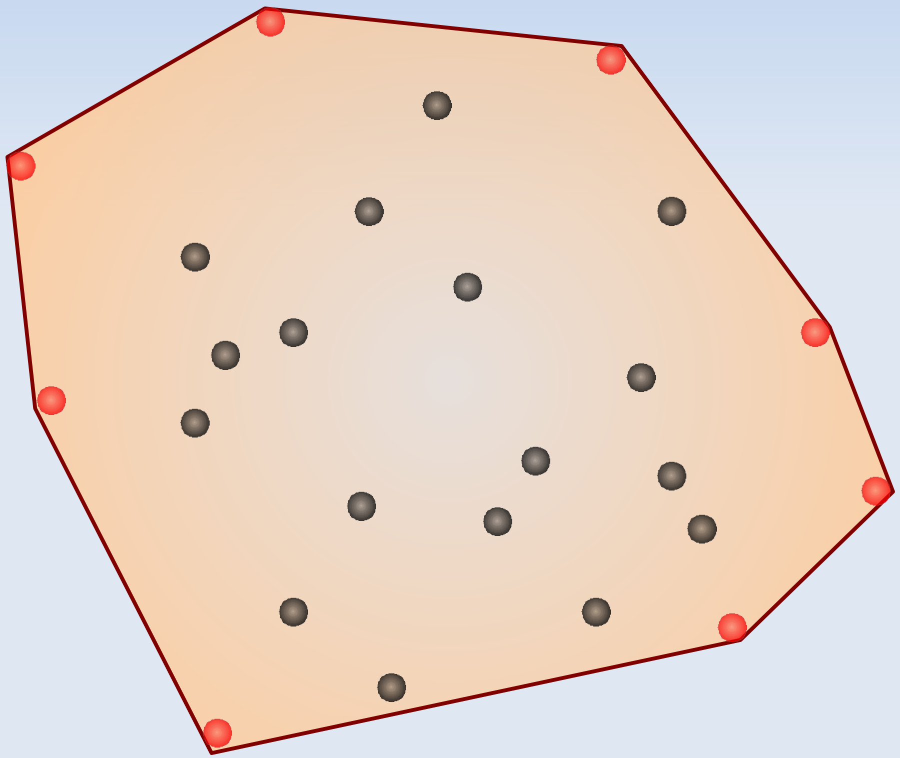
Week 5 Convex Hulls in 2D

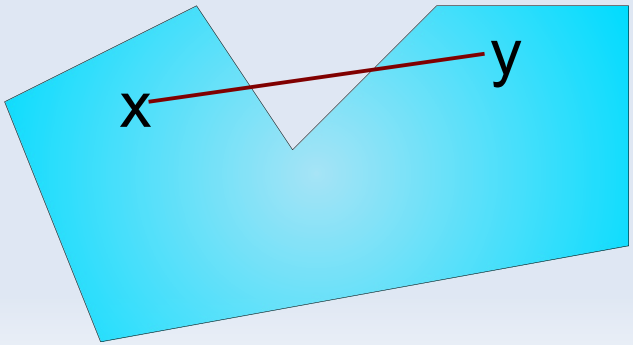


Week 5 Convex Hulls in 2D

- Applications
  - Collision avoidance
  - Fitting ranges with a line
  - Smallest box
  - Shape Analysis

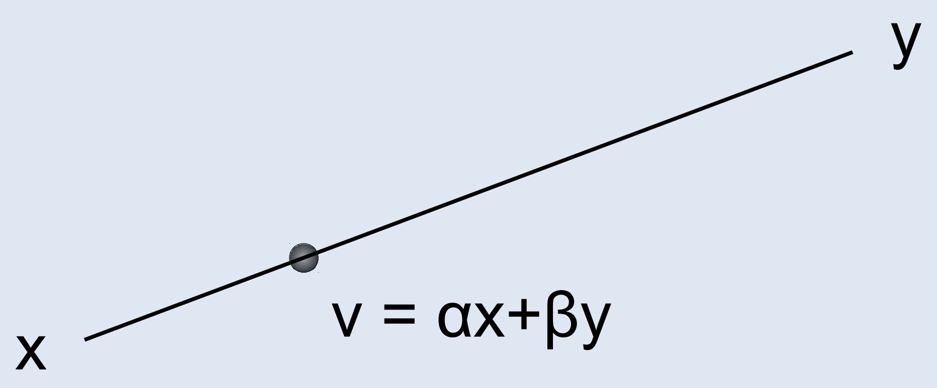
Convexity

- A set S is convex,
  - if x,y in S implies that
  - the segment xy is a subset of S
- Works in any dimensions
- A polygon with a reflex vertex is not convex



Formal Segment Definition

- The segment xy is the set of all points of the form $\alpha x+\beta y$ with $\alpha\geq 0$, $\beta\geq 0$ and $\alpha+\beta = 1$
- $\alpha x+\beta y = \alpha x+(1-\alpha)y = \alpha(x-y) + y = x + \beta(y-x)$

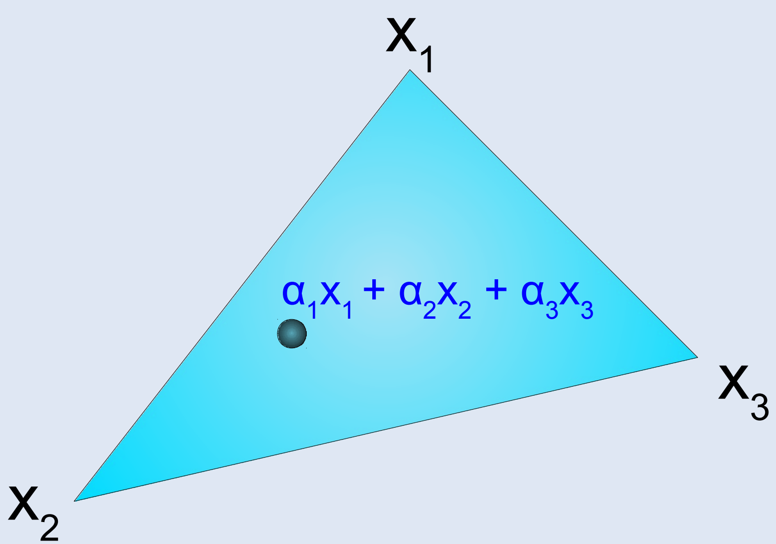


$$\frac{|xv|}{|xy|}=\beta$$

$$\frac{|vy|}{|xy|}=\alpha$$

Convex Combination

- A convex combination of points $x_1, x_2, \ldots, x_k$ is
  - a sum of the form $\alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_k x_k$
  - $\alpha_i \geq 0$ for all I
  - $\alpha_1 + \alpha_2 + \ldots + \alpha_k = 1$



Convex Hull

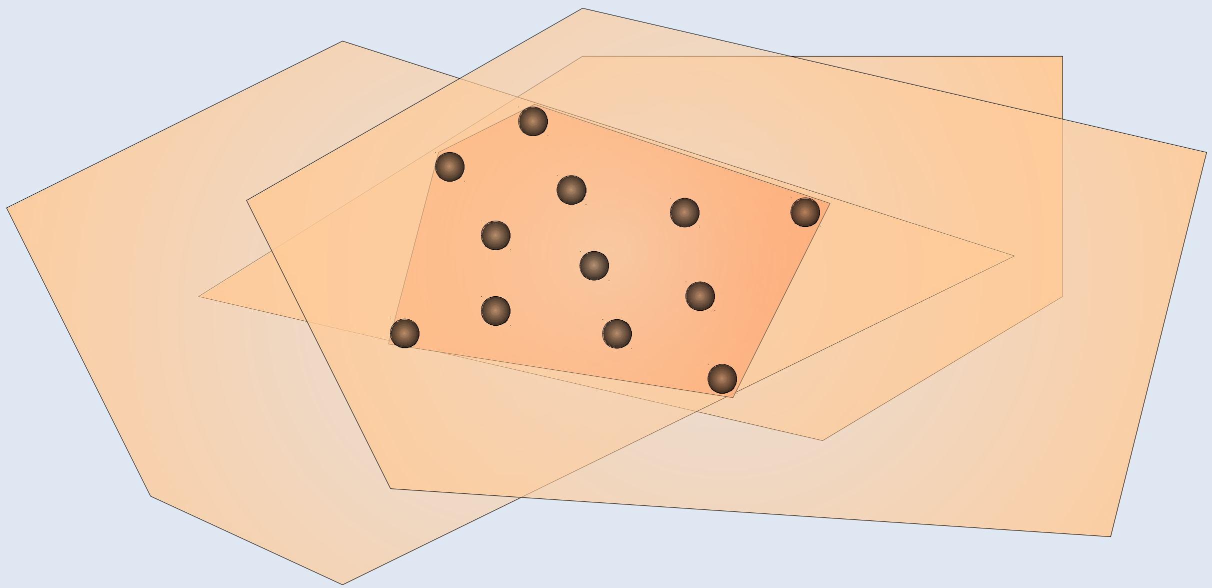
- Definition I
  - The convex hull of a set of points S is
    - the set of all convex combinations of points of S

Convex Hull

- Definition II
  - The convex hull of a set of points S
    - in d dimensions is
    - the set of all convex combinations of d+1 (or fewer) points of S
- For d=2, convex hull is the combination of all triangles

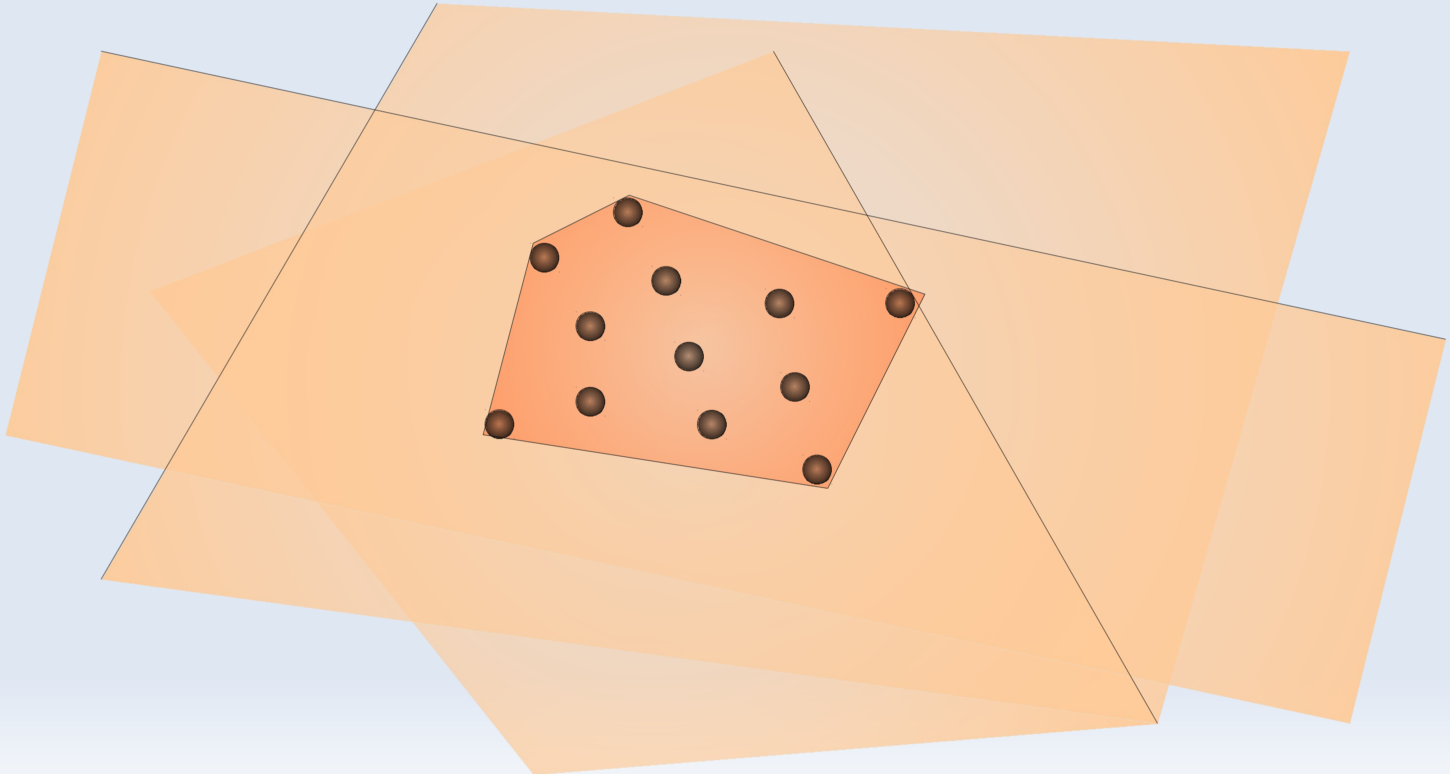
Convex Hull

- Definition III
  - The convex hull of a set of points S is
    - the intersection of all convex sets that contain S



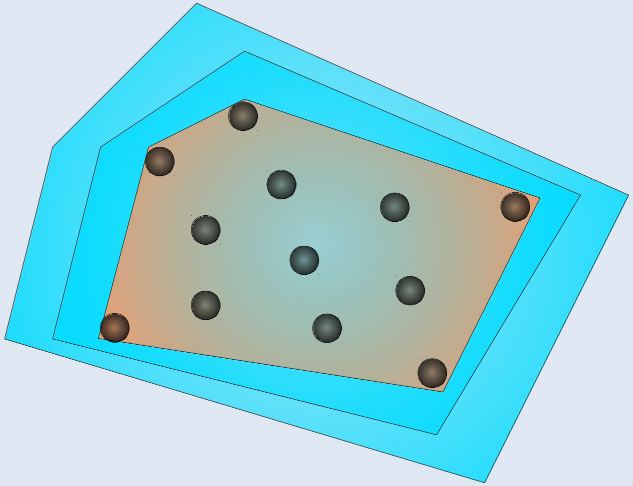
Convex Hull

- Definition IV
  - The convex hull of a set of points S is
    - the intersection of all halfspaces that contain S



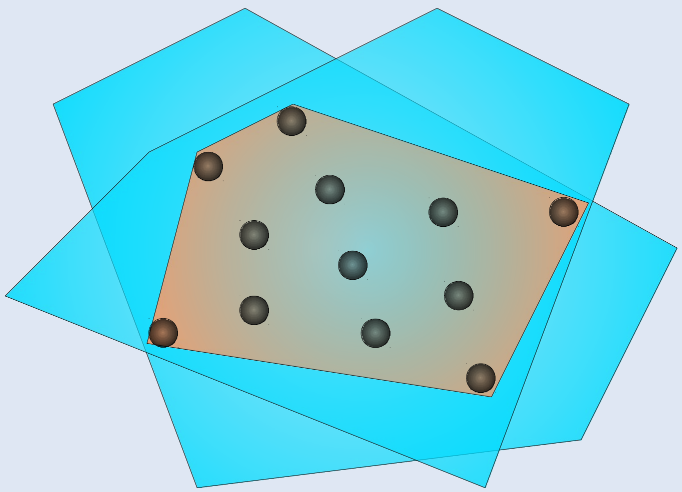
Convex Hull

- Definition V
  - The convex hull of a set of points S is
    - the smallest convex polygon that encloses S



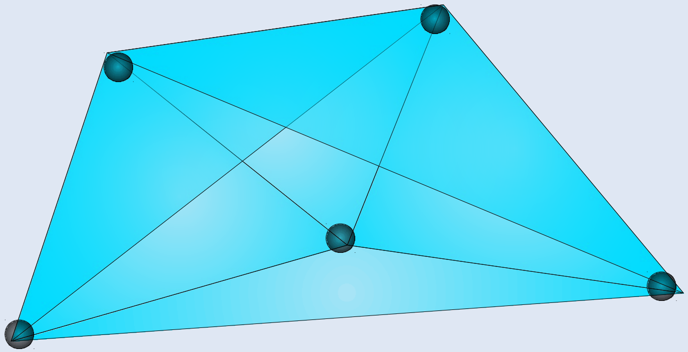
Convex Hull

- Definition VI
  - The convex hull of a set of points S is
    - the enclosing convex polygon with smallest area



Convex Hull

- Definition VII
  - The convex hull of a set of points S in the plane is
    - the union of all the triangles determined by points in S



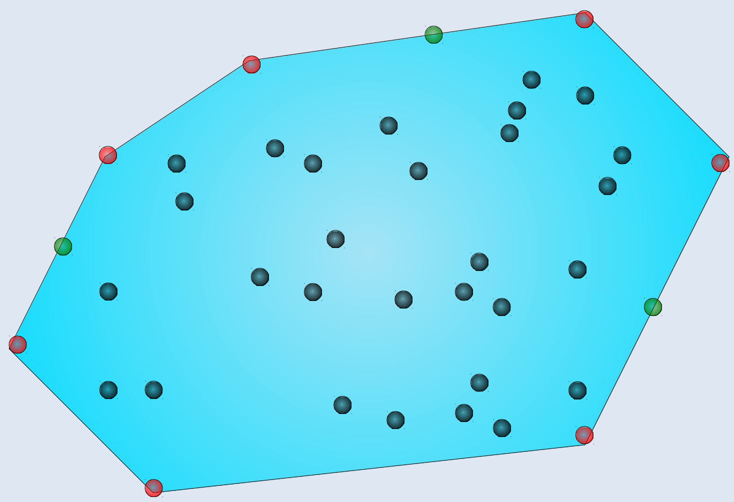
Output

- A convex hull algorithm may have the following outputs:
  - all the points on the hull, in arbitrary order
  - extreme points, in arbitrary order
  - all the points on the hull, in BTO*
  - extreme points, in BTO

* BTO: Boundary Traversal Order

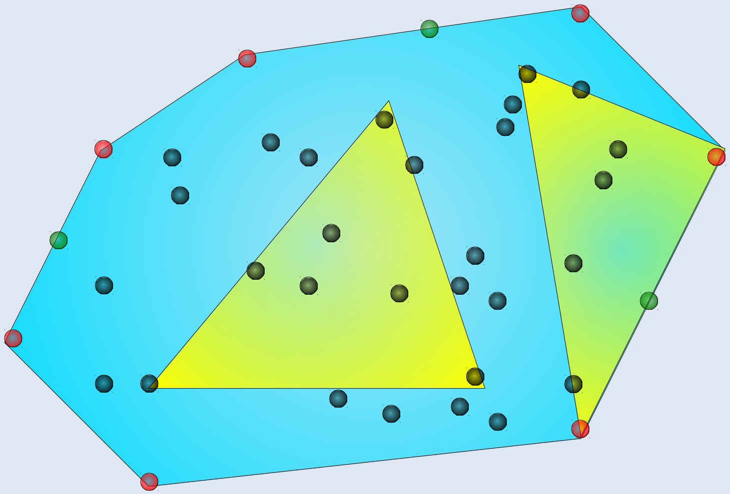
Extreme Points

- Extreme points
  - are the vertices of the convex hull at which the interior angle is strictly convex



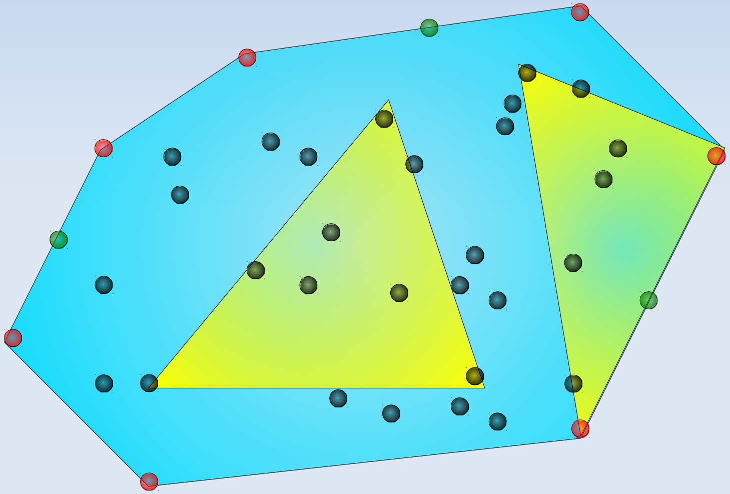
Naive Algorithms

- Nonextreme Points
  - A point is nonextreme iff it is inside some triangle whose vertices are points of the set and is not itself a corner of that triangle



Naive Algorithms

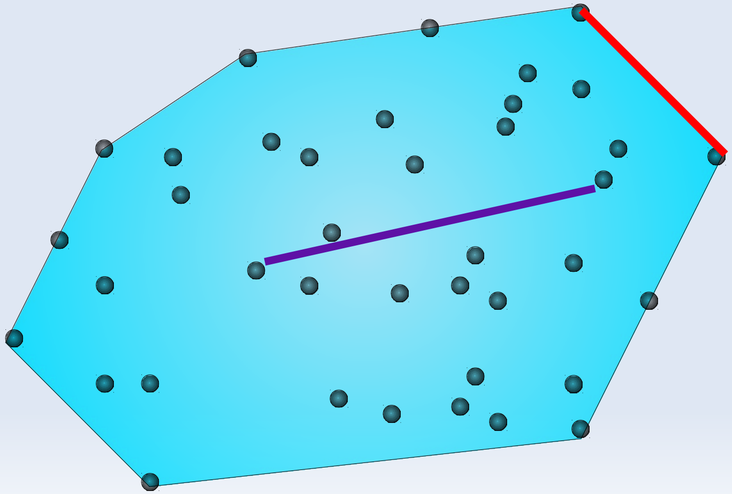
- Algorithm: INTERIOR POINTS
  - **for** each i **do**
  - **for** each j ≠ i **do**
  - **for** each k ≠ j ≠ i **do**
    - **for** each l ≠ k ≠ j ≠ i **do**
      - **if** p_l in triangle $p_i p_j p_k$ **then** p_l is nonextreme



- $O(n^4)$!!!

Extreme Edges

- Algorithm: EXTREME EDGES
  - **for** each i **do**
  - **for** each j ≠ i **do**
    - **for** each k ≠ j ≠ i **do**
      - **if** p_k is not (left or on) (p_i,p_j) **then** (p_i,p_j) is not extreme

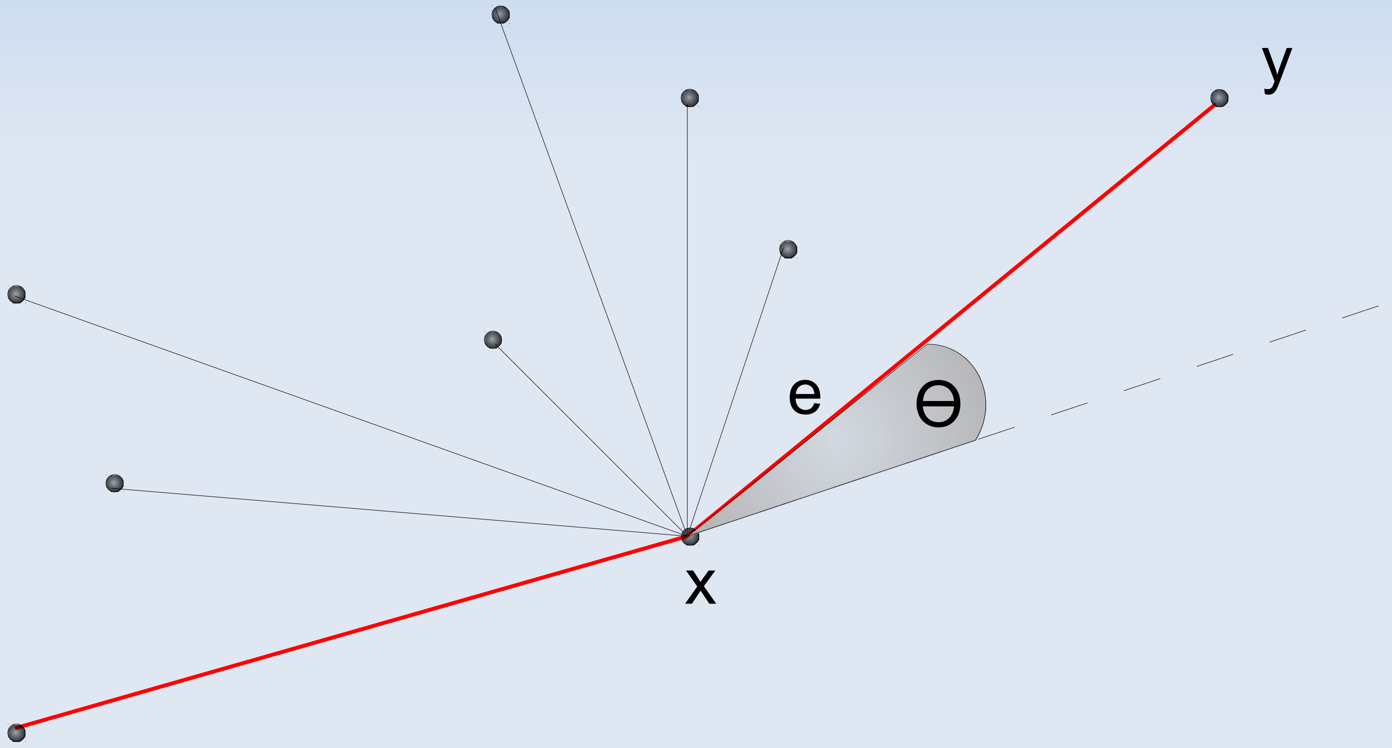


$O(n^3)$, still very slow

Gift Wrapping

- All edges of the convex hull are connected
- Find one, then search for the next
  - Use the lowest vertex to start with
- Works much faster: O(nh)

Gift Wrapping



Gift Wrapping

- For each edge
  - Compute theta with O(n) vertices
  - Choose the vertex with the smallest theta
- There are only h edges on the boundary
  - O(n) time for each edge
  - Overall running time O(nh)

Quickhull

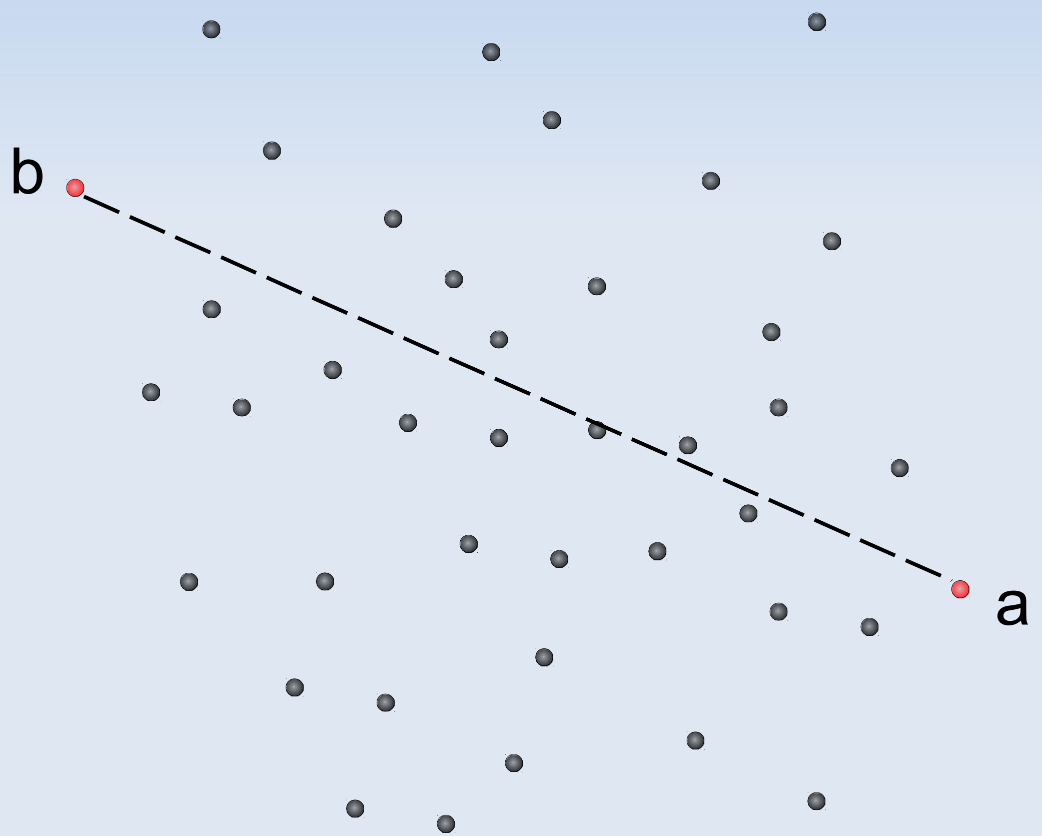
- Similar to Quicksort
- It is easy to discard many points
  - But may not work always
- Running time:
  - Best case: $O(n \log n)$
  - Worst case: $O(n^2)$

Quickhull

- Algorithm: QUICKHULL(a, b, S)
  - If $S = \varnothing$ then return
  - else
    - c ← index of point with max distance from ab
    - A ← points strictly right of (a, c)
    - B ← points strictly right of (c, b)
    - return QUICKHULL(a, c, A) + (c) + QUICKHULL(c, b, B)

Quickhull

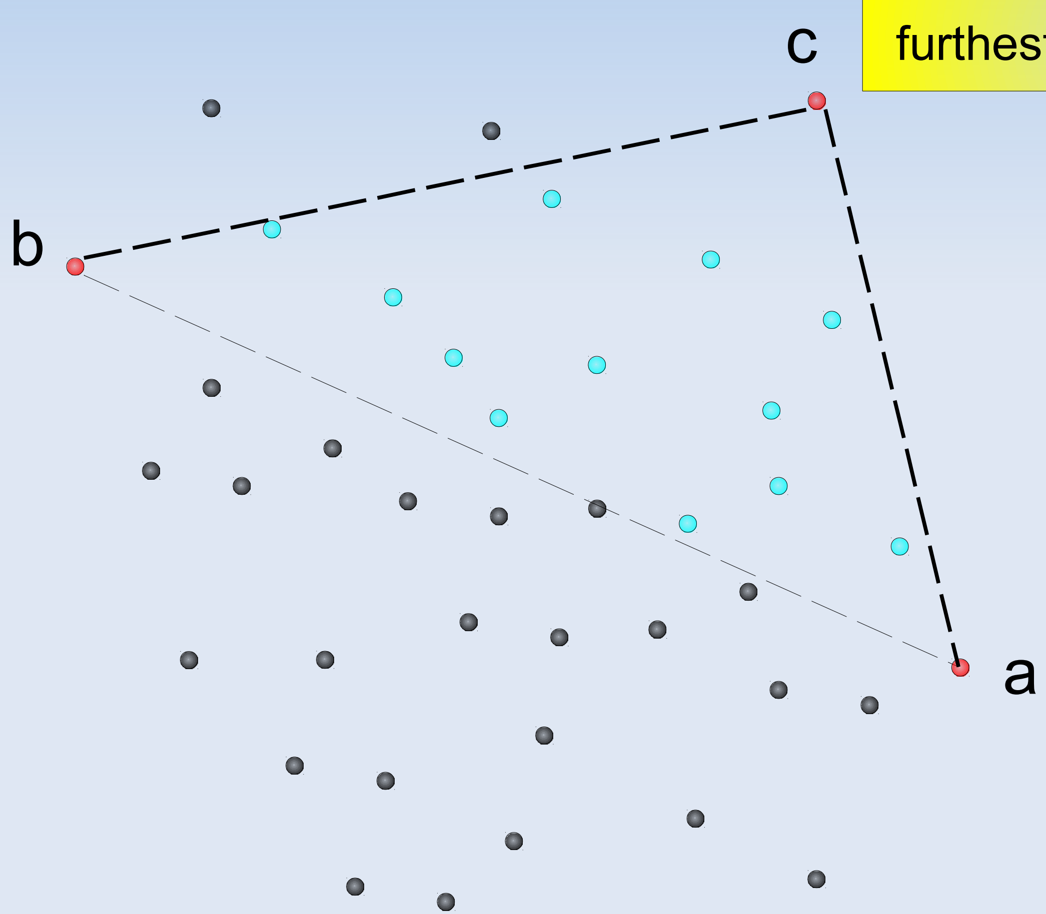
Start with the leftmost and rightmost vertices



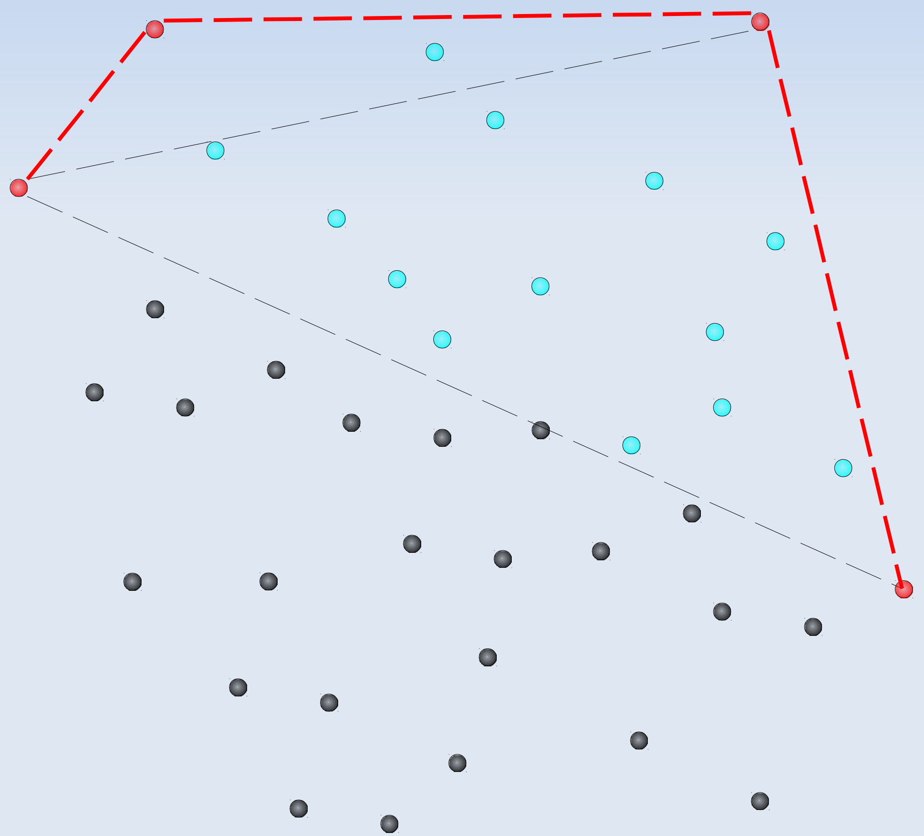
Quickhull

Points in the triangle acb are interior to the hull

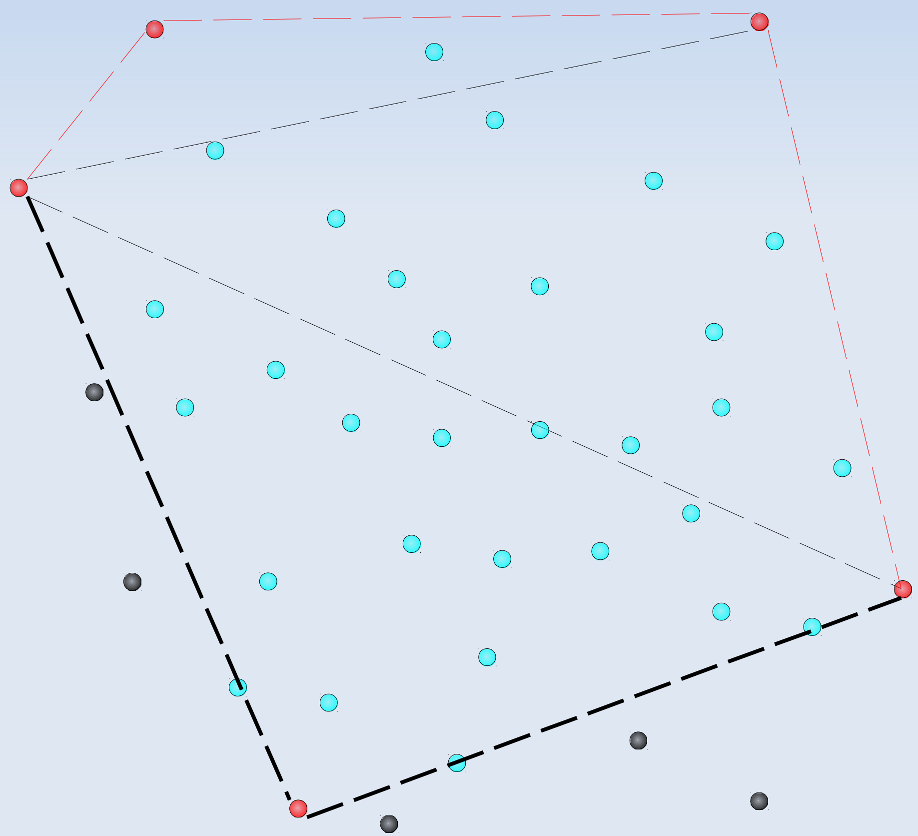
furthest away from ab



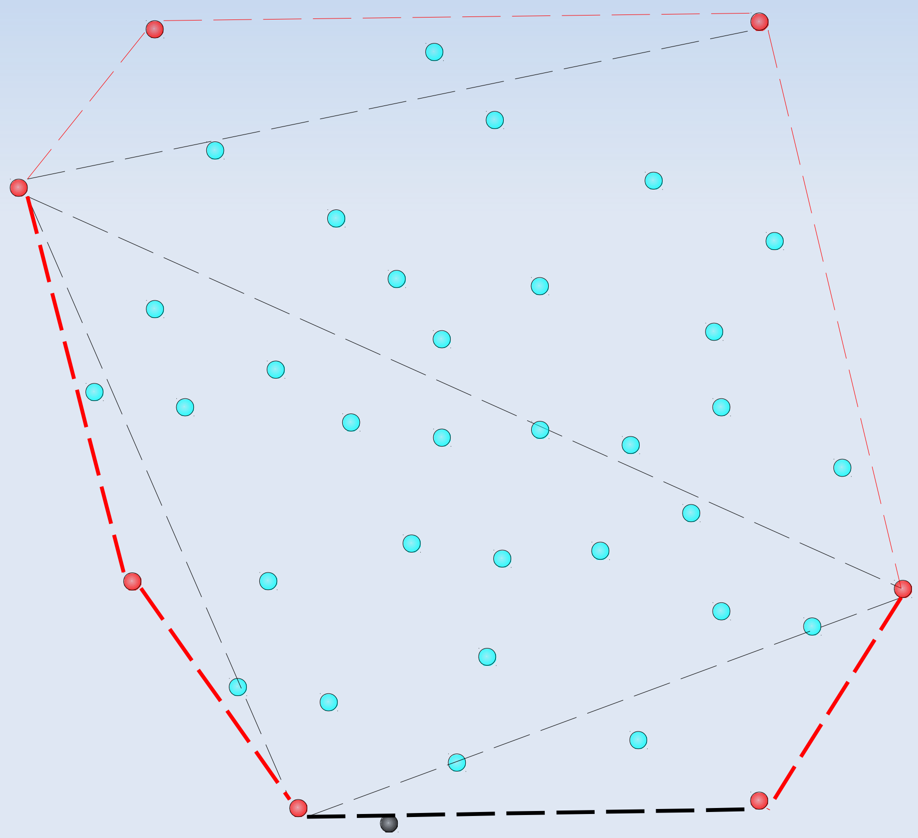
Quickhull



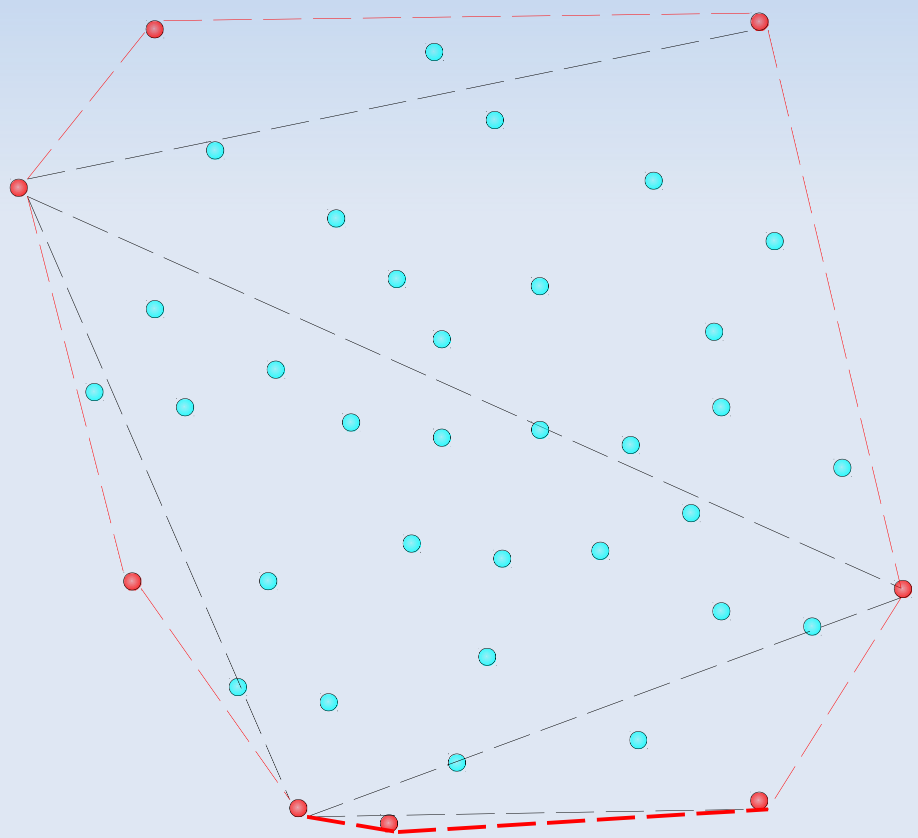
Quickhull



Quickhull



Quickhull



Quickhull

- Running Time:
  - Initial extremes a,b and separating S: $O(n)$
  - Finding extreme point c and eliminating points in triangle acb: $O(n)$
    - Recursive steps: $T(n) = O(n) + T(\alpha) + T(\beta)$
    - Best case: $T(n) = 2T(n/2) + O(n) = O(n \log n)$
    - Worst case: $T(n) = O(n) + T(n-1) = O(n^2)$

Graham's Scan

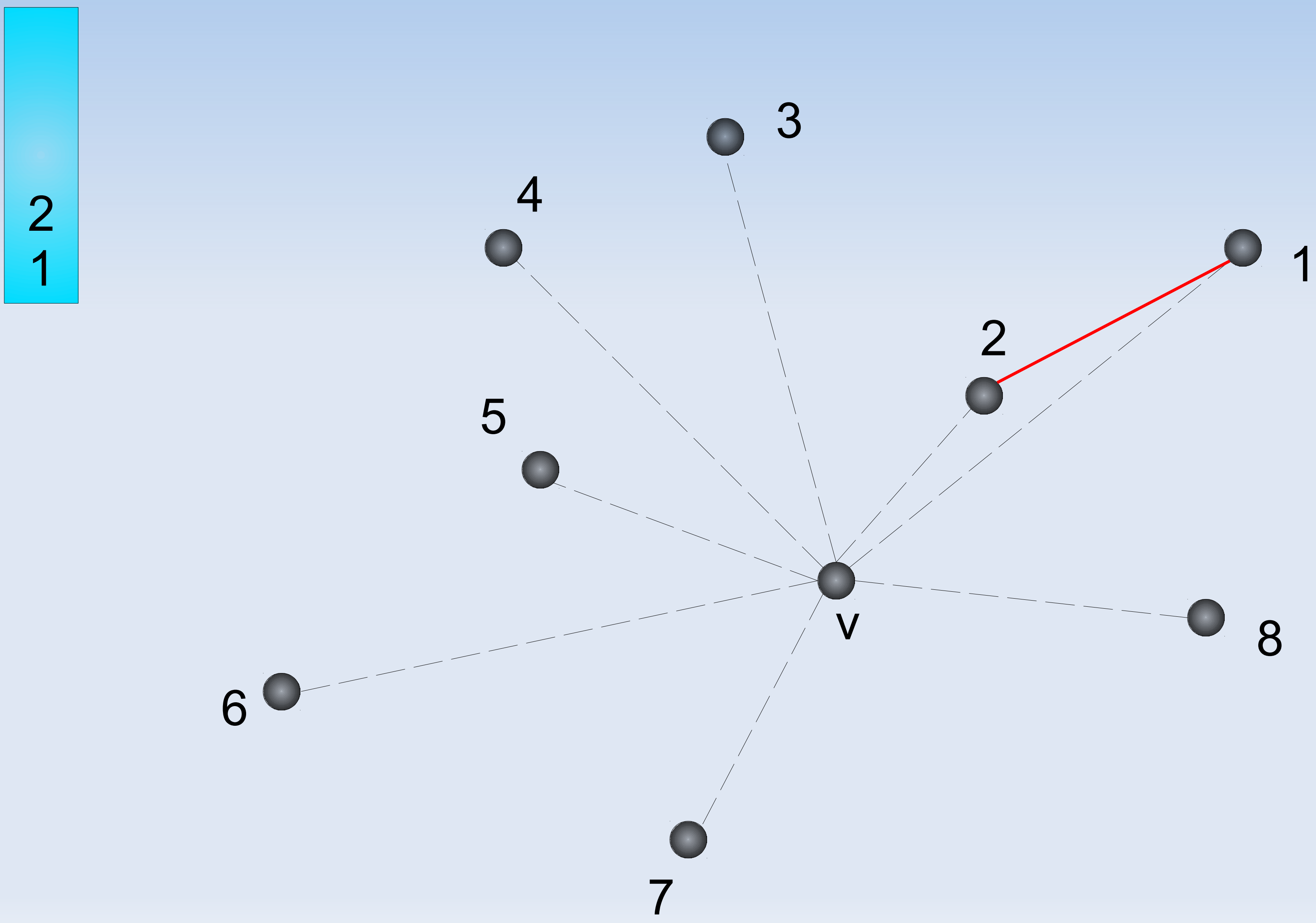
- Probably the first scientific paper in the history of Computational Geometry (1972)
- Bell laboratories required the hull of ~= 10,000 vertices
  - The O(n2) algorithm took too much time
    - n2 = 100,000,000
  - Graham invented this O(n log n) algorithm
    - n log n = 133,000

Outline

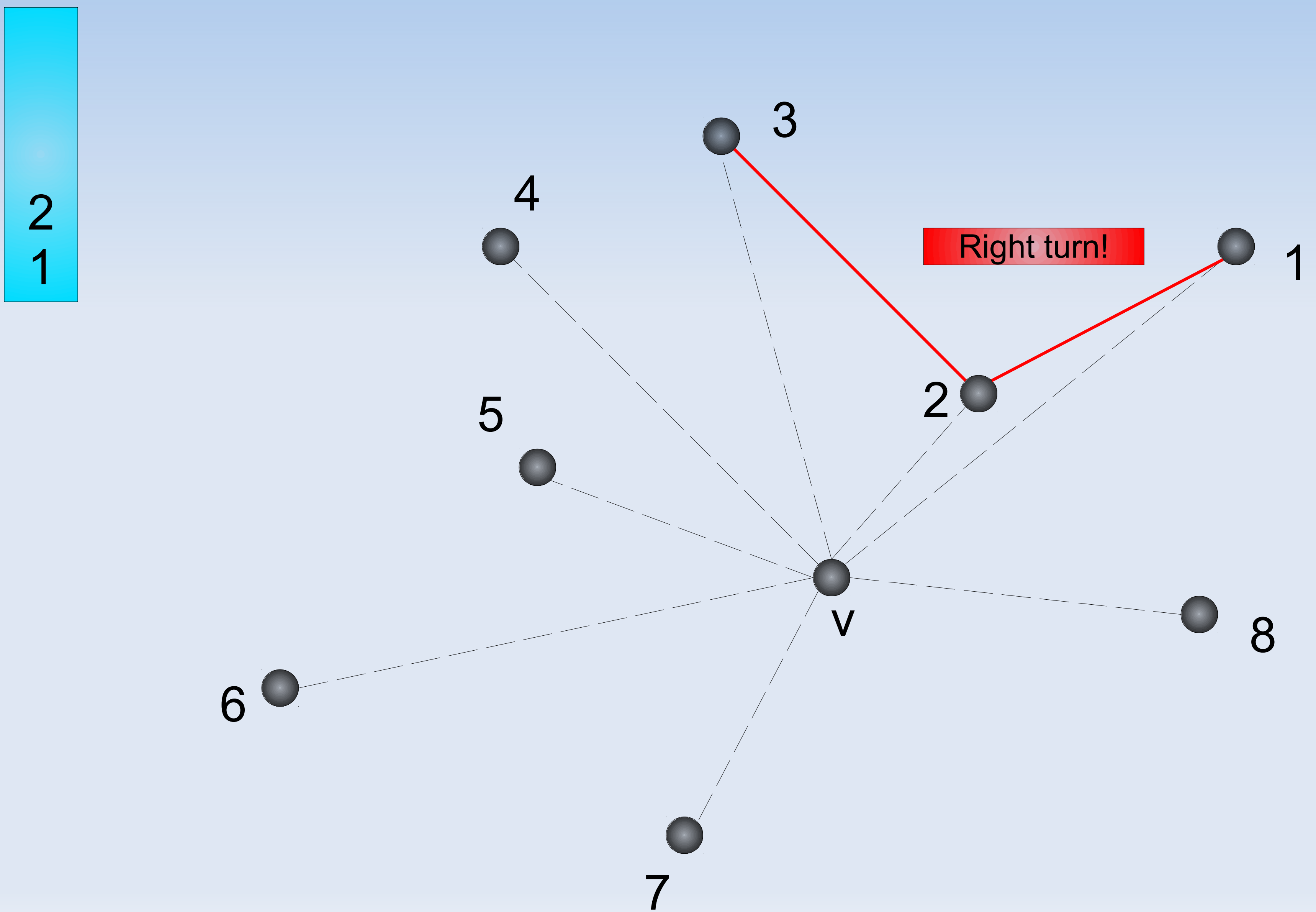
- Choose a vertex v inside the hull
- Sort the rest of the vertices in counter clockwise angular order around v
- Build the hull via left turns at each hull vertex
  - All turns on the convex hull are left turns during a boundary traversal

Graham's Scan

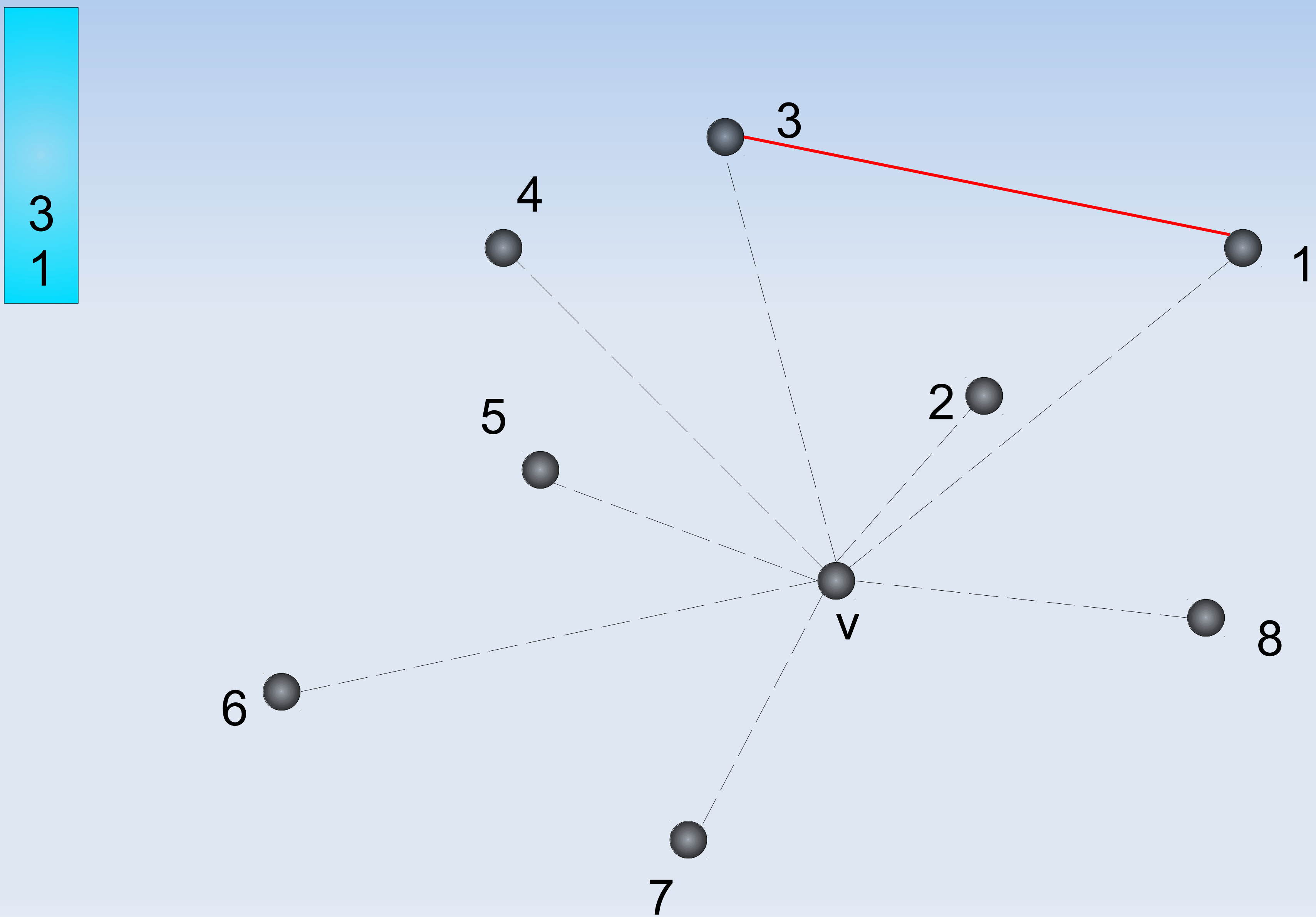
Graham's Scan



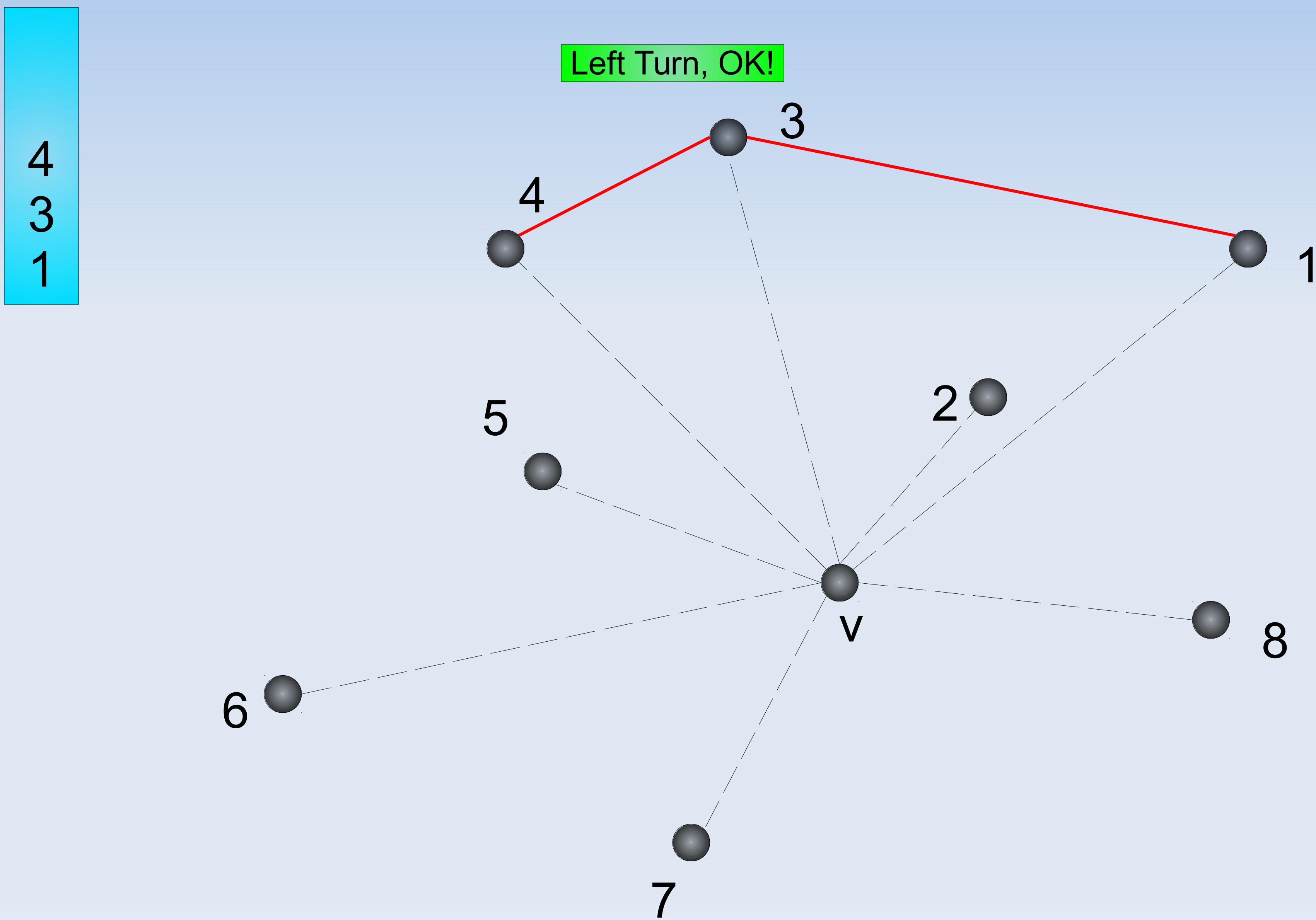
Graham's Scan



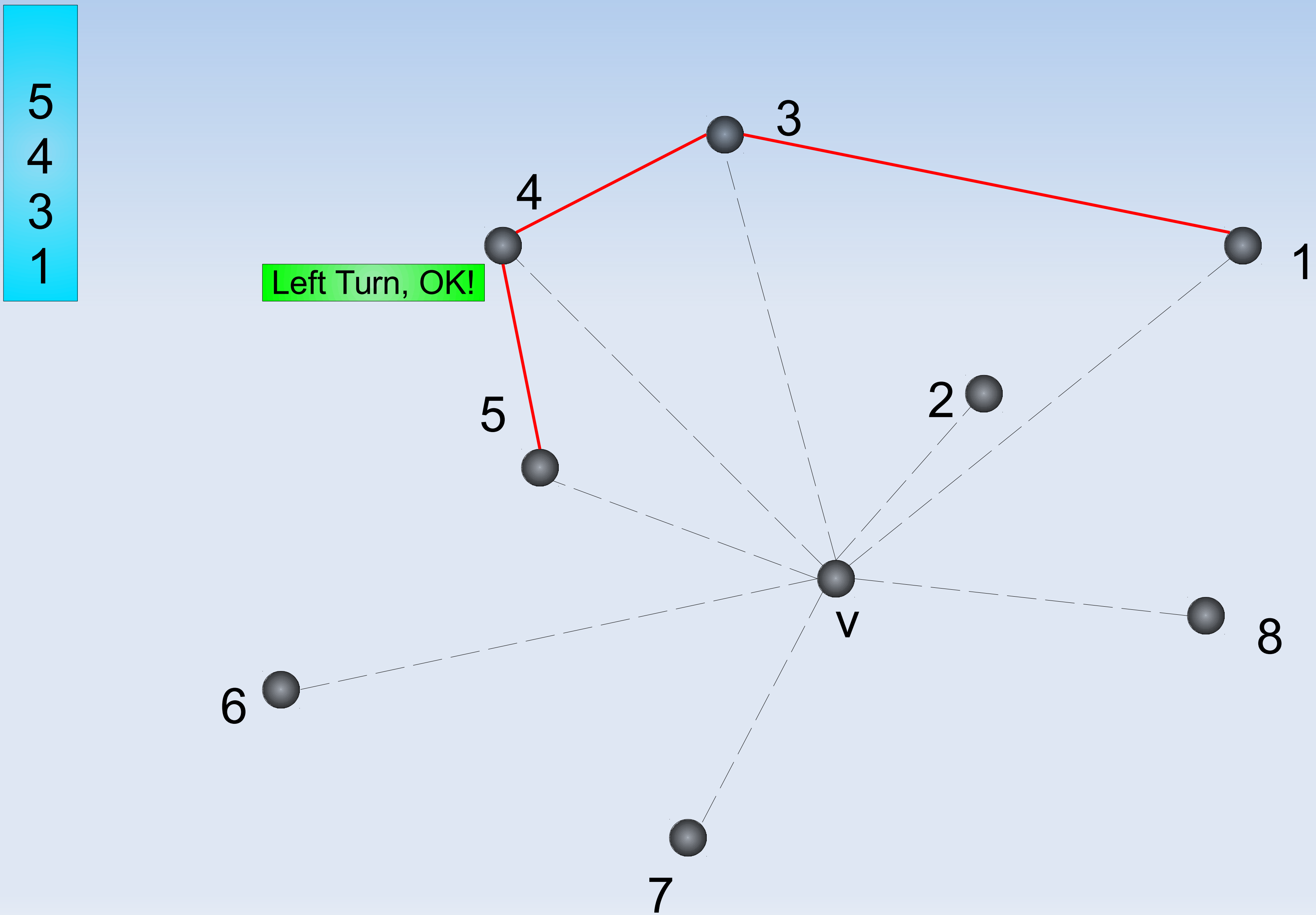
Graham's Scan



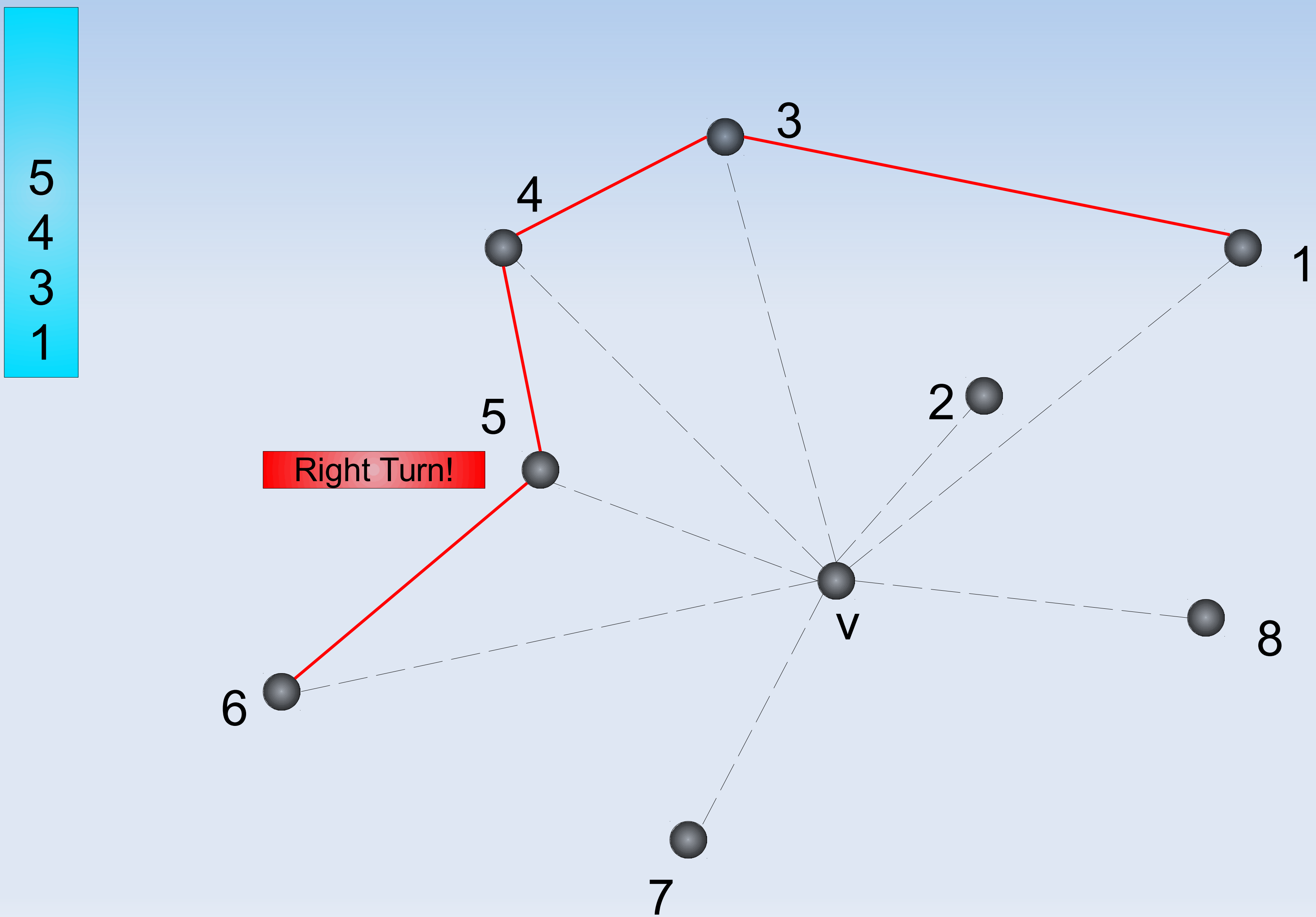
Graham's Scan



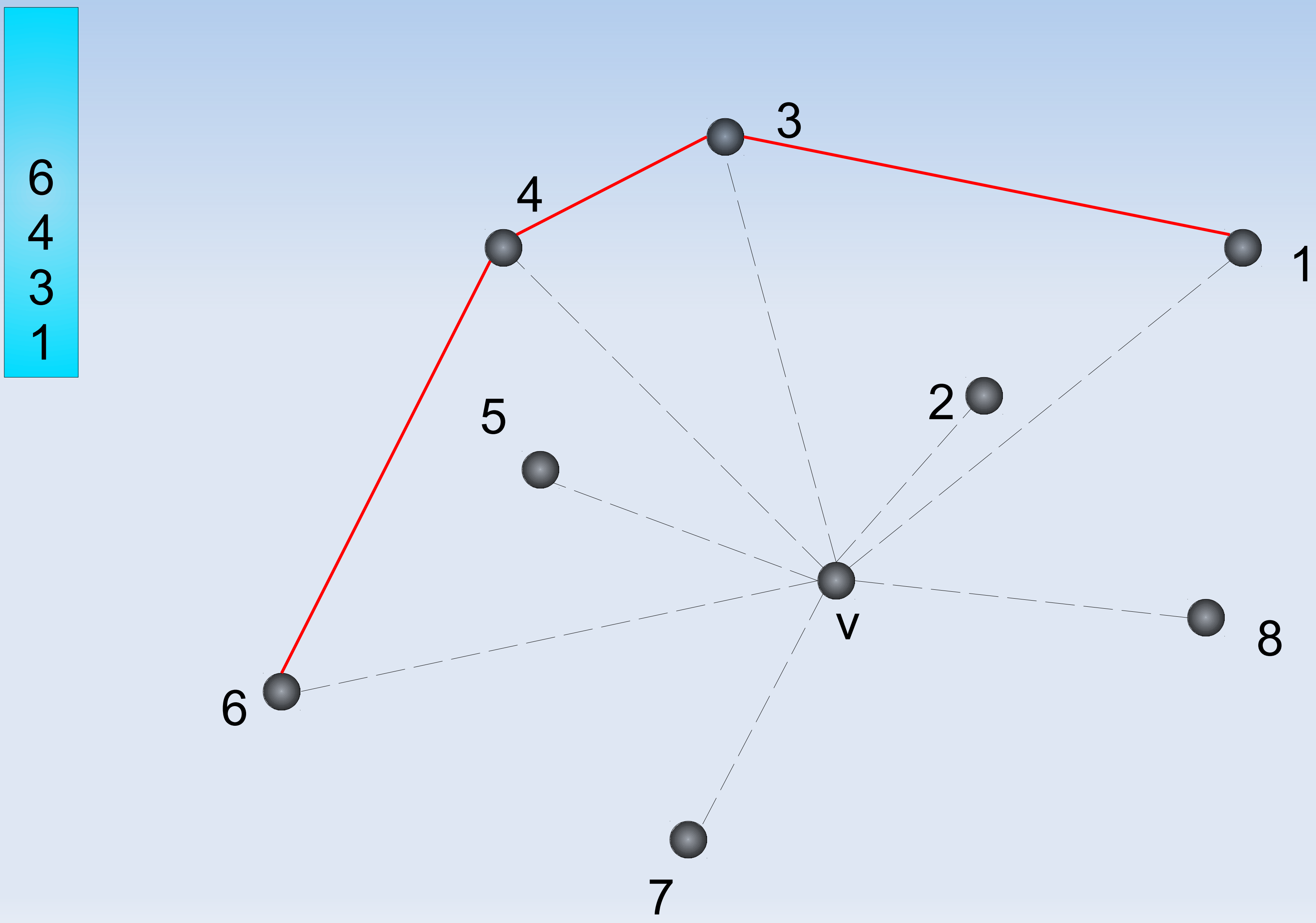
Graham's Scan



Graham's Scan

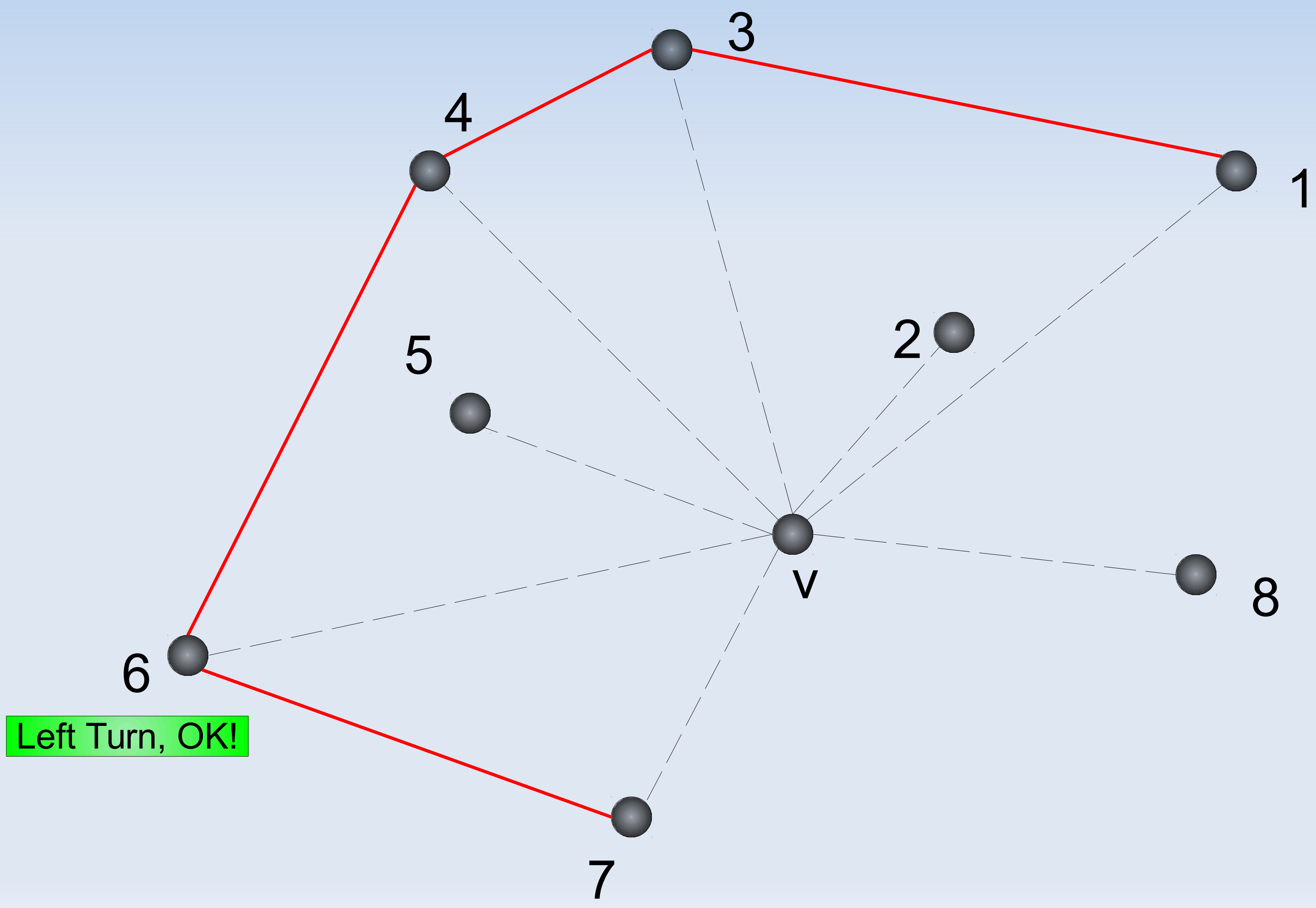


Graham's Scan



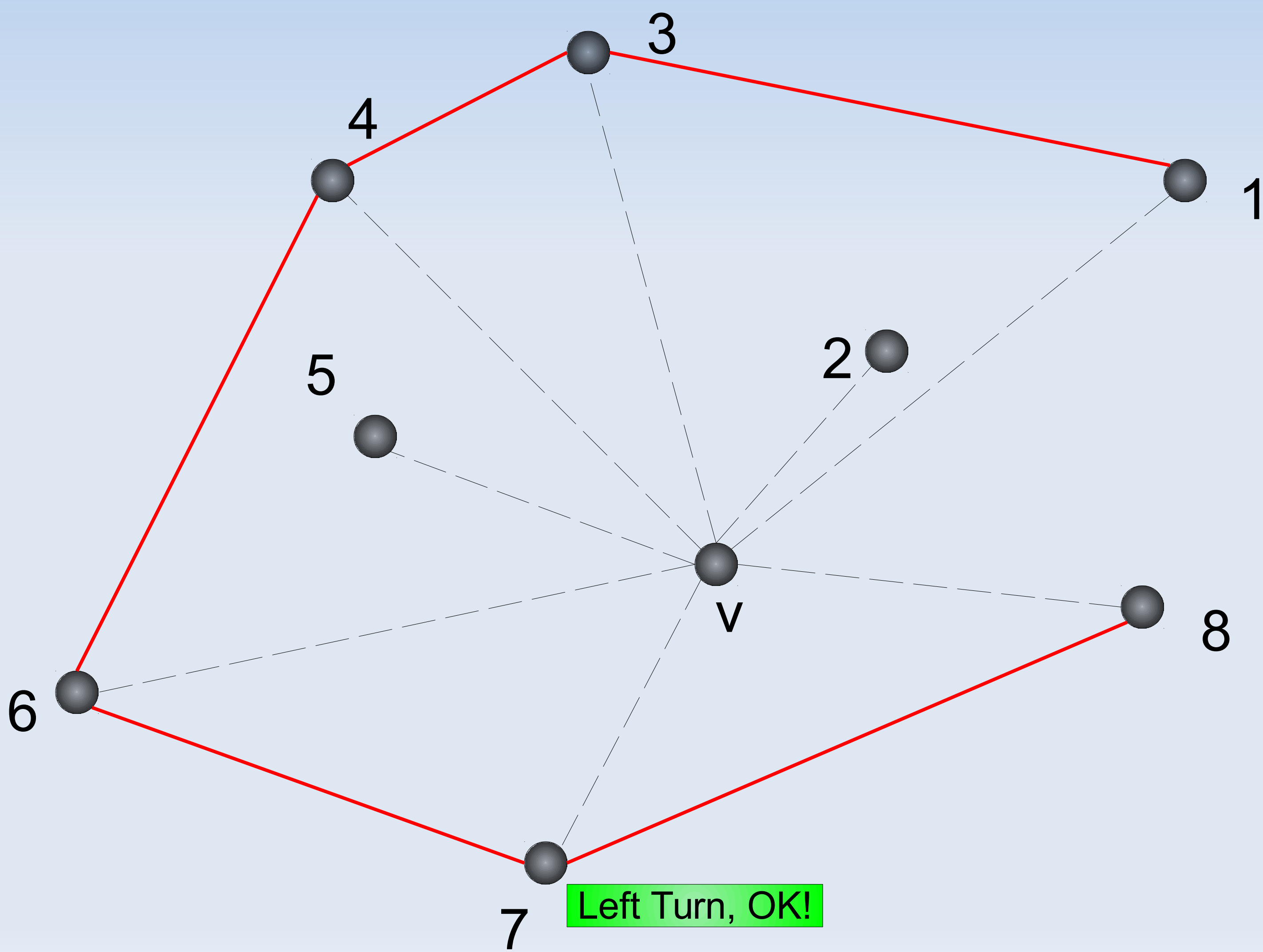
Graham's Scan

7
6
4
3
1

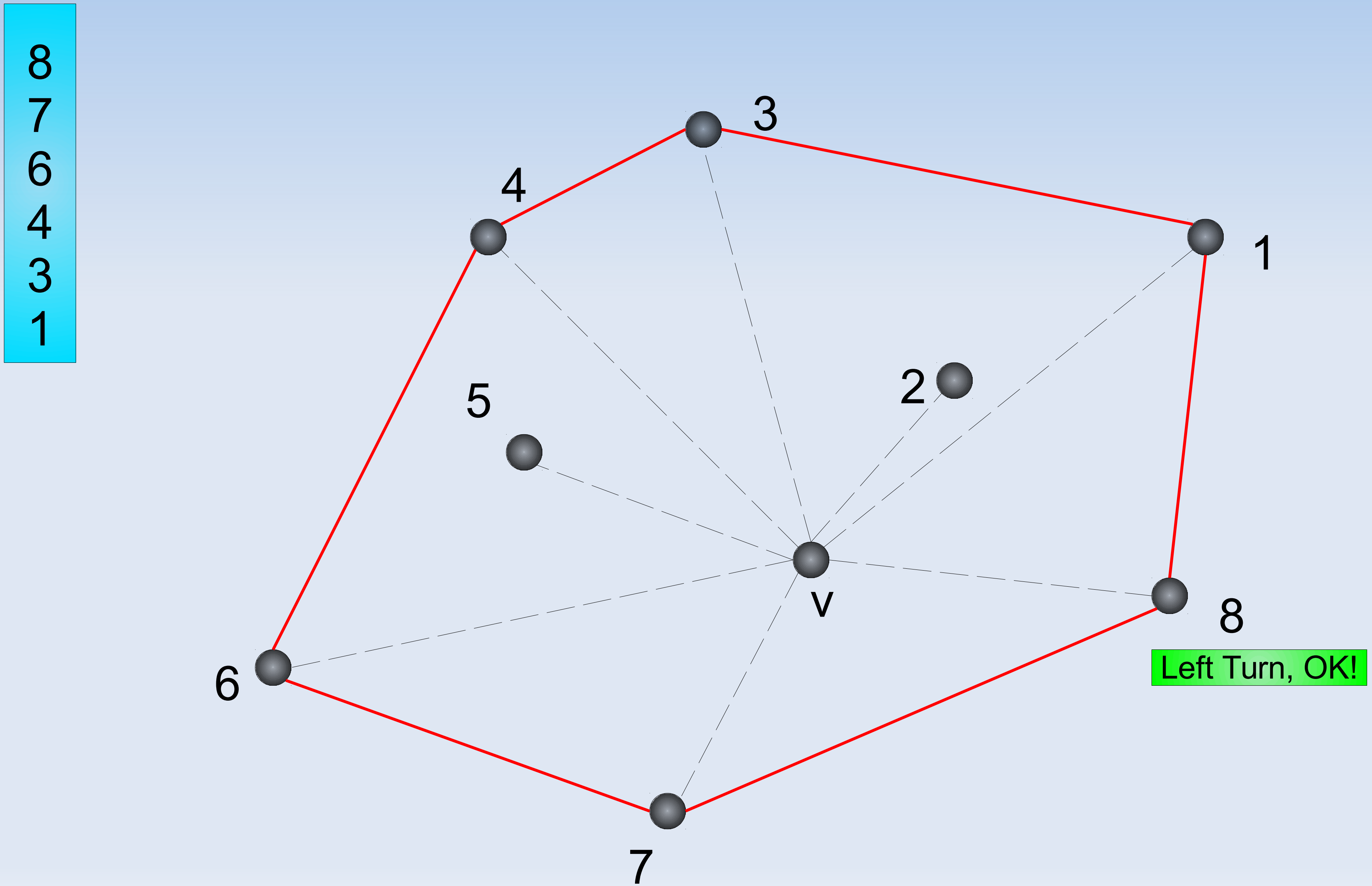


Graham's Scan

8
7
6
4
3
1



Graham's Scan



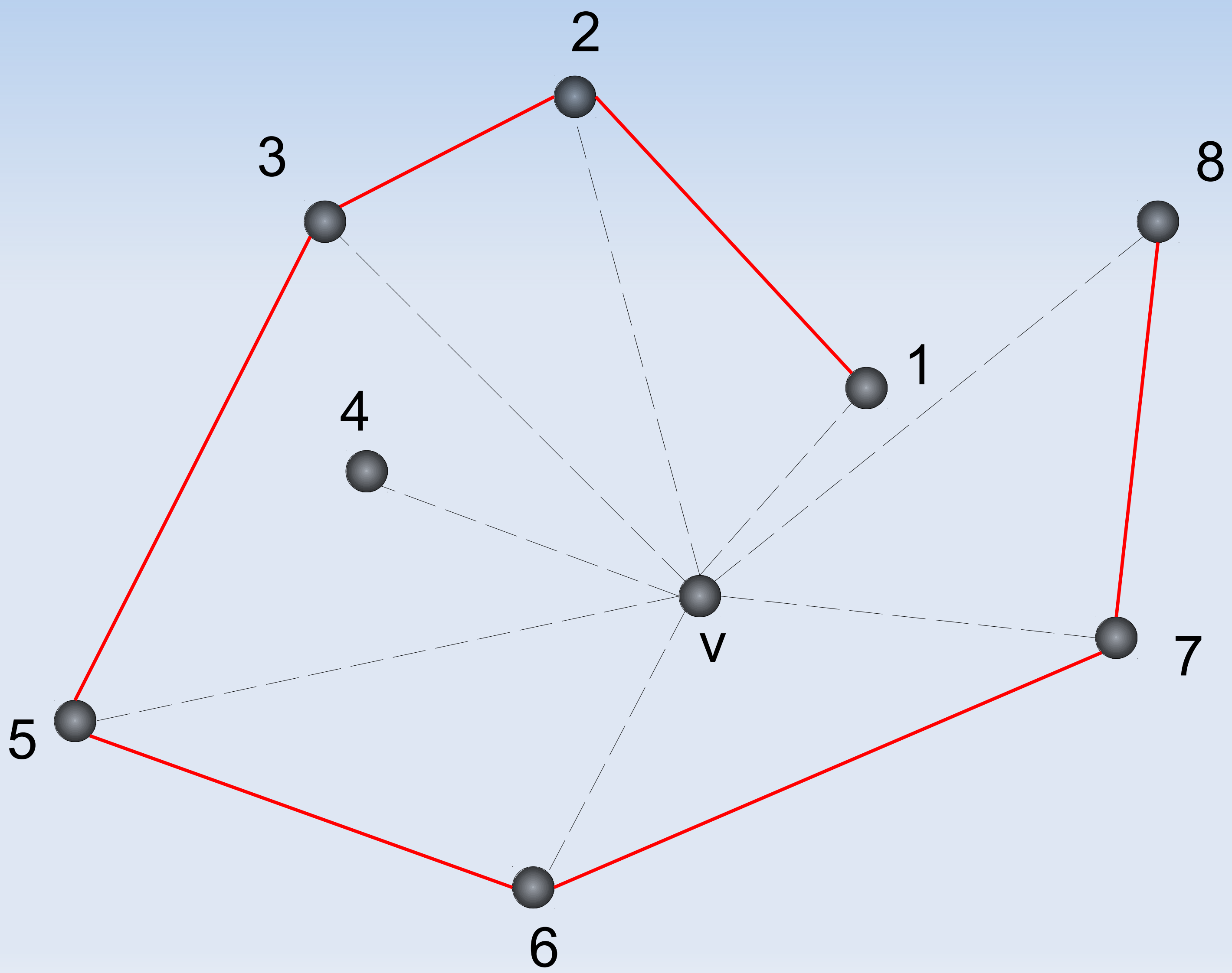
Running Time

- Sorting the vertices: O(n log n)
- Each vertex is added to the stack once
  - May be removed only once
- Overall running time: O(n log n)

Some Problems

- The algorithm so far is not perfect
  - What happens at the end if the bottom vertex is not a hull vertex?
  - What happens at the start if the second vertex is not a hull vertex?
  - Which vertex to use as origin for sorting?
  - How about collinear vertices?

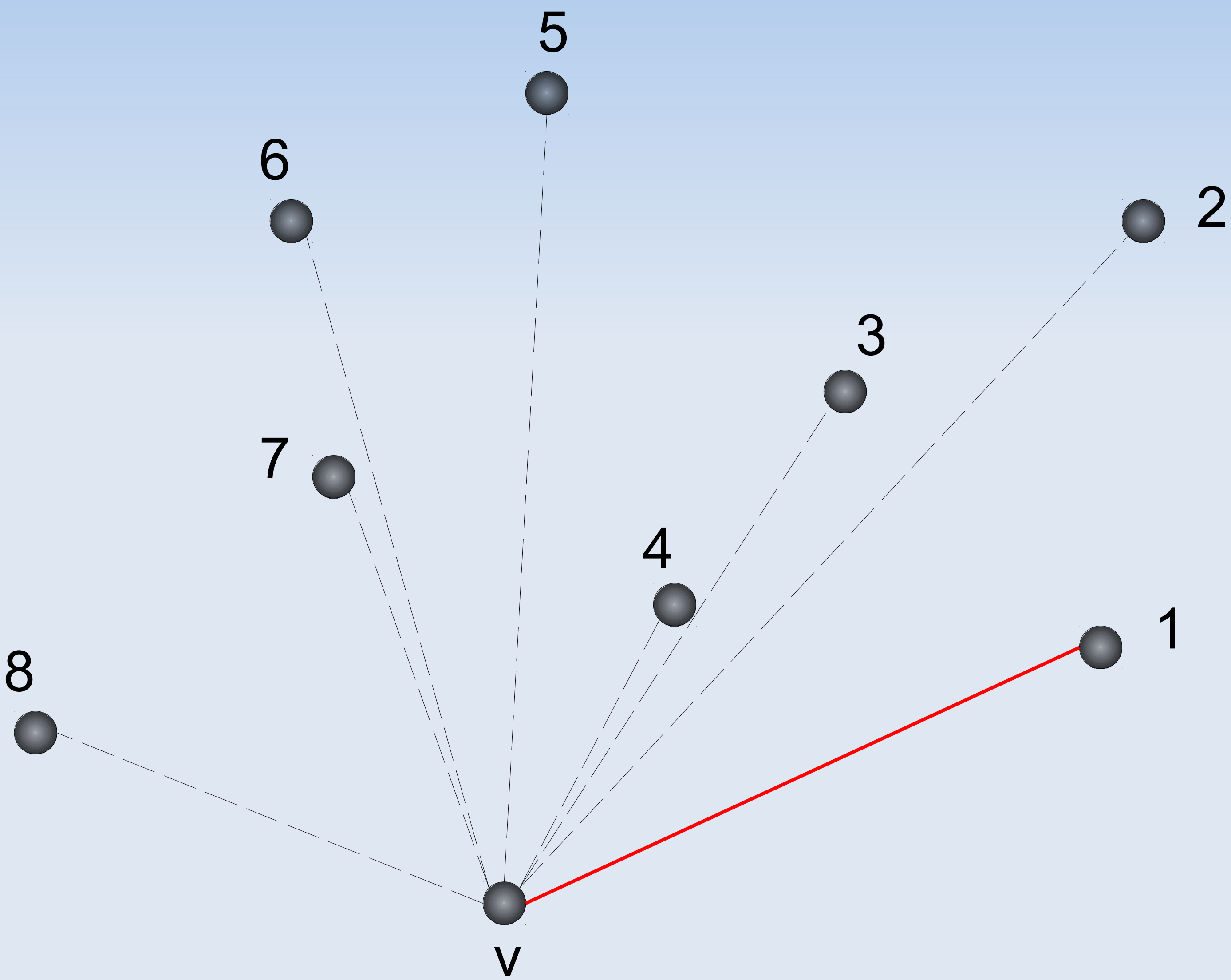
Some Problems



Solutions

- Origin of sorting: The rightmost vertex with smallest y-coordinate
  - Can be found in O(n) time
  - Always a hull vertex
  - After the sort, the first vertex is also a hull vertex

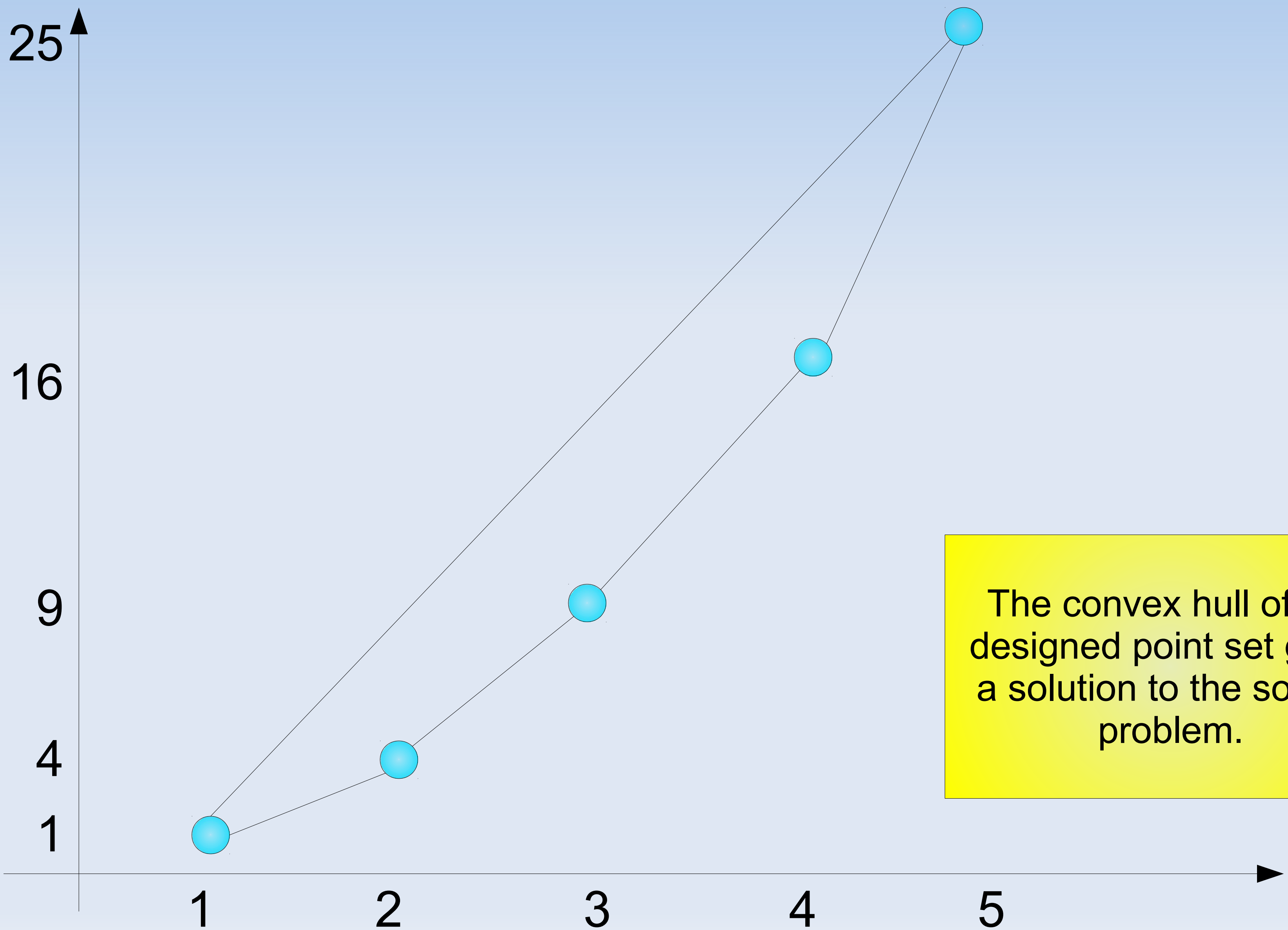
Solutions



Solutions

- Hull collinearities
  - require a strict left turn
- Sorting collinearities
  - Arbitrary solutions cause hull overlaps
    - or bugs in the implementation
  - Delete the vertices closer to the origin

Lower Bound



The convex hull of the designed point set gives a solution to the sorting problem.

QUIZ TIME

Prepare for the quiz!

QUIZ 4

Consider the following pseudocode on a set S of n points:

$i \leftarrow 1$
$S_1 \leftarrow S$
$n_1 \leftarrow n$
while ($n_i > 3$)
    compute the convex hull $H(S_i)$ of S_i
    $S_{i+1} \leftarrow S_i - H(S_i)$
    $n_{i+1} \leftarrow n_i - |H(S_i)|$
    $i \leftarrow i + 1$

where $|H(S_i)|$ is the number of points on $H(S_i)$

What is the maximum value i can obtain at the end?
Justify your answer.