UVA DATA SCIENCE

Foundation of Computer Science for Data Science

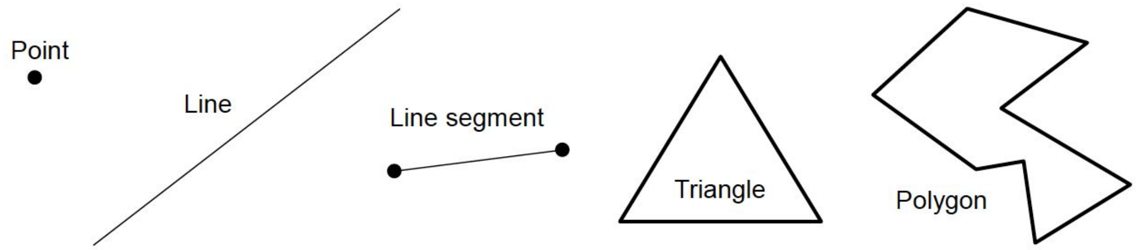
Computational Geometry
Convex Hull

Mai Dahshan

November 18, 2024

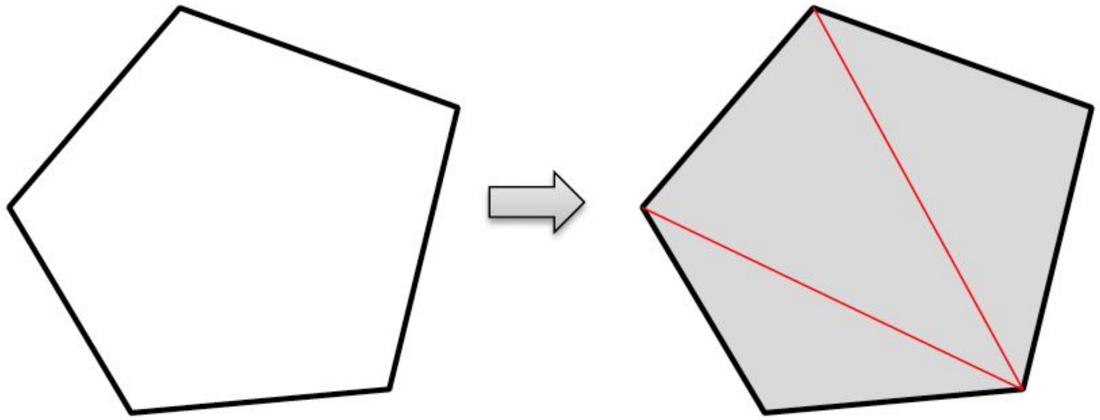
Computational Geometry

- Computational Geometry is a subfield of the Design and Analysis of Algorithms
- It deals with efficient data structures and algorithms for geometric problems (i.e., involving geometric input and output)



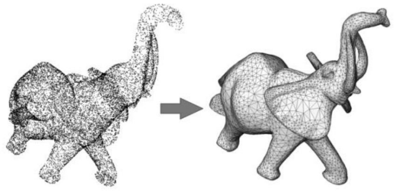
Computational Geometry - Example

- How to fill the inside of an n-vertex 2D polygon with n-2 triangles?



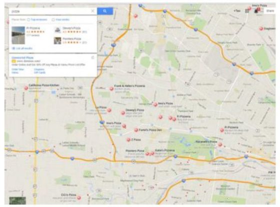
Computational Geometry - Applications

- Computer graphics
  - Surface construction
  - Collision detection



- Computer vision
  - Pattern recognition

- Geographical Information System
  - Range queries

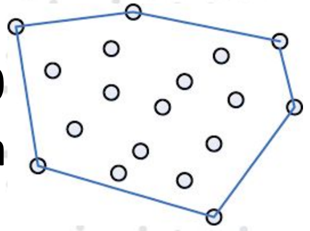


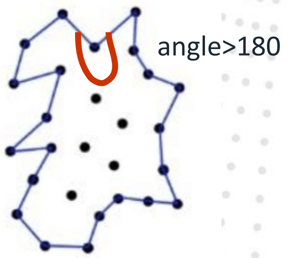
"Pizza near me"

UVA DATA SCIENCE

Basic Terminologies

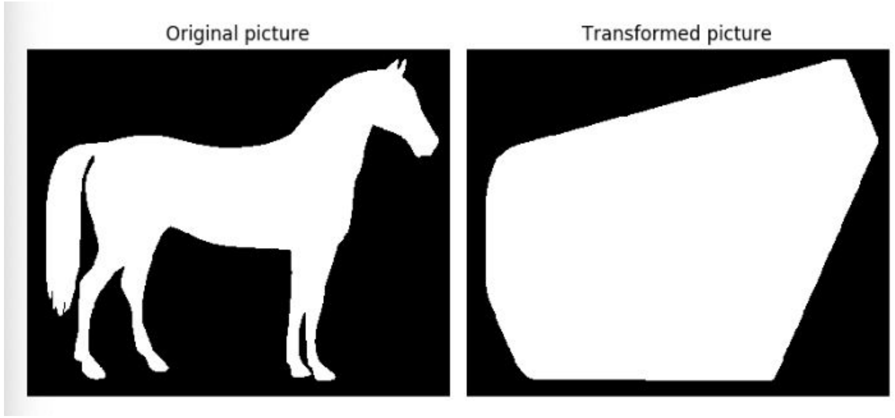
- A **polygon** is a two-dimensional geometric shape consisting of a finite number of straight line segments connected to form a closed chain or circuit.
- A polygon is **convex** if: 1) All interior angles are less than 180 ; 2) A line segment connecting any two points inside or on th boundary of the polygon lies entirely within the polygon
- A polygon is **concave** if: 1) At least one interior angle is greater than 180°; 2)A line segment connecting two points inside or on the boundary of the polygon can pass outside the polygon





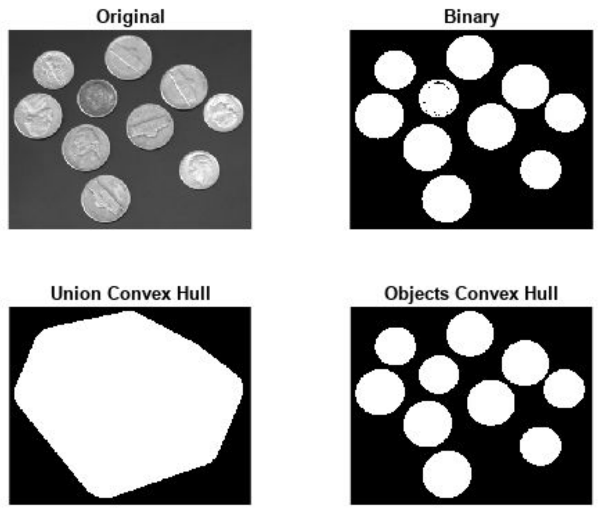
Convex Hull Applications

- Image simplification for matching tasks or served as conditions for generative tasks



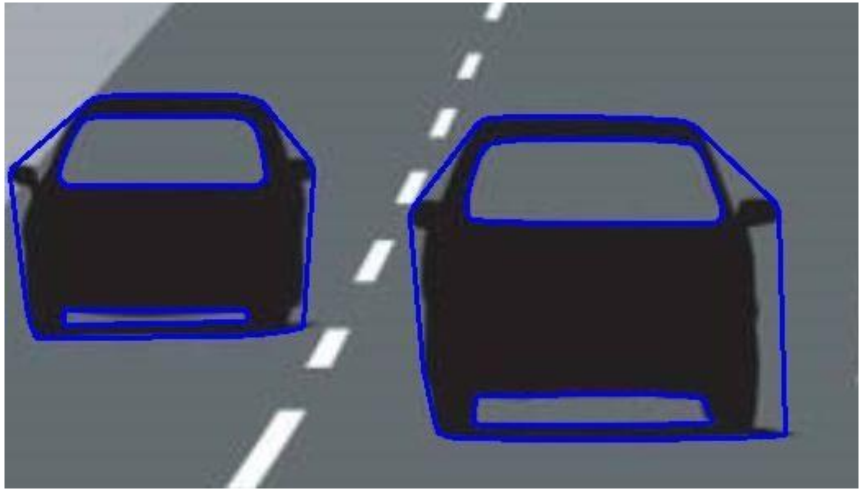
Convex Hull Applications

- Image segmentation problems: The segmented objects are usually given in convex hull for the sake of simplicity



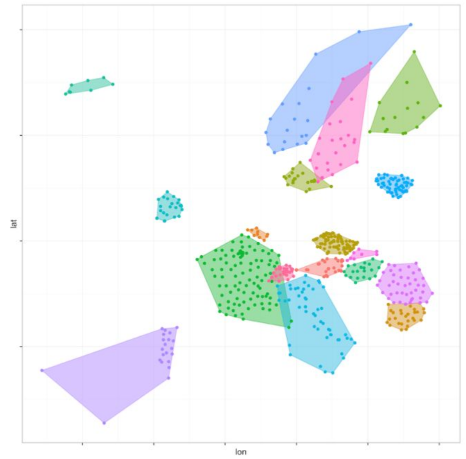
Convex Hull Applications

- In engineering problems, it is used in path planning and collision detection in robotic and autonomous vehicles



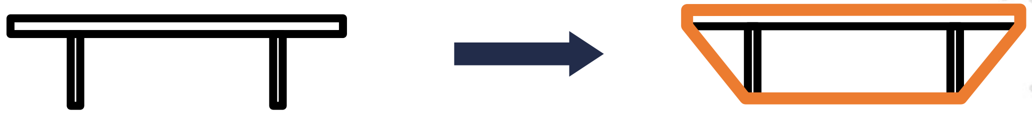
Convex Hull Applications

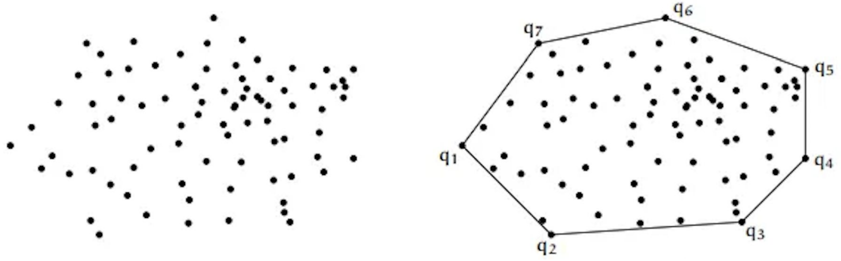
- Clusters Interference Detection
  - Given data clustered into groups, how do we know if these groups overlap or interference to each others?
  - Convex hulls for each groups can be created, to detect interference



Convex Hull Problem

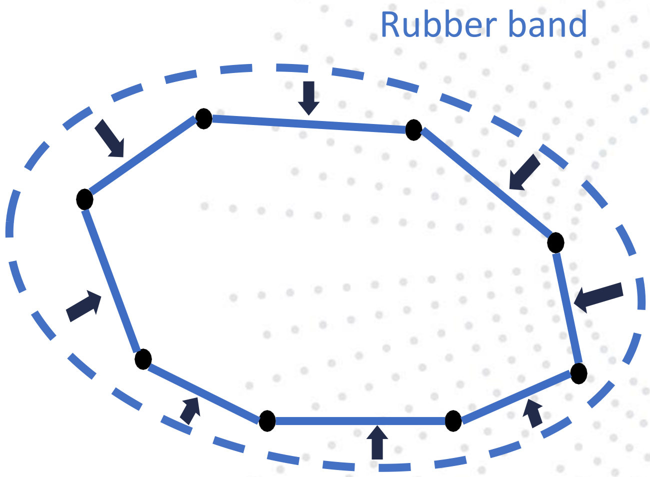
- Convex Hull Problem find the smallest <u>convex</u> polygon that bounds a shape (or more generally, a collection of points)





Convex Hull Problem

- **Rubber band analogy:** imagine the points are nails sticking out of a board and wrapping a rubber band to encompass the nails; convex hull is resulting shape

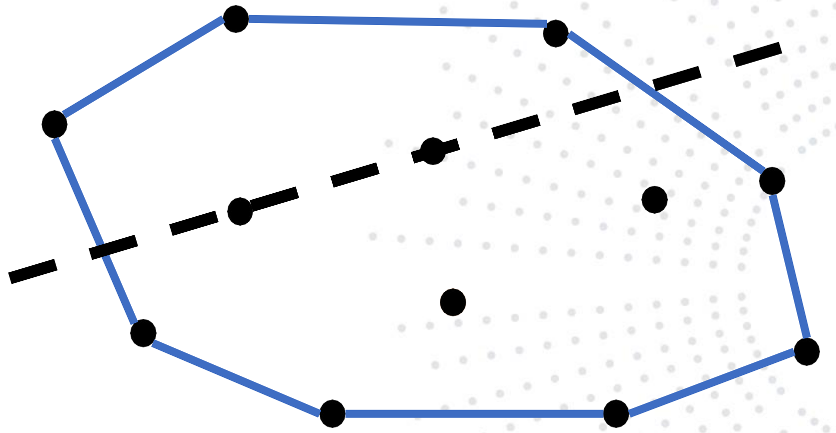


Convex Hull Problem Algorithms

- Several algorithms can solve the convex hull problem
    - A Brute Force Approach
    - Jarvis’ Algorithm (Gift Wrapping Method)
    - Graham’s Algorithm
    - Chan’s Algorithm

A Brute Force Approach

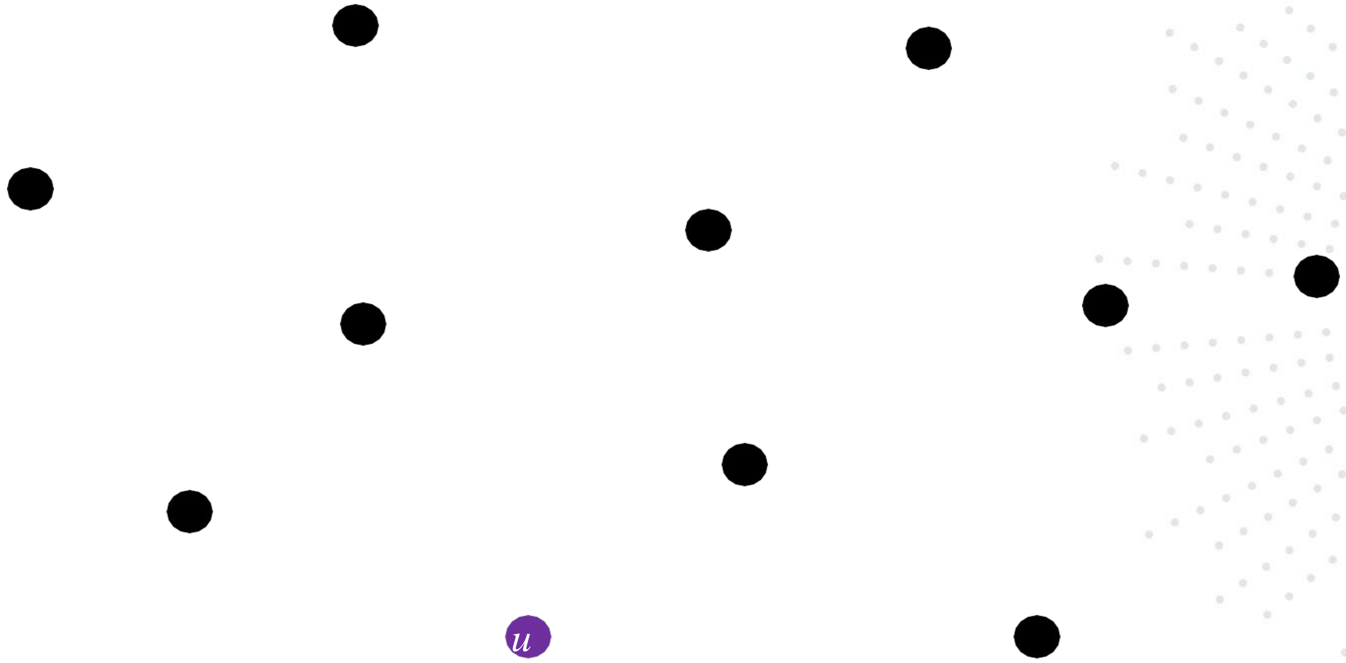
- Brute force approach: for every pair of points, check if all other points are on the same side of the line
- if there are points on both sides of the line, then the pair cannot be an edge in the convex hull
- Time complexity O(n^3)



Jarvis' Algorithm (Gift Wrapping Method)

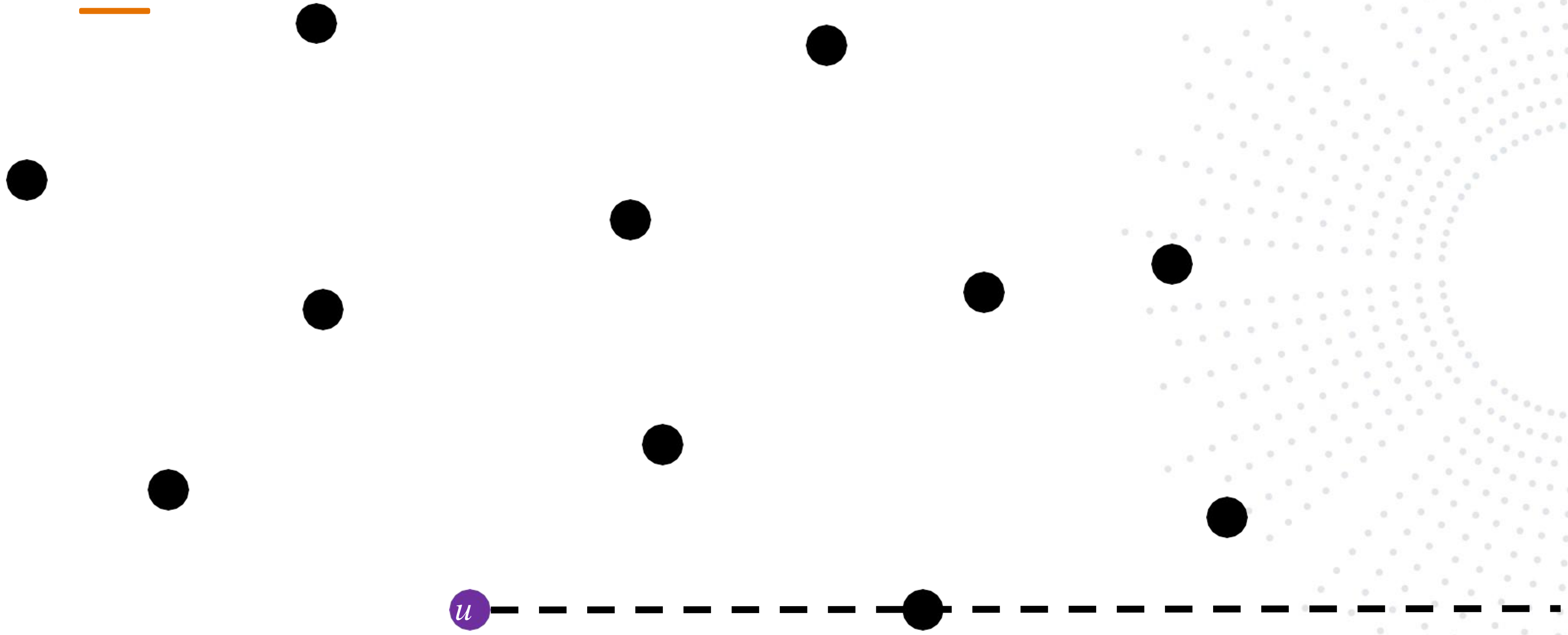
- Find the Leftmost , rightmost, bottommost point: Identify the point with the smallest or largest x-coordinate (if there are ties, use the smallest y-coordinate). This point will be the starting point of the convex hull.
- Initialize the current point on the hull as the leftmost or rightmost point
- Add this point to the convex hull
- For each point P in the set of points:
  - Check whether P is move clockwise or counter-clockwise relative to the current segment. Set the newly identified point as the current hull point
- Continue the process by repeating the previous step for the next hull point

Jarvis' Algorithm (Gift Wrapping Method)

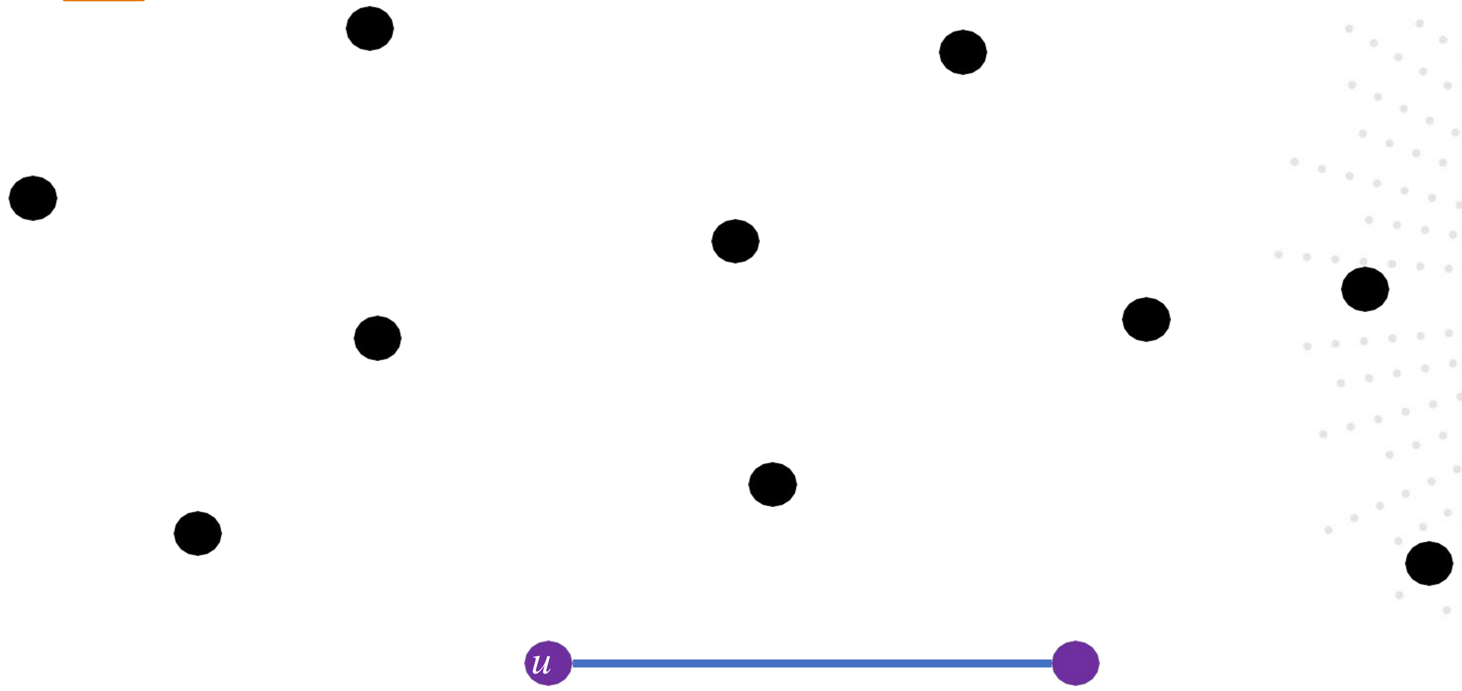


Idea: Start with leftmost,rightmost, bottommost, topmost point and "wrap" points in counter-clockwise fashion

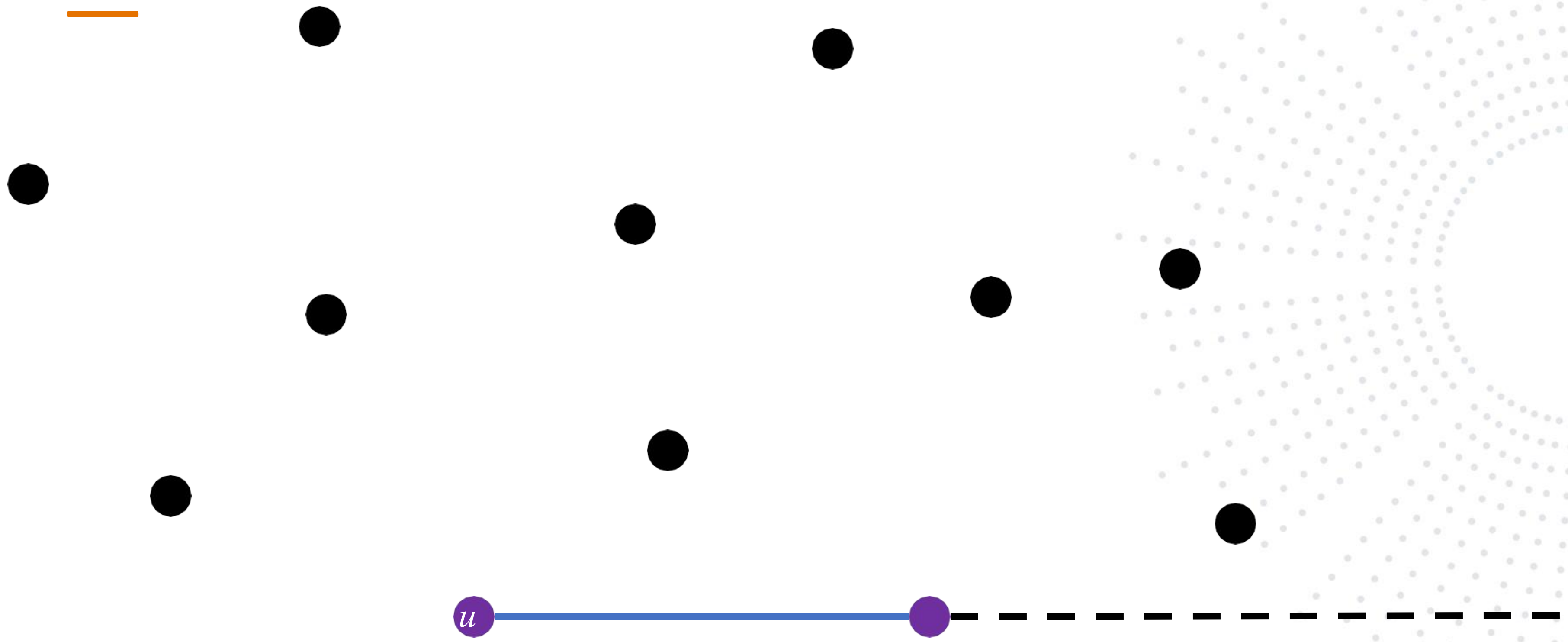
Jarvis' Algorithm (Gift Wrapping Method)



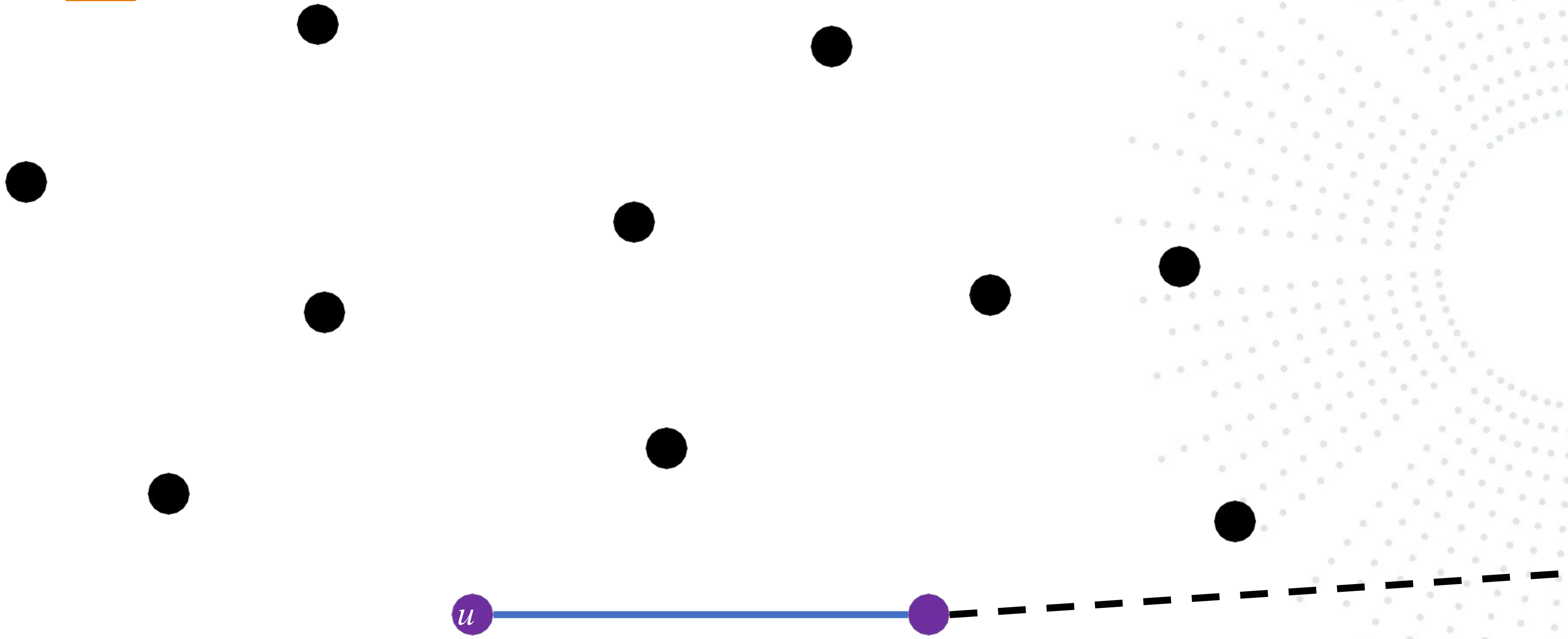
Jarvis' Algorithm (Gift Wrapping Method)



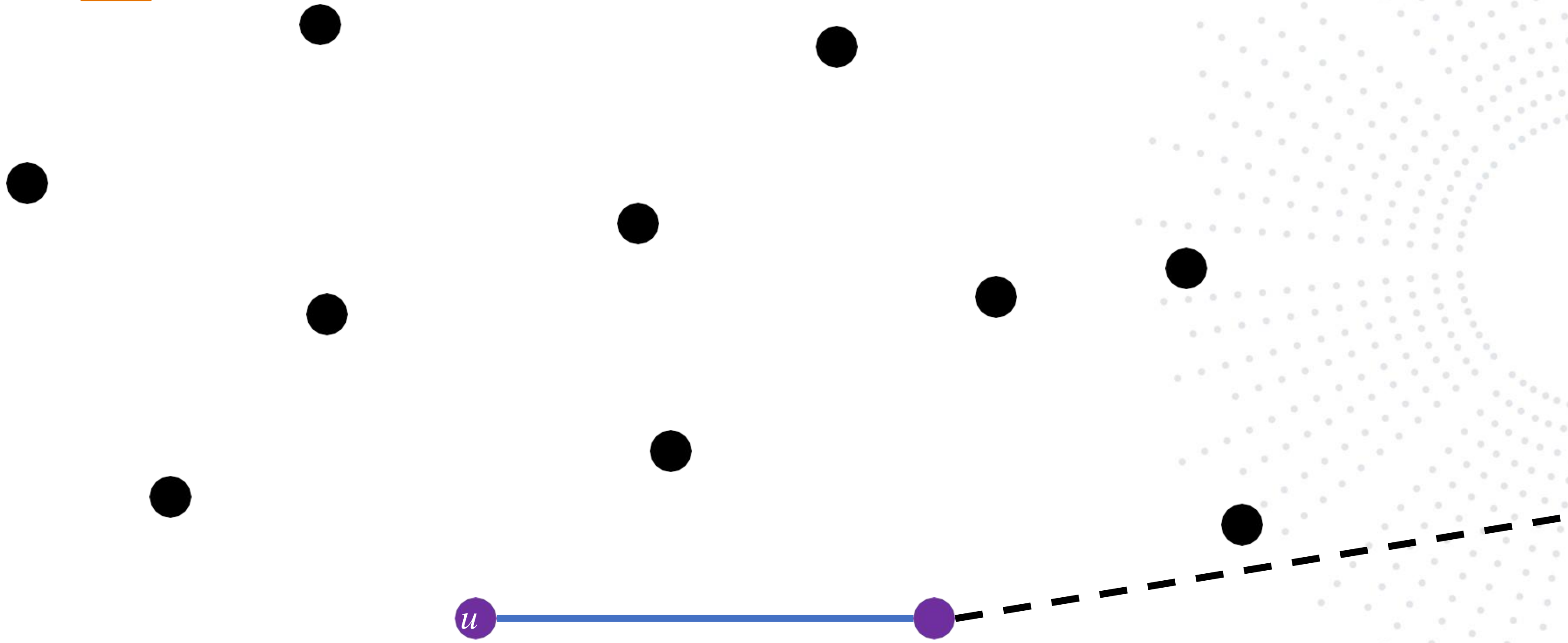
Jarvis' Algorithm (Gift Wrapping Method)



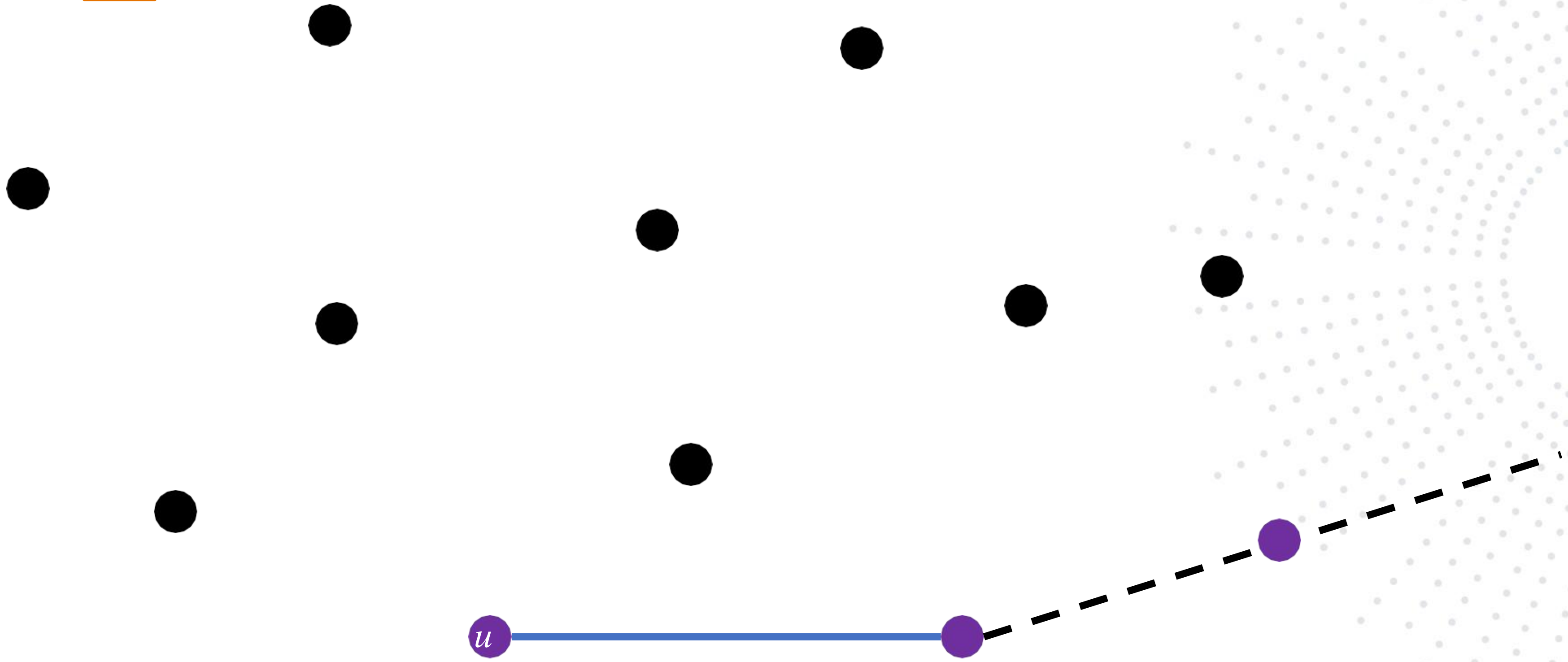
Jarvis' Algorithm (Gift Wrapping Method)



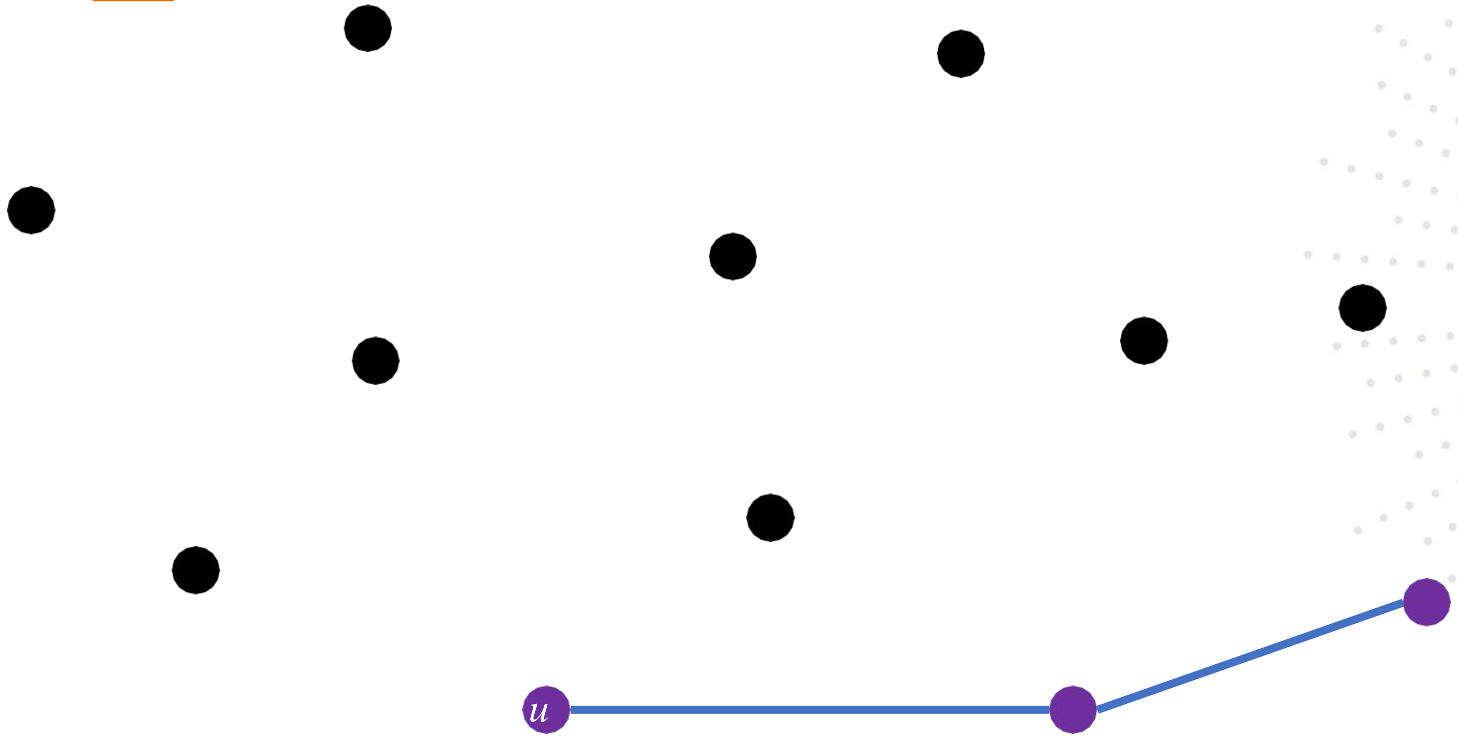
Jarvis' Algorithm (Gift Wrapping Method)



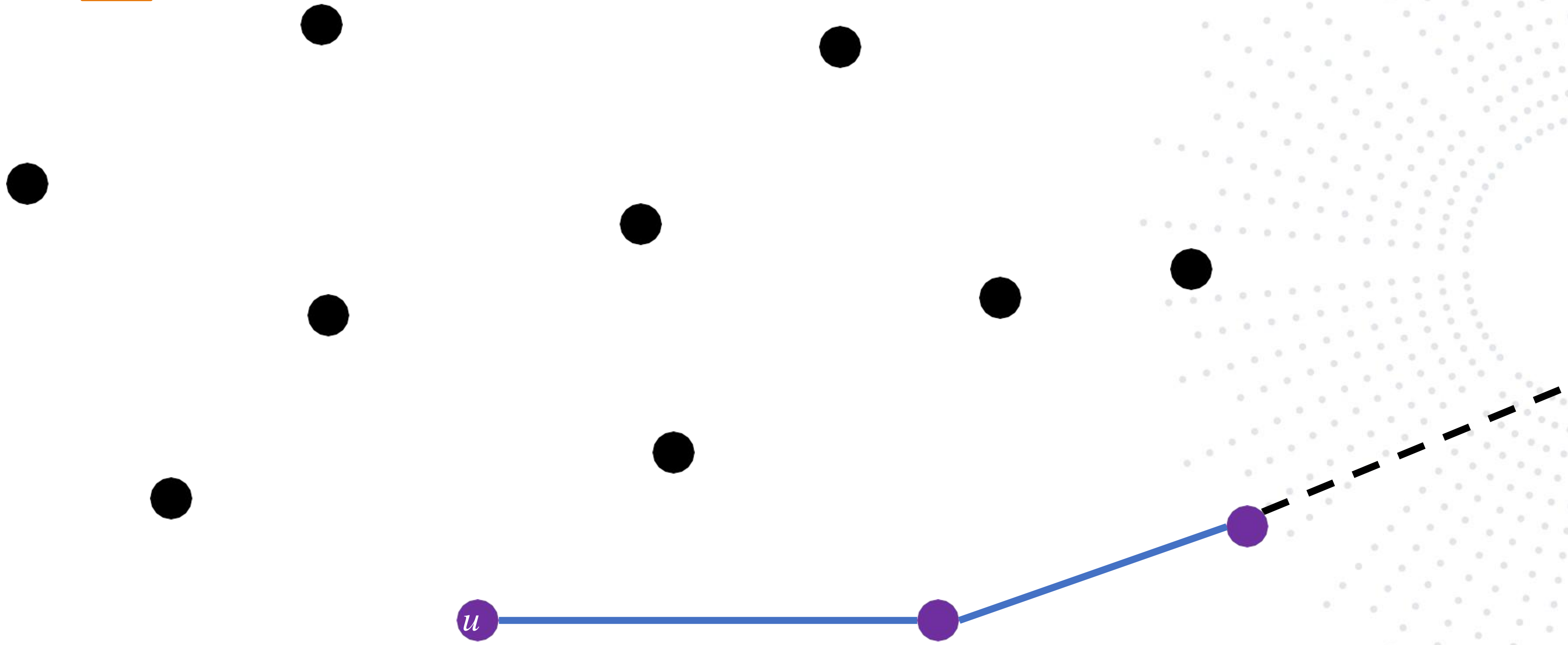
Jarvis' Algorithm (Gift Wrapping Method)



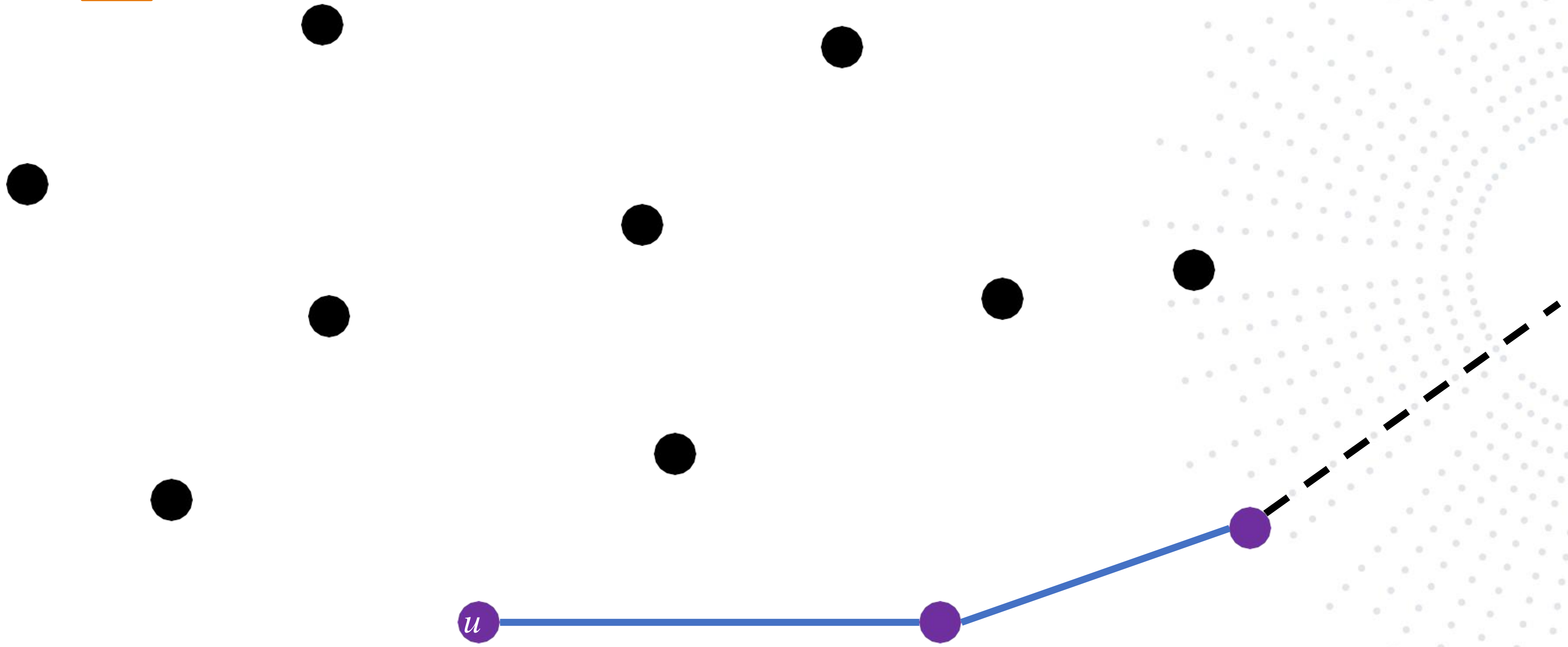
Jarvis' Algorithm (Gift Wrapping Method)



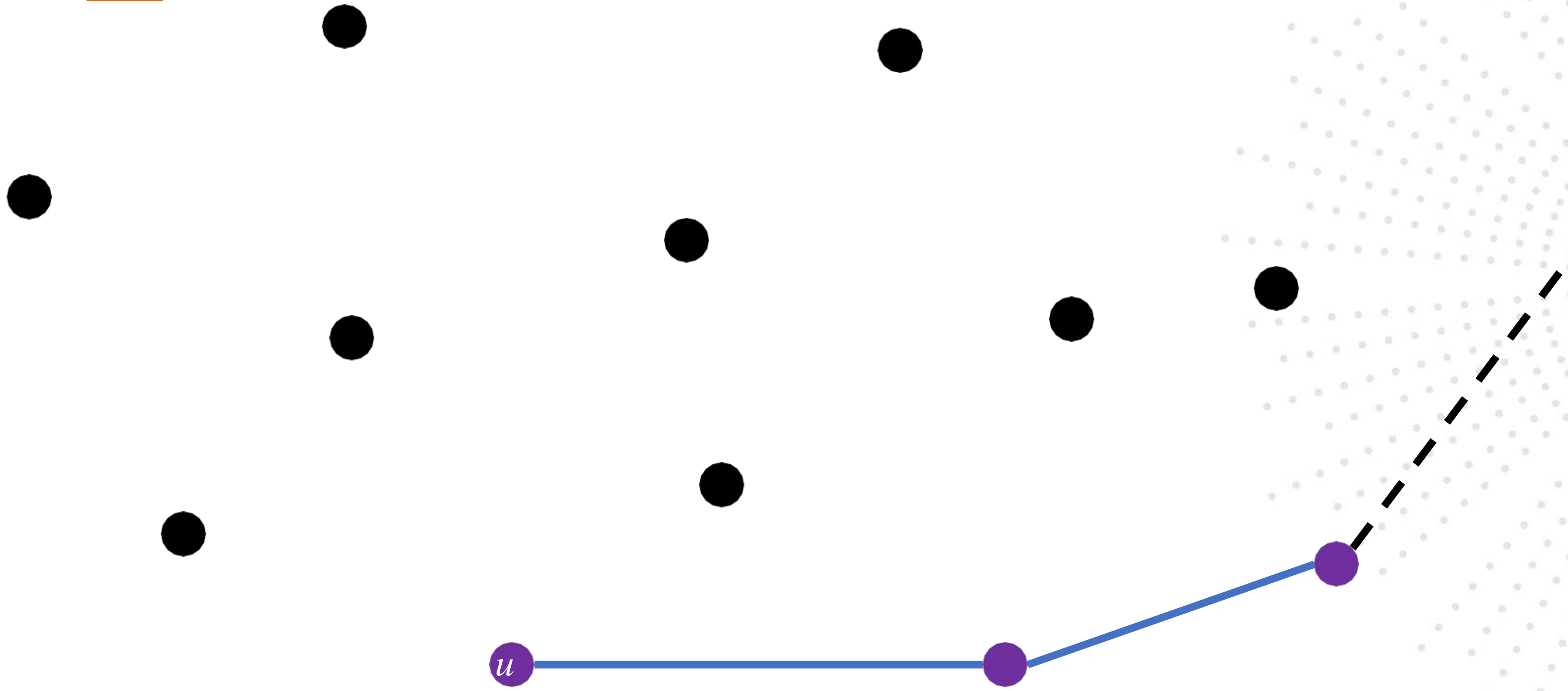
Jarvis' Algorithm (Gift Wrapping Method)



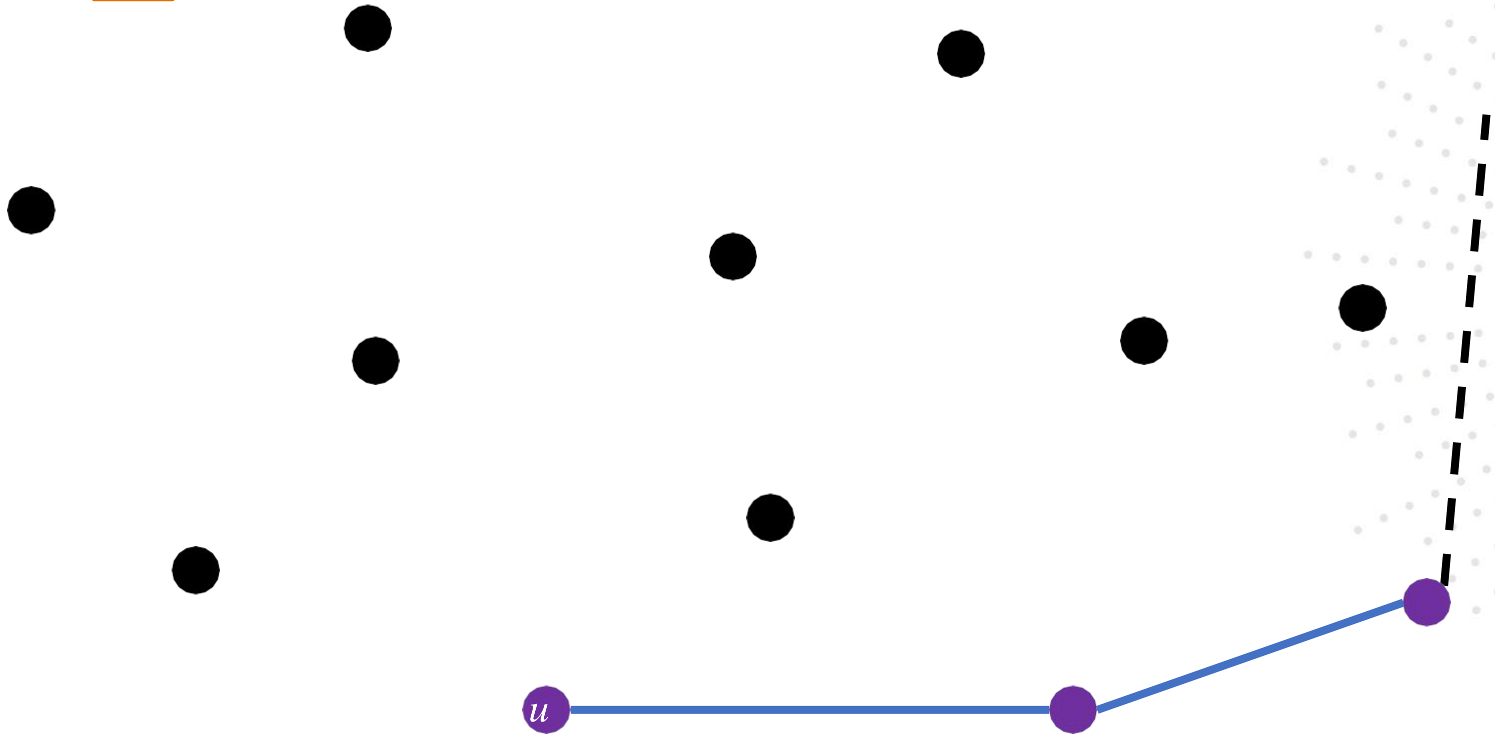
Jarvis' Algorithm (Gift Wrapping Method)



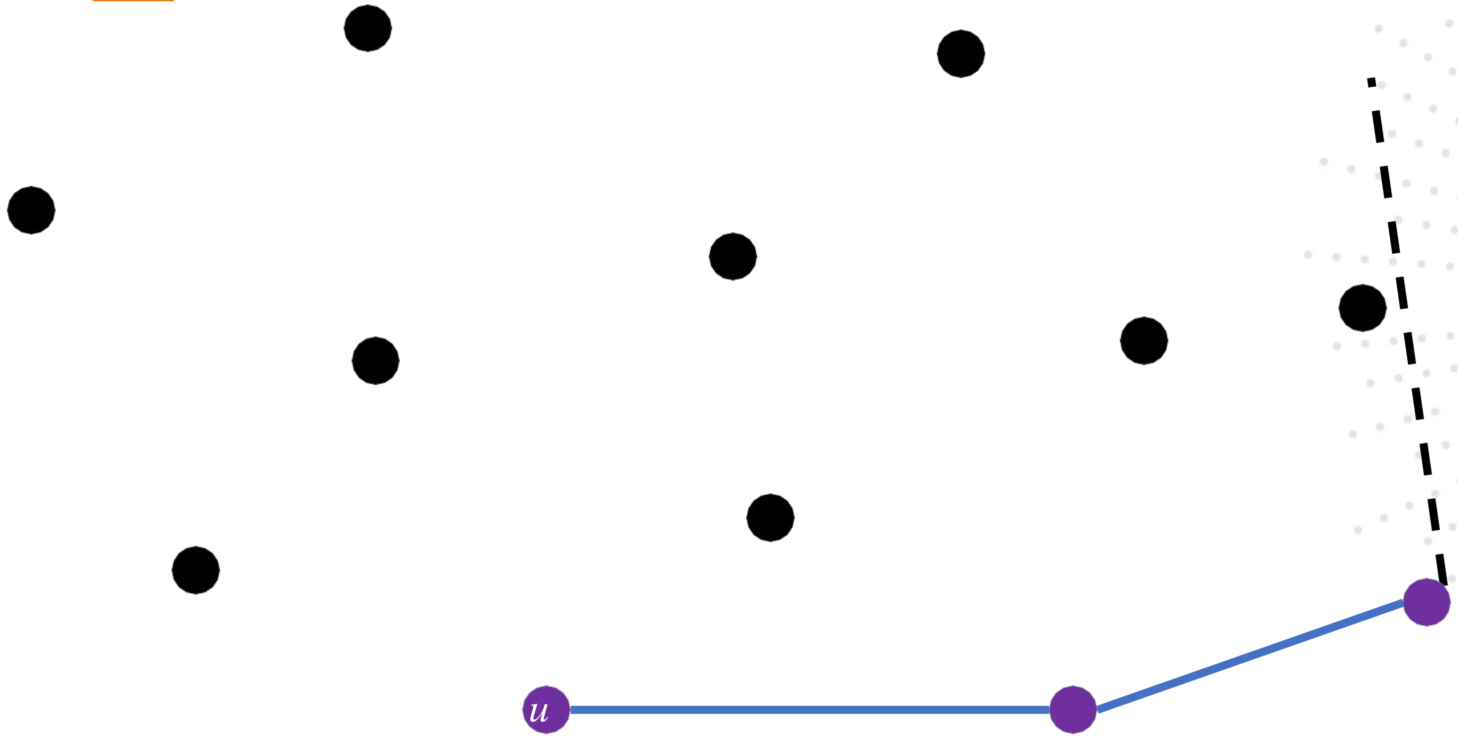
Jarvis' Algorithm (Gift Wrapping Method)



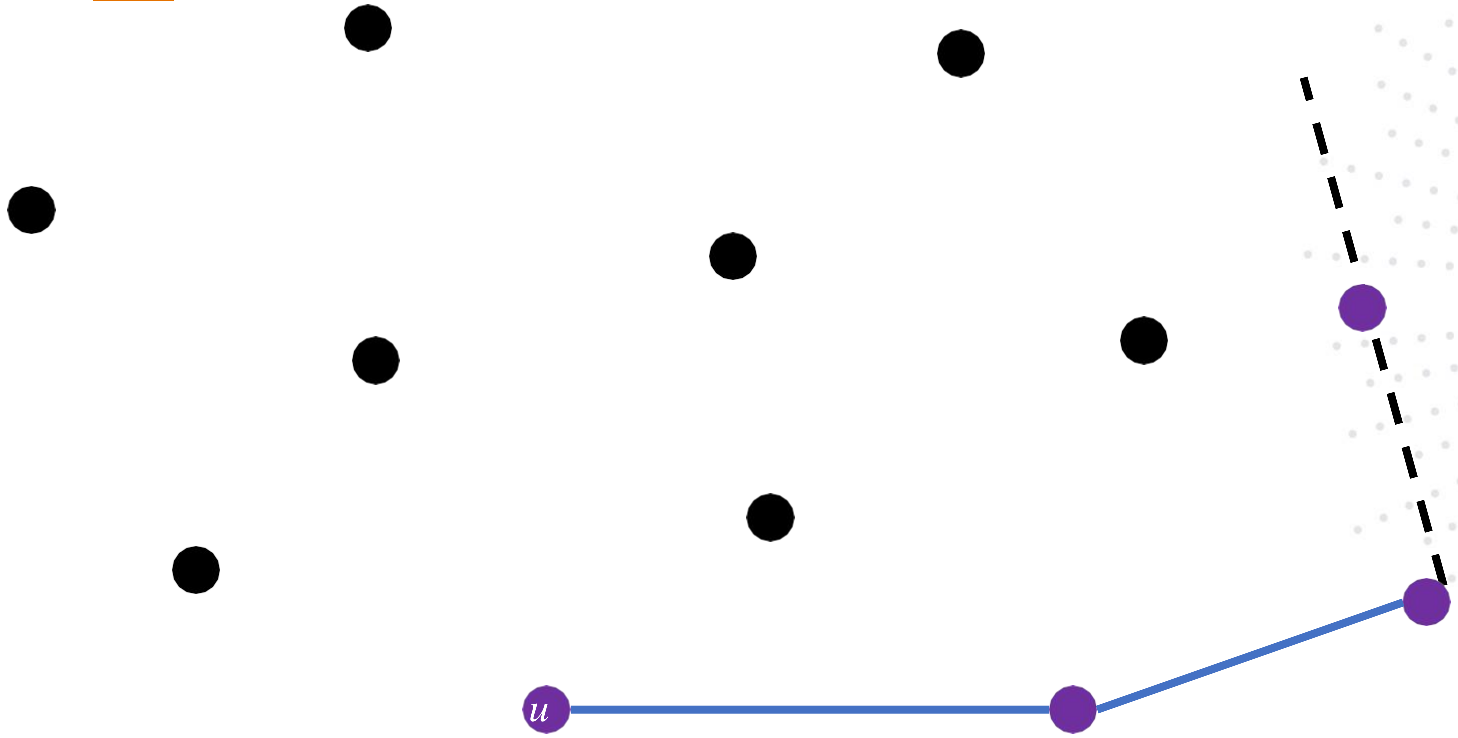
Jarvis' Algorithm (Gift Wrapping Method)



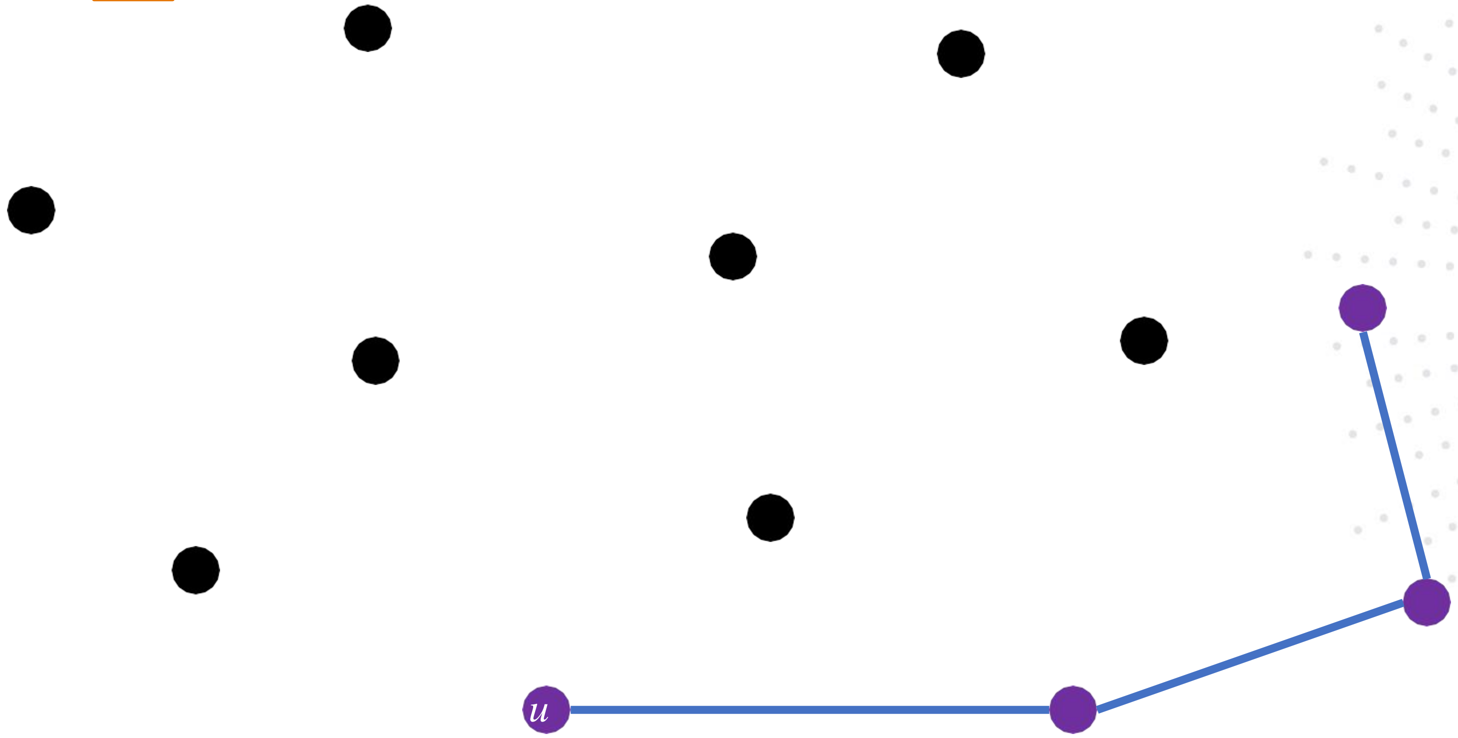
Jarvis' Algorithm (Gift Wrapping Method)



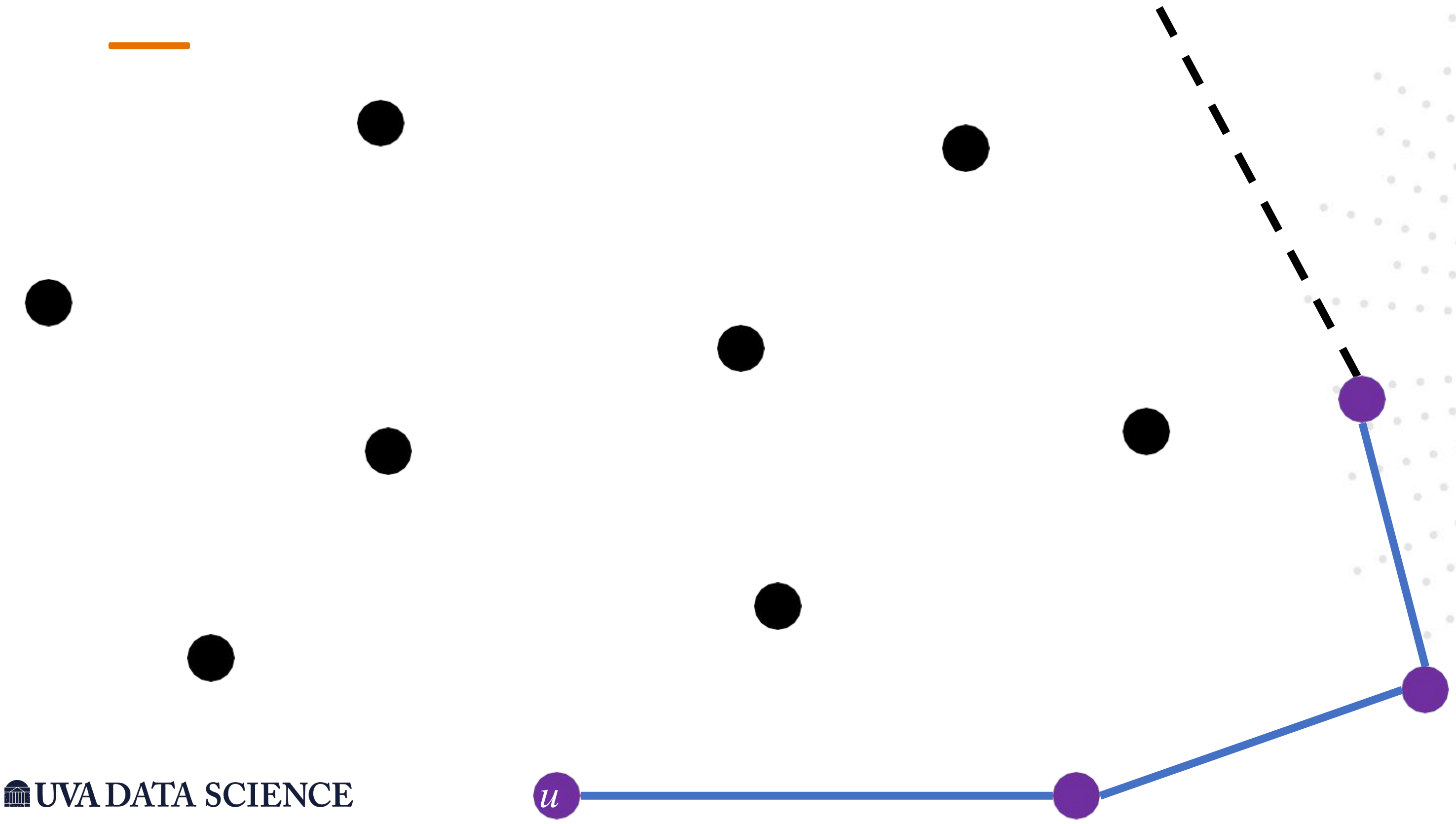
Jarvis' Algorithm (Gift Wrapping Method)



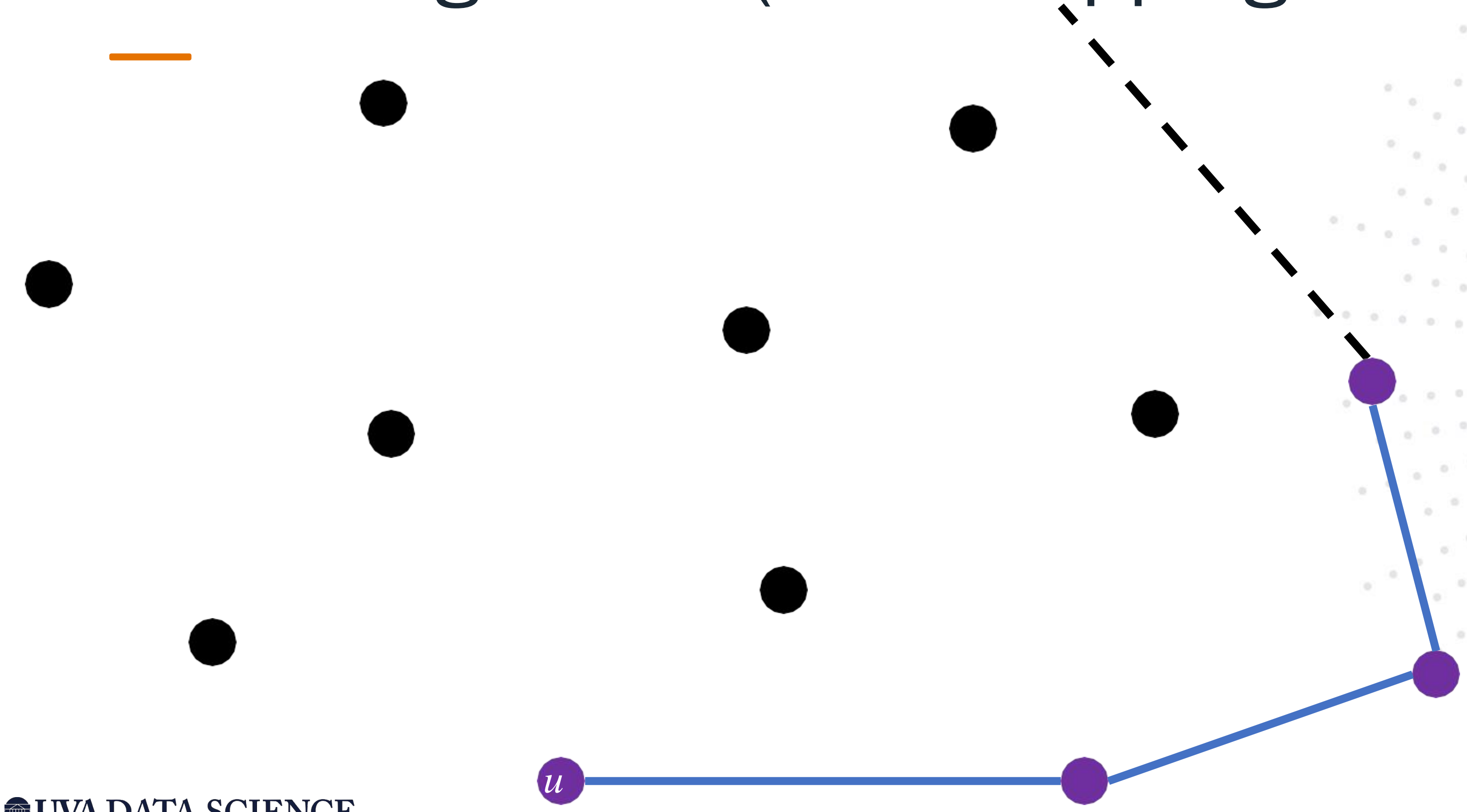
Jarvis' Algorithm (Gift Wrapping Method)



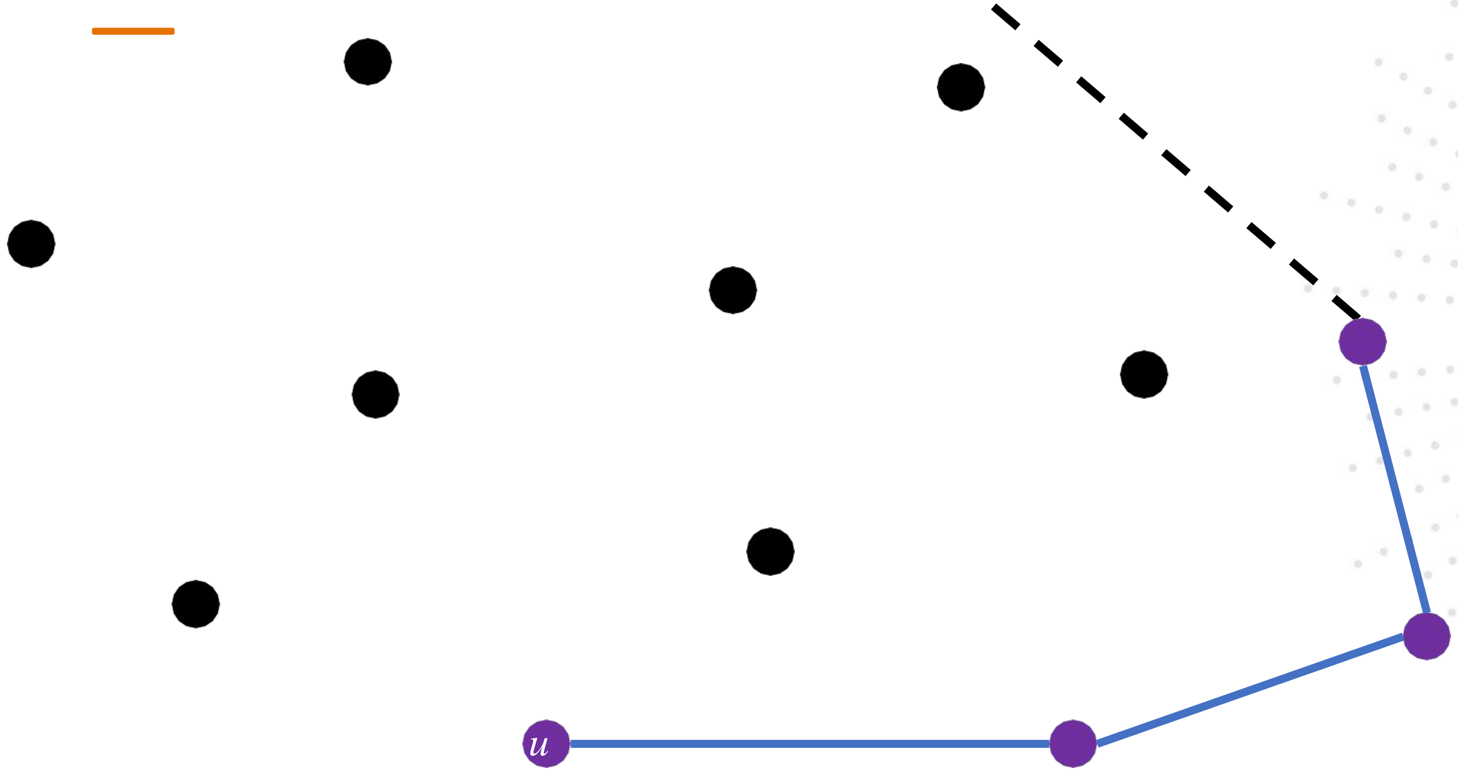
Jarvis' Algorithm (Gift Wrapping Method)



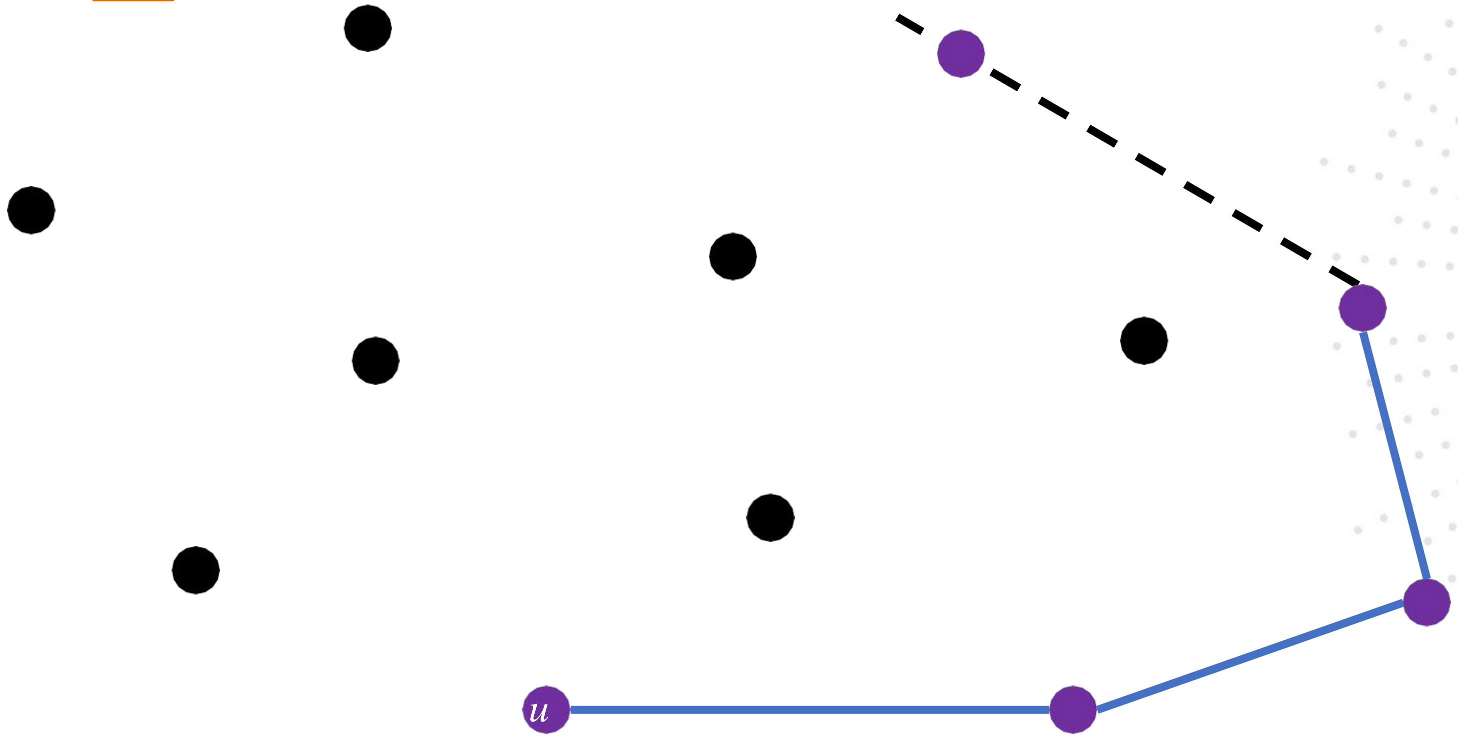
Jarvis' Algorithm (Gift Wrapping Method)



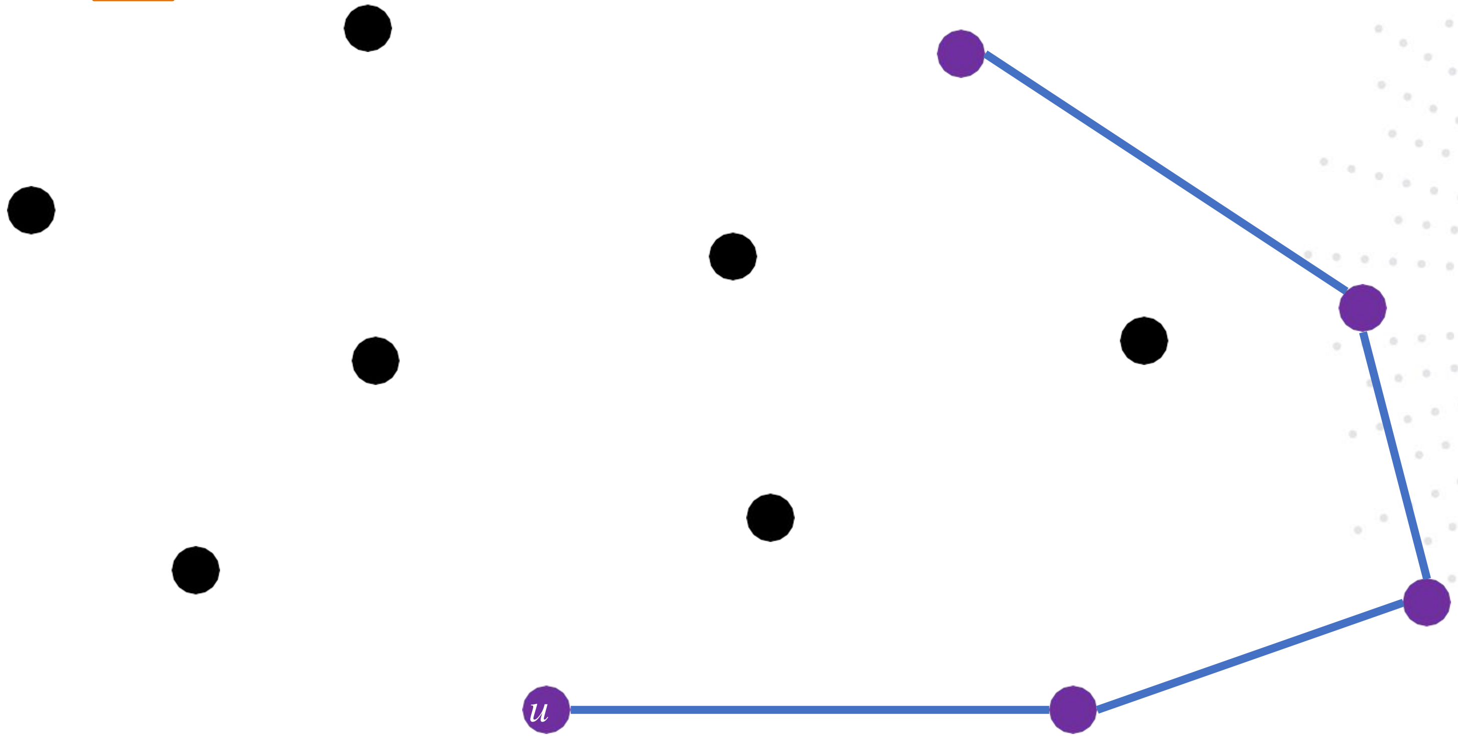
Jarvis' Algorithm (Gift Wrapping Method)



Jarvis' Algorithm (Gift Wrapping Method)

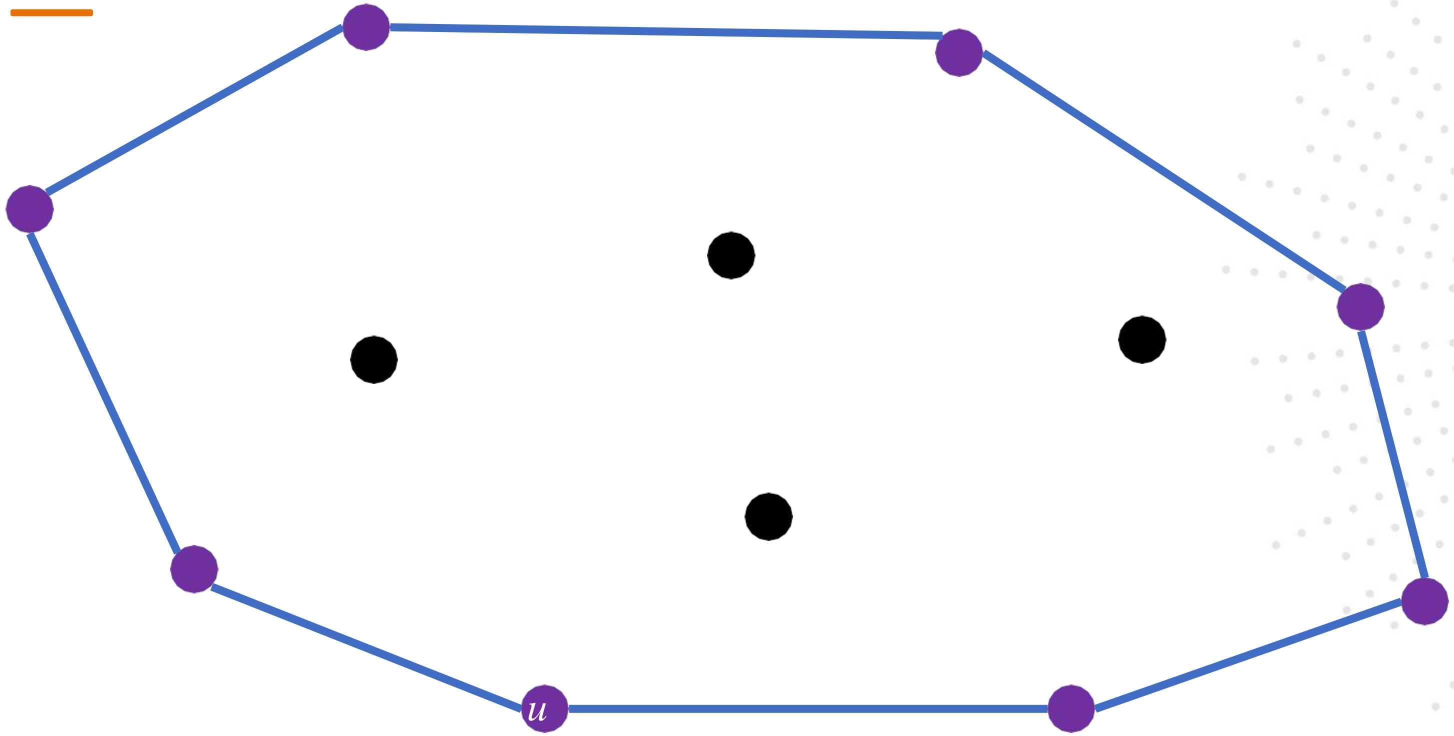


Jarvis' Algorithm (Gift Wrapping Method)



UVA DATA SCIENCE

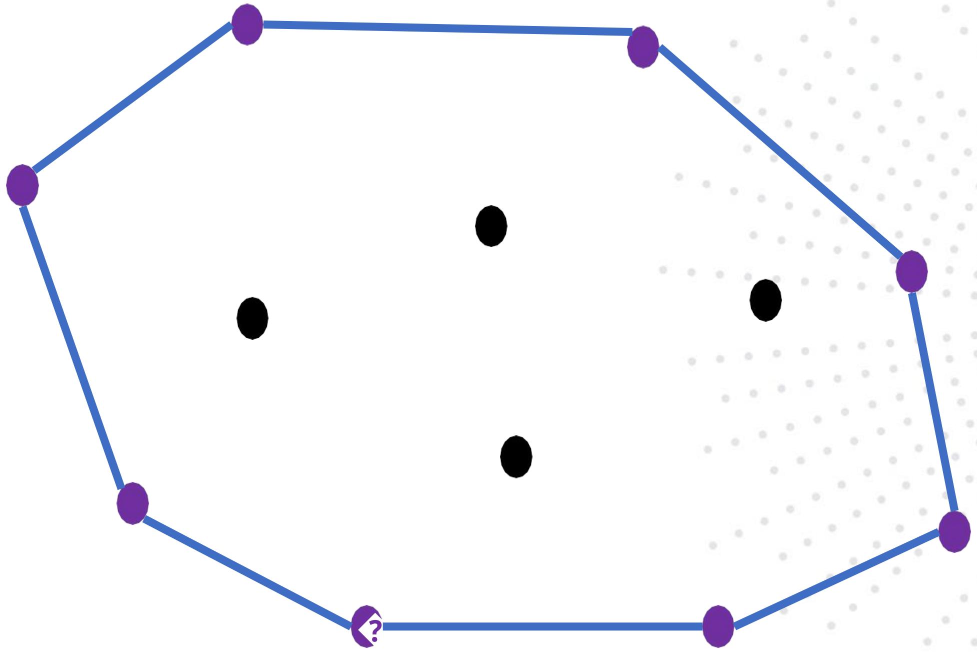
Jarvis' Algorithm (Gift Wrapping Method)



Jarvis’ Algorithm (Gift Wrapping Method)

Number of iterations: number of points on convex hull

Time Complexity: $O(nh)$ where h is the number of points on the convex hull and n is the number of points in the input set



Graham’s Algorithm

- Let p1 be the point with the smallest y-coordinate (and smallest x- coordinate if multiple points have the same minimum-y coordinate). This point is called the pivot or anchor point.
- Sort the remaining points by the polar angle relative to the pivot.
- Start with the pivot point as the first point in the hull (represented as **a stack**)
- Add the next two sorted points to the hull, as they will always be part of the convex hull.

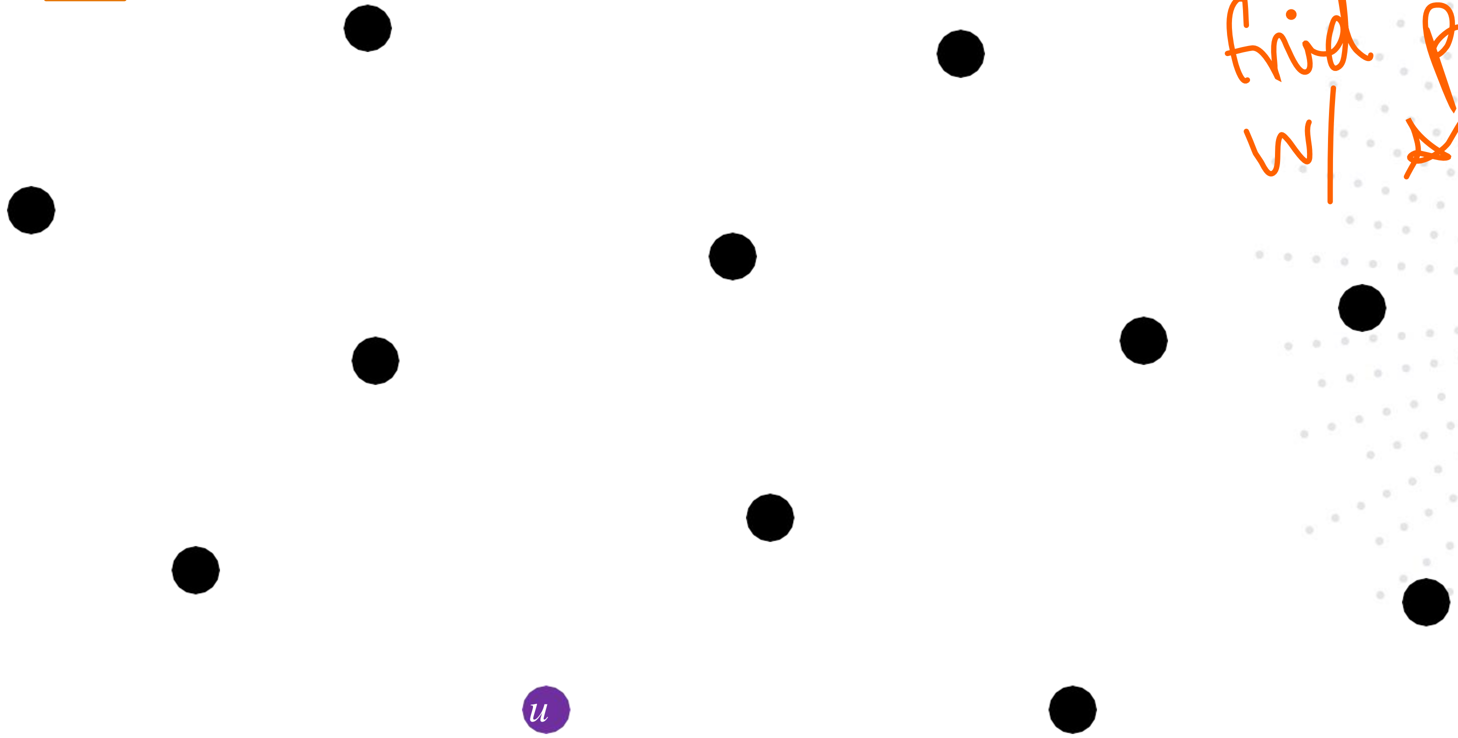
polar angle

Graham's Algorithm

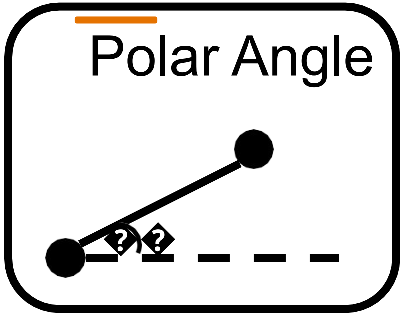
- For each point in order list, check if moving from the last point in the hull to the second-to-last point and then to the current point makes a left turn or a right turn:
  - If it makes a left turn (counterclockwise), the point is part of the convex hull, so add it to the hull.
  - If it makes a right turn (clockwise), remove the last point from the hull and check again with the new second-to-last point.
  - Repeat this process until the turn is counterclockwise.
- After processing all points, the points in the hull list represent the vertices of the convex hull in counterclockwise order

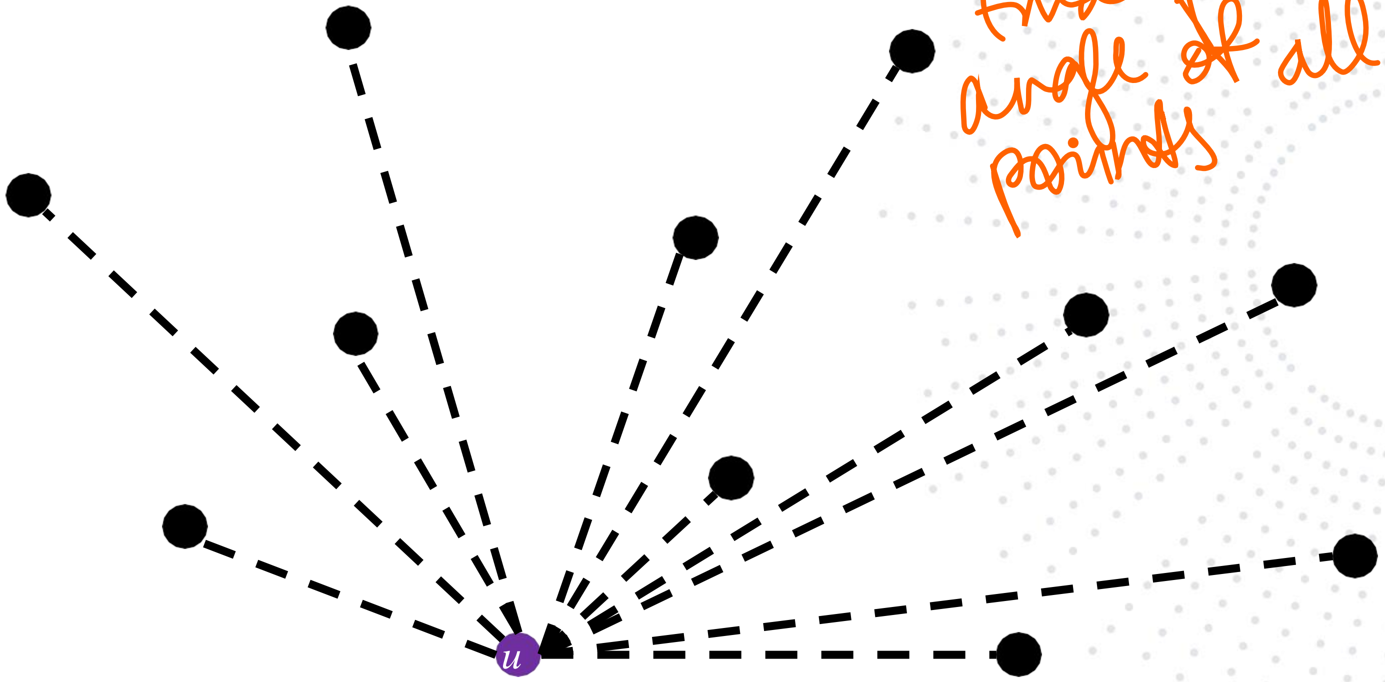
Graham's Algorithm

step 1
find point
w/ smallest
y-val



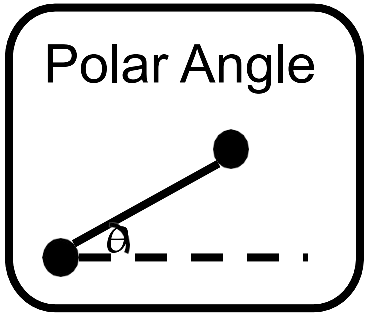
Graham's Algorithm


Polar Angle

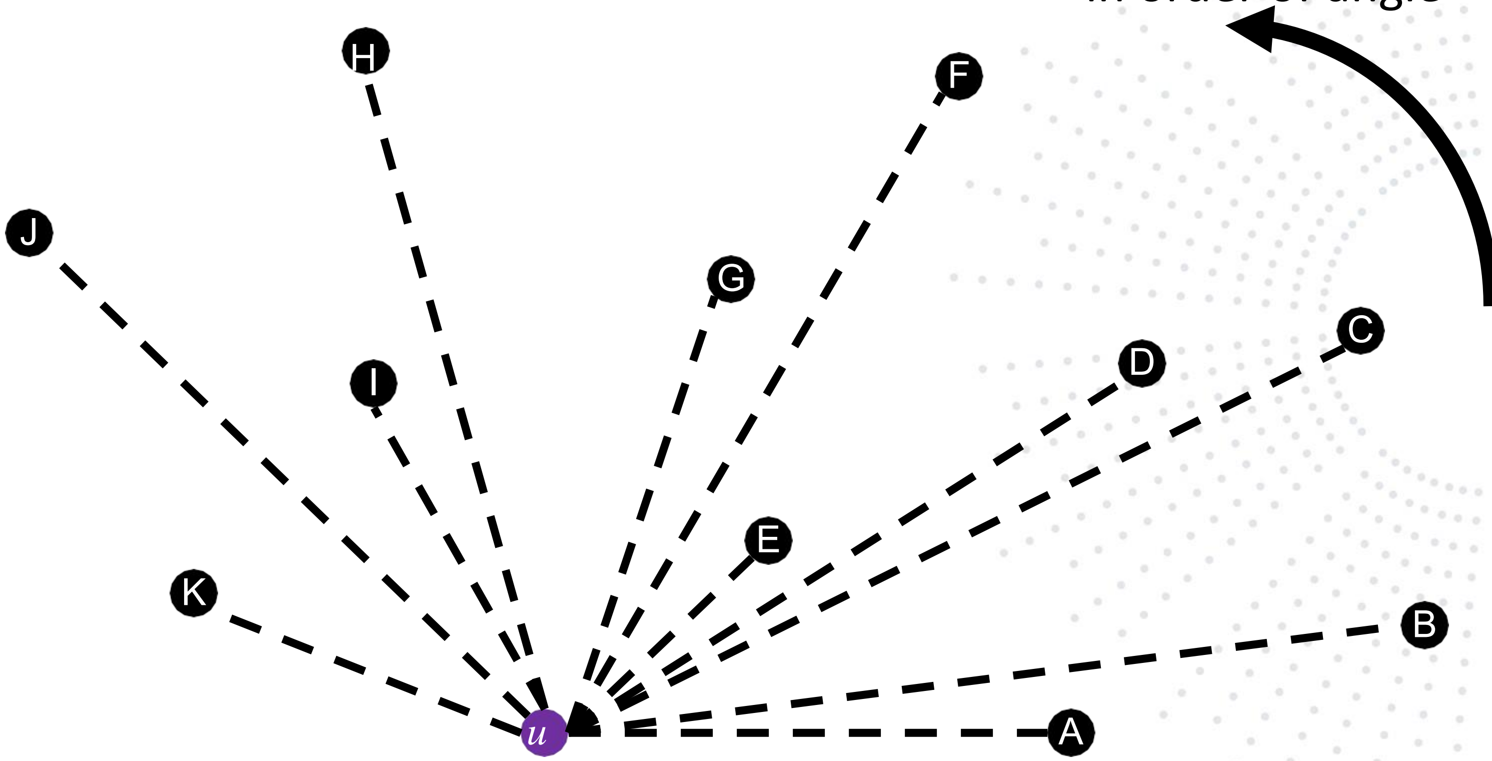


Step 2:
find polar angle of all points

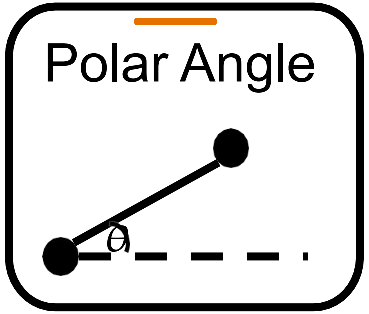
Graham’s Algorithm

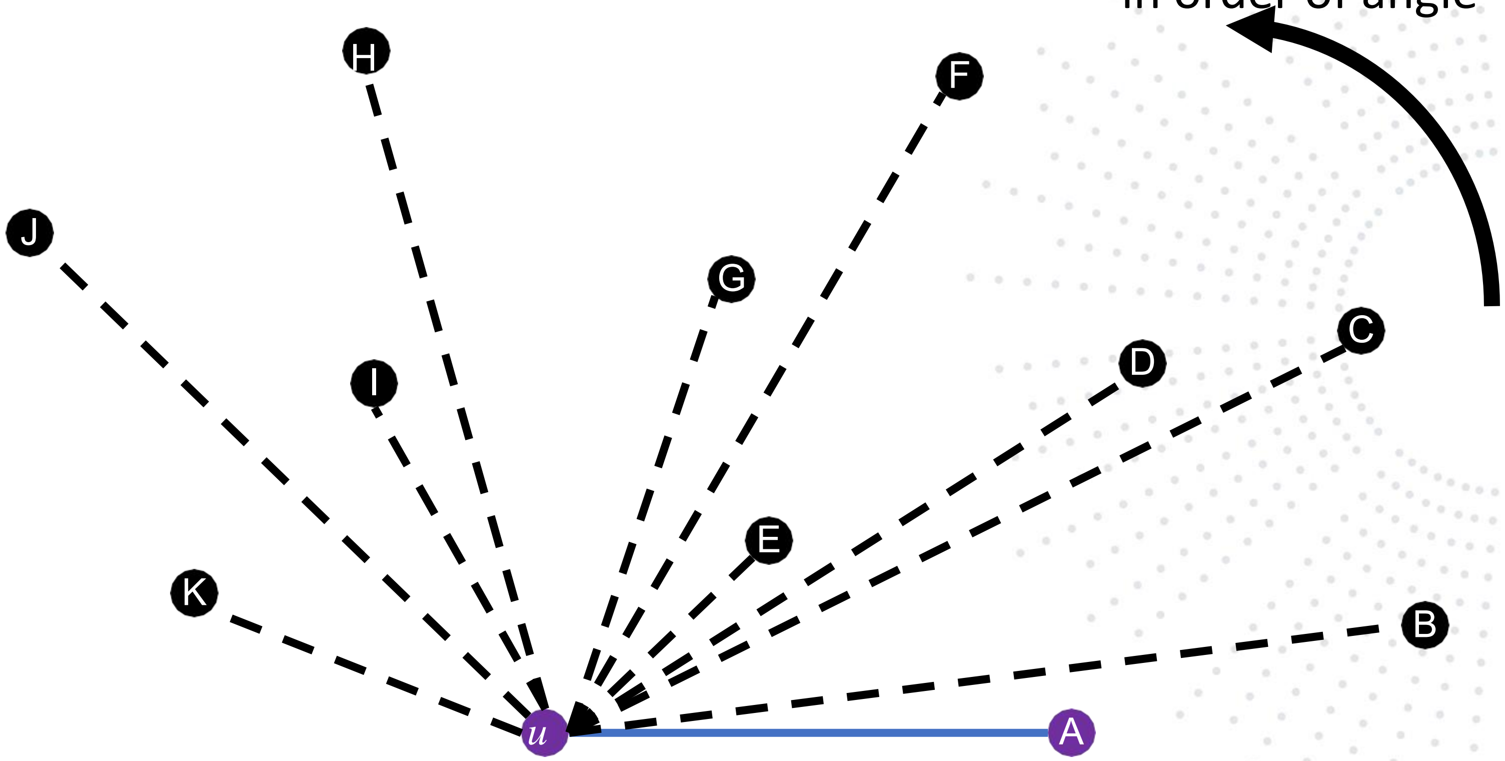


Scan the points in order of angle

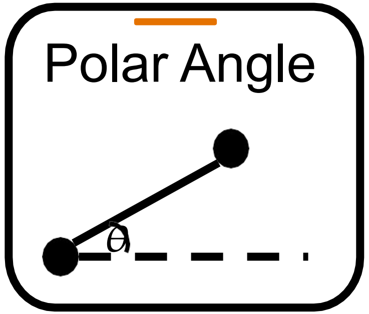


Graham's Algorithm

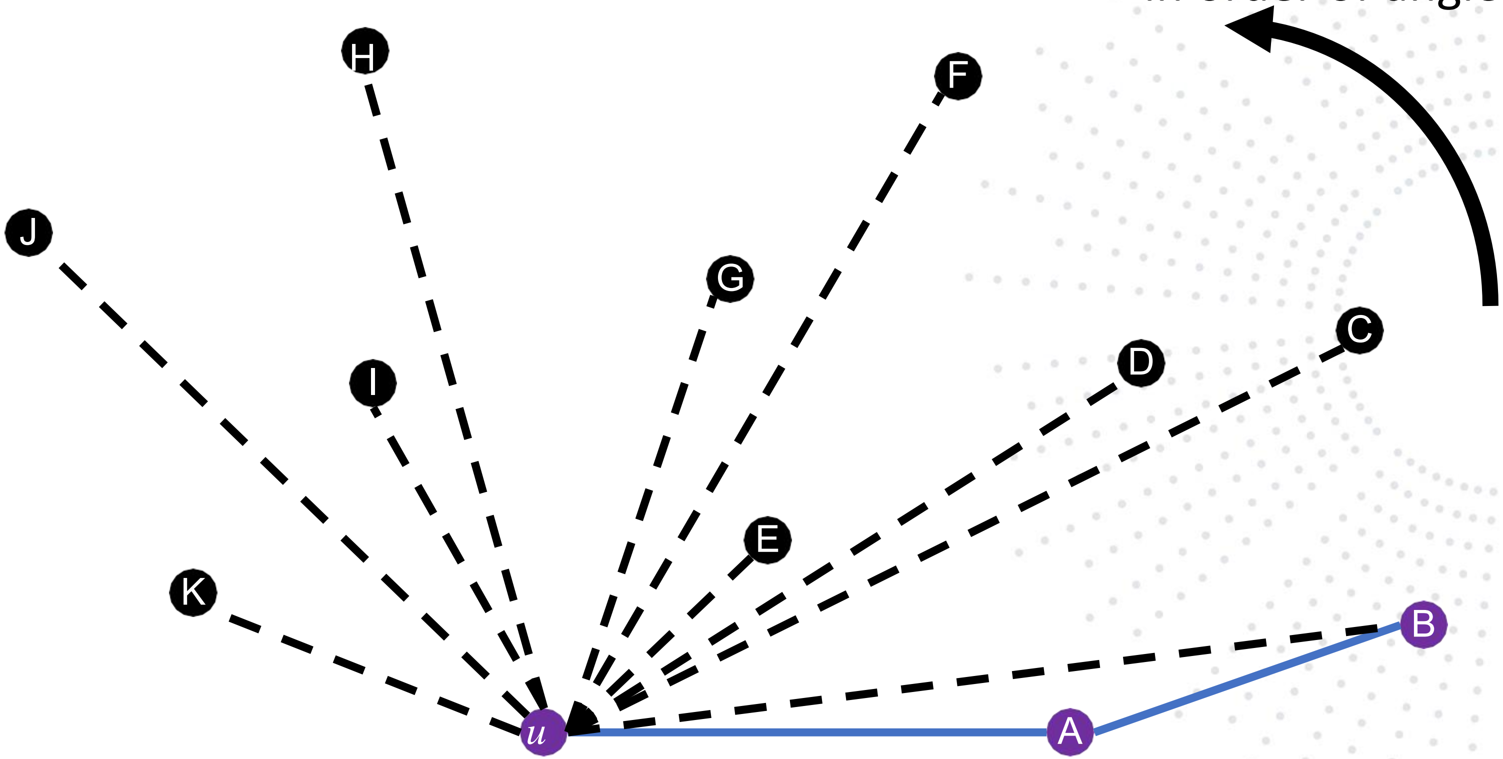




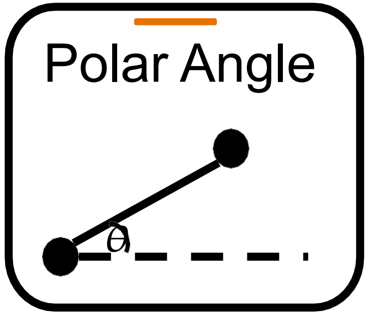
Graham's Algorithm

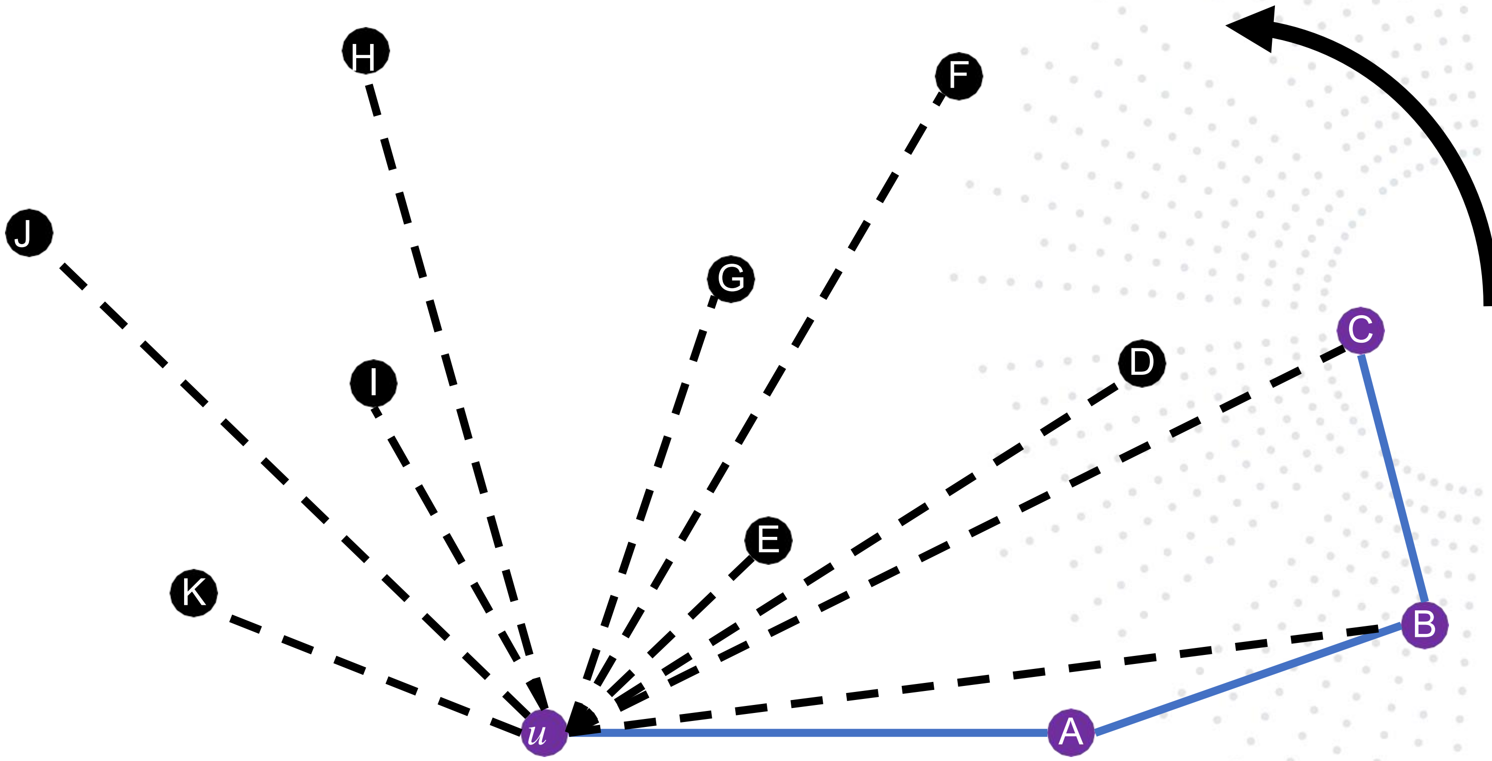


Scan the points in order of angle

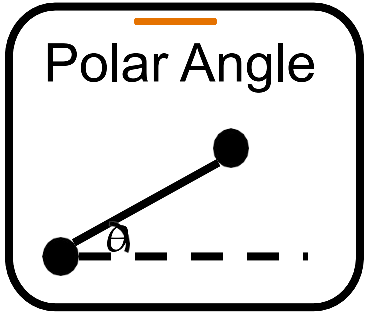


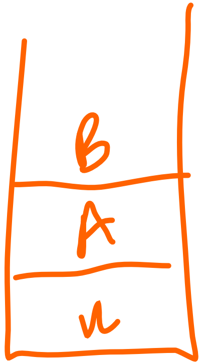
Graham's Algorithm

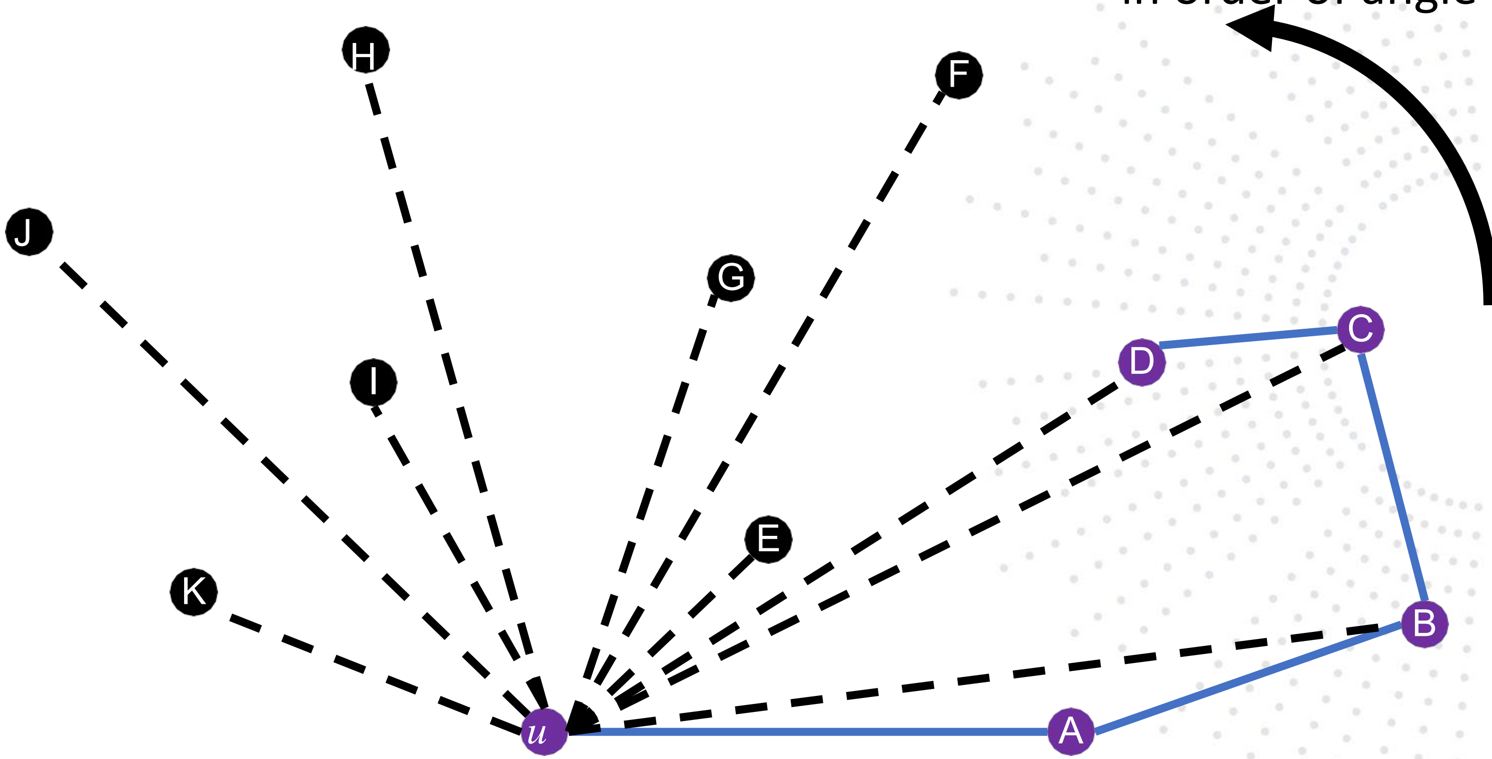




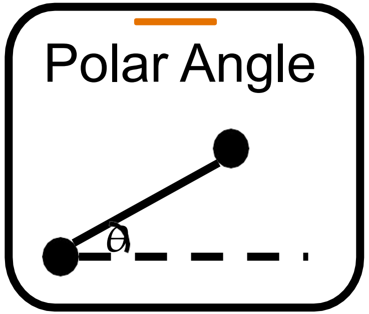
Graham's Algorithm



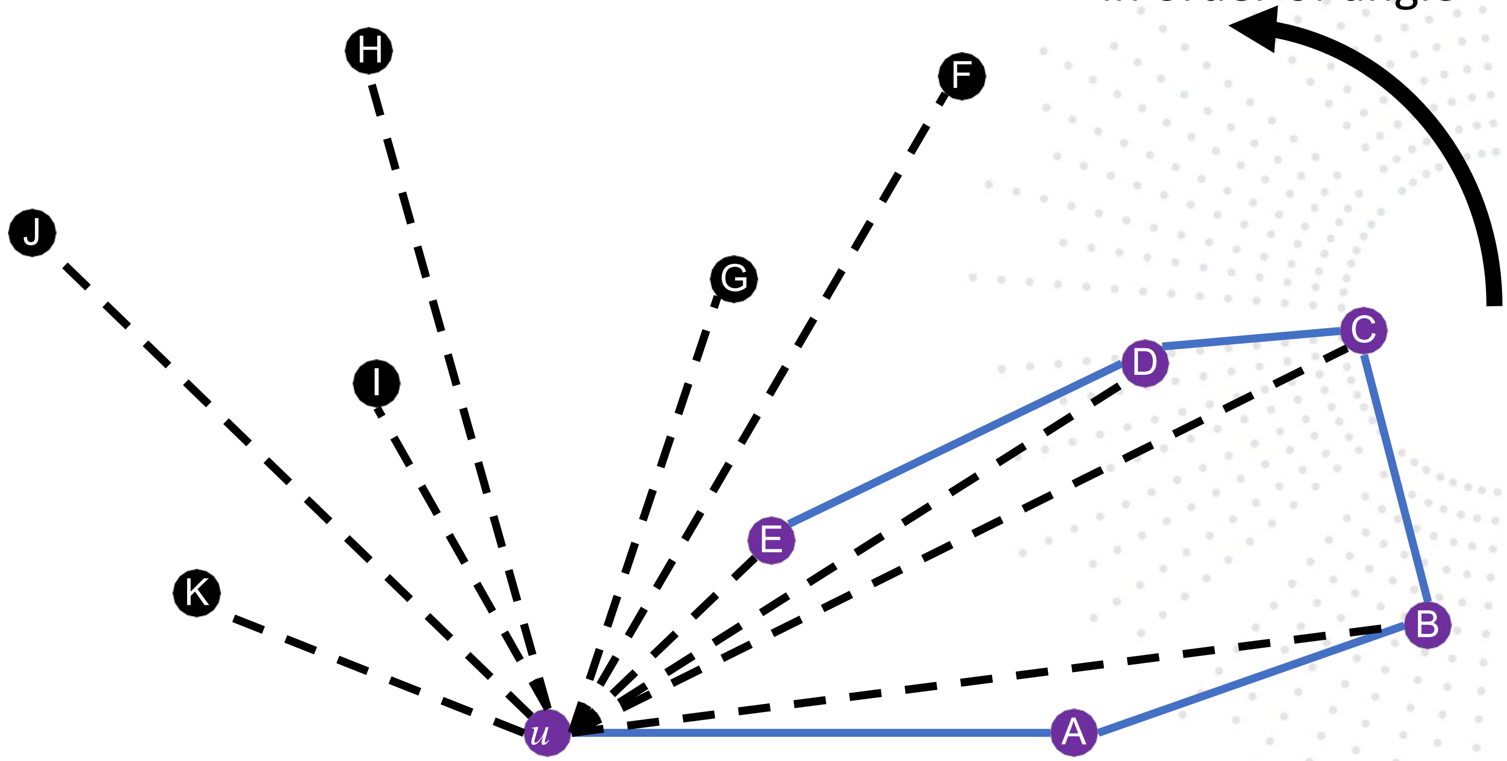




Graham's Algorithm



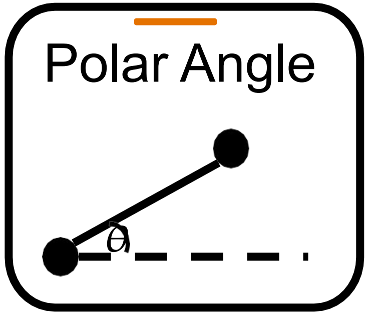
Scan the points in order of angle

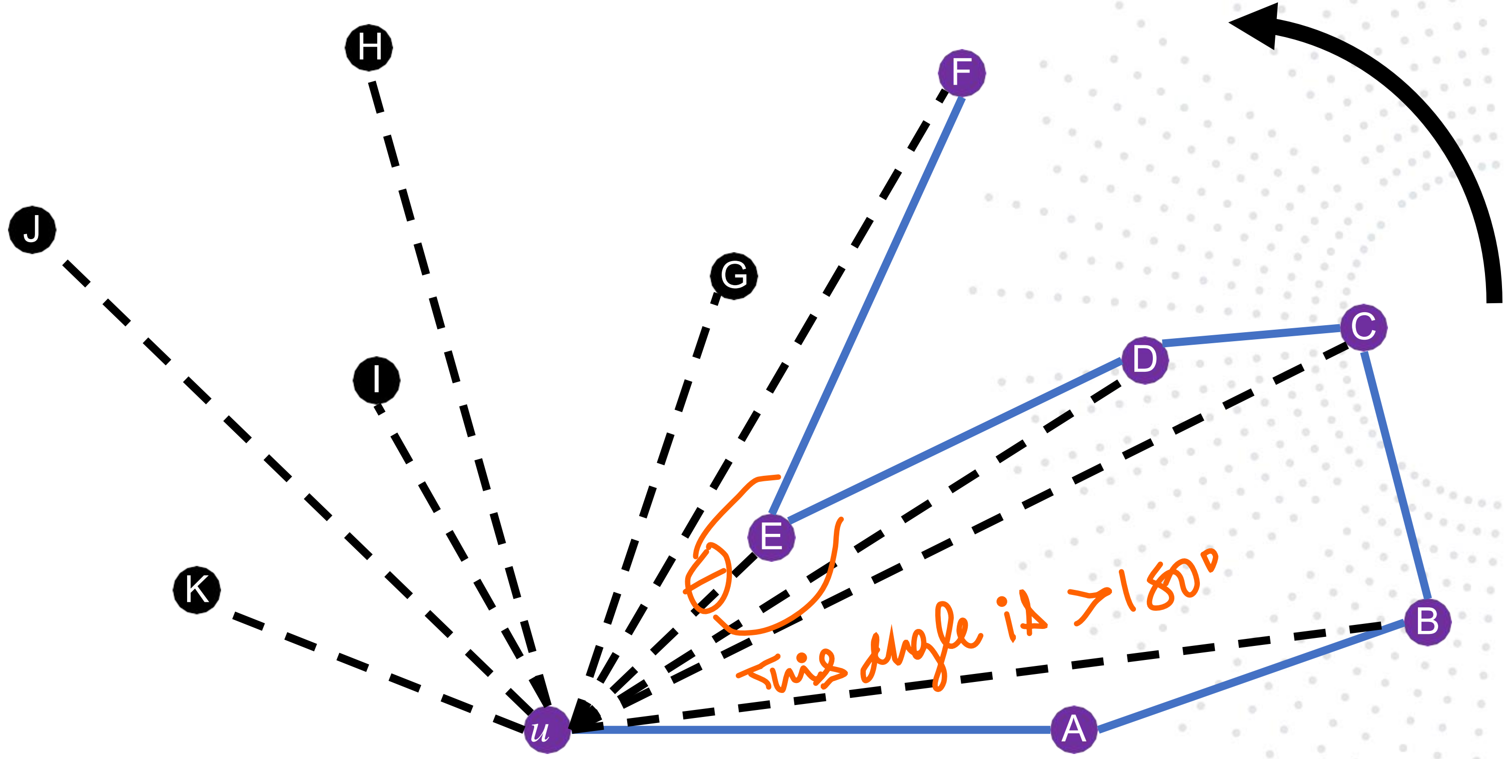


Graham's Algorithm

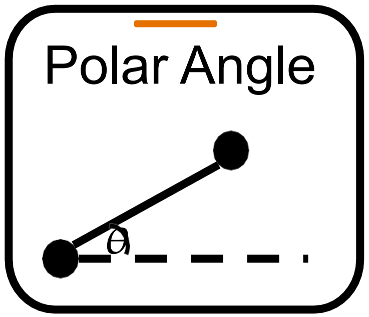
Not convex anymore!

Scan the points in order of angle

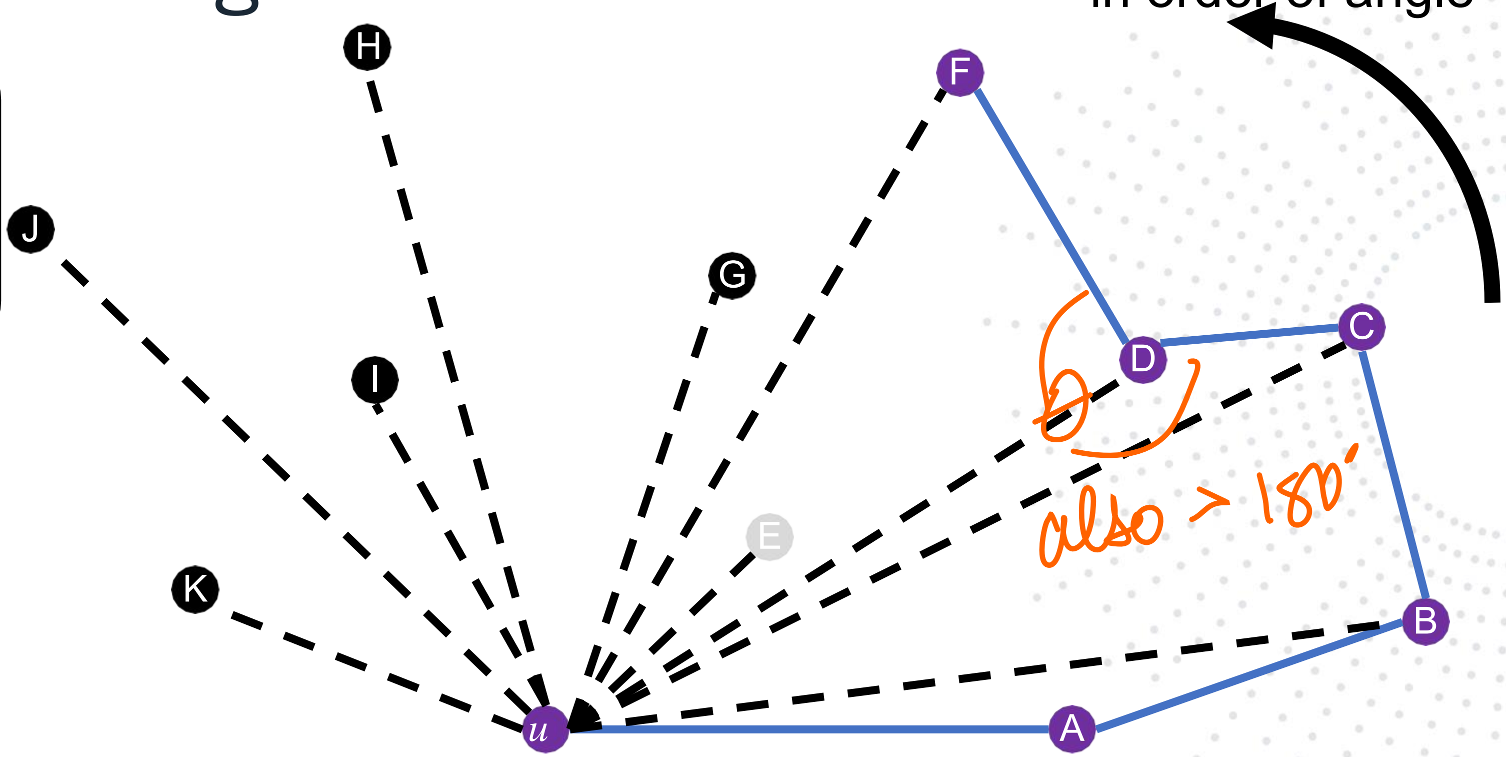




Graham's Algorithm

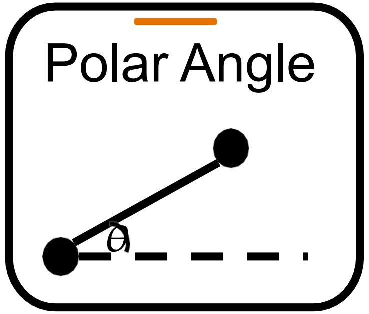


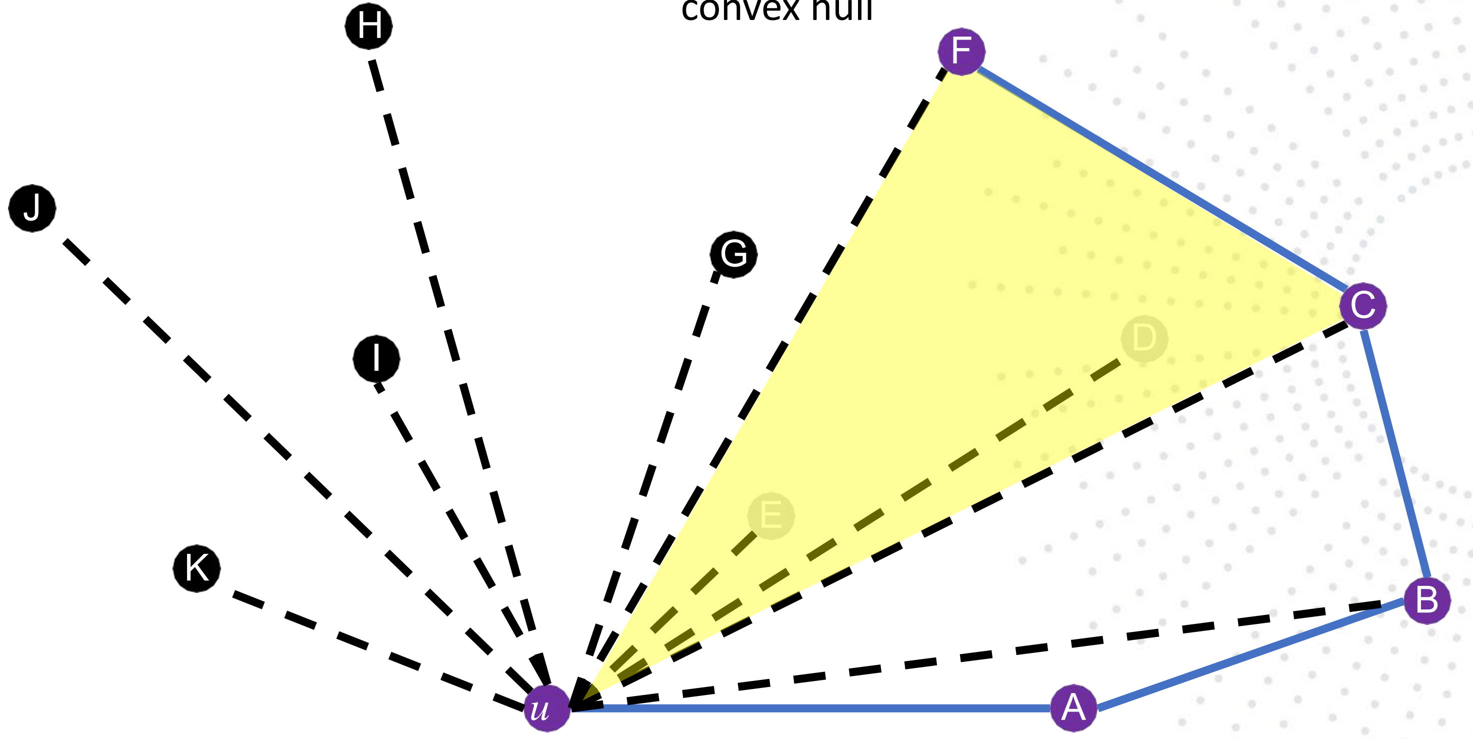
Scan the points in order of angle



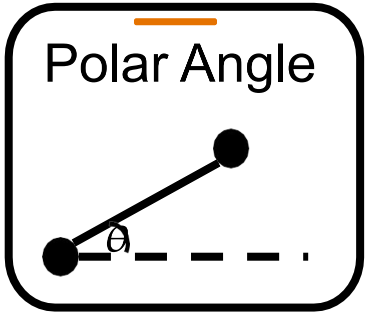
Graham's Algorithm

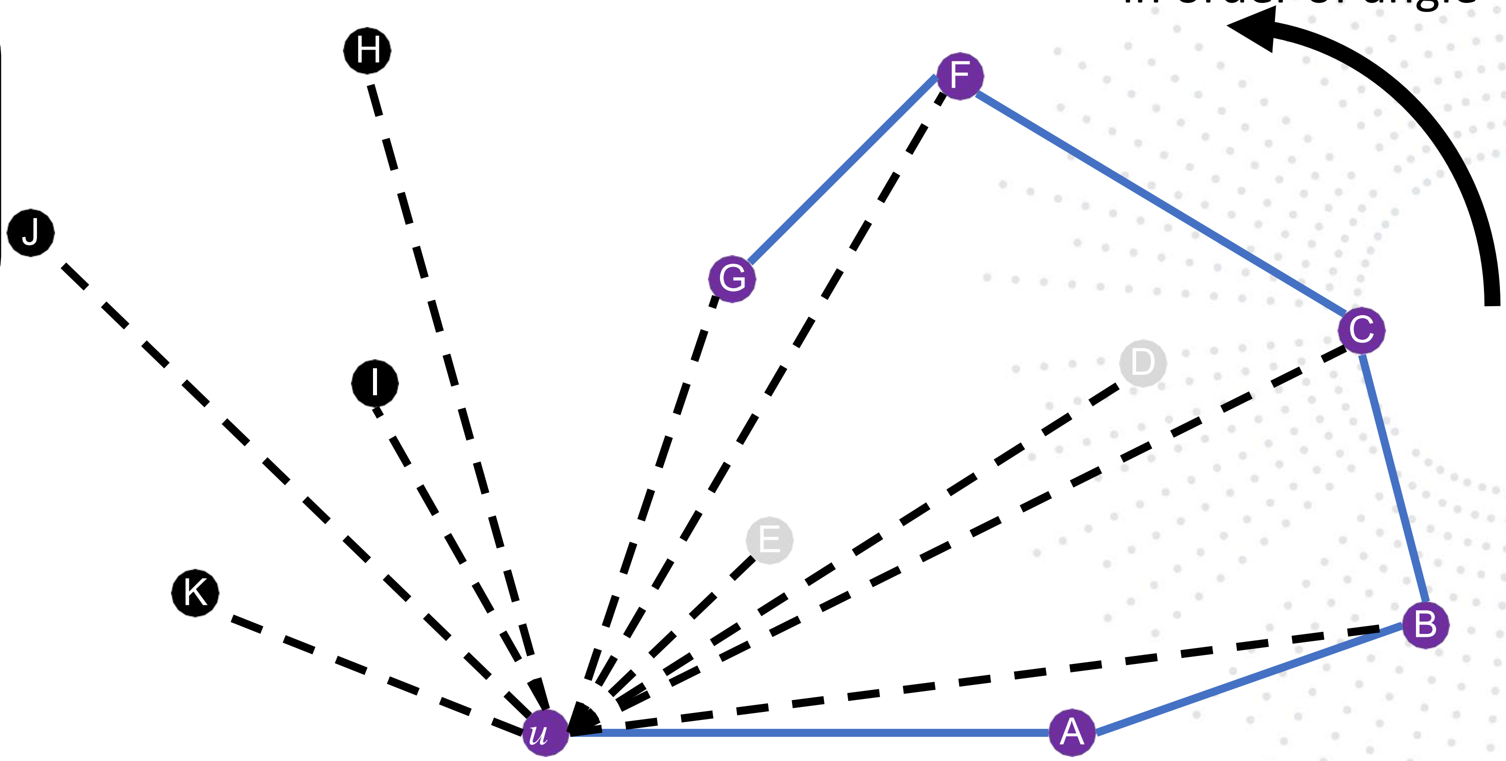
Observe: since points are sorted by angle, backtracking will never remove points from the convex hull





Graham's Algorithm

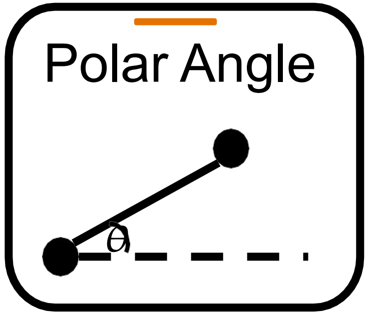


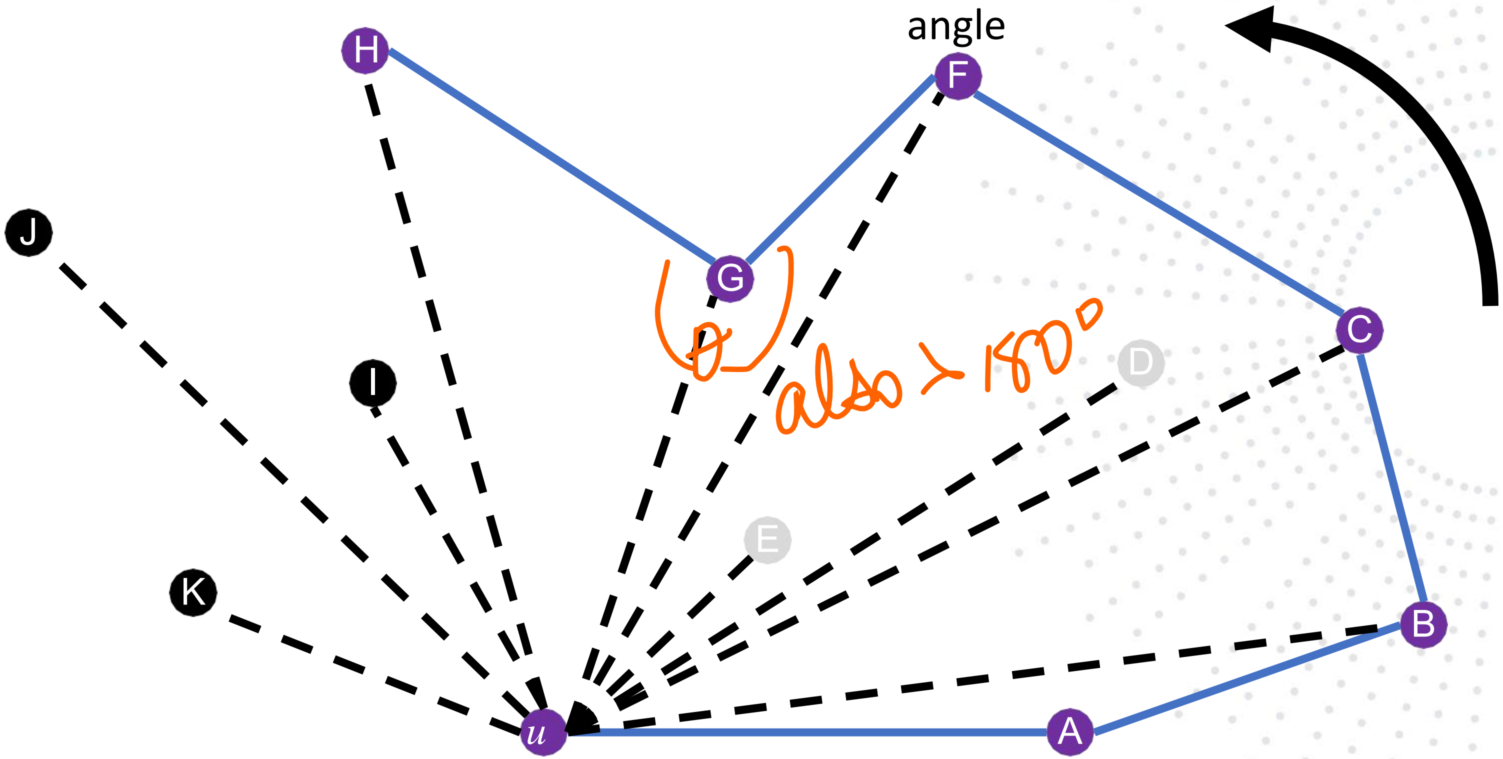


Graham's Algorithm

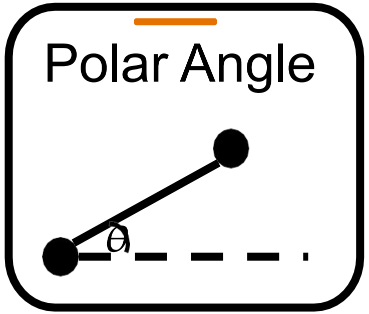
Not convex anymore!

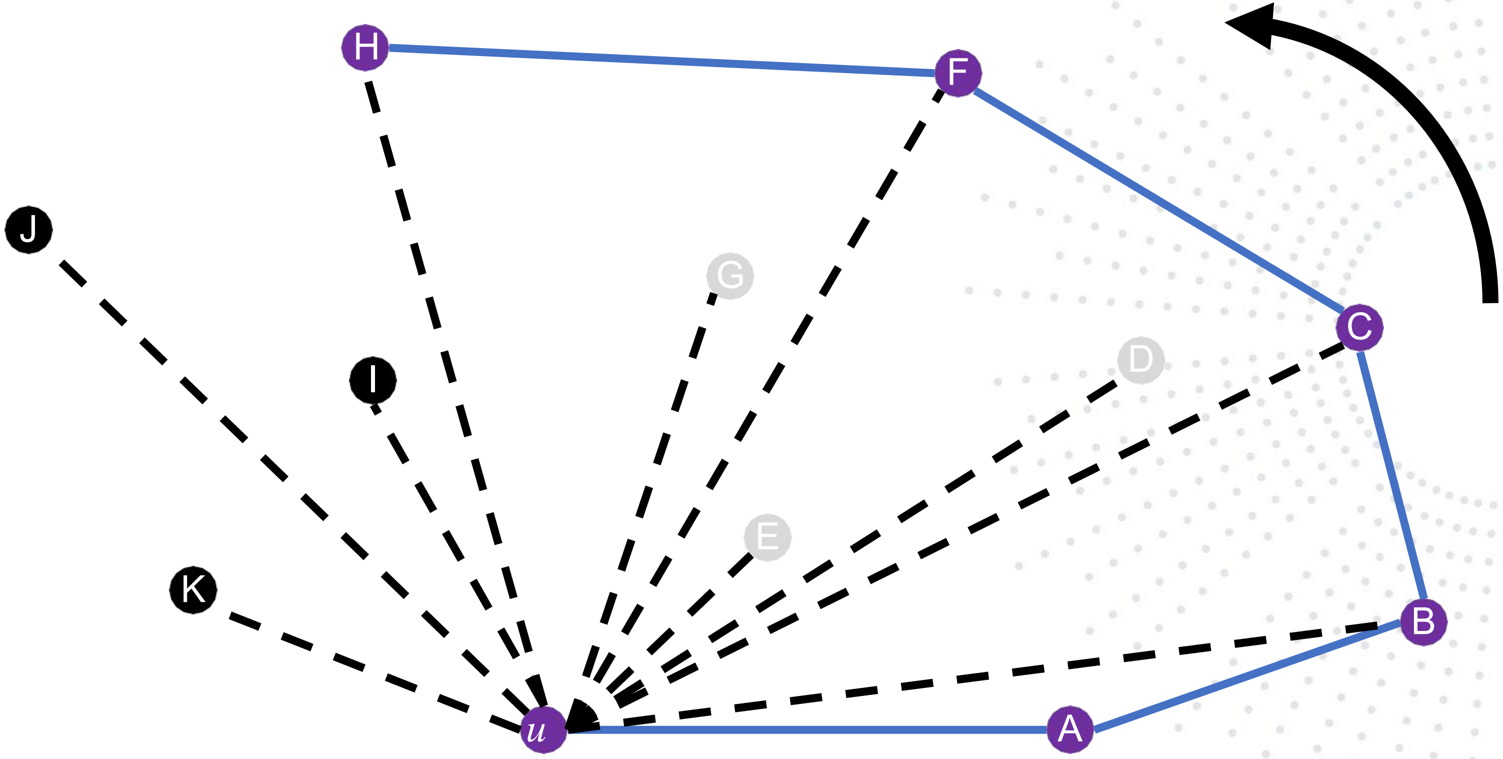
Scan the points in order of angle





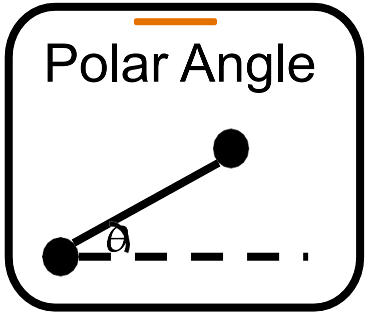
Graham's Algorithm

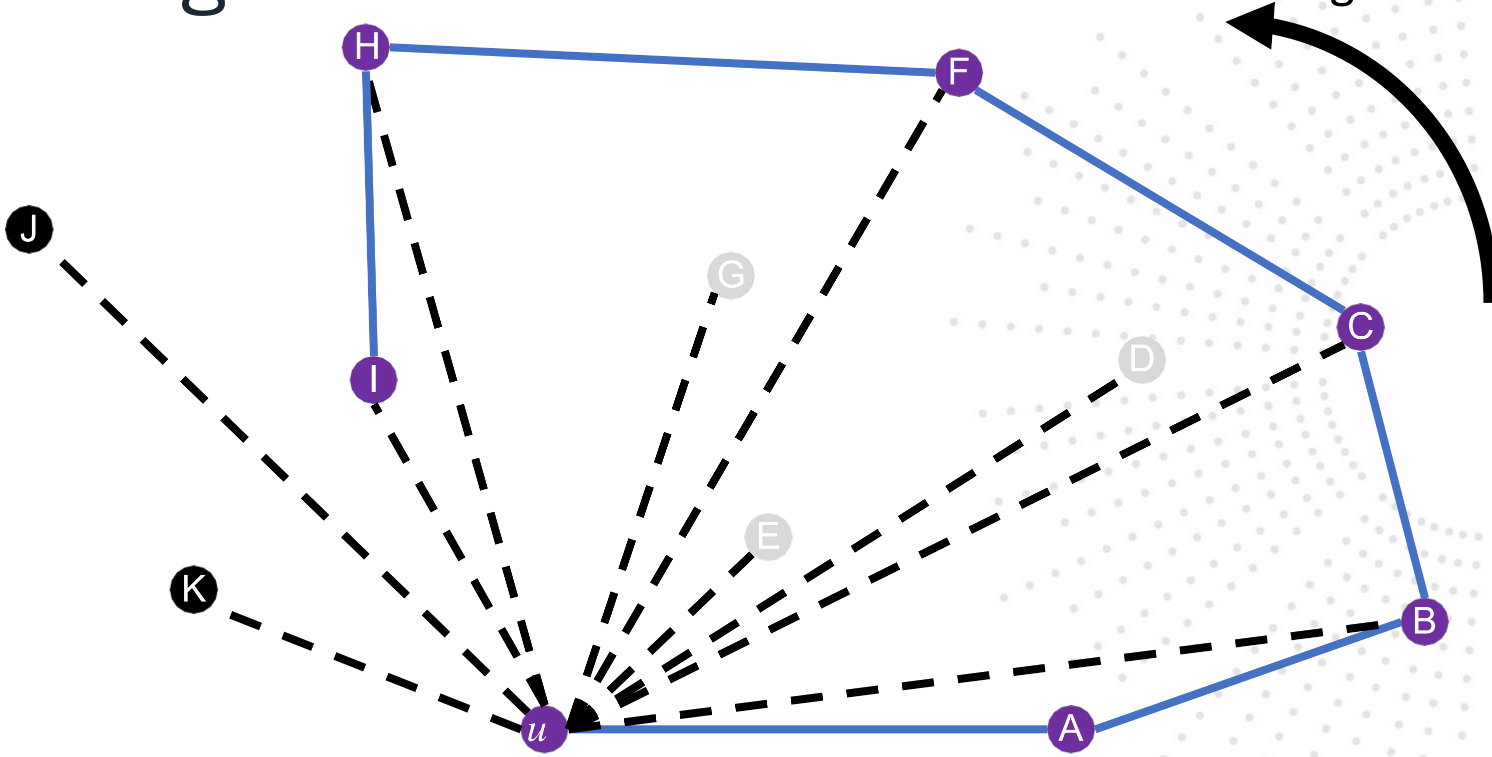




Graham's Algorithm

Scan the points in order of angle

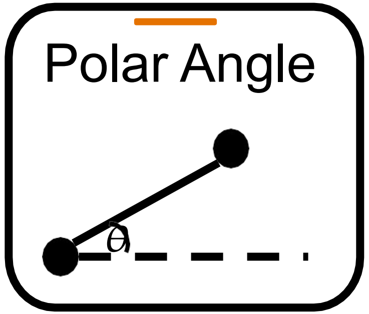


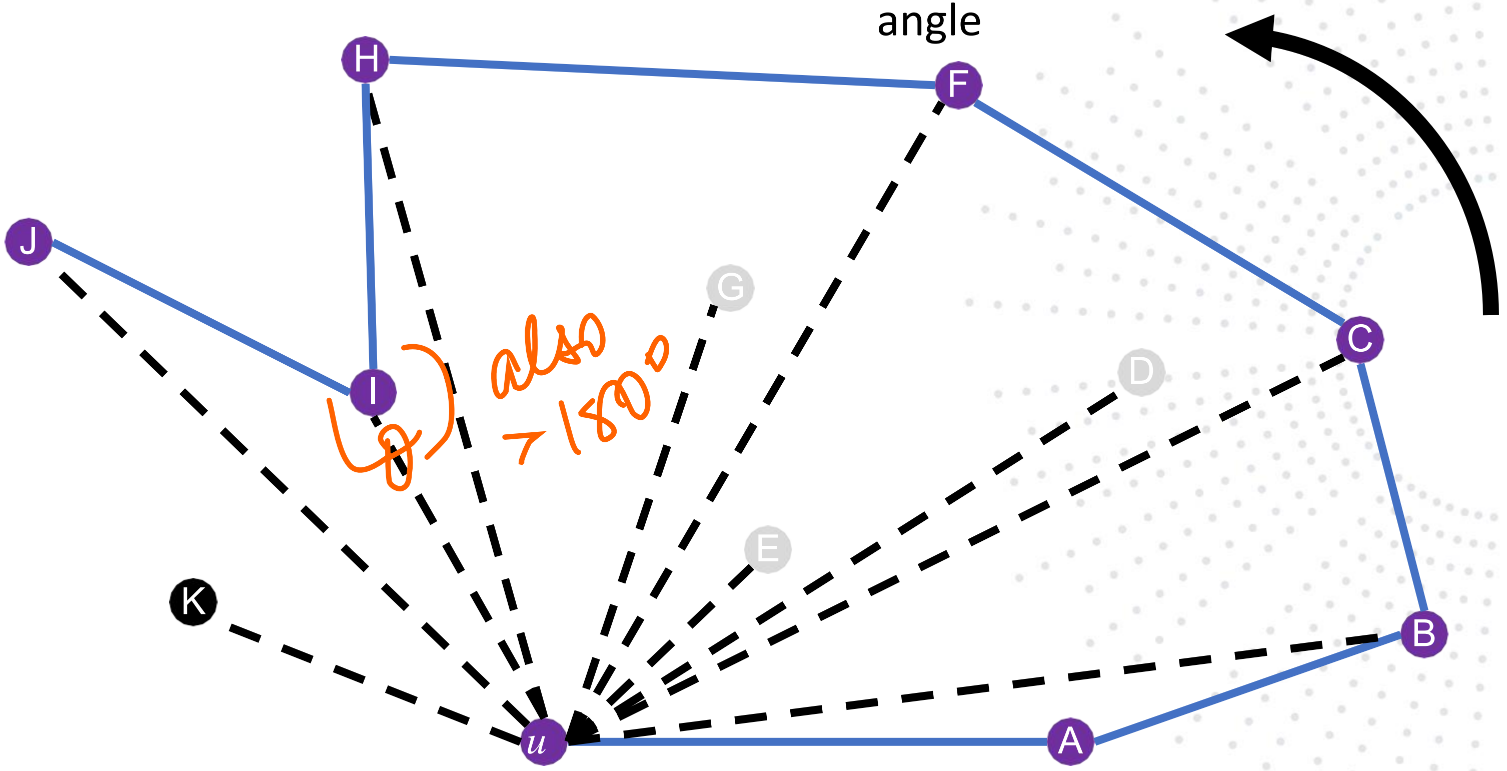


Graham's Algorithm

Not convex anymore!

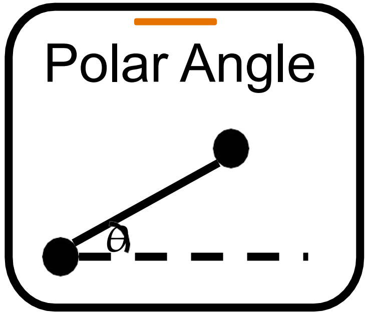
Scan the points in order of angle



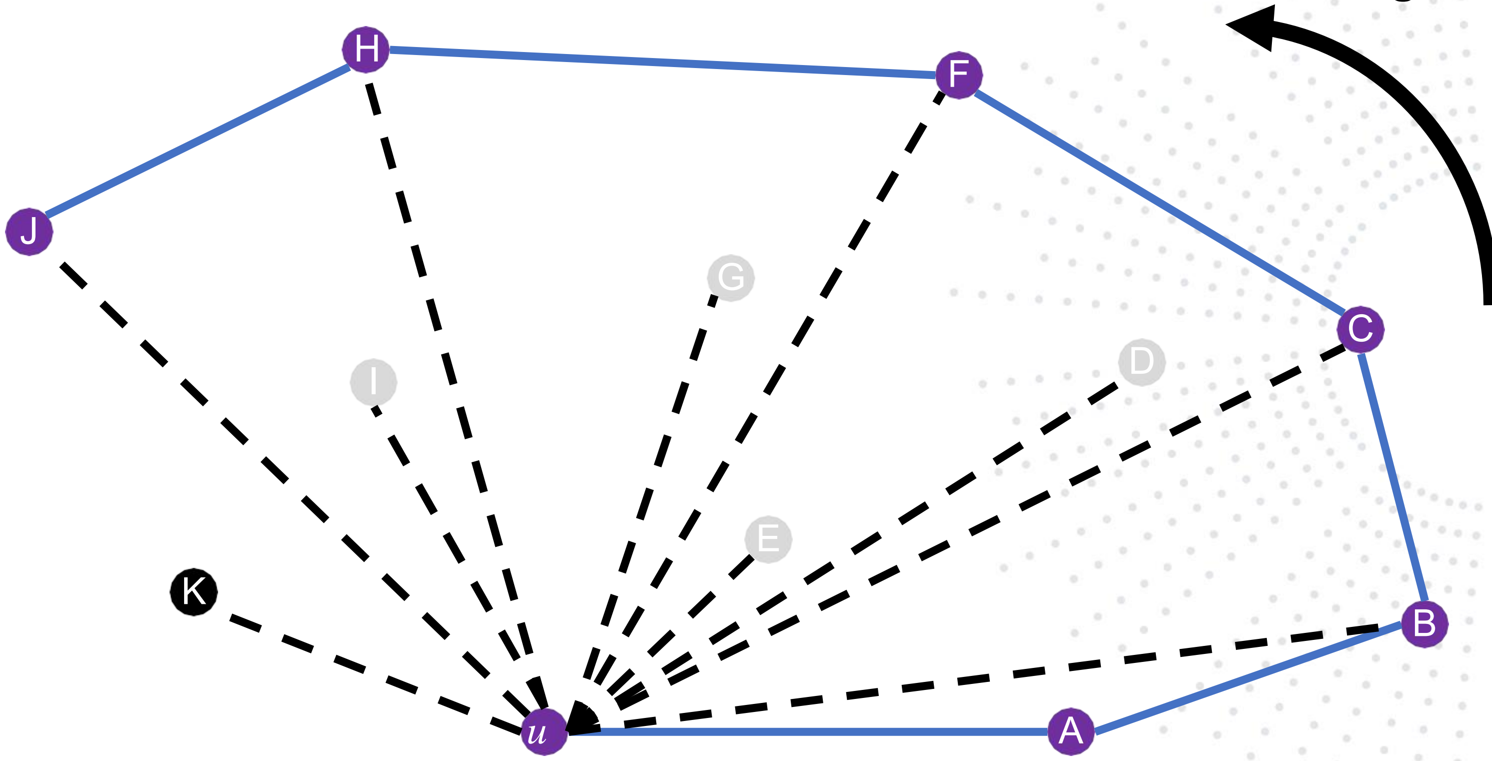


Graham's Algorithm

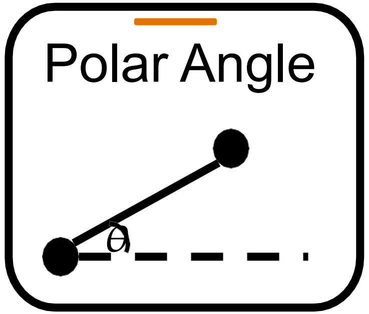
Polar Angle

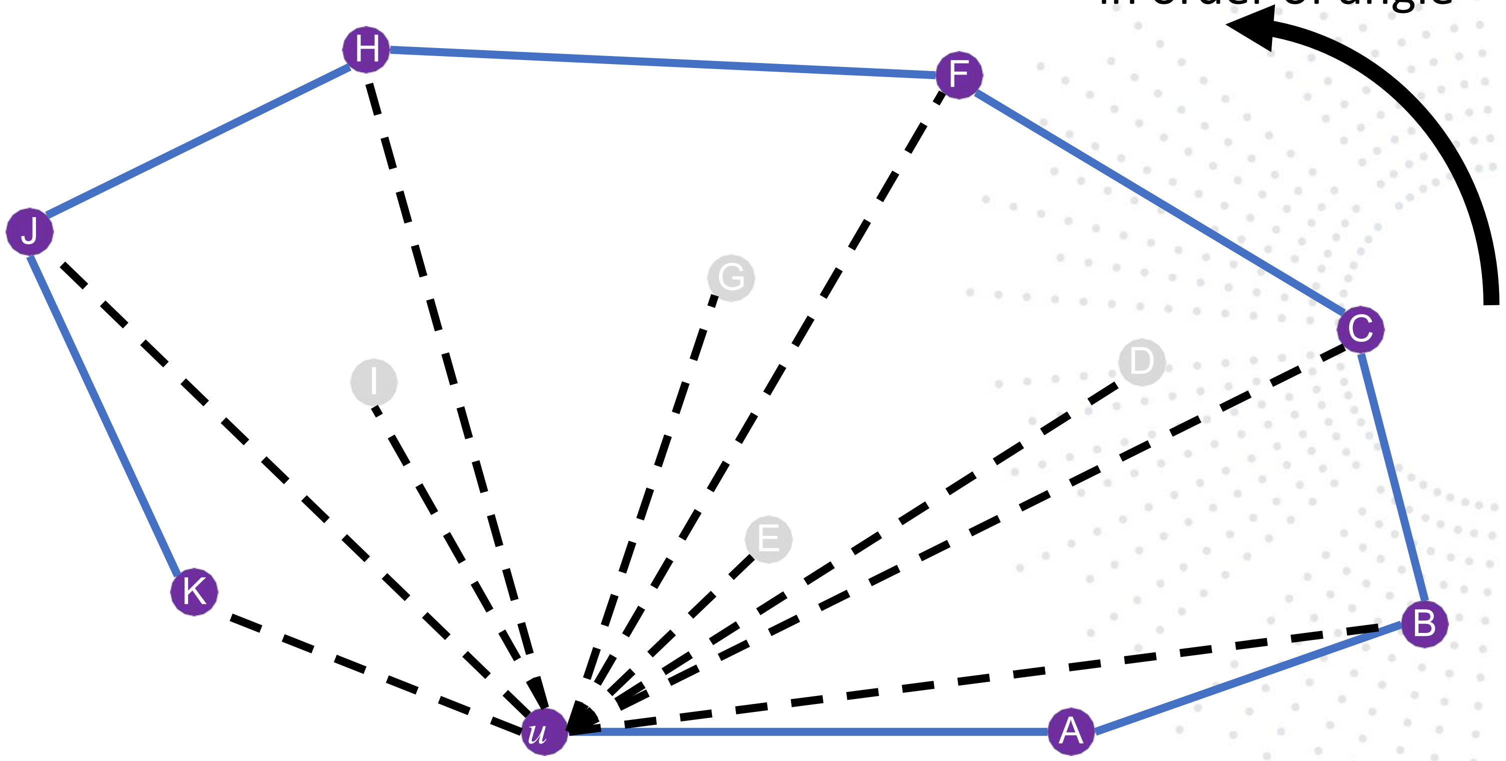


Scan the points in order of angle

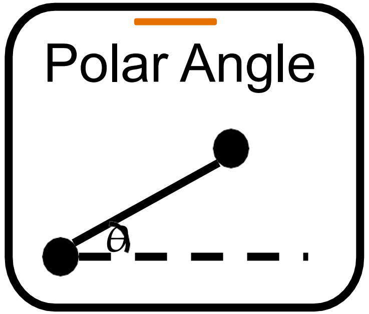


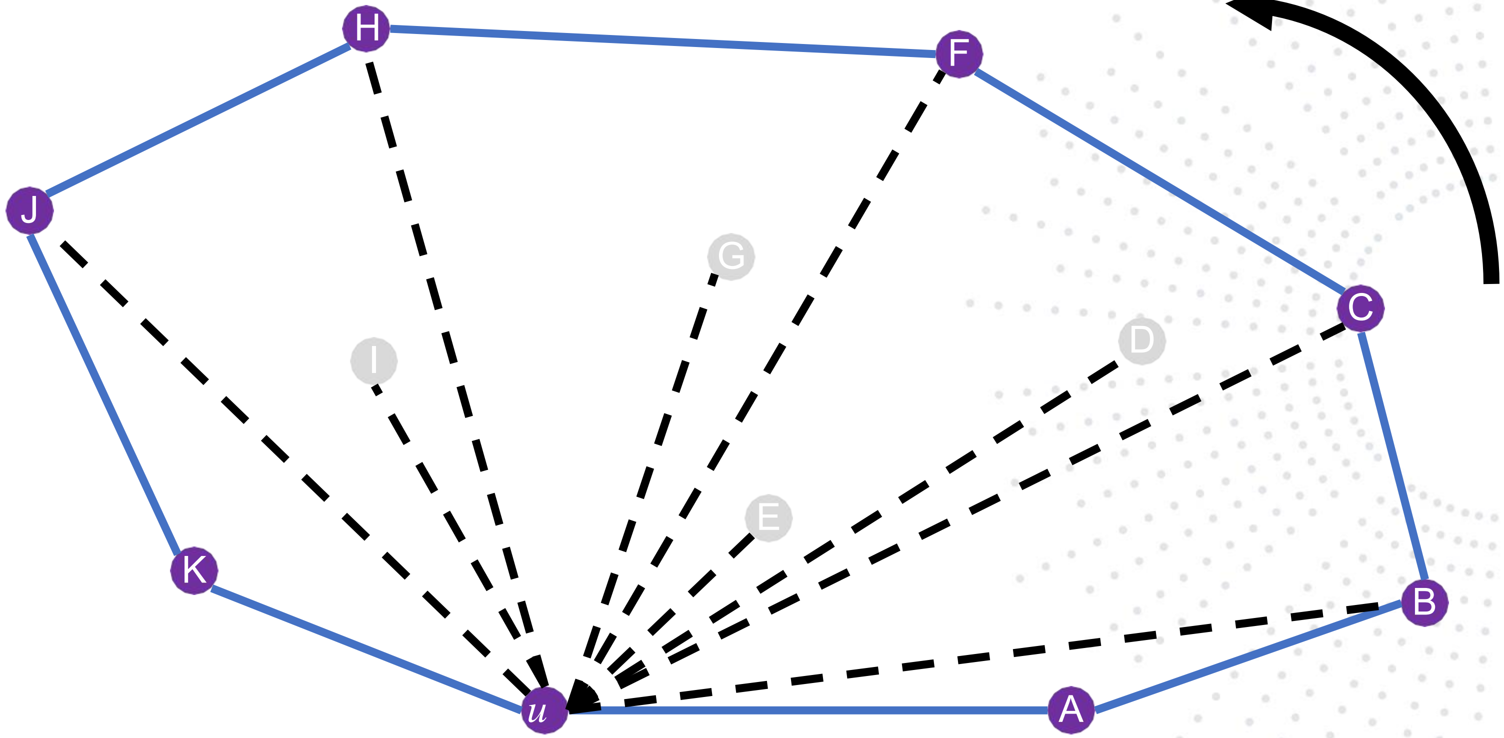
Graham's Algorithm



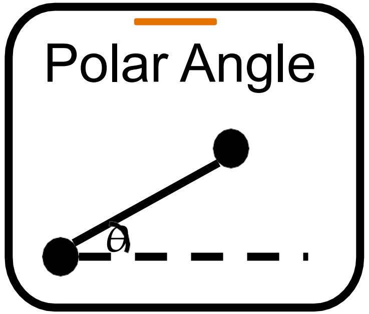


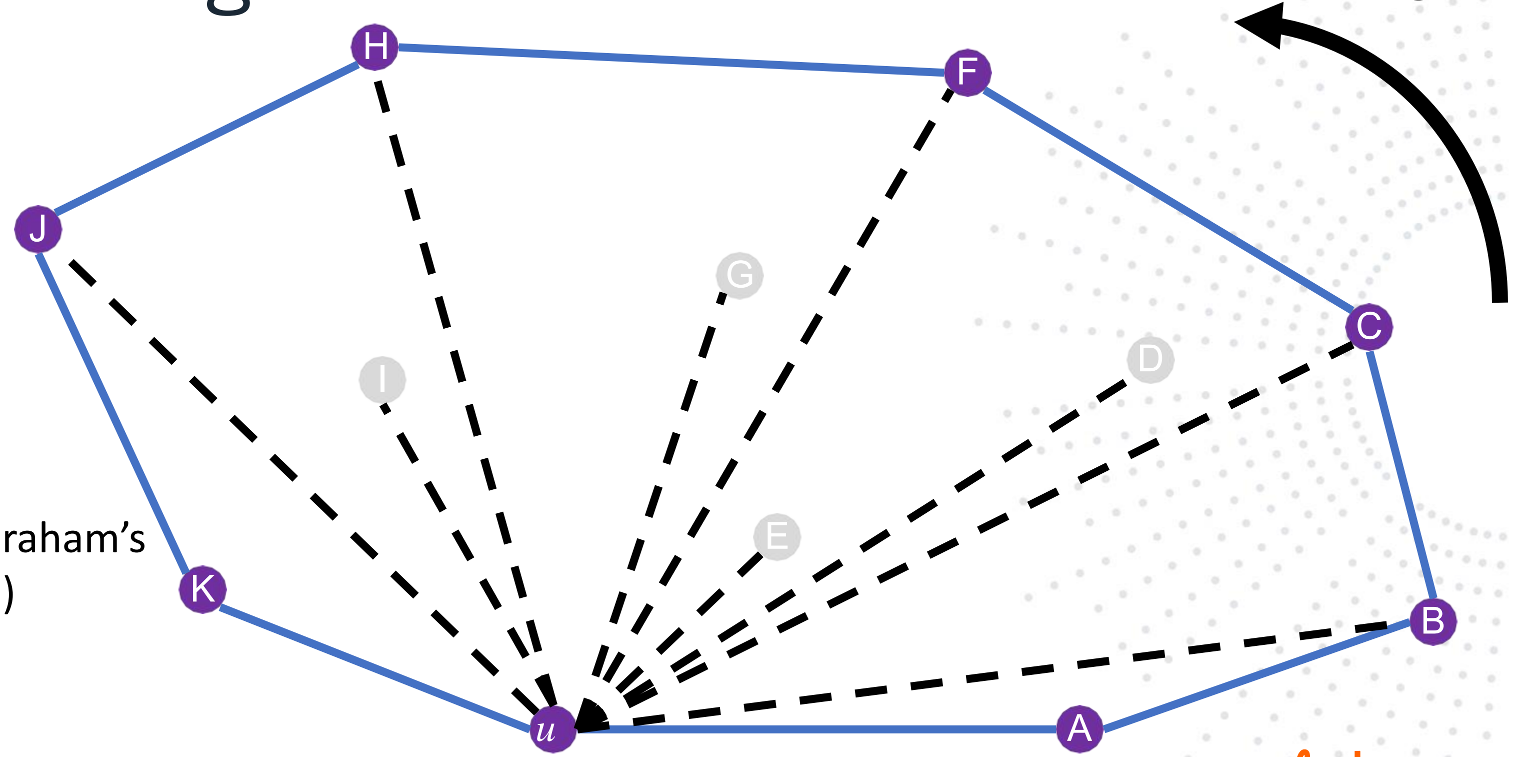
Graham's Algorithm





Graham's Algorithm





Scan the points in order of angle

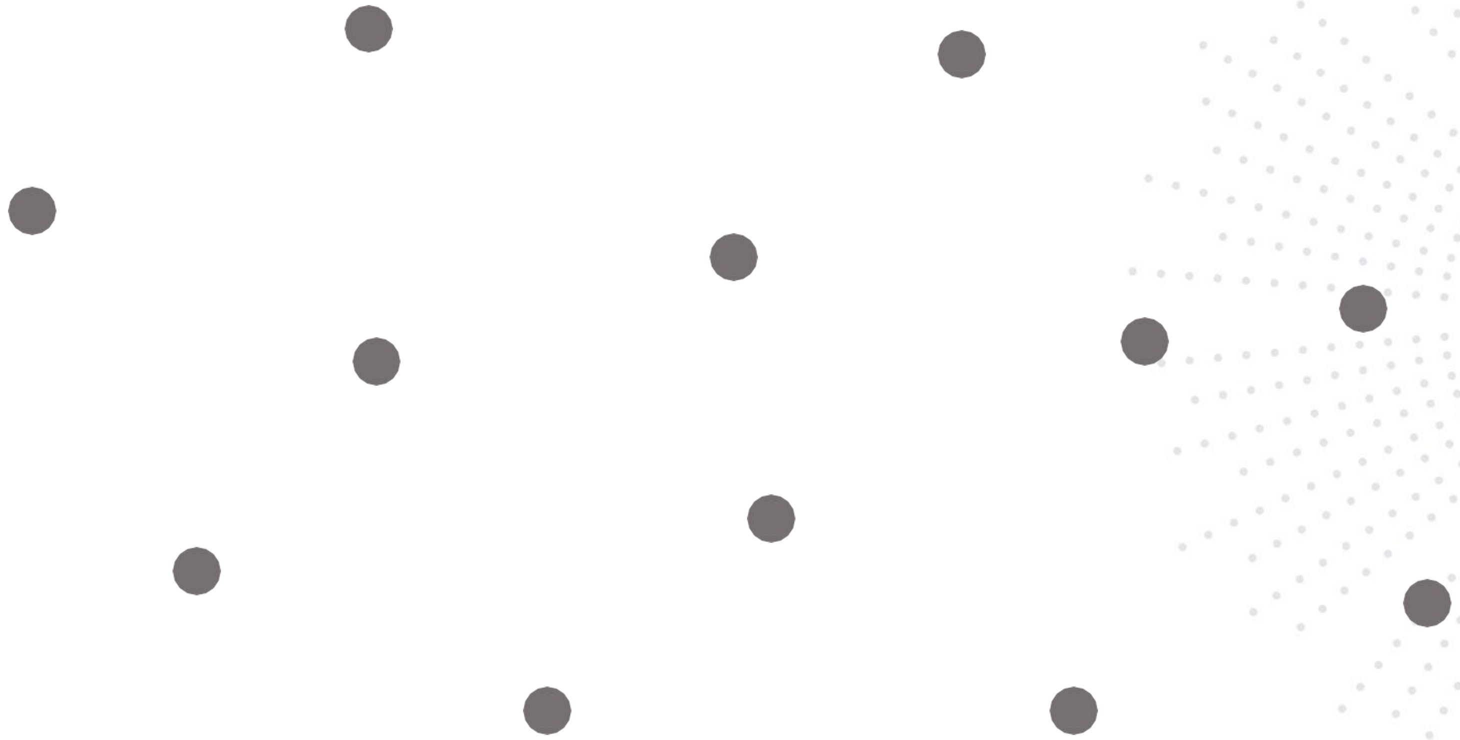
Time Complexity of Graham's algorithm is O(n log n)

our convex hole is complete

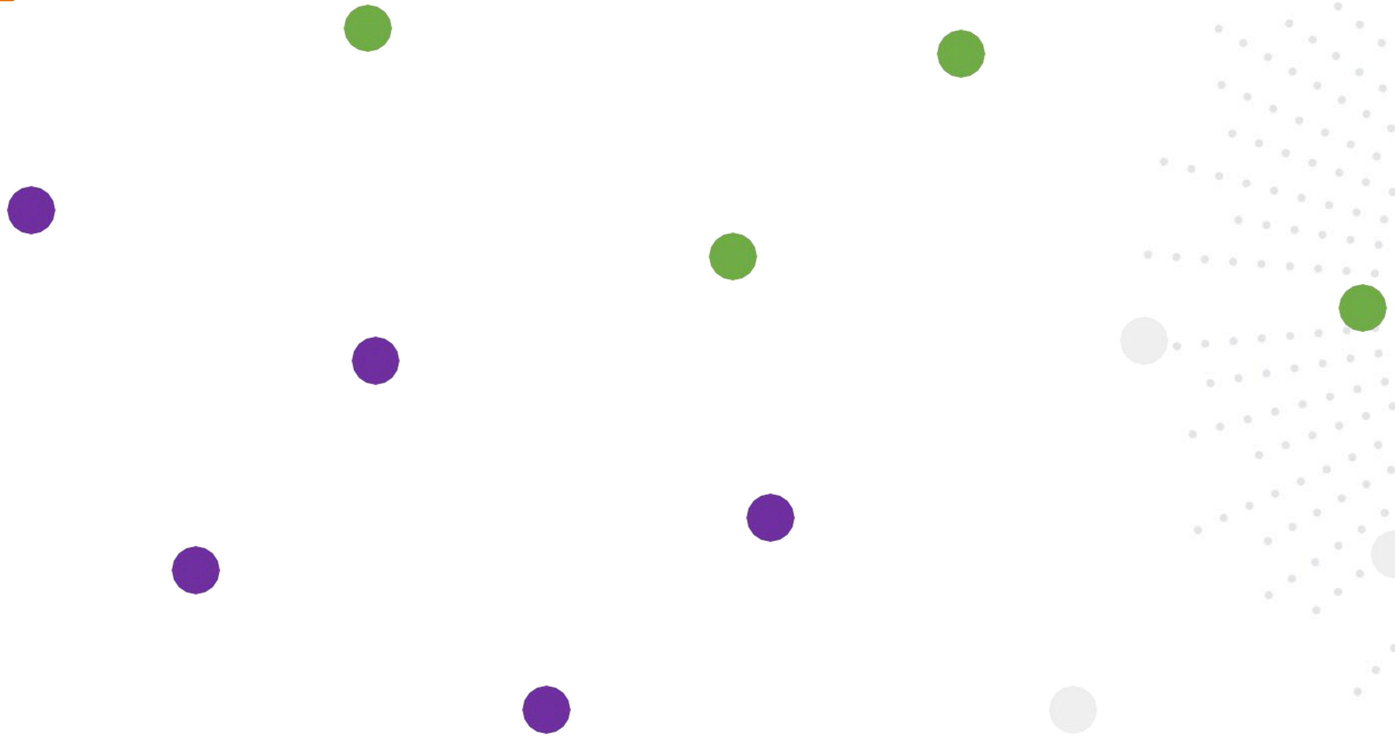
Chan's Algorithm $\rightarrow$ uses a divide & conquer approach

- Divide the set of points into k subsets
- For each of the k subsets, use Graham's Scan algorithm to compute the convex hull of each subset. This will give you k convex hulls
- Once you have the k convex hulls, you need to merge them into a single convex hull. This is where Jarvis's algorithm(Gift Wrapping).
    - Start by selecting a point on the convex hull
    - Find the point on the boundary of the hull that is farthest left relative to the current point. Move to that point
    - Repeat this process until you return to the starting point

Chan’s Algorithm

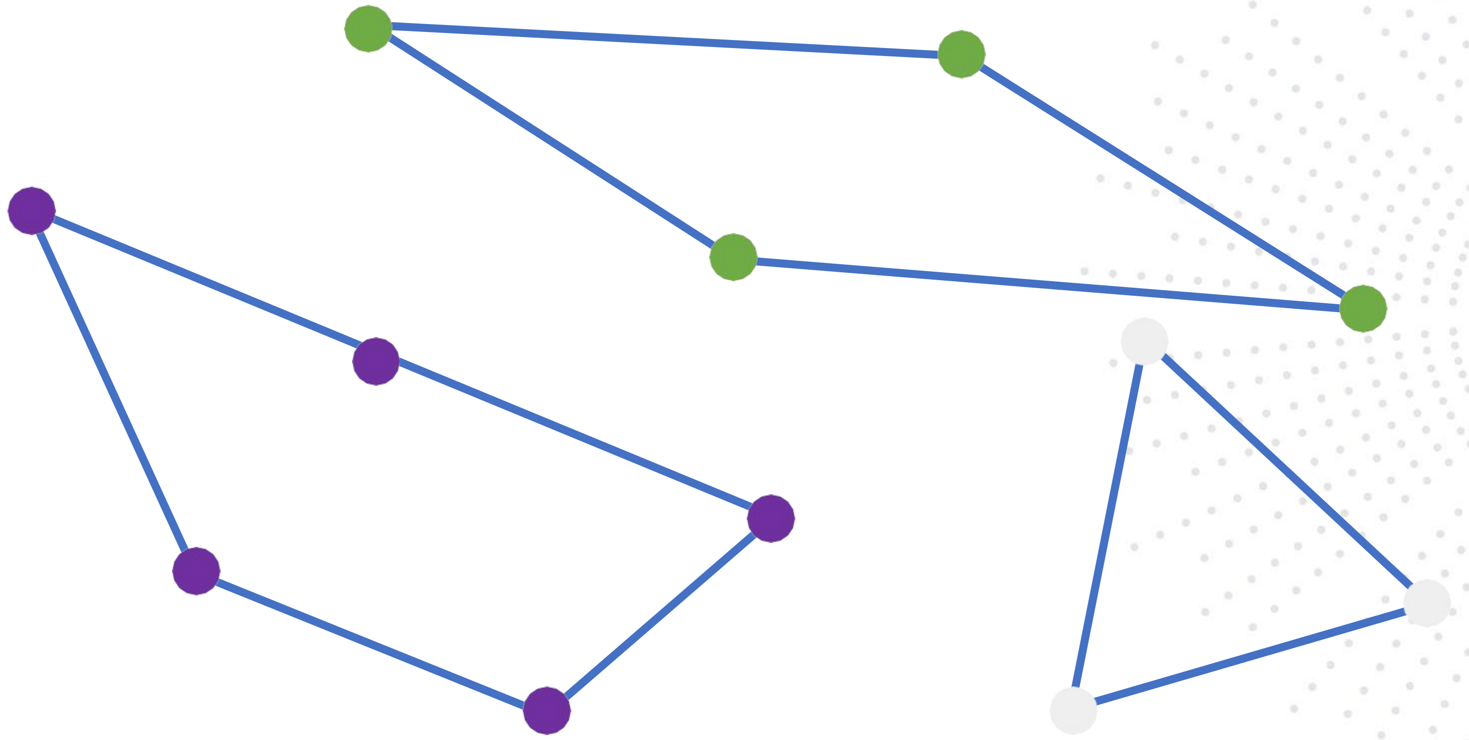


Chan's Algorithm



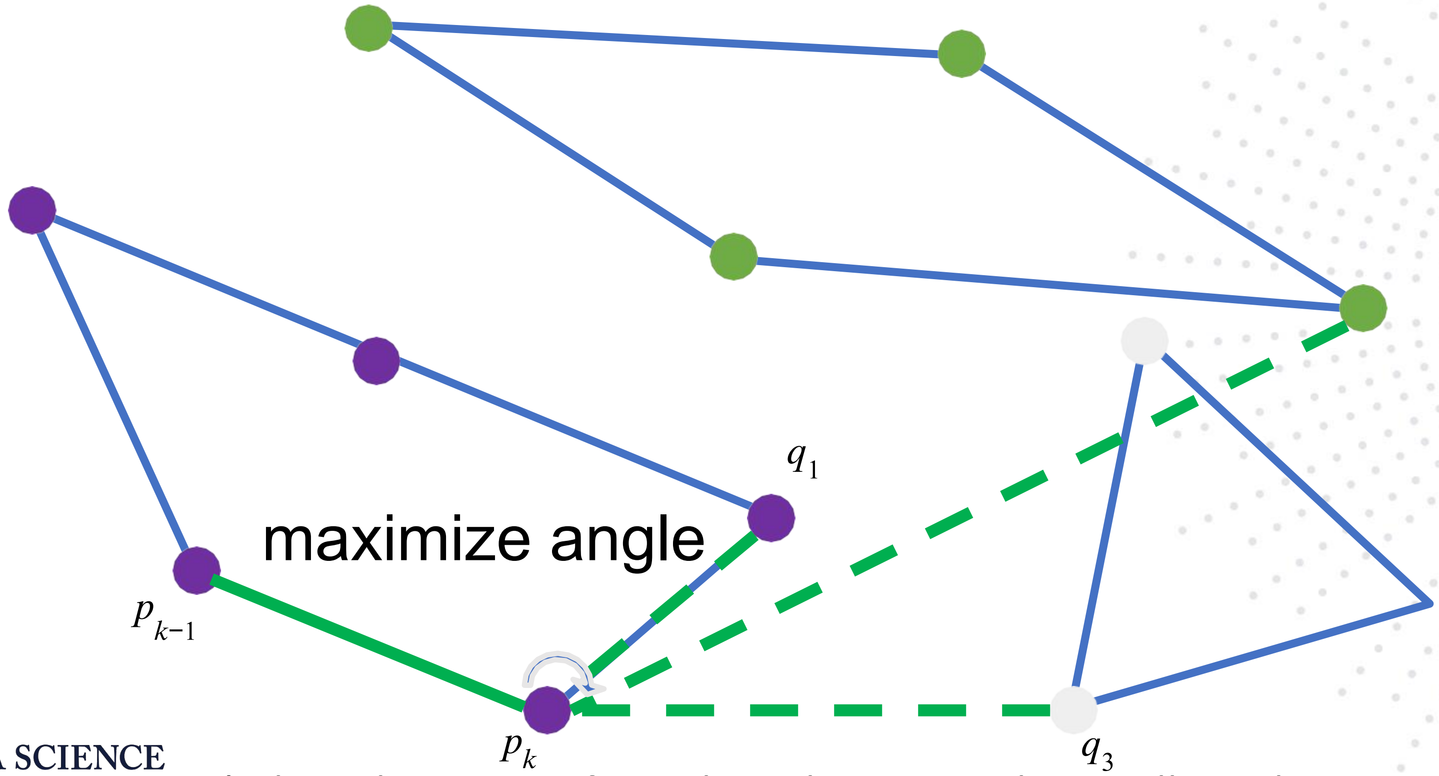
Divide into smaller subsets

Chan’s Algorithm



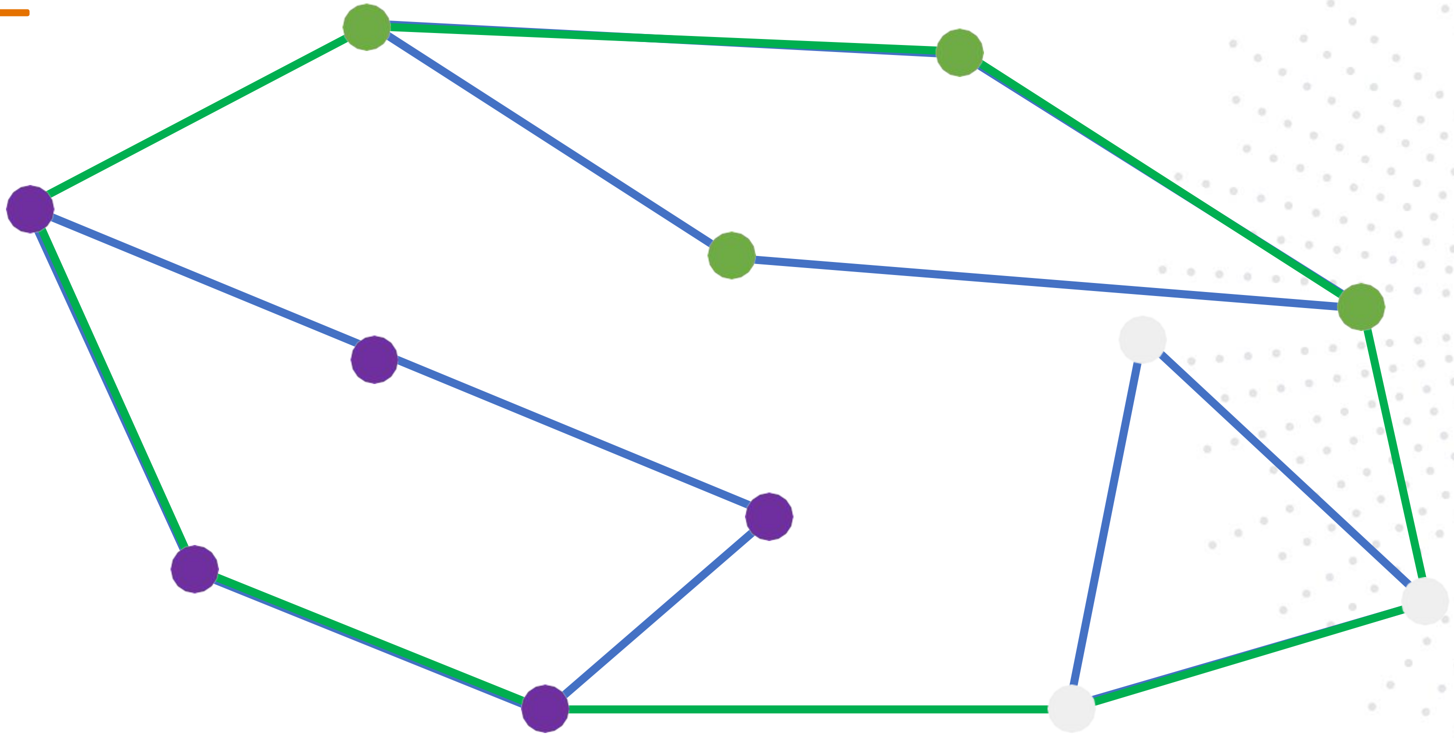
UVA DATA SCIENCE Use Graham’s Algorithm to **conquer** the smaller subsets

Chan’s Algorithm

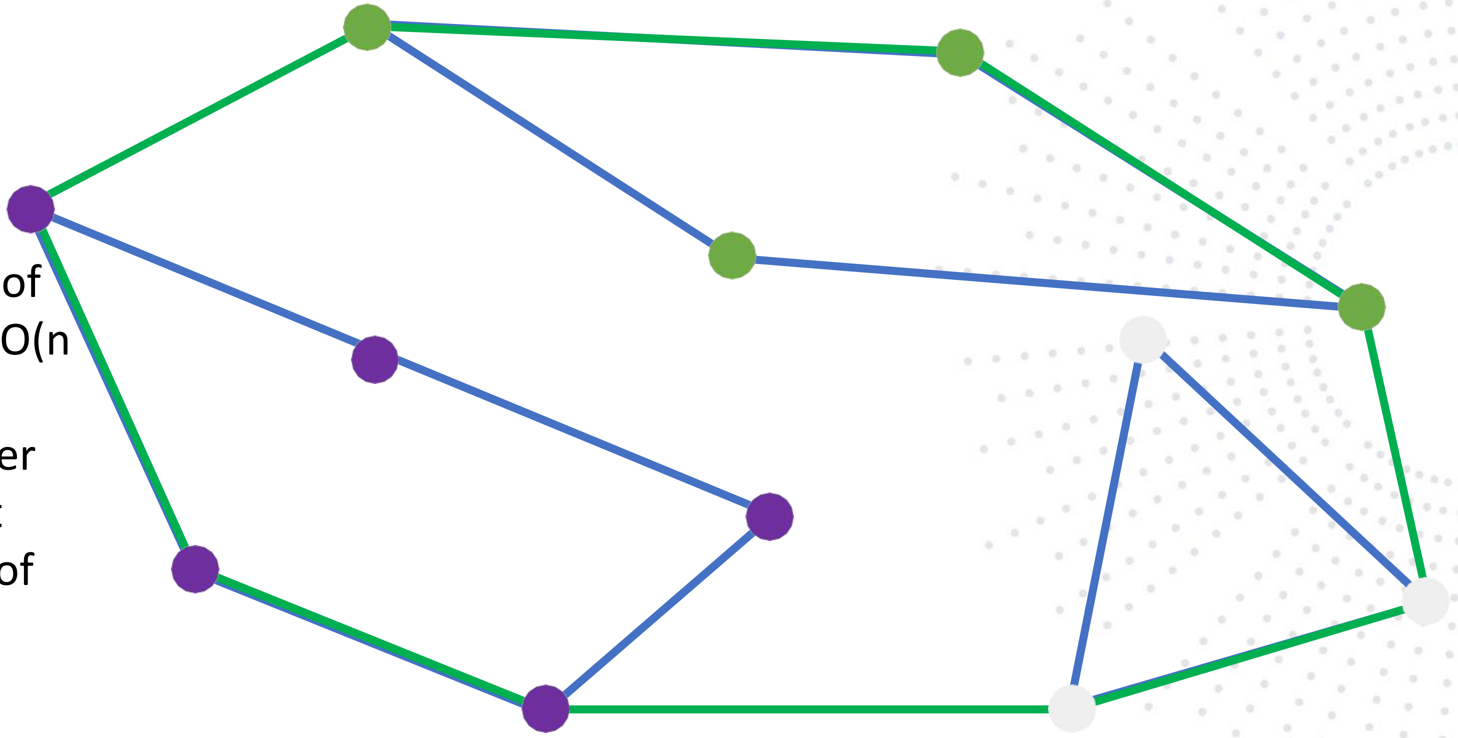


Use Jarvis’ Algorithm to **combine** the solutions to the smaller subsets

Chan’s Algorithm

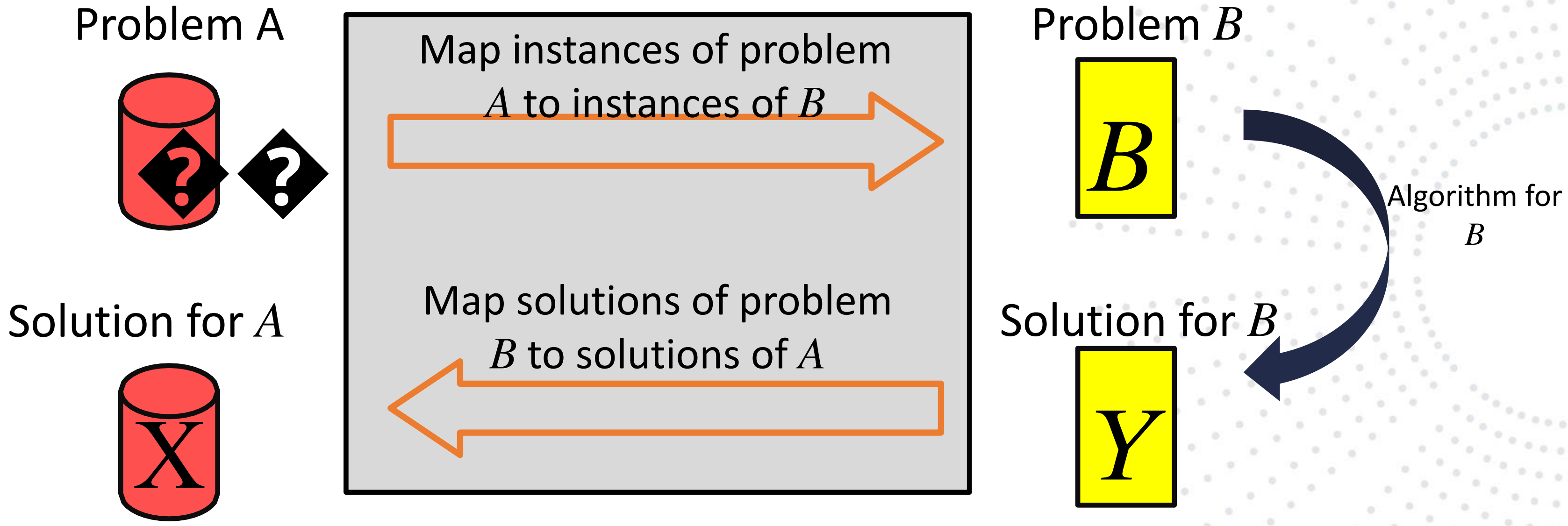


Chan’s Algorithm



The time complexity of Chan’s Algorithm is O(n log h)
where n is the number of points in the input set. h is the number of points on the convex hull

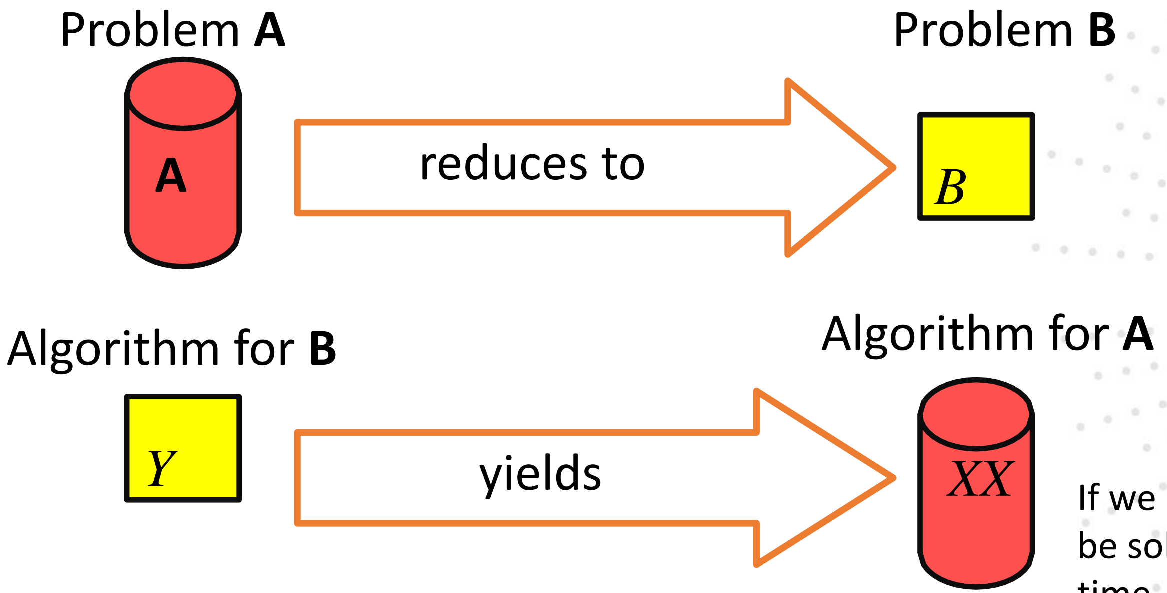
Reduction


Problem A
?
?
Map instances of problem A to instances of B
Problem B
B
Algorithm for B
Solution for A
X
Map solutions of problem B to solutions of A
Solution for B
Y

Reduction

$A \leq B$: there is a reduction from A to B

Sorting to Convex Hull Reduction

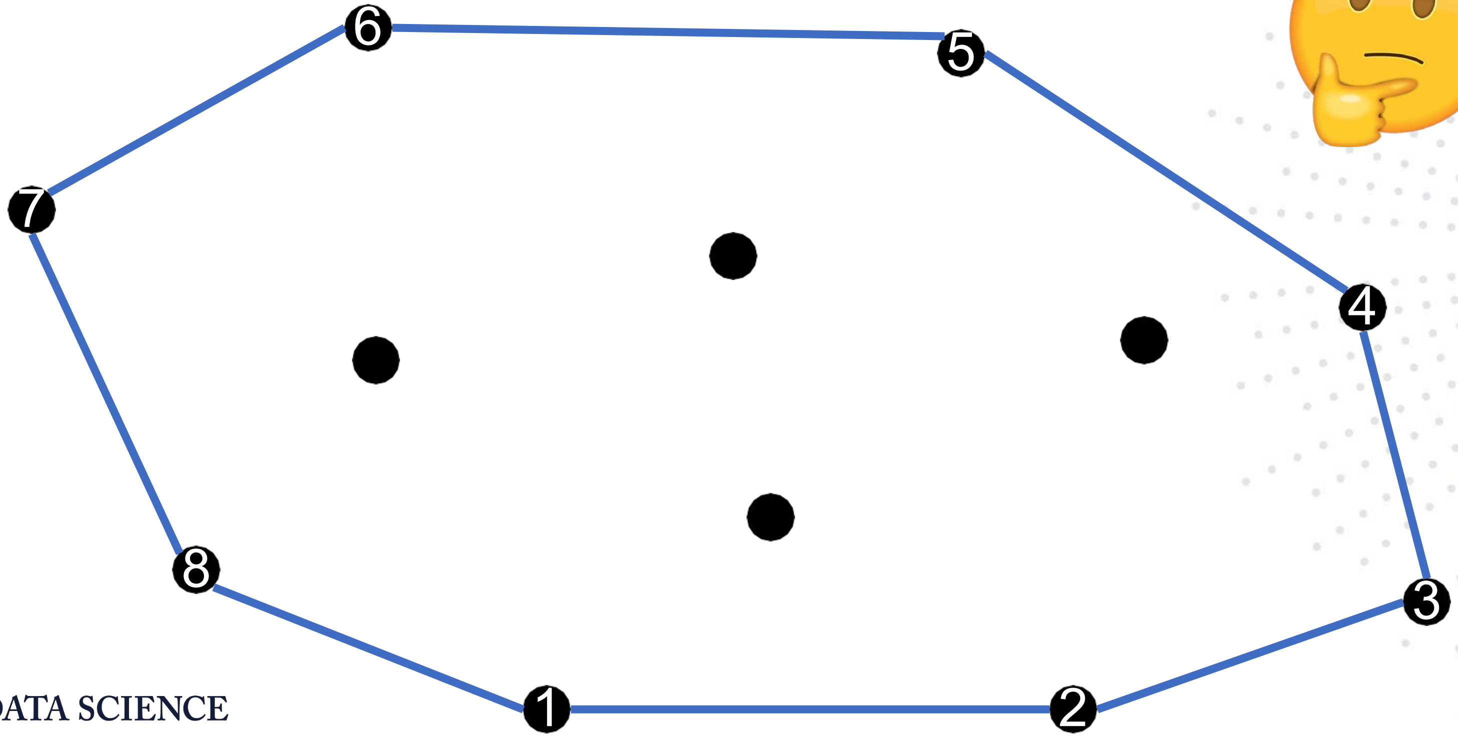


If we know that A cannot be solved in polynomial time, then B cannot be solved in polynomial time

Implication: A is no more difficult than B (denoted $A \leq B$)

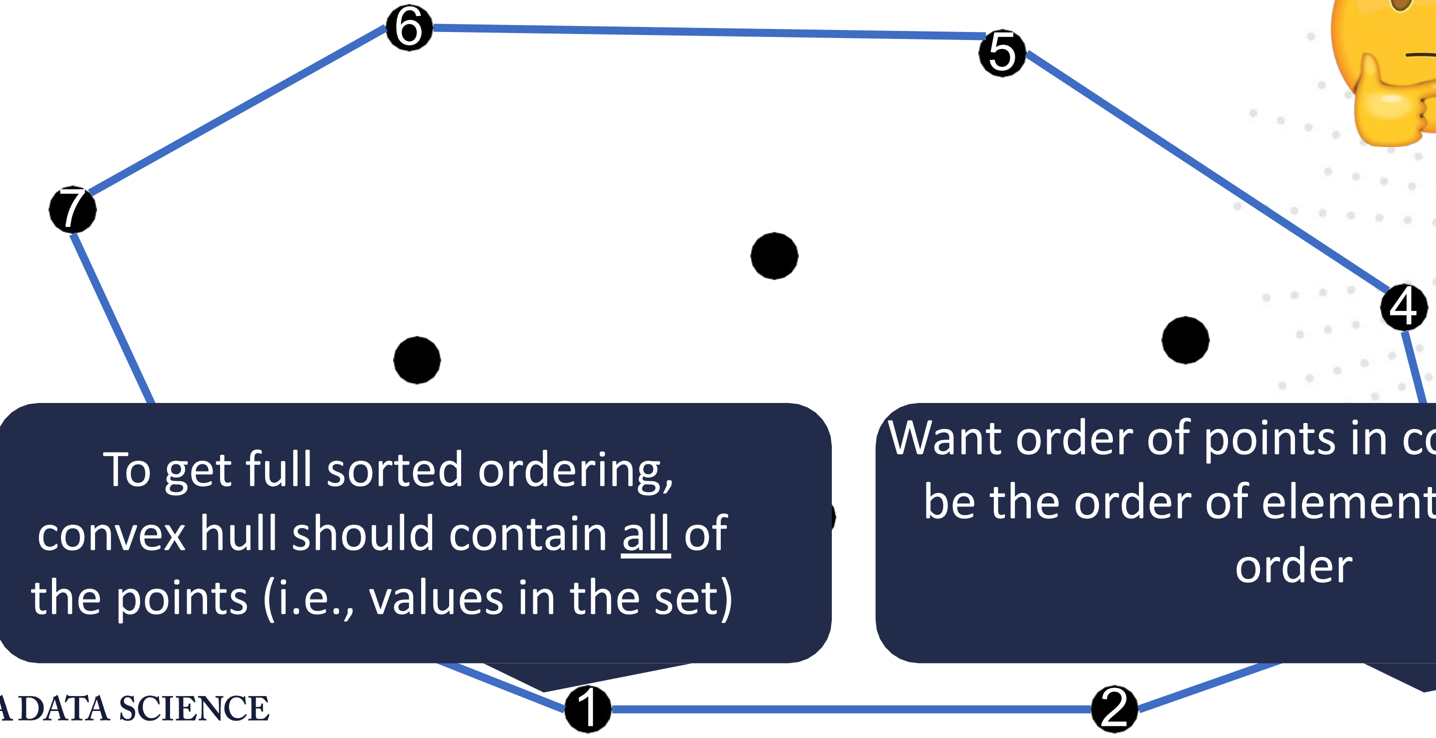
Sorting to Convex Hull Reduction

Can we use this to sort?



Sorting to Convex Hull Reduction

Can we use this to sort?

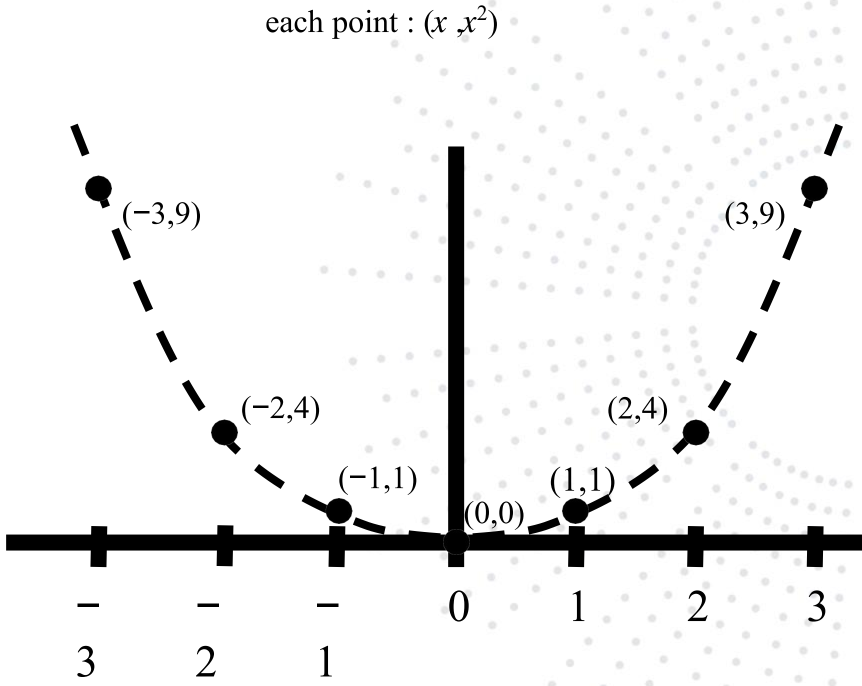


To get full sorted ordering, convex hull should contain all of the points (i.e., values in the set)

Want order of points in convex hull to be the order of elements in sorted order

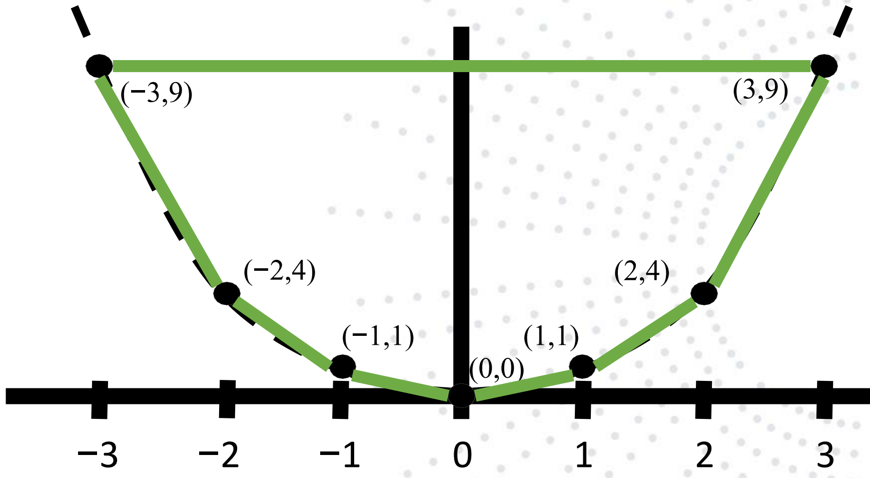
Sorting to Convex Hull Reduction

- **Step 1: Represent the Sorting Problem Geometrically**
  - **Map Each Element to a Point**: We are given a set of numbers x1, x2, x_n. To apply the convex hull technique, we need to represent each number as a point in the plane
    - We will use y=x^2 to construct the y-coordinate of each point.
    - Therefore, for each element x_i, we represent it as the point (x_i,x_i^2))



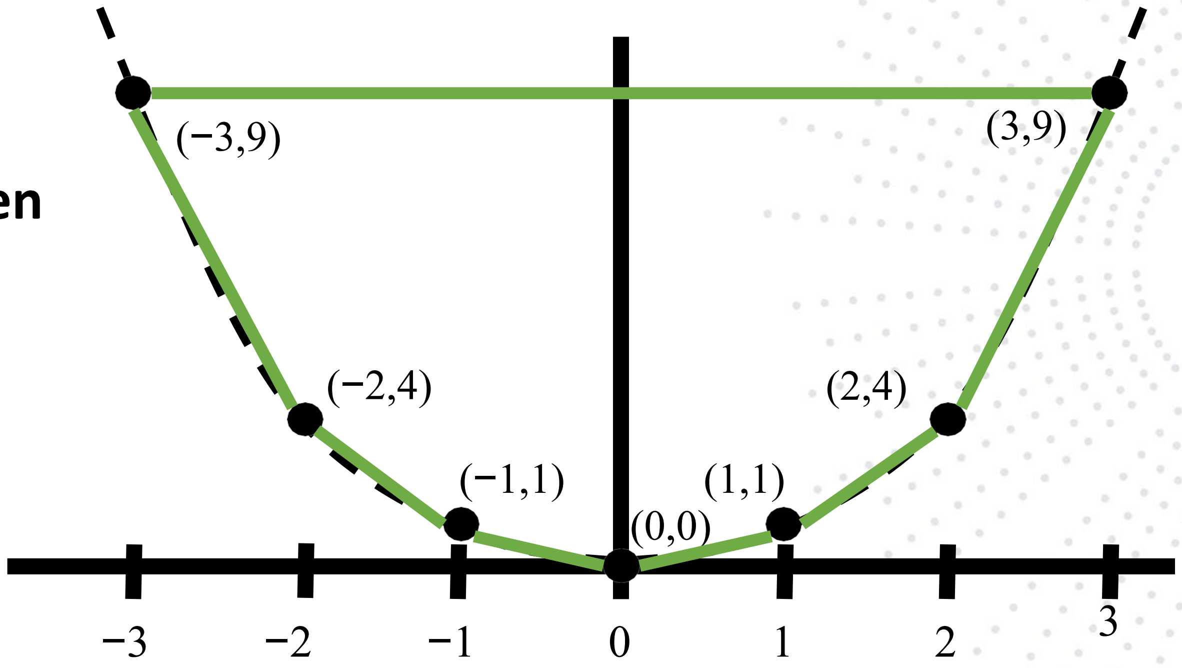
Sorting to Convex Hull Reduction

- **Step 2: Apply the Convex Hull Algorithm(like Graham's scan or Jarvis's)**
- **Step 3: Extract Sorted Order from Convex Hull**
  - Once we have computed the convex hull, the key observation is that the convex hull will naturally traverse the points in increasing order of x-coordinates.
  - Convex hull algorithm works by processing the points in a manner that reflects the left-to-right traversal along the x-axis



Sorting to Convex Hull Reduction

Conclusion: If we can solve convex hull, then we can sort numeric values



Sorting to Convex Hull Reduction

List of numbers to sort

(-2,1, -3, 0, 2, 3, -1)

Sorted List

$(-3, -2, -1, 0, 1, 2, 3)$

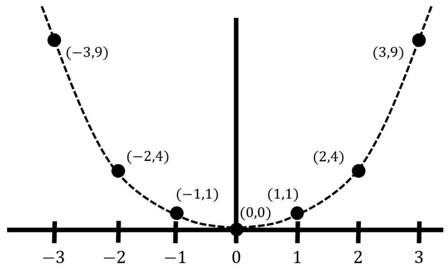
Map instances of problem A to instances of B

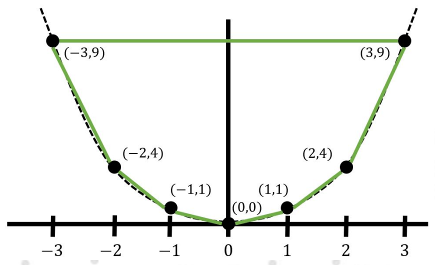
$O(n)$

Map solutions of problem B to solutions of A

$O(n)$

convex hull





Sorting to Convex Hull Reduction

- **Overall Time Complexity in average case: O(nlogn)**
    - Mapping elements to points: O(n)
    - Convex hull algorithm: O(n logn) (if using Graham's scan)
    - Extracting sorted order: O(n)