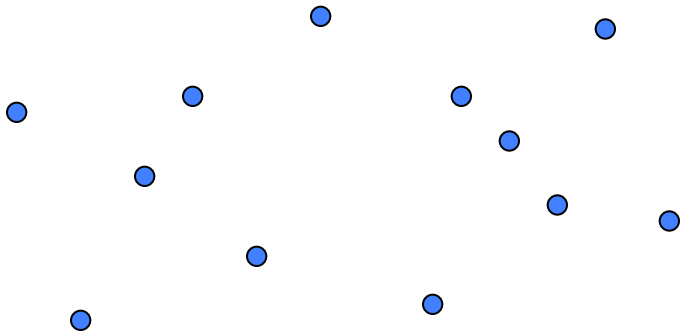
Convex Hulls

Convex Hull

Def.

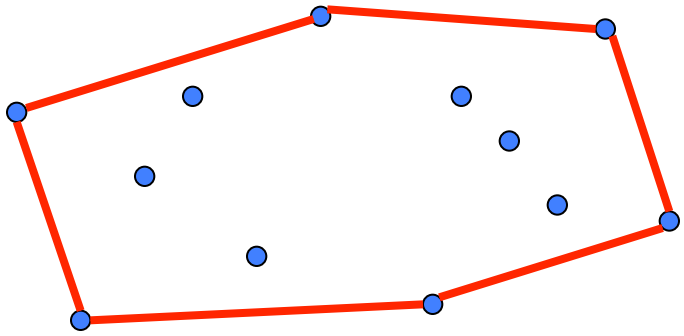
- Given a set of points (x_1, y_1), (x_2, y_2), (x_n, y_n) the convex hull is the smallest convex polygon containing all of the points
  - A polygon is convex if given any two points of the polygon, all of the points on the line segment connecting them are also part of the polygon
  - A vertex of a convex polygon can not be expressed as a convex combination of two points of the polygon



Convex Hull

Def.

- Given a set of points (x_1, y_1), (x_2, y_2), (x_n, y_n) the convex hull is the smallest convex polygon containing all of the points
  - A polygon is convex if given any two points of the polygon, all of the points on the line segment connecting them are also part of the polygon
  - A vertex of a convex polygon can not be expressed as a convex combination of two points of the polygon



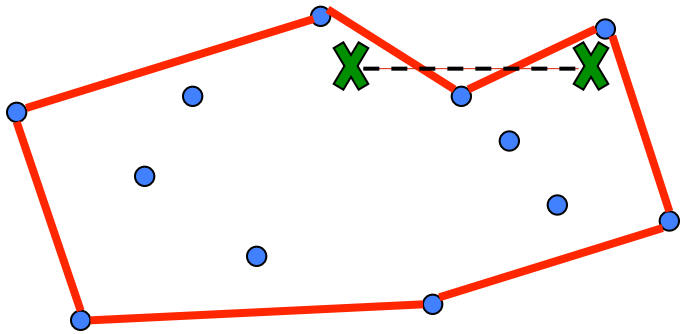
Yes

Think of stretching out a rubber band and placing it around the points

Convex Hull

Def.

- Given a set of points (x_1, y_1), (x_2, y_2), (x_n, y_n) the convex hull is the smallest convex polygon containing all of the points
  - A polygon is convex if given any two points of the polygon, all of the points on the line segment connecting them are also part of the polygon
  - A vertex of a convex polygon can not be expressed as a convex combination of two points of the polygon

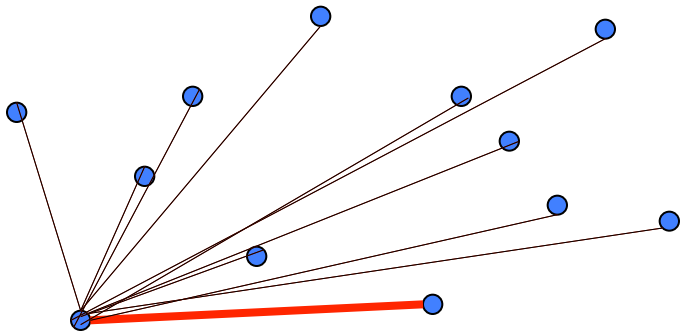


No

Convex Hull Algorithms

Gift Wrapping

- Starting at the bottom point (bottom left if there are ties), "wrap" around the convex hull by repeatedly identifying the point with smallest polar angle relative to previous segment and add to hull
  - All points identified in the process are the vertices of the convex hull



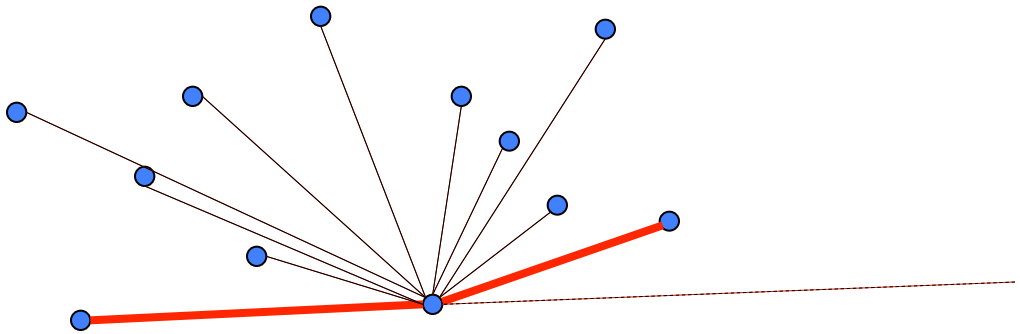
If there are multiple points at the same polar angle, choose the one farthest away

Convex Hull Algorithms

Gift Wrapping

- Starting at the bottom point (bottom left if there are ties), "wrap" around the convex hull by repeatedly identifying the point with smallest polar angle relative to previous segment and add to hull

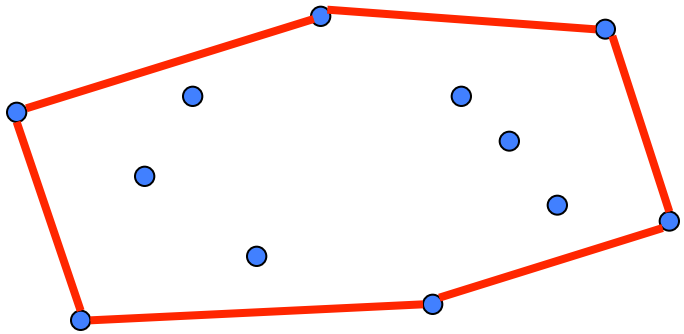
Use law of cosines or other trigonometric relationship to compute polar angle



Convex Hull Algorithms

Gift Wrapping Computational Complexity (given n points)

- Find the bottom-most vertex:
  - O(n)
- Compute the minimum polar angle with respect to other points:
  - O(1) to compute any given angle, so O(n) total
- Repeat for however many vertices the convex hull ends up with
- For a convex hull with h vertices, total complexity is:
  - O(nh)



Convex Hull Algorithms

A Divide and Conquer Method

- Initially sort the points by their x value
- Recursive Algorithm
  - If number of points is ≤ 5, solve by gift-wrapping
  - Else

    Compute the convex hull of the left $\frac{1}{2}$ of the points

    Compute the convex hull of the right $\frac{1}{2}$ of the points

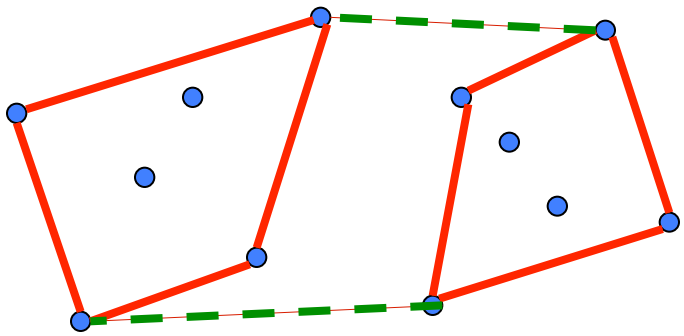
    Merge the hulls into a single convex hull

- Note the similarity to MergeSort.
- Recursive tree has depth roughly log n. If the cost at each step to merge the hulls is O(n), then the entire algorithm is O(n log n)

Convex Hull Algorithms

A Divide and Conquer Method: Merging Convex Hulls

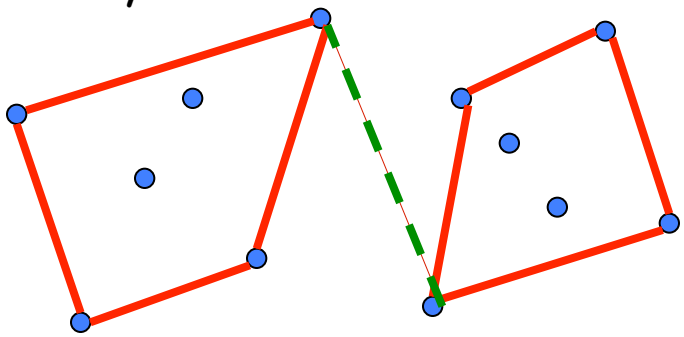
- Overall Goal: find upper tangent and lower tangent lines that connect the two convex hulls.
- Upper tangent is defined as line connecting the two convex hulls such that all other hull points lie below it.
- Lower tangent defined oppositely.



Convex Hull Algorithms

Merging Convex Hulls: Finding the Upper Tangent

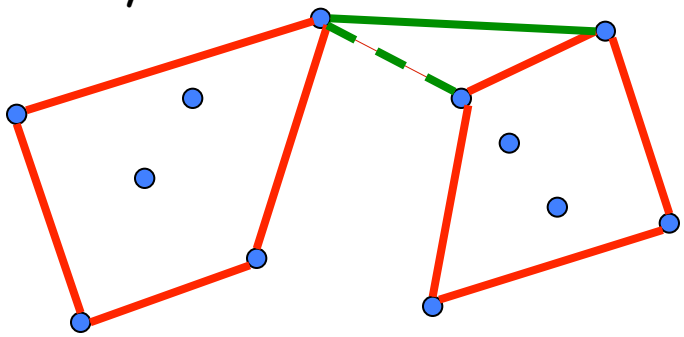
- Start at the rightmost vertex of CH_1 and the leftmost vertex of CH_2.
- Upper tangent will be a line segment such that it makes a left turn with next counter-clockwise vertex of CH_1 and makes a right turn with next clockwise vertex of CH_2.
- If the line segment connecting them is not the upper tangent
  - If the line segment does not make a left turn with next counter-clockwise vertex of CH_1, rotate to the next counter-clockwise vertex in CH_1.
  - Else if the line segment does not make a right turn with the next clockwise vertex of CH_2, rotate to the next clockwise vertex in CH_2.
  - Repeat as necessary.



Convex Hull Algorithms

Merging Convex Hulls: Finding the Upper Tangent

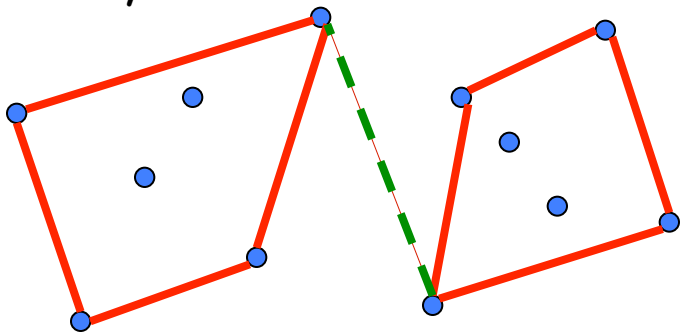
- Start at the rightmost vertex of CH_1 and the leftmost vertex of CH_2.
- Upper tangent will be a line segment such that it makes a left turn with next counter-clockwise vertex of CH_1 and makes a right turn with next clockwise vertex of CH_2.
- If the line segment connecting them is not the upper tangent
  - If the line segment does not make a left turn with next counter-clockwise vertex of CH_1, rotate to the next counter-clockwise vertex in CH_1.
  - Else if the line segment does not make a right turn with the next clockwise vertex of CH_2, rotate to the next clockwise vertex in CH_2.
  - Repeat as necessary.



Convex Hull Algorithms

Merging Convex Hulls: Finding the Lower Tangent

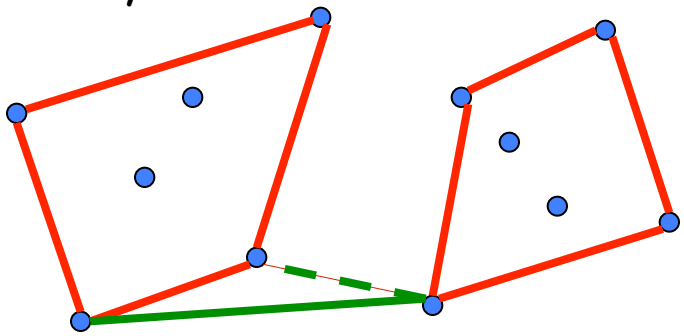
- Start at the rightmost vertex of CH_1 and the leftmost vertex of CH_2.
- Lower tangent will be a line segment such that it makes a right turn with next clockwise vertex of CH_1 and makes a left turn with next counter-clockwise vertex of CH_2.
- If the line segment connecting them is not the lower tangent
  - If the line segment does not make a right turn with next clockwise vertex of CH_1, rotate to the next clockwise vertex in CH_1.
  - Else if the line segment does not make a left turn with the next counter-clockwise vertex of CH_2, rotate to the next counter-clockwise vertex in CH_2.
  - Repeat as necessary.



Convex Hull Algorithms

Merging Convex Hulls: Finding the Lower Tangent

- Start at the rightmost vertex of CH_1 and the leftmost vertex of CH_2.
- Lower tangent will be a line segment such that it makes a right turn with next clockwise vertex of CH_1 and makes a left turn with next counter-clockwise vertex of CH_2.
- If the line segment connecting them is not the lower tangent
  - If the line segment does not make a right turn with next clockwise vertex of CH_1, rotate to the next clockwise vertex in CH_1.
  - Else if the line segment does not make a left turn with the next counter-clockwise vertex of CH_2, rotate to the next counter-clockwise vertex in CH_2.
  - Repeat as necessary.



Convex Hull Algorithms

A Divide and Conquer Method – Computational Complexity: T(n)

- Initially sort the points by their x value: $O(n \log n)$
- Recursive Algorithm: R(n)
  - If number of vertices is ≤ 5, solve by gift-wrapping: $O(1)$
  - Else
    Compute the convex hull of the left $\frac{1}{2}$ of the points: R(n/2)
    Compute the convex hull of the right $\frac{1}{2}$ of the points: R(n/2)
    Merge the hulls into a single convex hull: $O(n)$

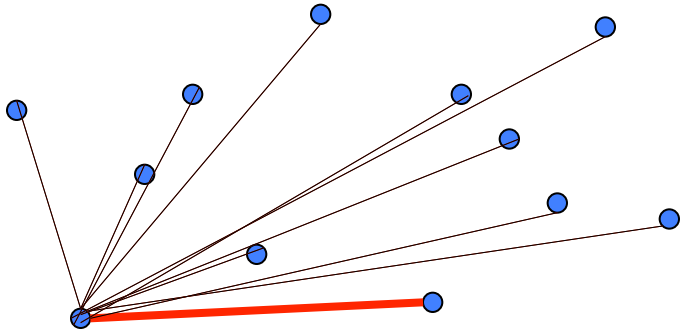
- The Merge hull operation visits each node at most once (we will not prove this): $O(n)$

- So $R(n) = 2\,R(n/2) + O(n)$
  - Just like MergeSort, this is $O(n \log n)$
- So $T(n) = O(n \log n) + O(n \log n)$
  - T(n) is $O(n \log n)$

Convex Hull Algorithms

Graham-Scan: Another O(n log n) approach

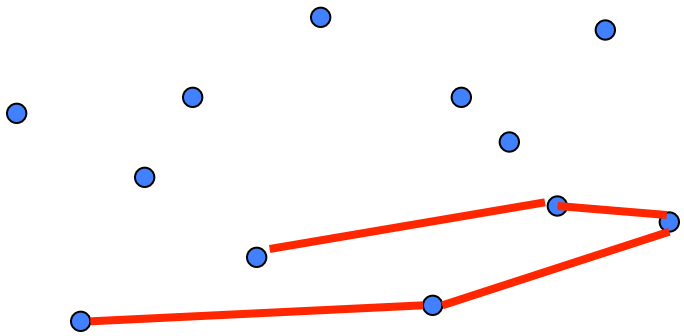
- Basic Idea: Identify the point with smallest y-coordinate to start.
- Sort remaining points by polar angle with respect to starting point.
- Put the first three points (including start) into a stack.
- For each next point, consider the angle formed by that point plus the two top elements of the stack.
  - For as long as this angle is a non-left turn, pop the element from the top of the stack and consider the next angle.
  - When the angle corresponds to a left turn, add this point to the stack
- Since points are added to the stack once, and removed at most once, the entire search is O(n) once the points are sorted.



Convex Hull Algorithms

Graham-Scan: Another O(n log n) approach

- Basic Idea: Identify the point with smallest y-coordinate to start.
- Sort remaining points by polar angle with respect to starting point.
- Put the first three points (including start) into a stack.
- For each next point, consider the angle formed by that point plus the two top elements of the stack.
  - For as long as this angle is a non-left turn, pop the element from the top of the stack and consider the next angle.
  - When the angle corresponds to a left turn, add this point to the stack
- Since points are added to the stack once, and removed at most once, the entire search is O(n) once the points are sorted.

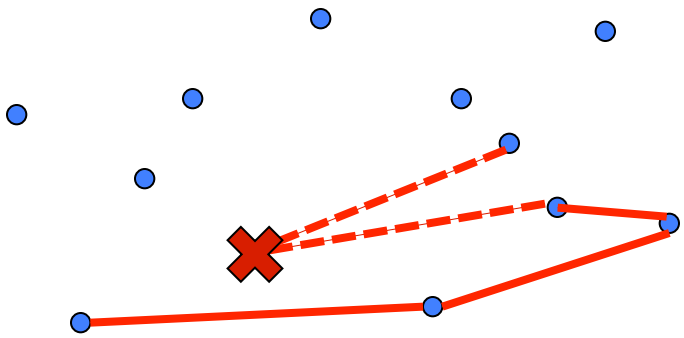


OK at this point, but not for long…

Convex Hull Algorithms

Graham-Scan: Another O(n log n) approach

- Basic Idea: Identify the point with smallest y-coordinate to start.
- Sort remaining points by polar angle with respect to starting point.
- Put the first three points (including start) into a stack.
- For each next point, consider the angle formed by that point plus the two top elements of the stack.
  - For as long as this angle is a non-left turn, pop the element from the top of the stack and consider the next angle.
  - When the angle corresponds to a left turn, add this point to the stack
- Since points are added to the stack once, and removed at most once, the entire search is O(n) once the points are sorted.

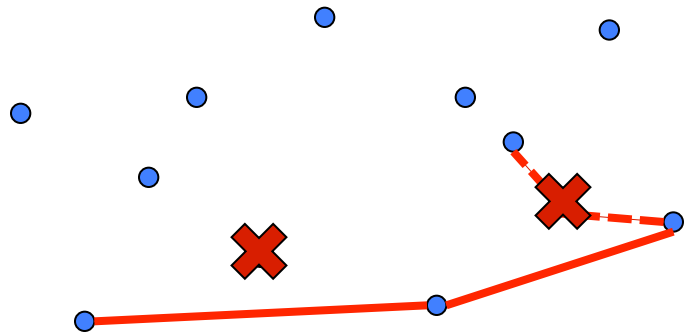


Right angle formed, so pop the top of the stack (the vertex of the angle)

Convex Hull Algorithms

Graham-Scan: Another O(n log n) approach

- Basic Idea: Identify the point with smallest y-coordinate to start.
- Sort remaining points by polar angle with respect to starting point.
- Put the first three points (including start) into a stack.
- For each next point, consider the angle formed by that point plus the two top elements of the stack.
  - For as long as this angle is a non-left turn, pop the element from the top of the stack and consider the next angle.
  - When the angle corresponds to a left turn, add this point to the stack
- Since points are added to the stack once, and removed at most once, the entire search is O(n) once the points are sorted.

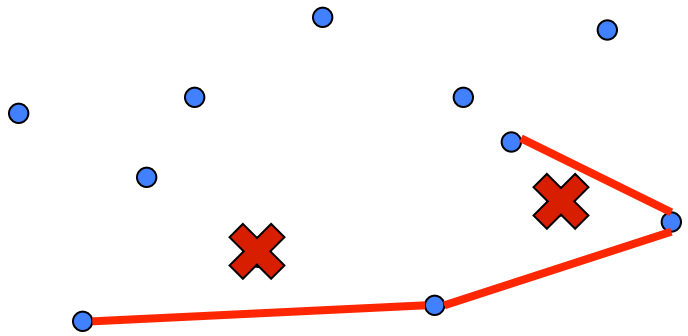


Makes a right turn, so we pop that vertex

Convex Hull Algorithms

Graham-Scan: Another O(n log n) approach

- Basic Idea: Identify the point with smallest y-coordinate to start.
- Sort remaining points by polar angle with respect to starting point.
- Put the first three points (including start) into a stack.
- For each next point, consider the angle formed by that point plus the two top elements of the stack.
  - For as long as this angle is a non-left turn, pop the element from the top of the stack and consider the next angle.
  - When the angle corresponds to a left turn, add this point to the stack
- Since points are added to the stack once, and removed at most once, the entire search is O(n) once the points are sorted.



OK at this point, but not for long…

Convex Hull Algorithms

Graham-Scan: Details

- If multiple points have the same lowest y-coordinate value, choose the leftmost one as the starting point.
- Polar angle is taken with respect to horizontal line through the starting point.
  - In this case, computing the polar angle just uses:

    $$\tan(\theta) = \Delta y / \Delta x$$
- After sorting by polar angle with respect to the starting point, if there are multiple points with the same angle, retain only the one that is farthest away.
  - The others can't be vertices of the convex hull.