Computational Geometry

- Algorithms for geometric problems.

- Relevant for Games, Computer Graphics, Robotics and GIS.

- We focus on “combinatorial computational geometry”, that is, the objects under study are basic geometrical objects, such as points, lines segments, polygons and polyhedra.

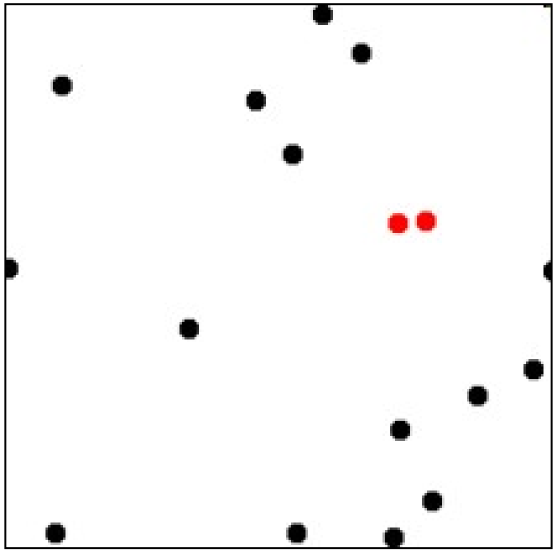
Computational Geometry

Readings:

- S. Skiena, M. Revilla, Programming Challenges, Chapter 13
- S. Skiena, The Algorithm Design Manual, Chapter 17
- T. Cormen et al., Introduction to Algorithms, Chapter 33
- David Goldberg. "What Every Computer Scientist Should Know About Floating-Point Arithmetic". ACM Computing Surveys, 23 (1): 5–48, 1991 (link)

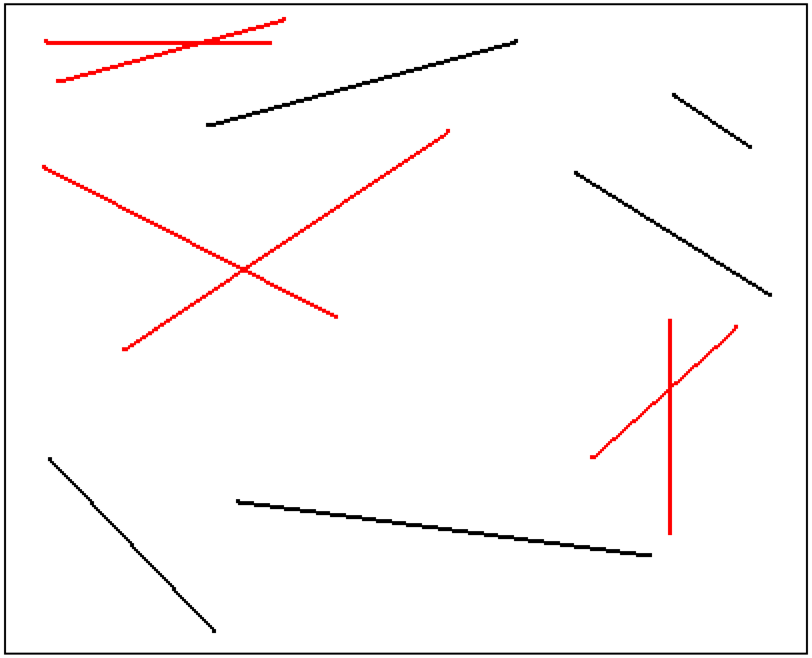
Computational Geometry

- Closest pair problem



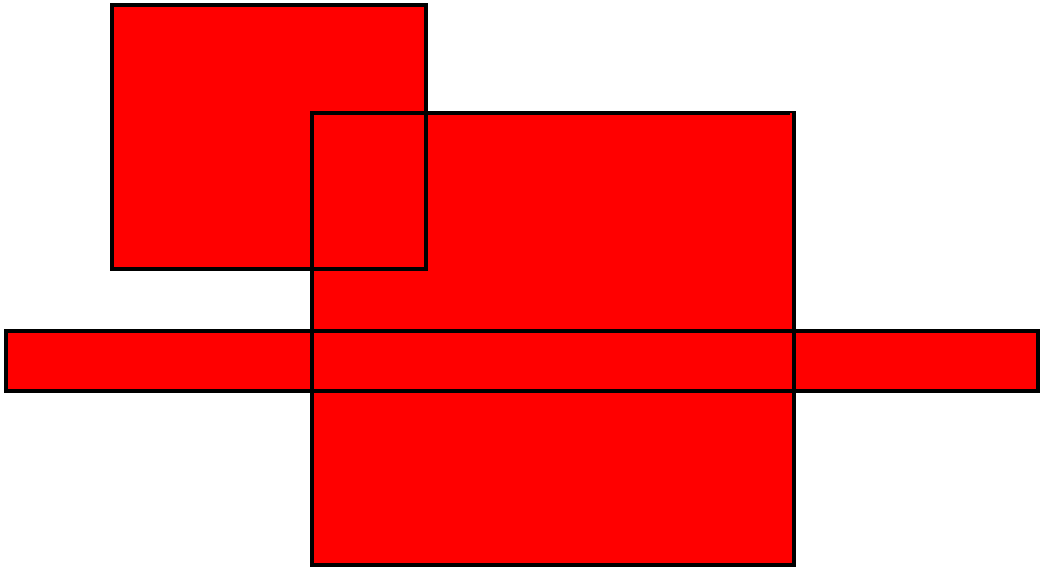
Computational Geometry

- Line intersection problem



Computational Geometry

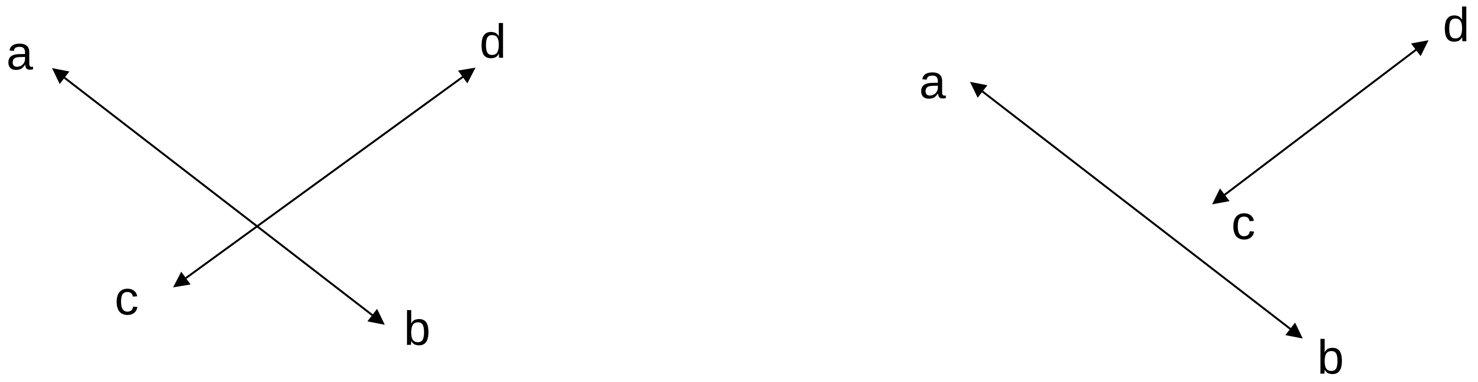
- Area of the union of rectangles problem



Computational Geometry

- Line intersection problem

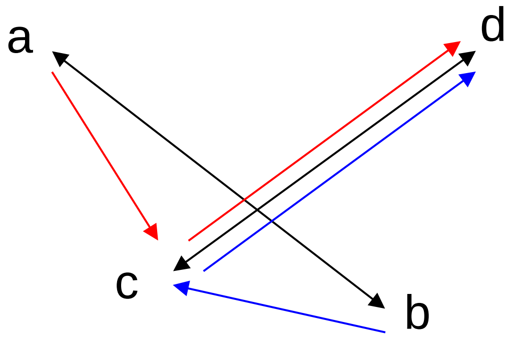
- Two segments *ab* and *cd* intersect if and only if:
  - the endpoints *a* and *b* are on opposite sides of line *cd*, and
  - the endpoints *c* and *d* are on opposite sides of line *ab*.

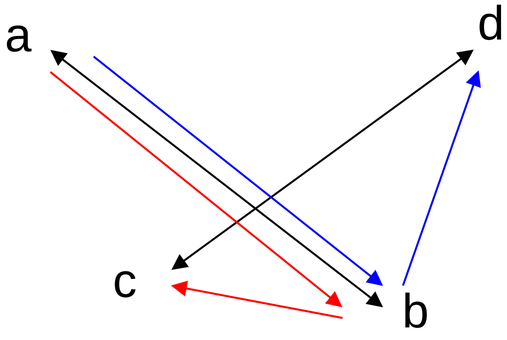


Computational Geometry

- Line intersection problem

- Two segments *ab* and *cd* intersect if and only if:
  - the endpoints *a* and *b* are on opposite sides of line *cd*, and
  - the endpoints *c* and *d* are on opposite sides of line *ab*.

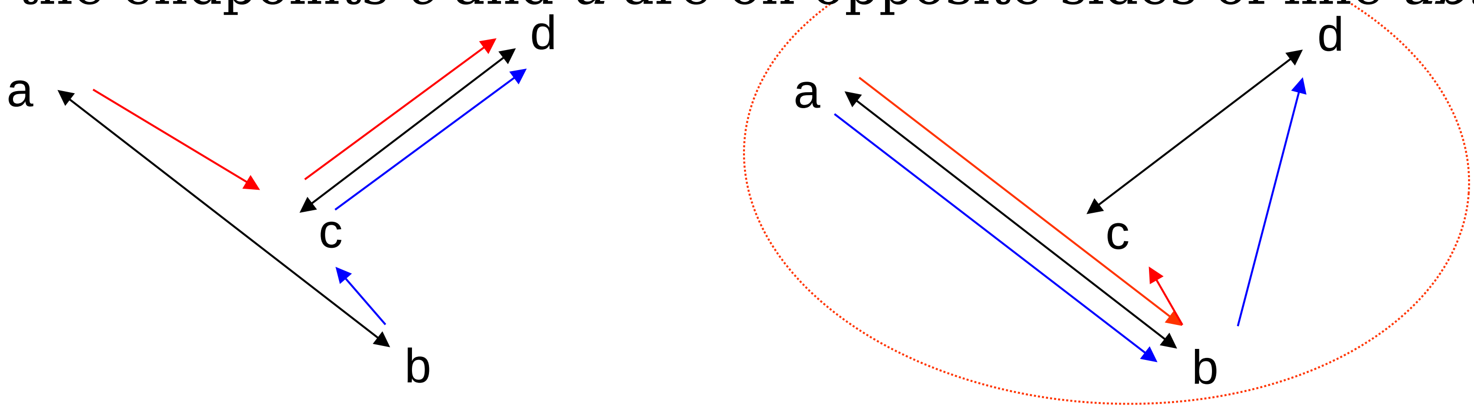




Computational Geometry

- Line intersection problem

- Two segments *ab* and *cd* intersect if and only if:
  - the endpoints *a* and *b* are on opposite sides of line *cd*, and
  - the endpoints *c* and *d* are on opposite sides of line *ab*.

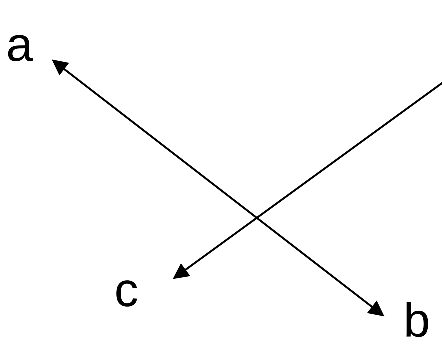


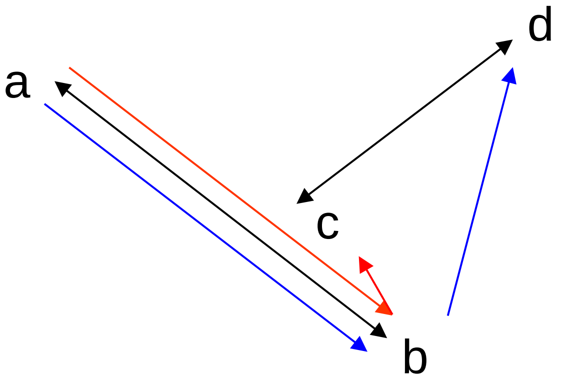
Computational Geometry

- Line intersection problem

- CCW(*a,b,c*) tests whether *a,b,c* are in counterclockwise order.
- Two segments *ab* and *cd* intersect if and only if:

  CCW(*a,c,d*) ≠ CCW(*b,c,d*) and CCW(*a,b,c*) ≠ CCW(*a,b,d*)





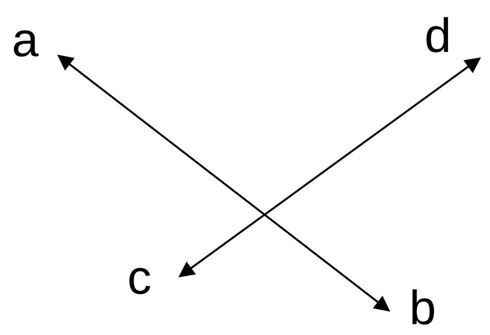
Computational Geometry

- Line intersection problem

- CCW(*a,b,c*) tests whether *a,b,c* are in counterclockwise order.

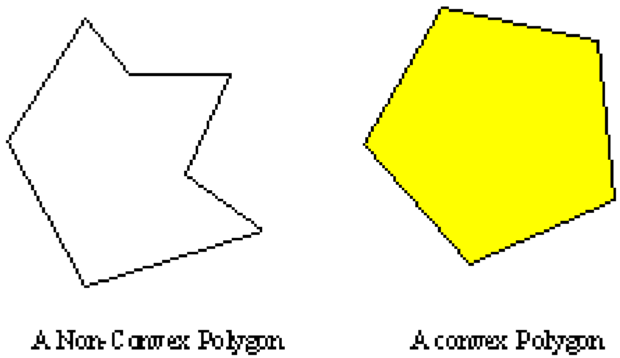
- CCW(*a,b,c*) is True if det(*M*) >0, False otherwise.

$$M = \begin{vmatrix} 1 & x_a & y_a \\ 1 & x_b & y_b \\ 1 & x_c & y_c \end{vmatrix}$$



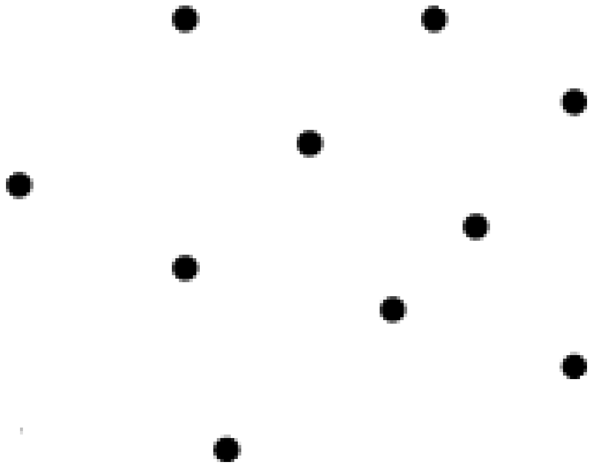
Computational Geometry

- Convex Hull
  - Convex hull of a set A of points: The boundary of the minimal convex set containing A
  - An object is convex if for every pair of points within the object, every point on the segment that joins them is also within the object.



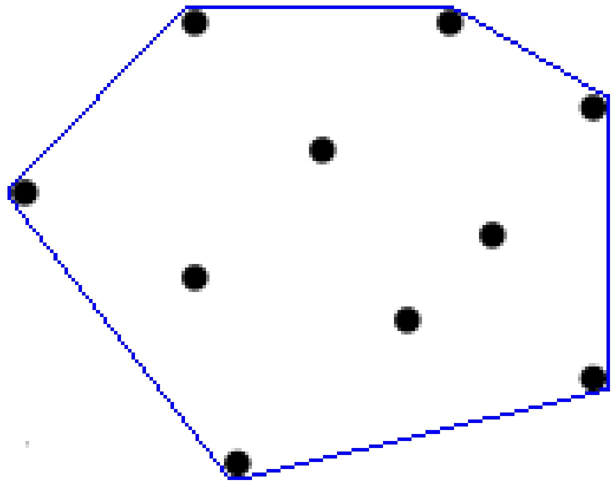
Computational Geometry

- Convex Hull



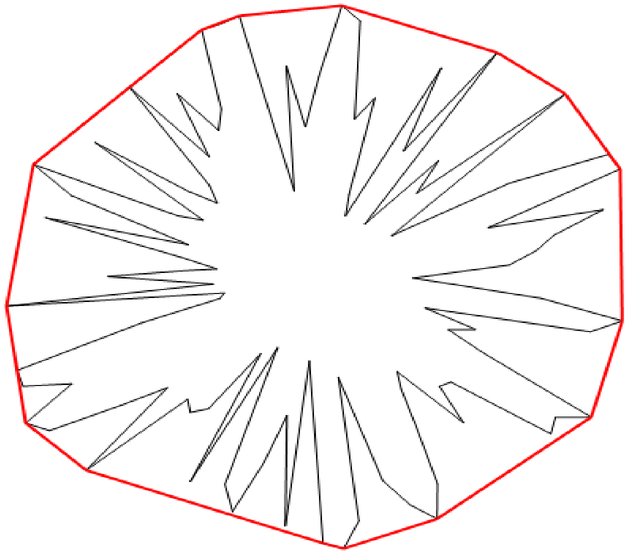
Computational Geometry

- Convex Hull



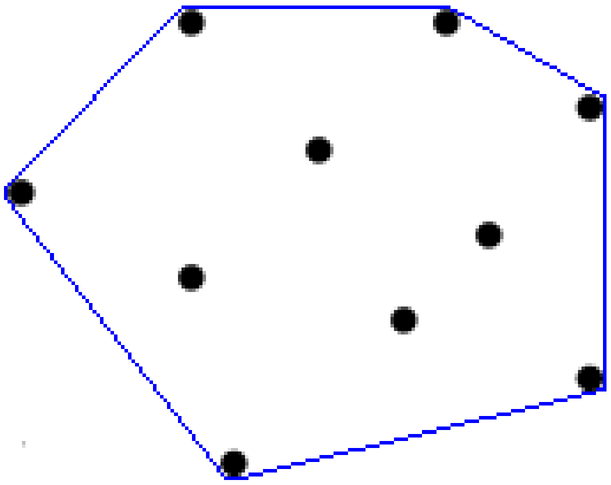
Computational Geometry

- Convex Hull Applications
  - Robot path planning
  - Bounding representation for polygons in GIS
  - Triangulations



Computational Geometry

- Convex Hull



Some invariants:

- The bottomost, topmost, leftmost and rightmost points belong to the convex hull
- Let p be a point in the convex-hull. The next point in the convex hull minimizes the polar angle centered in *p* and relative to the x-axis.

Computational Geometry

- Convex Hull – 2D Algorithms

Jarvis March algorithm (or Gift Wrapping algorithm)

Step 1: Start from the bottom-most point p (it is in the convex hull)

Step 2: While going up

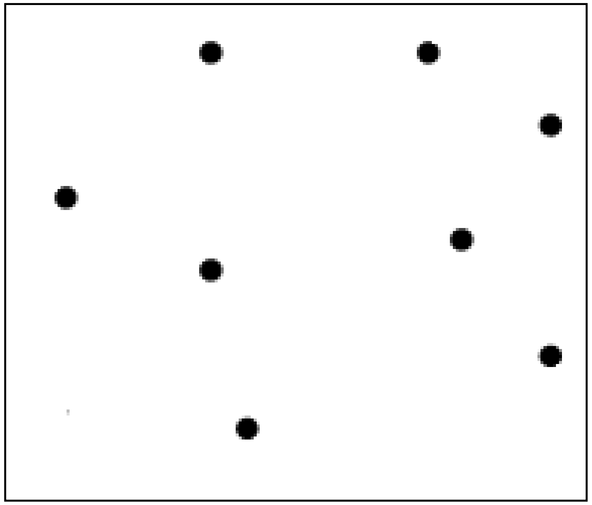
Step 2.1: Select the point q with the smallest polar angle centered in p and relative to the positive x-axis. Let p = q.

Step 3: While going down

Step 3.1: Select the point q with the smallest polar angle centered in p and relative to the negative x-axis. Let p = q.

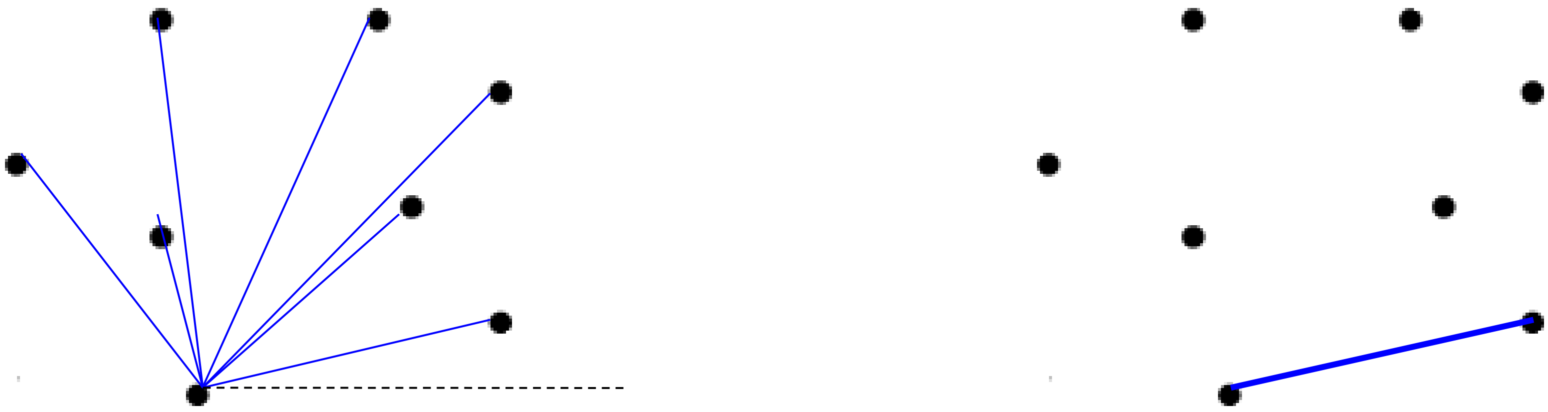
Computational Geometry

- Convex Hull – 2D Algorithms
  - Jarvis March algorithm



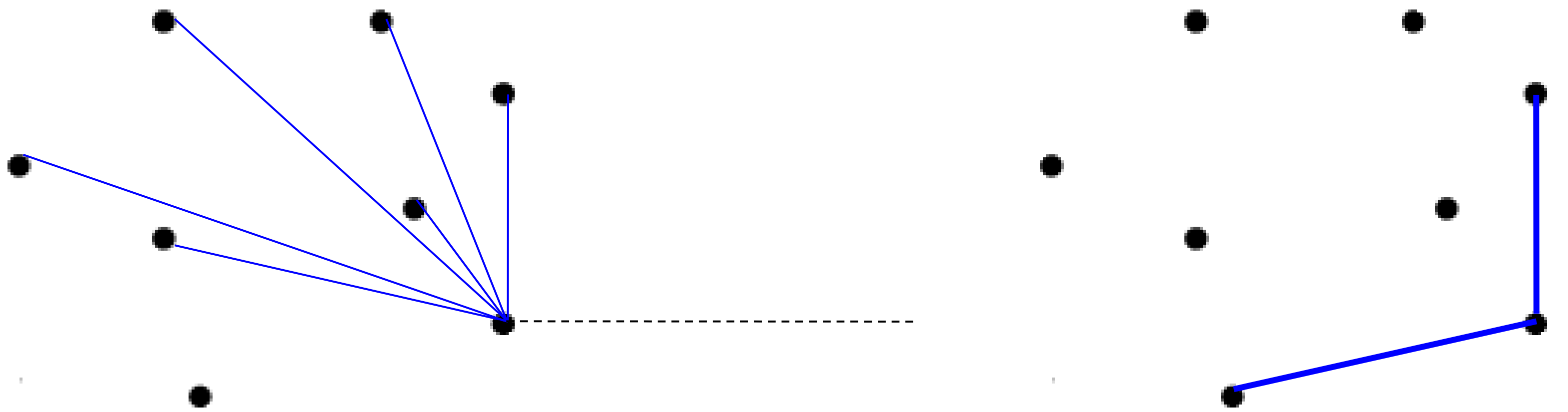
Computational Geometry

- Convex Hull – 2D Algorithms
  - Jarvis March algorithm



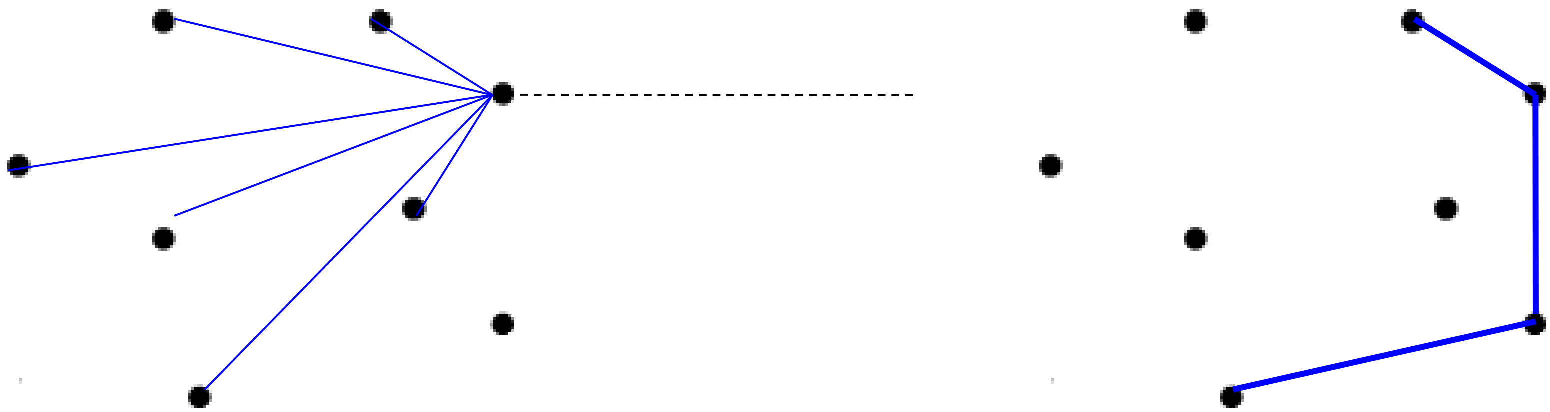
Computational Geometry

- Convex Hull – 2D Algorithms
  - Jarvis March algorithm



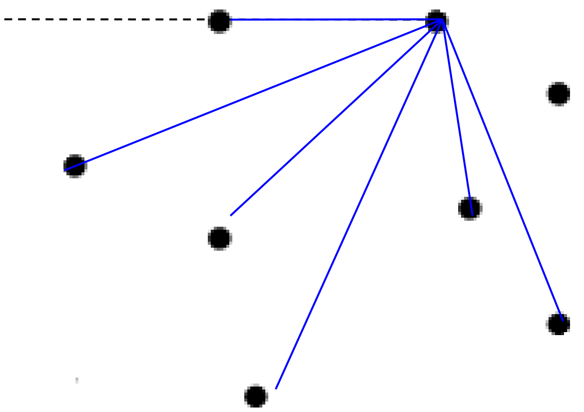
Computational Geometry

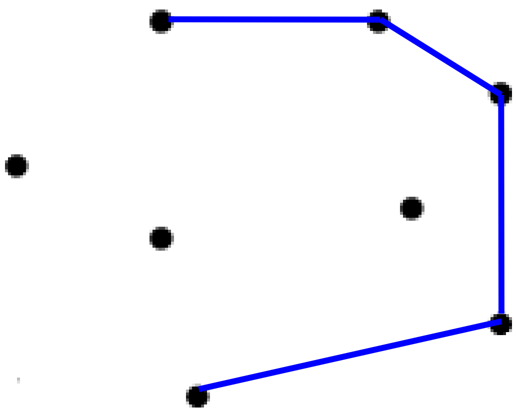
- Convex Hull – 2D Algorithms
  - Jarvis March algorithm



Computational Geometry

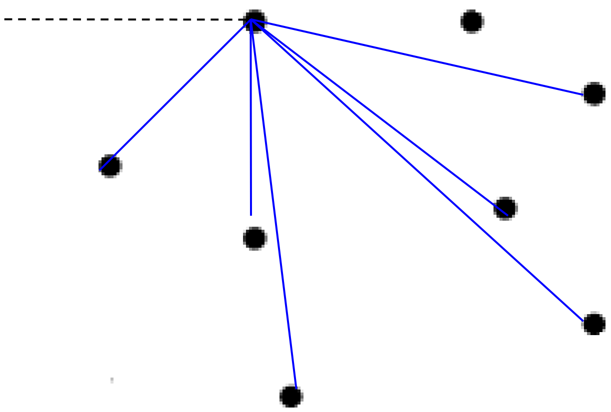
- Convex Hull – 2D Algorithms
  - Jarvis March algorithm

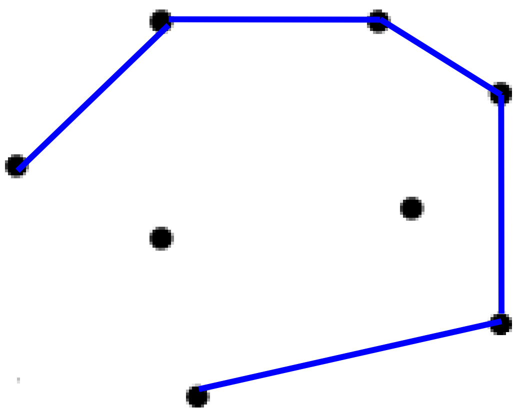




Computational Geometry

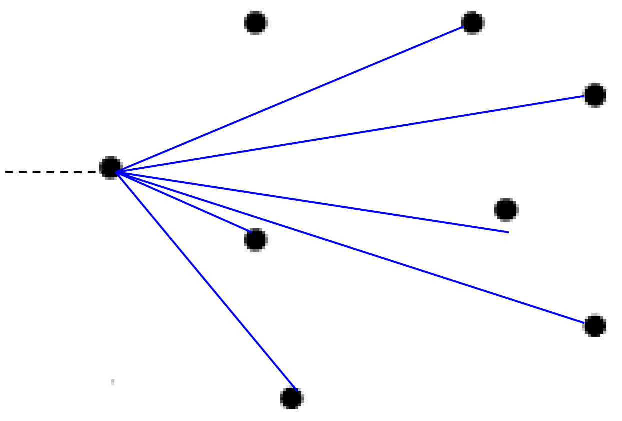
- Convex Hull – 2D Algorithms
  - Jarvis March algorithm

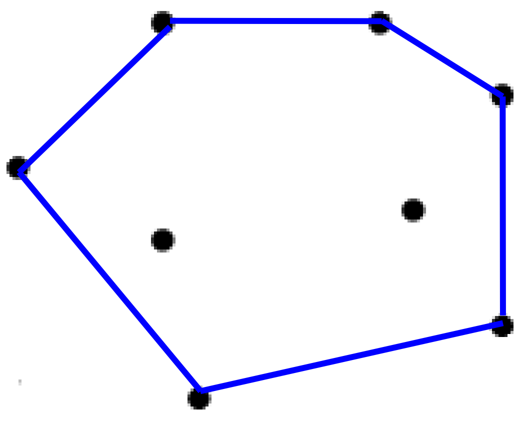




Computational Geometry

- Convex Hull – 2D Algorithms
  - Jarvis March algorithm



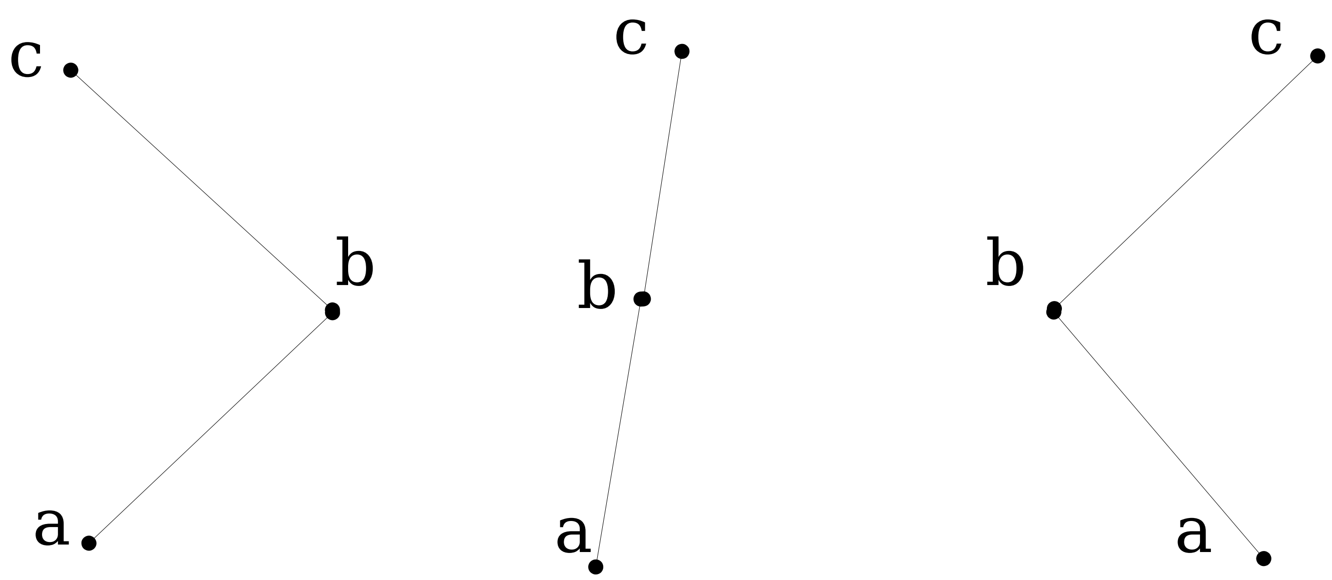


Computational Geometry

- Convex Hull – 2D Algorithms
  - Jarvis March algorithm
  - Compare polar angles against all n points: $O(n)$-time
  - Overall time complexity: $O(nk)$-time
  - It is possible to use CCW test for comparing polar angles.

Computational Geometry

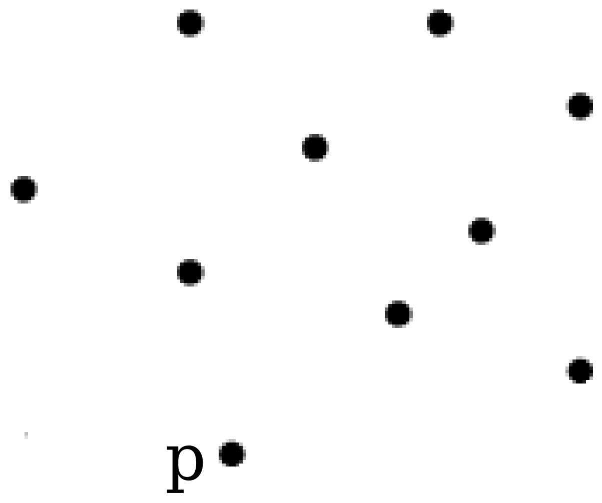
- CCW (counterclockwise) test



CCW(a,b,c) > 0 CCW(a,b,c) = 0 CCW(a,b,c) < 0

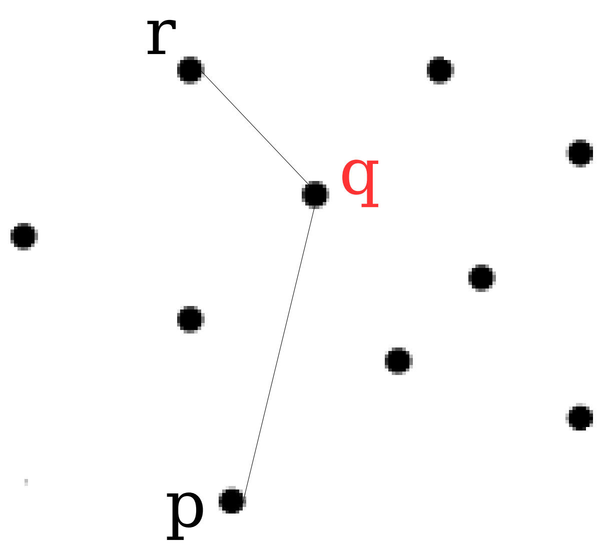
Computational Geometry

- CCW (counterclockwise) test



Computational Geometry

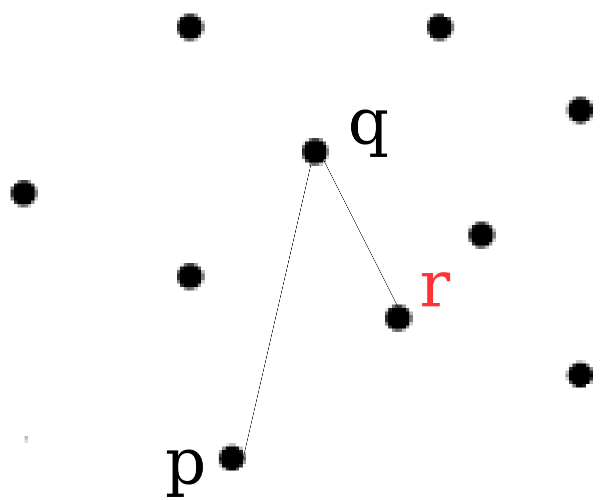
- CW (counterclockwise) test



CCW(p,q,r) > 0

Computational Geometry

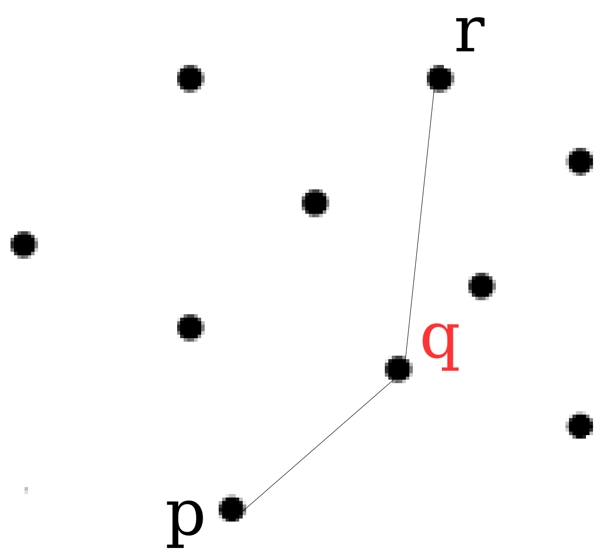
- CCW (counterclockwise) test



CCW(p,q,r) < 0

Computational Geometry

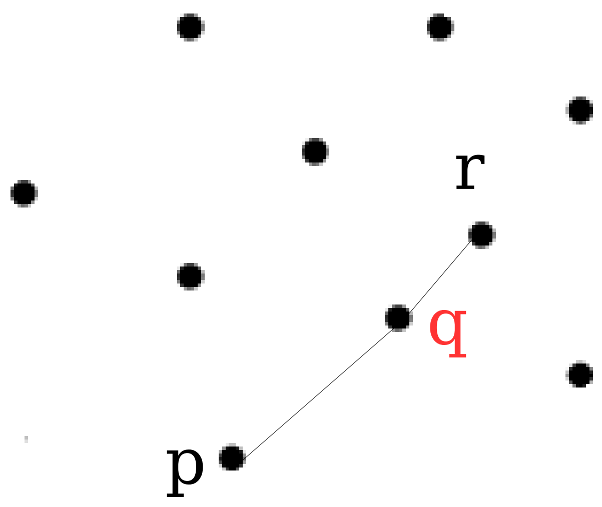
- CCW (counterclockwise) test



CCW(p,q,r) > 0

Computational Geometry

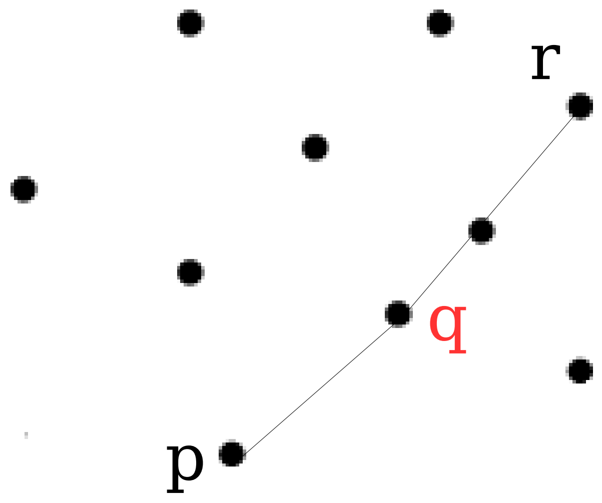
- CCW (counterclockwise) test



CCW(p,q,r) > 0

Computational Geometry

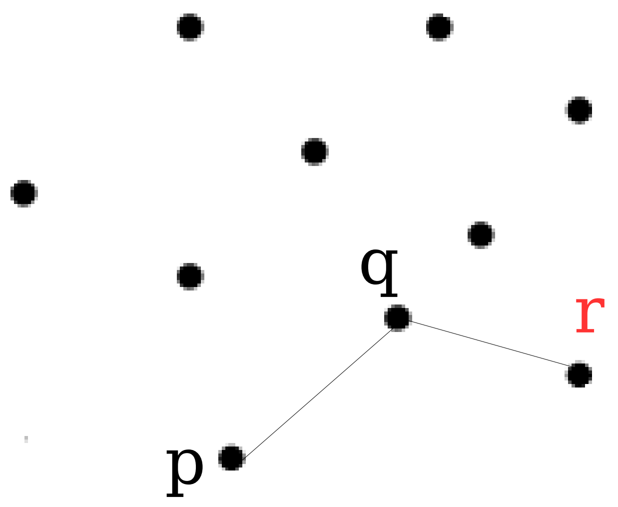
- CCW (counterclockwise) test



CCW(p,q,r) > 0

Computational Geometry

- CCW (counterclockwise) test



CCW(p,q,r) < 0

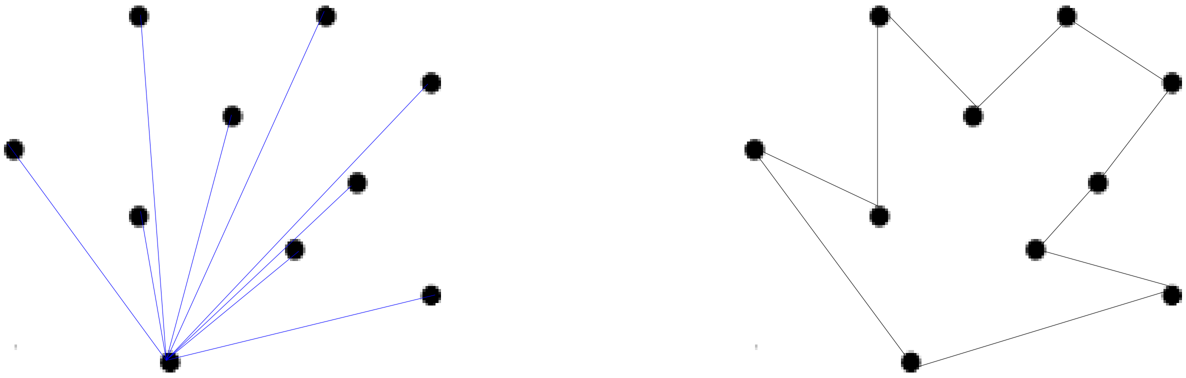
Computational Geometry

- Next point in the convex hull

```
Function Next (Points, p)
    q = p
    for r in Points
        if CCW(p,q,r) <0 or (CCW(p,q,r) = 0 and d(p,r) > d(p,q))
            q= r
    return q
```

Computational Geometry

- Convex Hull – 2D Algorithms
  - Graham Scan algorithm
  - Complexity $O(n \log n)$ due to a pre-processing step.



Computational Geometry

Convex Hull – 2D Algorithms

- Graham Scan algorithm

Step 1: Find the bottom-most point p_0

Step 2: Sort the points in counterclockwise order (or polar angle) with respect to p_0. (use CCW test in the comparison function)

Step 3: Push p_0, p_1 onto stack S

Step 4: For $i = 2$ to $n-1$

While CCW($S(before\_top), S(top), p_i$) < 0

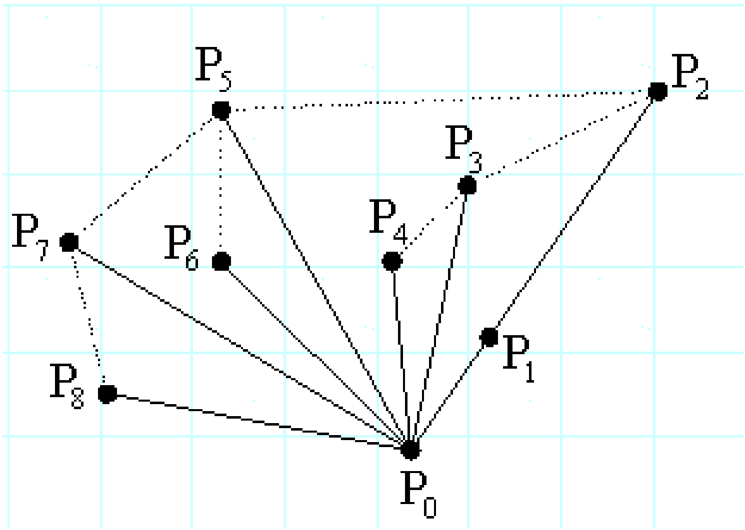
Pop S.

Push p_i onto S.

Computational Geometry

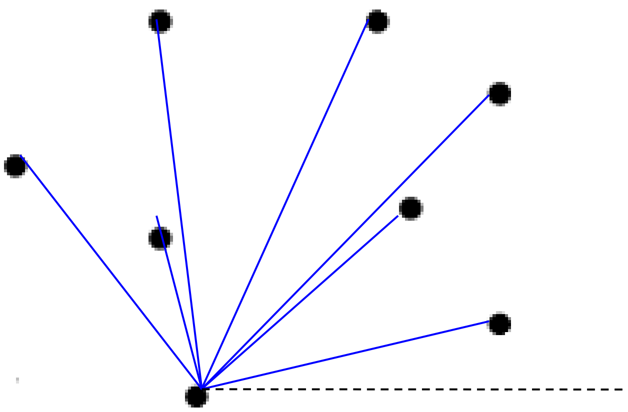
- Convex Hull – 2D Algorithms
  - Graham Scan algorithm

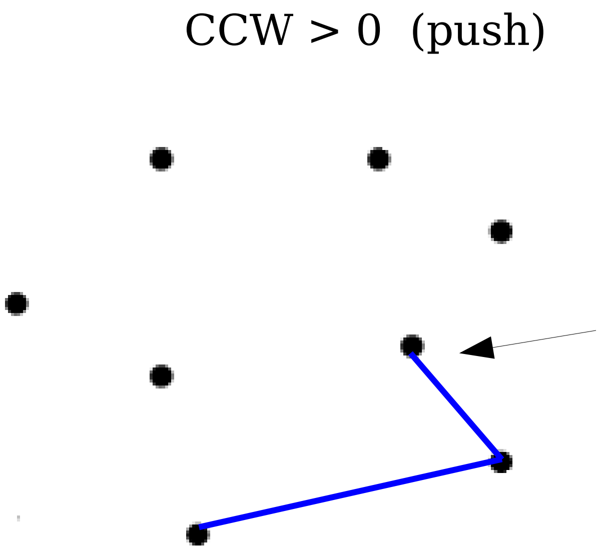
Step 2: Sort the points in counterclockwise order with respect to p_0.



Computational Geometry

- Convex Hull – 2D Algorithms
  - Graham Scan algorithm

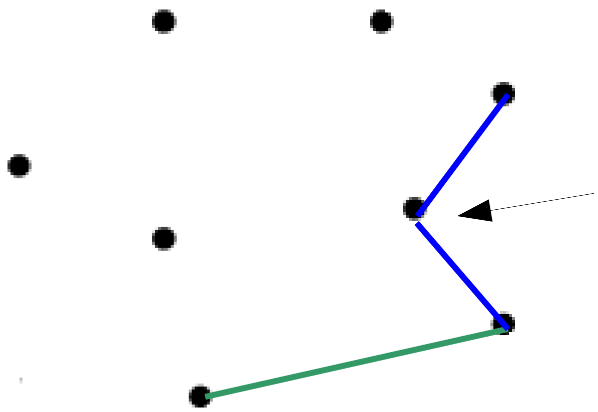




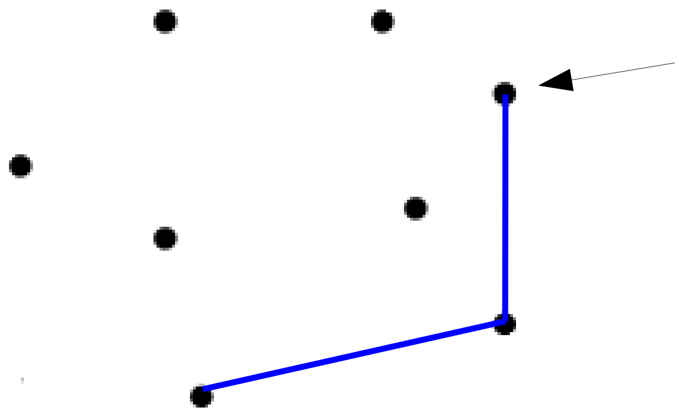
Computational Geometry

- Convex Hull – 2D Algorithms
  - Graham Scan algorithm

CCW < 0 (pop)



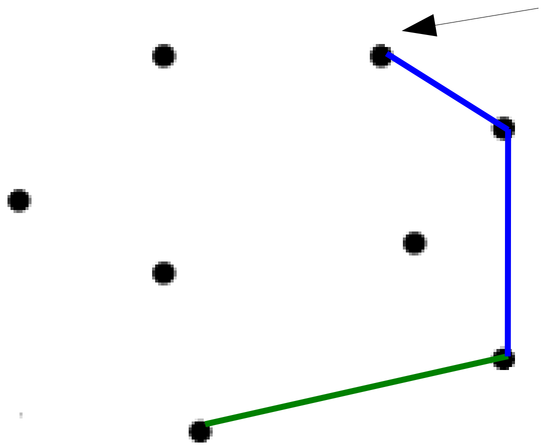
CCW > 0 (push)



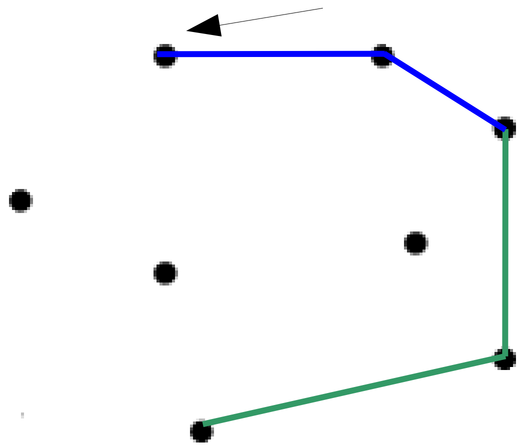
Computational Geometry

- Convex Hull – 2D Algorithms
  - Graham Scan algorithm

CCW > 0 (push)



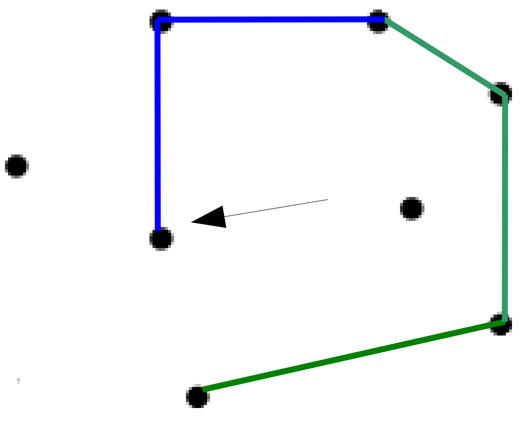
CCW > 0 (push)



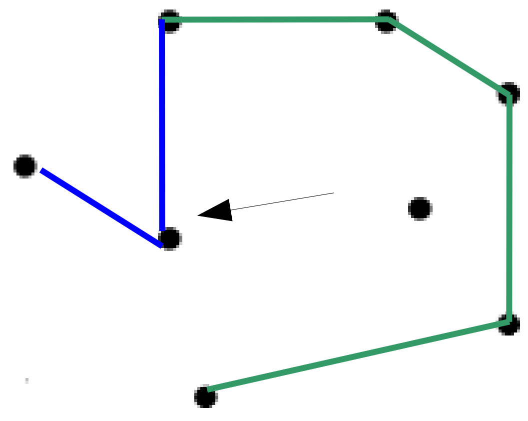
Computational Geometry

- Convex Hull – 2D Algorithms
  - Graham Scan algorithm

CCW > 0 (push)



CCW < 0 (pop)



Computational Geometry

- Convex Hull – 2D Algorithms
  - Graham Scan algorithm

CCW > 0 (push)