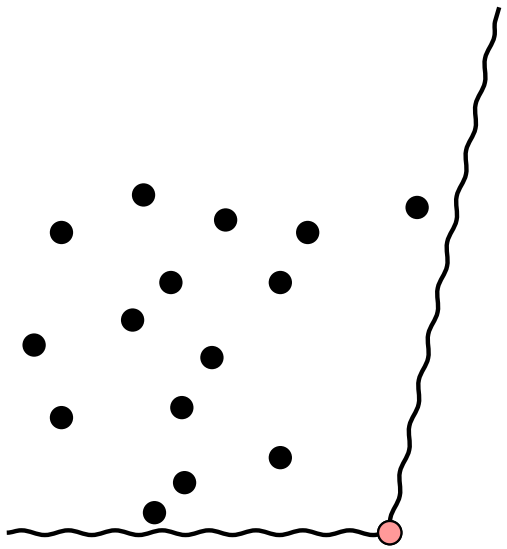
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

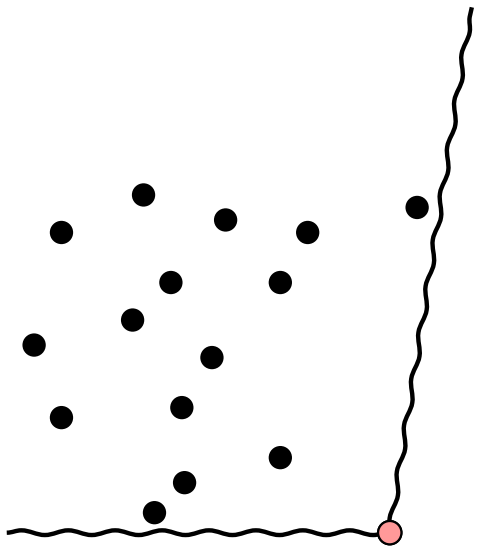
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

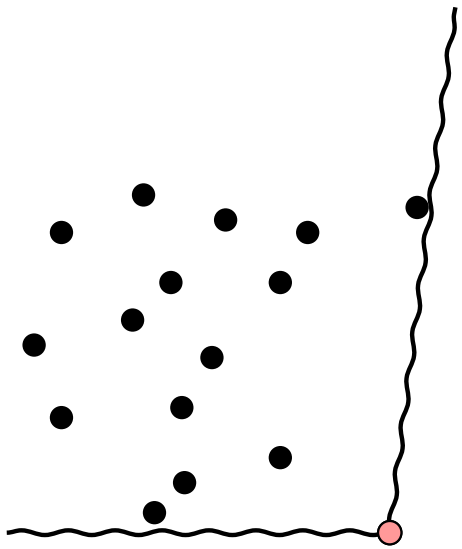
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

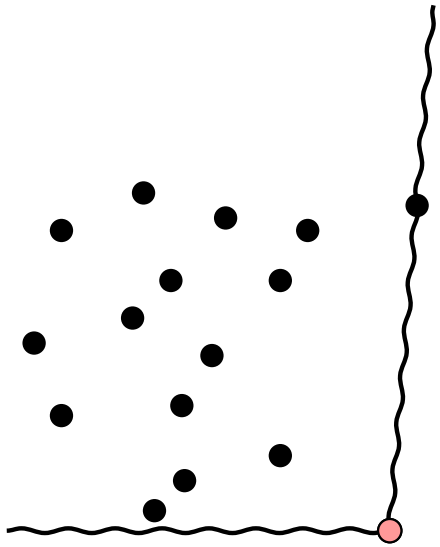
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

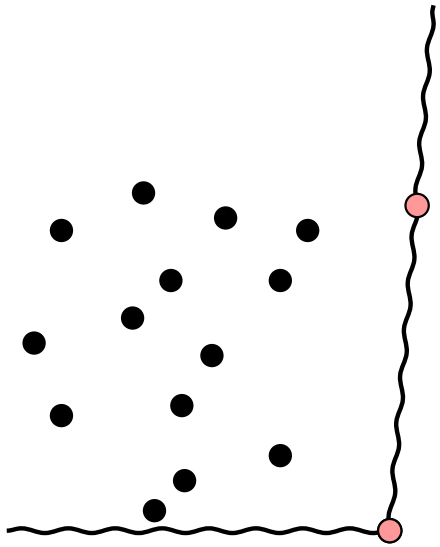
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point



Jarvis' March (Gift wrapping)

Intuition

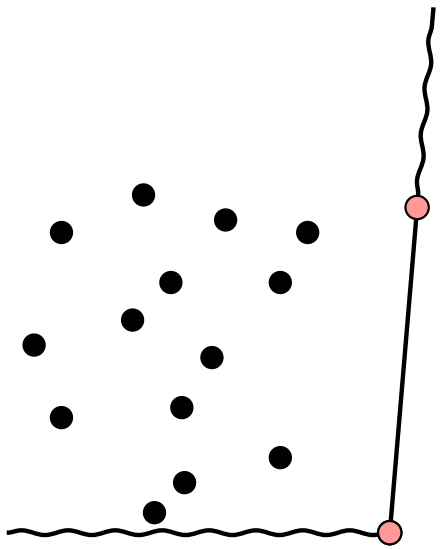
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point
  3. Tape paper and go to 2



Jarvis' March (Gift wrapping)

Intuition

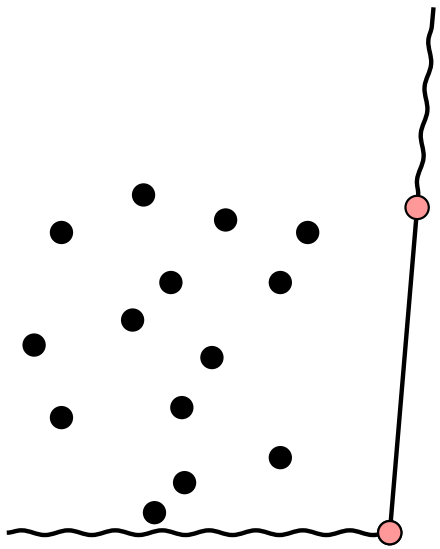
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point
  3. Tape paper and go to 2



Jarvis' March (Gift wrapping)

Intuition

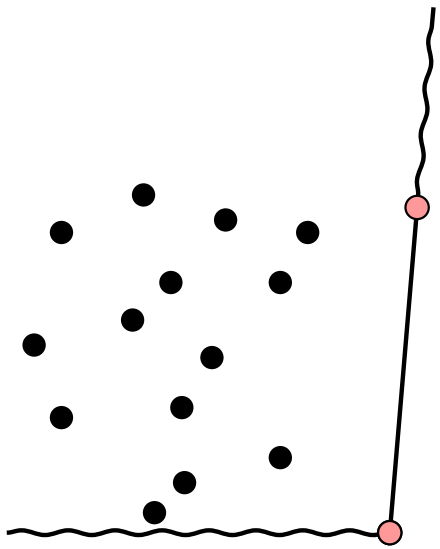
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point
  3. Tape paper and go to 2



Jarvis' March (Gift wrapping)

Intuition

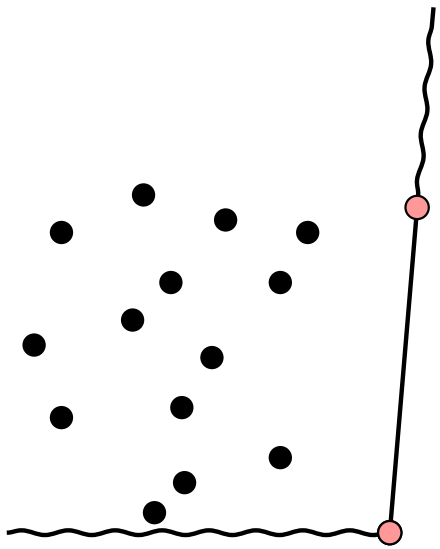
- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point
  3. Tape paper and go to 2



Jarvis' March (Gift wrapping)

Intuition

- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point
  3. Tape paper and go to 2



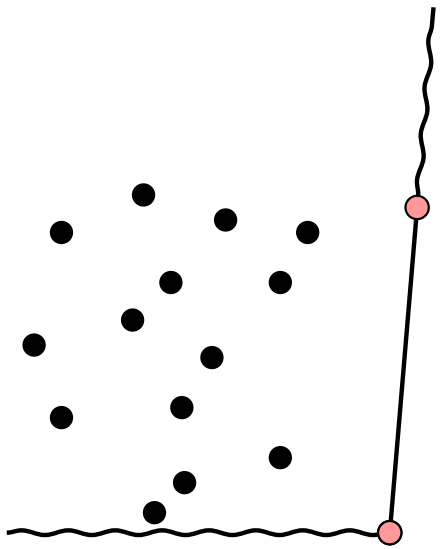
Algorithm

1. Let p_0 be the lowest point
2. Next point the one with smallest angle w.r.t. p_0

Jarvis' March (Gift wrapping)

Intuition

- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point
  3. Tape paper and go to 2



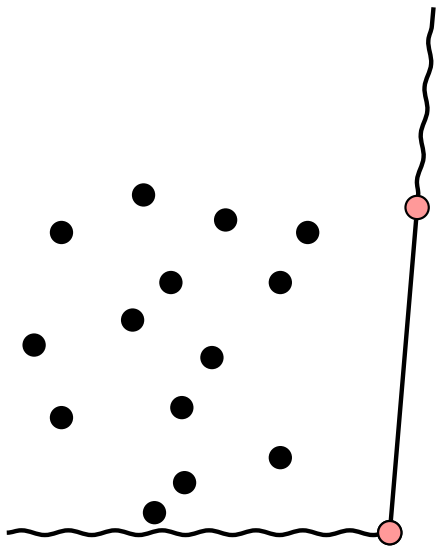
Algorithm

1. Let p_0 be the lowest point
2. Next point the one with smallest angle w.r.t. p_0
3. Continue until highest point p_k

Jarvis' March (Gift wrapping)

Intuition

- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point
  3. Tape paper and go to 2



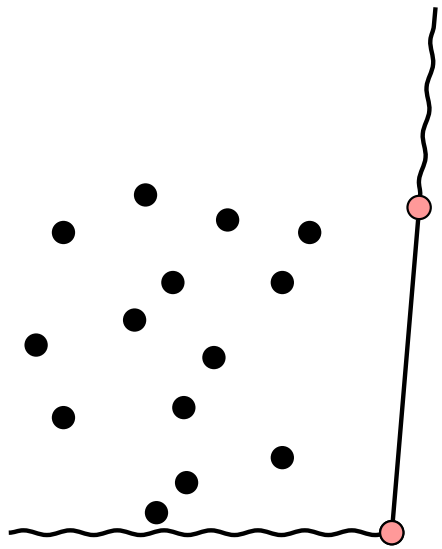
Algorithm

1. Let p_0 be the lowest point
2. Next point the one with smallest angle w.r.t. p_0
3. Continue until highest point p_k
4. Next point the one with smallest angle w.r.t. p_k

Jarvis' March (Gift wrapping)

Intuition

- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point
  3. Tape paper and go to 2



Algorithm

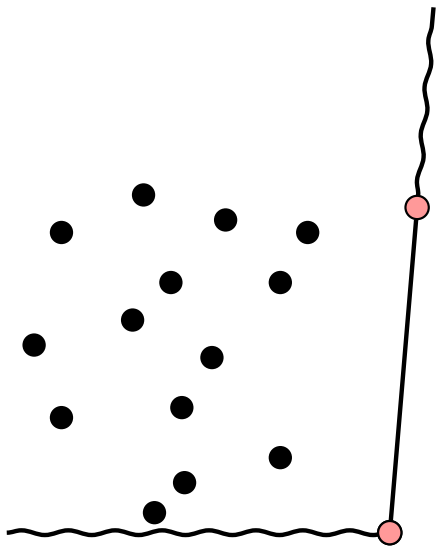
1. Let p_0 be the lowest point
2. Next point the one with smallest angle w.r.t. p_0
3. Continue until highest point p_k
4. Next point the one with smallest angle w.r.t. p_k

Here, we rotate the coordinate system by 180°!

Jarvis' March (Gift wrapping)

Intuition

- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point
  3. Tape paper and go to 2



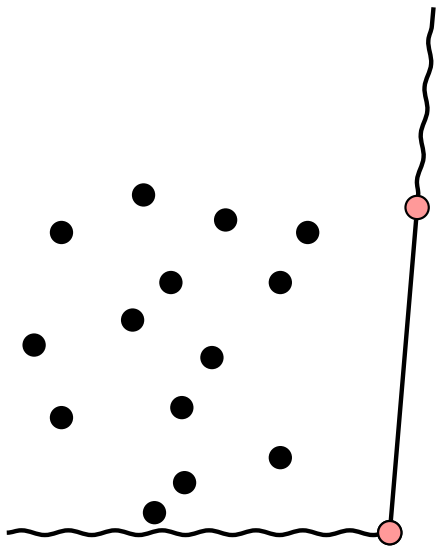
Algorithm

1. Let p_0 be the lowest point
2. Next point the one with smallest angle w.r.t. p_0
3. Continue until highest point p_k
4. Next point the one with smallest angle w.r.t. p_k
5. Continue until p_0 is reached

Jarvis' March (Gift wrapping)

Intuition

- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point
  3. Tape paper and go to 2



Algorithm

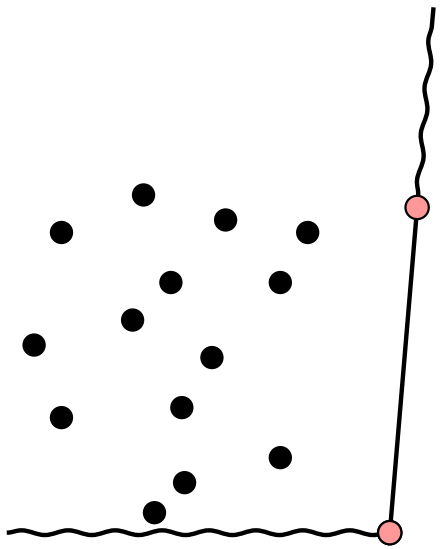
1. Let p_0 be the lowest point
2. Next point the one with smallest angle w.r.t. p_0
3. Continue until highest point p_k
4. Next point the one with smallest angle w.r.t. p_k
5. Continue until p_0 is reached

Runtime: $O(n \cdot h)$, where h is the number of points on the convex hull

Jarvis' March (Gift wrapping)

Intuition

- Wrapping taut paper around the points
  1. Tape end of paper at lowest point
  2. Pull paper to the right until it touches a point
  3. Tape paper and go to 2



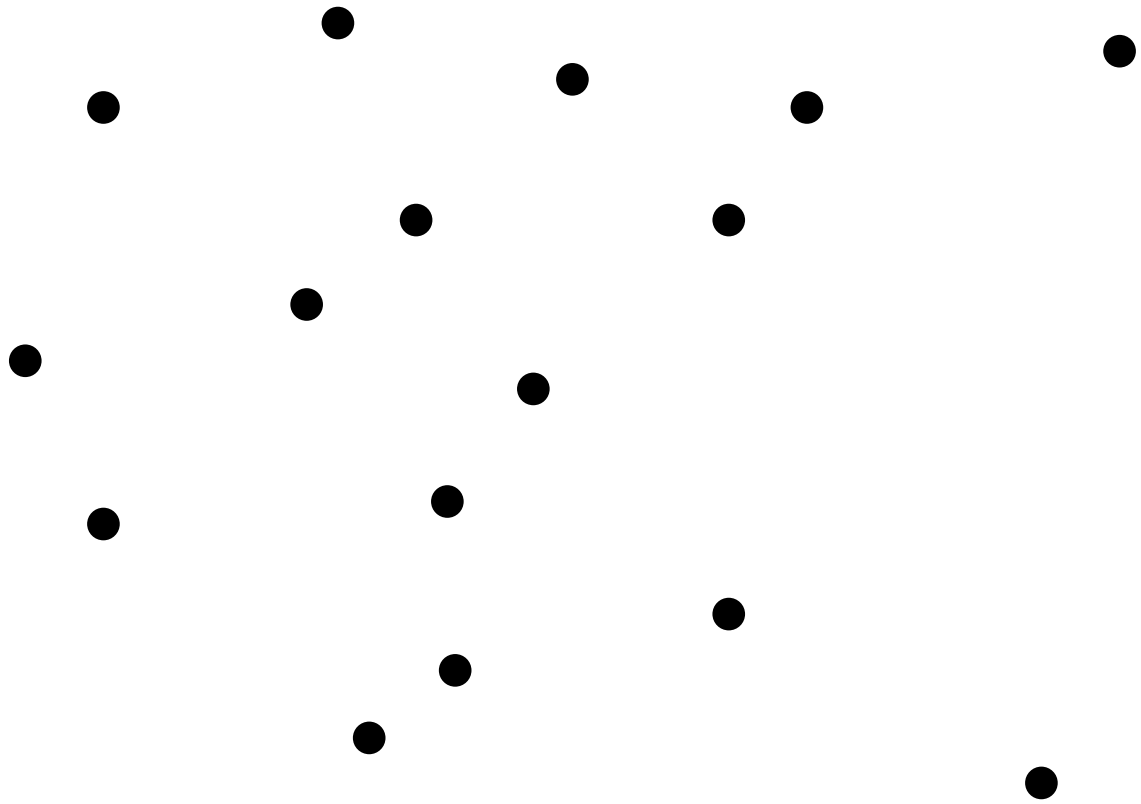
Algorithm

1. Let p_0 be the lowest point
2. Next point the one with smallest angle w.r.t. p_0
3. Continue until highest point p_k
4. Next point the one with smallest angle w.r.t. p_k
5. Continue until p_0 is reached

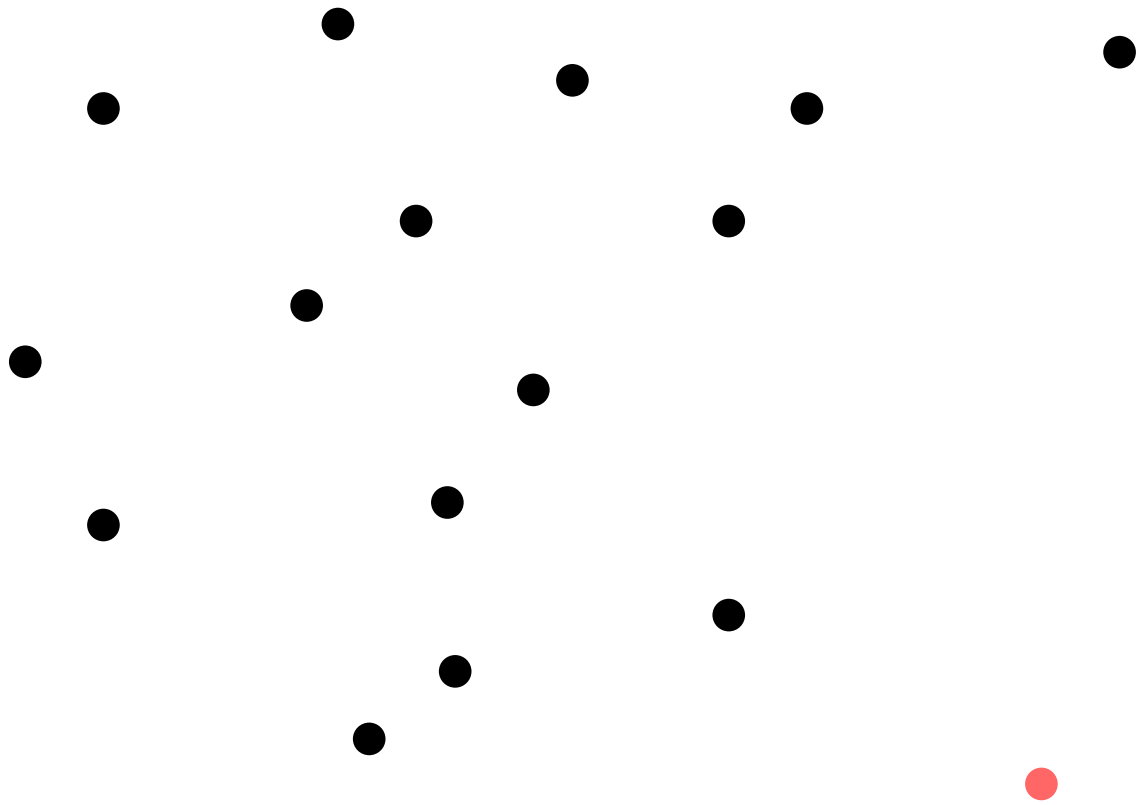
Runtime: $O(n \cdot h)$, where h is the number of points on the convex hull

Output sensitive algorithm!

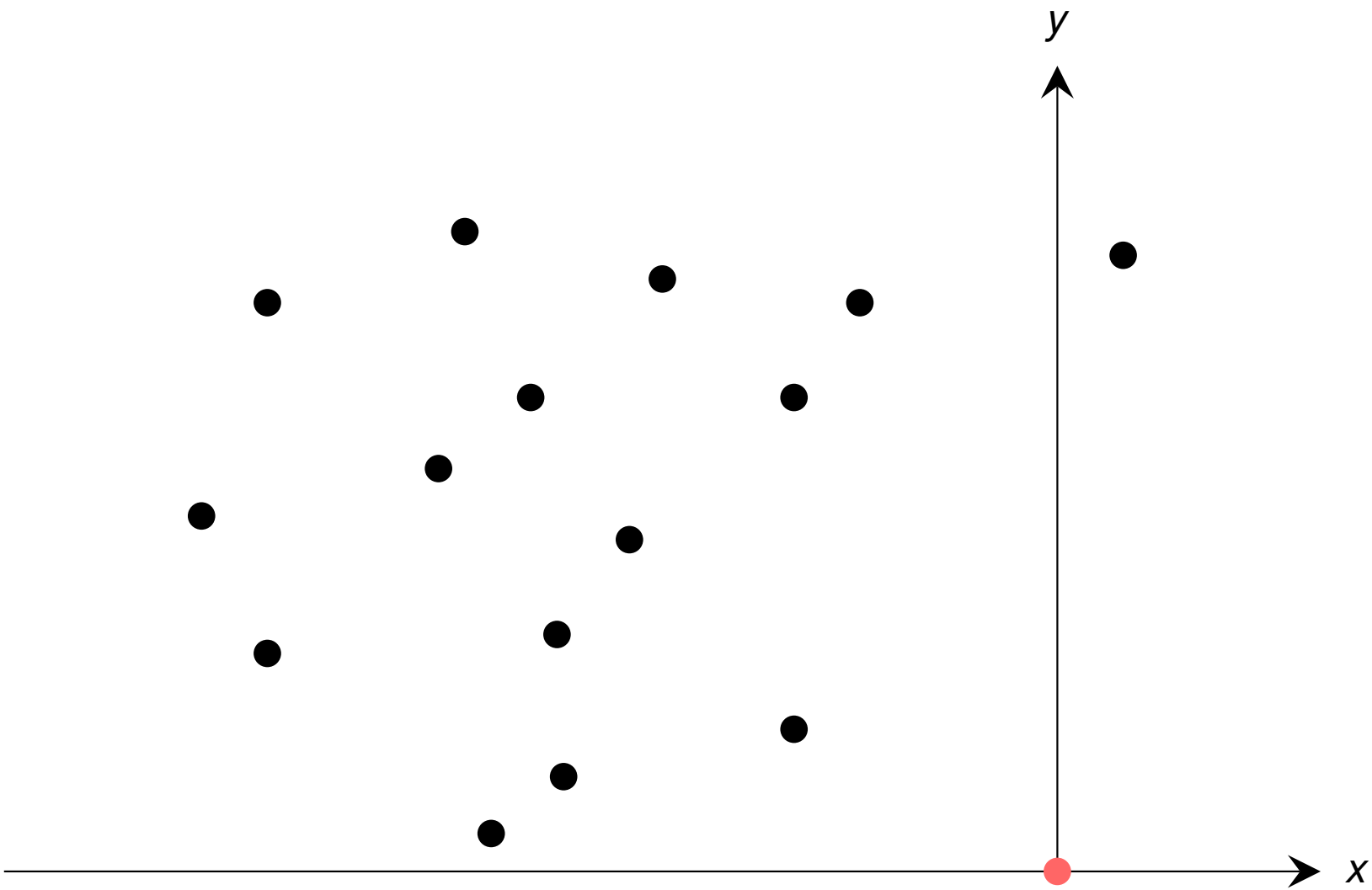
Execution of Jarvis' March



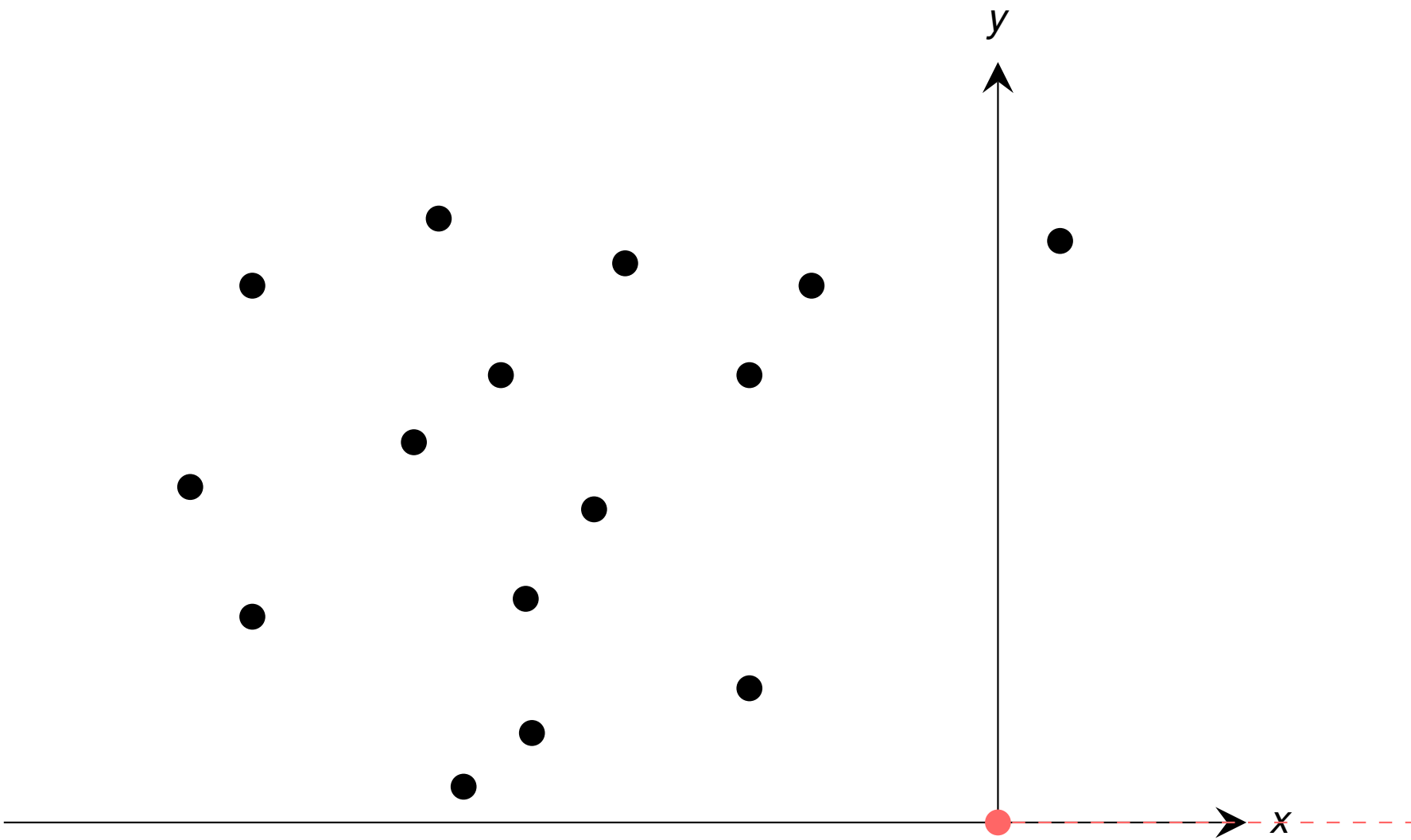
Execution of Jarvis' March



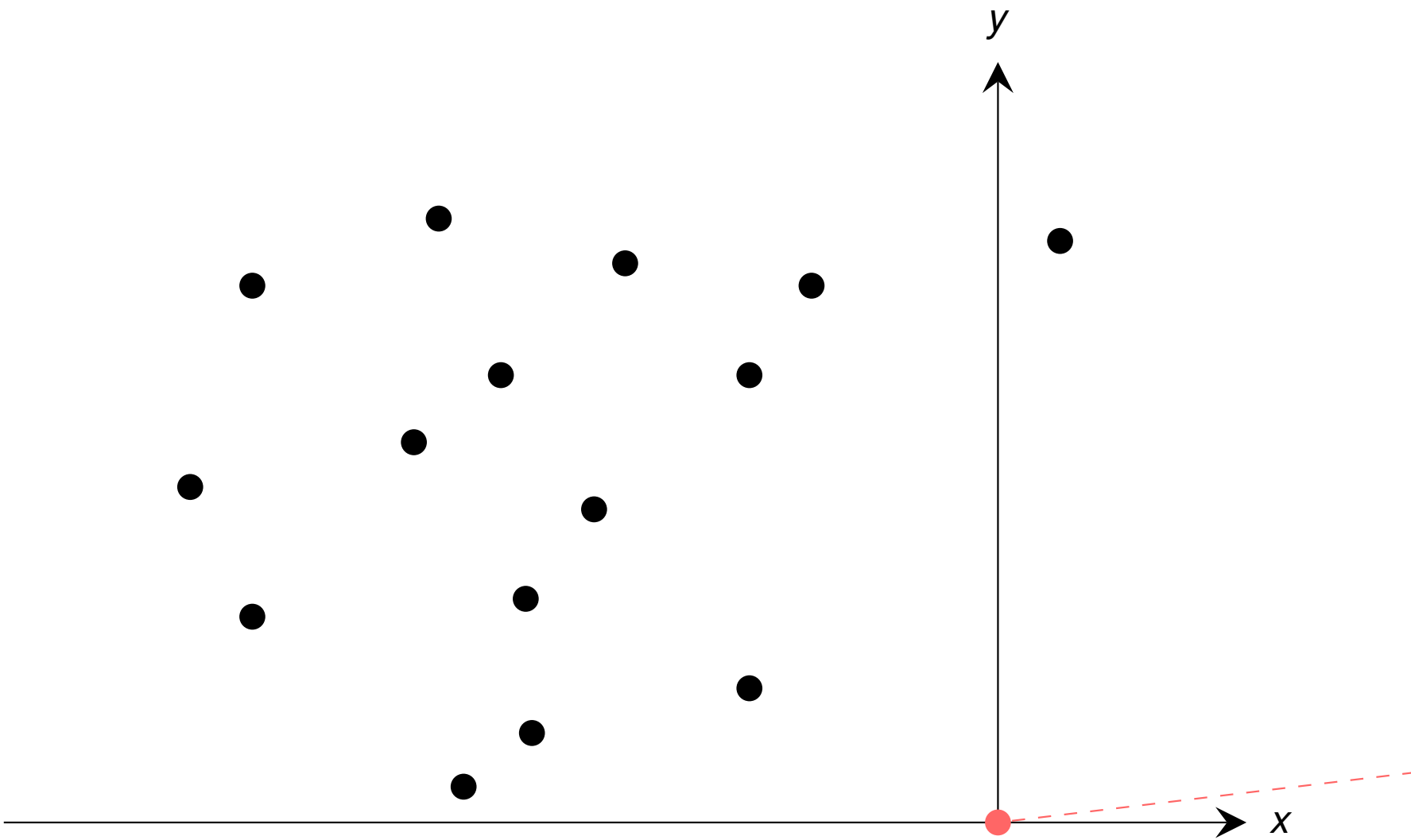
Execution of Jarvis' March



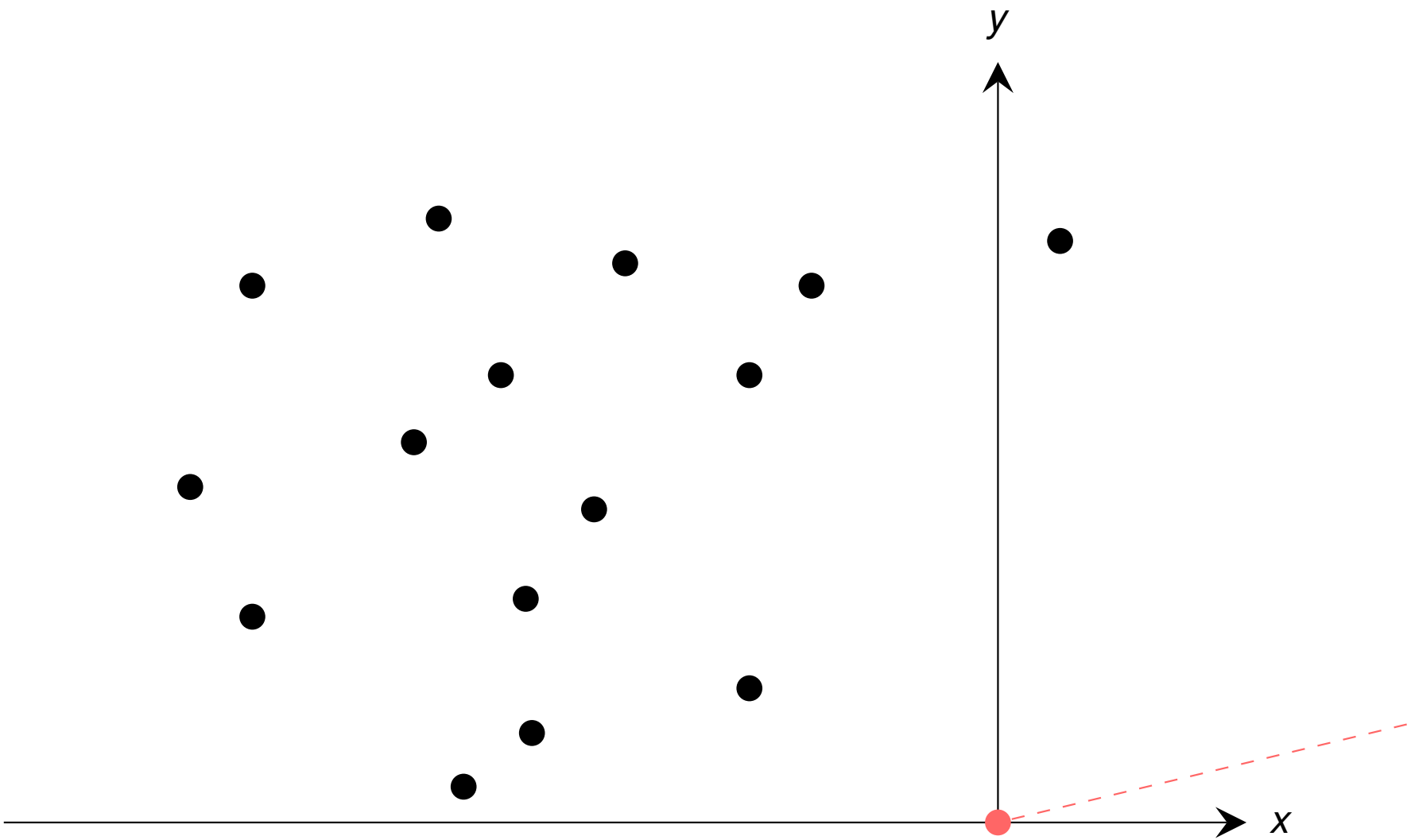
Execution of Jarvis' March



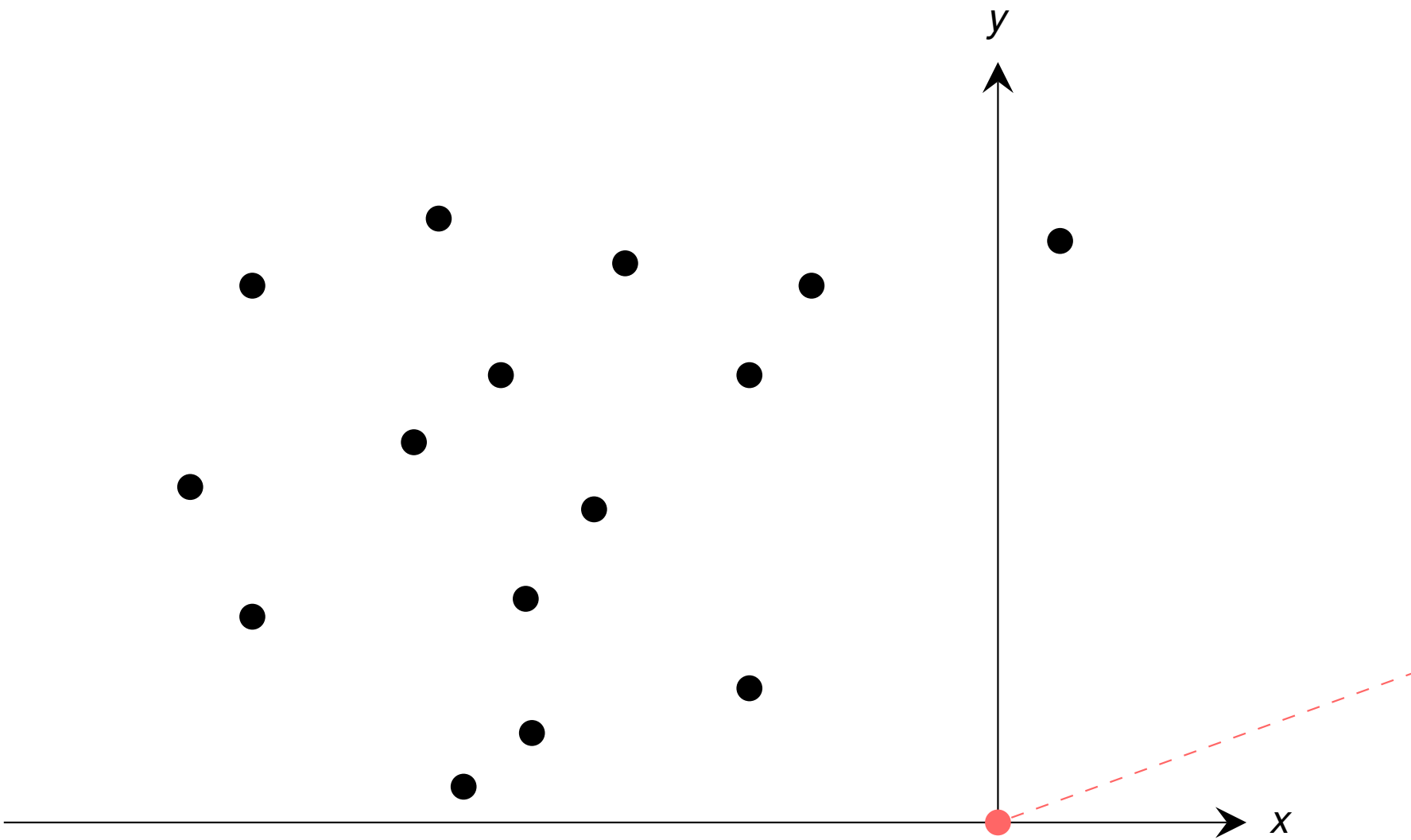
Execution of Jarvis' March



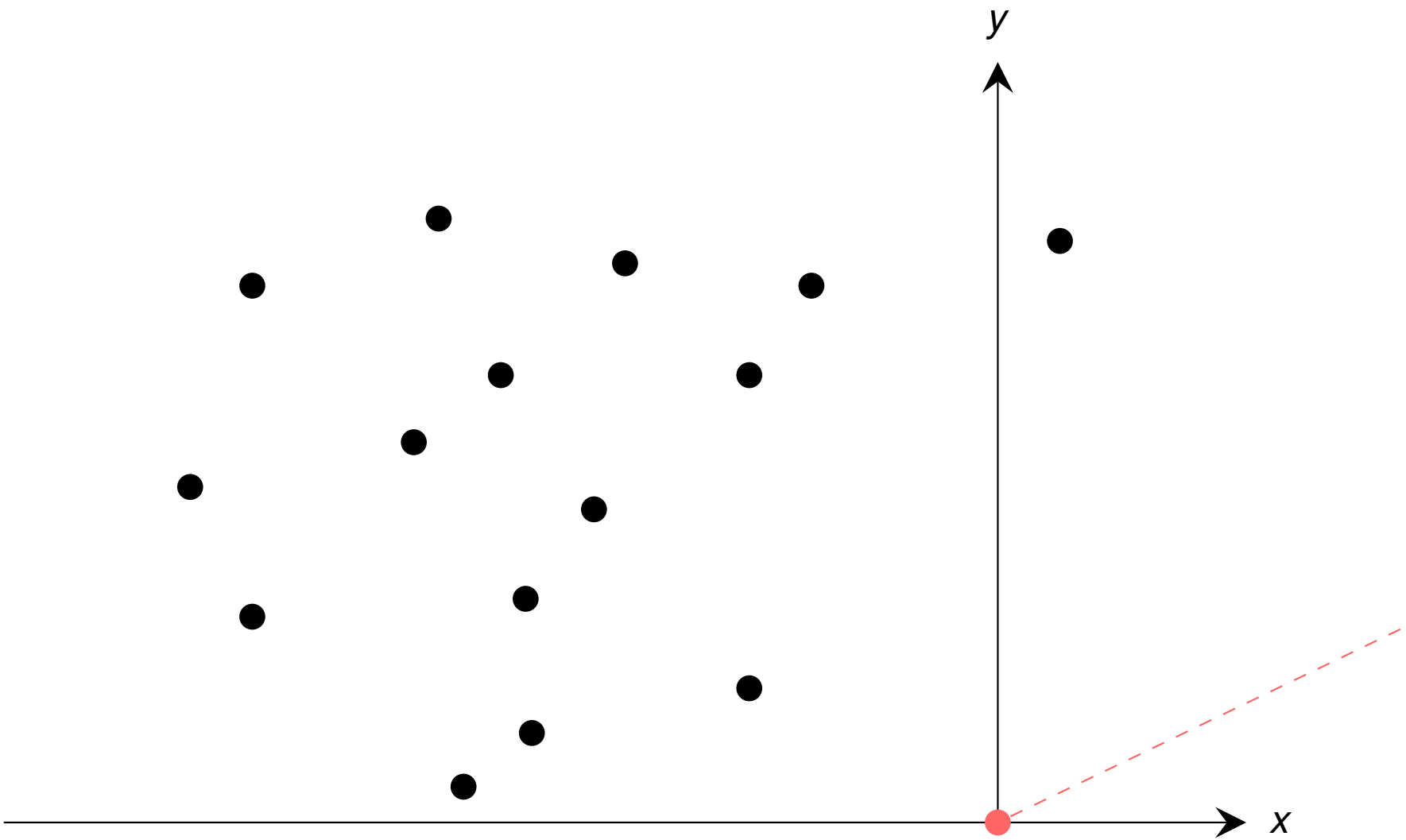
Execution of Jarvis' March



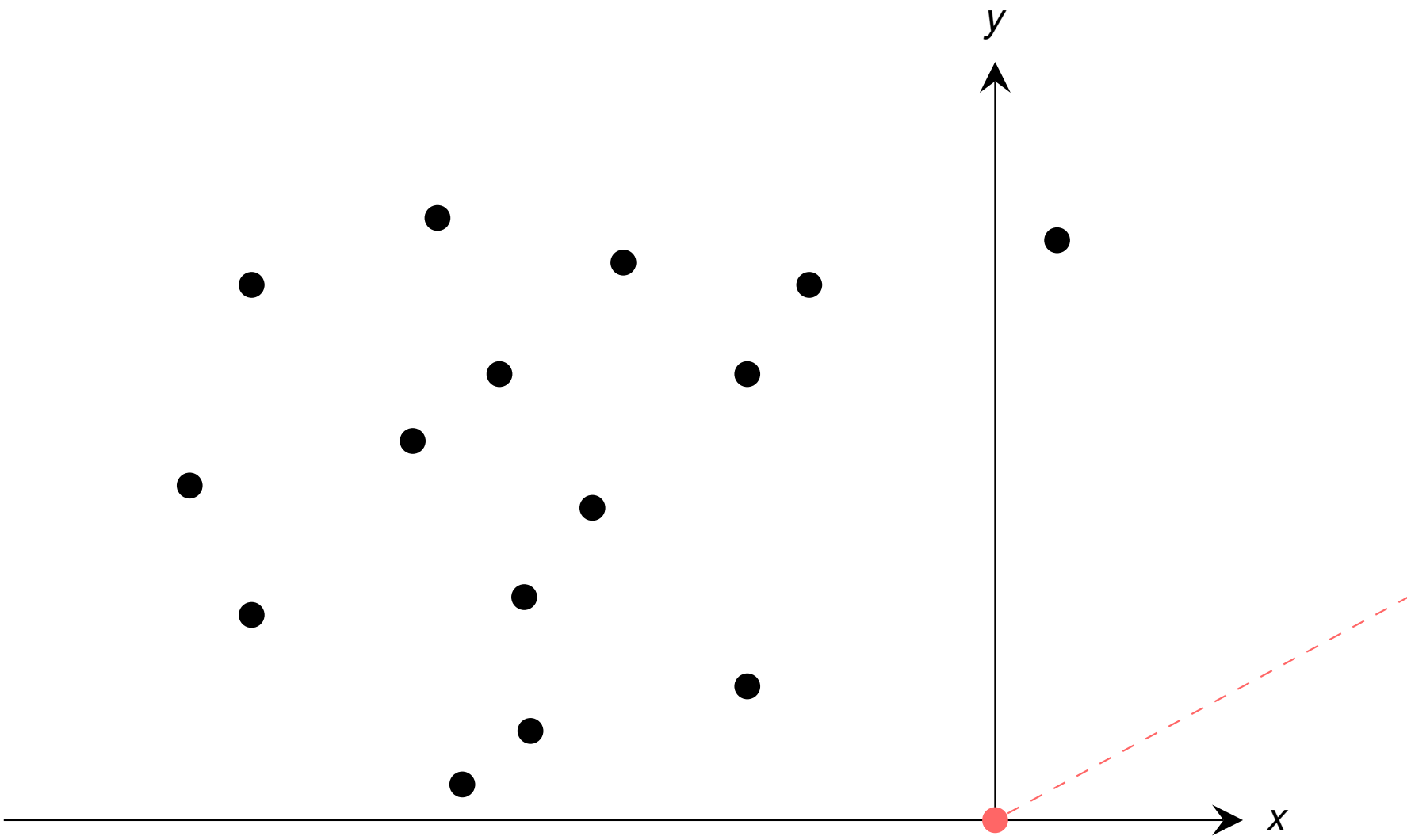
Execution of Jarvis' March



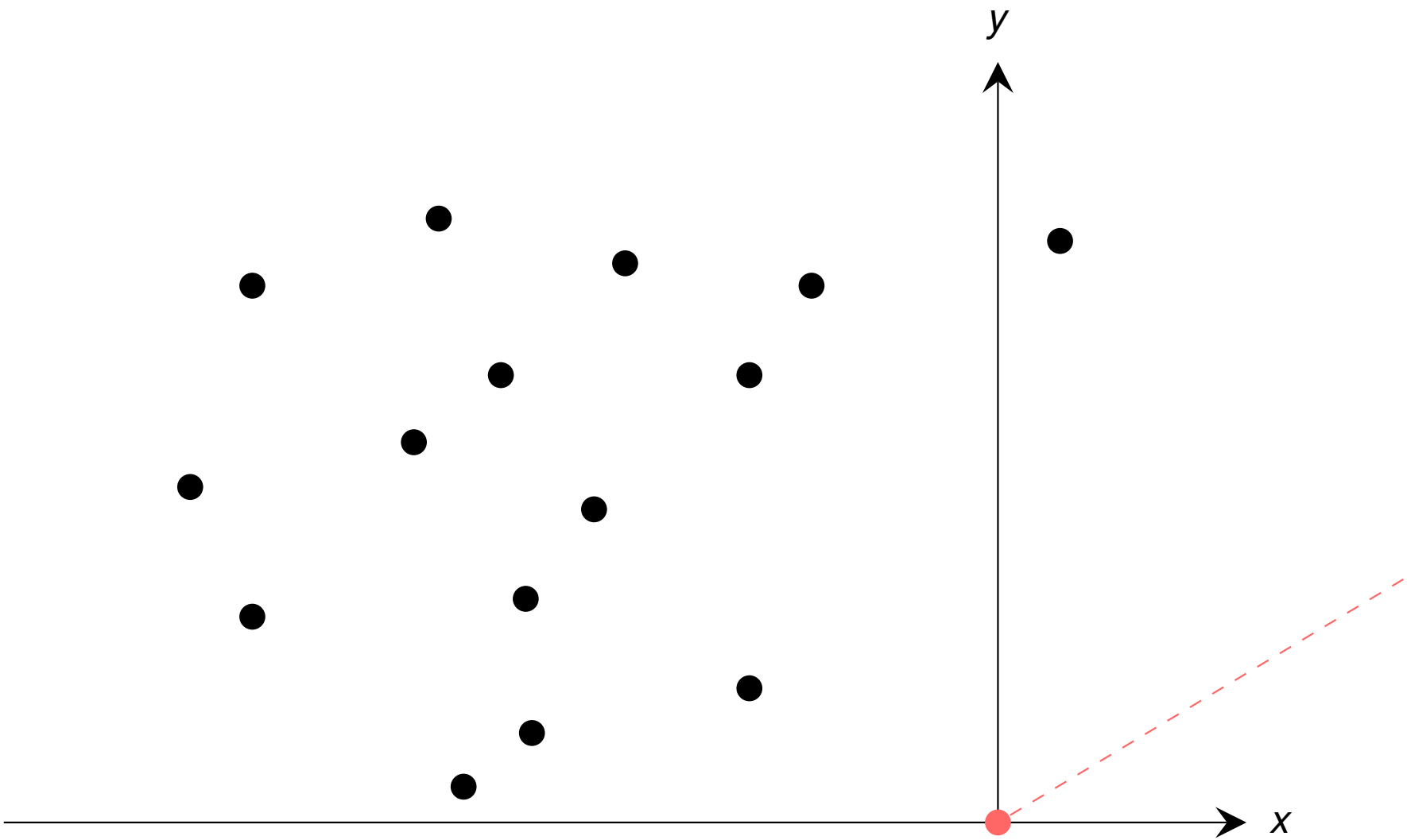
Execution of Jarvis' March



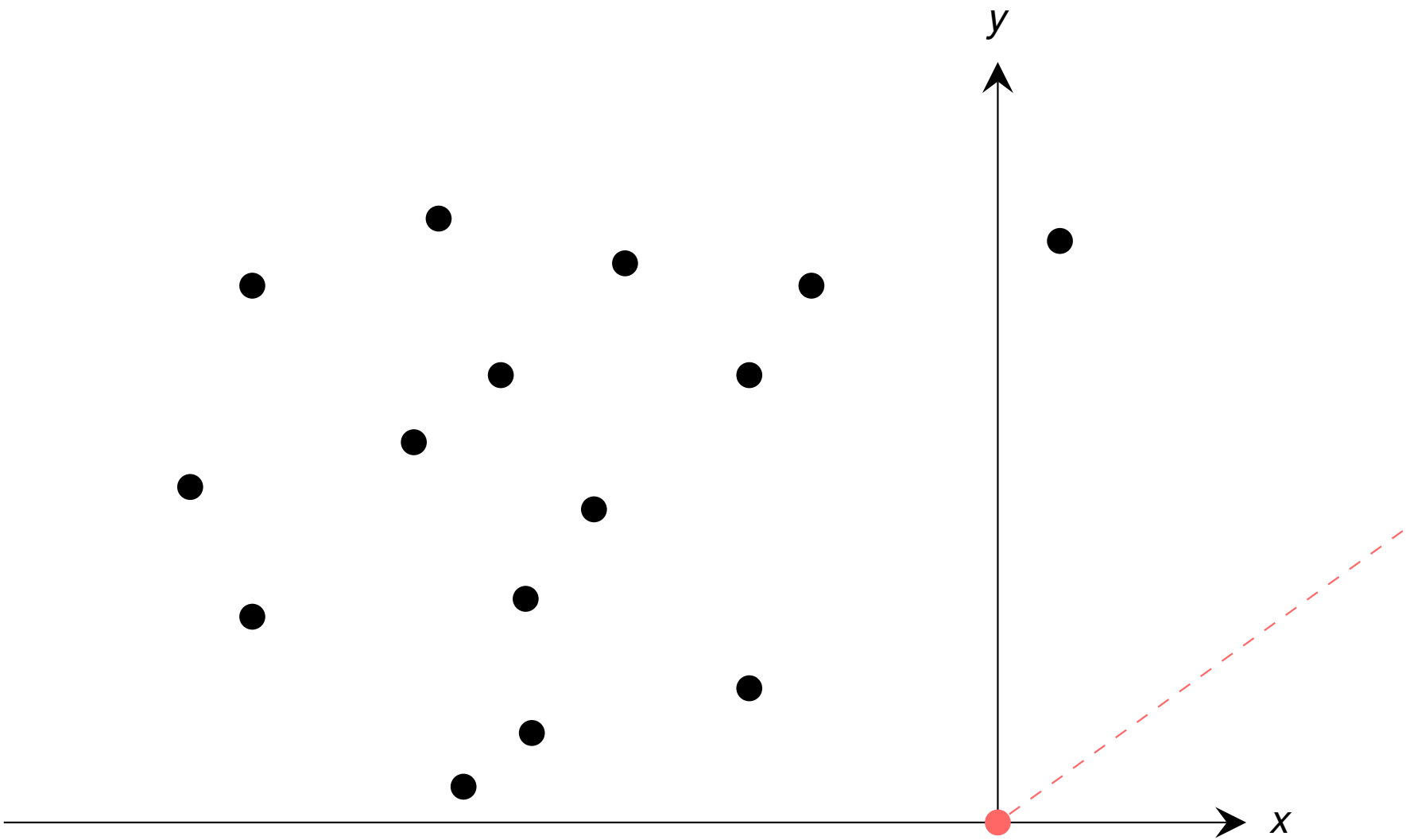
Execution of Jarvis' March



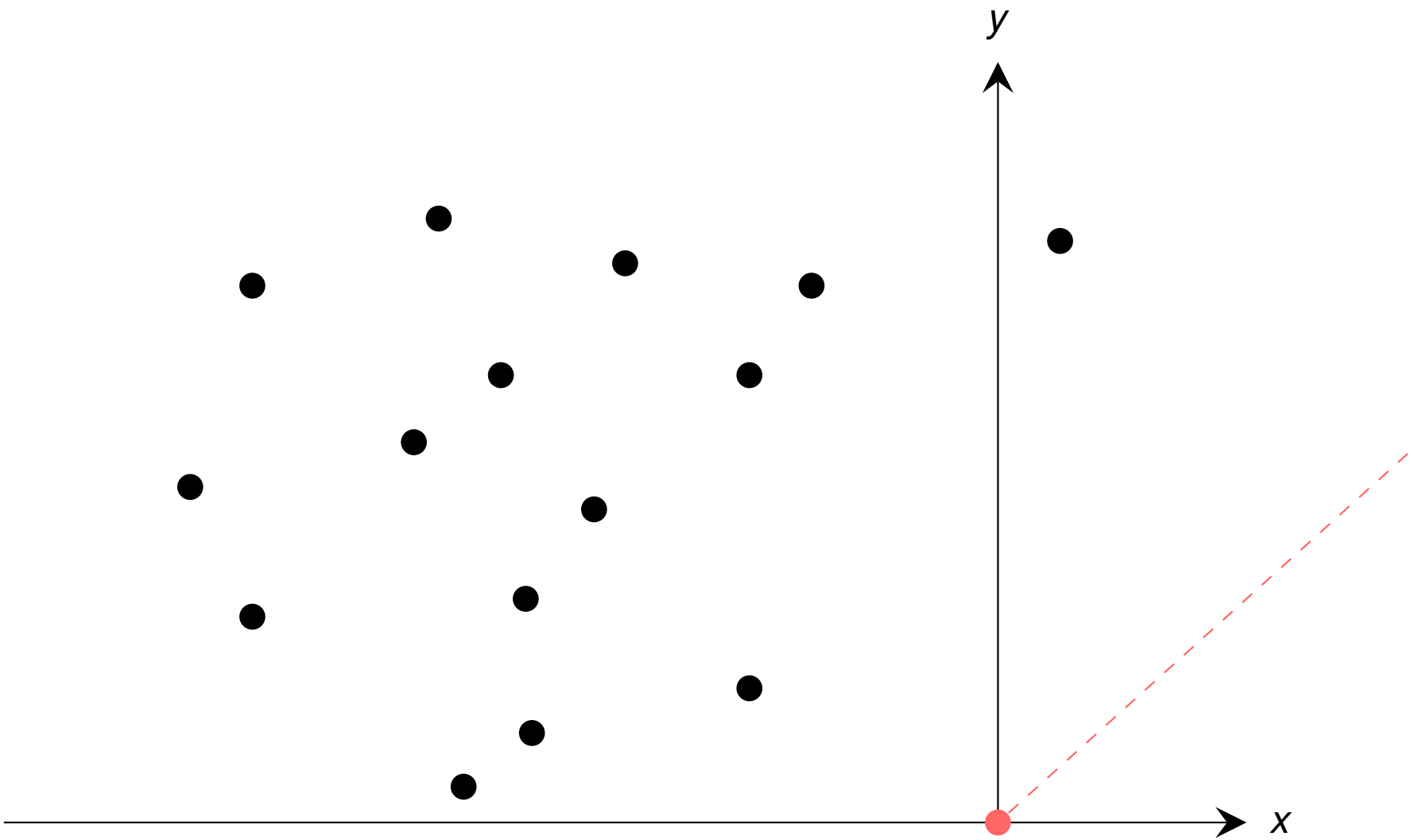
Execution of Jarvis' March



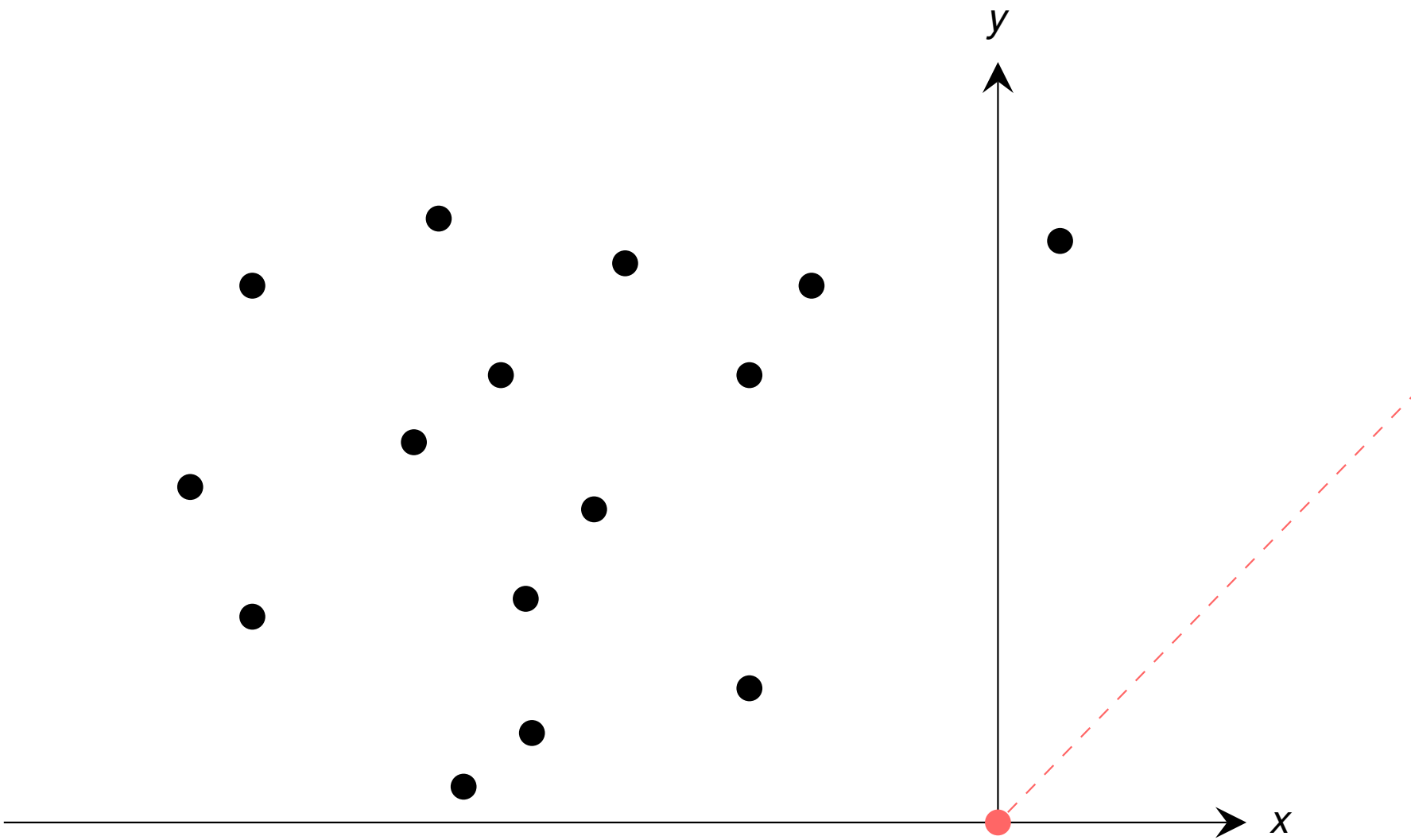
Execution of Jarvis' March



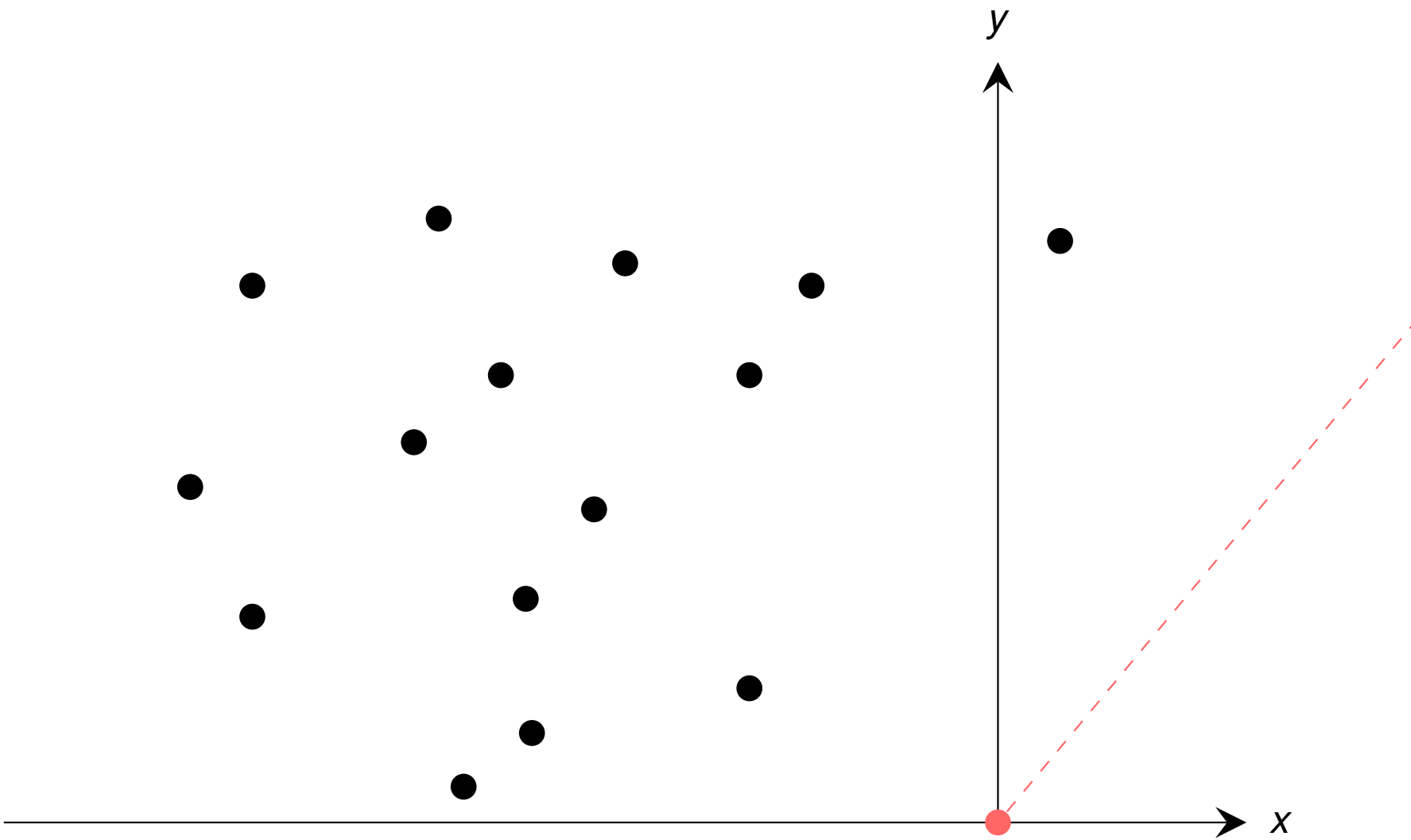
Execution of Jarvis' March



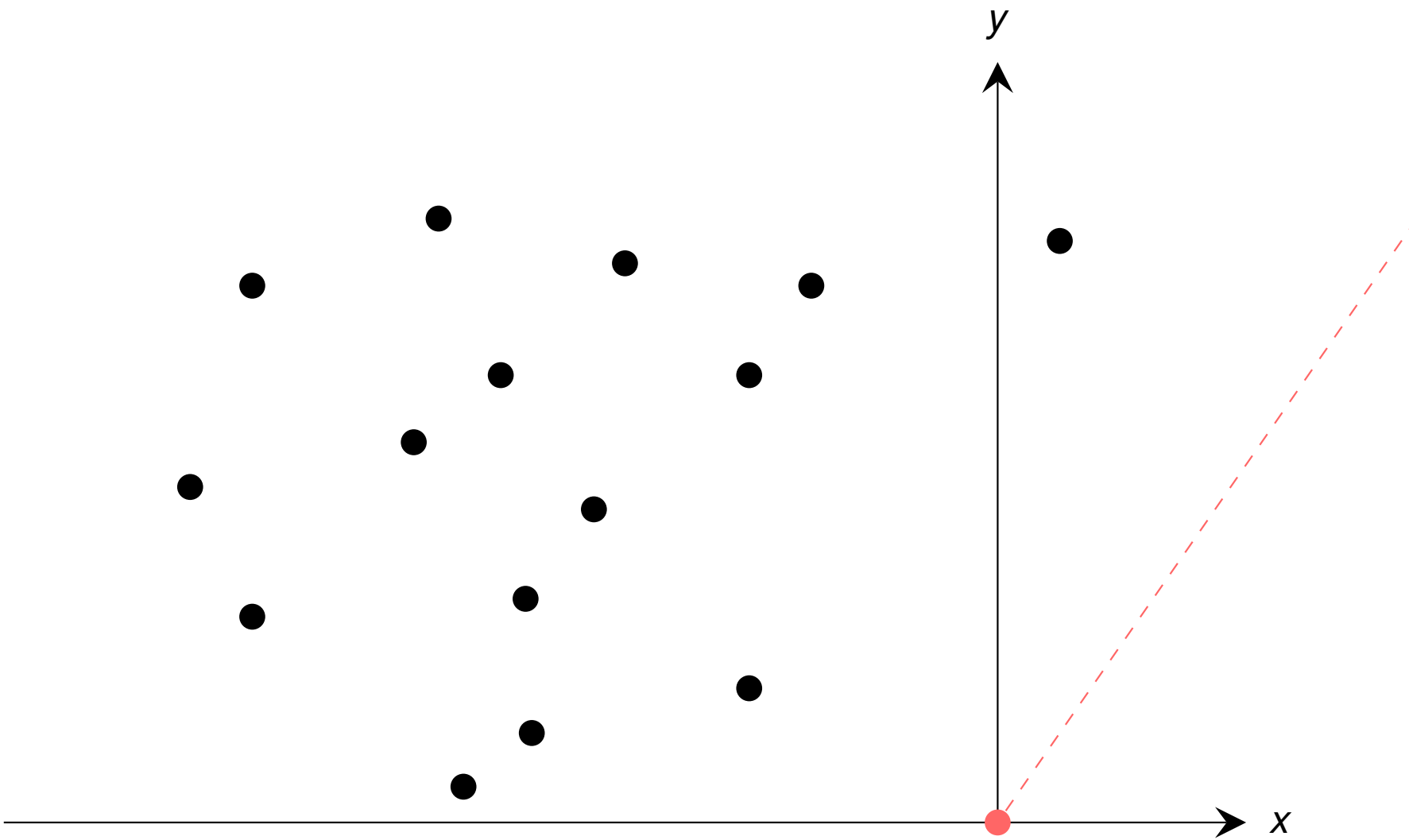
Execution of Jarvis' March



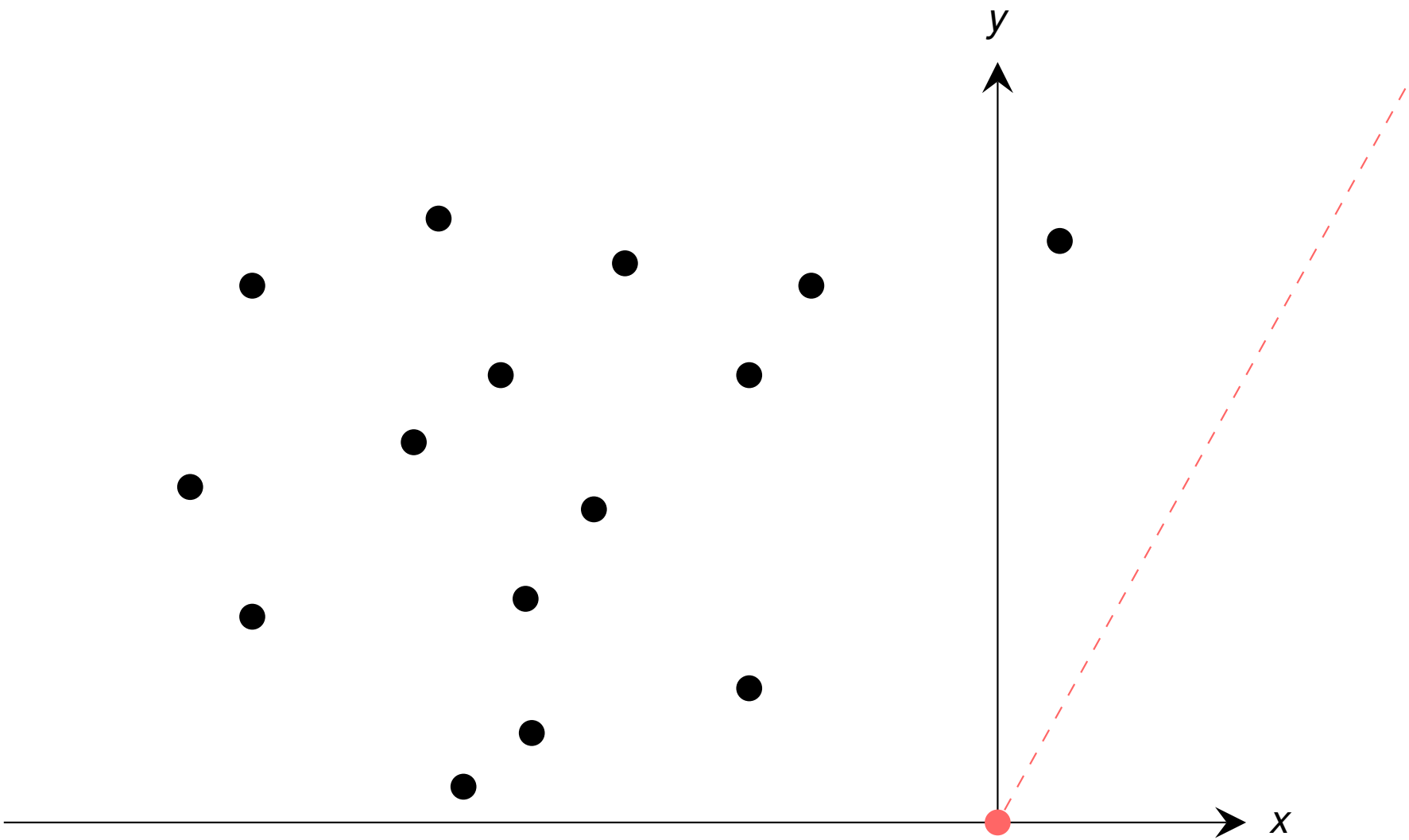
Execution of Jarvis' March



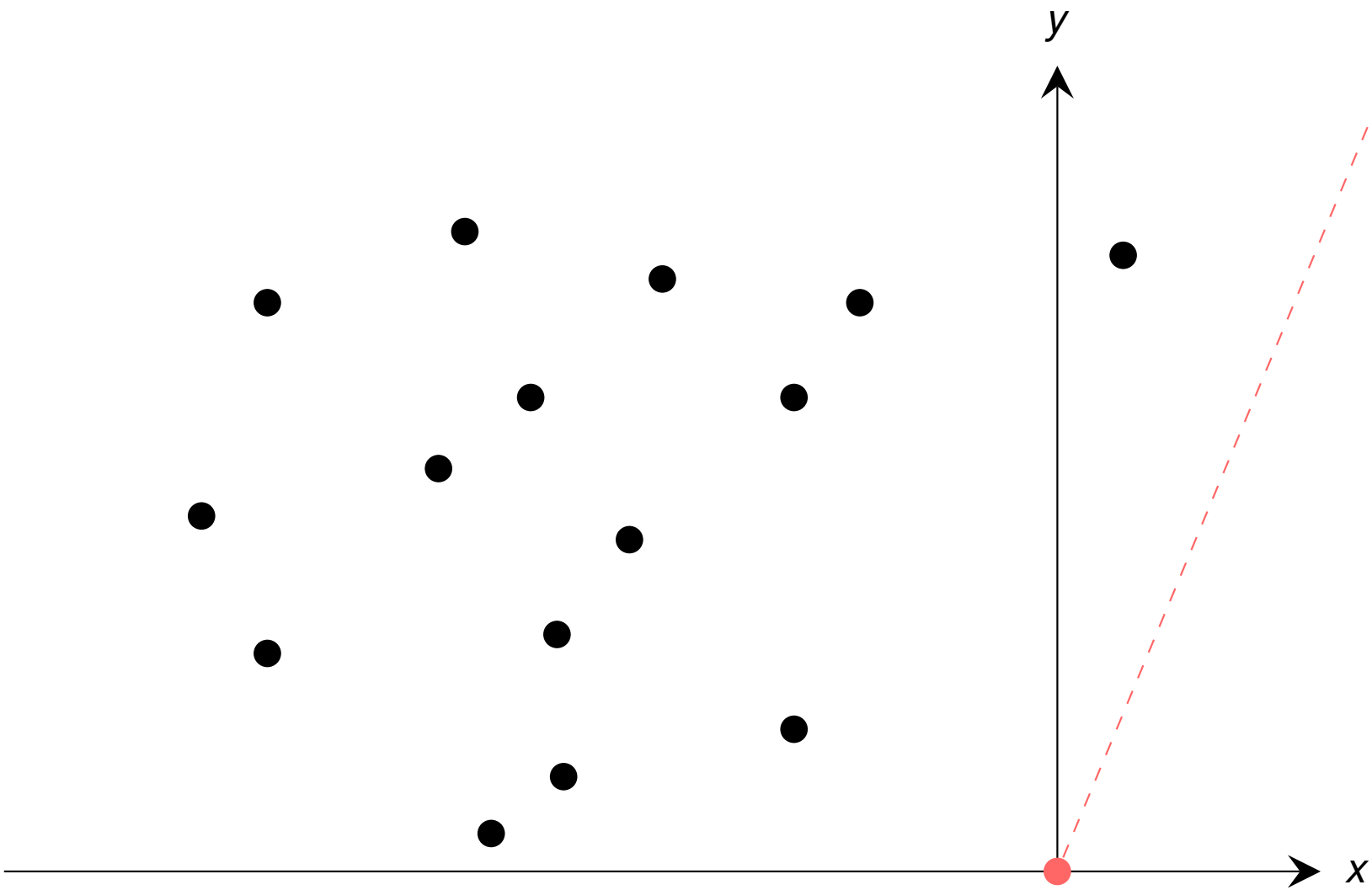
Execution of Jarvis' March



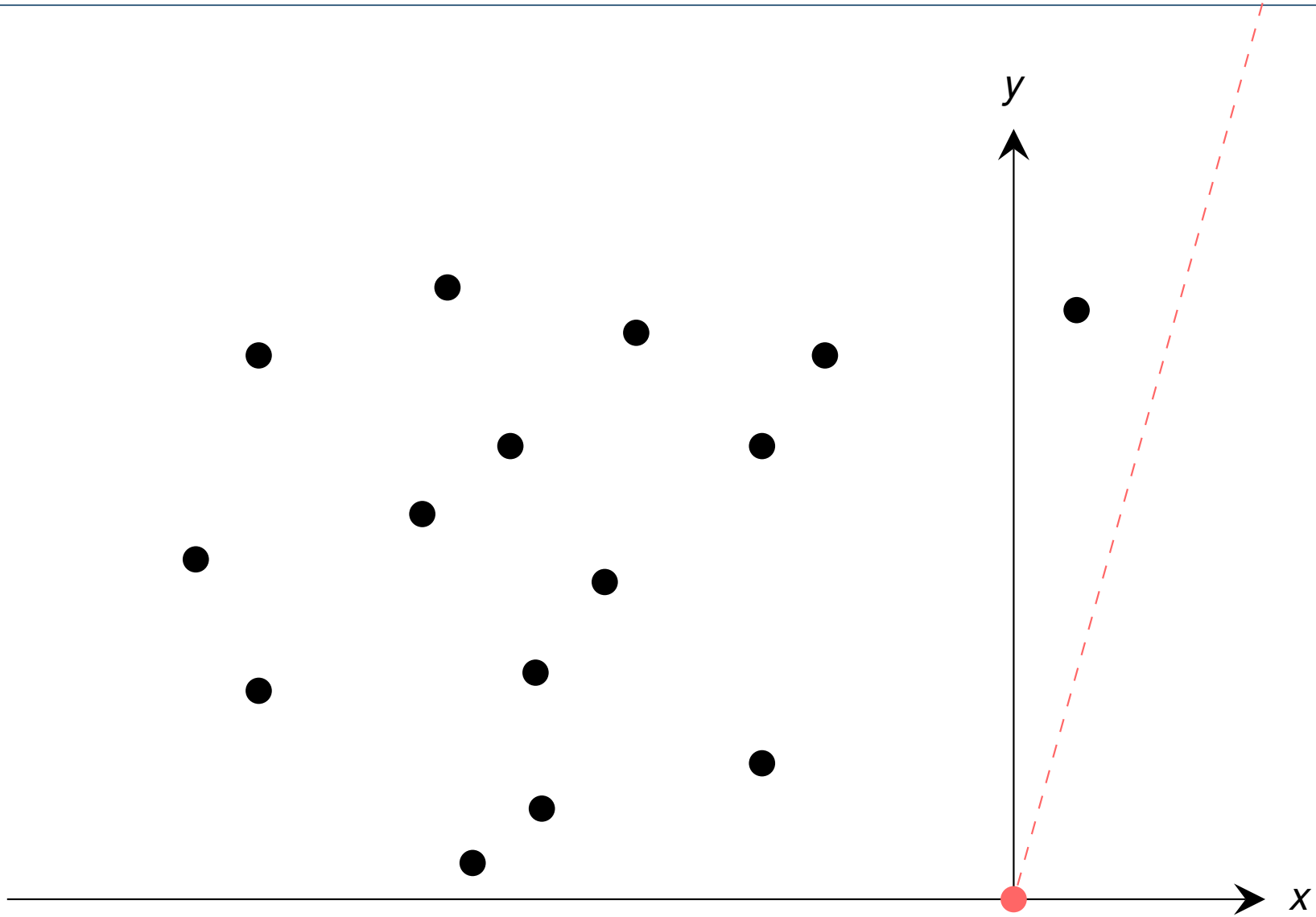
Execution of Jarvis' March



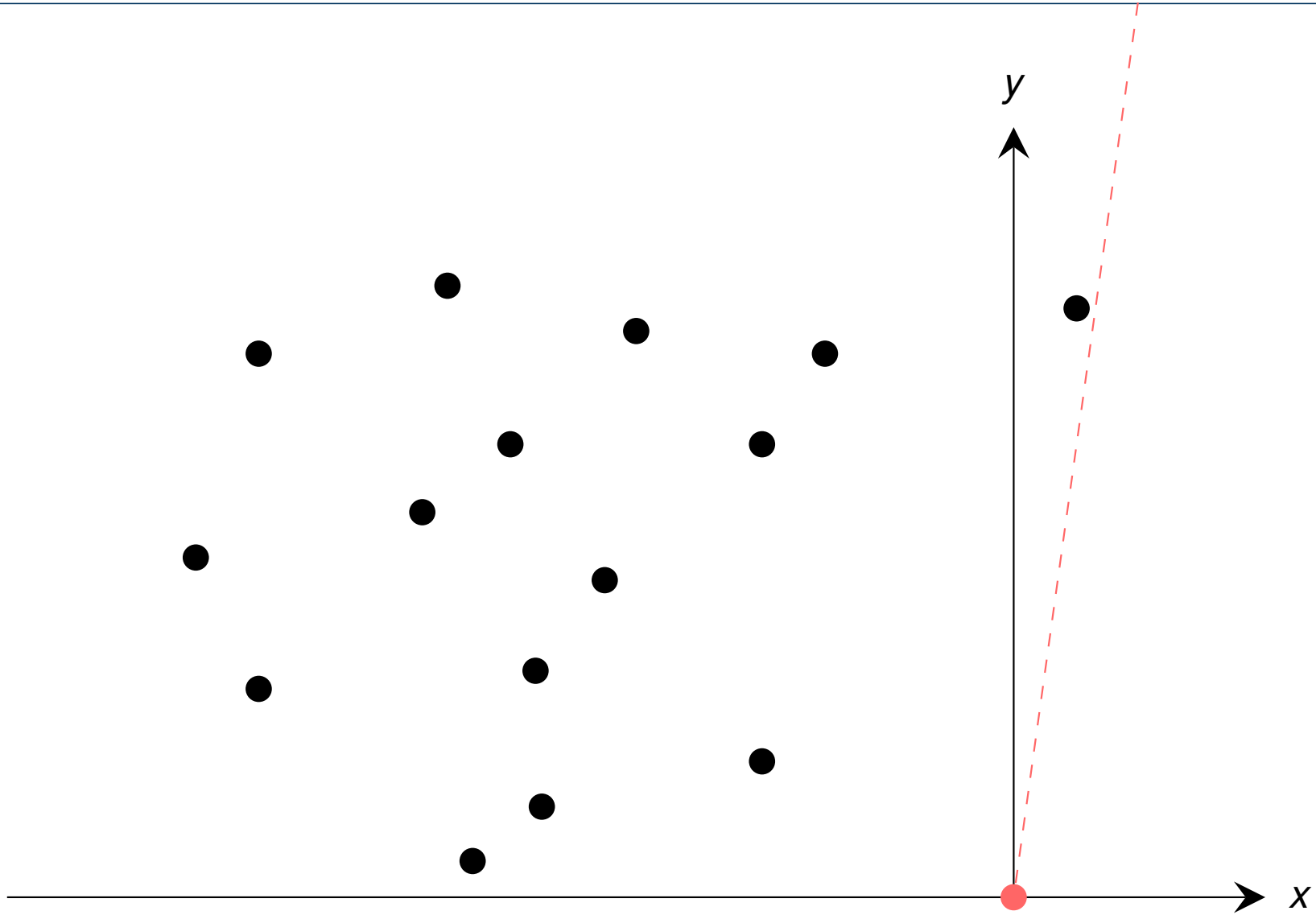
Execution of Jarvis' March



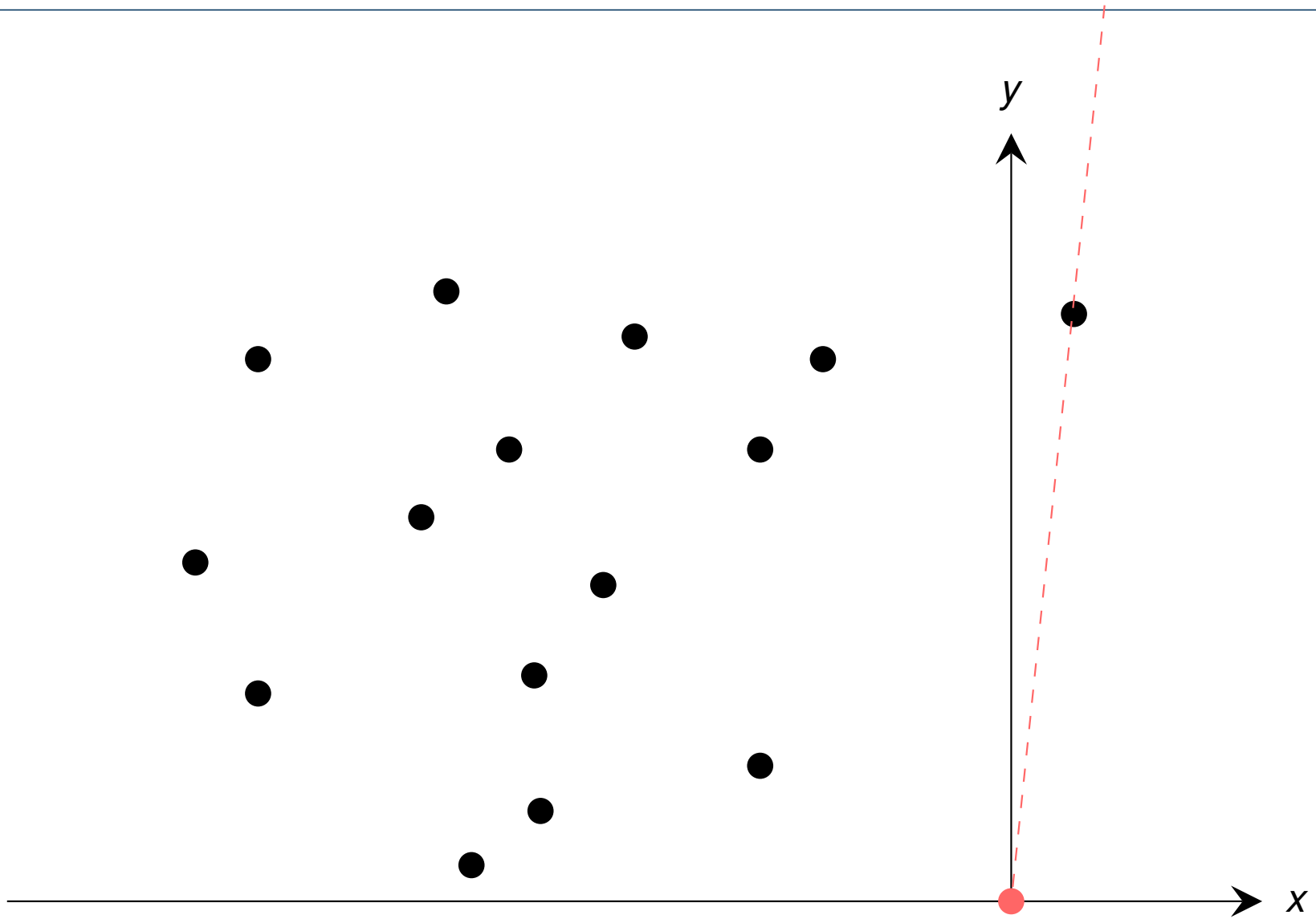
Execution of Jarvis' March



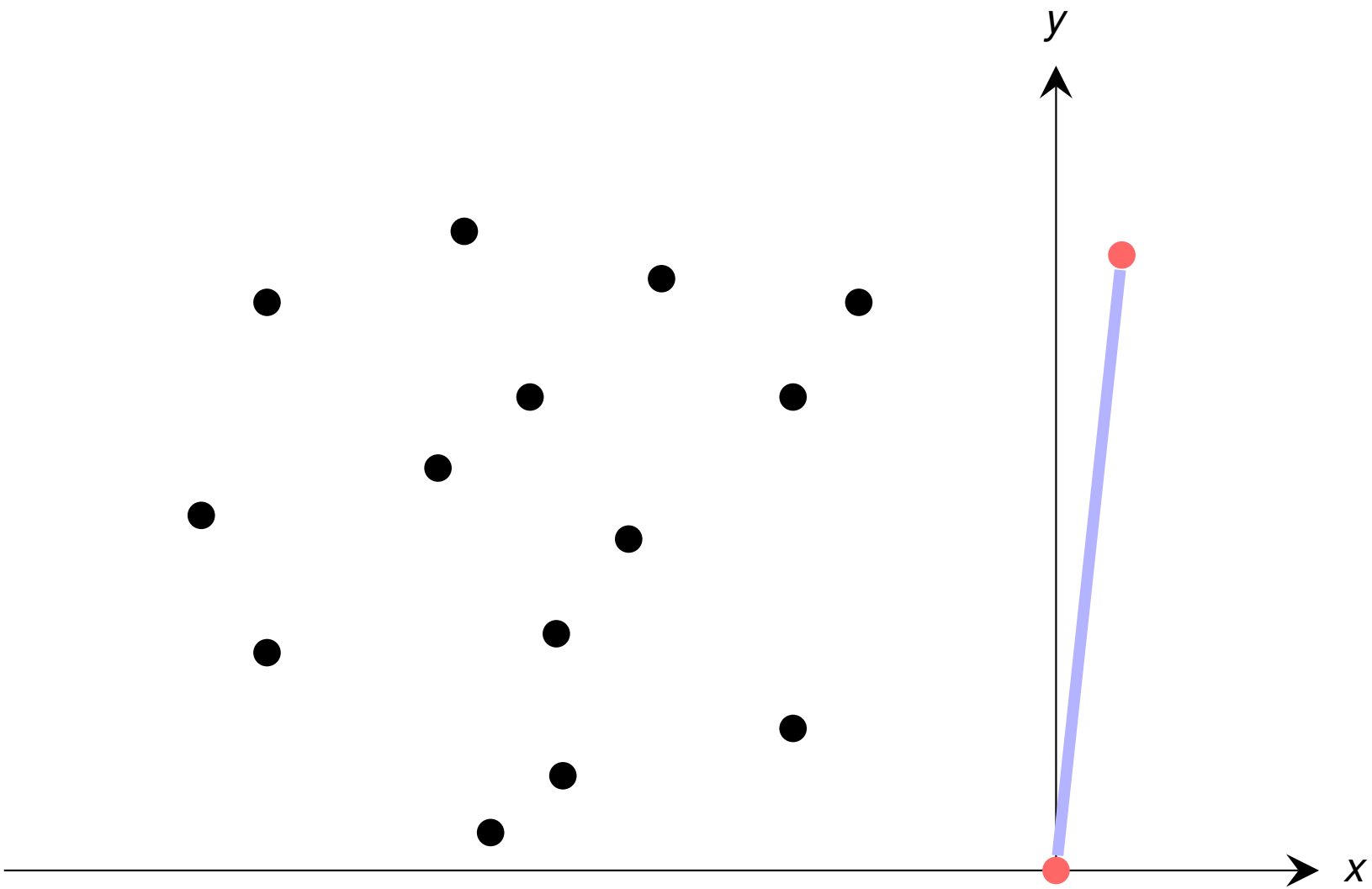
Execution of Jarvis' March



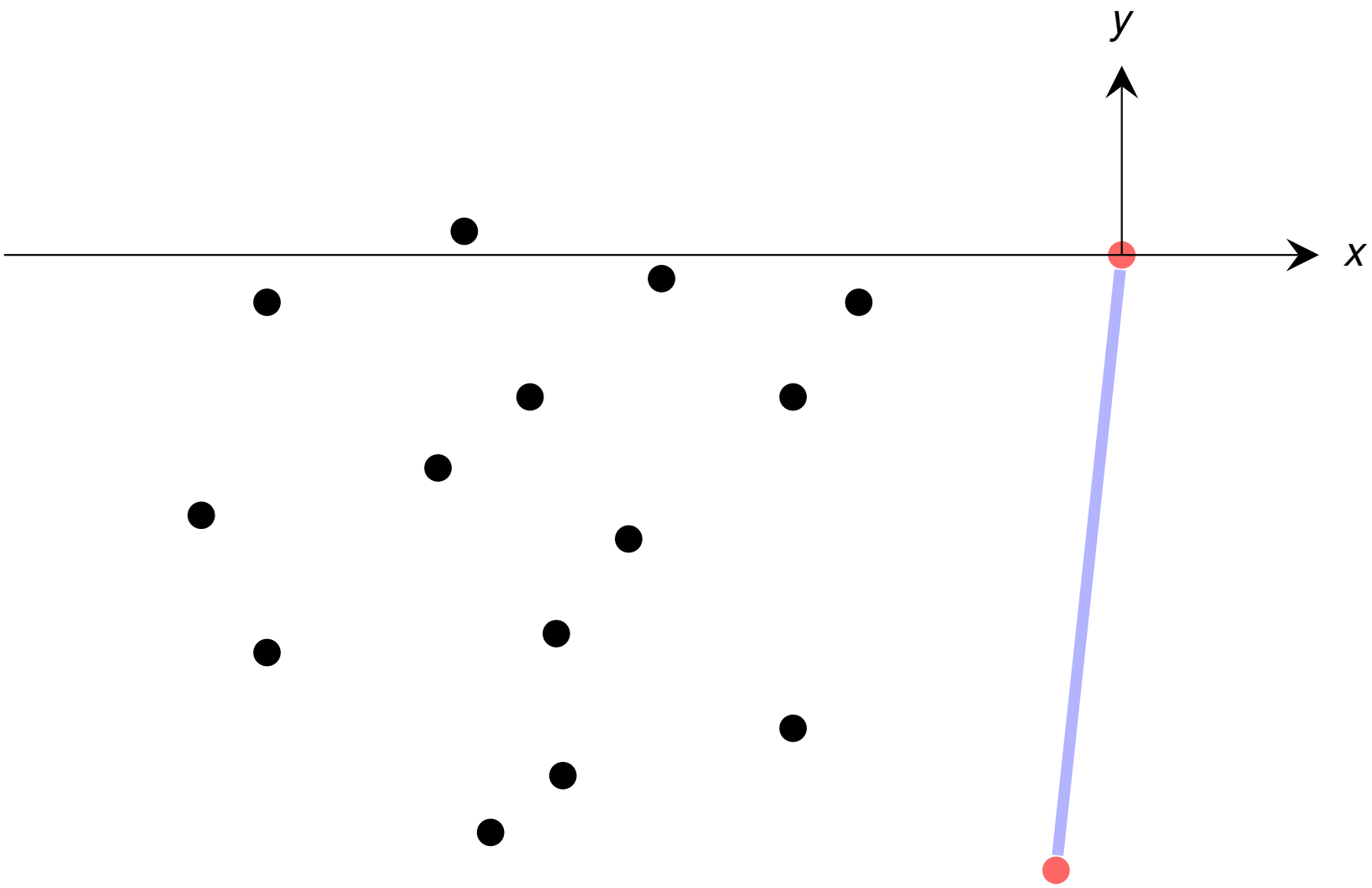
Execution of Jarvis' March



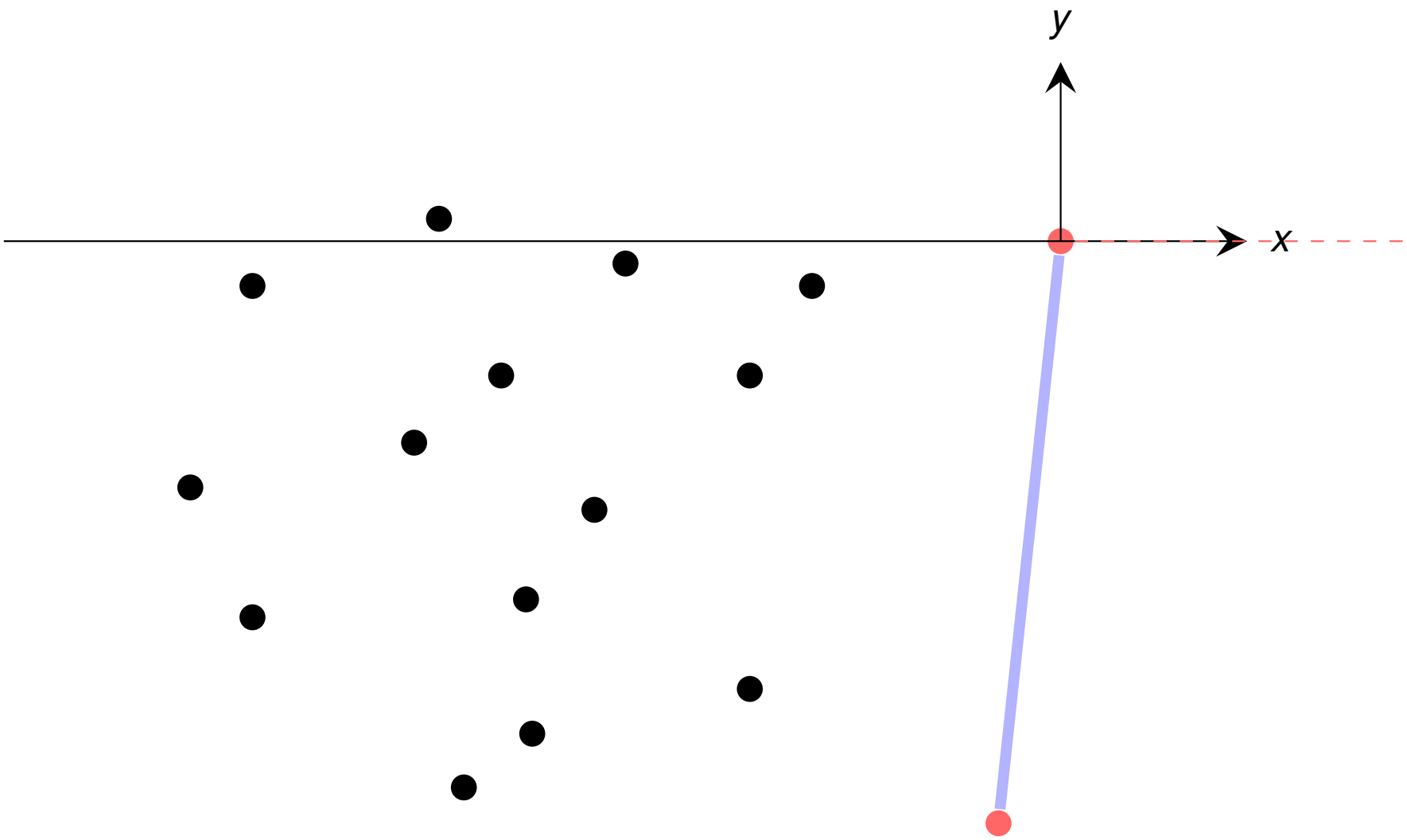
Execution of Jarvis' March



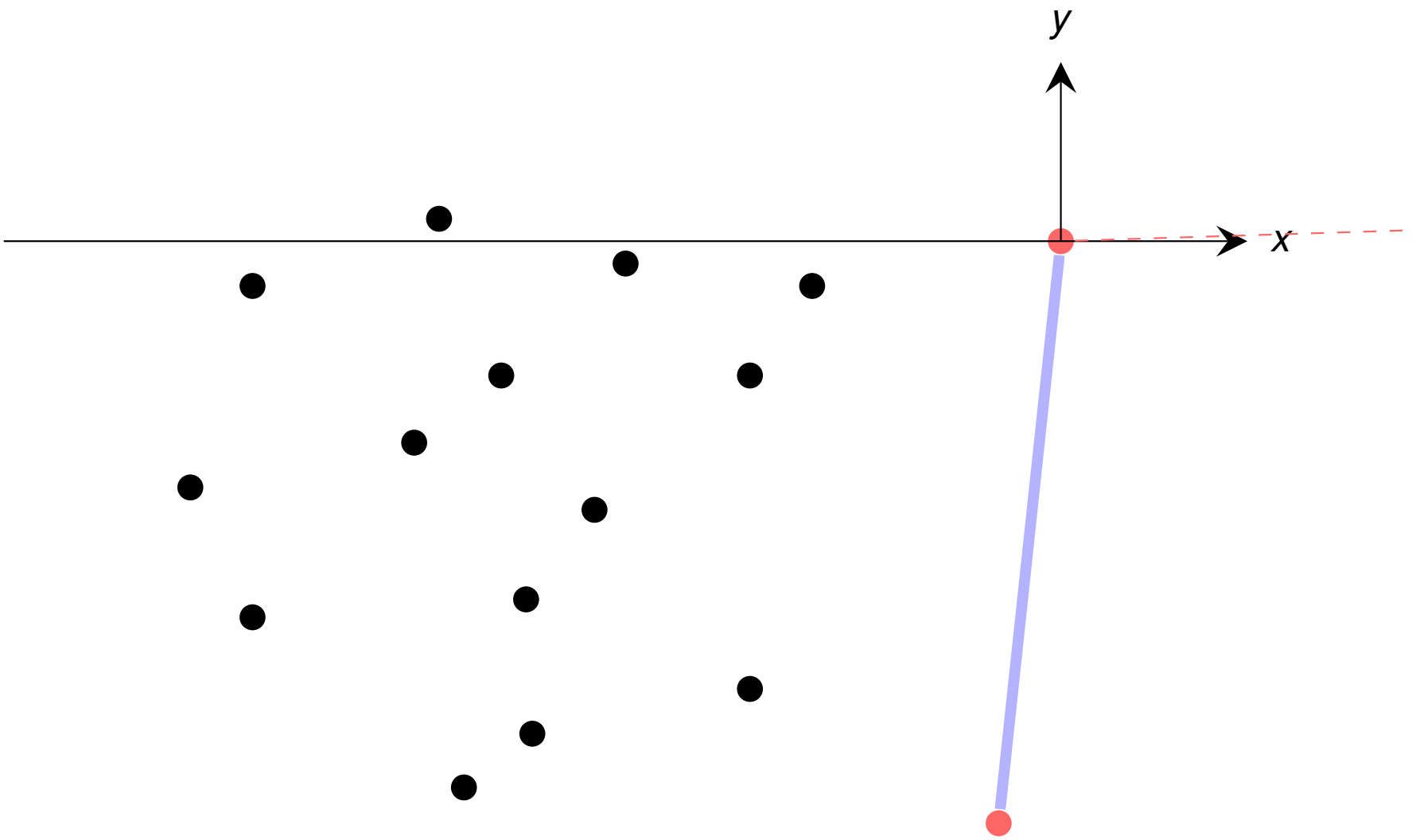
Execution of Jarvis' March



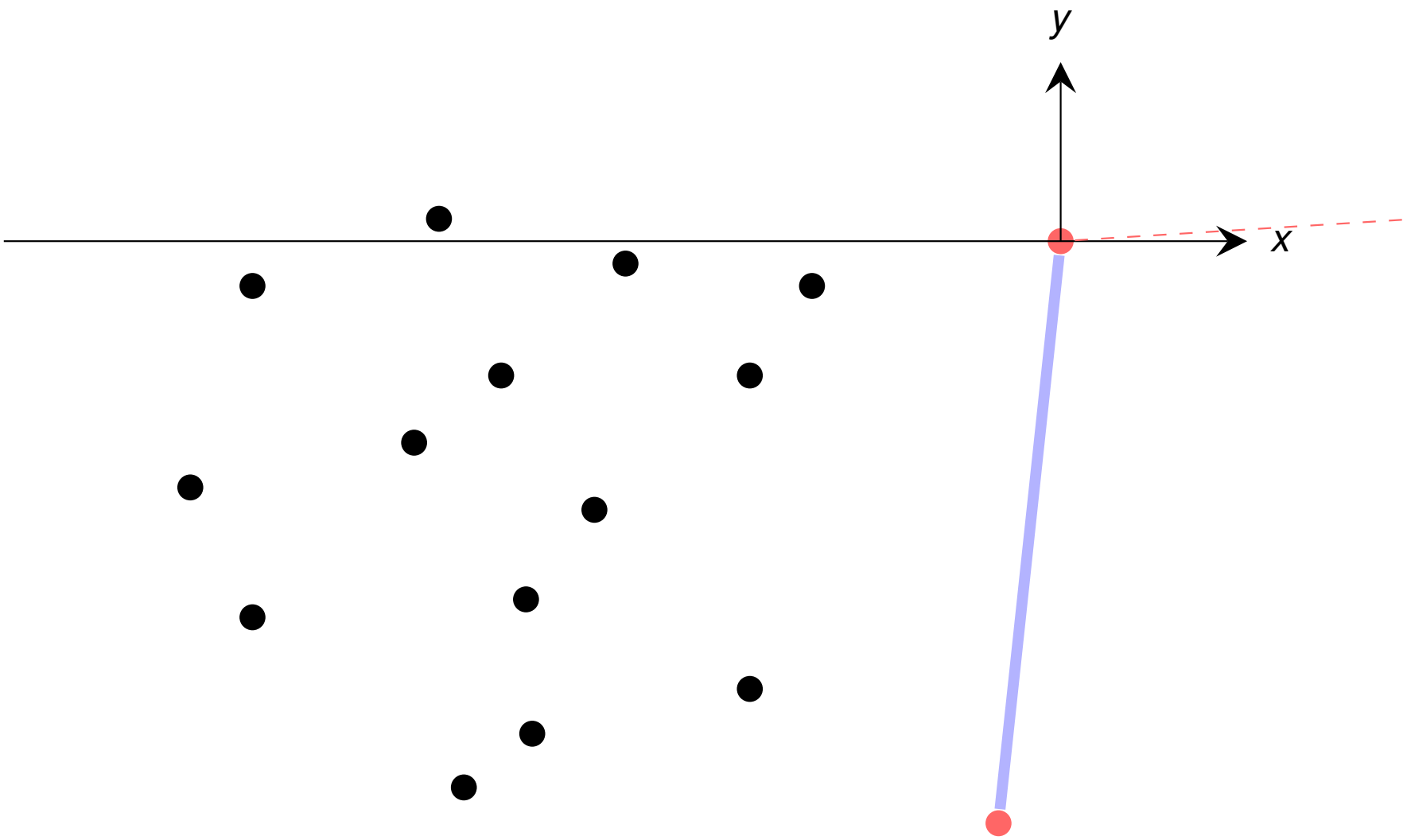
Execution of Jarvis' March



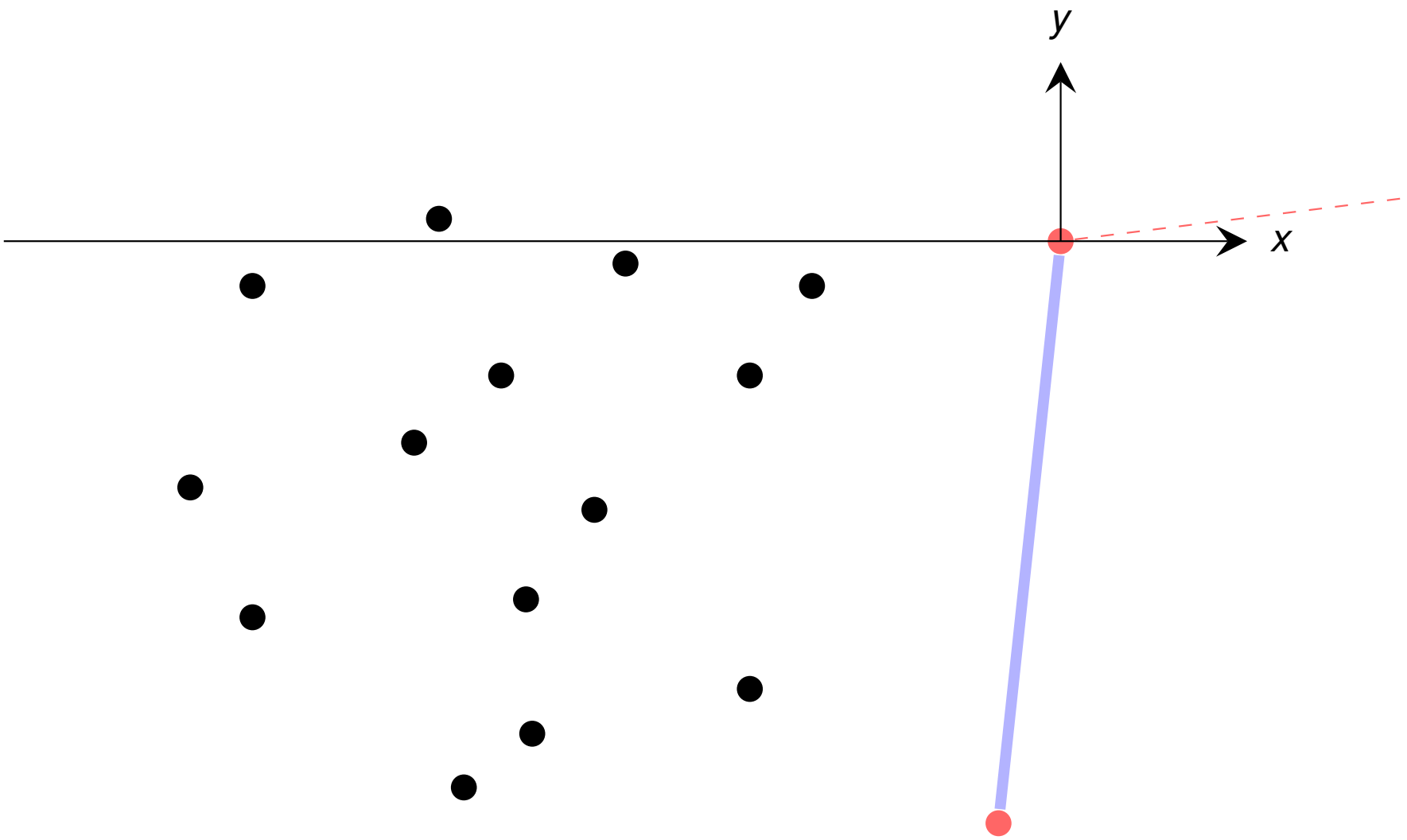
Execution of Jarvis' March



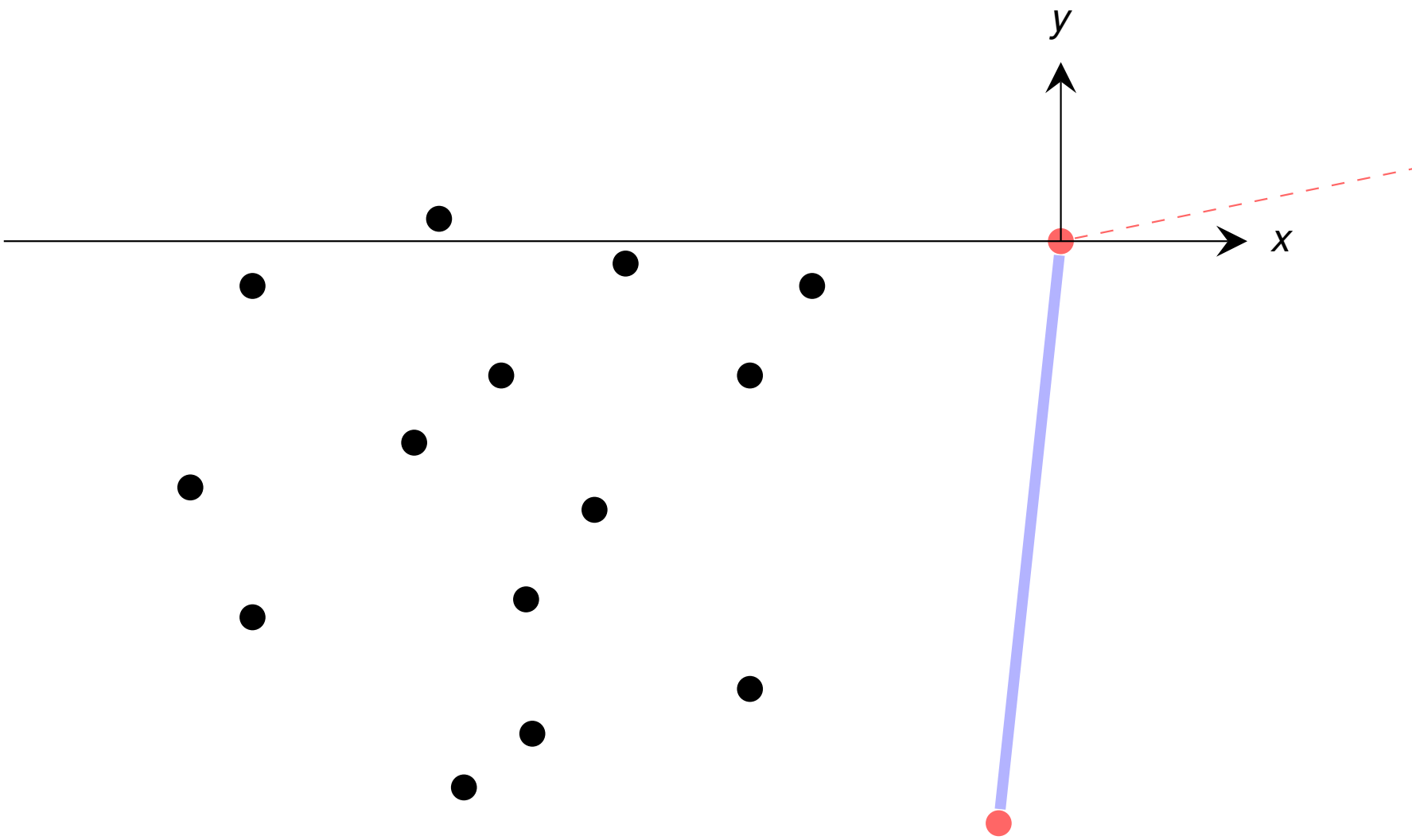
Execution of Jarvis' March



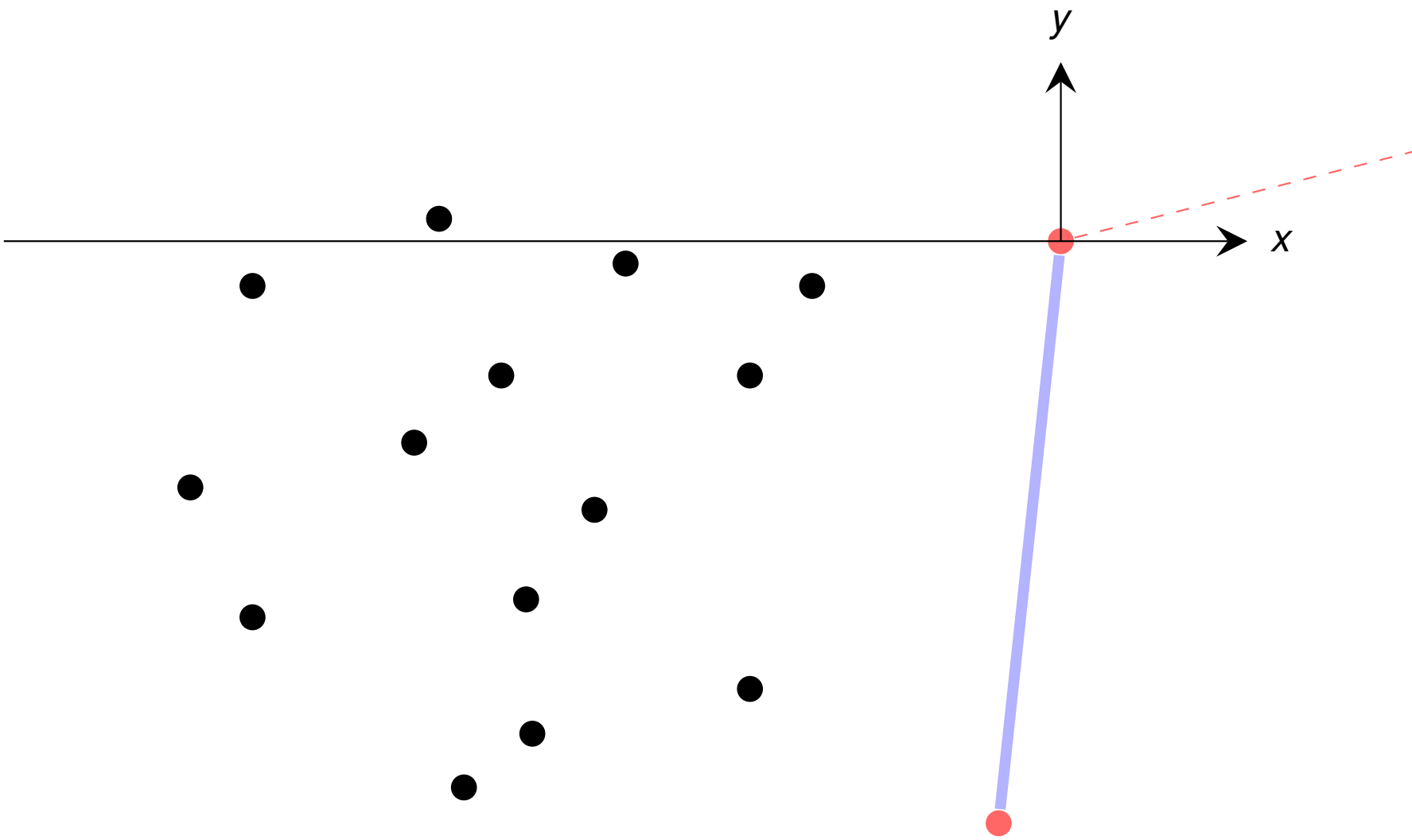
Execution of Jarvis' March



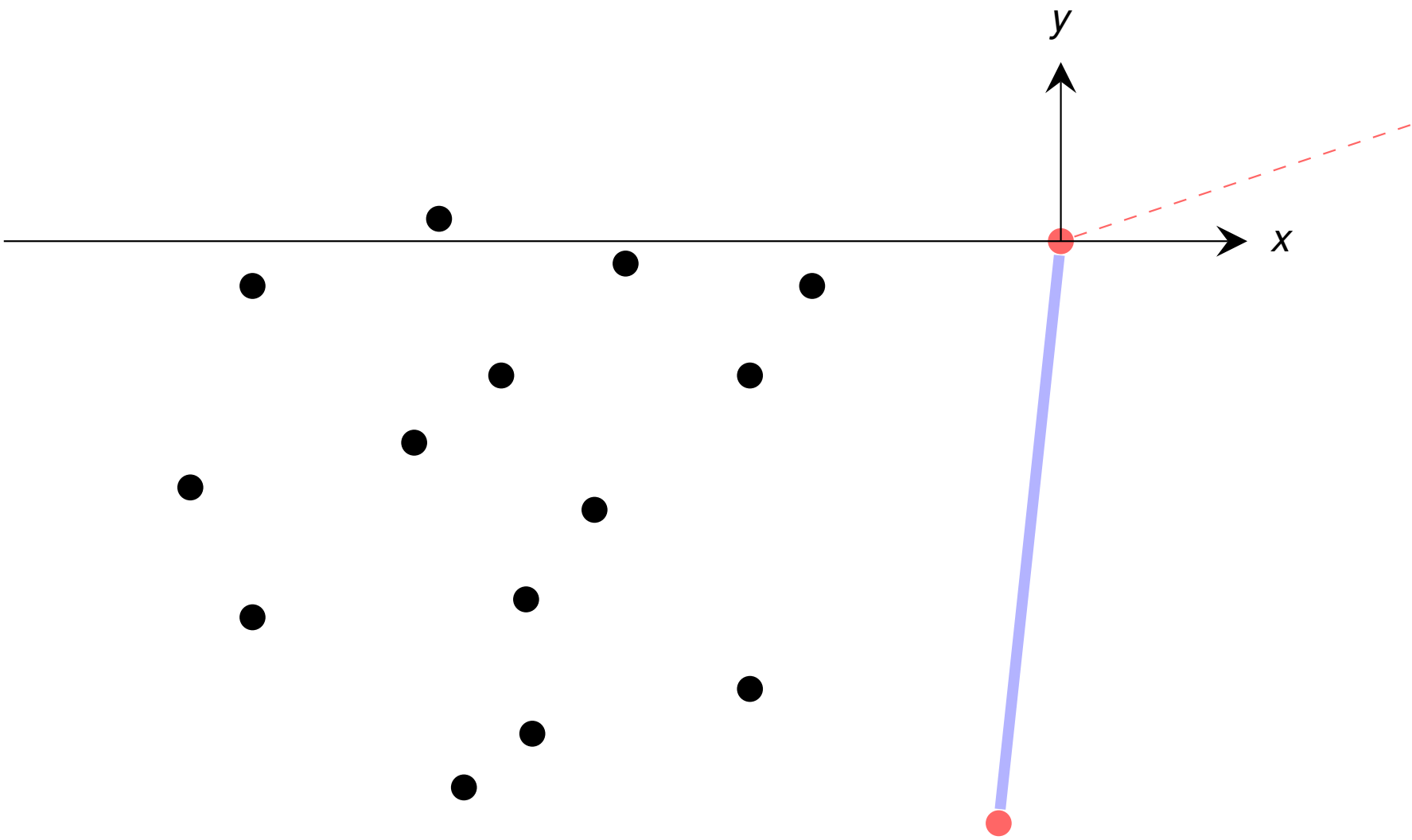
Execution of Jarvis' March



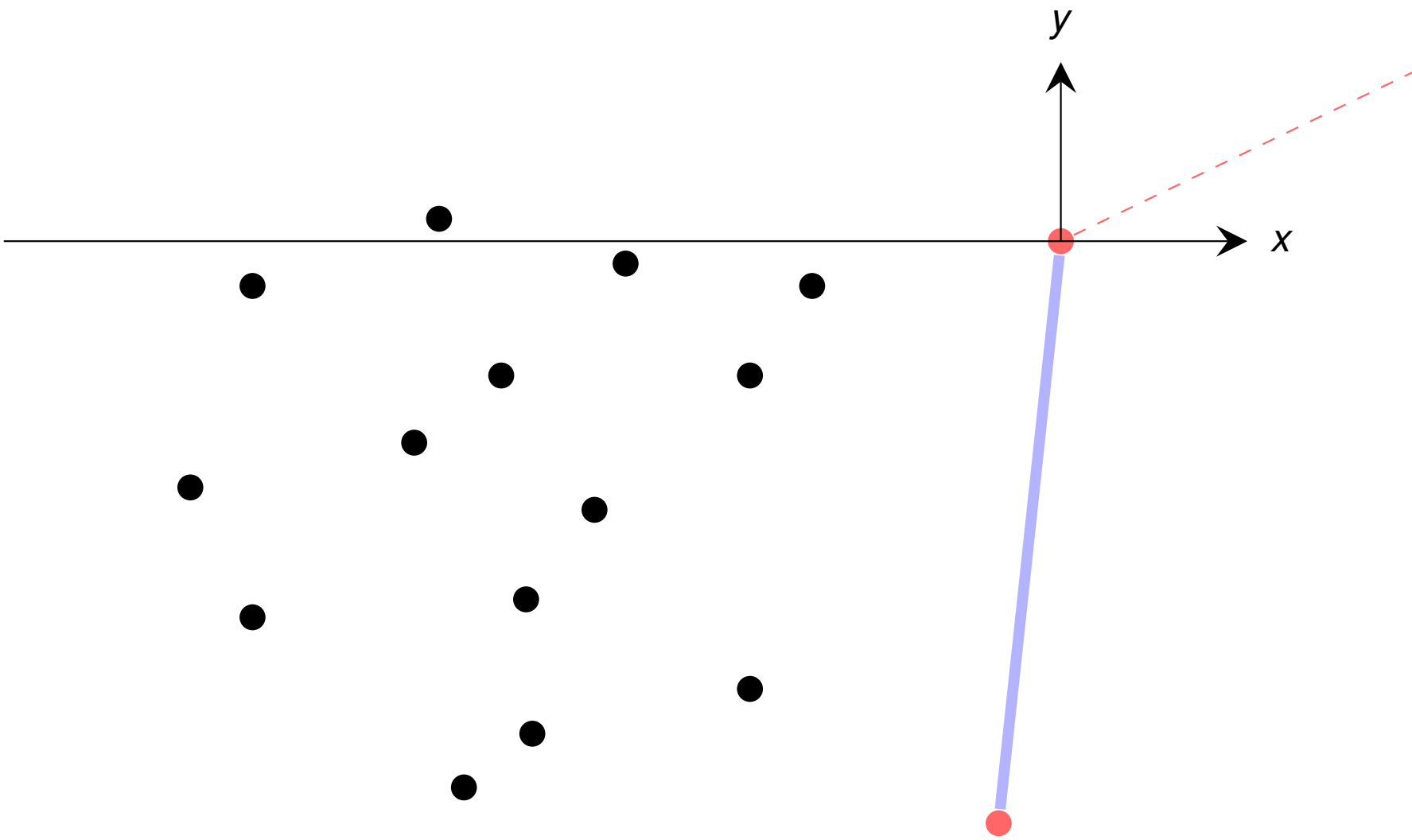
Execution of Jarvis' March



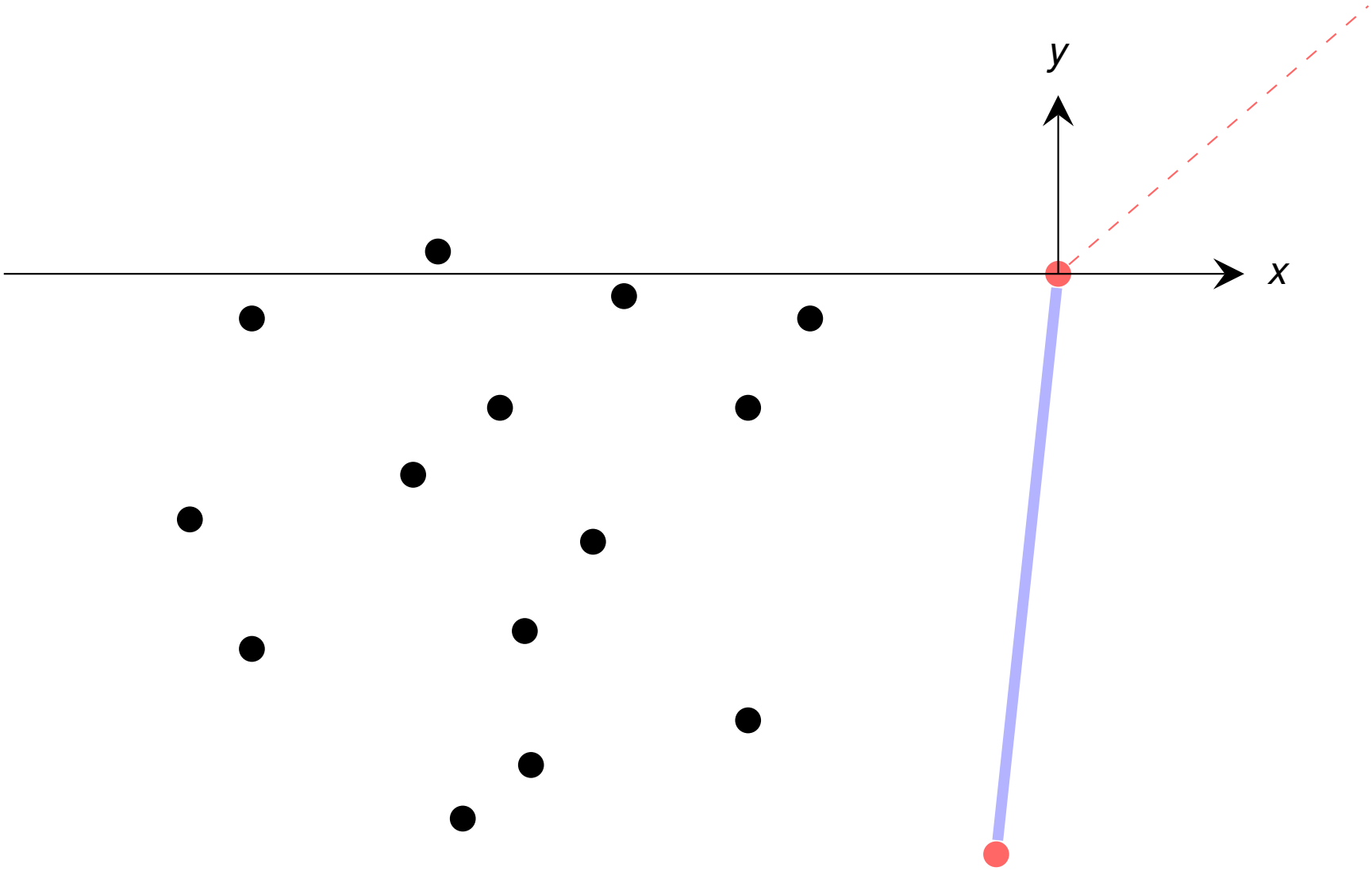
Execution of Jarvis' March



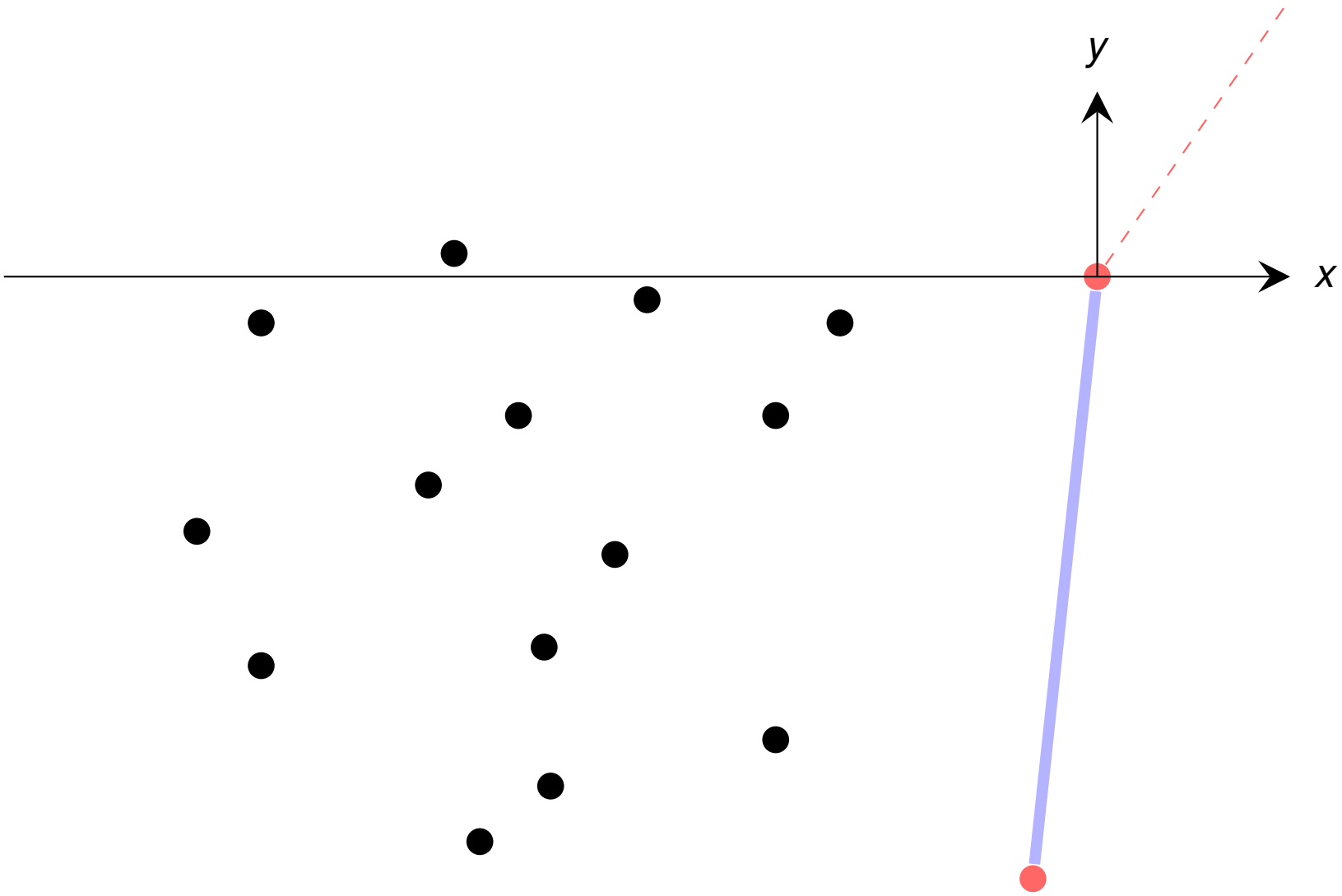
Execution of Jarvis' March



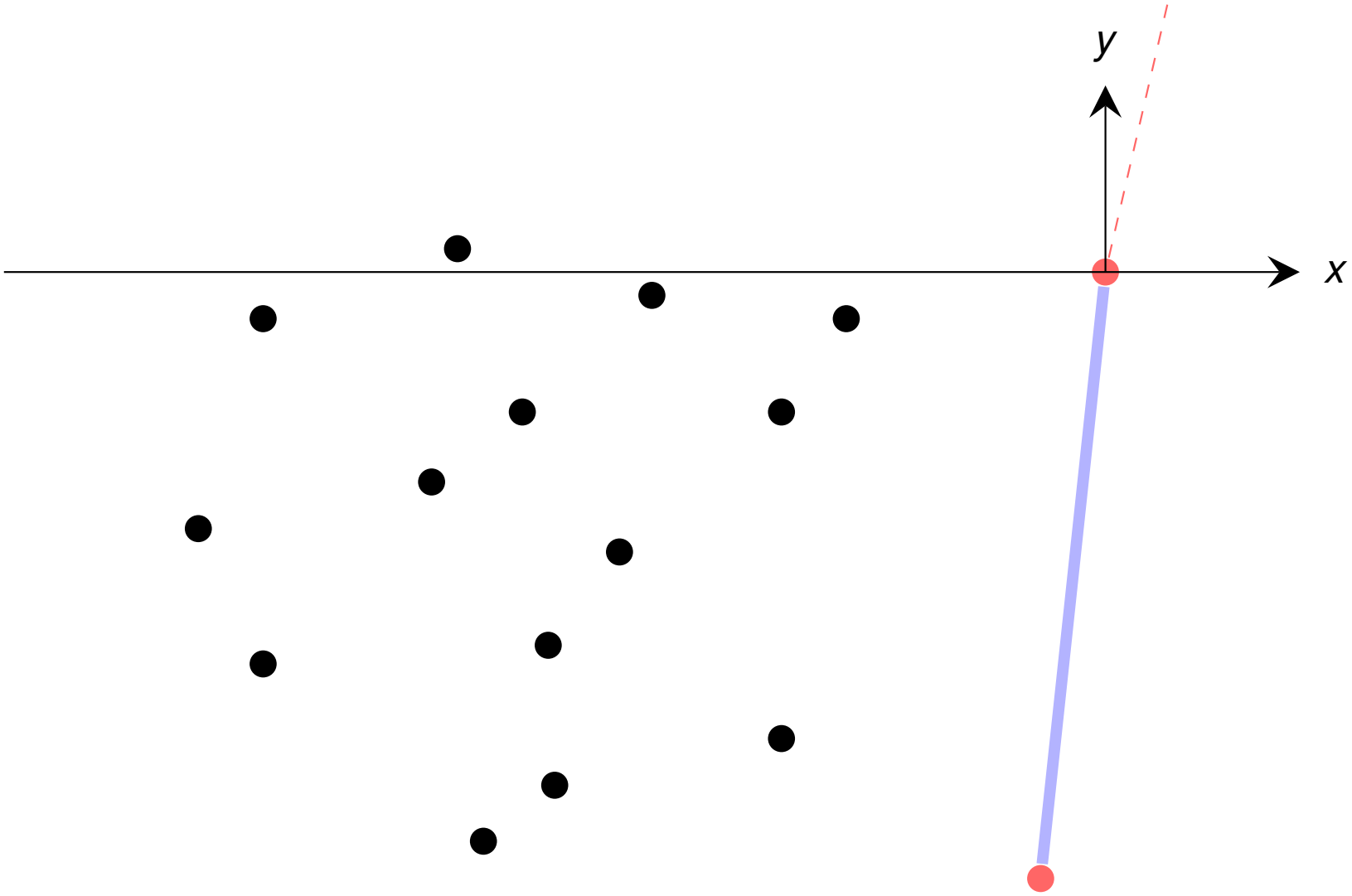
Execution of Jarvis' March



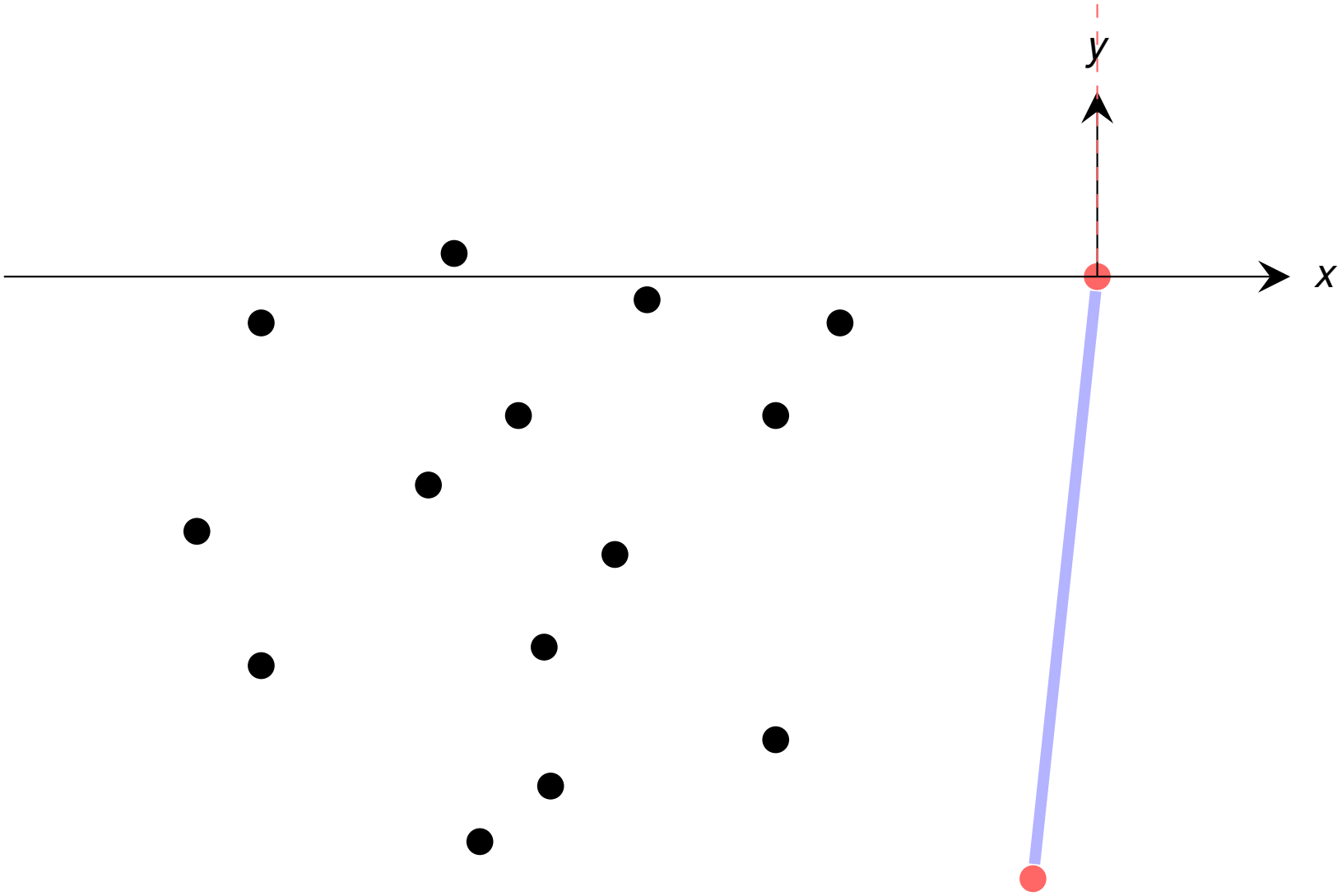
Execution of Jarvis' March



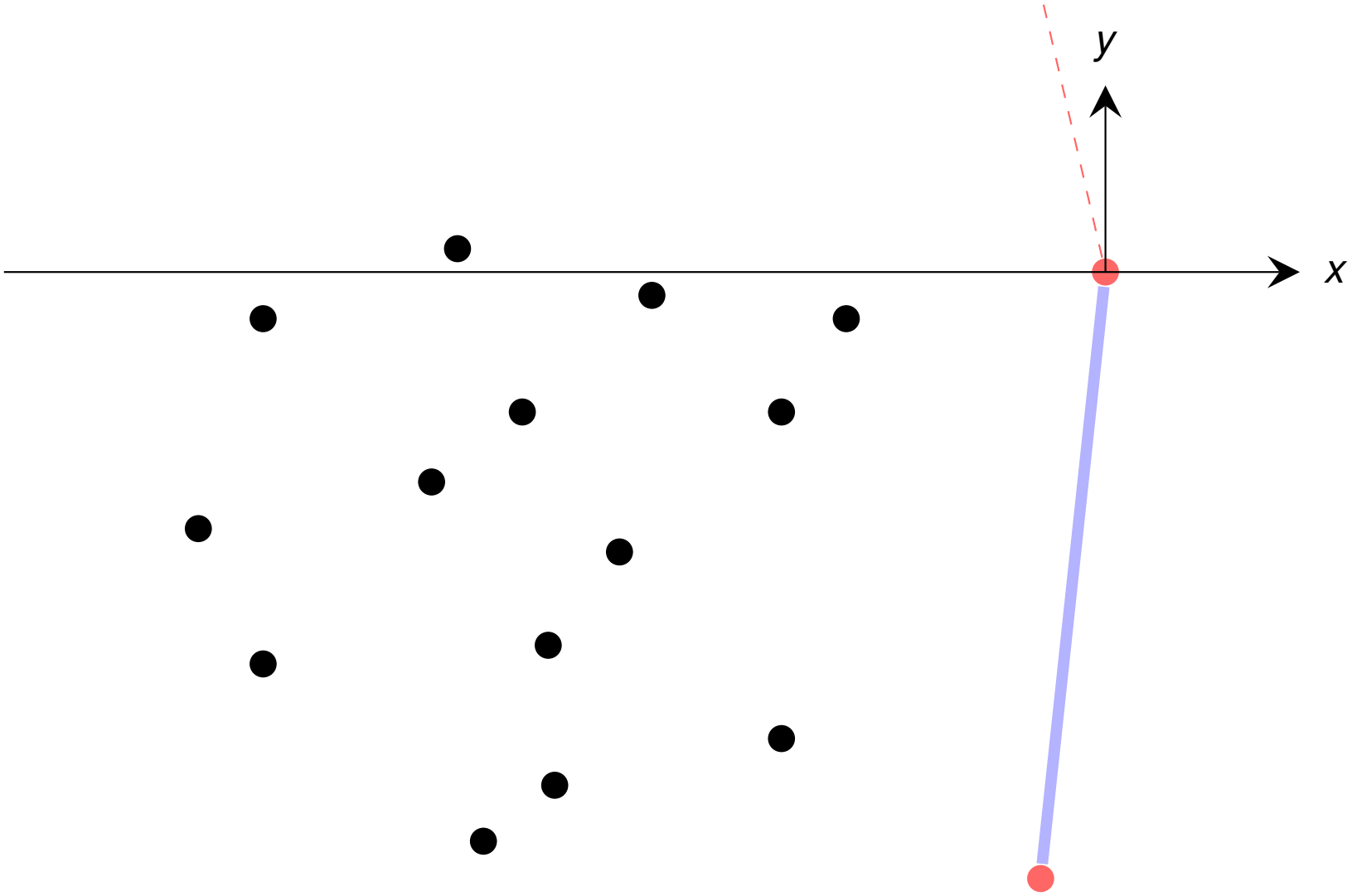
Execution of Jarvis' March



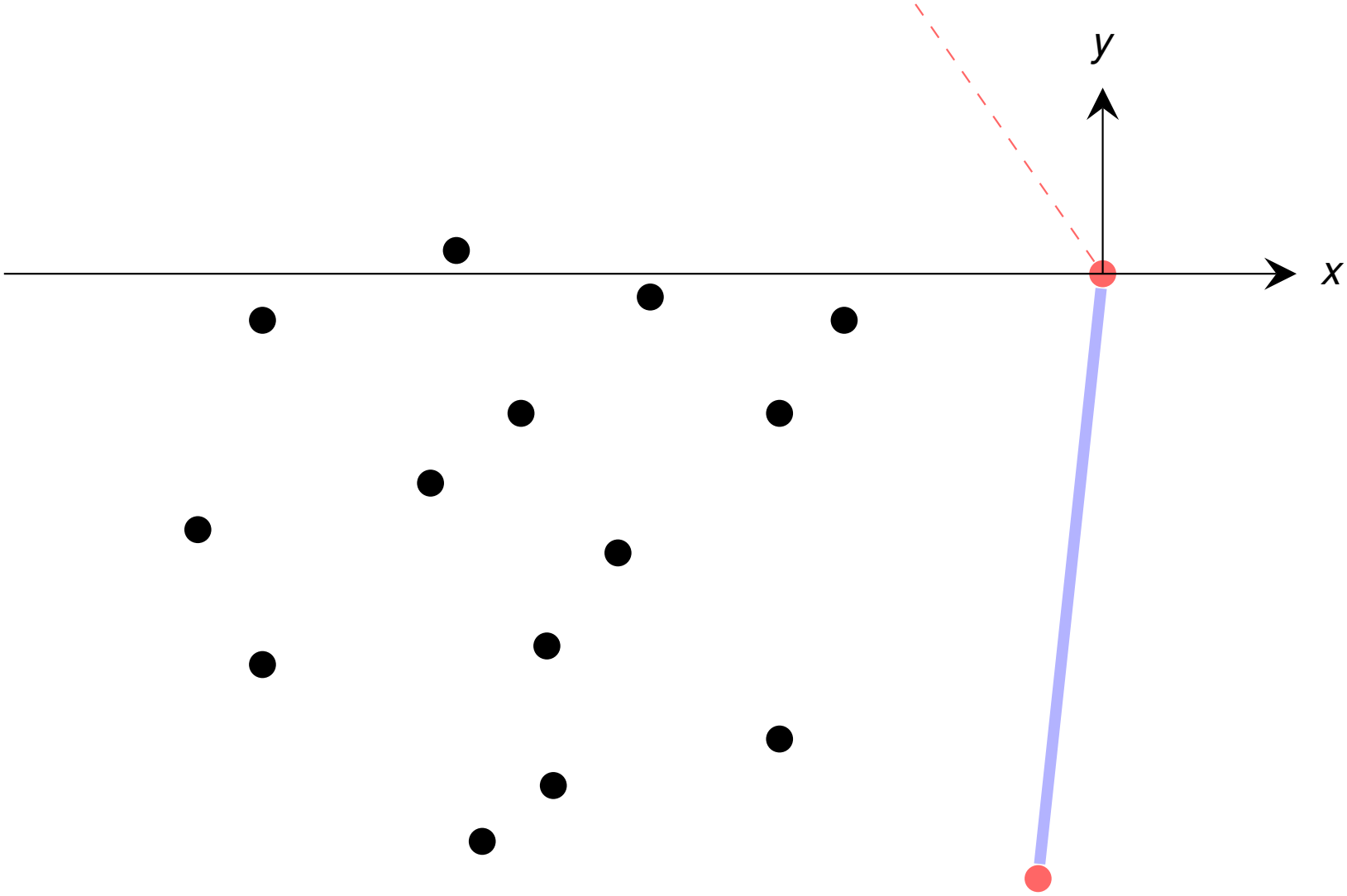
Execution of Jarvis' March



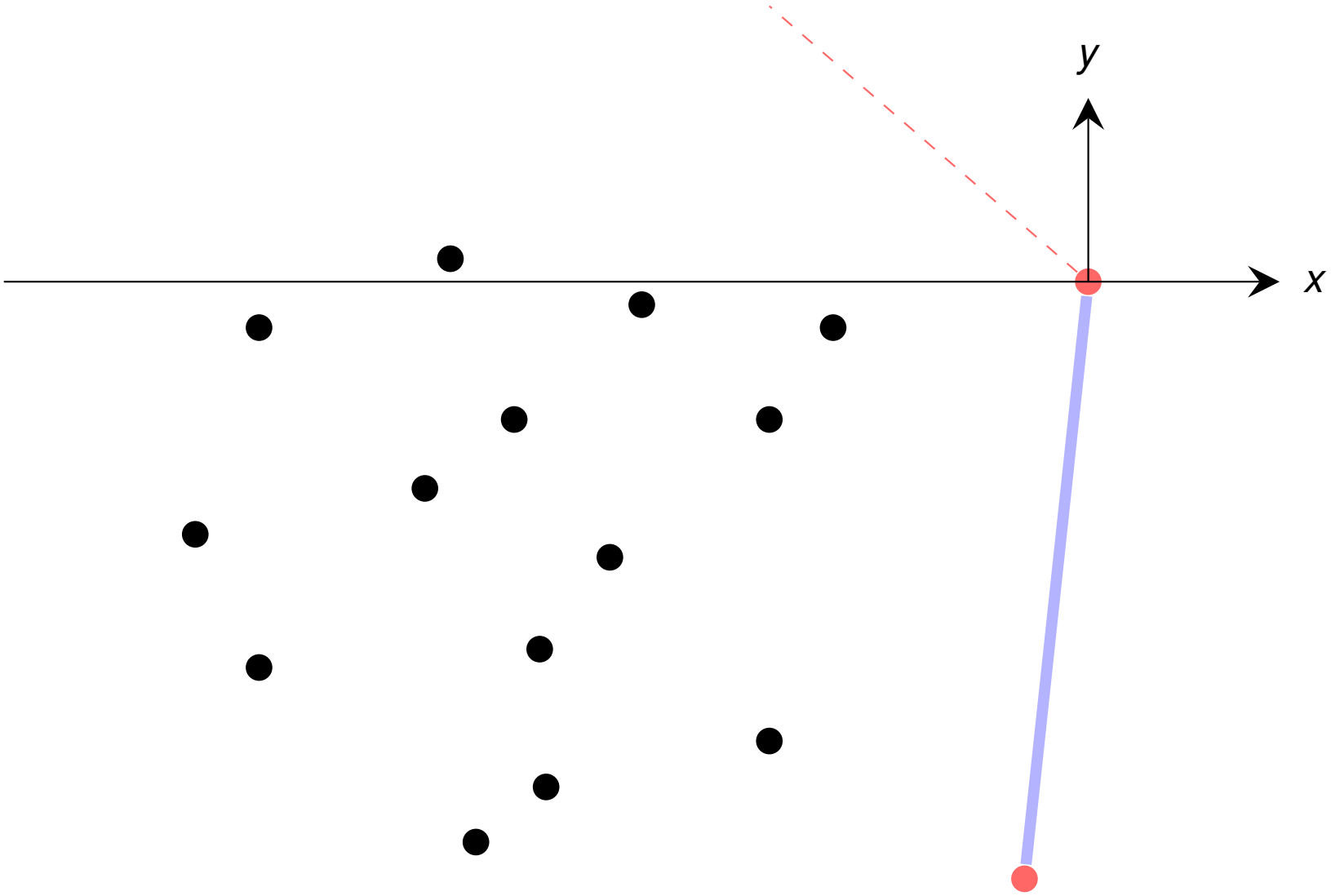
Execution of Jarvis' March



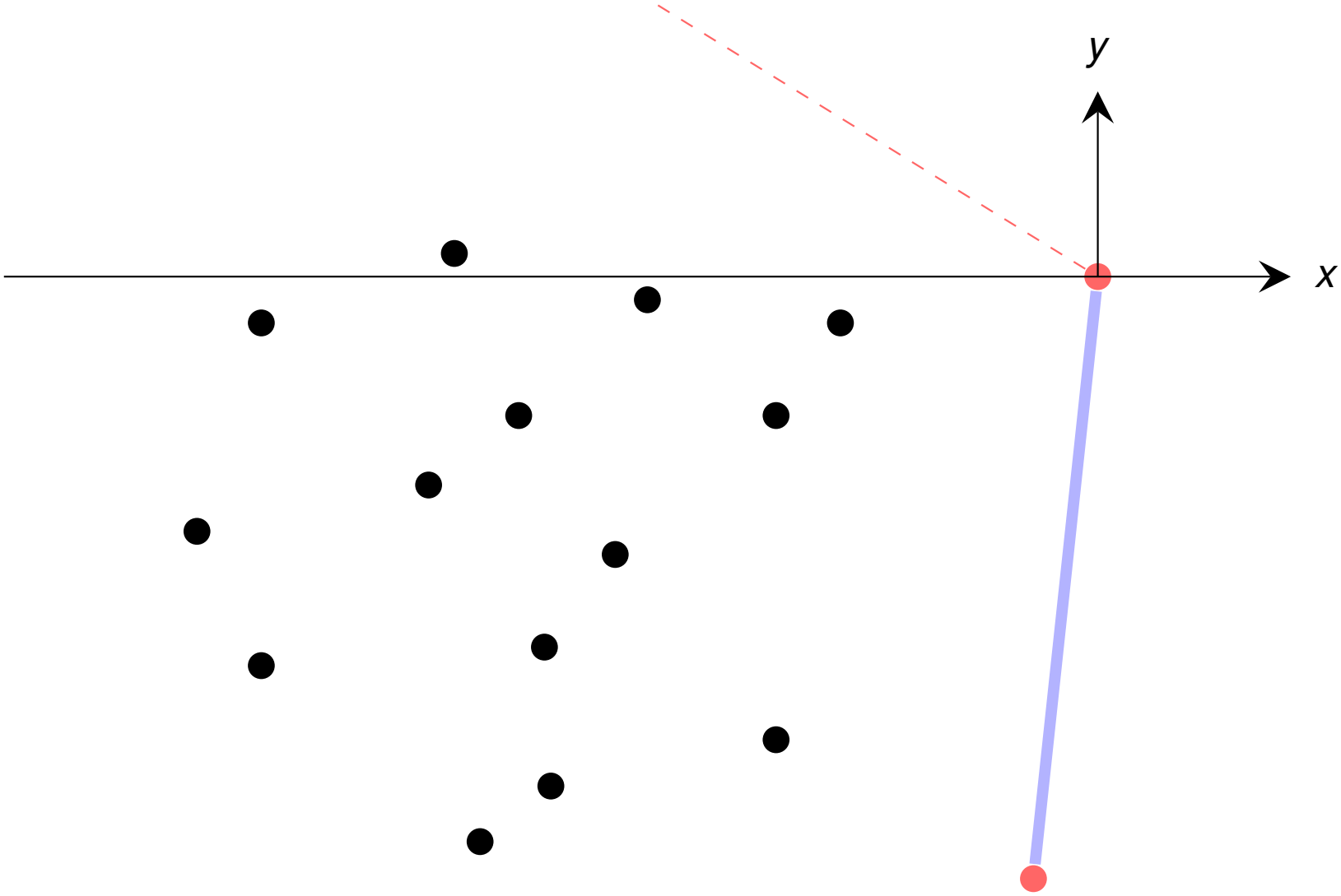
Execution of Jarvis' March



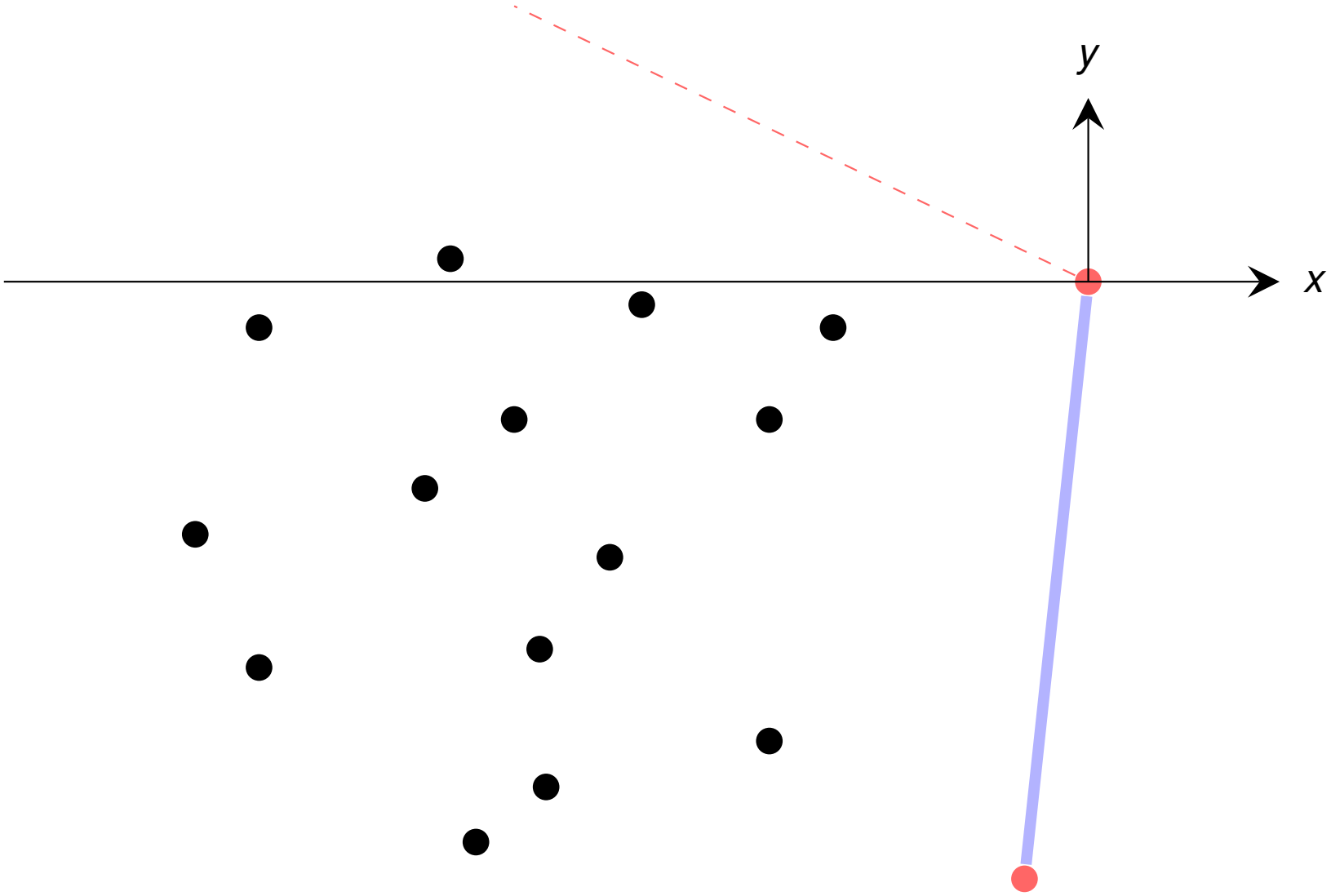
Execution of Jarvis' March



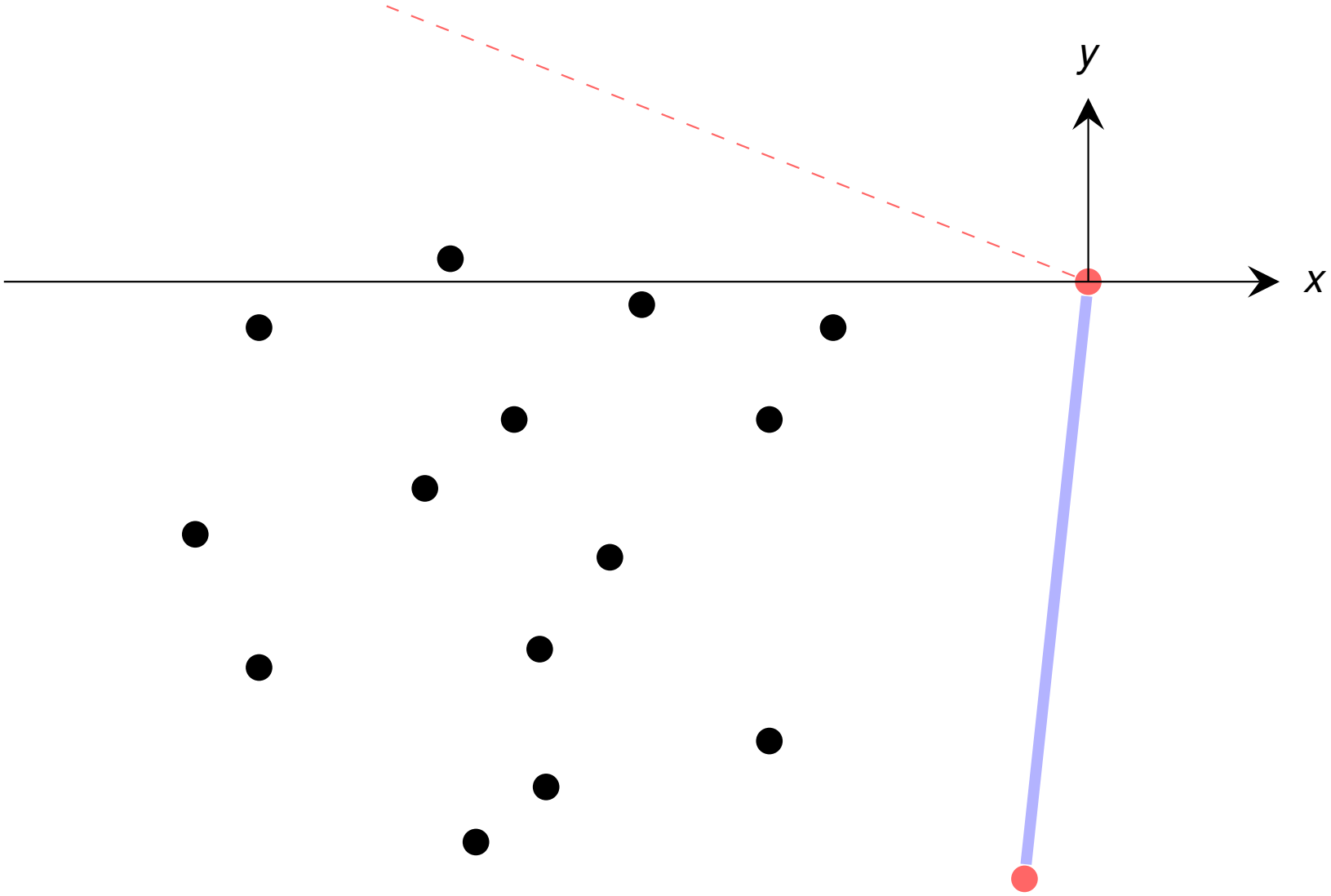
Execution of Jarvis' March



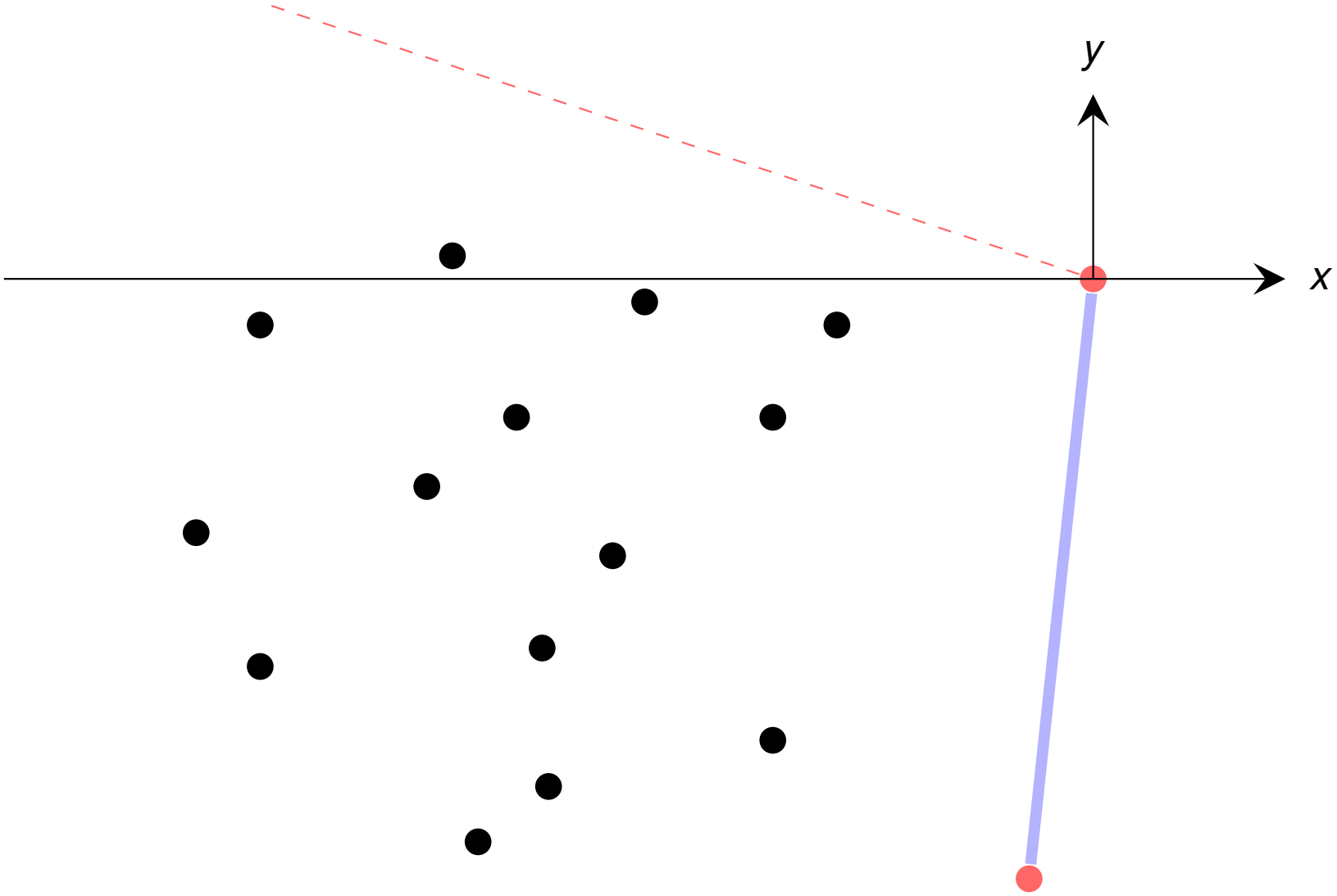
Execution of Jarvis' March



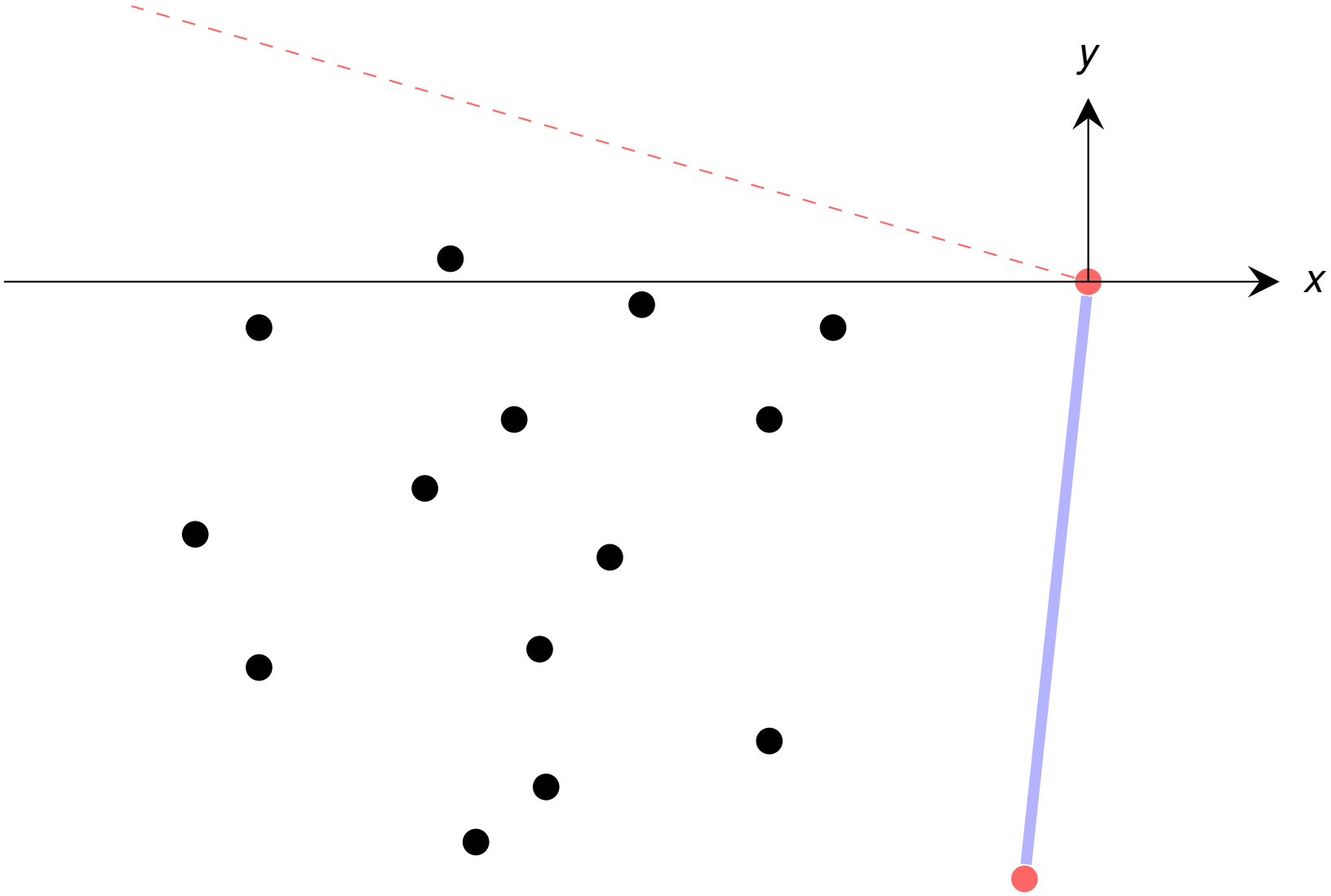
Execution of Jarvis' March



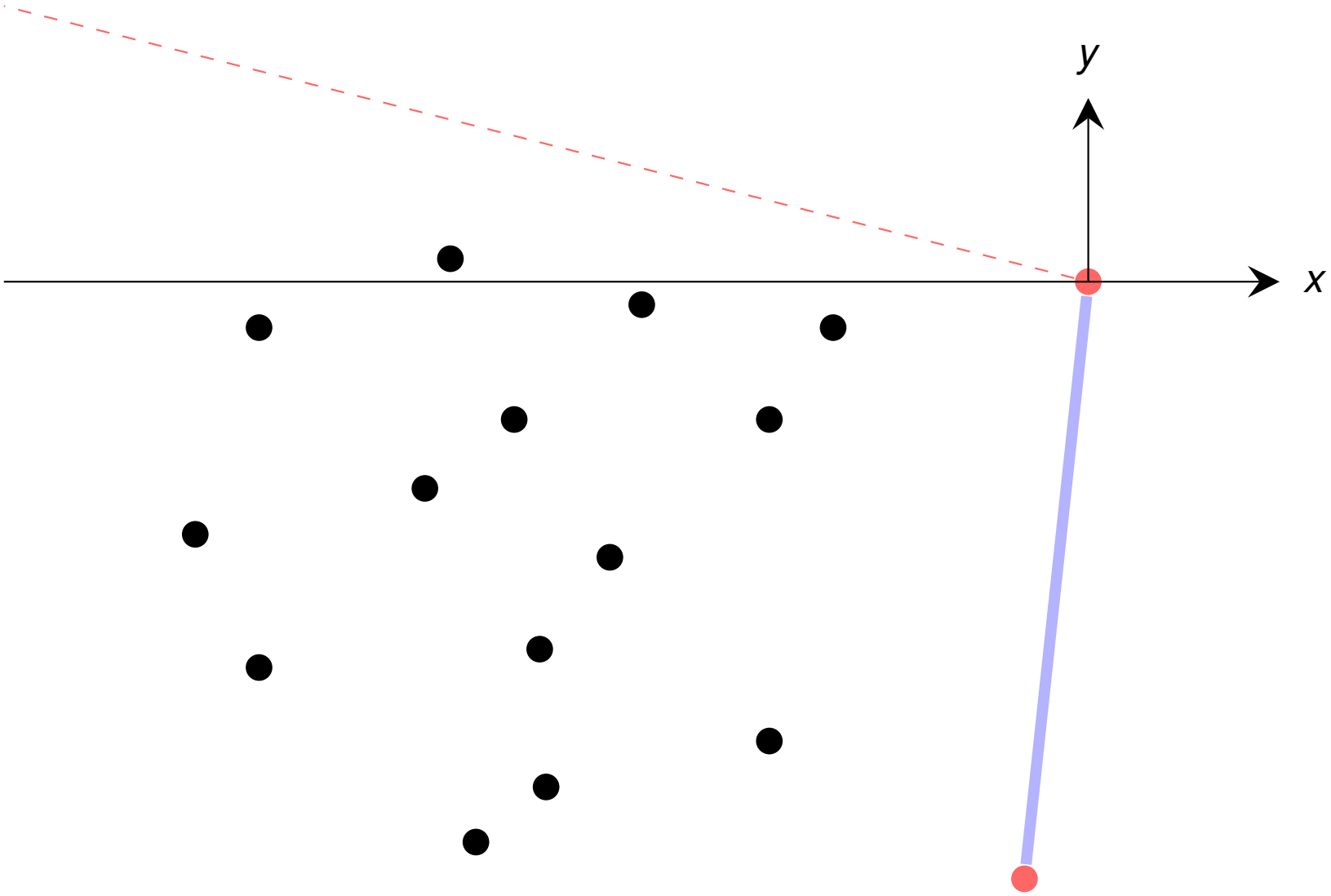
Execution of Jarvis' March



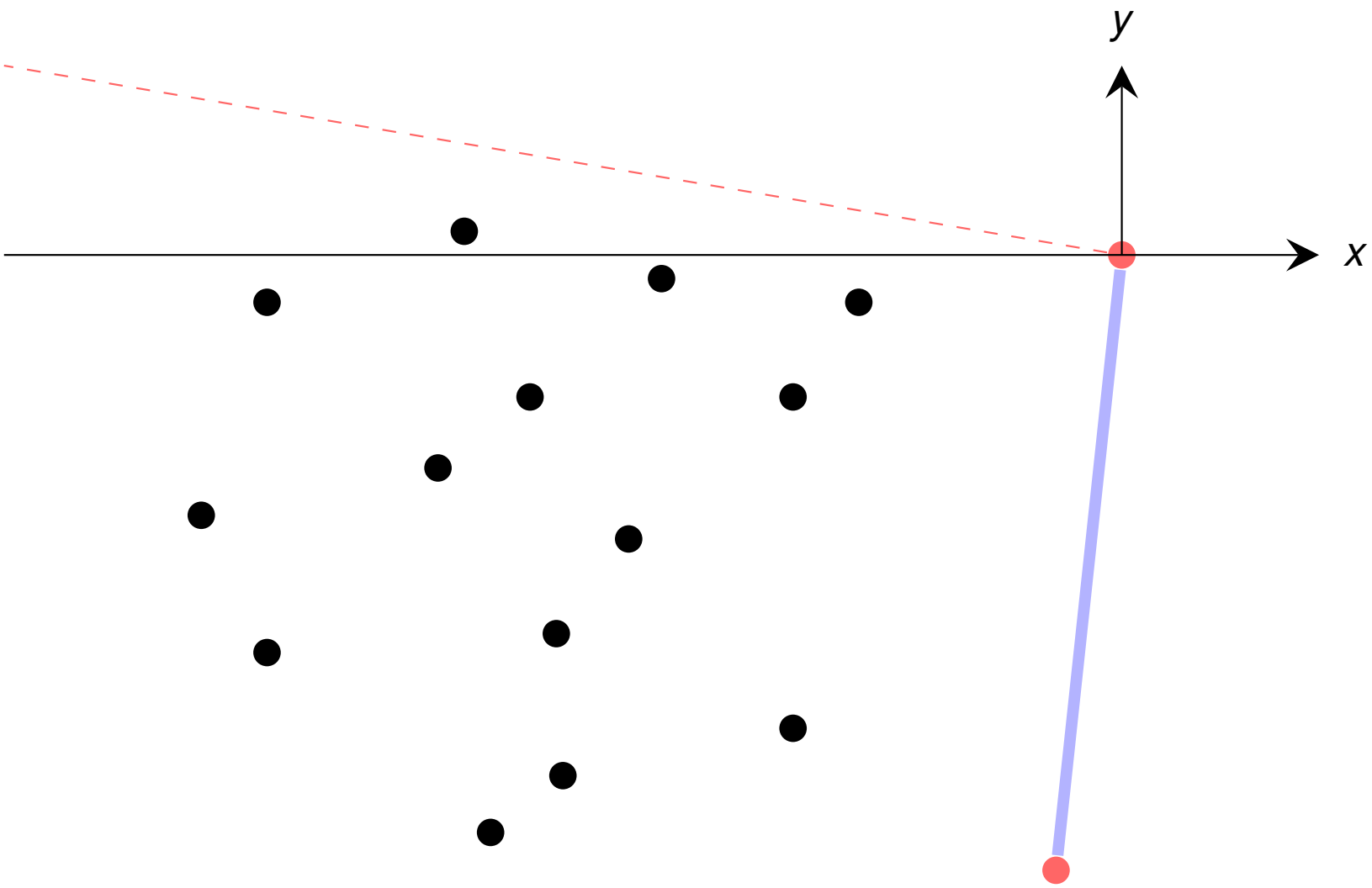
Execution of Jarvis' March



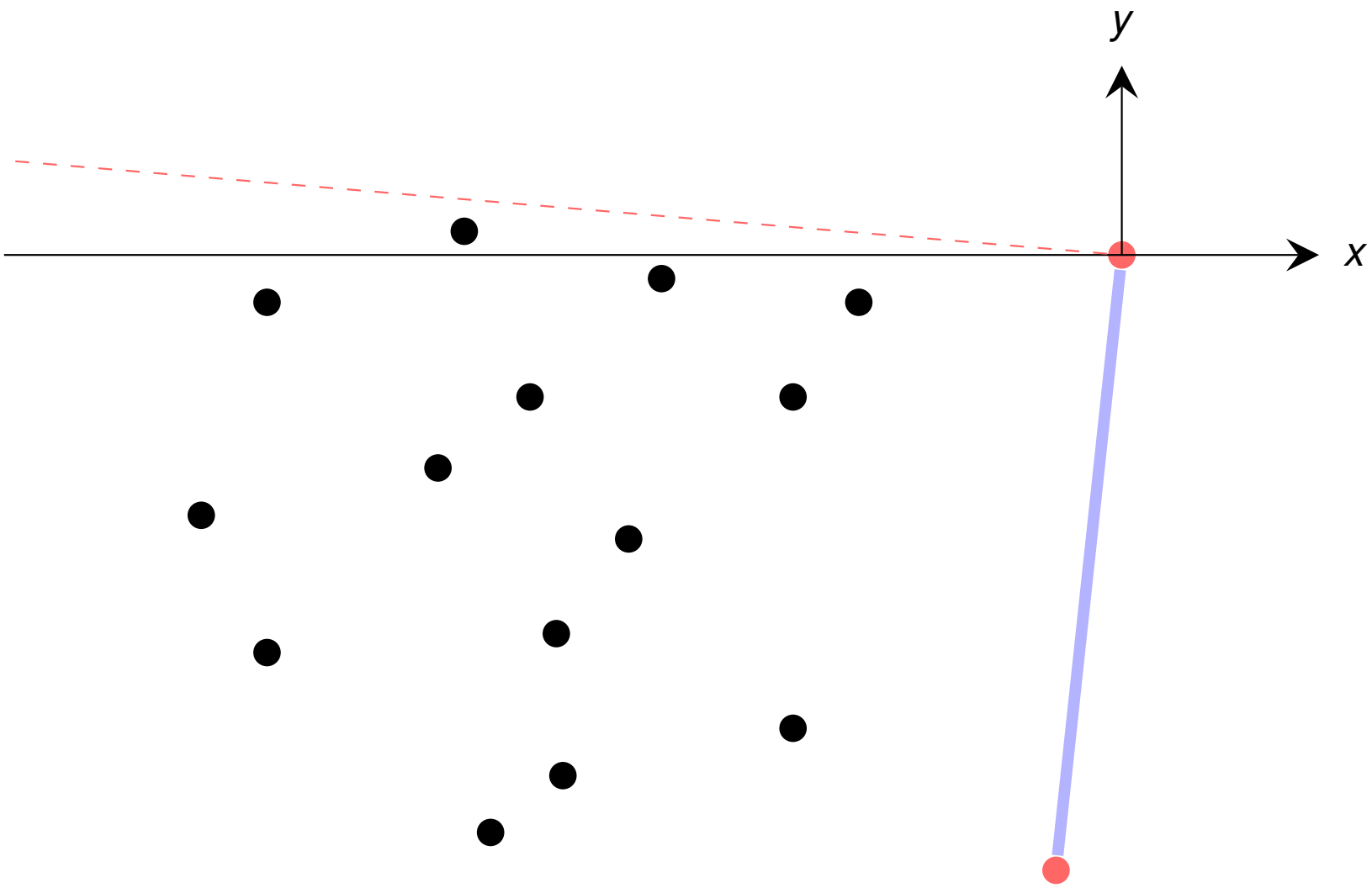
Execution of Jarvis' March



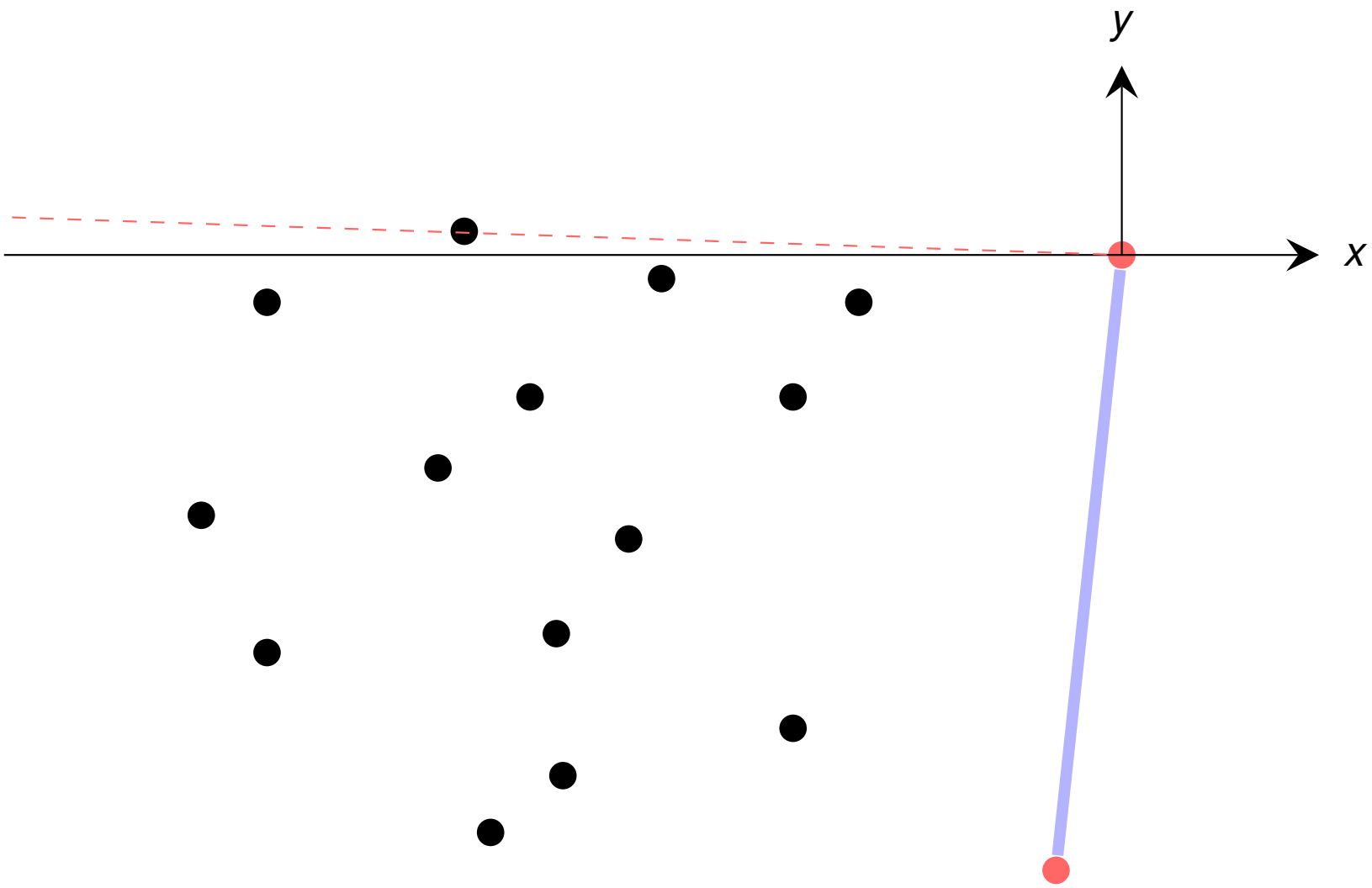
Execution of Jarvis' March



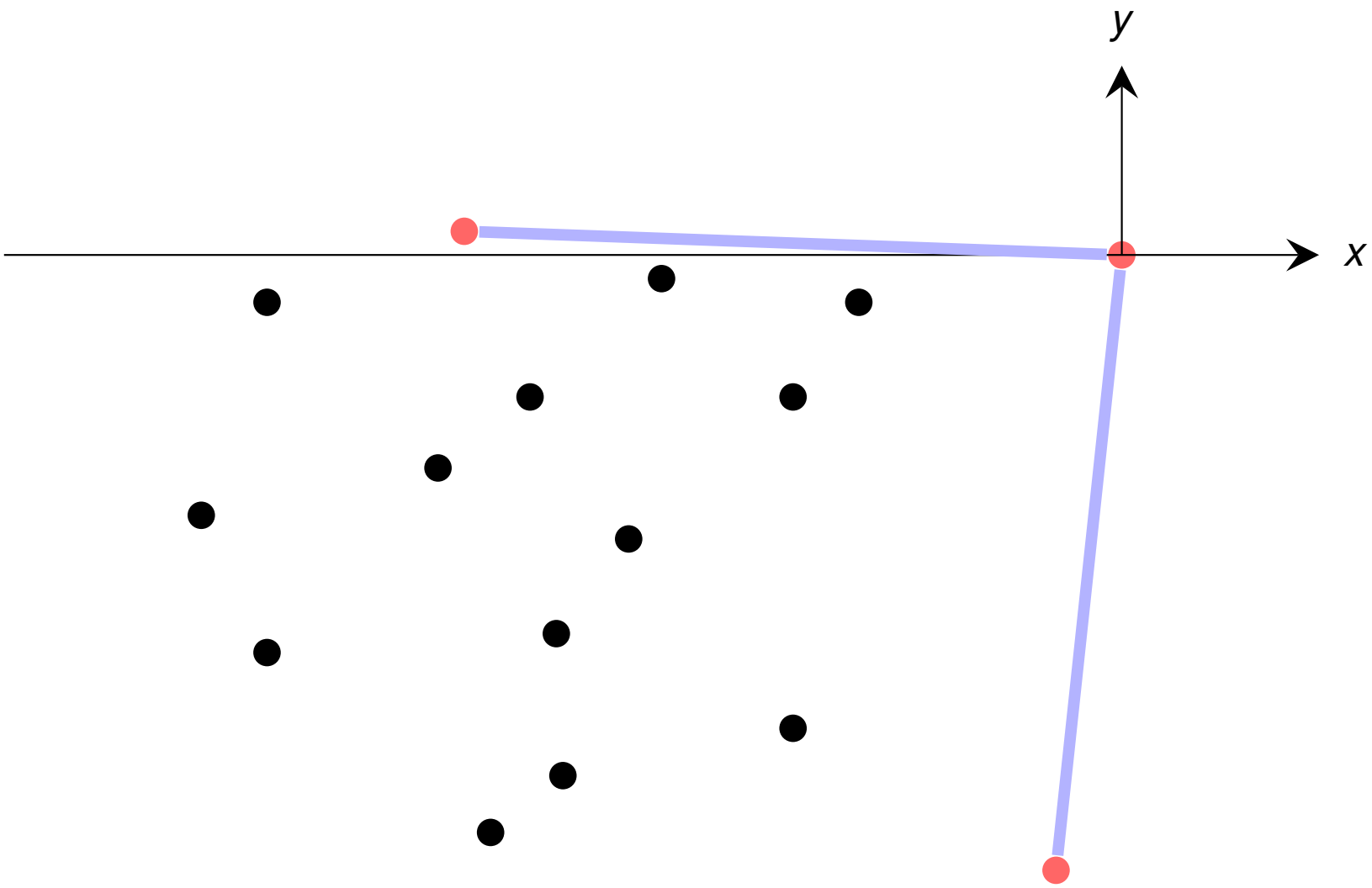
Execution of Jarvis' March



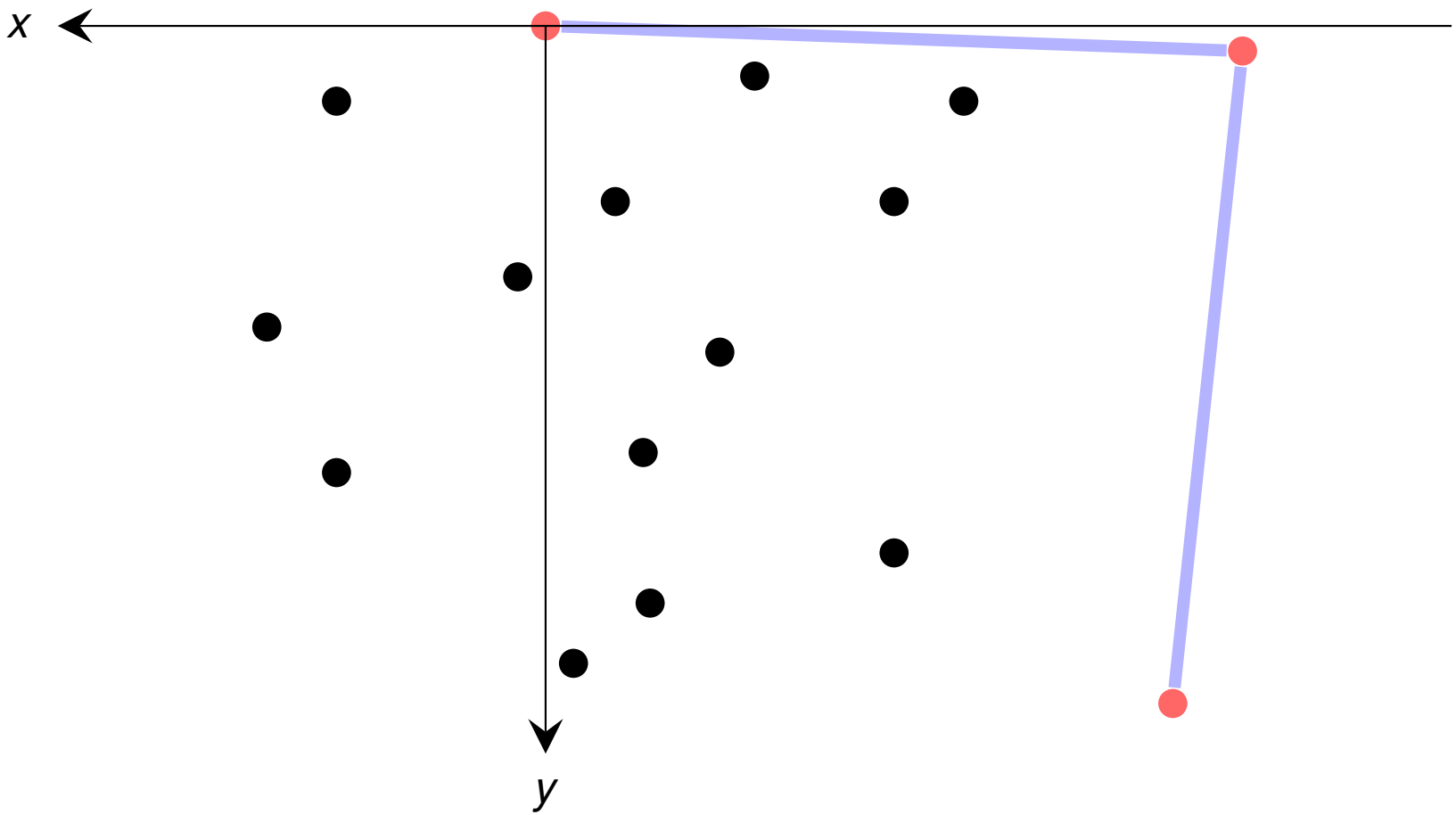
Execution of Jarvis' March



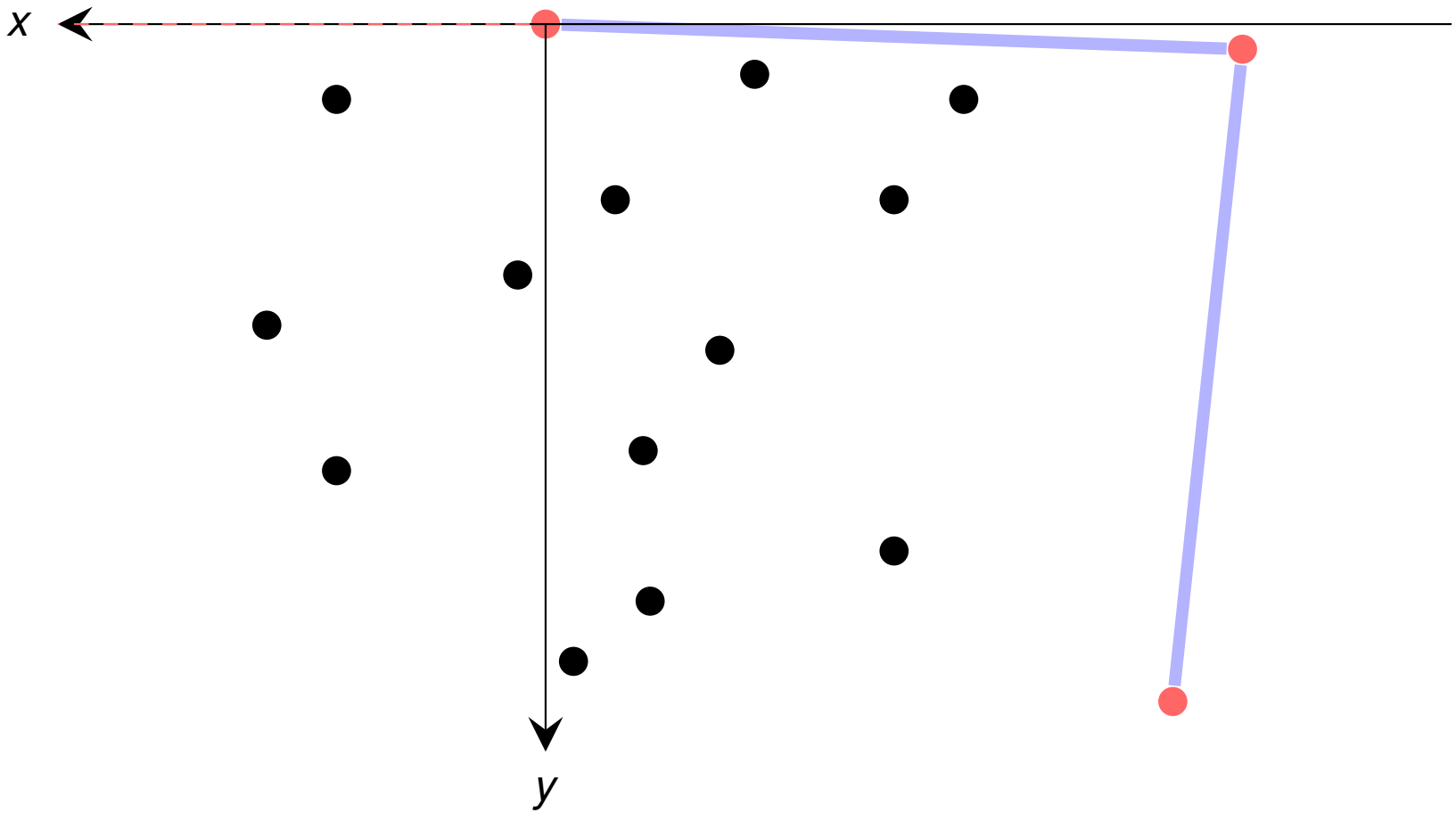
Execution of Jarvis' March



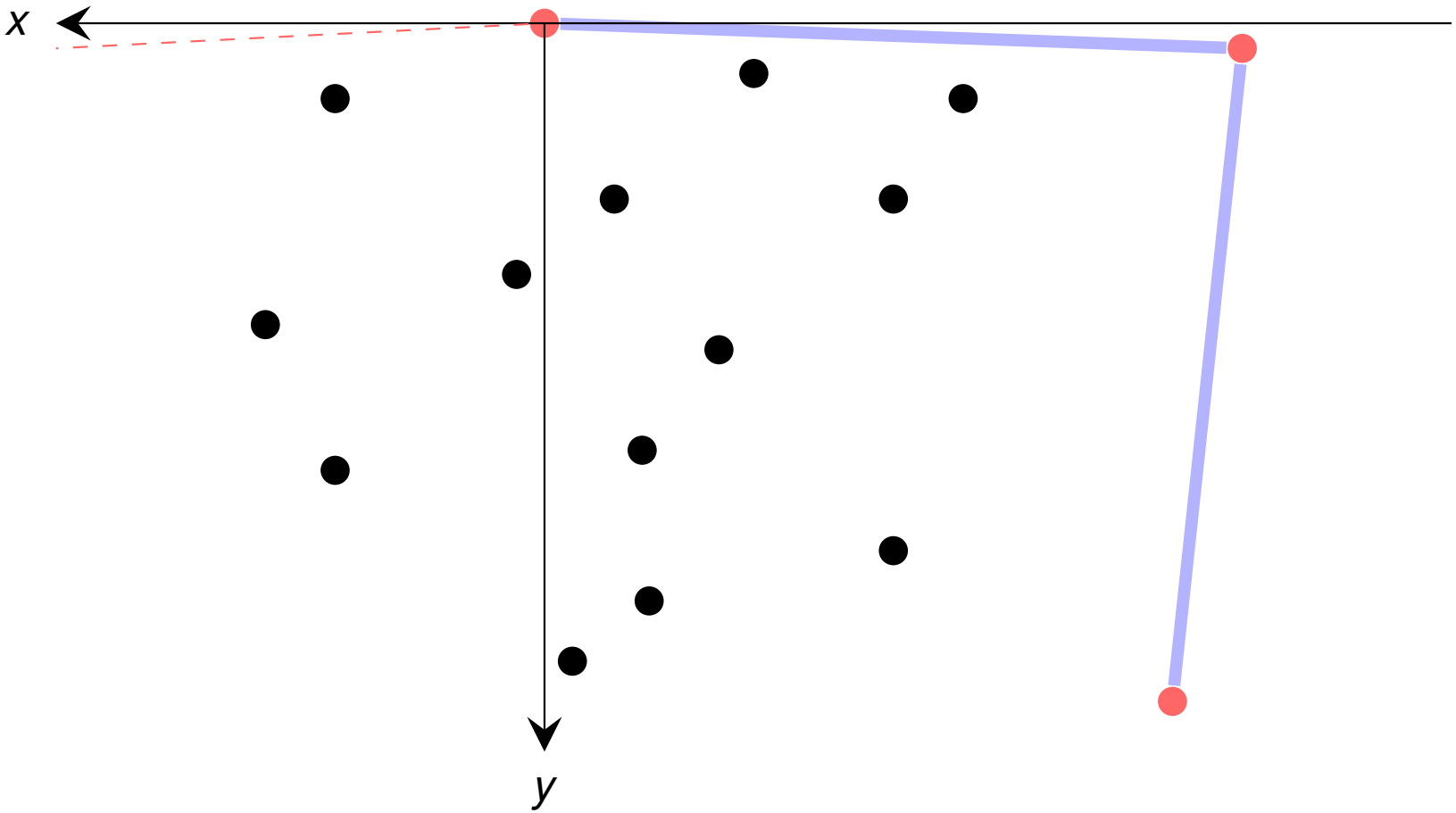
Execution of Jarvis' March



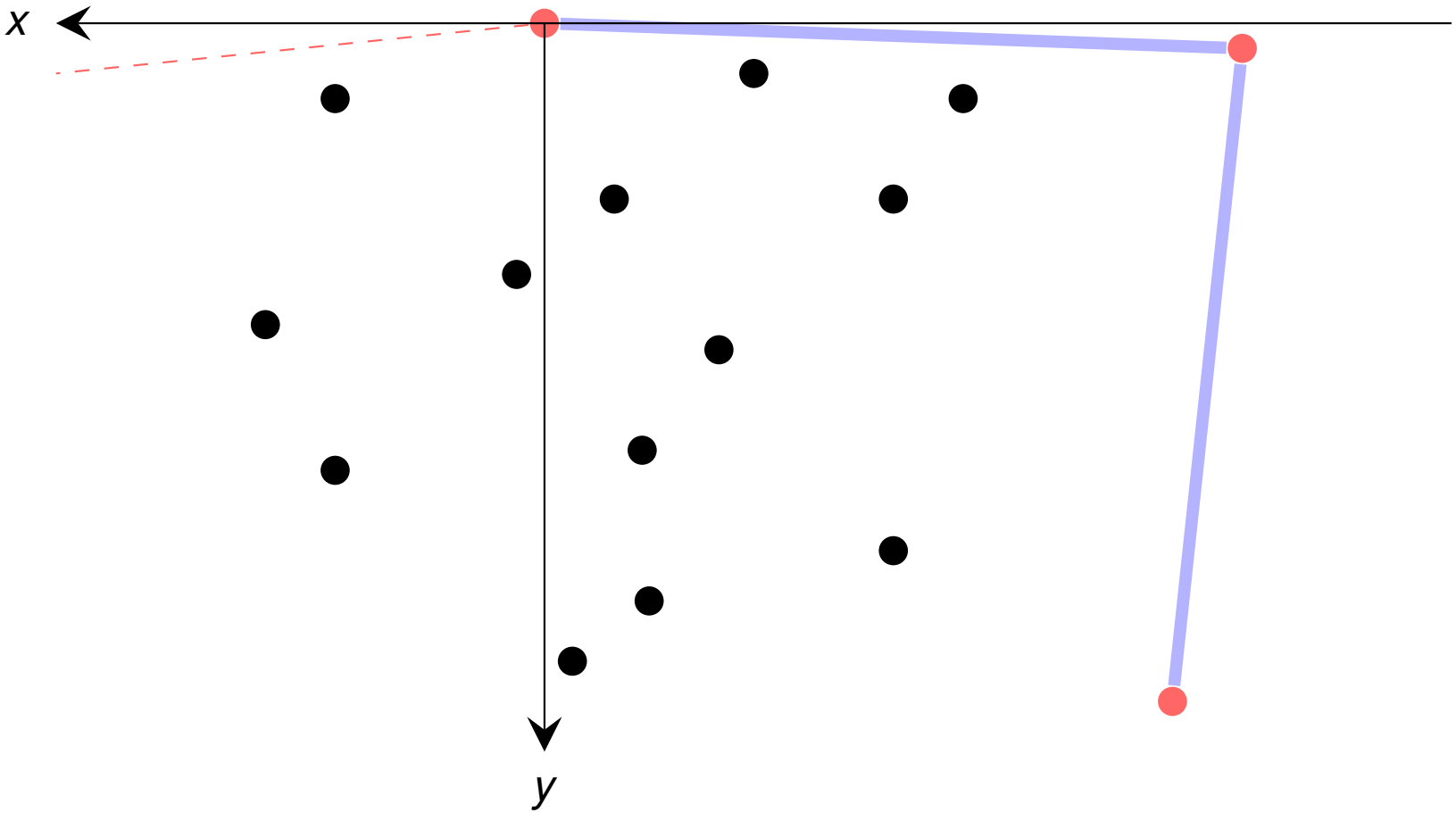
Execution of Jarvis' March



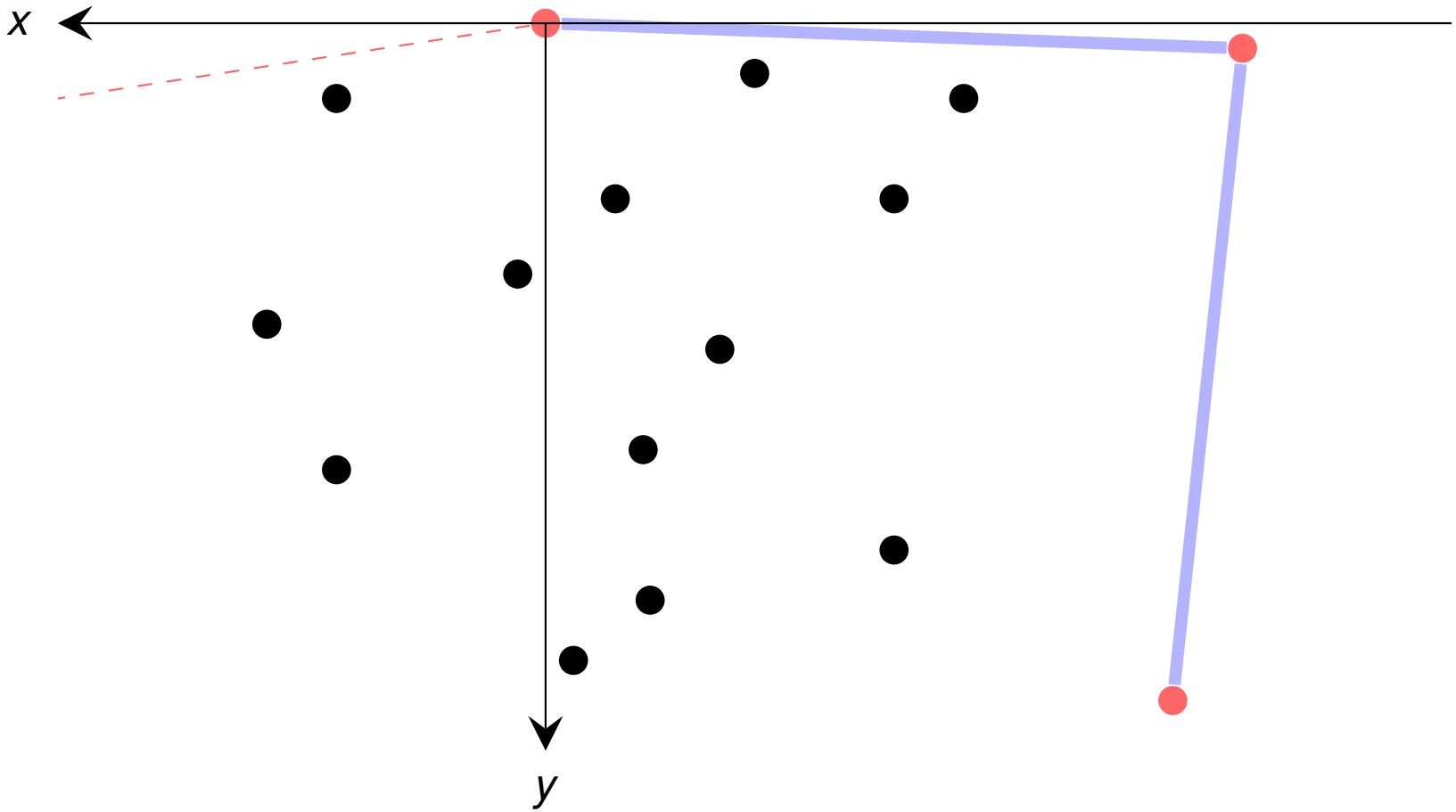
Execution of Jarvis' March



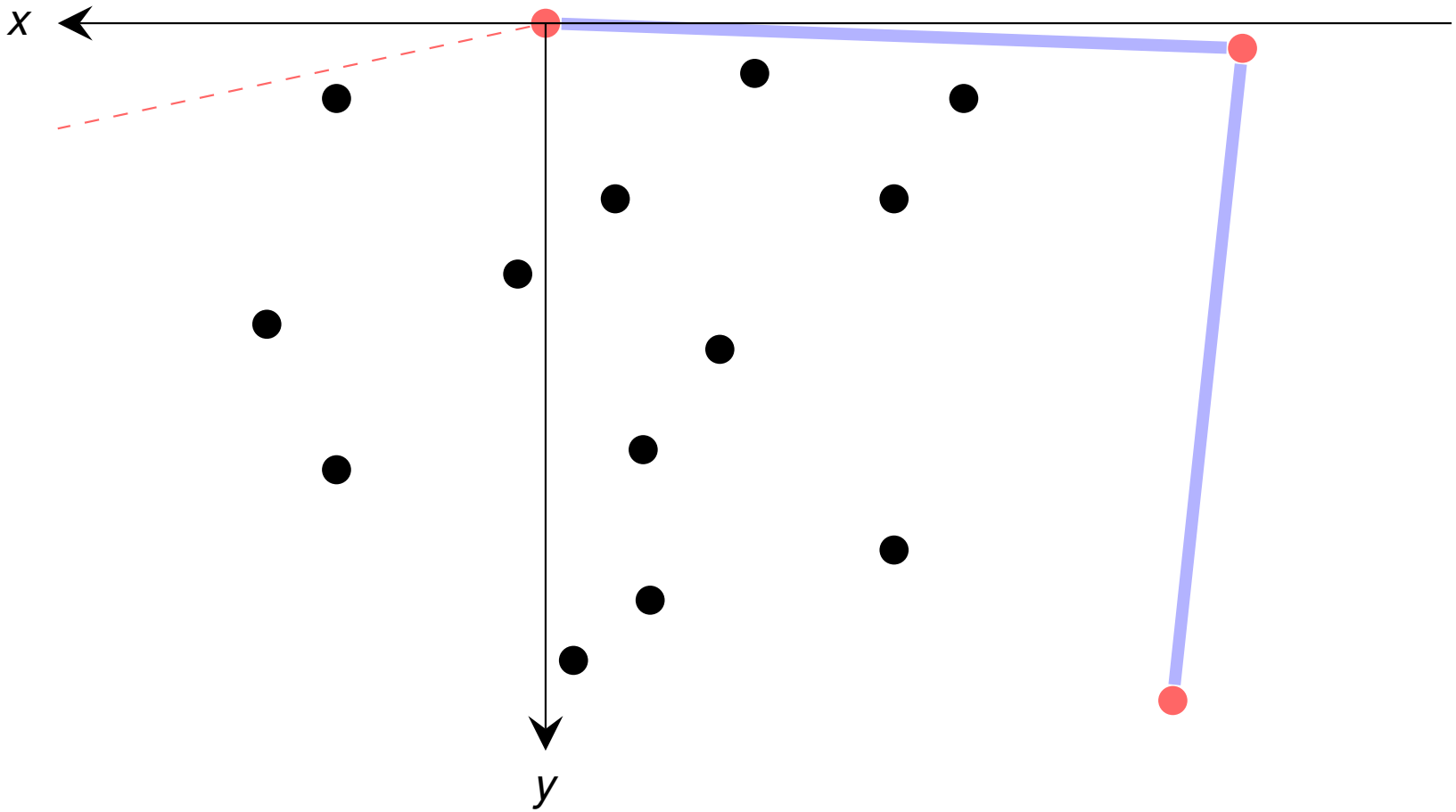
Execution of Jarvis' March



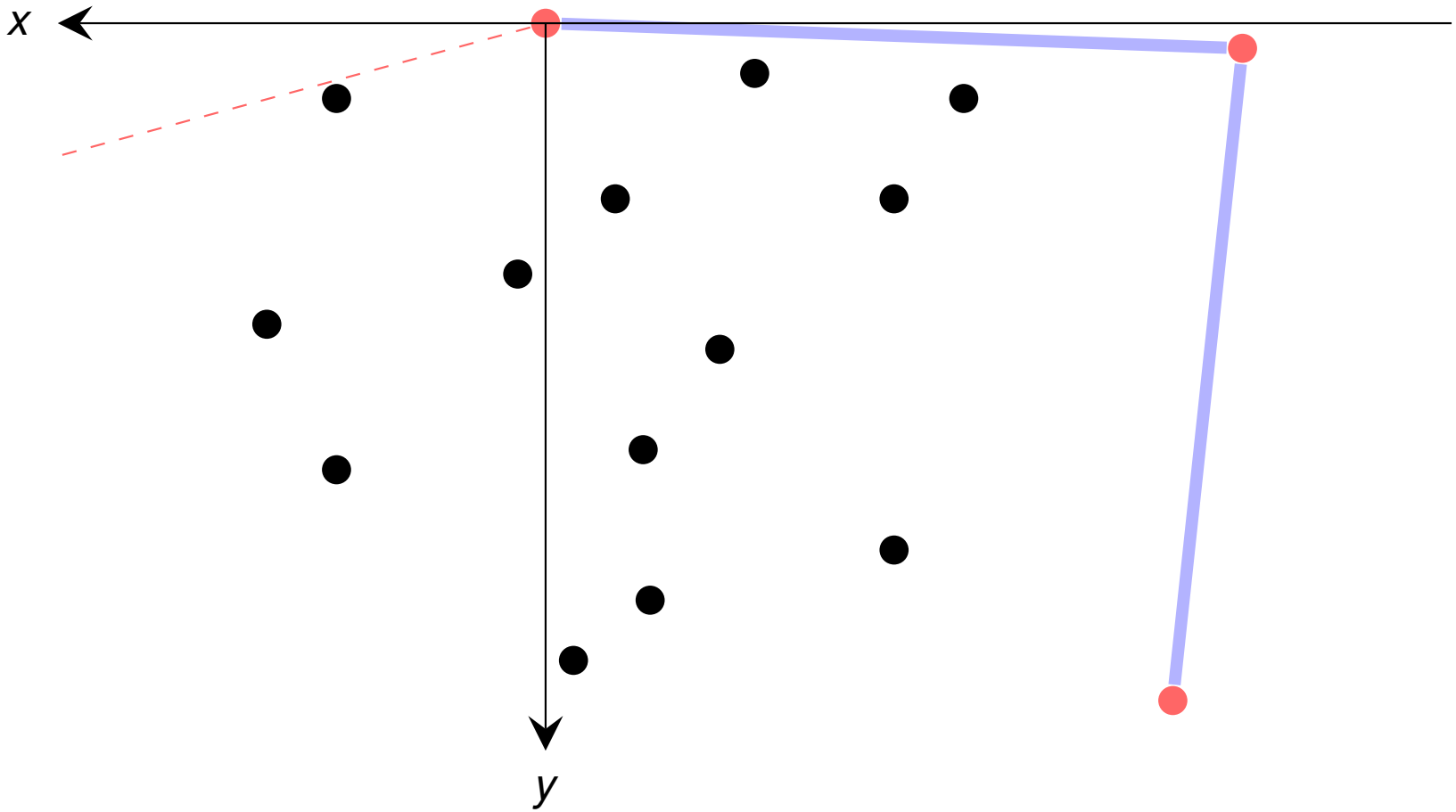
Execution of Jarvis' March



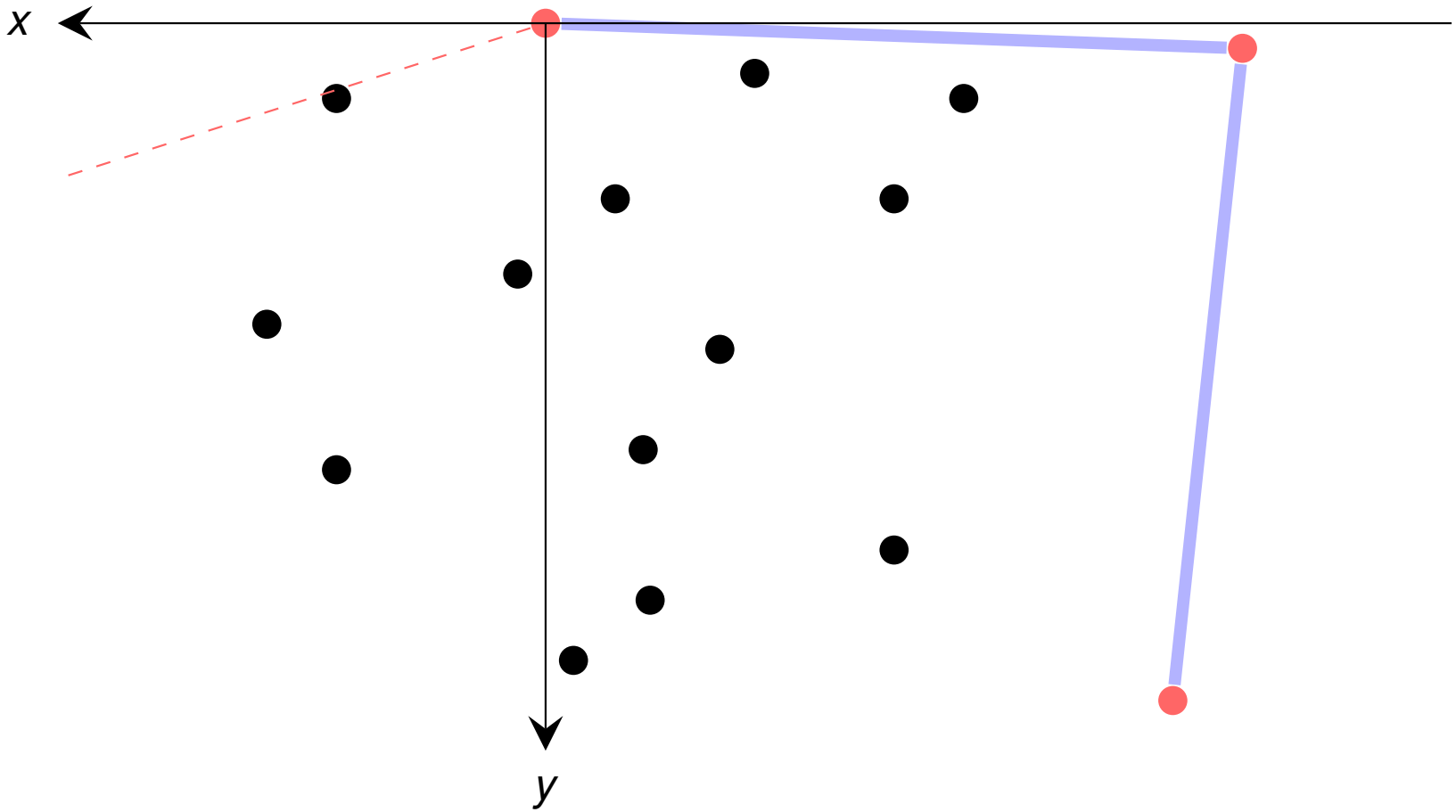
Execution of Jarvis' March



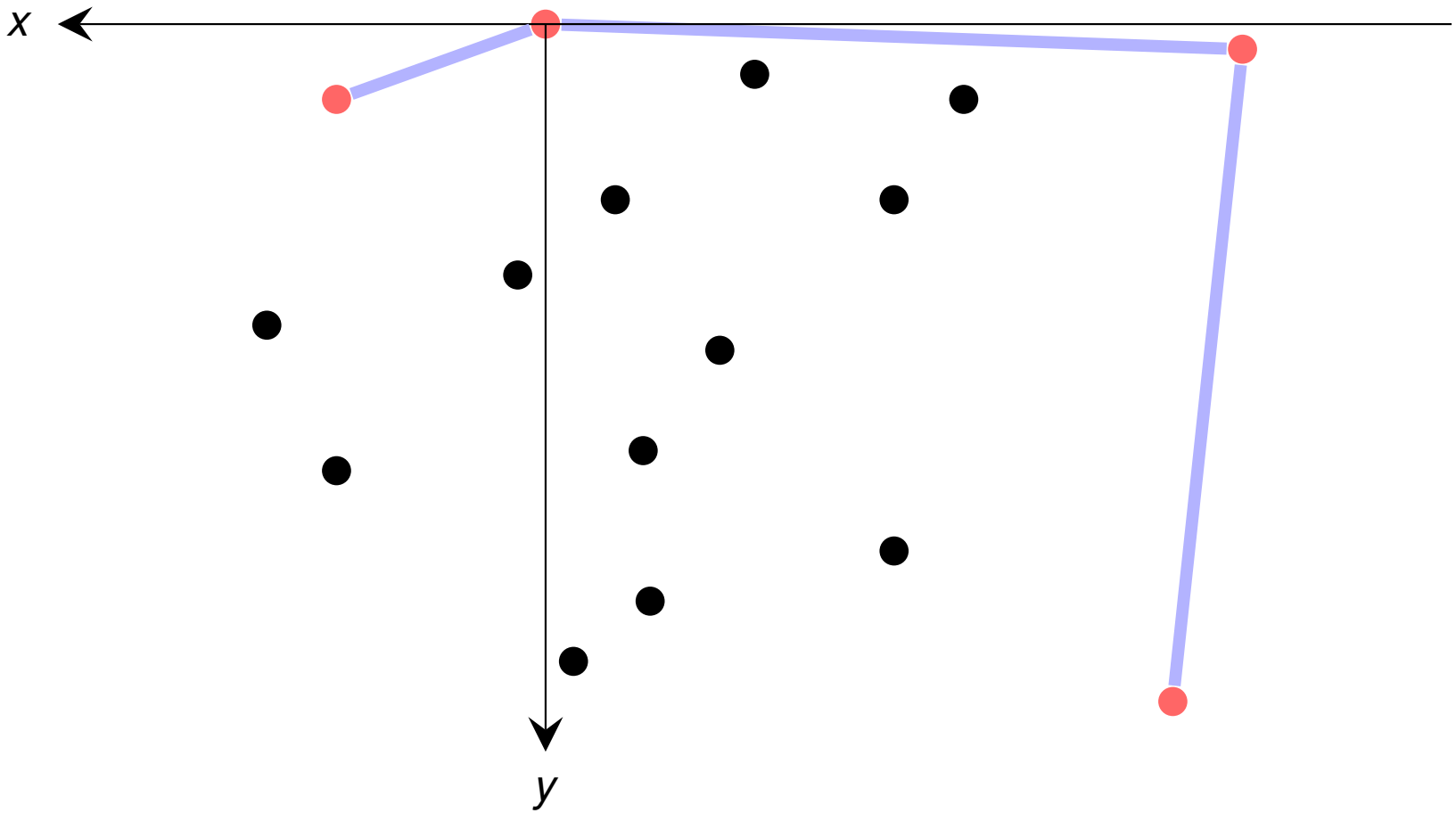
Execution of Jarvis' March



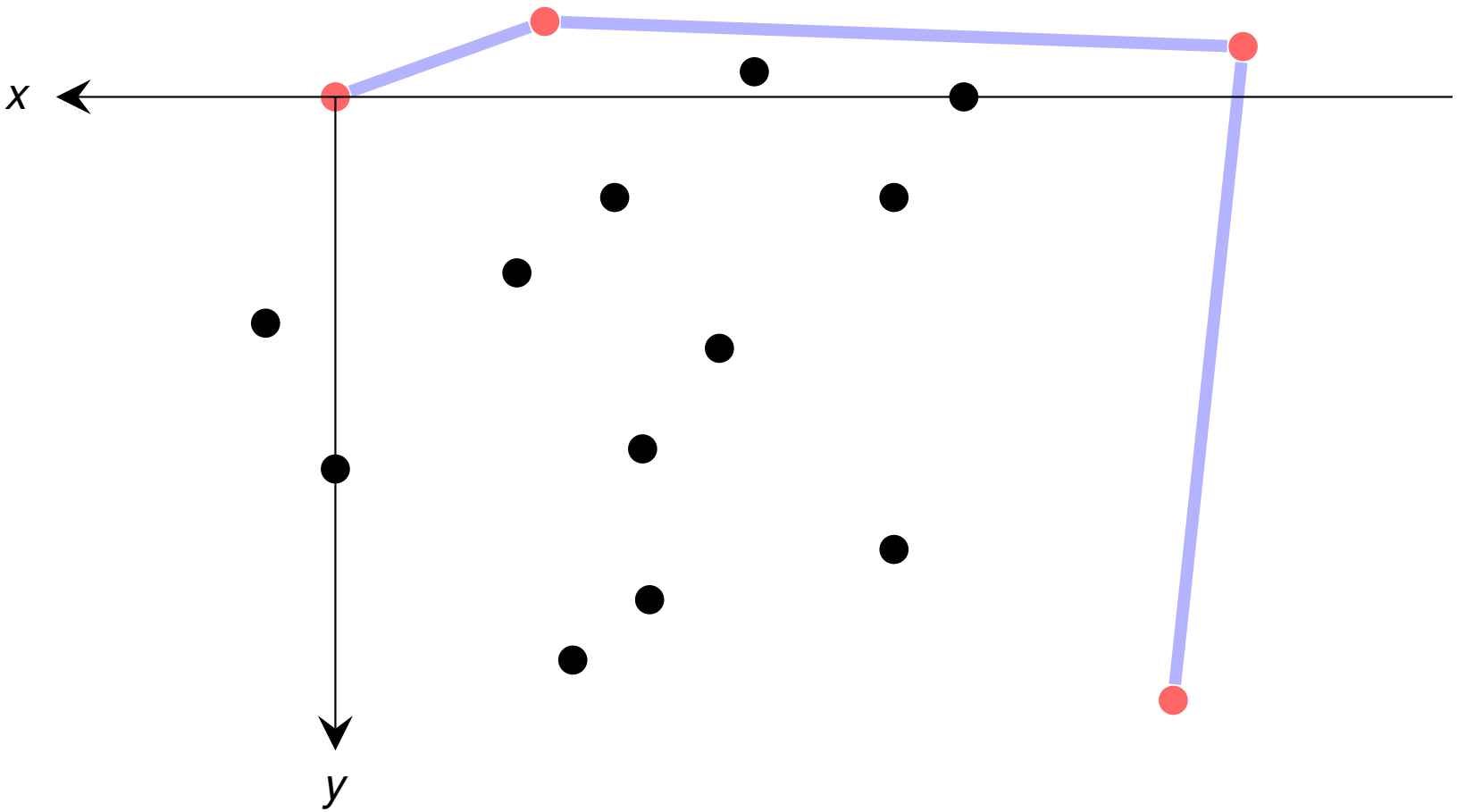
Execution of Jarvis' March



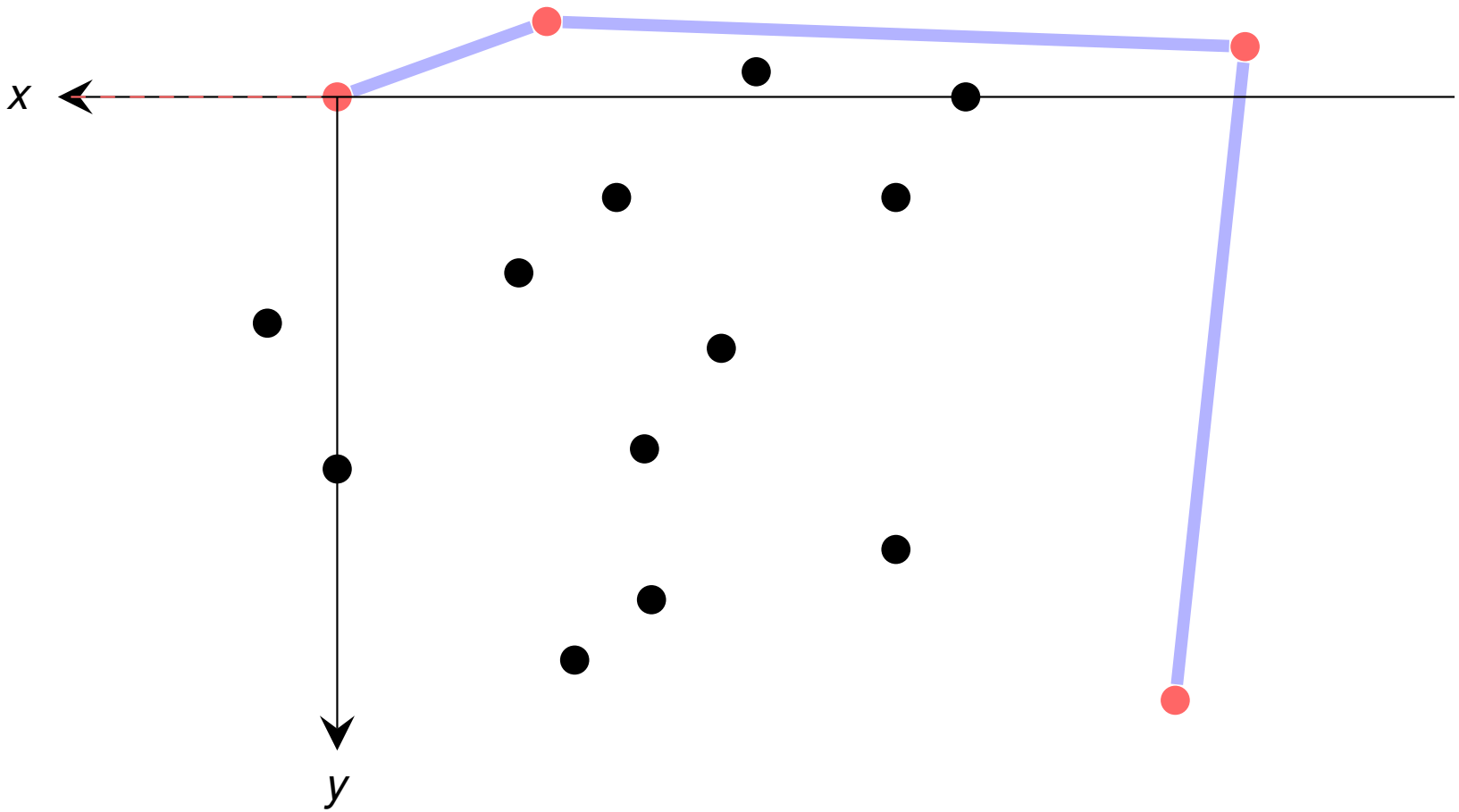
Execution of Jarvis' March



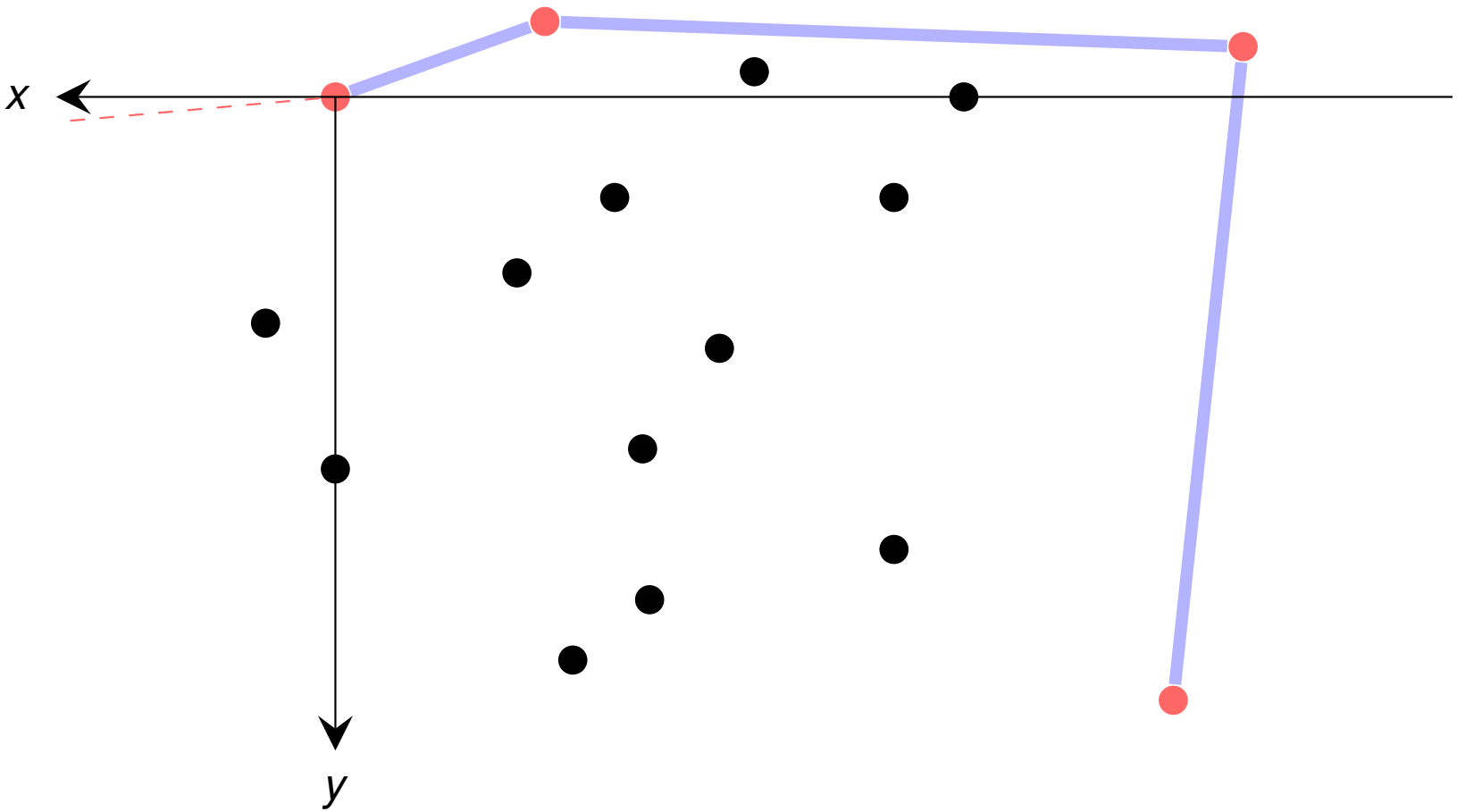
Execution of Jarvis' March



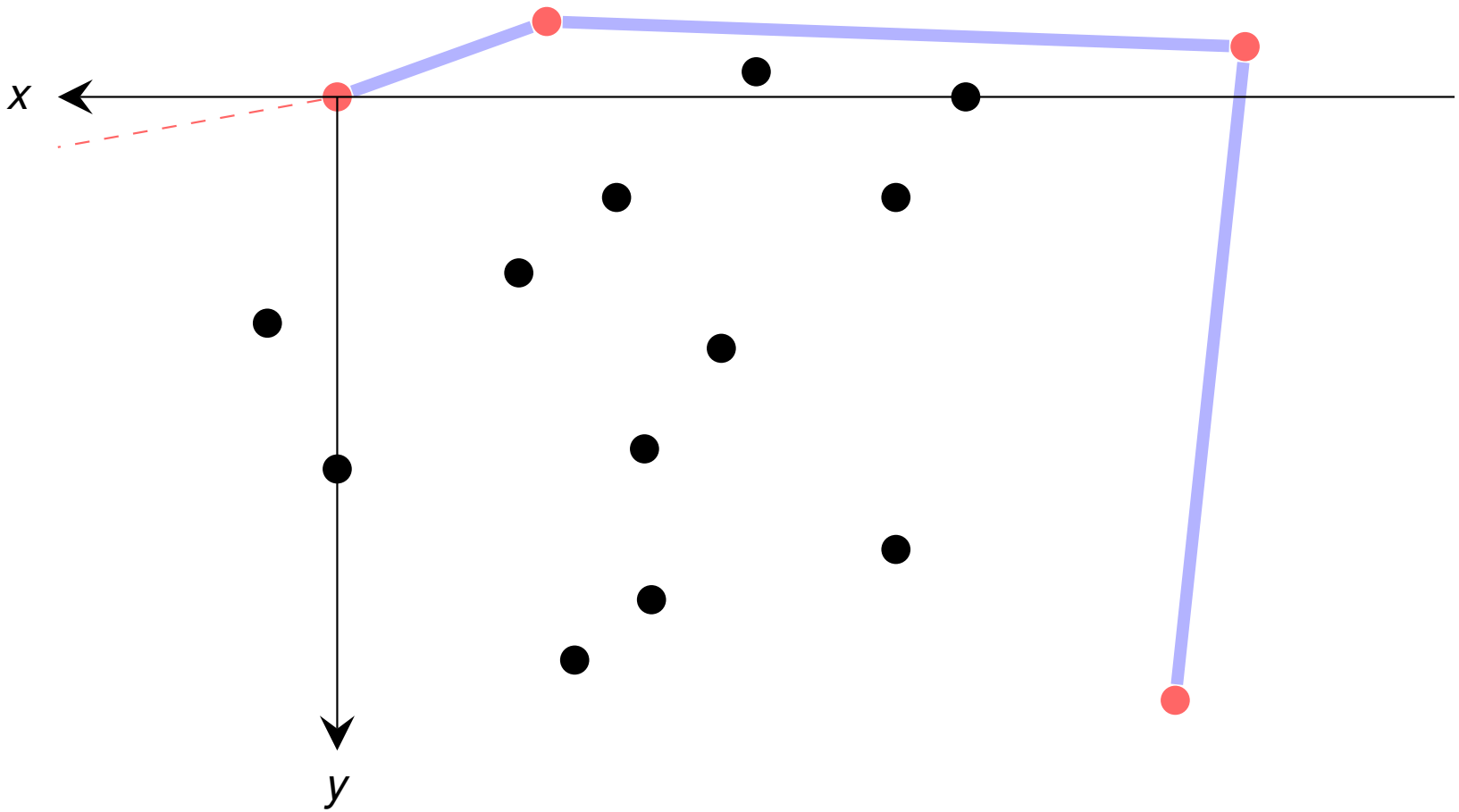
Execution of Jarvis' March



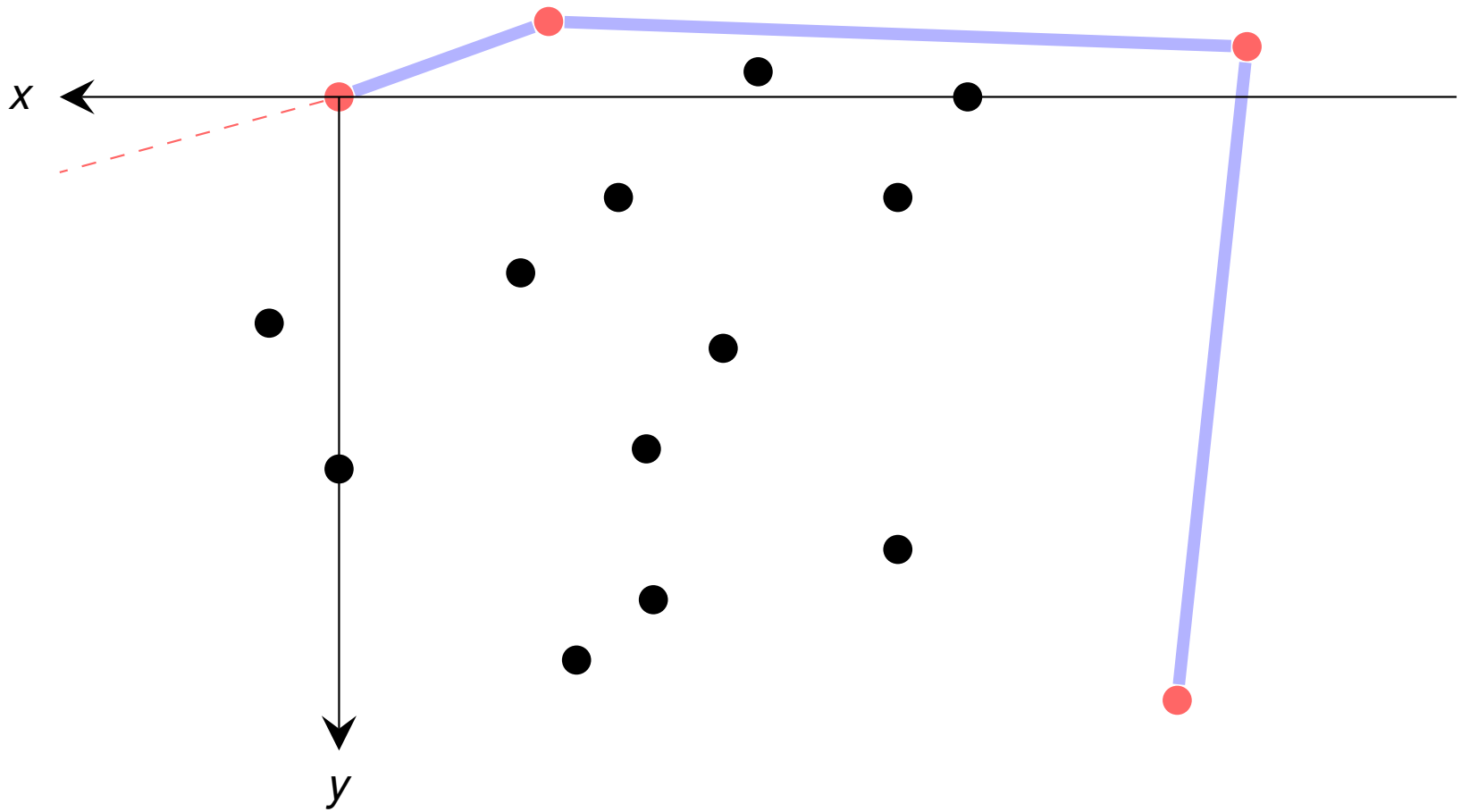
Execution of Jarvis' March



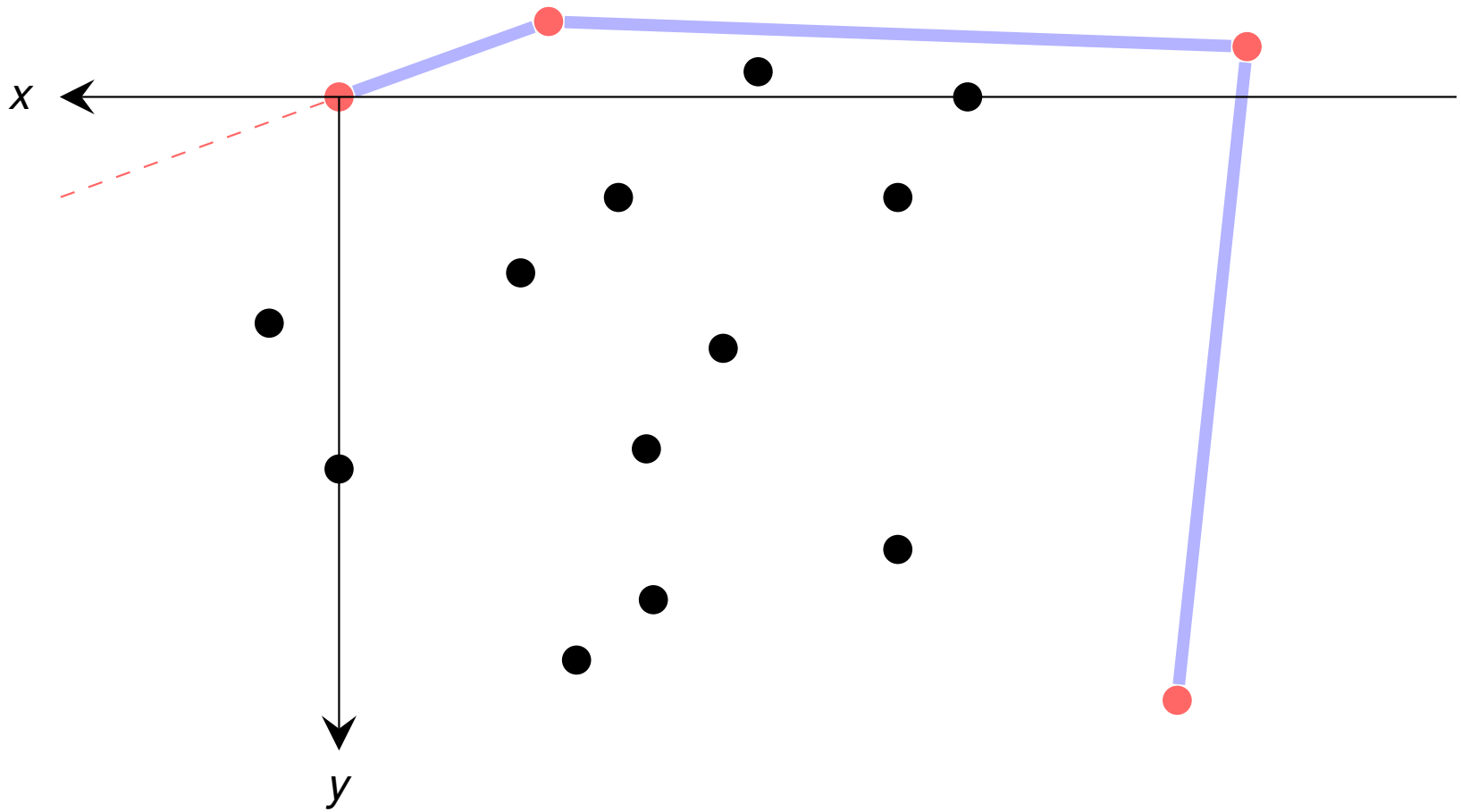
Execution of Jarvis' March



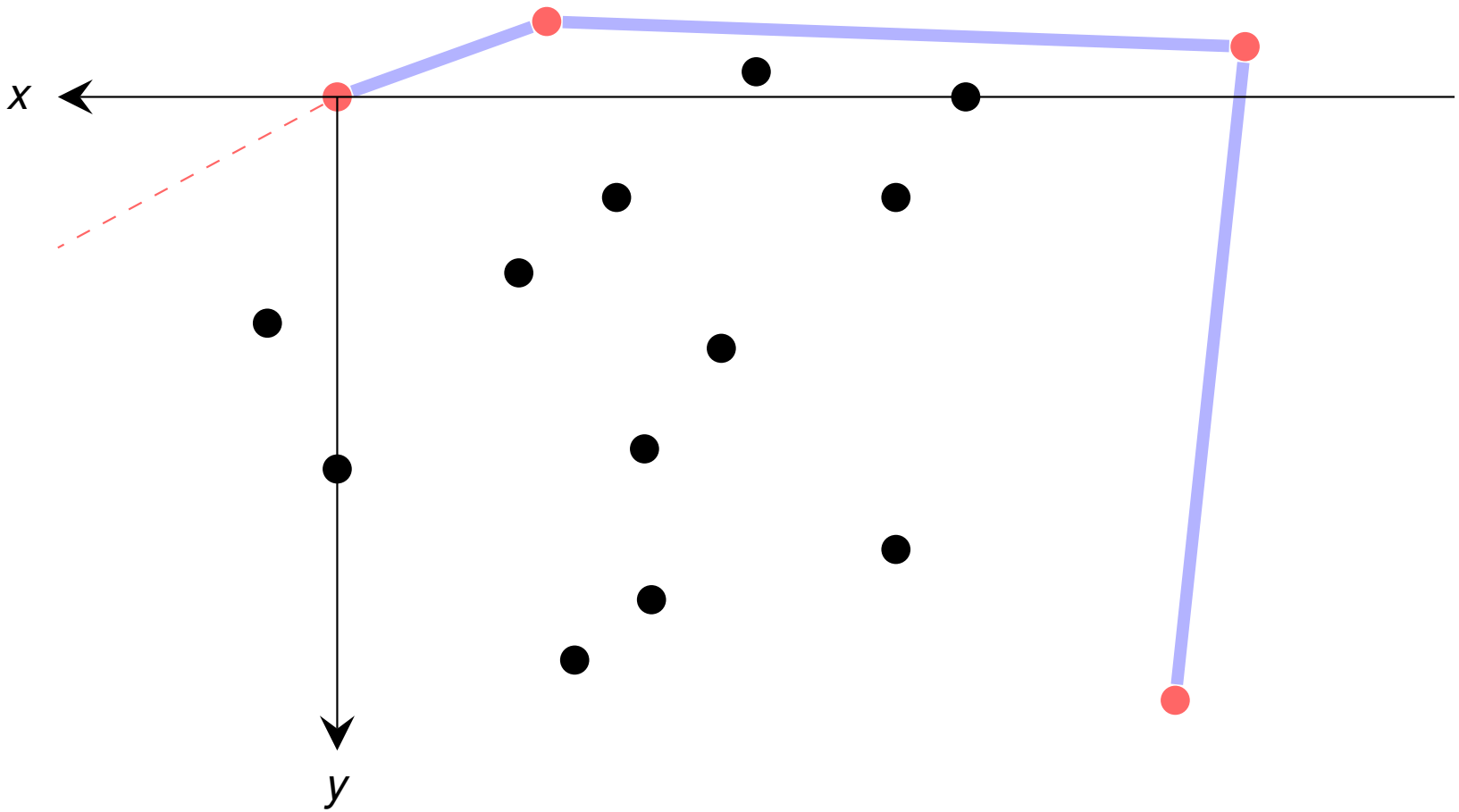
Execution of Jarvis' March



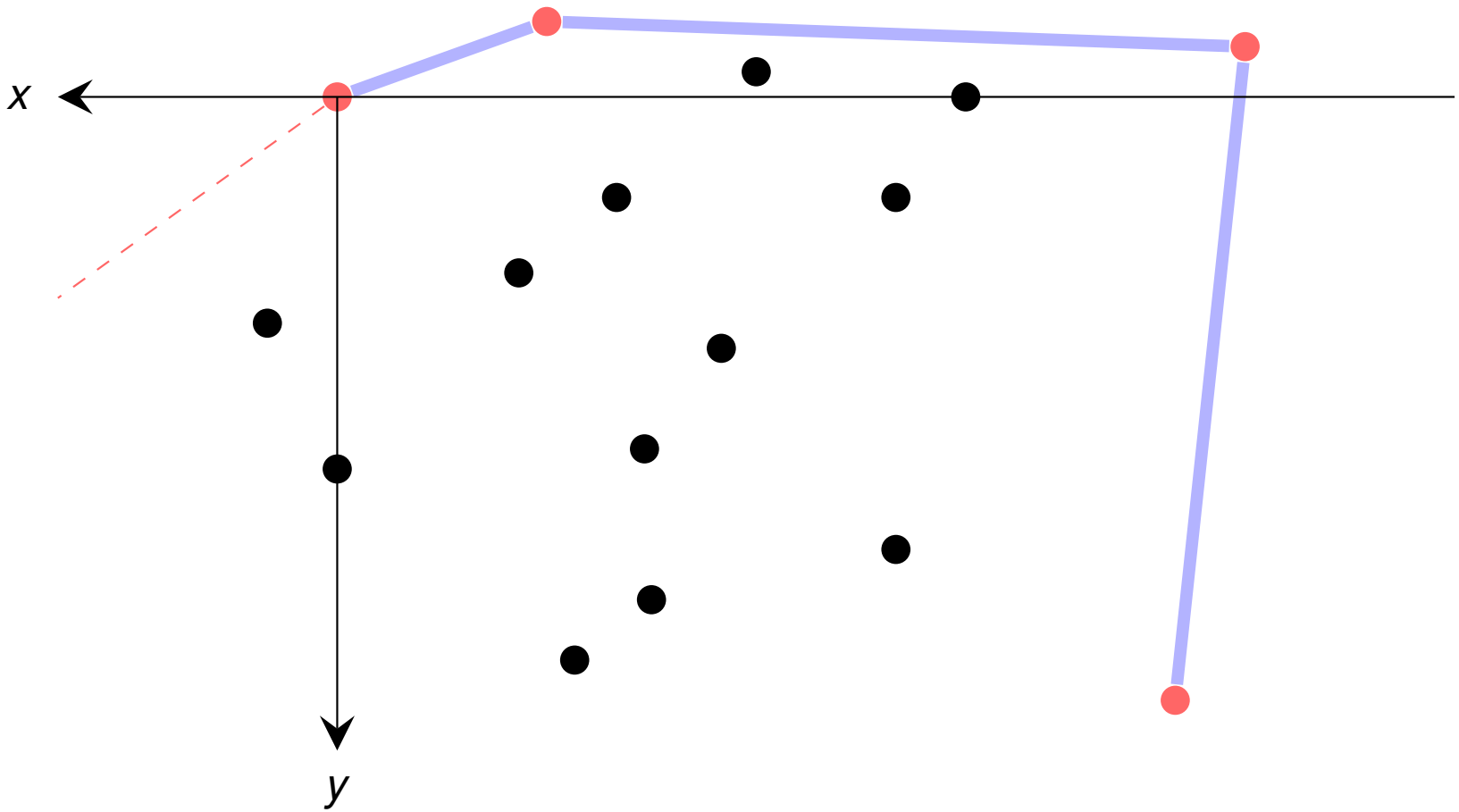
Execution of Jarvis' March



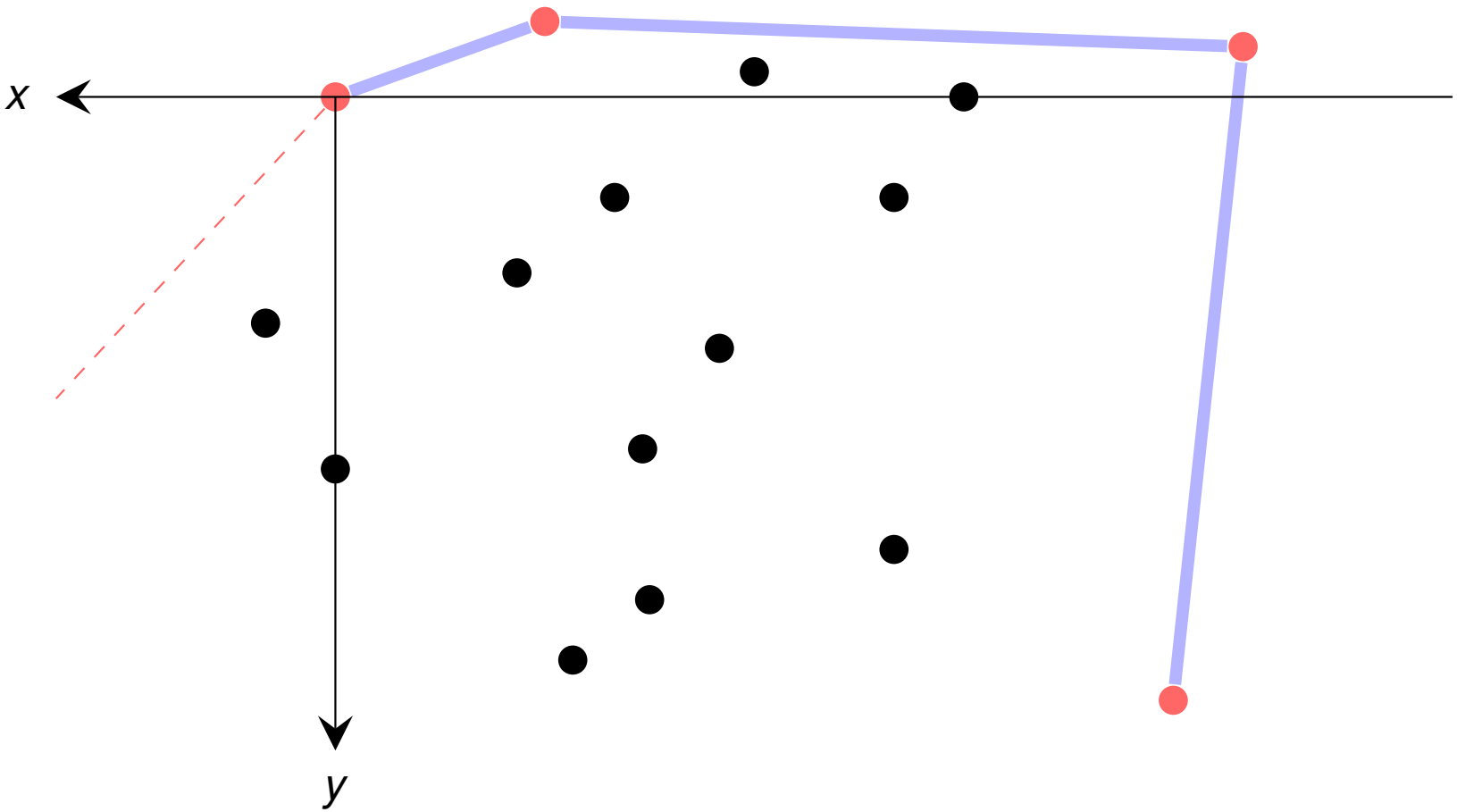
Execution of Jarvis' March



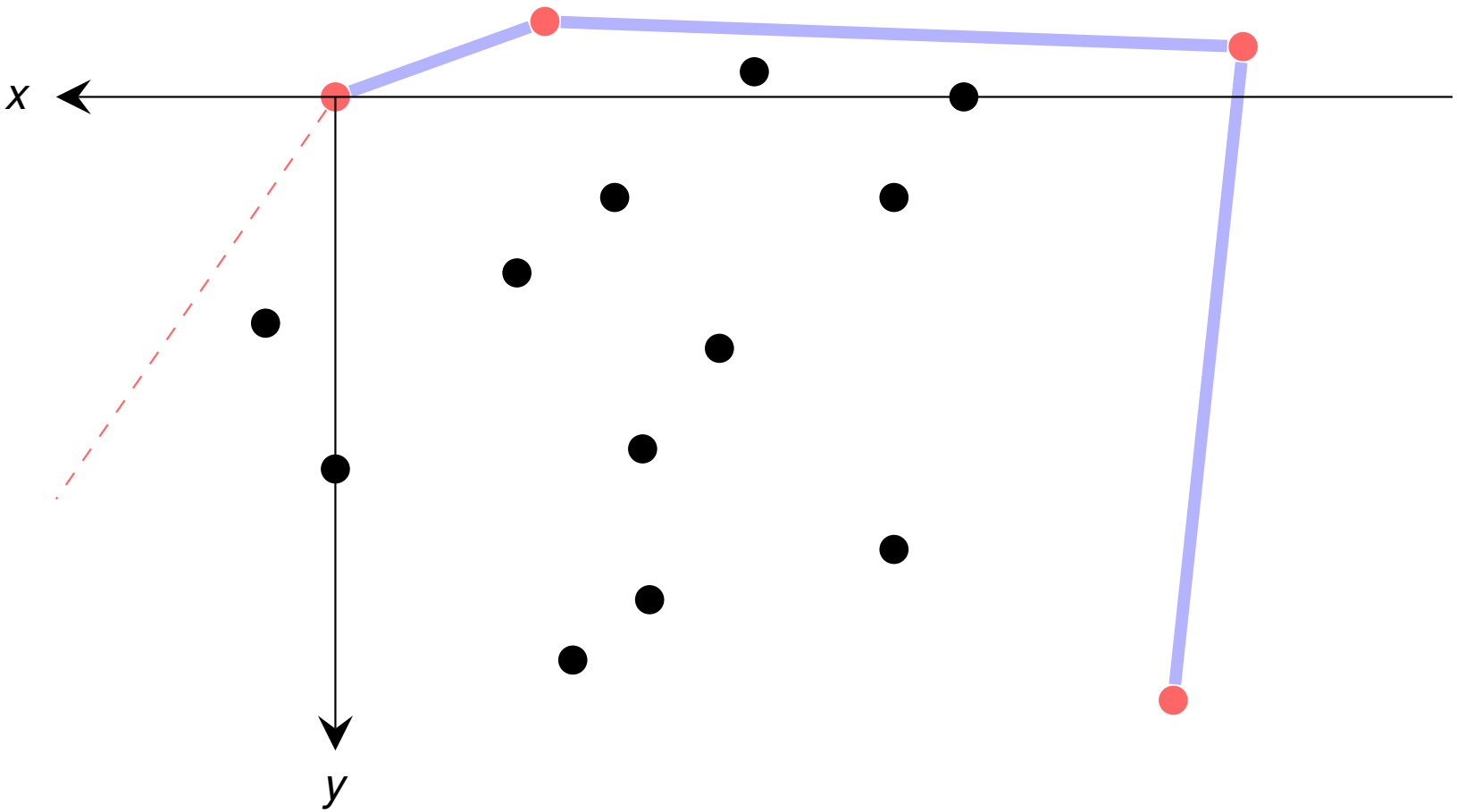
Execution of Jarvis' March



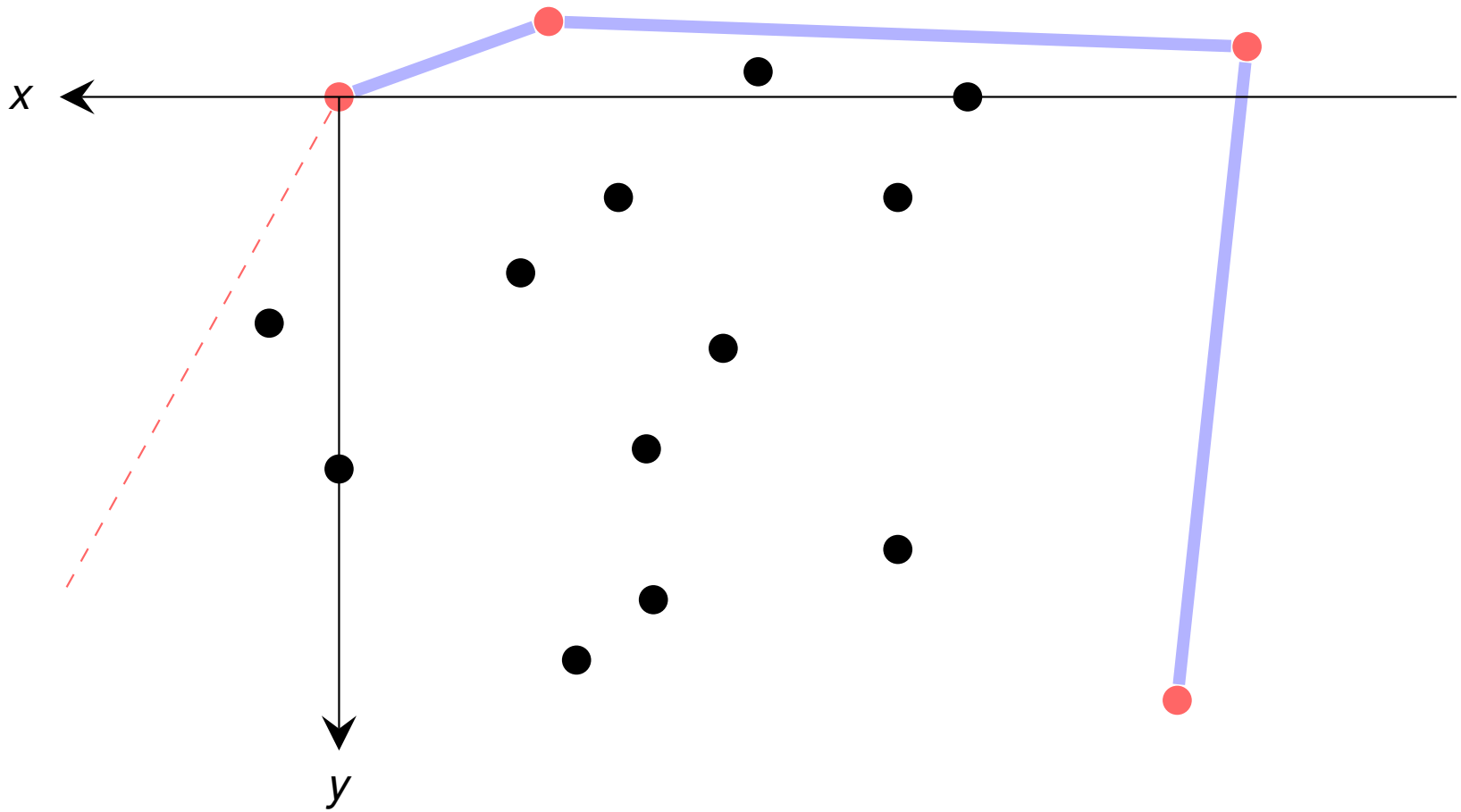
Execution of Jarvis' March



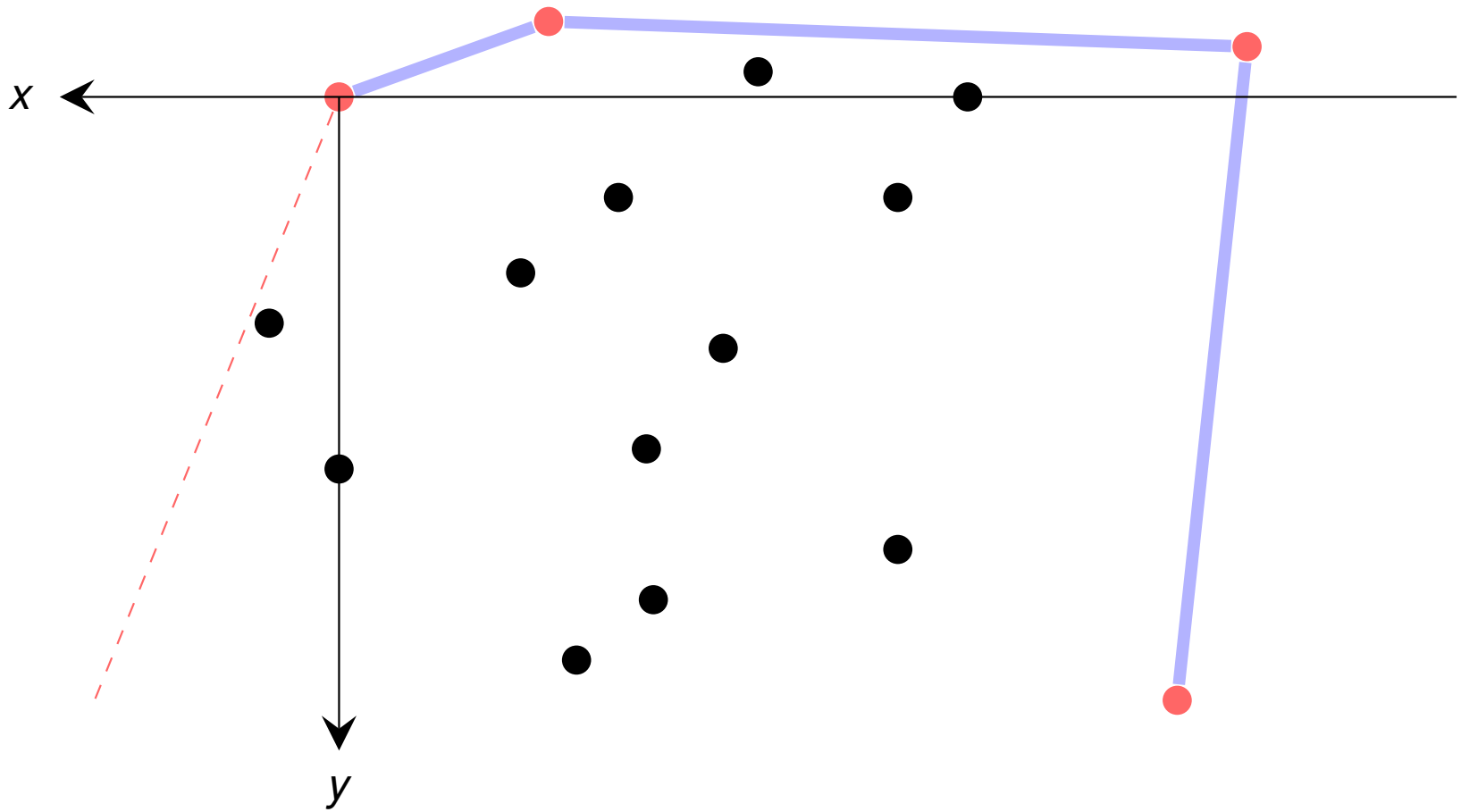
Execution of Jarvis' March



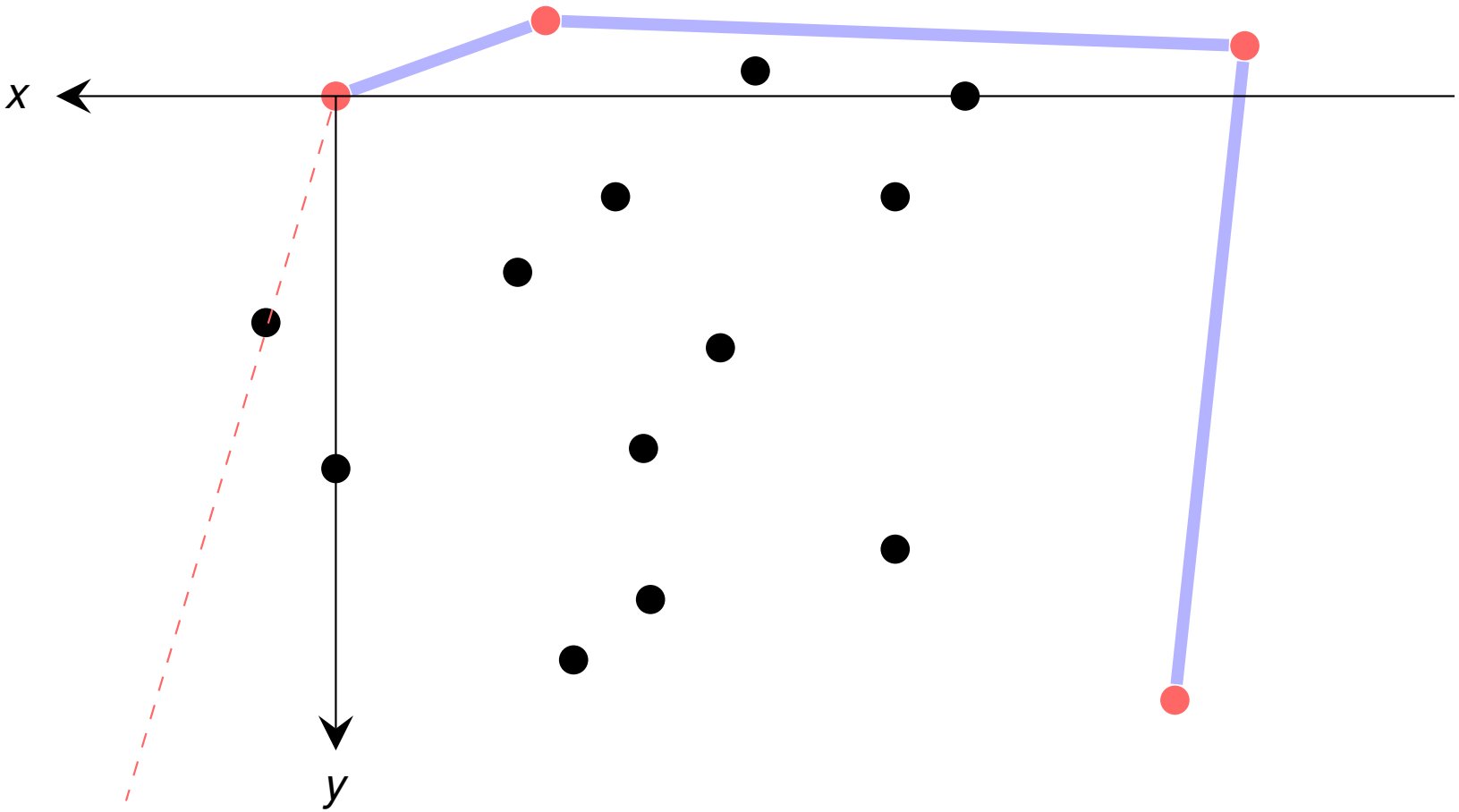
Execution of Jarvis' March



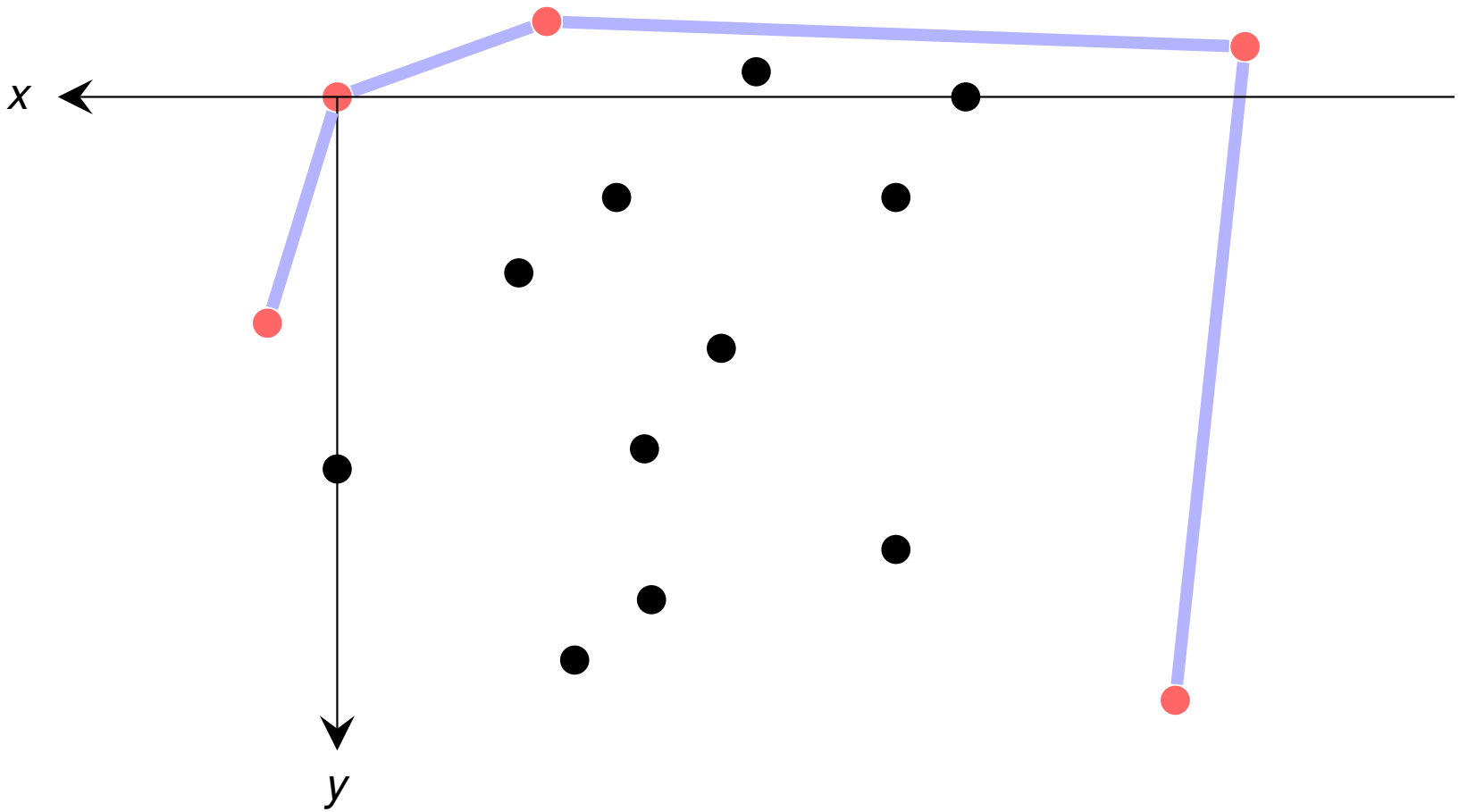
Execution of Jarvis' March



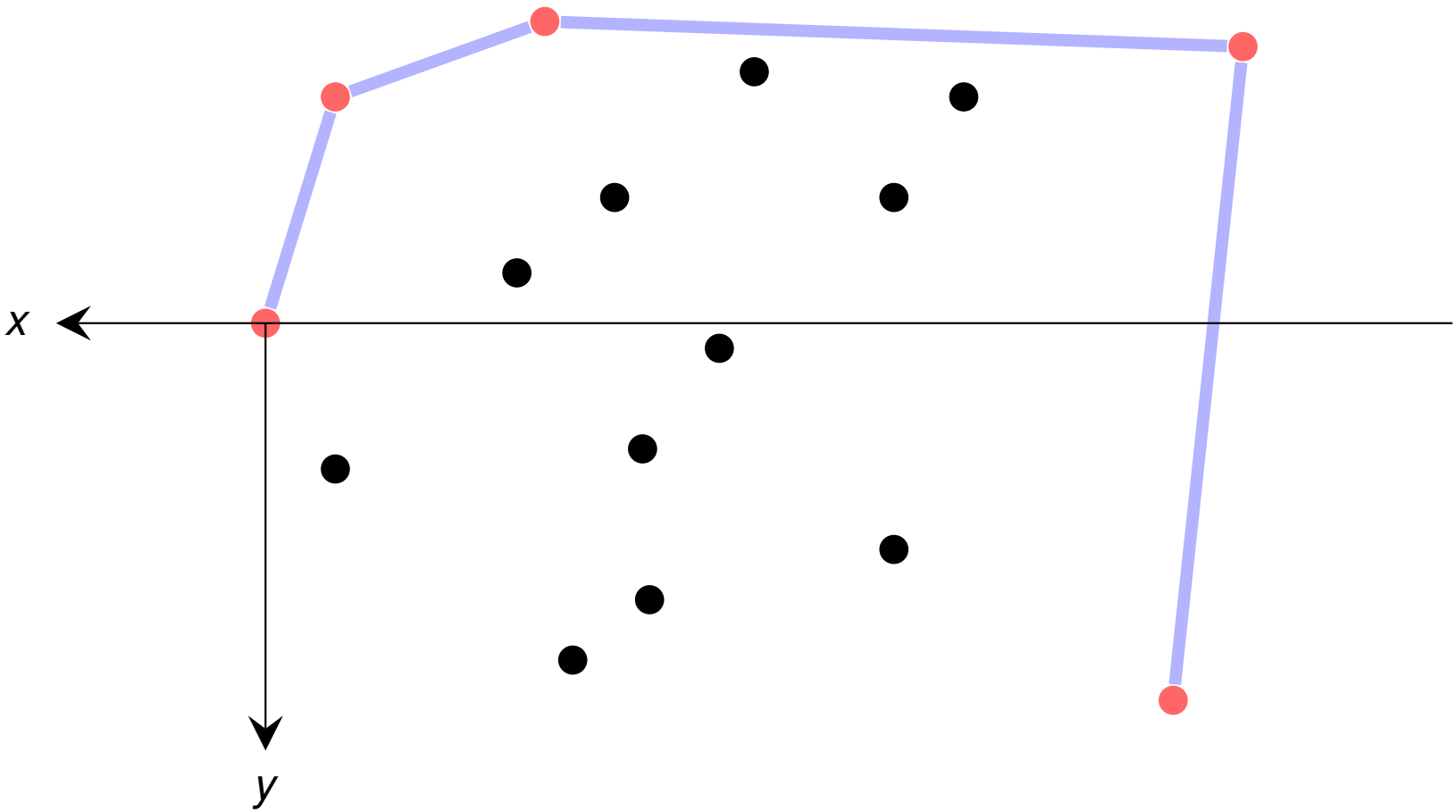
Execution of Jarvis' March



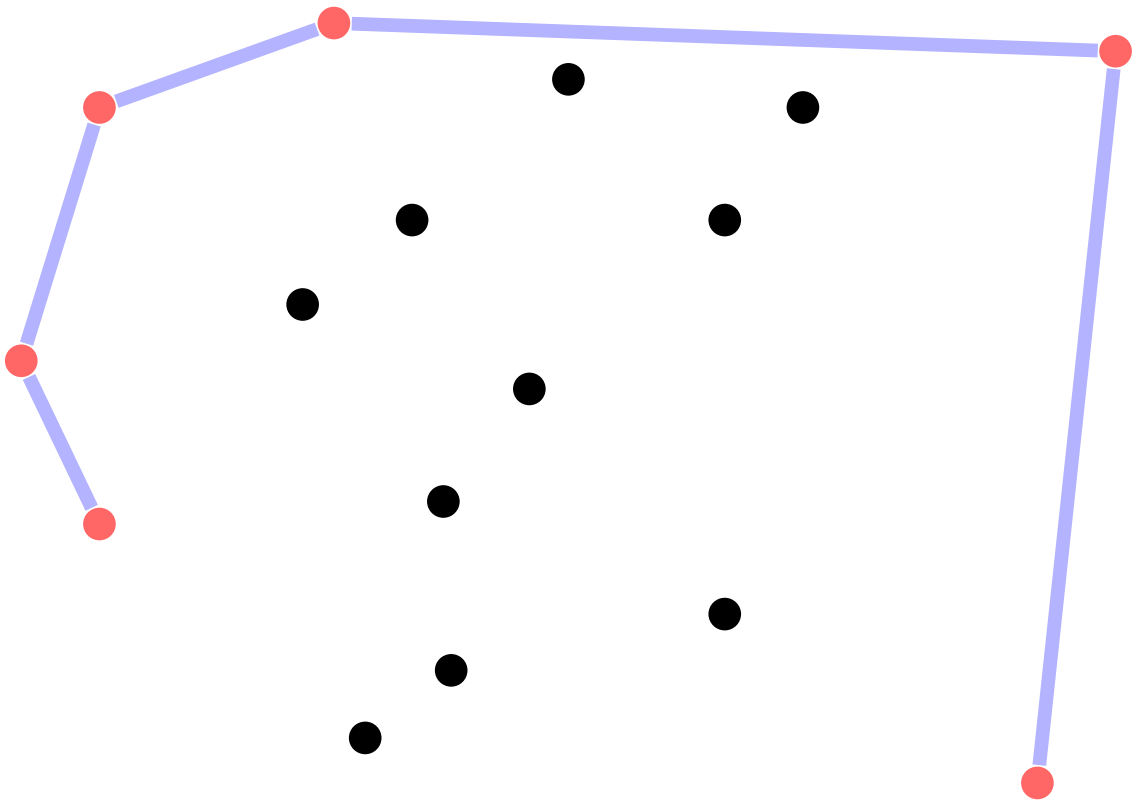
Execution of Jarvis' March



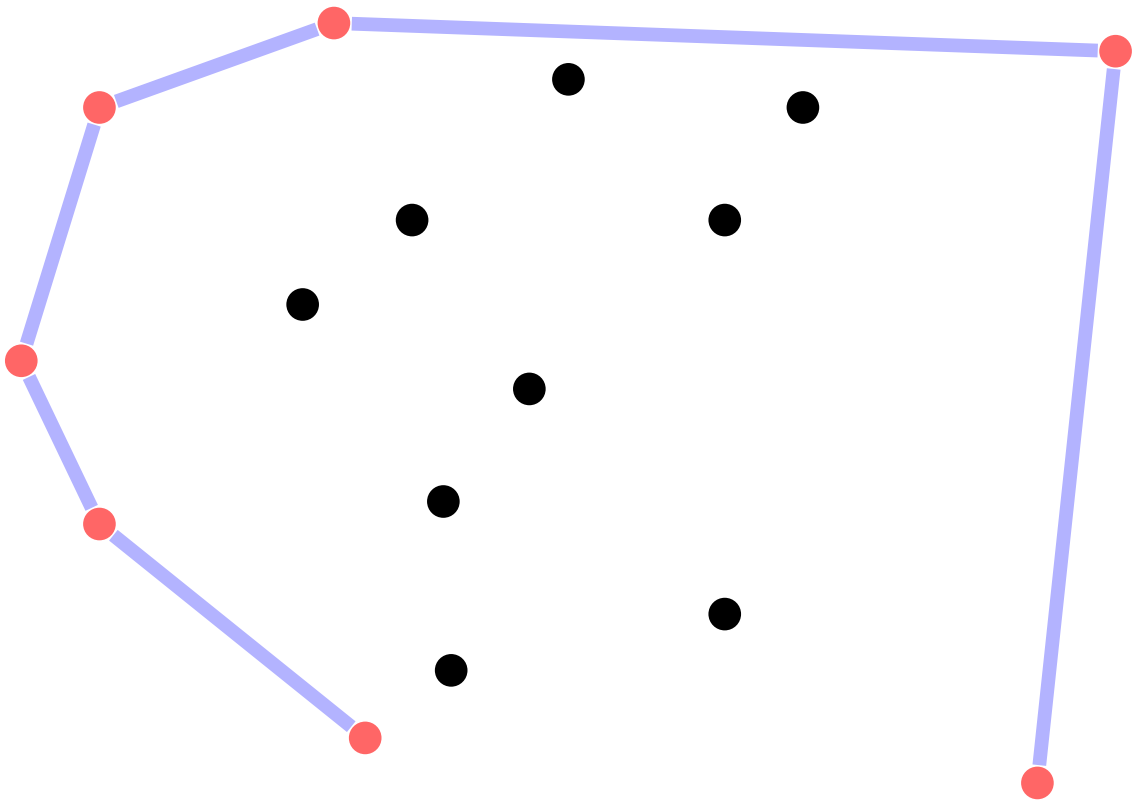
Execution of Jarvis' March



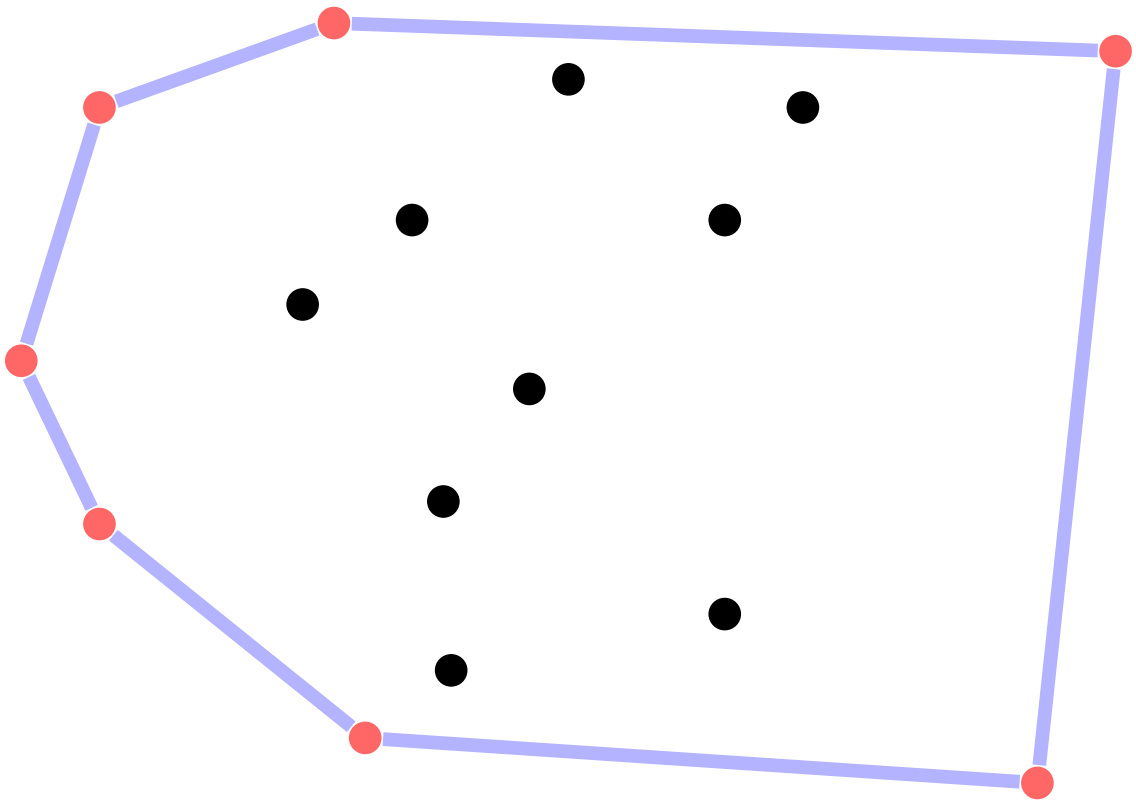
Execution of Jarvis' March



Execution of Jarvis' March



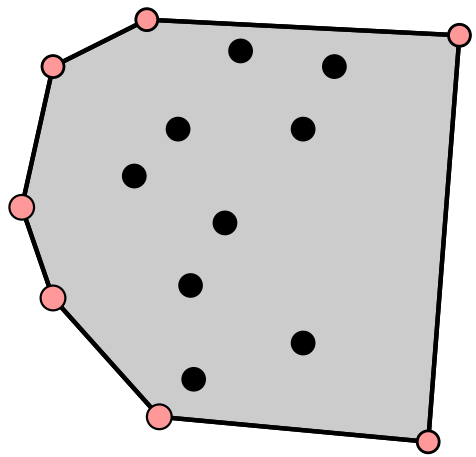
Execution of Jarvis' March



Computing Convex Hull: Summary

Graham's Scan

- natural backtracking algorithm
- cross-product avoids computing polar angles
- Runtime dominated by sorting $\rightsquigarrow O(n \log n)$



Computing Convex Hull: Summary

Graham's Scan

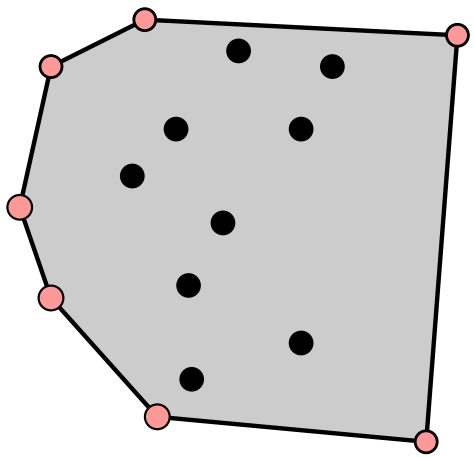
- natural backtracking algorithm
- cross-product avoids computing polar angles
- Runtime dominated by sorting $\rightsquigarrow O(n \log n)$

Jarvis' March

- proceeds like wrapping a gift
- Runtime $O(n \cdot h) \rightsquigarrow$ output-sensitive

Improves Graham's scan only if $h = O(\log n)$

There exists an algorithm with $O(n \log h)$ runtime!



Computing Convex Hull: Summary

Graham's Scan

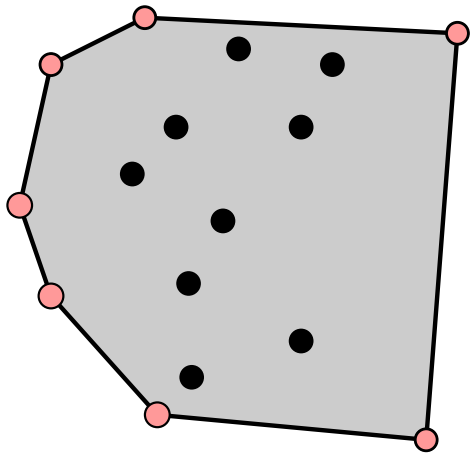
- natural backtracking algorithm
- cross-product avoids computing polar angles
- Runtime dominated by sorting $\rightsquigarrow O(n \log n)$

Jarvis' March

- proceeds like wrapping a gift
- Runtime $O(n \cdot h) \rightsquigarrow$ output-sensitive

Improves Graham's scan only if $h = O(\log n)$

There exists an algorithm with $O(n \log h)$ runtime!



Lessons Learned

- cross product very powerful tool
- need to take care of degenerate cases, numerical precision

The End

Thank you for attending this course &
Best wishes for the rest of your Tripos!

- Don't forget to visit the online feedback page!
- Please send comments on the slides (typos, criticsm, praise etc.) to: **`tms41@cam.ac.uk`**