CPSC 490 – Problem Solving in Computer Science

Lecture 16: Convex Hull

Jason Chiu and Raunak Kumar
Based on slides by Nasa Rouf (2014)

2017/03/01

University of British Columbia

Convex Set

A set S is convex $\Leftrightarrow \forall x, y \in S, 0 \leq \lambda \leq 1,\ \lambda x + (1-\lambda)y \in S$

$\Leftrightarrow$ the line segment $x \rightarrow y$ is inside the set

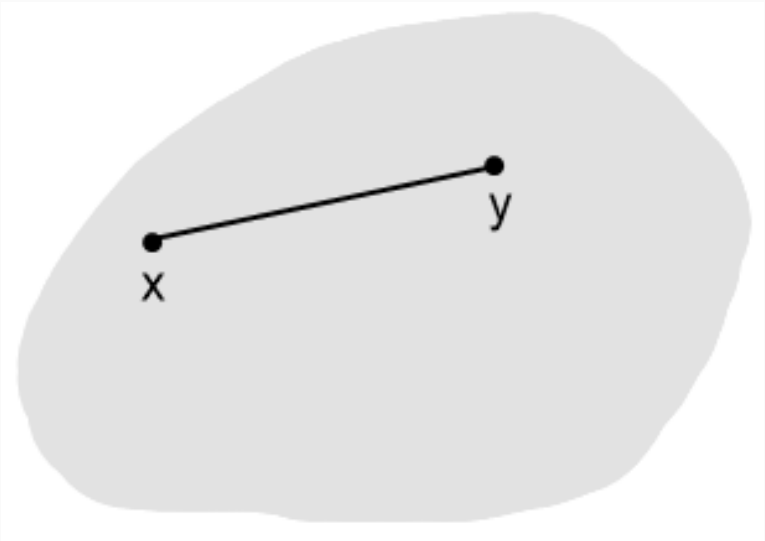


Figure 1: In a convex set, the line segment between any 2 points lies in a set

Convex Hull

A convex hull of a set is the smallest convex set containing the set.

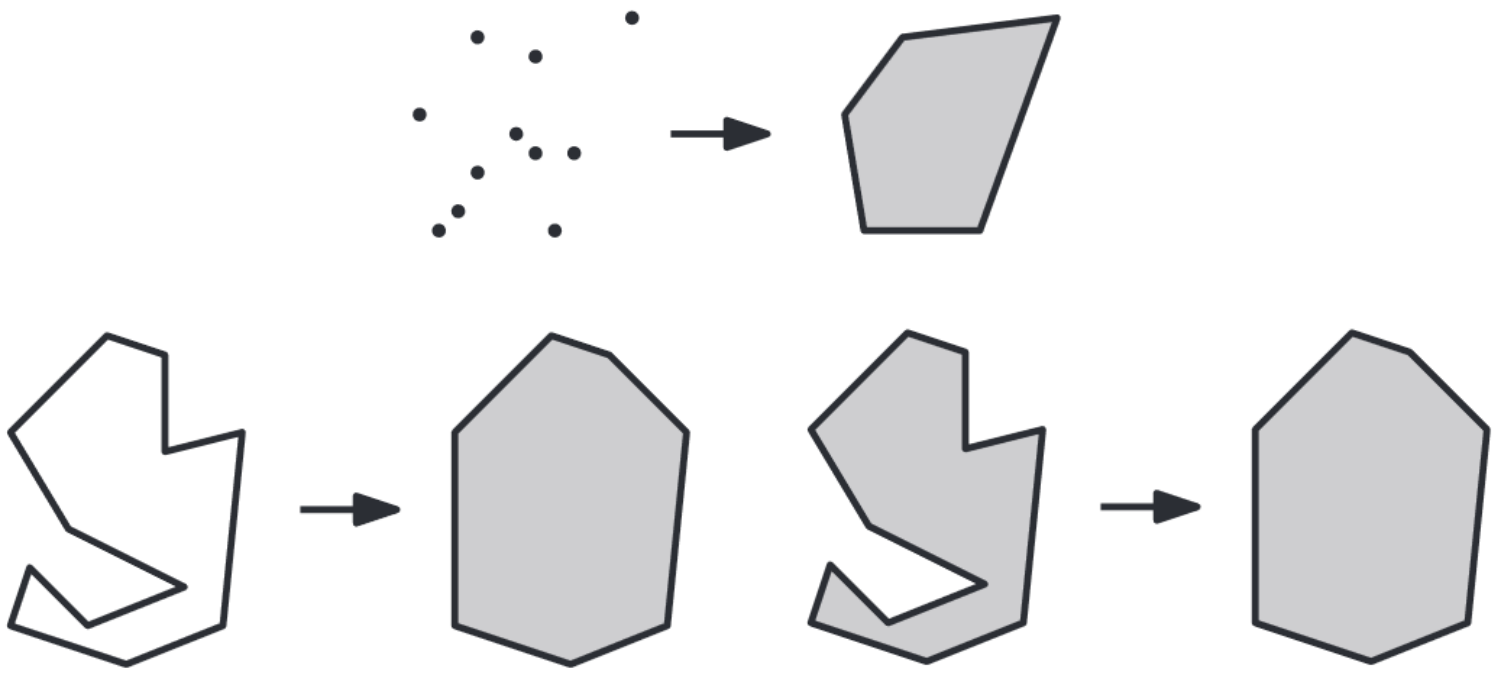


Figure 2: Some sets and their convex hulls

Monotone Chain Algorithm

Step 1: scan from left to right, “wrap” around the top
Step 2: scan from right to left, “wrap” around the bottom.

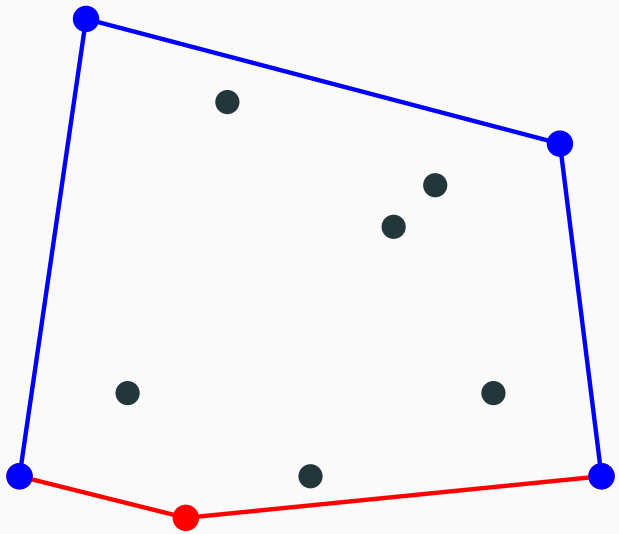
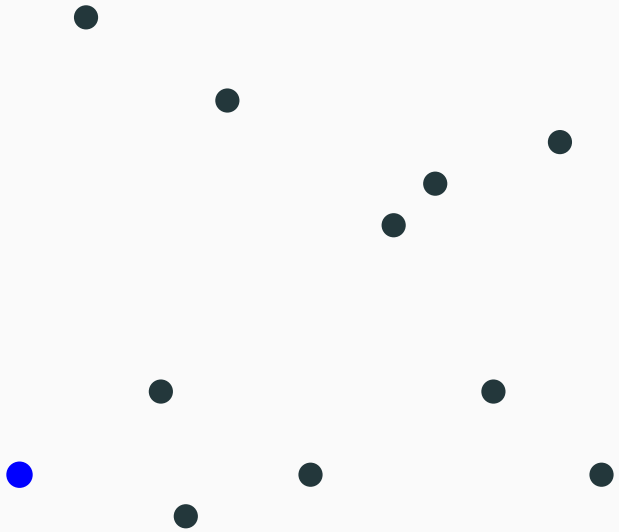
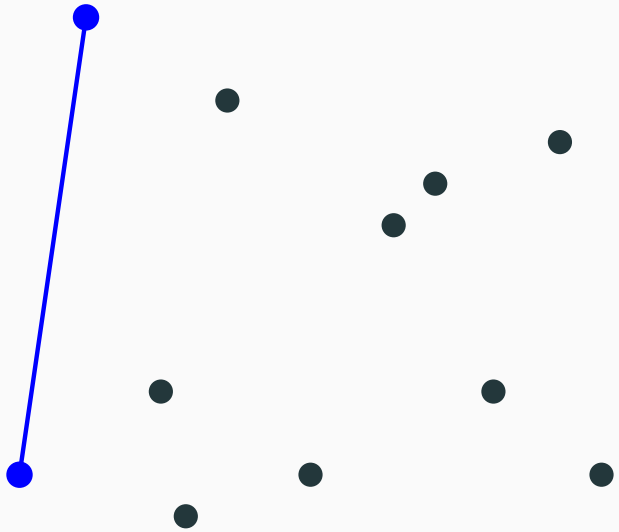


Figure 3: Upper and lower hulls built by monotone chain algorithm

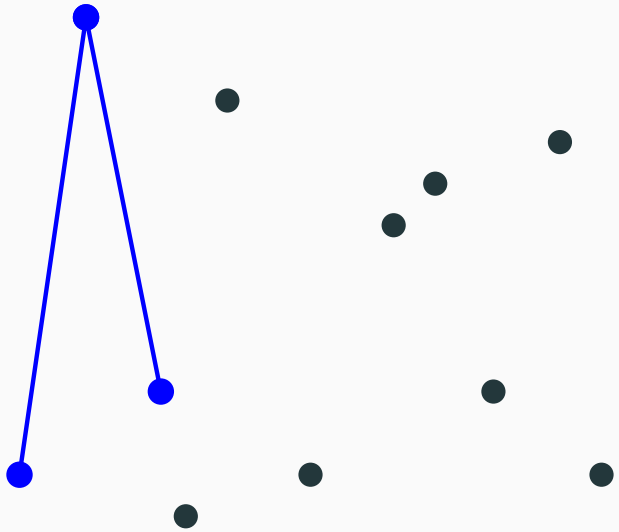
Monotone Chain Algorithm



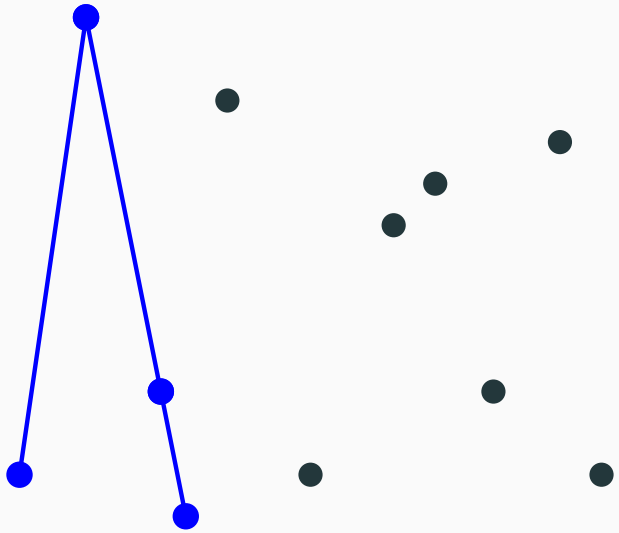
Monotone Chain Algorithm



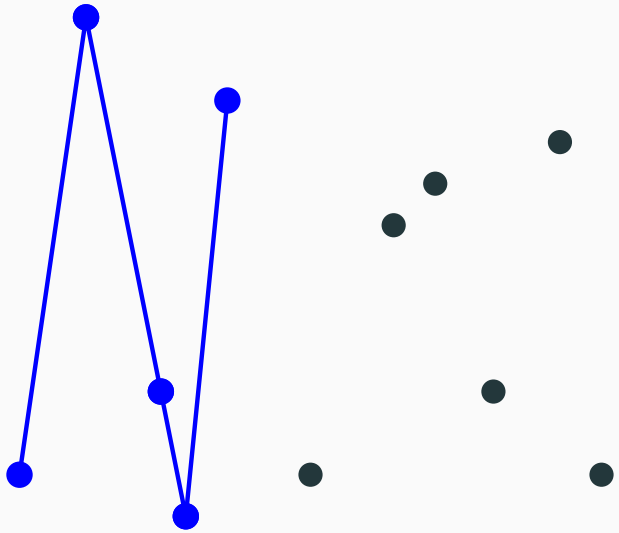
Monotone Chain Algorithm



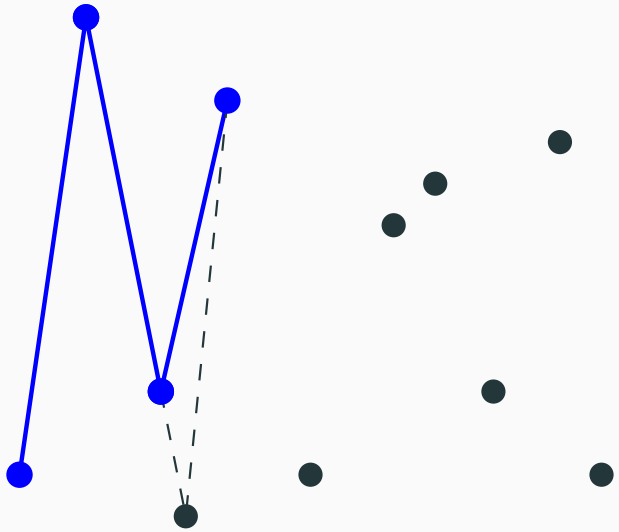
Monotone Chain Algorithm



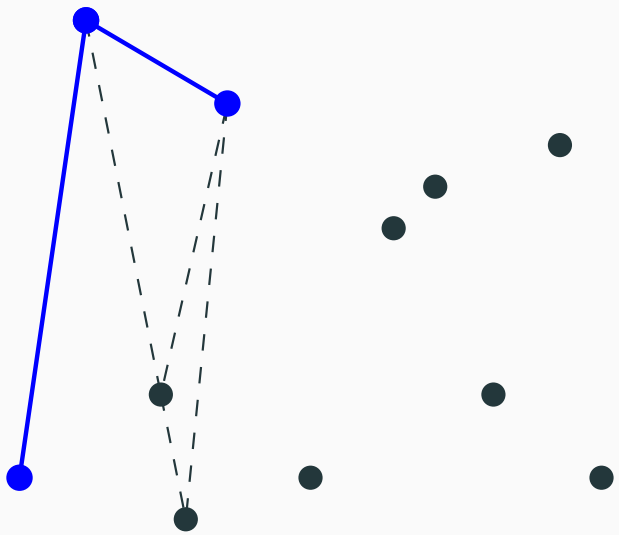
Monotone Chain Algorithm



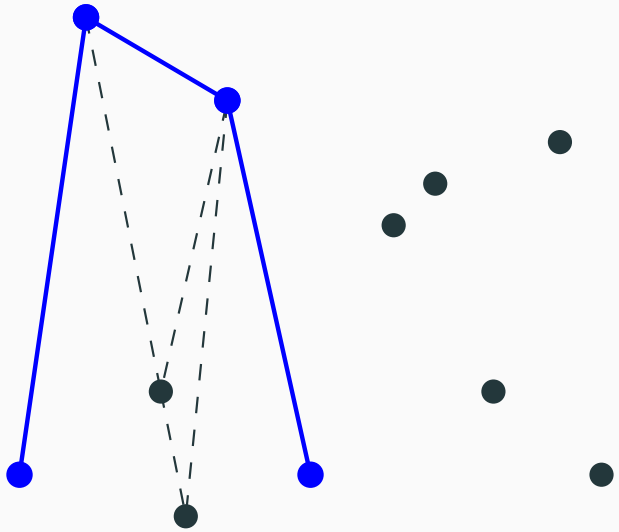
Monotone Chain Algorithm



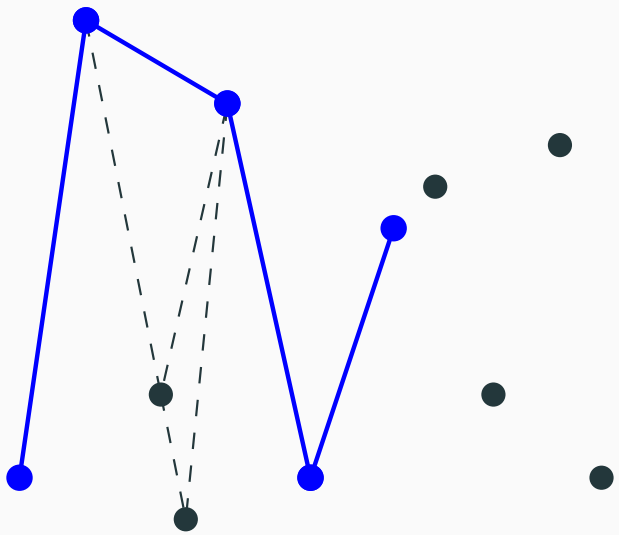
Monotone Chain Algorithm



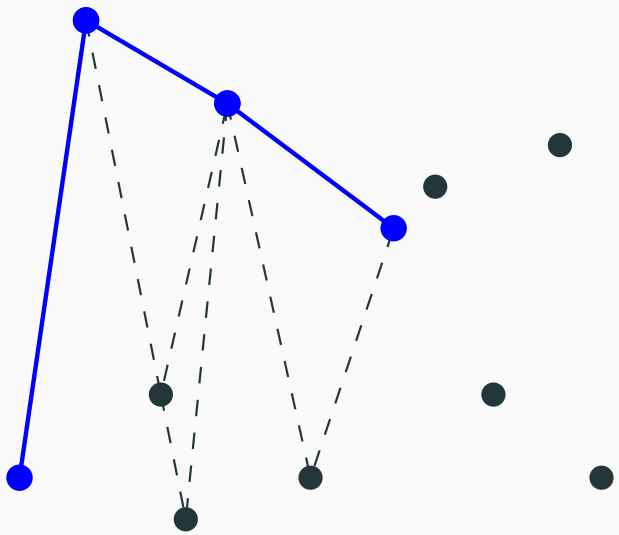
Monotone Chain Algorithm



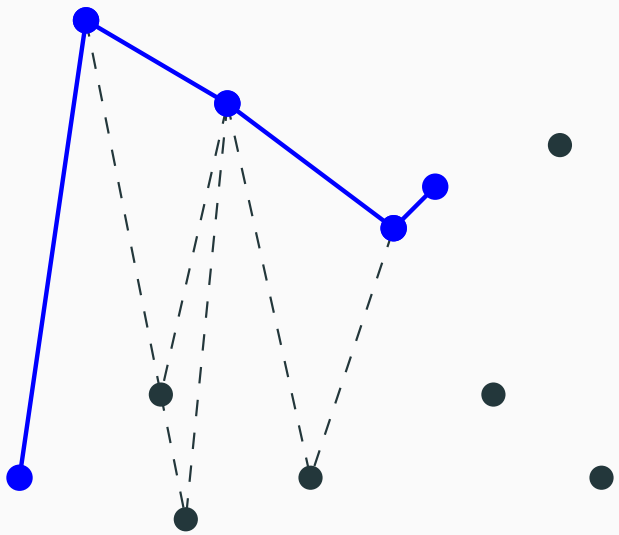
Monotone Chain Algorithm



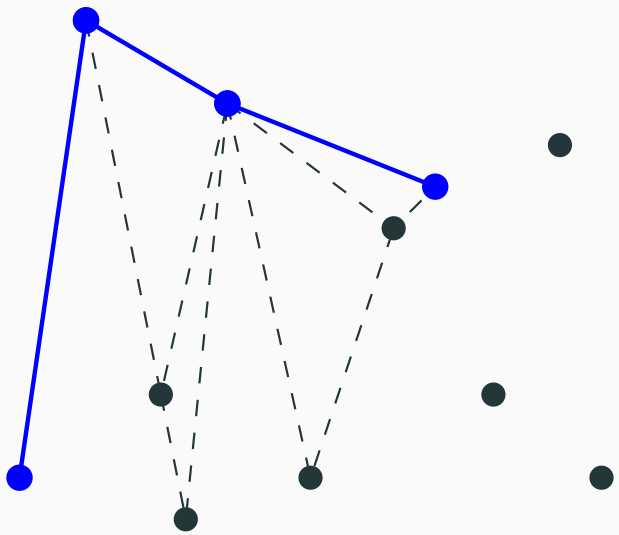
Monotone Chain Algorithm



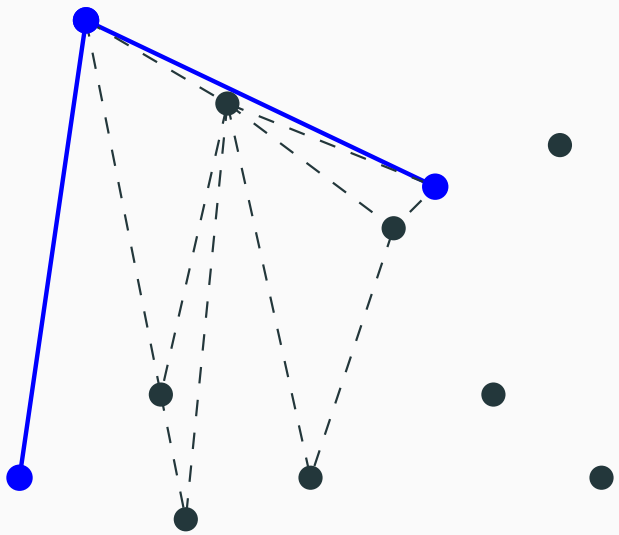
Monotone Chain Algorithm



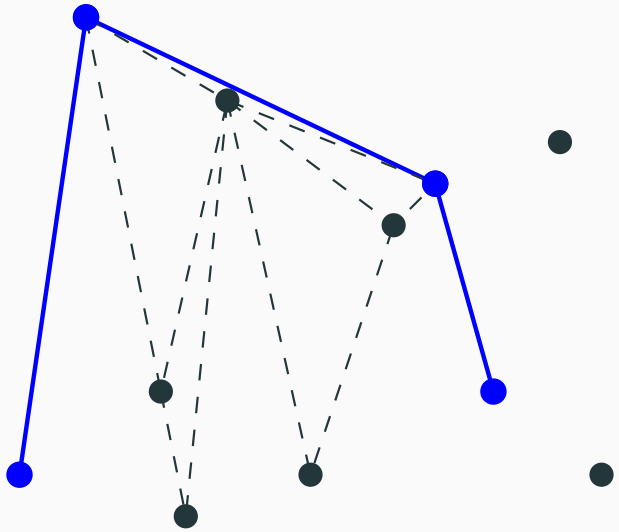
Monotone Chain Algorithm



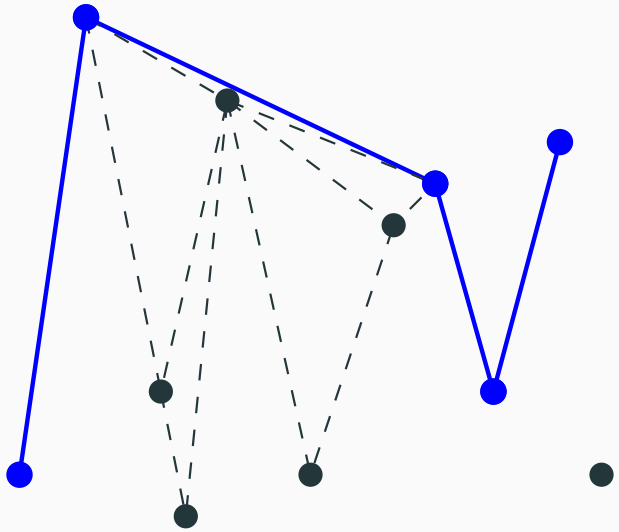
Monotone Chain Algorithm



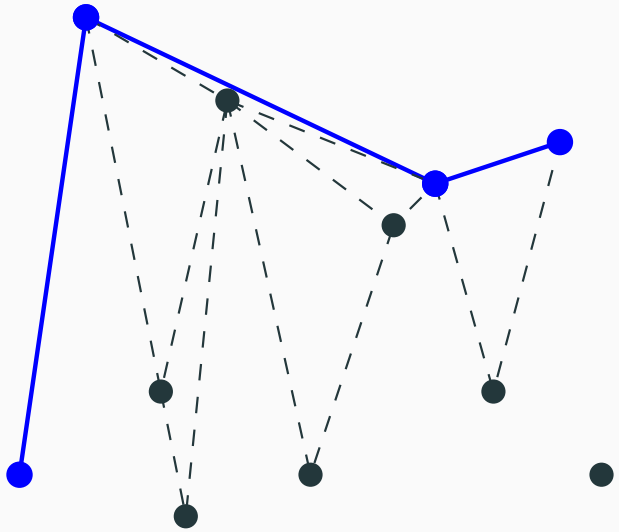
Monotone Chain Algorithm



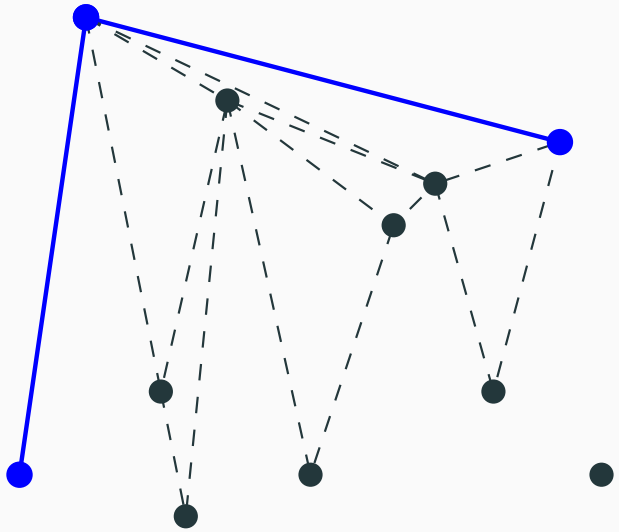
Monotone Chain Algorithm



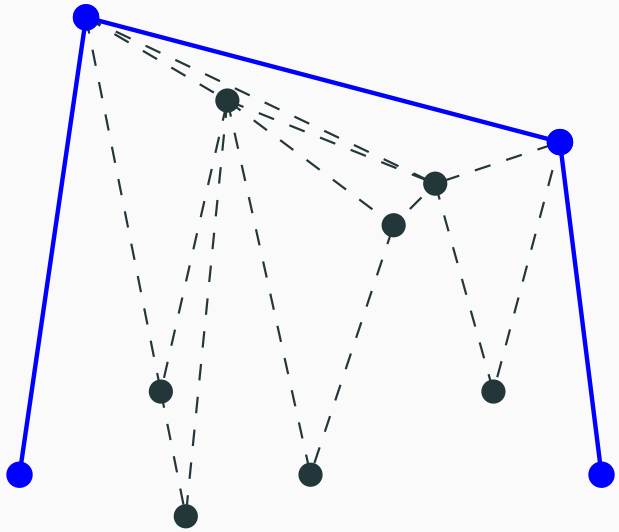
Monotone Chain Algorithm



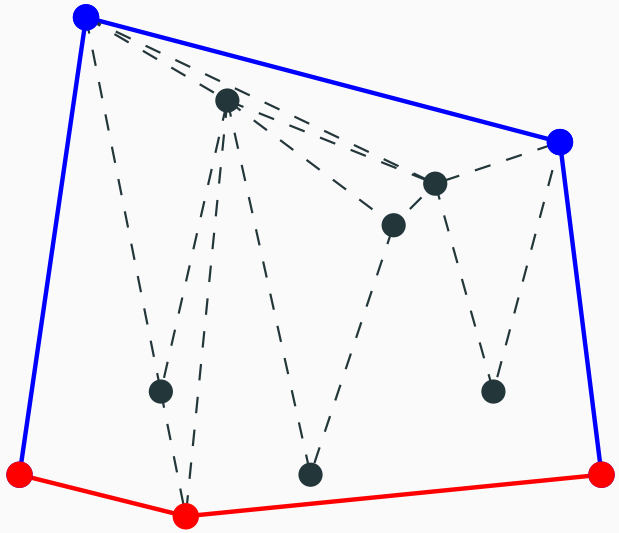
Monotone Chain Algorithm



Monotone Chain Algorithm



Monotone Chain Algorithm



Monotone Chain Algorithm

Algorithm summary

- Sort all points by x coordinates
- Start from left most point
- Iteratively add points into the hull
  - If adding a point causes CCW turn, pop points from hull until this is no longer the case before adding!
- Repeat the above steps from right to left.

Monotone Chain Algorithm

Edge cases

- Convex hull of 1, 2, 3 points
- Colinear points

Monotone Chain Algorithm

Time compexity analysis

- Sort all points by x coordinates: $O(n \log n)$
- Each point is added and removed at most once: $O(n)$

$\Rightarrow$ Time complexity: $O(n \log n)$

Monotone Chain Algorithm

Time compexity analysis

- Sort all points by x coordinates: $O(n \log n)$
- Each point is added and removed at most once: $O(n)$

$\Rightarrow$ Time complexity: $O(n \log n)$

Can we do better?

Monotone Chain Algorithm

Time compexity analysis

- Sort all points by x coordinates: $O(n \log n)$
- Each point is added and removed at most once: $O(n)$

$\Rightarrow$ Time complexity: $O(n \log n)$

Can we do better?

Arrange n points on a circle $\Rightarrow$ equivalent to to sorting, $\Omega(n \log n)$.

However, not all points on the hull, so actually $O(n \log h)$ possible.

Merging Convex Hulls

Consider the following alternate divide & conquer approach:

1. Split points into two halves
2. Compute convex hull of each half
3. Merge two convex hulls into one

How do we merge?

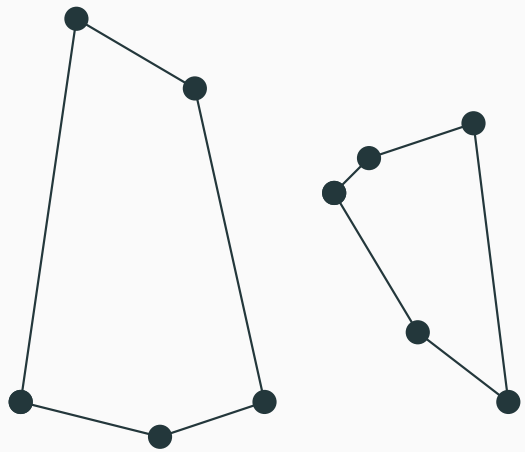
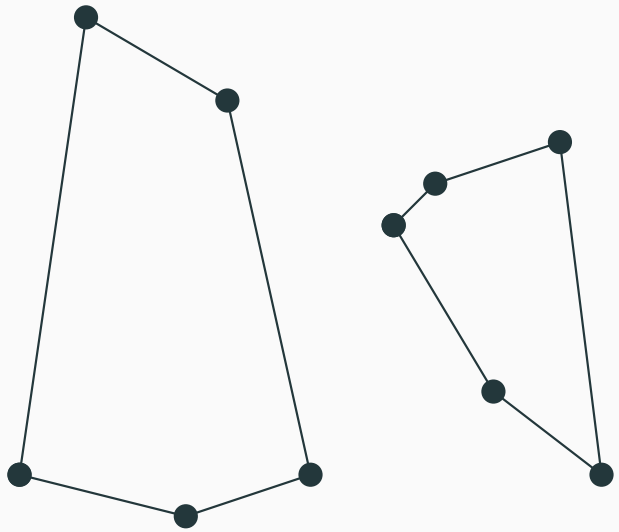
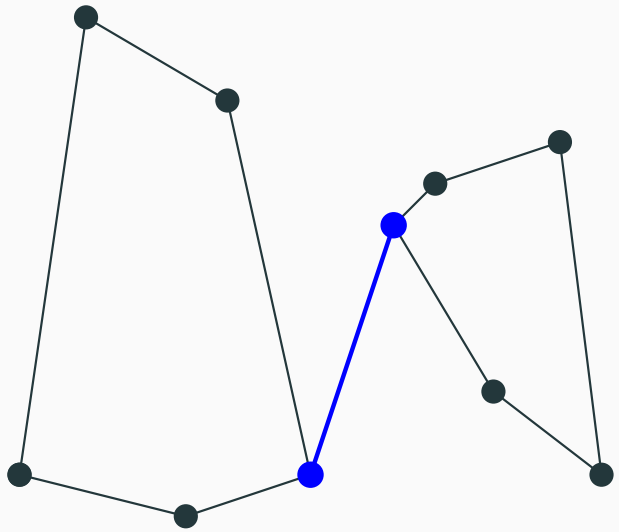


Figure 4: Two convex hulls

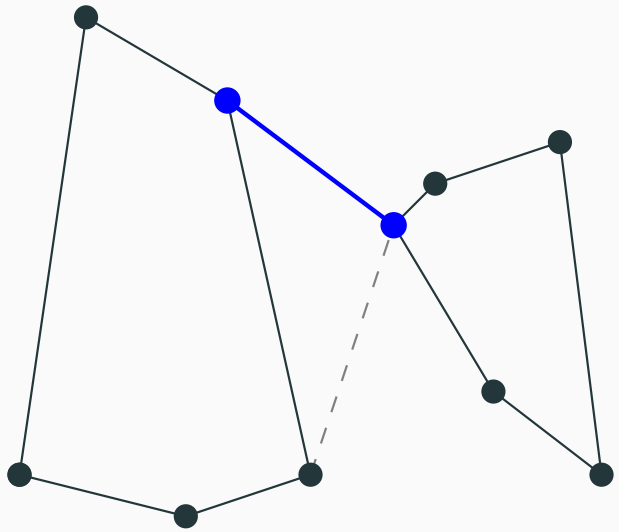
Merging Convex Hulls



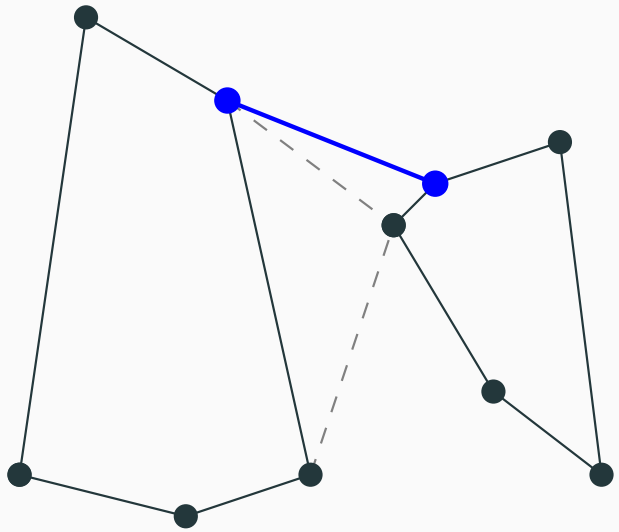
Merging Convex Hulls



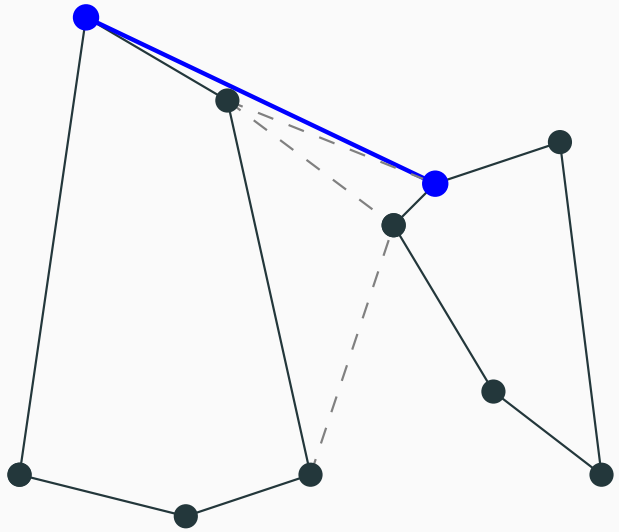
Merging Convex Hulls



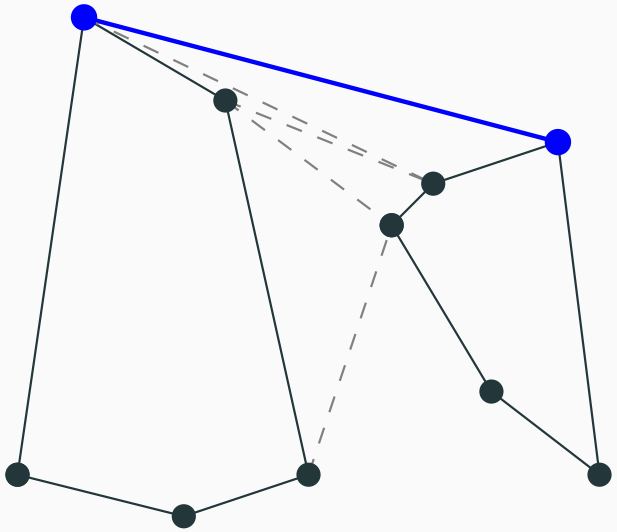
Merging Convex Hulls



Merging Convex Hulls



Merging Convex Hulls



Merging Convex Hulls

Basic idea

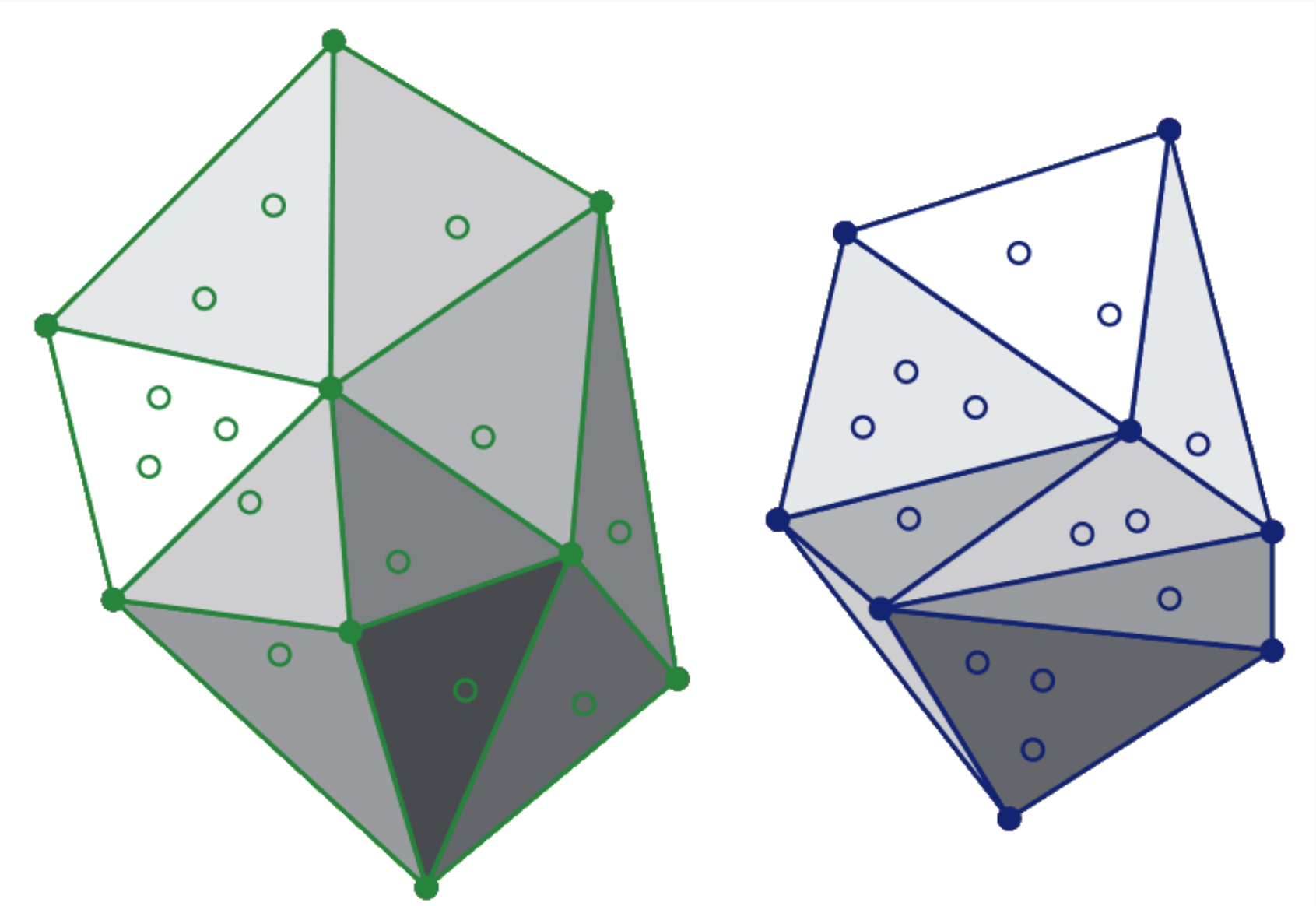
- Start with the nearest two points
- Alternate moving left side CW and right side CCW along hull
- Stop when cannot move anymore

$\Rightarrow$ Time complexity: $O(n)$

$\Rightarrow$ Divide and conquer also solves convex hull in $O(n \log n)$.

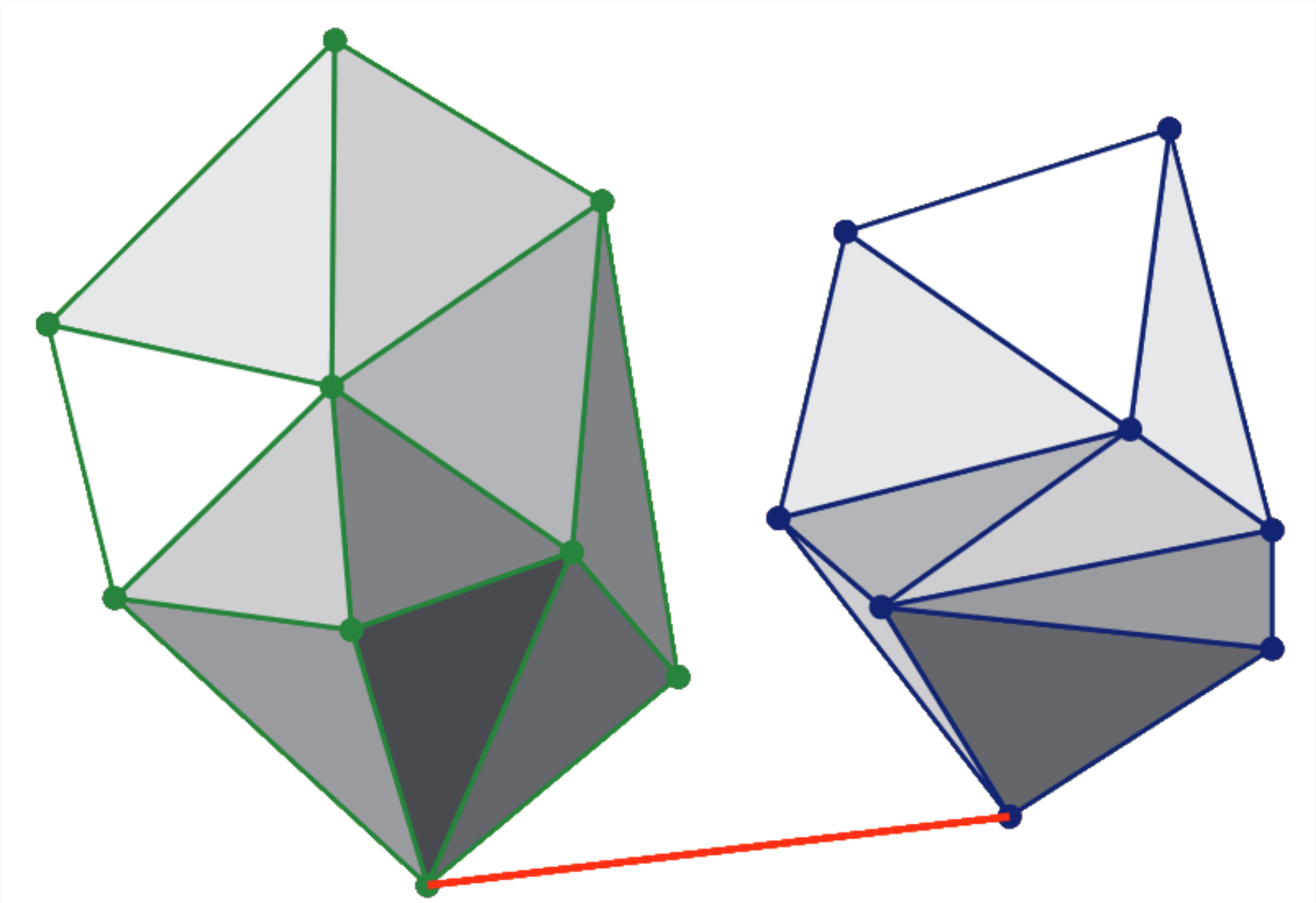
Merging Convex Hulls in 3D

Similar idea works in 3D!



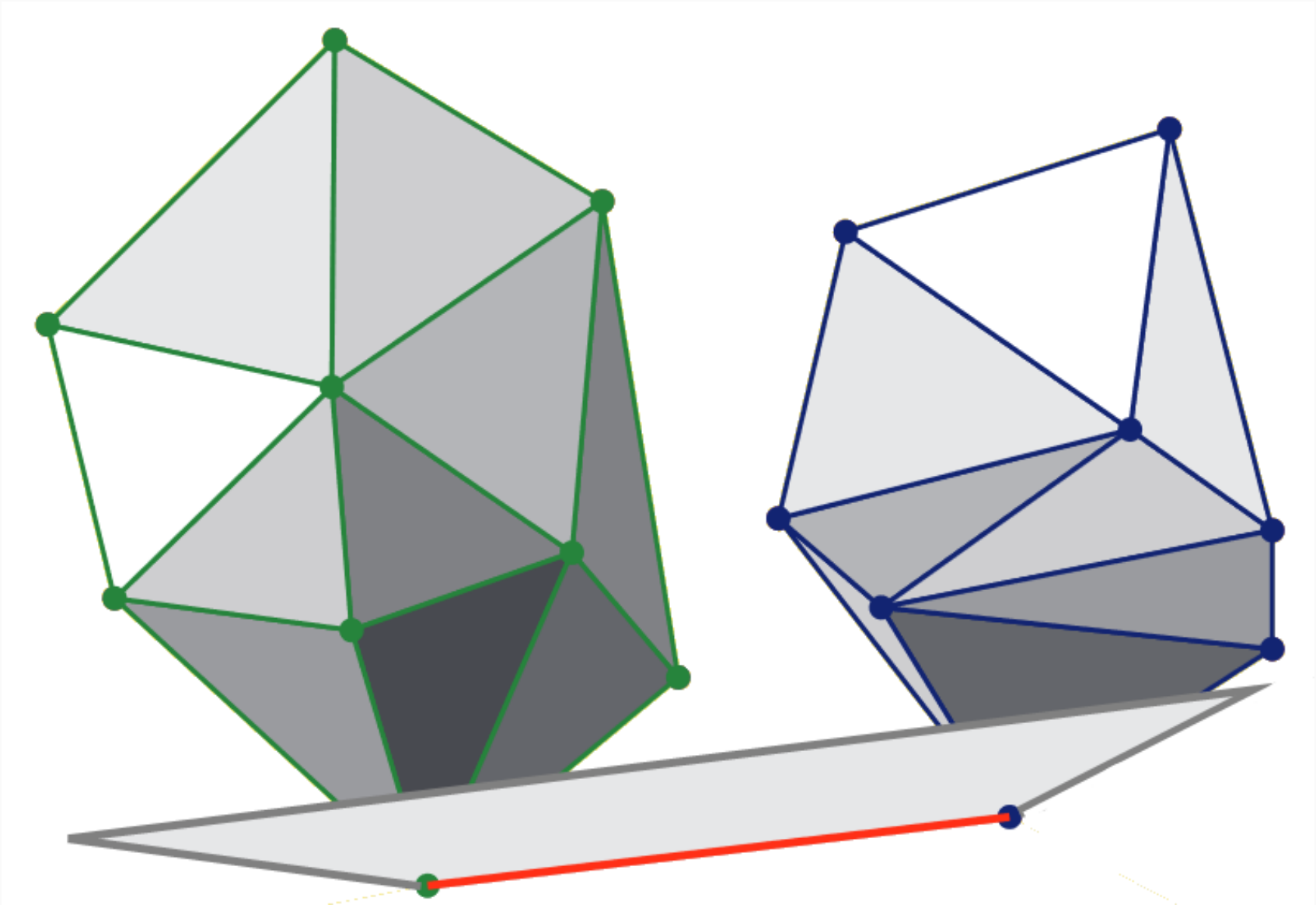
Merging Convex Hulls in 3D

Project to 2D, use 2D convex hulls to find “bottom edge”



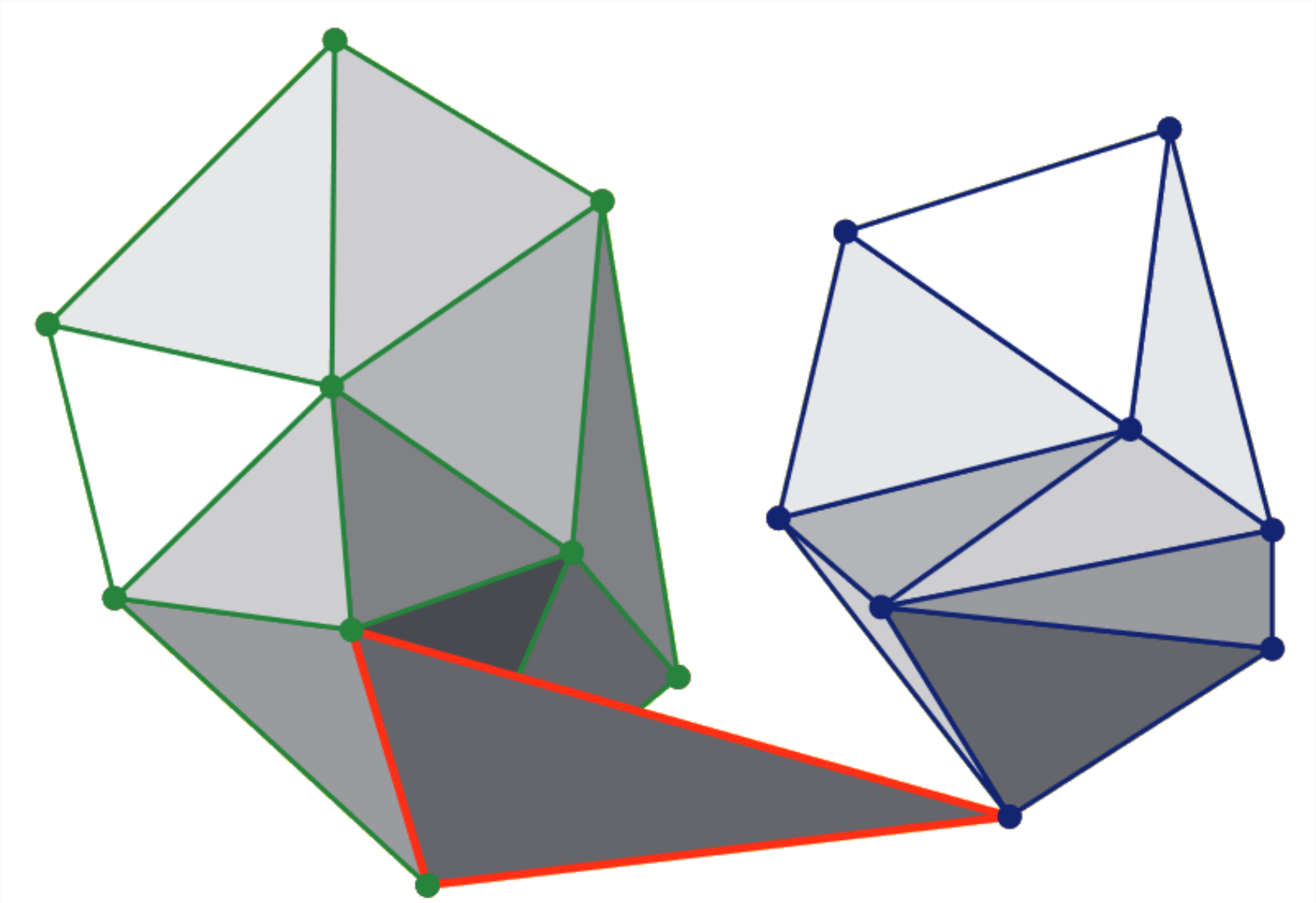
Merging Convex Hulls in 3D

"Gift-wrap" to find faces, moving to neighbor vertex of current edge



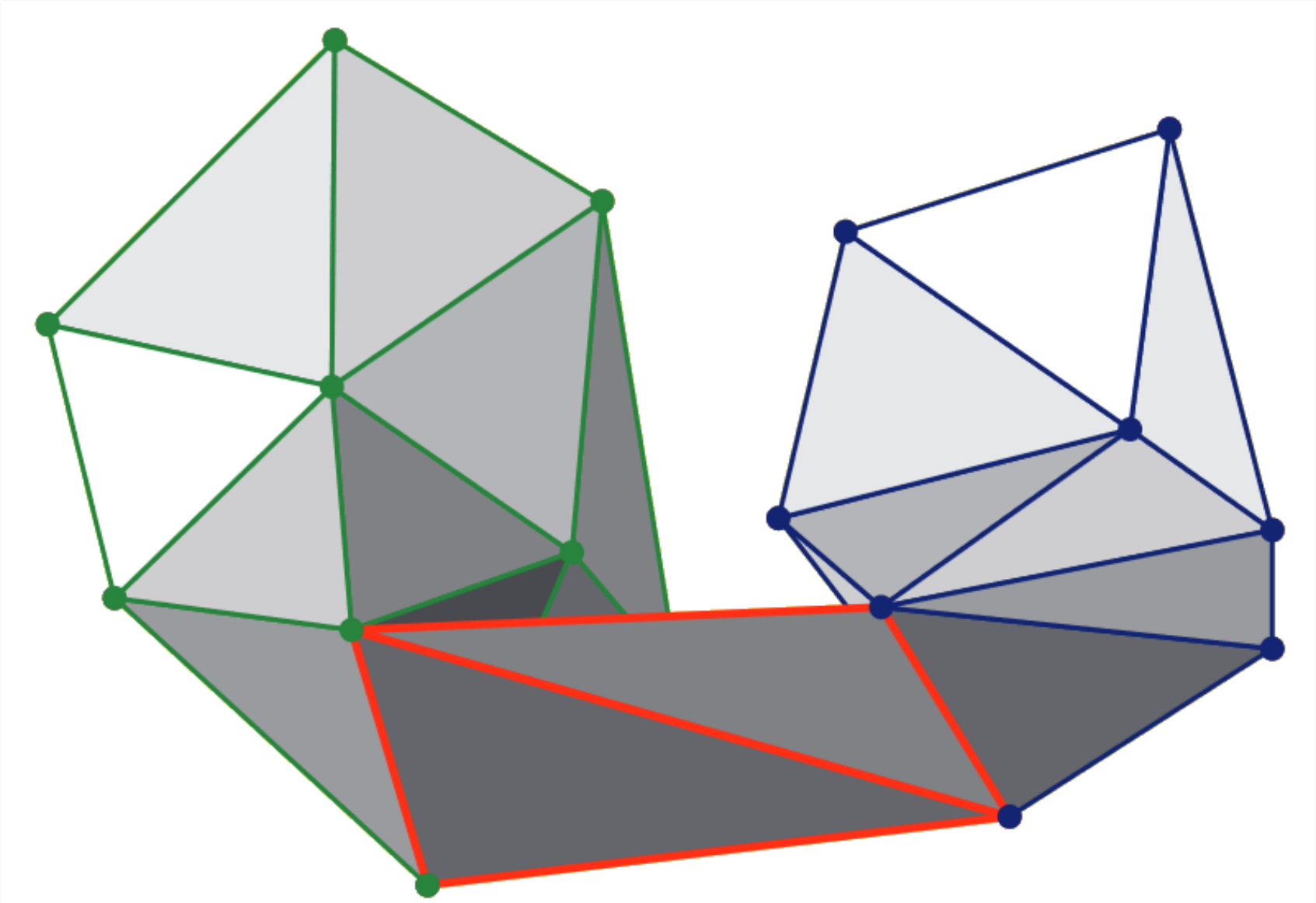
Merging Convex Hulls in 3D

"Gift-wrap" to find faces, moving to neighbor vertex of current edge



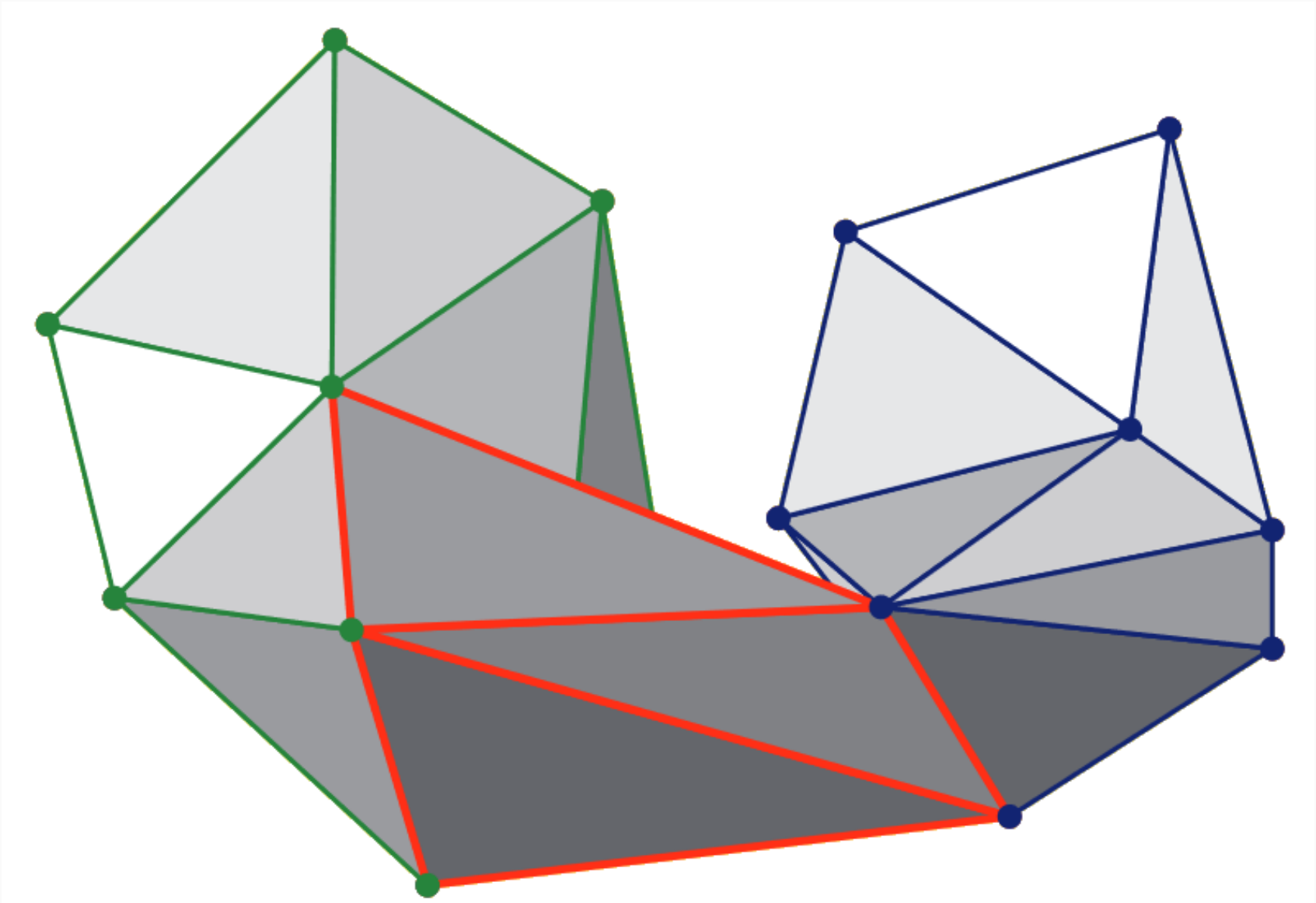
Merging Convex Hulls in 3D

"Gift-wrap" to find faces, moving to neighbor vertex of current edge



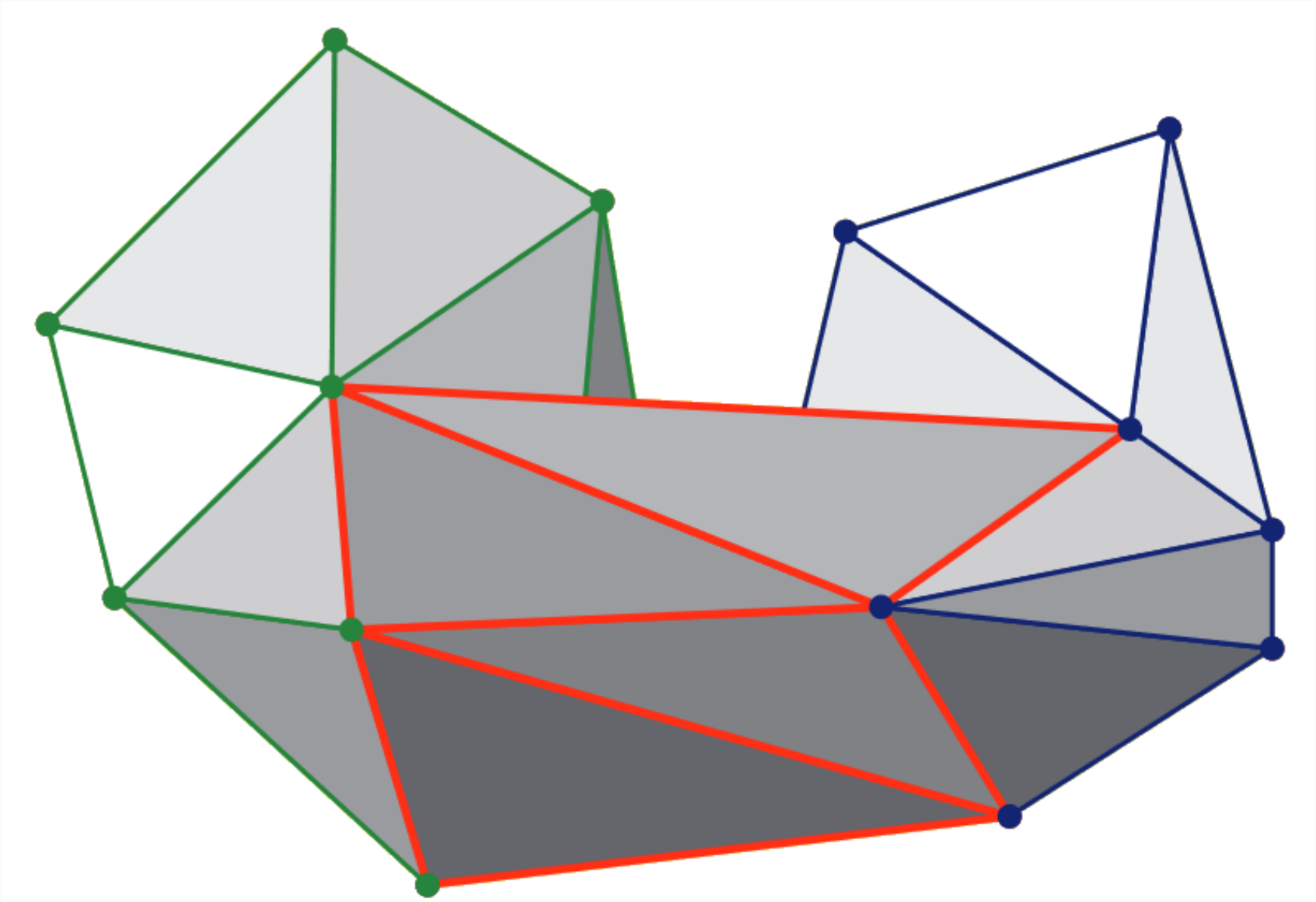
Merging Convex Hulls in 3D

"Gift-wrap" to find faces, moving to neighbor vertex of current edge



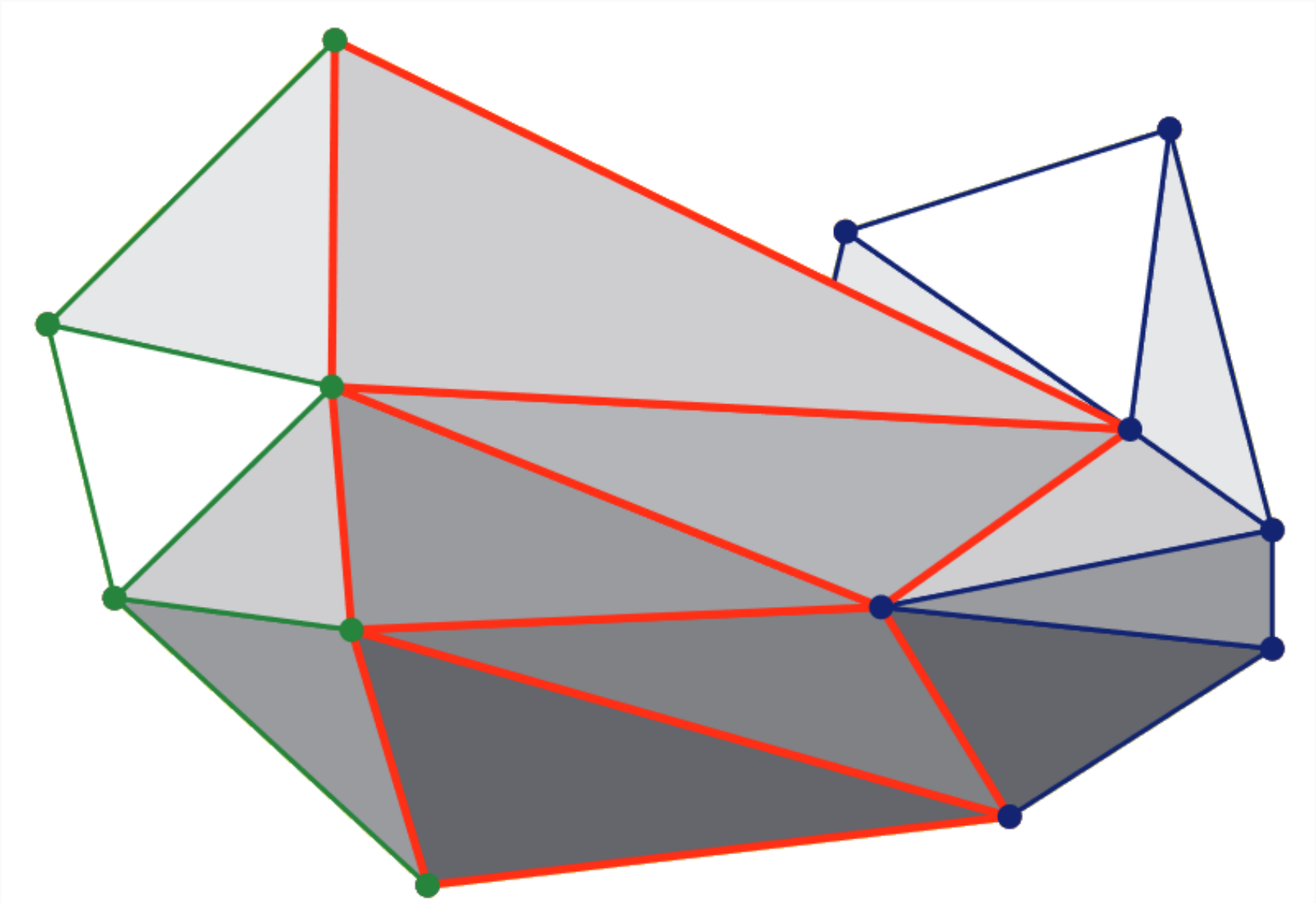
Merging Convex Hulls in 3D

"Gift-wrap" to find faces, moving to neighbor vertex of current edge



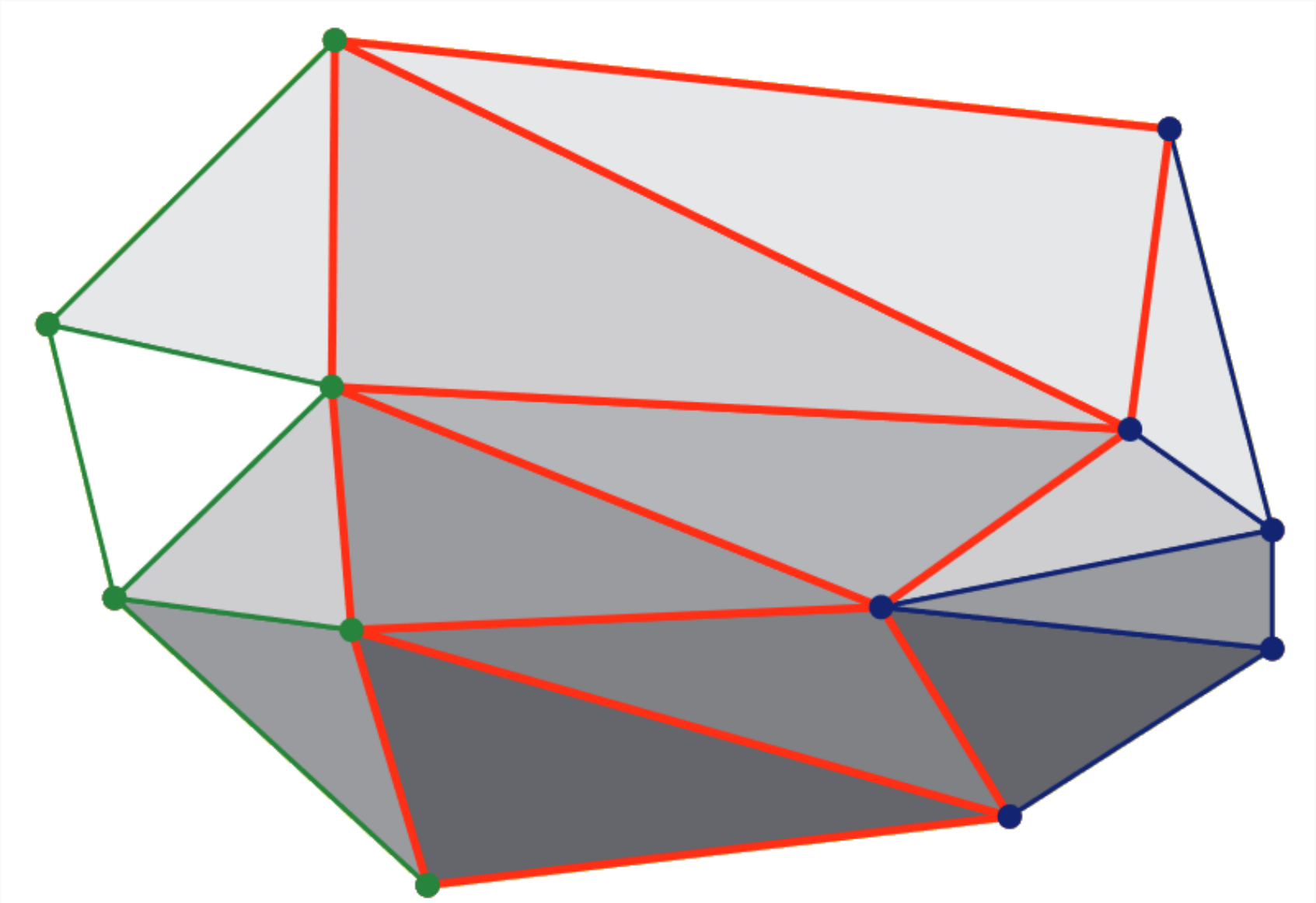
Merging Convex Hulls in 3D

"Gift-wrap" to find faces, moving to neighbor vertex of current edge



Merging Convex Hulls in 3D

"Gift-wrap" to find faces, moving to neighbor vertex of current edge



Merging Convex Hulls in 3D

Summary

- Compute hull of 2D projection and find “bottom edge”
- “Gift-wrap” smartly to find each face around in a circle
  - Sort neighbors of left vertex CCW, neighbors of right vertex CW
  - For left & right vertices, find their “candidate neighbor”
    - Candidate neighbor of left vertex = neighbor of left vertex such that left convex hull is entirely on one side of plane of new triangle
    - There is only one possible candidate
  - Gift wrap: find which candidate neighbor (left or right) causes “least rotation” of the plane and add that one
  - Update new: for the new vertex sort neighbors and find candidate
  - Update old: for vertex that did not move, new candidate neighbor must be further along in the sorted list! $\Rightarrow$ amortized $O(\text{degree})$

$\Rightarrow$ Time complexity: $O(n)$ merge

$\Rightarrow$ $O(n \log n)$ divide and conquer for 3D convex hull!

Problem 1: Binary Search on a Convex Hull

Given $N \leq 100,000$ points on the plane, answer $Q \leq 100,000$ queries of the following type:

> *If you "look" from point P, what's the left and right most points you can see?*

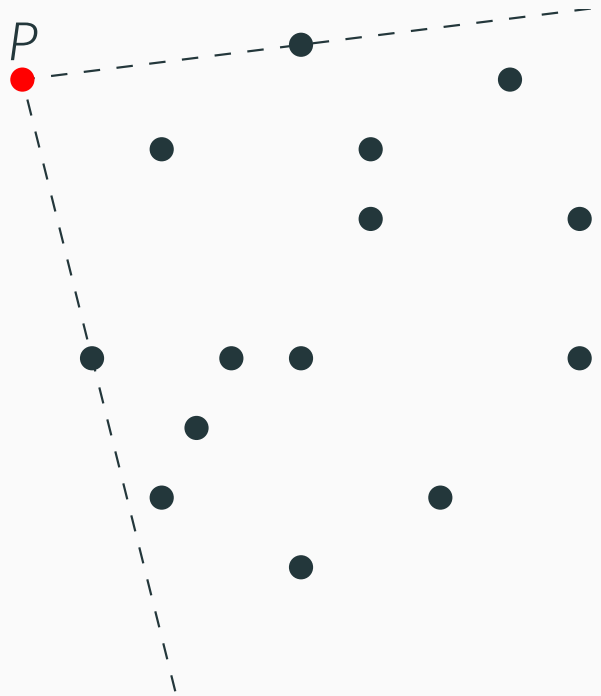
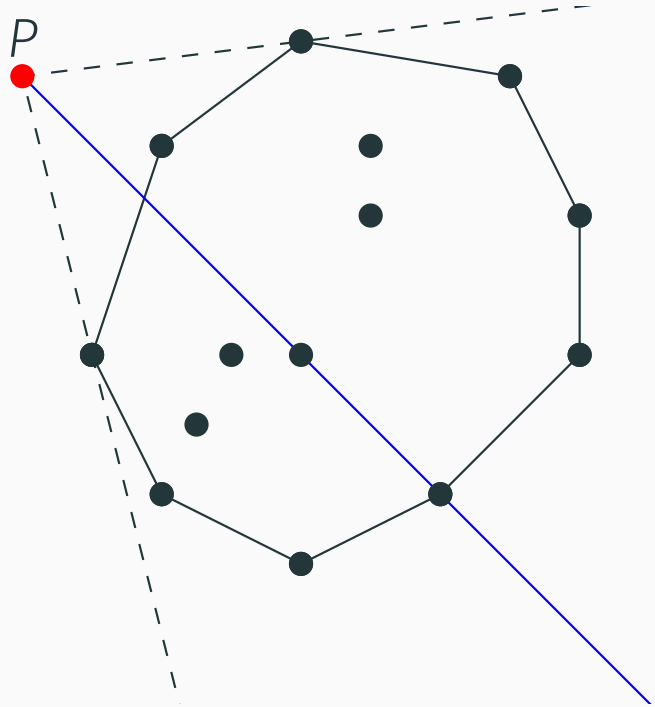


Figure 5: Finding the left most and right most point when viewed from P

Problem 1: Solution

Observation 1: solution lies on convex hull

Observation 2: pick an arbitrary point on hull, draw a line through, then one extremum on each side of the line.



Problem 1: Solution

Observation 1: solution lies on convex hull

Observation 2: pick an arbitrary point on hull, draw a line through, then one extremum on each side of the line.

$\Rightarrow$ binary search to find start/end vertices on hull for each half, then binary search on each half of the hull for when $P \to p_i \to p_{i+1}$ changes from CCW to CW (or vice versa).

Time complexity: $O(N \log N + Q \log N)$

Problem 2: Dynamic Programming on a Convex Hull

Given $N \leq 100$ points on the plane, and $K \leq N$, what's the maximum area you can enclose by picking up to K out of N points to form a polygon?

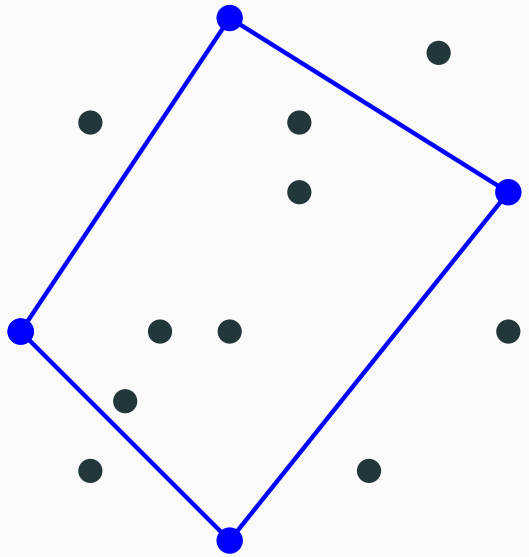


Figure 6: Enclosing an area by picking a subset of points to form polygon.

Problem 2: Solution

Observation 1: must pick points on the convex hull

Observation 2: convex hull is convex polygon
$\Rightarrow$ do interval dp on the convex hull, just like in Lecture 06!

Main idea: build the polygon by adding triangles in CCW order

$$dp(i, j, k) = \text{maximum enclosed area using } k \text{ vertices out of } i, i+1, \ldots, j$$
$$\text{given that } i \to j \text{ is an edge on the boundary}$$
$$= \max_{i<x<j} \text{Area}(i, x, j) + dp(x, j, k-1)$$

Time complexity: $O(N^3 K)$

Seems pretty slow! Next class we will learn how to do it in $O(NK)$!

References

- http://www.cs.uu.nl/docs/vakken/ga/slides1.pdf
- http://www.loria.fr/~pougetma/enseignement/webimpa/2-3D-enveloppe-convexe-od.pdf
- http://www3.jouy.inra.fr/miaj/public/vigneron/talks/diam.pdf
- http://www.cs.umd.edu/~mount/754/Lects/754lects.pdf

Next Class

Rotating Calipers