Convex Hull

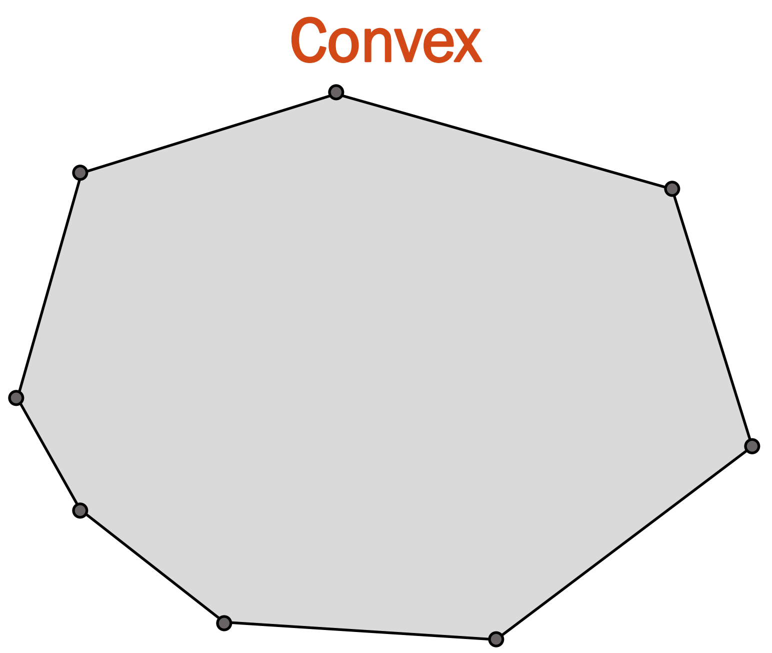
Dr. Bibhudatta Sahoo

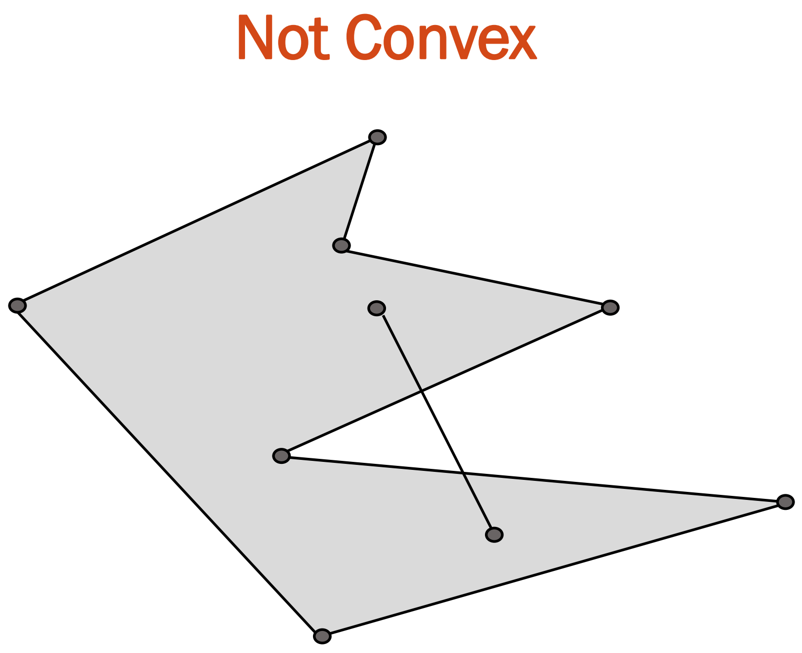
Communication & Computing Group

Department of Computer Science & Engineering, NIT Rourkela

Email: bdsahu@nitrkl.ac.in, 9937324437, 2462358

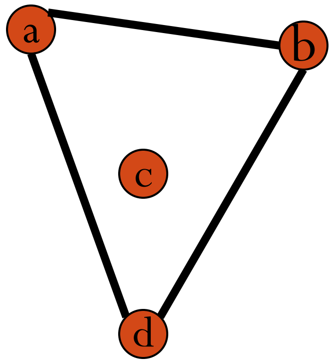
Convex Polygon

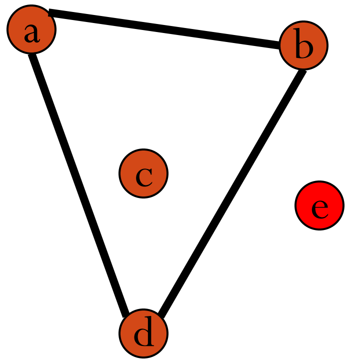


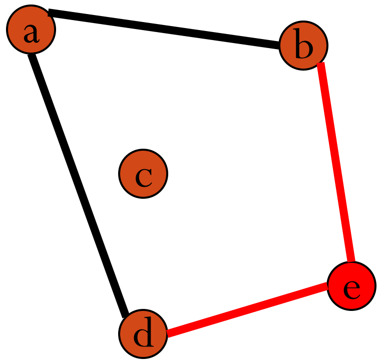


A polygon is *convex* iff for any two points in the polygon (interior ∪ boundary) the segment connecting the points is entirely within the polygon.

Convex Polygon

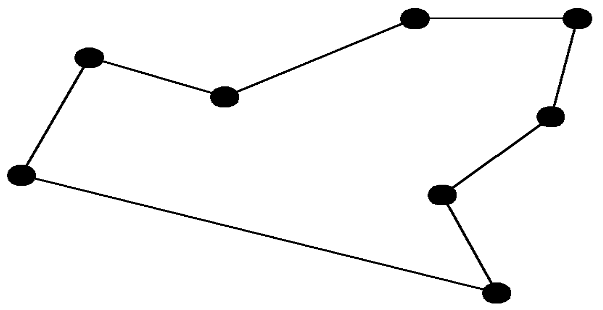




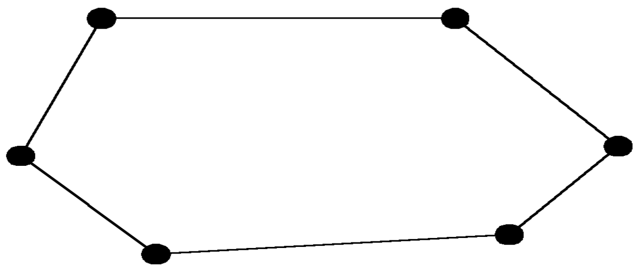


The convex hull

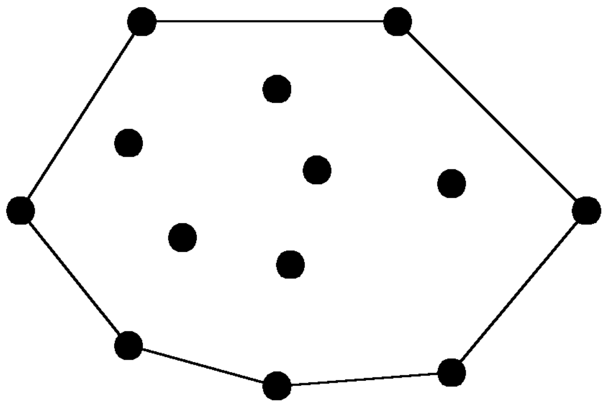
concave polygon:



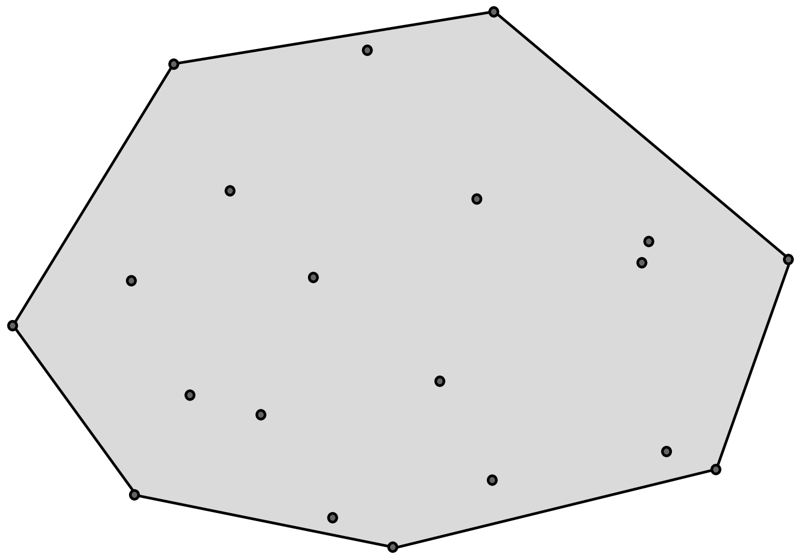
convex polygon:



- A planar **polygon** is convex if it contains *all* the **line segments** connecting any pair of its points.
- A planar polygon that is not convex is said to be a **concave polygon**.
- The convex hull of a set of planar points is the smallest convex polygon containing all of the points.



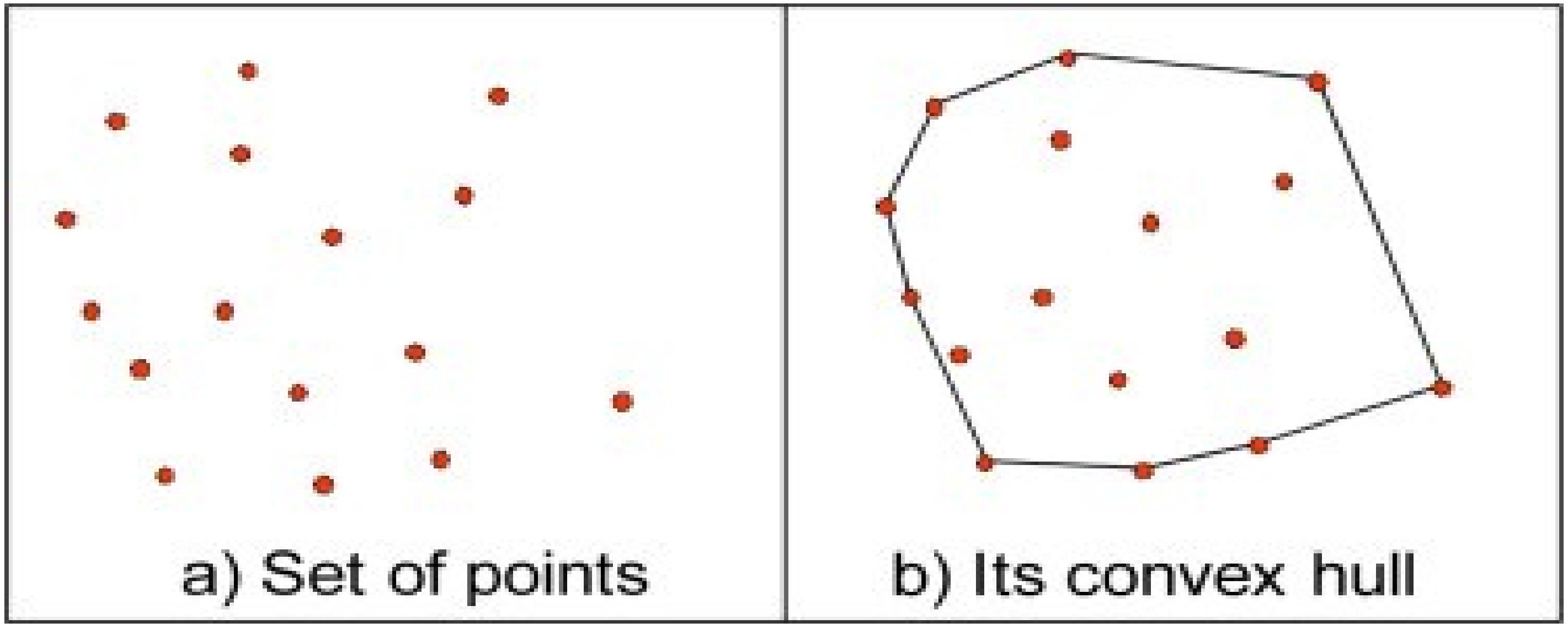
Convex Hull



Given a set $S = \{p_1, p_2, \ldots, p_N\}$ of points in the plane, the convex hull $H(S)$ is the **smallest** convex polygon in the plane that contains all of the points of S.

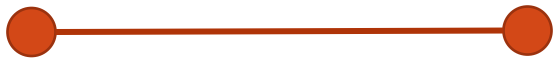
Convex Hull

- The **Convex Hull** is the line completely enclosing a set of points in a plane so that there are no concavities in the line.
- More formally, we can describe it as the smallest convex polygon which encloses a set of points such that each point in the set lies within the polygon or on its perimeter.

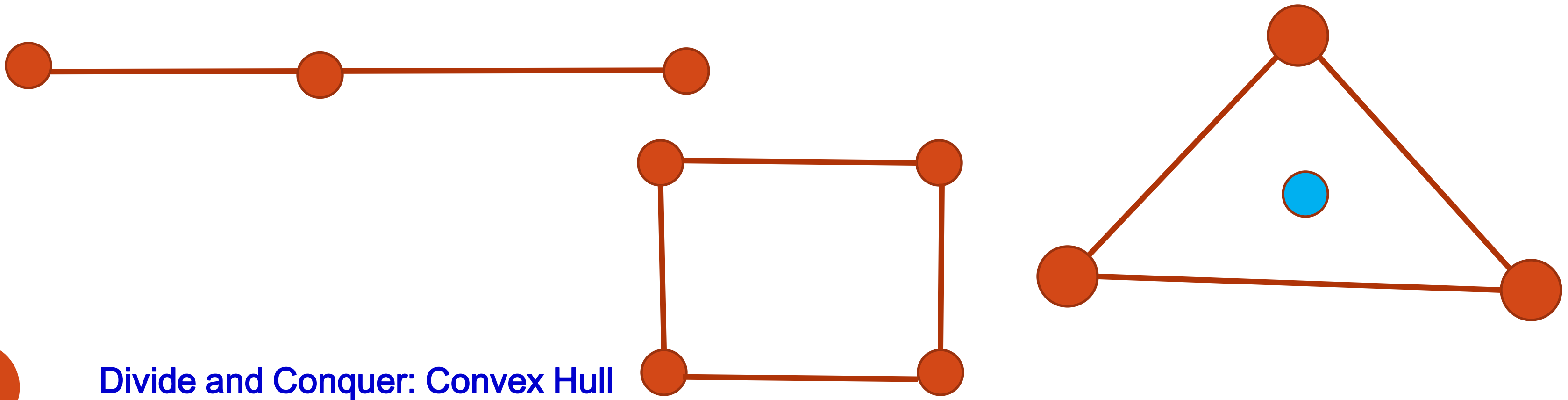


Basic Concepts

- If the set of points given contains only one point, that point is its own convex hull.
- The convex hull of two unique points is the line segment between the two points.

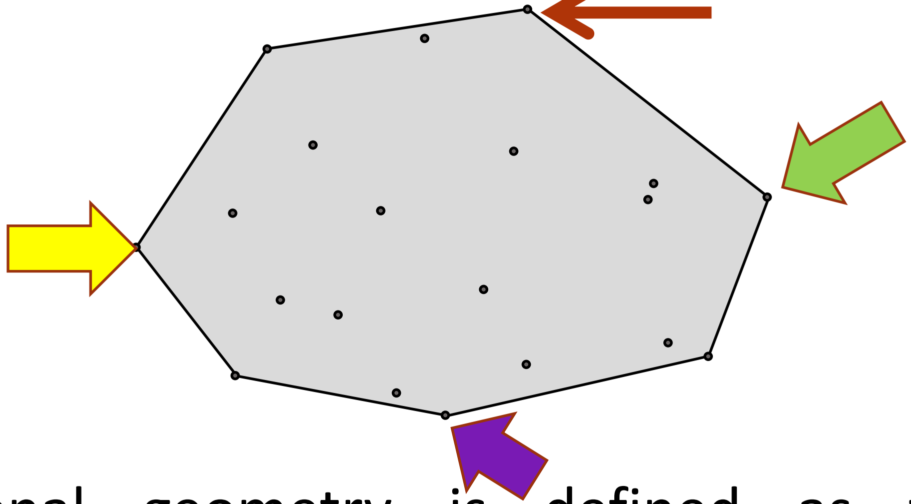


- The convex hull of three nonlinear points is the triangle formed by these three points. If the points are collinear, the convex hull is the line segment connecting the points with the endpoints as extreme points.



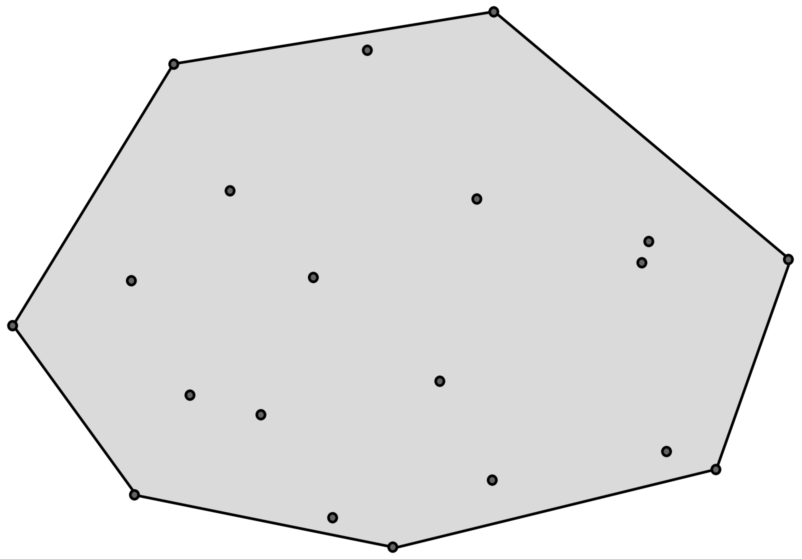
Basic Concepts

- An extreme point is a point that lies on the convex hull. In other words, one of the vertices on the polygon that the hull forms.
- Four extreme points are obvious when given a set those to the furthest right and left, and those at the top and bottom.



- Computational geometry is defined as the study of algorithms to solve problems in computer science in terms of geometry. There are several different algorithms with different efficiencies used to solve the problem of the convex hull.

Convex Hull: Introduction 1/3



Given a set $S = \{p_1, p_2, \ldots, p_N\}$ of points in the plane, the convex hull $H(S)$ is the smallest convex polygon in the plane that contains all of the points of S.

Convex Hull: Introduction 2/3

- most ubiquitous structure in computational geometry
- useful to construct other structures
- many applications: robot motion planning, shape analysis etc.
- a beautiful object, one of the early success stories in computational geometry that sparked interest among Computer Scientists by the invention of O(nlogn) algorithm rather than a $O(n^3)$ algorithm.
- intimately related to sorting algorithm for both lower and upper bound.

Convex Hull: Introduction 3/3

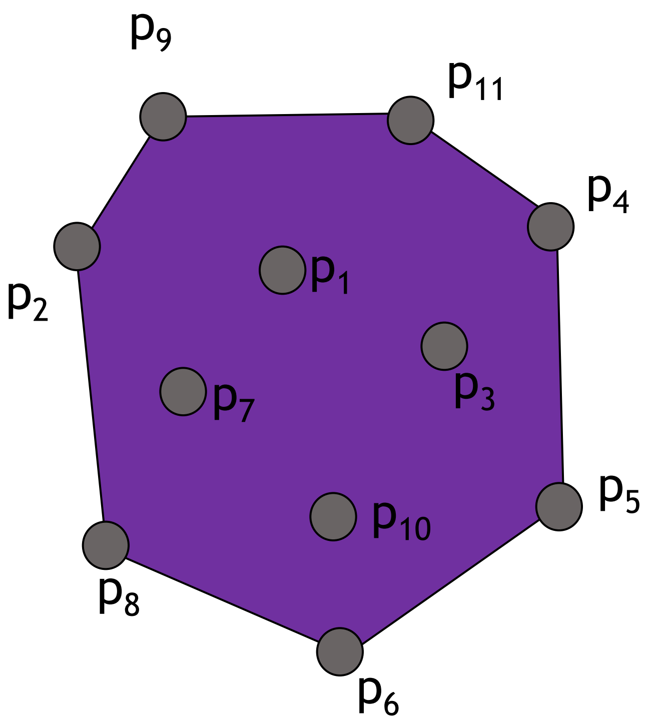
- Computing a convex hull (or just "hull") is one of the first sophisticated geometry algorithms, and there are many variations of it. The most common form of this algorithm involves determining the smallest convex set (called the "convex hull") containing a discrete set of points.
- This algorithm also applies to a polygon, or just a set of segments, whose hull is the same as the hull of its vertex point set.
- There are numerous applications for convex hulls: collision avoidance, hidden object determination, and shape analysis to name a few.
- There are several algorithms that can determine the convex hull of a given set of points. Some famous algorithms are the **gift wrapping algorithm** and the **Graham scan algorithm**.

Example: Convex Hull

- Assume that there are a few nails hammered half-way into a plank of wood as shown in Figure (p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}).
- You take a rubber band, stretch it to enclose the nails and let it go.
- It will fit around the outermost nails (shown in blue) and take a shape that minimizes its length(p_2, p_9, p_{11}, p_4, p_5, p_6, p_8, p_2).
- The area enclosed by the rubber band is called the **convex hull** of the set of nails.

Input: $p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8,p_9,p_{10},p_{11}$

Output: $p_2,p_9,p_{11},p_4,p_5,p_6,p_8,p_2$



Running times for different sizes of input.

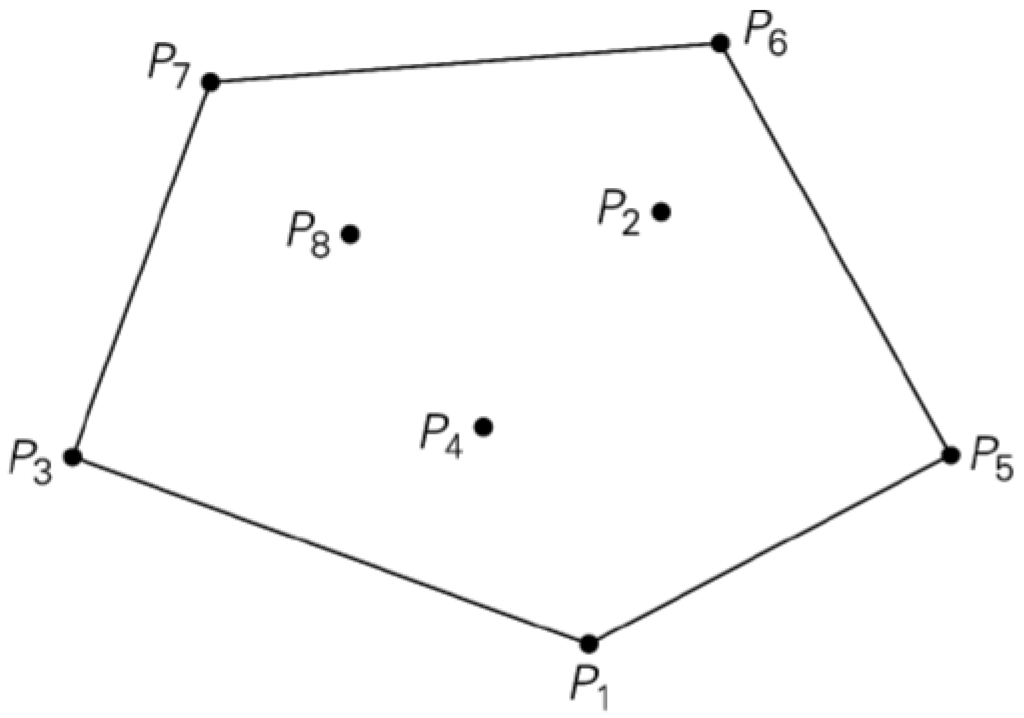
n	$\log n$	n	$n \log n$	n^2	n^3	2^n
8	3 nsec	0.01 μ	0.02 μ	0.06 μ	0.51 μ	0.26 μ
16	4 nsec	0.02 μ	0.06 μ	0.26 μ	4.10 μ	65.5 μ
32	5 nsec	0.03 μ	0.16 μ	1.02 μ	32.7 μ	4.29 sec
64	6 nsec	0.06 μ	0.38 μ	4.10 μ	262 μ	5.85 cent
128	0.01 μ	0.13 μ	0.90 μ	16.38 μ	0.01 sec	10^{20} cent
256	0.01 μ	0.26 μ	2.05 μ	65.54 μ	0.02 sec	10^{58} cent
512	0.01 μ	0.51 μ	4.61 μ	262.14 μ	0.13 sec	10^{135} cent
2048	0.01 μ	2.05 μ	22.53 μ	0.01 sec	1.07 sec	10^{598} cent
4096	0.01 μ	4.10 μ	49.15 μ	0.02 sec	8.40 sec	10^{1214} cent
8192	0.01 μ	8.19 μ	106.50 μ	0.07 sec	1.15 min	10^{2447} cent
16384	0.01 μ	16.38 μ	229.38 μ	0.27 sec	1.22 hrs	10^{4913} cent
32768	0.02 μ	32.77 μ	491.52 μ	1.07 sec	9.77 hrs	10^{9845} cent
65536	0.02 μ	65.54 μ	1048.6 μ	0.07 min	3.3 days	10^{19709} cent
131072	0.02 μ	131.07 μ	2228.2 μ	0.29 min	26 days	10^{39438} cent
262144	0.02 μ	262.14 μ	4718.6 μ	1.15 min	7 mnths	10^{78894} cent
524288	0.02 μ	524.29 μ	9961.5 μ	4.58 min	4.6 years	10^{157808} cent
1048576	0.02 μ	1048.60 μ	20972 μ	18.3 min	37 years	10^{315634} cent

Note: "nsec" stands for nanoseconds, " μ"is one microsecond and "cent" stands for centuries. The explosive running time (measured in centuries) when it is of the order 2^n.

Convex-Hull Problem

- The ***convex-hull problem*** is the problem of constructing the convex hull for a given set *S* of *n* points.
- A set of points(finite or infinite)in the plane is called **convex** if for any two points *P* and *Q* in the set, the entire line segment with the end points at *P* and *Q* belongs to the set.
- **Definition**: The ***convex hull*** of a set *S* of points is the **smallest** convex set/polygon containing *S*.
- **Theorem**: The convex hull of any set *S* of $n>2$ points (not all on the same line)is a convex polygon with the vertices at some of the points of *S*.

Convex-Hull Example



The convex hull for this set of eight points is the convex polygon with vertices at P_1, P_5, P_6, P_7, and P_3.

Convex-Hull Brute-Force Algorithm

- A line segment connecting two points *Pi* and *Pj* of a set of *n* points is a part of its convex hull's boundary if and only if all the other points of the set lie on the same side of the straight line through these two points.
- Repeating this test for every pair of points yields a list of line segments that make up the convex hull's boundary.

• The straight line through $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ can be defined by the equation

$$ax + by = c,$$

where $a = y_2 - y_1$, $b = x_1 - x_2$, $c = x_1y_2 - y_1x_2$.

Convex-Hull Brute-Force Algorithm

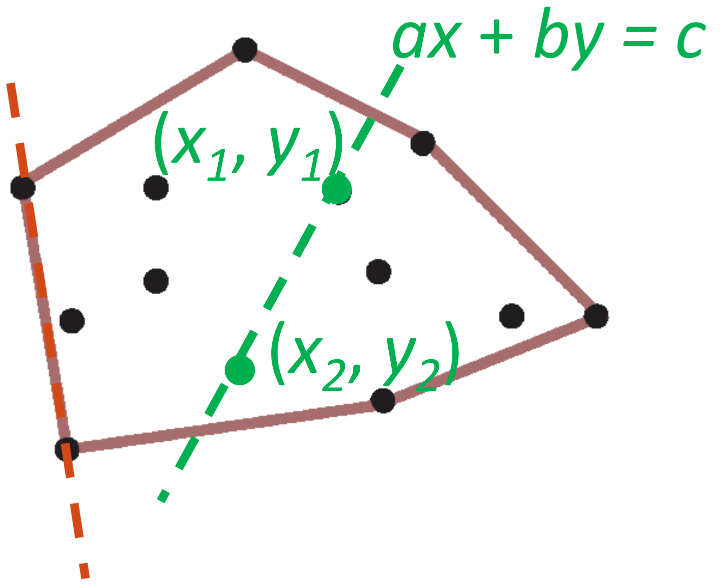
- The straight line through $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ can be defined by the equation

$$ax + by = c,$$

where $a = y_2 - y_1$, $b = x_1 - x_2$, $c = x_1 y_2 - y_1 x_2$.

- The line P_1P_2 divides the plane into two half-planes:
  - for all the points in one of them, $ax + by > c$,
  - for all the points in the other, $ax + by < c$,
  - for the points on the line, $ax + by = c$.
- Thus, to check whether certain points lie on the same side of the line, we can check whether the expression $ax + by = c$ has the same sign at each of these points.

Brute-Force Convex Hull Algorithm



$a = y_2 - y_1$
$b = x_1 - x_2$
$c = x_1y_2 - y_1x_2$

- The time efficiency of this algorithm is in $O(n^3)$.
  - for each of $n(n - 1)/2$ pairs of distinct points, we may need to find the sign of $ax + by - c$ for each of the other $n - 2$ points.

Brute Force Convex Hull

- Consider the line defined by the first pair of points
- If all of the other points lie on one side of that line, the line is part of the convex hull
- Repeat this process for every other pair of points
- This process is **O(*N*^{3})**

High level pseudo code for Convex Hull (Brute-Force)

```
1. for each point $P_i$
2. for each point $P_j$ where $P_j$ # $P_i$
3. Compute the line segment for $P_i$ and $P_j$
4. for every other point $P_k$ where $P_k$ # $P_i$ and $P_k$ # $P_j$
5. If each $P_k$ is on one side of the line segment, label $P_i$ and $P_j$ in
   the convex hull
```

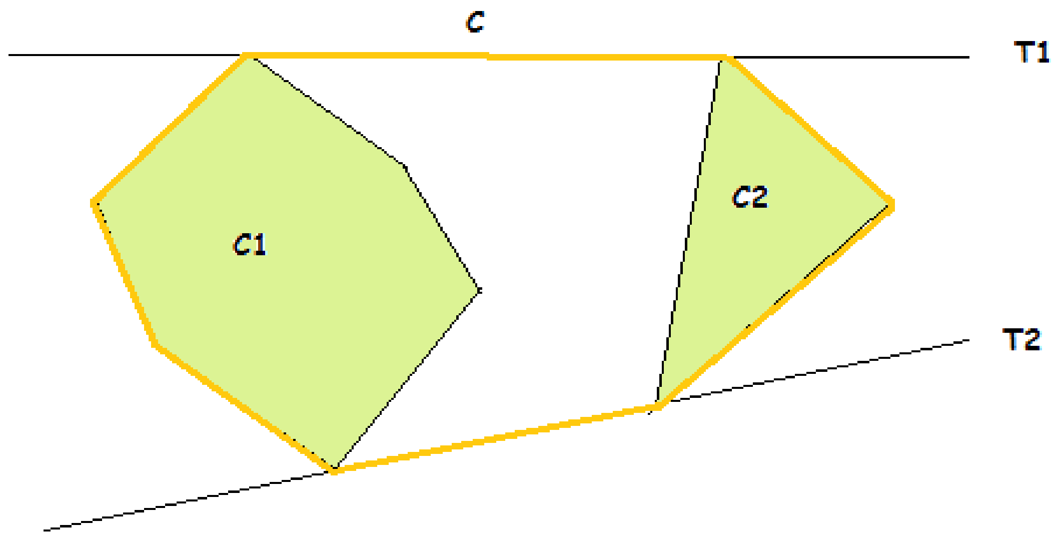
Divide-and-Conquer Convex Hull Algorithm

Divide and conquer algorithm for Conex Hull

- **Given S: the set of points for which we have to find the convex hull.**
- Let us divide S into two sets:
- S1: the set of left points
- S2: the set of right points
- Note that all points in S1 is left to all points in S2.
- Suppose we know the convex hull of the left half points S1 is C1 and the right half points S2 is C2.
- Then the problem now is to merge these two convex hulls C1 and C2 and determine the convex hull C for the complete set S.
  This can be done by finding the **upper and lower tangent to the right and left convex hulls C1 and C2**.

Divide and conquer algorithm for Conex Hull

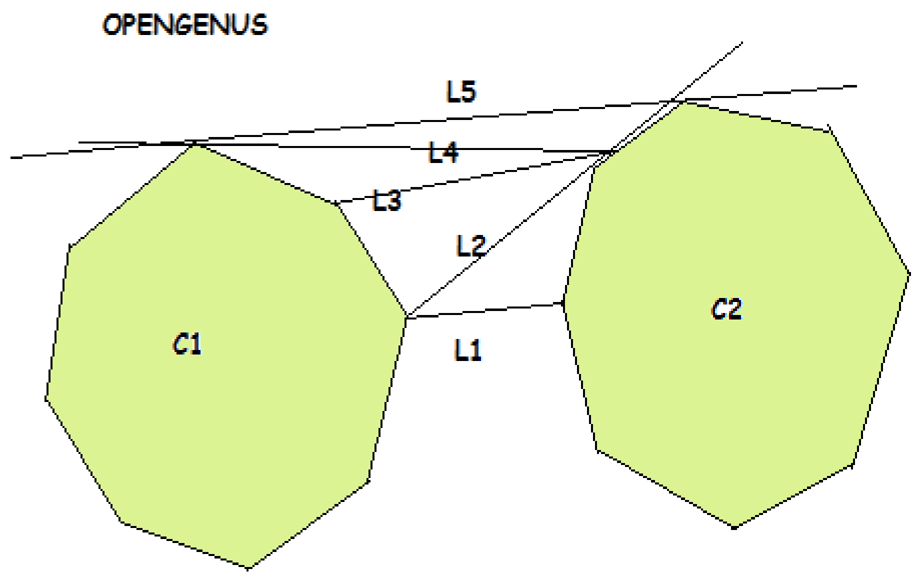
- Let the left convex hull be C1 and the right convex hull be C2. Then the lower and upper tangents are named as T1 and T2 respectively, as shown in the figure.
- Then the yellow outline shows the final convex hull.



DEMONSTRATION OF MERGING OF TWO CONVEX HULLS (OPENGENUS)

The merging of the left and the right convex hulls take **O(n)** time

Tangents between two convex polygons

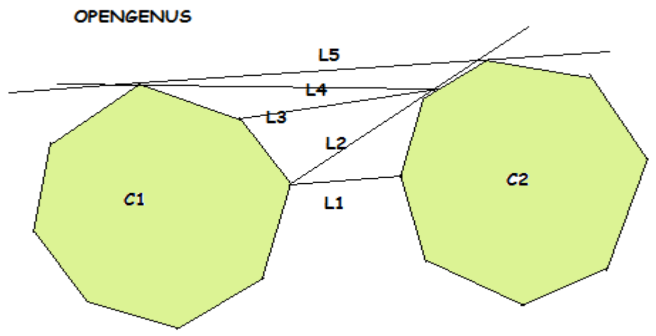


DEMONSTRATION OF FINDING UPPER TANGENT OF TWO POLYGONS C1 AND C2

- The rightmost point (say A) of left convex hull C1 and leftmost point (say B) of right convex hull C2. The line joining them is labelled as L1.

Tangents between two convex polygons

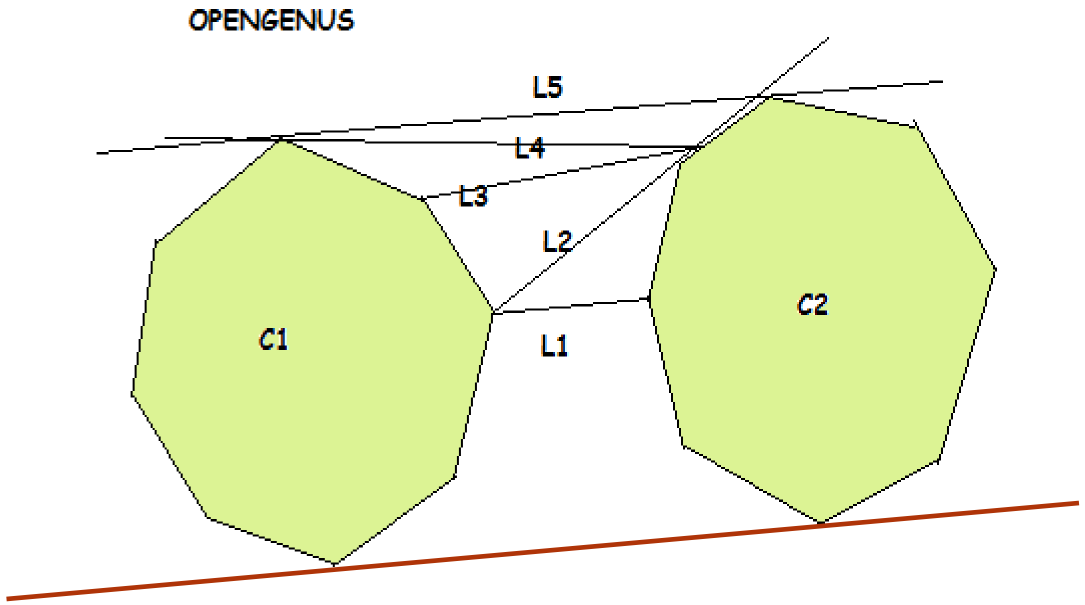
- The rightmost point (say A) of left convex hull C1 and leftmost point (say B) of right convex hull C2. The line joining them is labelled as L1.
- As this line passes through the polygon C2 (is not above polygon B) so we take the anti-clockwise next point on C2, the line is labelled 2. Now the line is above the polygon C2, fine! But the line is crossing the polygon C1, so we move to the clockwise next point, labelled as 3 in the picture. This again crossing the polygon a so we move to line 4. This line is crossing b so we move to **line 5**. Now this line is crossing neither of the points. So this is the upper tangent for the given polygons.



DEMONSTRATION OF FINDING UPPER TANGENT OF TWO POLYGONS C1 AND C2

Tangents between two convex polygons

- For finding the lower tangent we need to move inversely through the polygons i.e. if the line is crossing the polygon C2 we move to clockwise next and to anti-clockwise next if the line is crossing the polygon C1.



DEMONSTRATION OF FINDING UPPER TANGENT OF TWO POLYGONS C1 AND C2

Divide-and-conquer for convex hull

Input : A set S of planar points

Output : A convex hull for S

Step 1: If S contains no more than five points, use exhaustive searching to find the convex hull and return.

Step 2: Find a median line perpendicular to the X-axis which divides S into S_L and S_R ; S_L lies to the left of S_R .

Step 3: Recursively construct convex hulls for S_L and S_R. Denote these convex hulls by Hull(S_L)=C1 and Hull(S_R)=C2 respectively.

Step 4: Apply the merging procedure to merge Hull C1 and C2 together to form a convex hull.

$T(n) = c$, when $n \le 5$; $T(n) = 2T(n/2) + O(n)$, when $n > 5$

$\Rightarrow T(n) = O(n \log n)$

Complexity analysis

- Polygons C1 and C2 can be merged in time proportional to the number of points in C1 and C2 together.
- Merging process by finding the lower and upper tangents as T1 and T2can be found in O(|C1|+|C2|) times, that is in linear time in the number of hull vertices.

Recurrence relation for time complexity

$T(n) = b$, when $n=1$;
$T(n) = 2T(n/2) + cn$, when $n>1$
c and b are constants

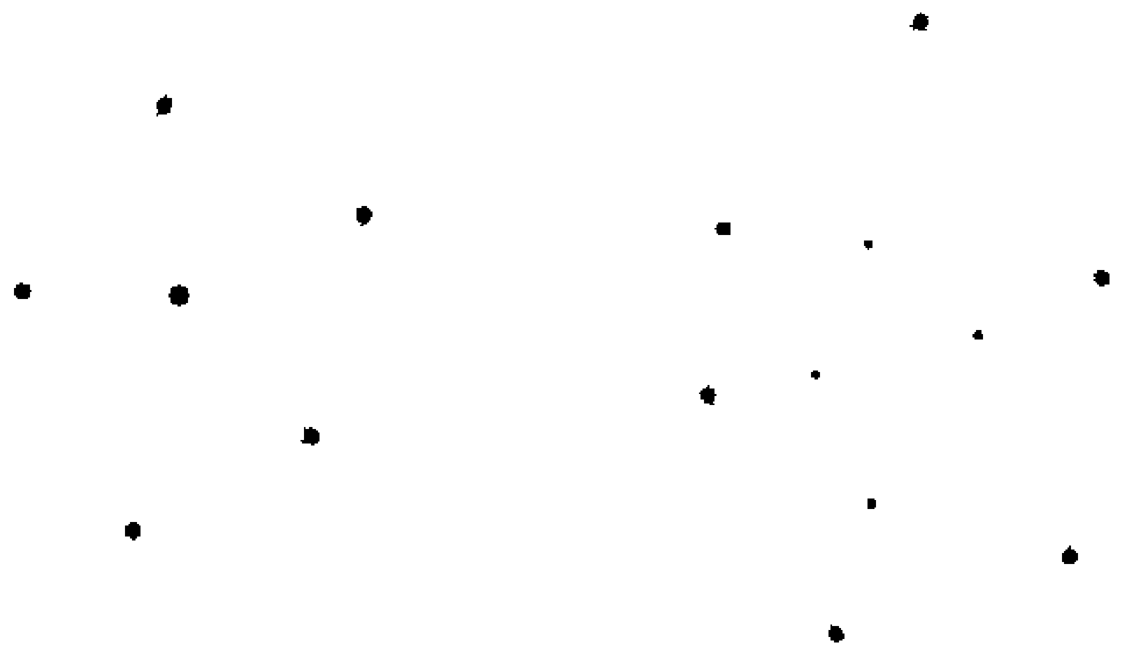
$T(n) = b$, when $n \le 5$;
$T(n) = 2T(n/2) + cn$, when $n>5$
c and b are constants

$\Rightarrow T(n) = O(n \log n)$

An algorithm to Construct a Convex Hull

- Input set S of planar points
- Output Convex hull of S

Initial Points



An algorithm to Construct a Convex Hull(Cont'd)

- **Step 1**.

  If S contains no more than five points, use exhaustive searching to find the convex hull and return.

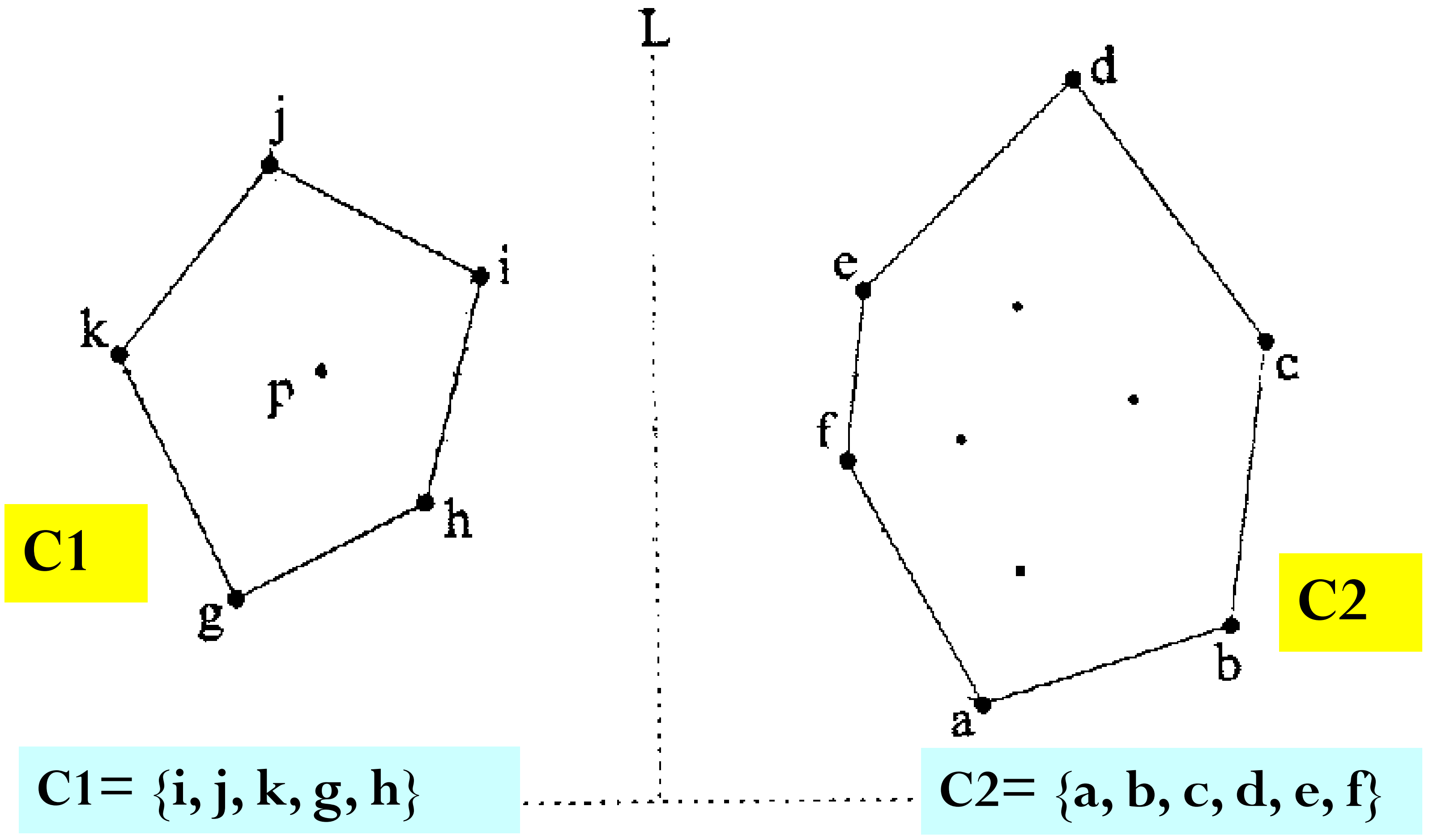
- **Step 2**.

  Find a median line perpendicular to the x-axis which divides S into S1 and S2.

- **Step 3**.

  Recursively construct convex hulls for S1 and S2. Denote these convex hulls by Hull by C1=Hull(S1) and C2=Hull(S2) respectively.

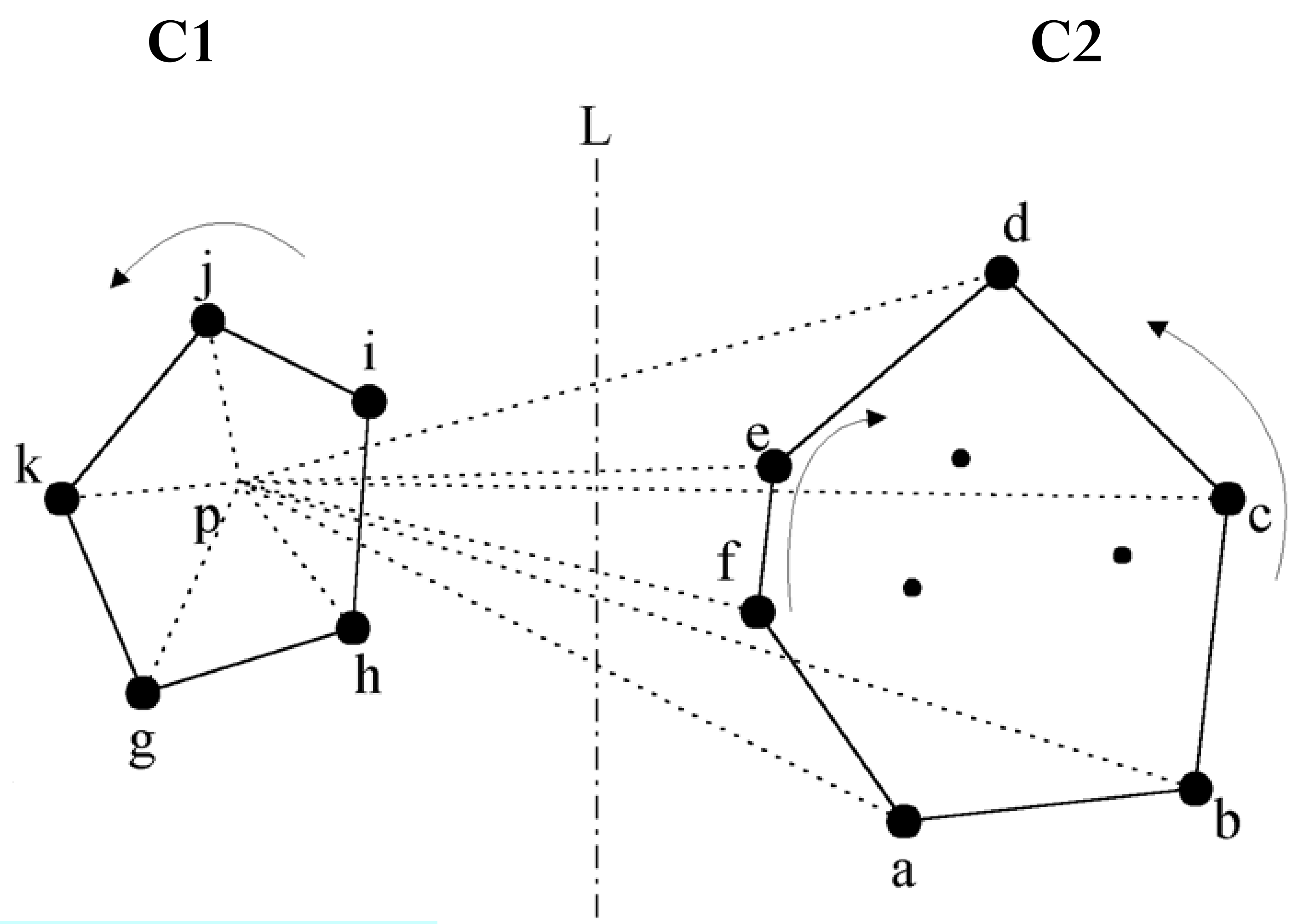
Convex Hulls after Step 3



C1= {i, j, k, g, h}

C2= {a, b, c, d, e, f}

An algorithm to Construct a Convex Hull(Cont'd)



C1= {i, j, k, g, h}

C2= {a, b, c, d, e, f}

An algorithm to Construct a Convex Hull(Cont’d)

- **Step 4.**

  Find an interior point P of C1. Find the vertices v1 and v2 of Hull(C2) which divide the vertices of Hull(C2) into two sequences of vertices which have increasing polar angles with respect to P. Without loss of generality, let us assume that v1 has y-value greater than v2. Then form three sequences as follows :

  (a) Sequence 1 : all of the convex hull vertices of Hull(C1) in counterclockwise direction.

  (b) Sequence 2 : the convex hull vertices of Hull(C2) from v2 to v1 in counterclockwise direction.

  (c) Sequence 3 : the convex hull vertices of Hull (C2) from v2 to v1 in clockwise direction

An algorithm to Construct a Convex Hull(Cont'd)

The merging procedure:

1. Select an interior point p.
2. There are 3 sequences of points which have increasing polar angles with respect to p.

   (1) g, h, i, j, k

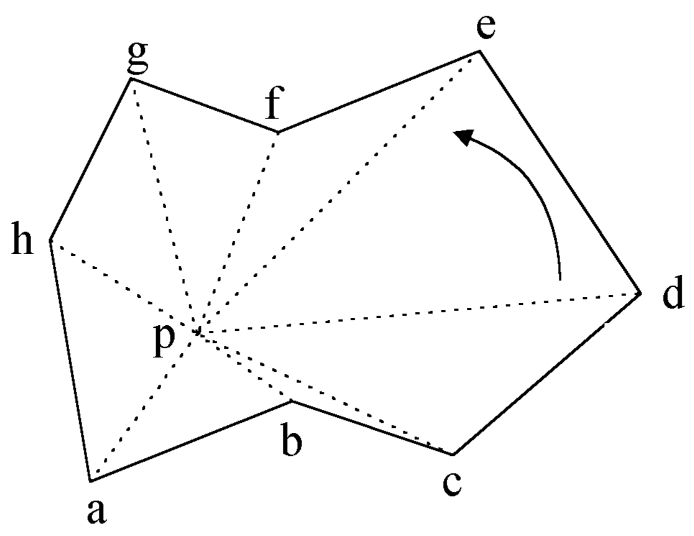
   (2) a, b, c, d

   (3) f, e
3. Merge these 3 sequences into 1 sequence:

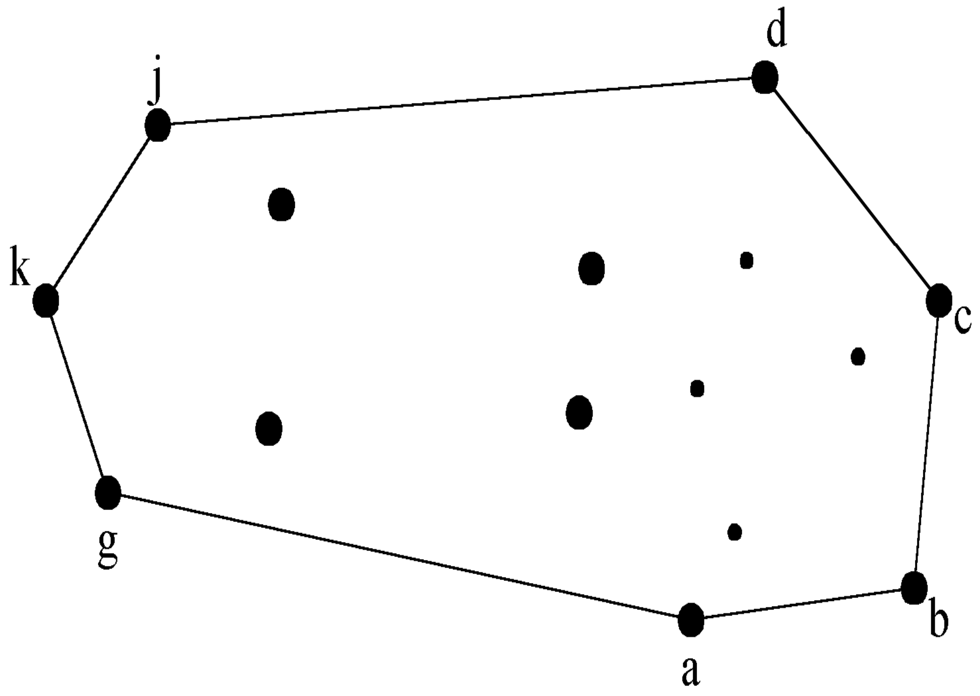
   g, h, a, b, f, c, e, d, i, j, k.
4. Apply Graham scan to examine the points one by one and eliminate the points which cause reflexive angles.

Construct a Convex Hull(Cont'd)

e.g. points b and f need to be deleted.



Final result:



Graham's Scan algorithm

Graham-Scan ALGORITHM for constructing CH

- Given a set of points on the plane, Graham's scan computes their convex hull.
- The algorithm works in three phases:

1. Find an extreme point. This point will be the *pivot*, is guaranteed to be on the hull, and is chosen to be the point with largest *y* coordinate.
2. Sort the points in order of increasing angle about the pivot. We end up with a star-shaped polygon (one in which one special point, in this case the pivot, can "see" the whole polygon).
3. Build the hull, by marching around the star-shaped poly, adding edges when we make a left turn, and back-tracking when we make a right turn.

Algorithm Graham's scan has these steps:

1. Find p_0, the point with the minimum y coordinate,
2. Sort all the remaining points in order of their polar angle from p_0
3. Initialize a stack, ***S***
4. push(S,p_0)
5. push(S,p_1)
6. push(S,p_2)
7. for i=3 to ***n*** do
   while (angle formed by topnext(S), top(S) and p_i makes a right turn
   pop(S)
8. push(S,p_i)
9. return S

This algorithm takes O(n log n) time.

Time Complexity of Graham Scan

Phase 1 takes time O(N log N)

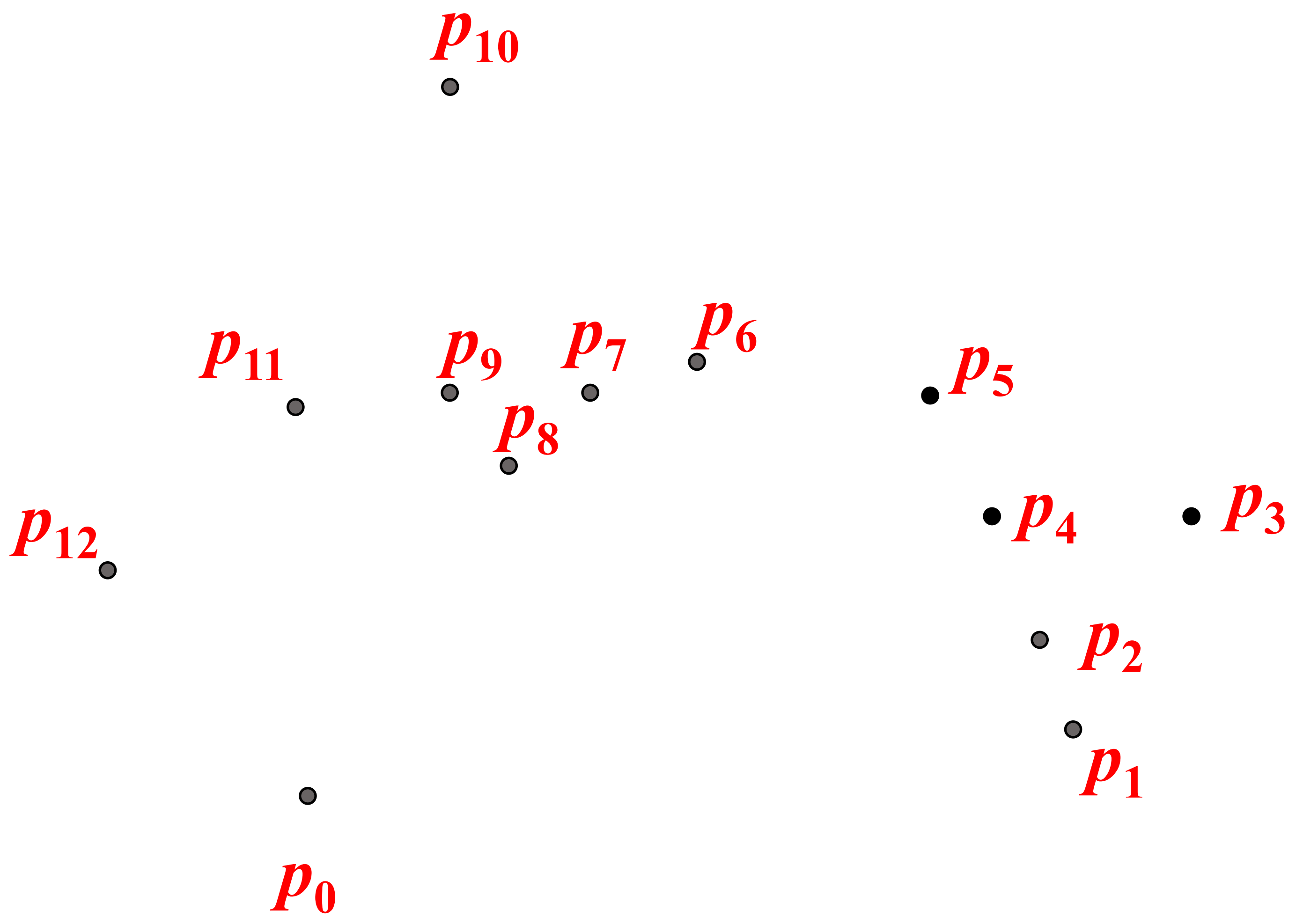
points are sorted by angle around the anchor

Phase 2 takes time O(N)

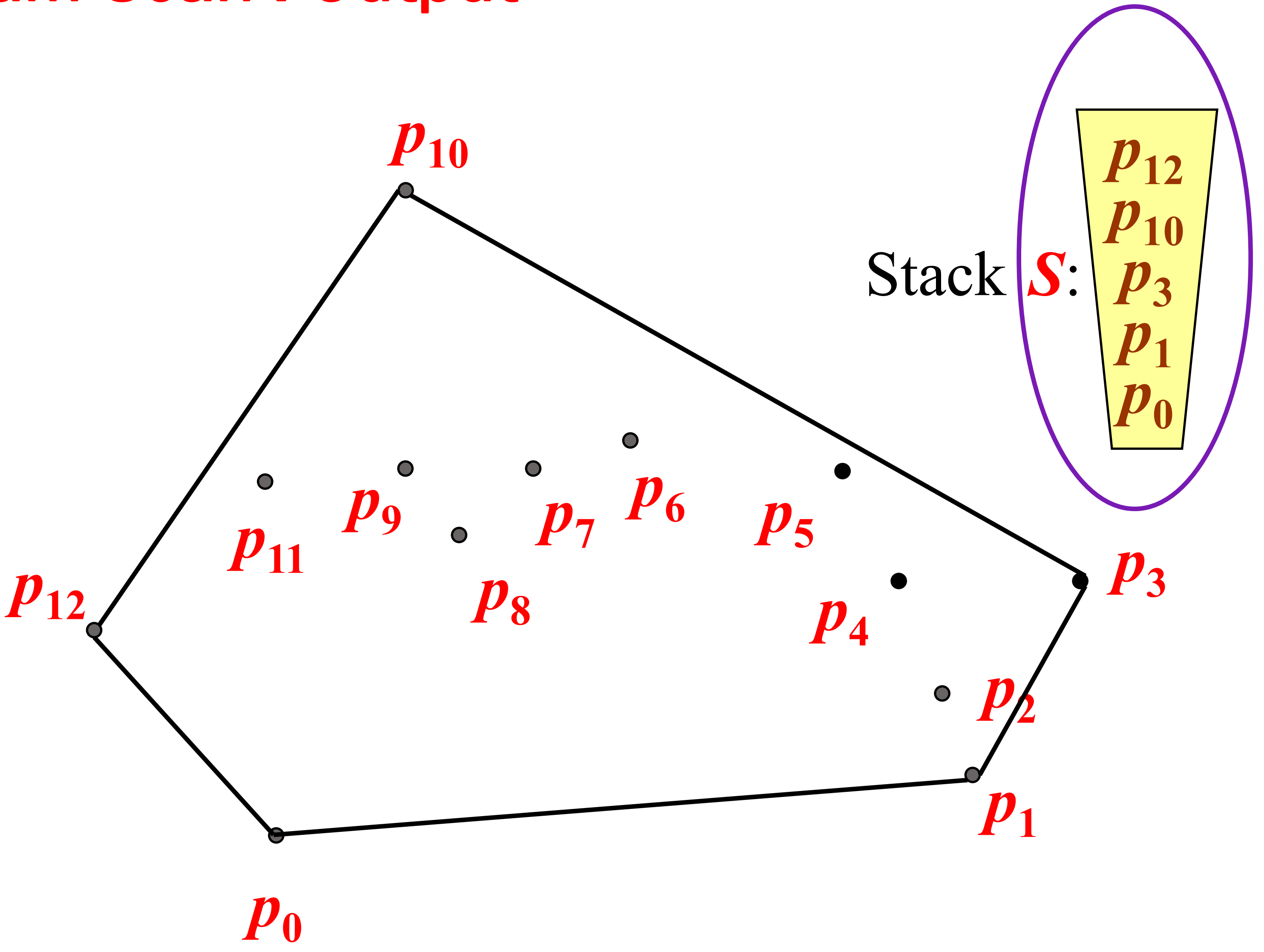
each point is inserted into the sequence exactly once, and each point is removed from the sequence at most once

- Total time complexity **O(N log N)**

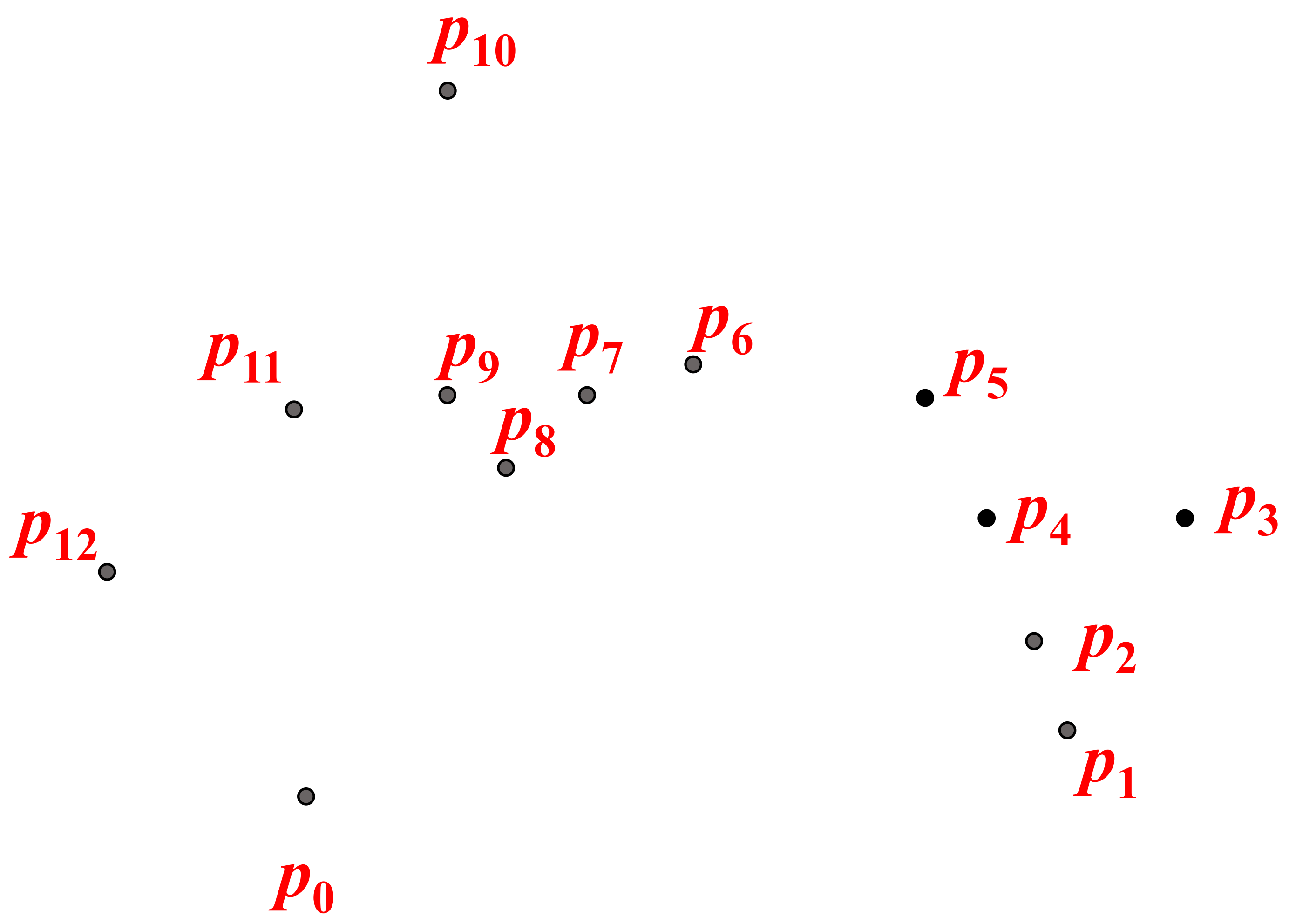
Graham-Scan : Input



Graham-Scan : Output



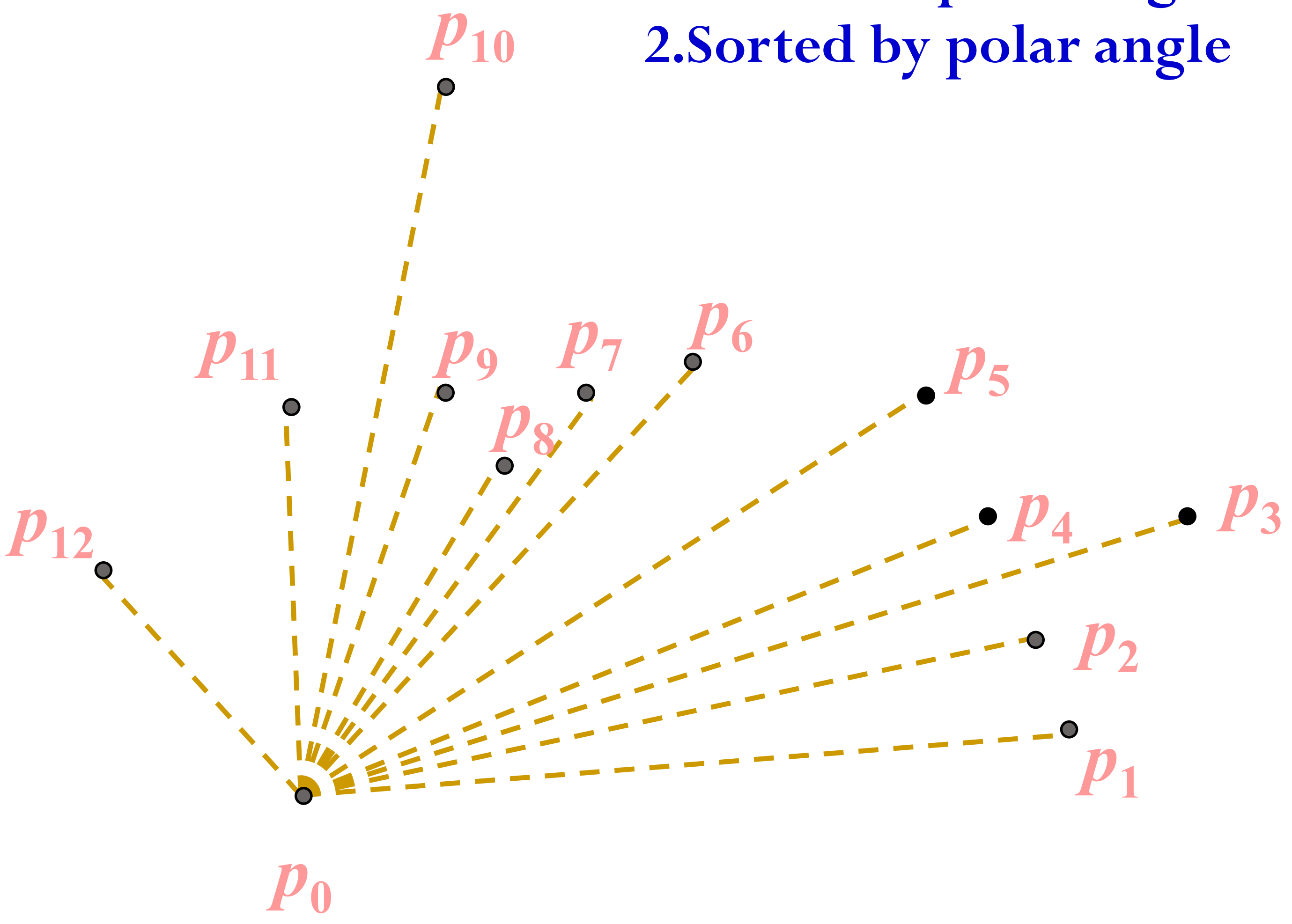
Graham-Scan : Input



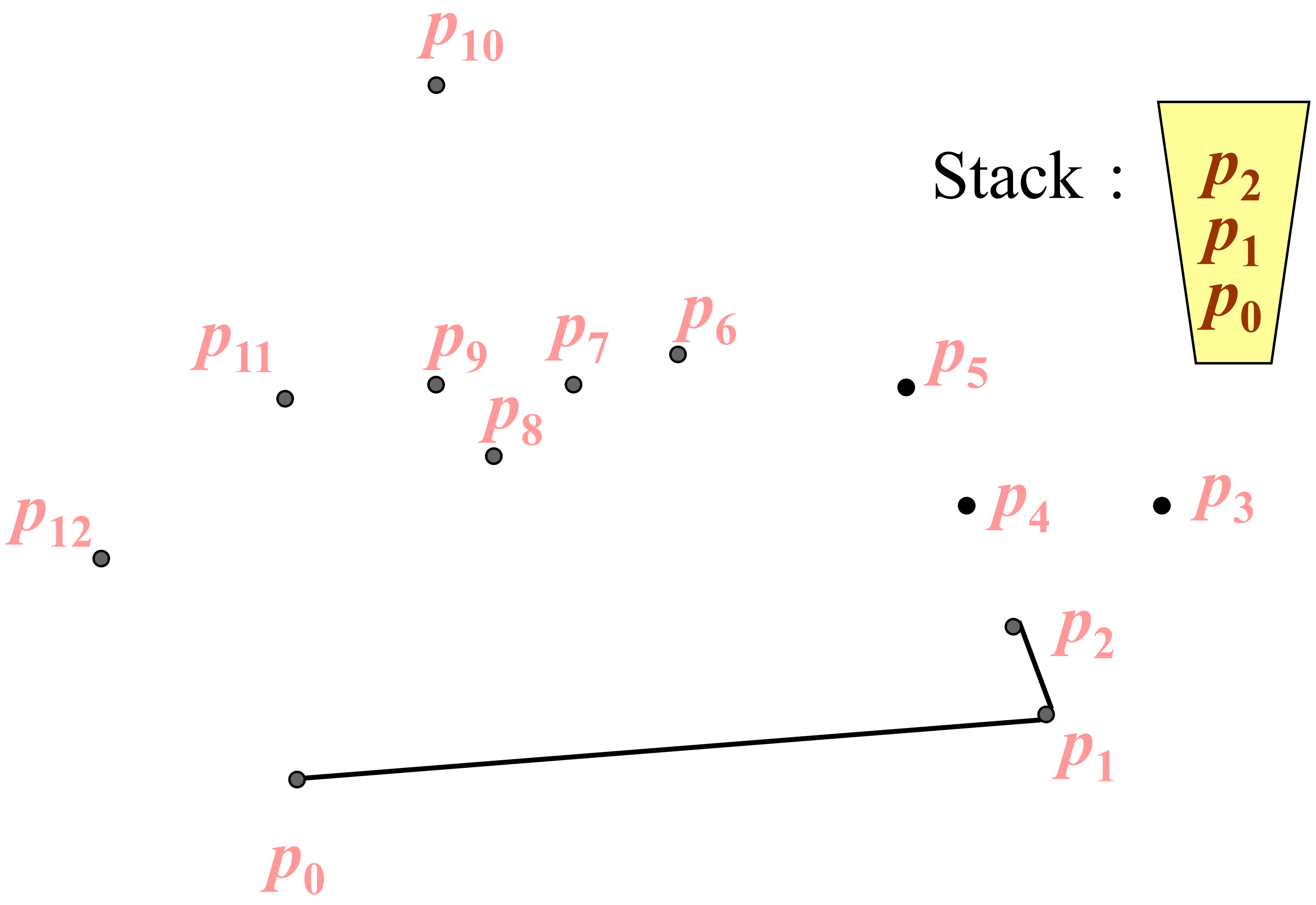
Graham-Scan :(1/11)

1.Calculate polar angle

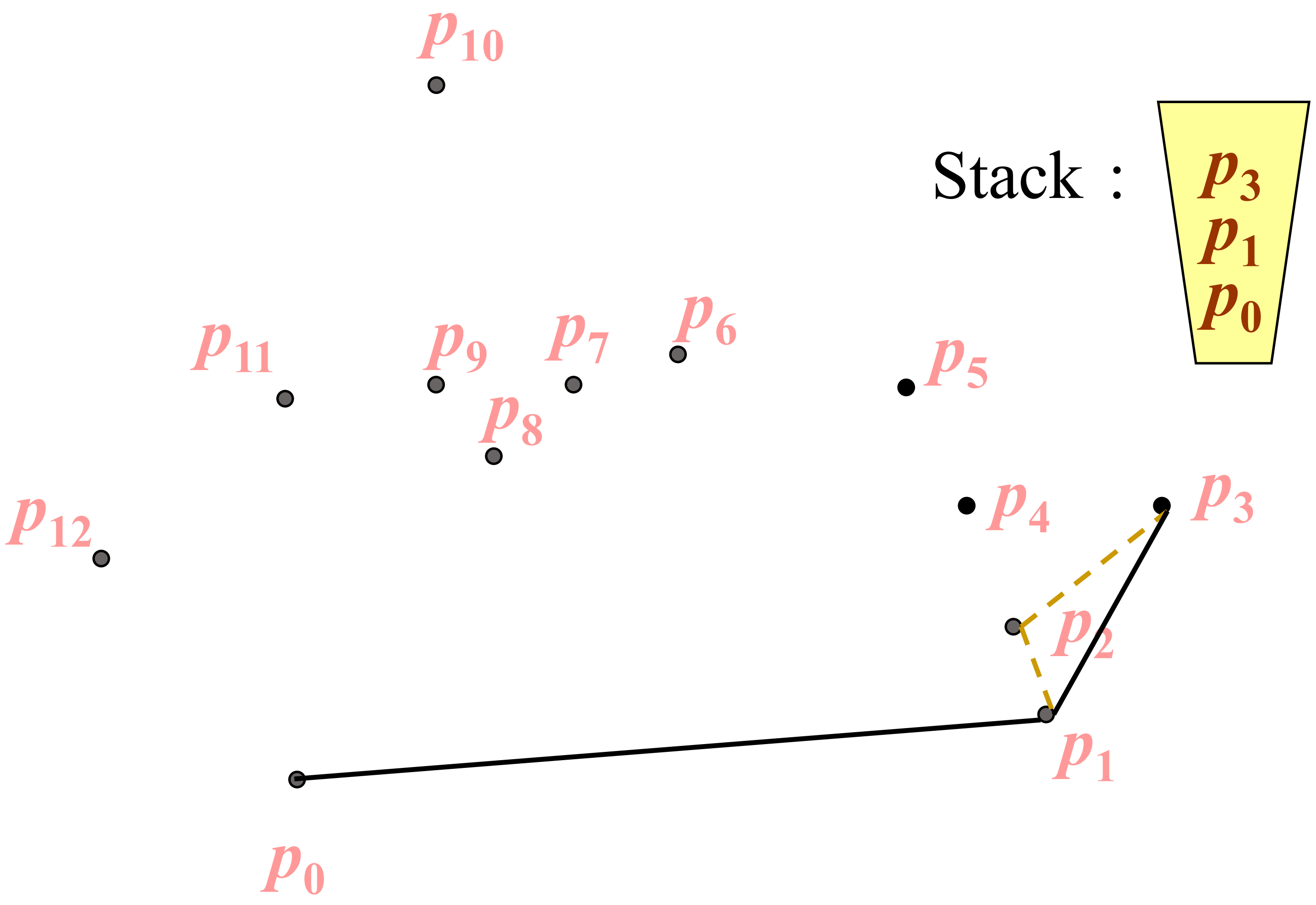
2.Sorted by polar angle



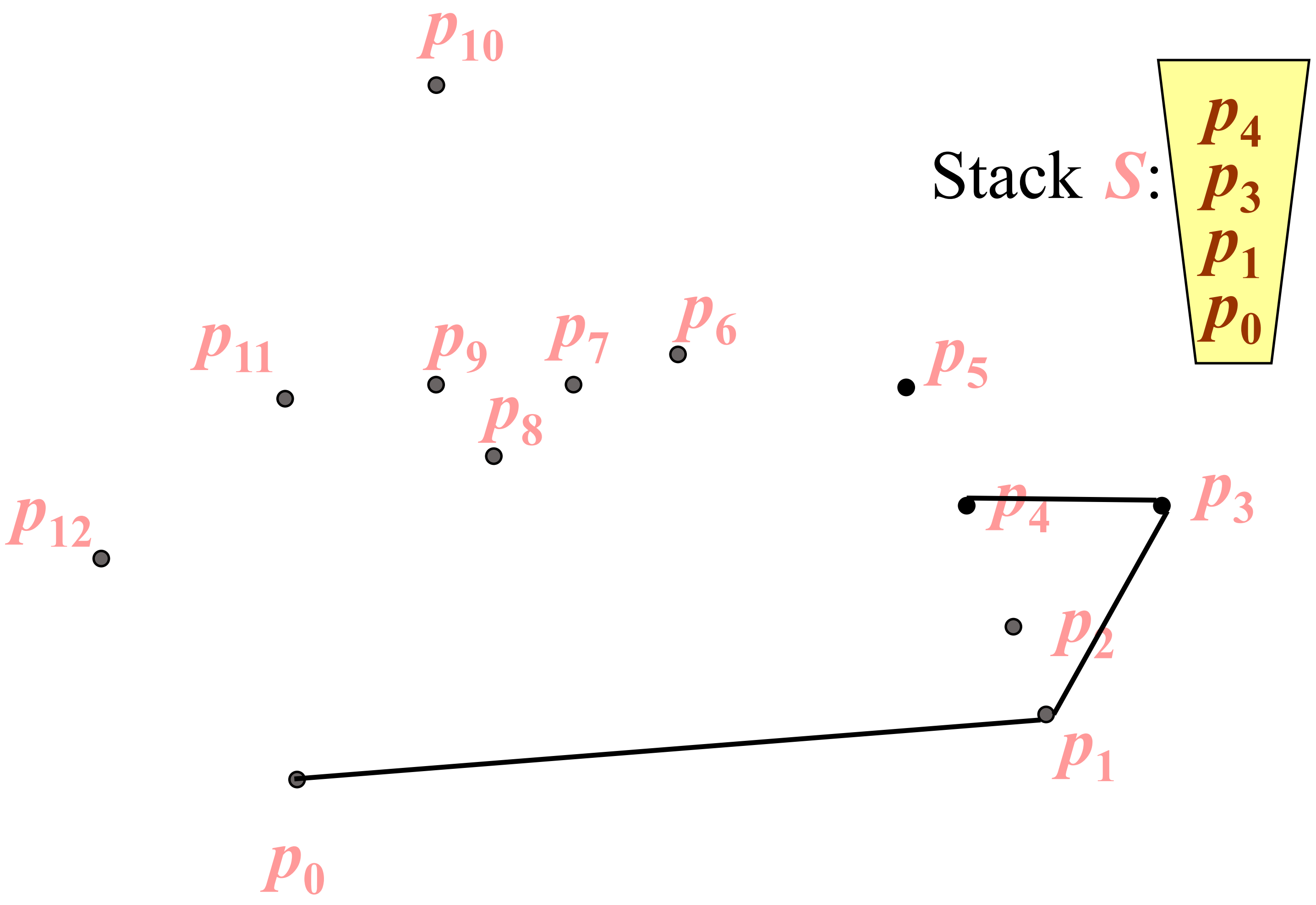
Graham-Scan : (2/11)



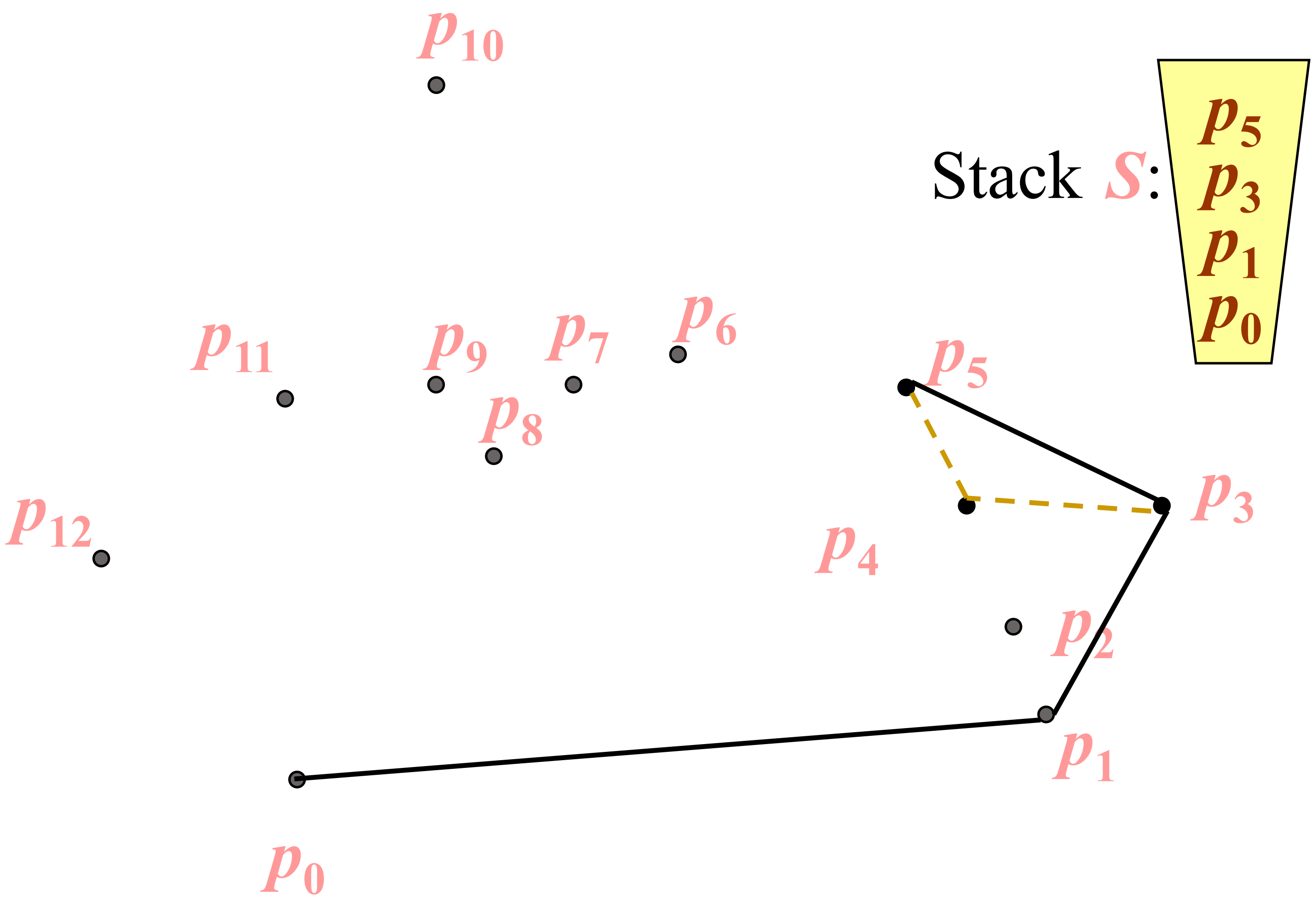
Graham-Scan : (3/11)



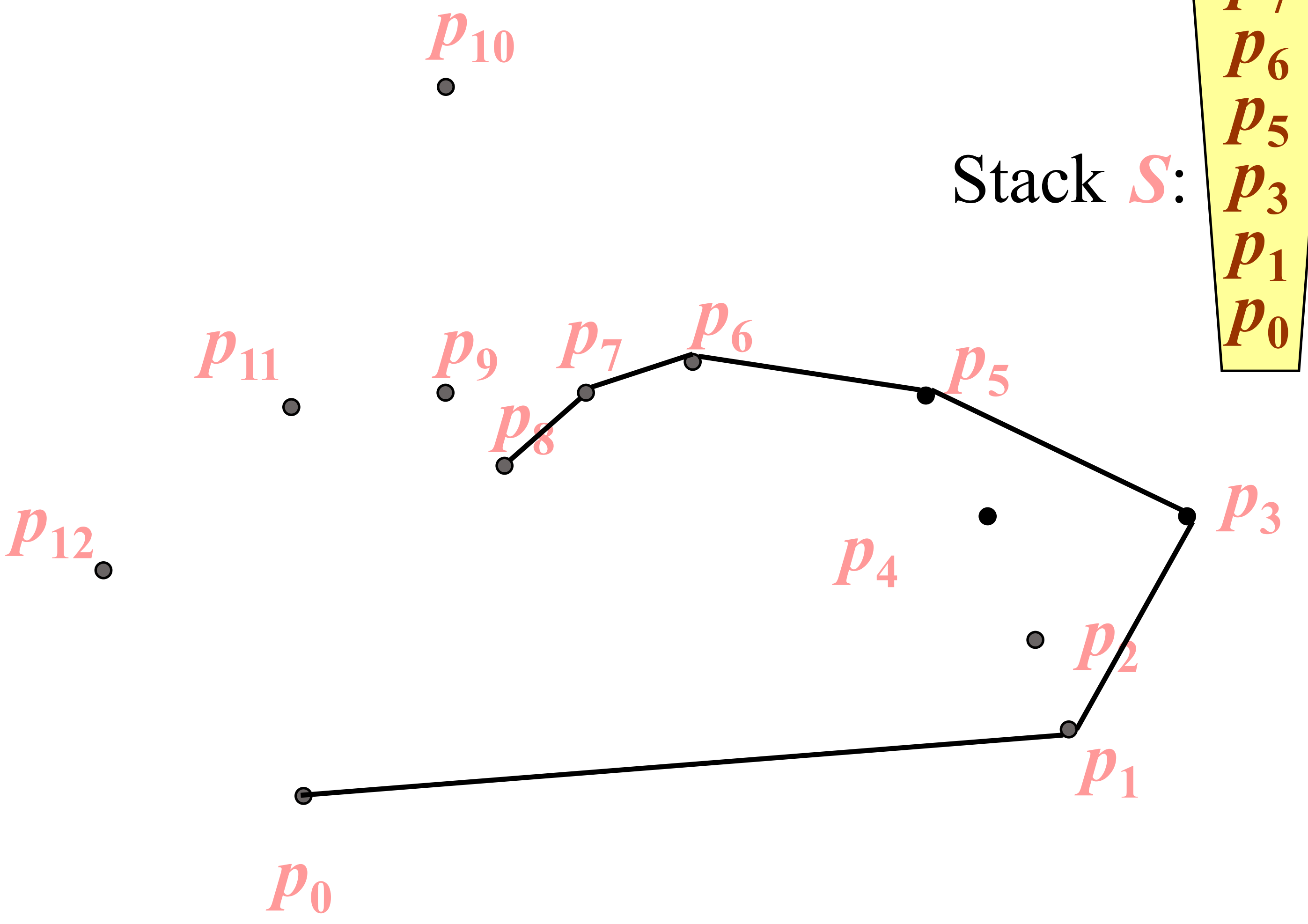
Graham-Scan : (4/11)



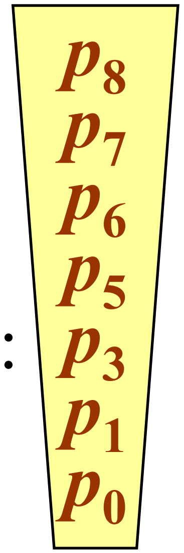
Graham-Scan (5/11)



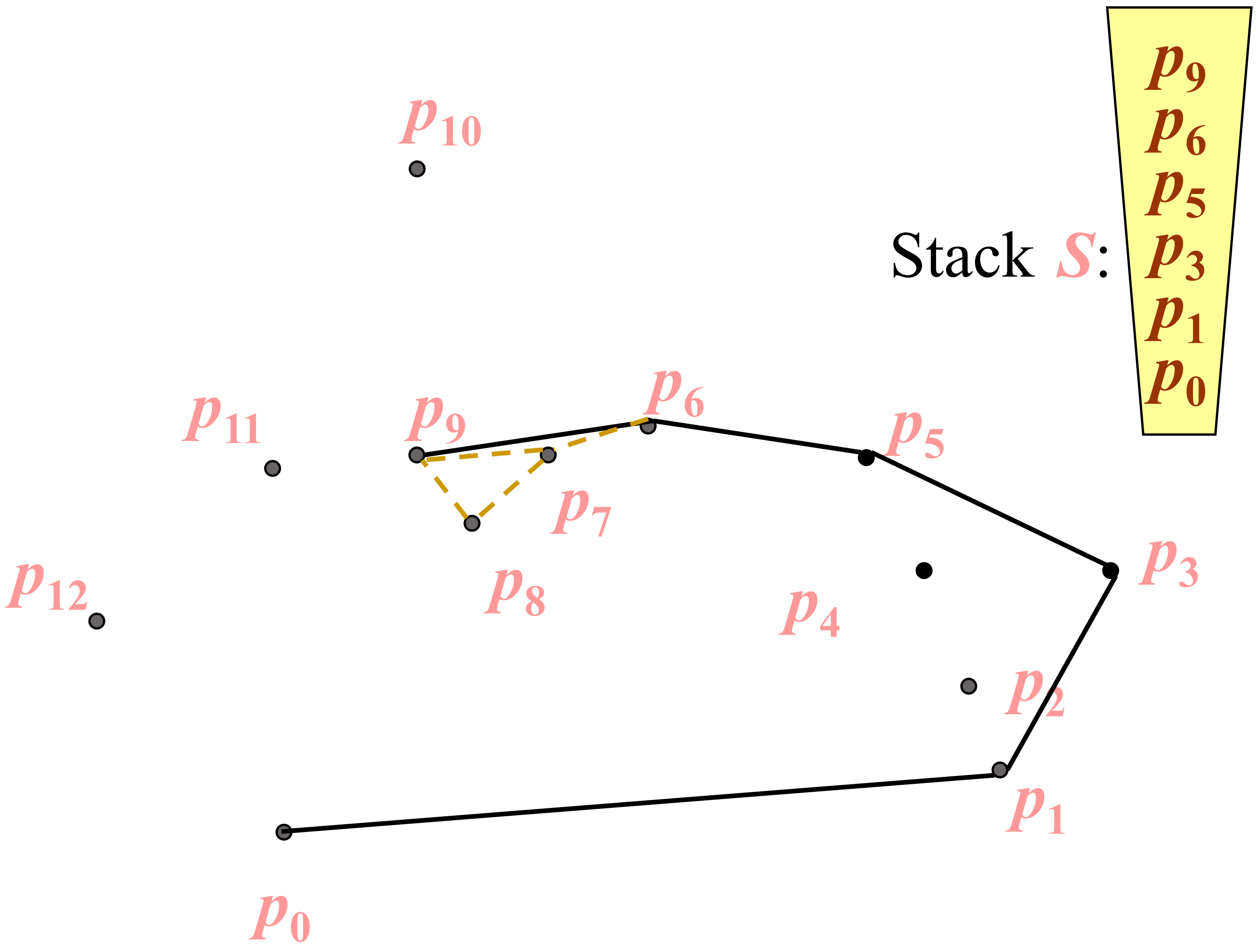
Graham-Scan (6/11)



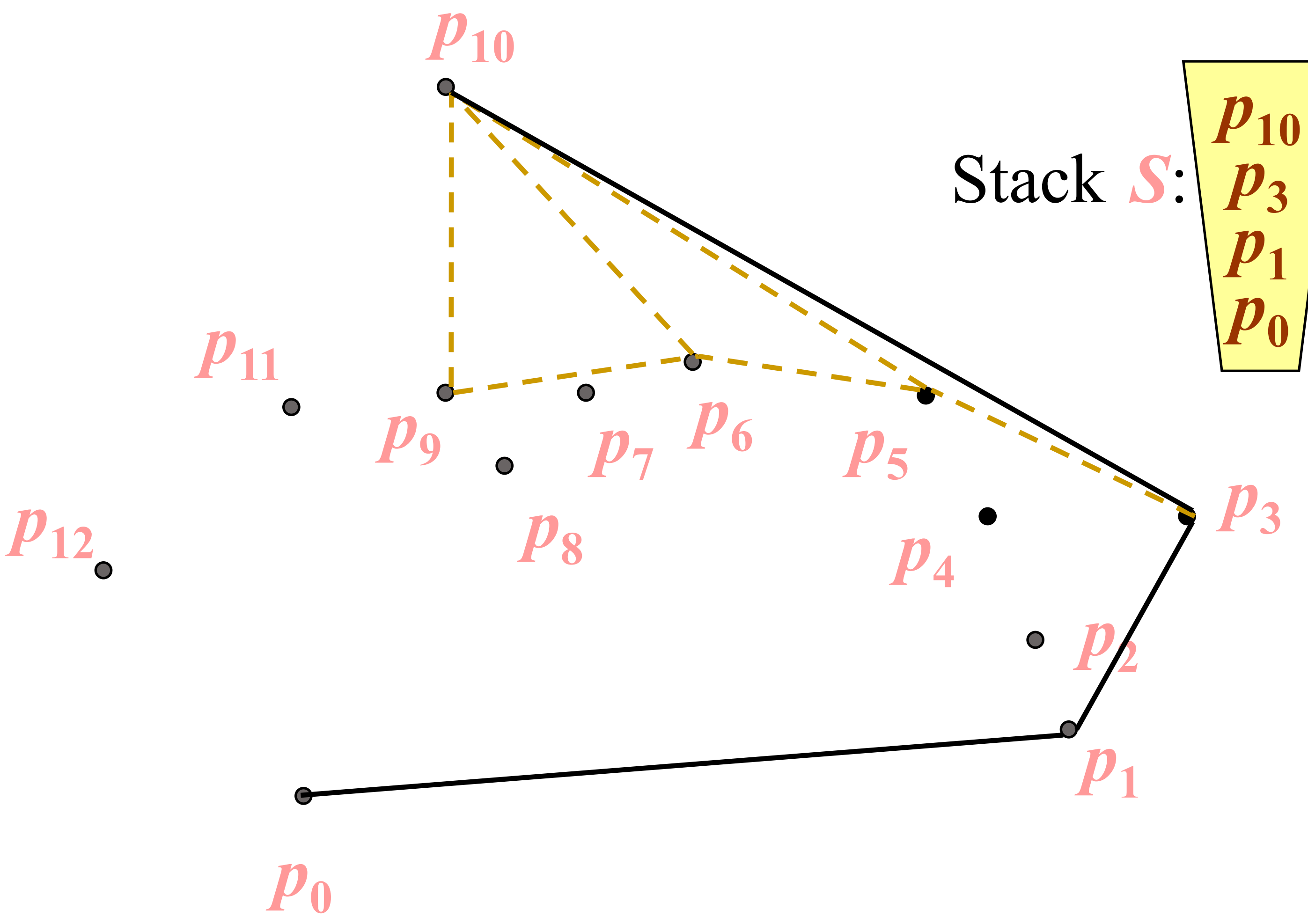
Stack S:



Graham-Scan (7/11)



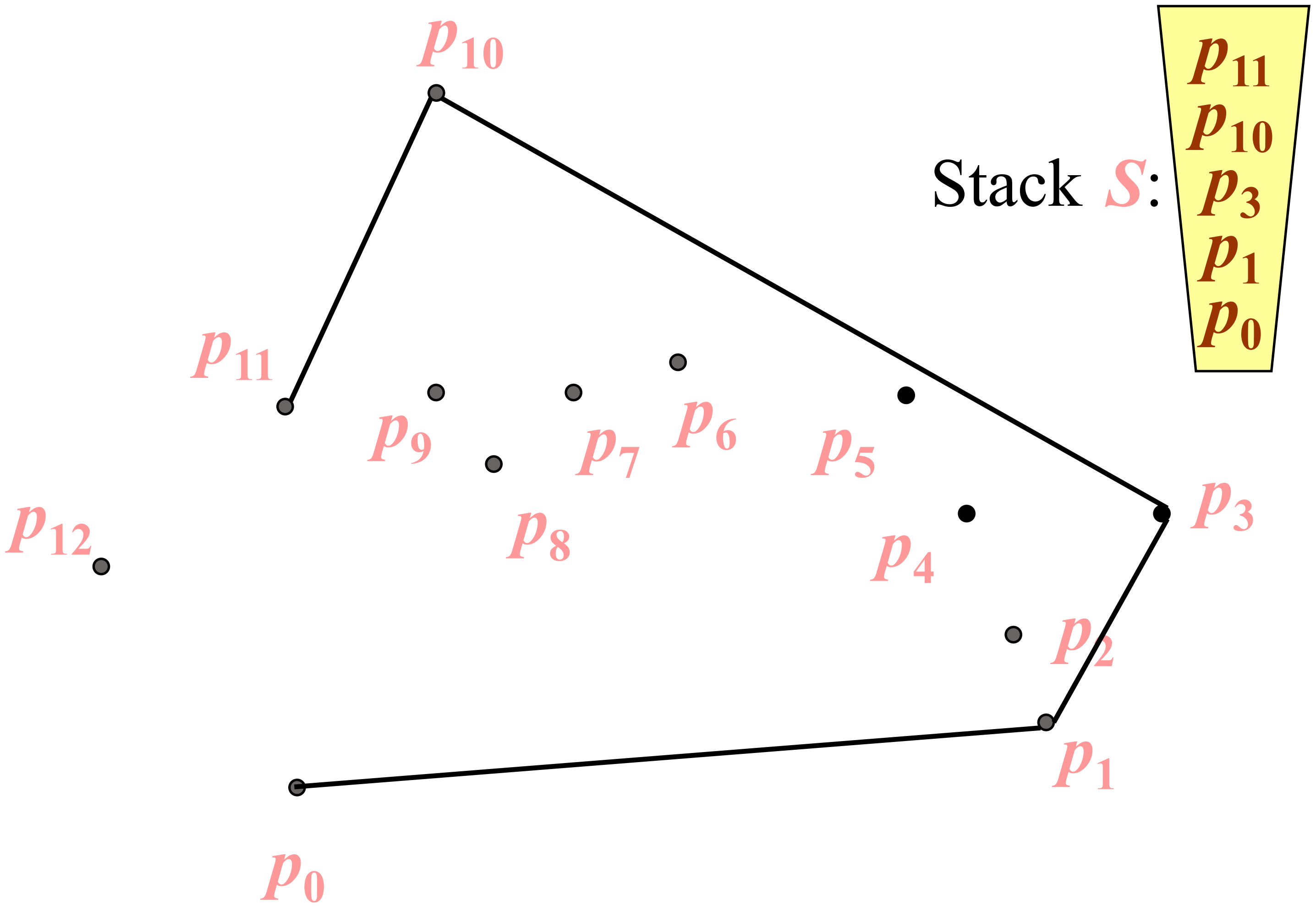
Graham-Scan (8/11)



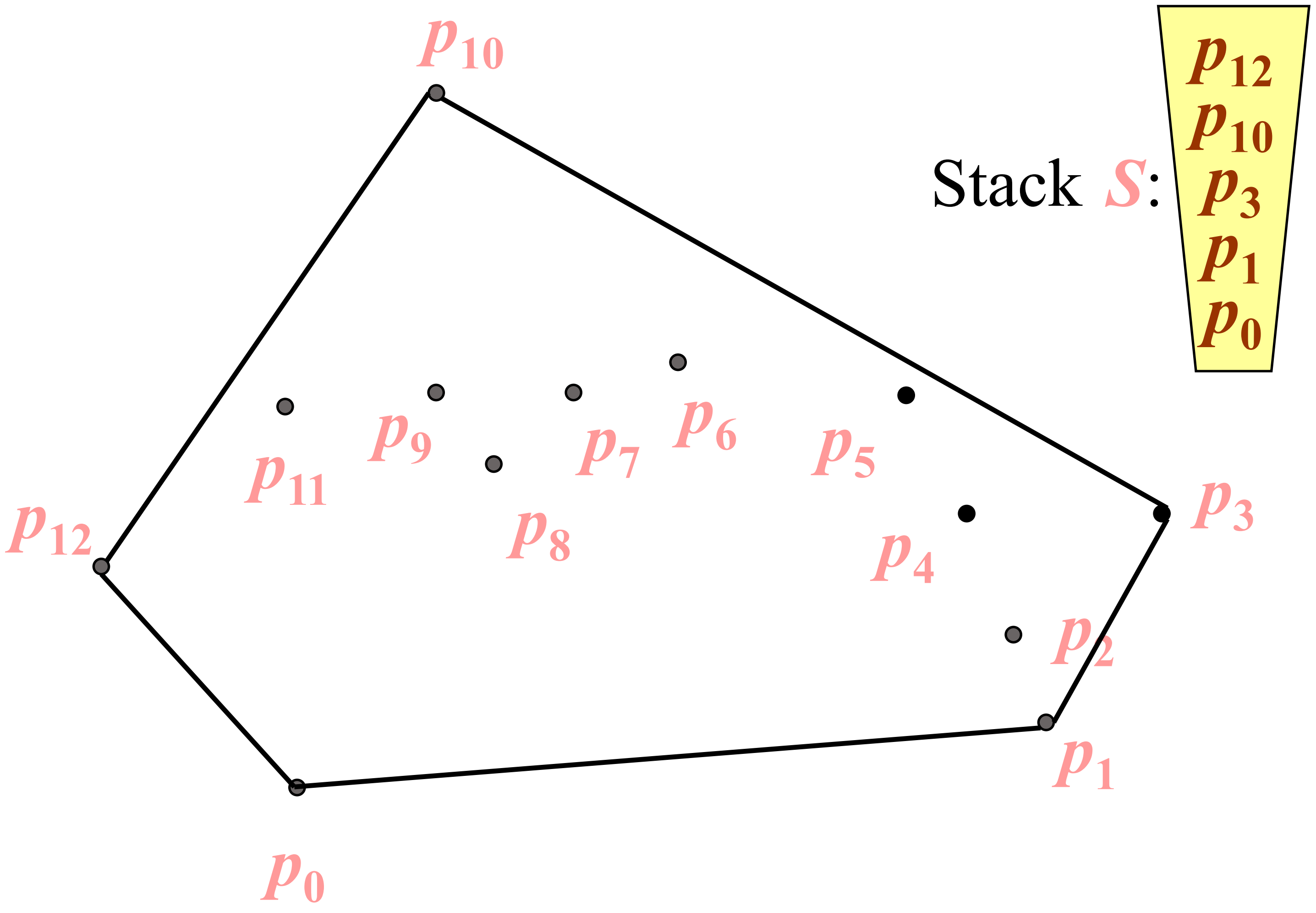
Stack S:

p_{10}
p_3
p_1
p_0

Graham-Scan (9/11)



Graham-Scan (10/11)



Jarvis March

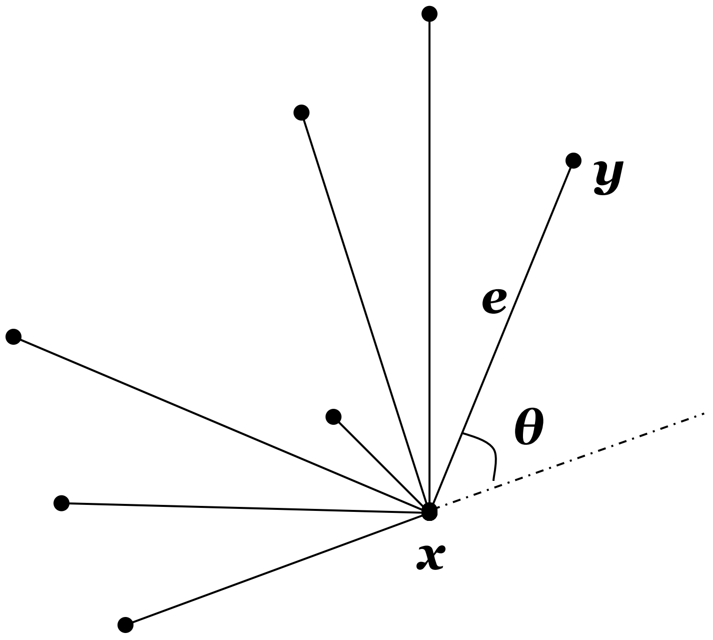
Jarvis March

- Also known as the "gift wrapping" technique.
- Start at some extreme point on the hull.
- In a counterclockwise fashion, each point within the hull is visited and the point that undergoes the largest right-hand turn from the current extreme point is the next point on the hull.
- Finds the points on the hull in the order in which they appear.

Gift Wrapping (a more realistic hull algorithm)

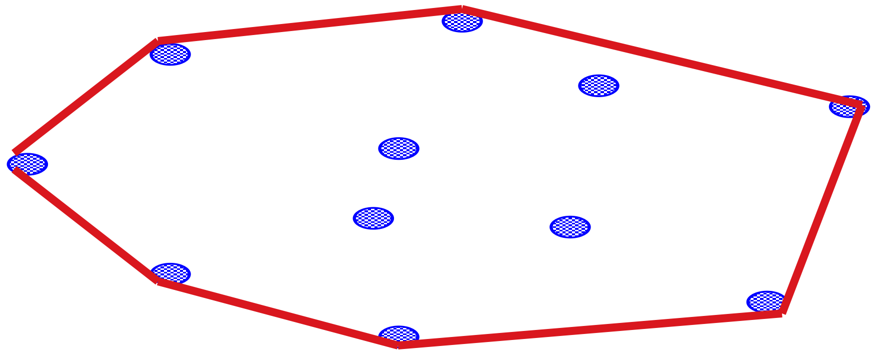
- A minor variation of Extreme Edge algorithm can accelerate it by factor n as well as output the points in order
- The idea is to use one extreme edge as an anchor for finding the next
- Suppose the algorithm found an extreme edge whose unlinked endpoint is x

- For each y of set S we compute the angle θ
- The point that yields the smallest θ must determine an extreme edge
- The output of this algorithm is all the points on the hull in boundary traversal order



Gift wrapping

Idea: Think of wrapping a gift. Put the paper in contact with the gift and continue to wrap around from one surface to the next until you get all the way around.



Algorithm: Gift Wrapping

```
Find the lowest point (smallest y coordinate)
Let $i_0$ be its index, and set $i \leftarrow i_0$
repeat
 for each $j \neq i$ do
    compute counterclockwise angle $\theta$ from previous hull edge
    Let k be the index of the point with the smallest $\theta$
    Output $(p_i, p_k)$ as a hull edge
    $i \leftarrow k$
until $i = i_0$
```

- We use the lowest point as the anchor
- The order is $O(n^2)$
- The cost is $O(n)$ for each hull edge
- The point set is wrapped by a string that bends the that bends with minimum angle from previous to next hull edge

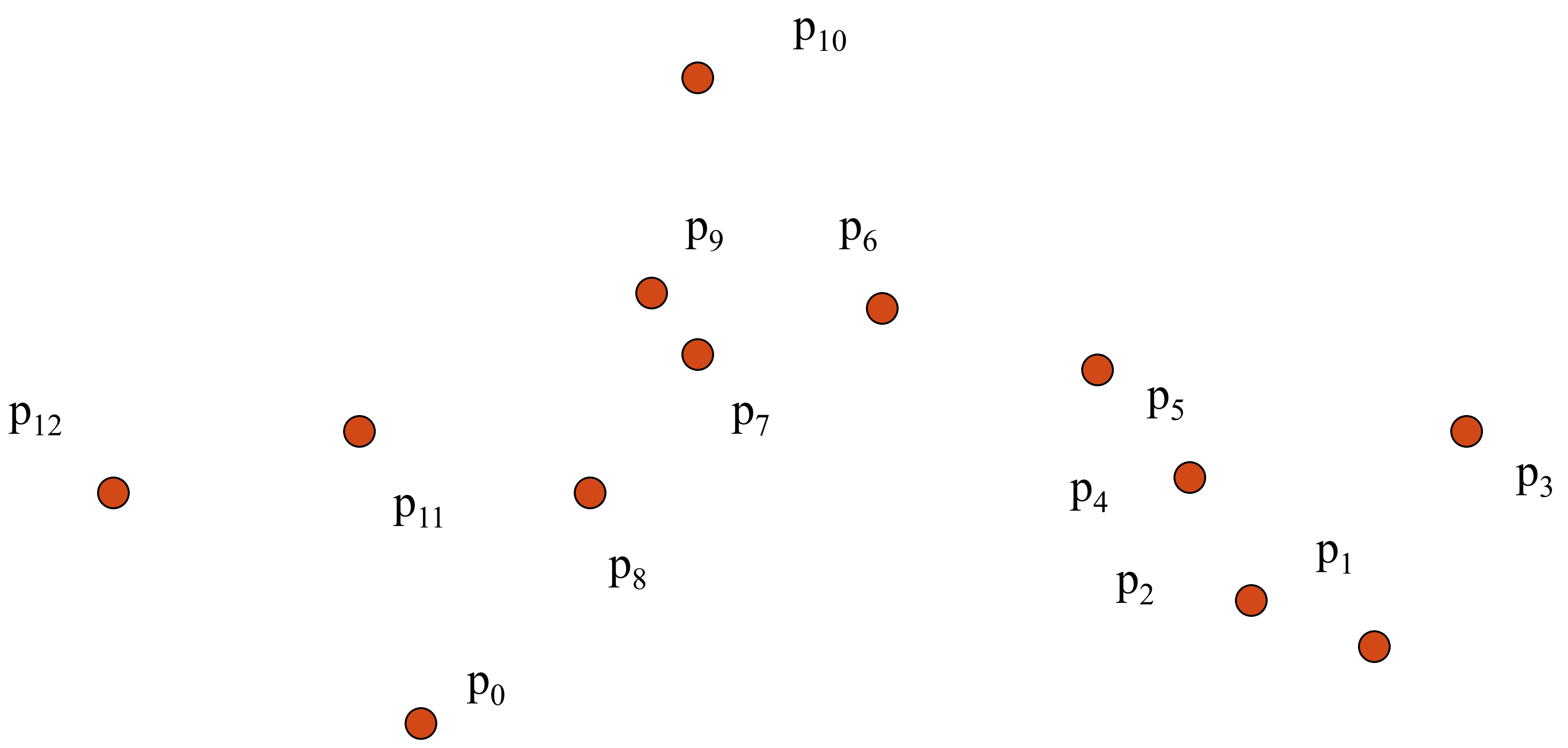
Implementation

- In order to find the biggest right hand turn, the last two points added to the hull, p1 and p2, must be known.
- Assuming p1 and p2 are known, the angle these two points form with the remaining possible hull points is calculated. The point that maximizes the angle is the next point in the hull.
- To do this, we have two line segments p0p1 and p1p2. We are trying to determine the value of the right hand turn at p1. To do this we find the cross product of (p1 – p0) x (p2 – p0). The point that gives the largest value in this calculation is the next point on the hull.
- Continue to make right turns until the original point is reached and the hull is complete.

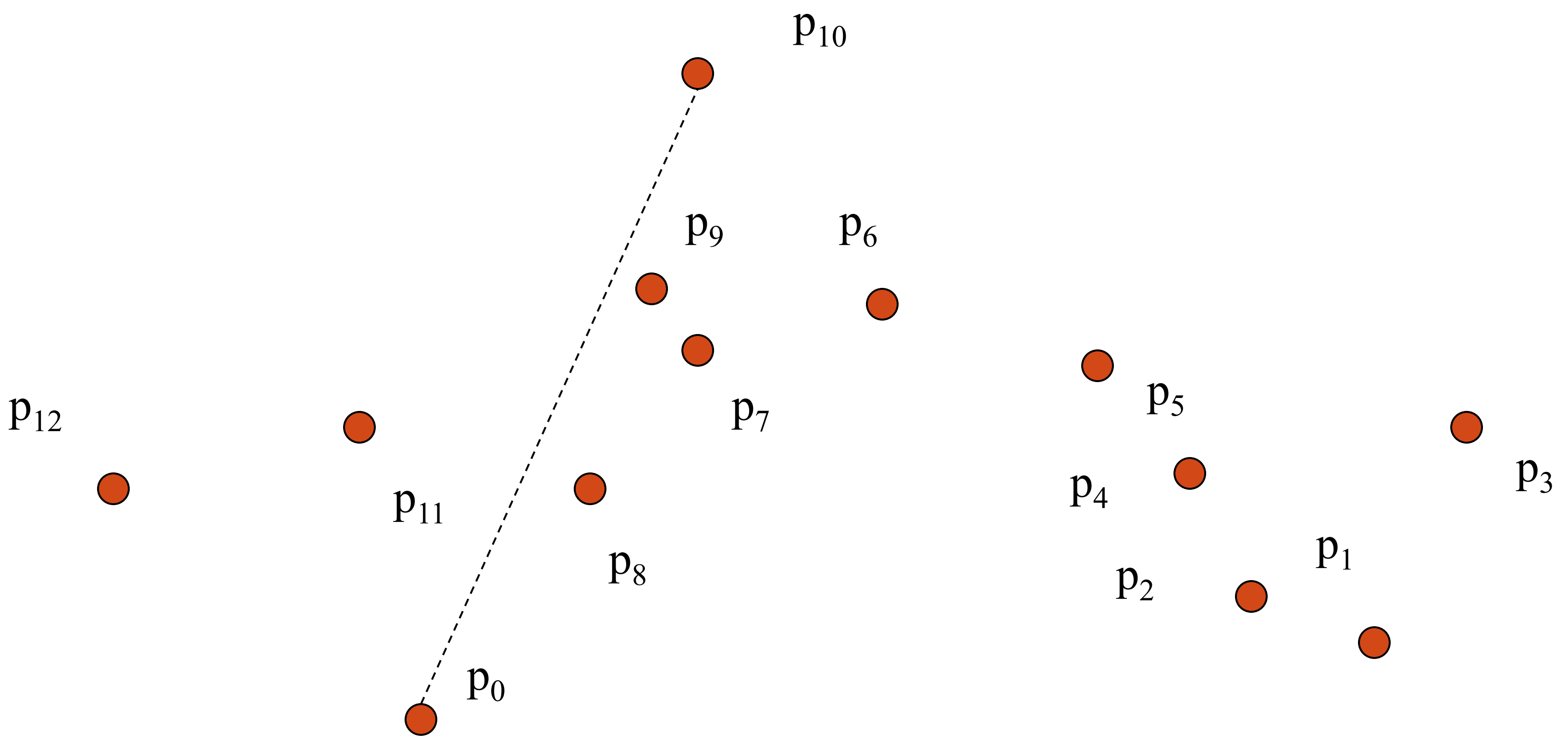
Efficiency

- Proposed by R.A. Jarvis in 1973
- $\boldsymbol{O(nh)}$ complexity, with n being the total number of points in the set, and $\boldsymbol{h}$ being the number of points that lie in the convex hull.
- This implies that the time complexity for this algorithm is the number of points in the set multiplied by the number of points in the hull
- The worst case for this algorithm is denoted by $\boldsymbol{O(n^2)}$, which is not optimal.
- Favorable conditions in which to use the Jarvis march include problems with a very low number of total points, or a low number of points on the convex hull in relation to the total number of points.

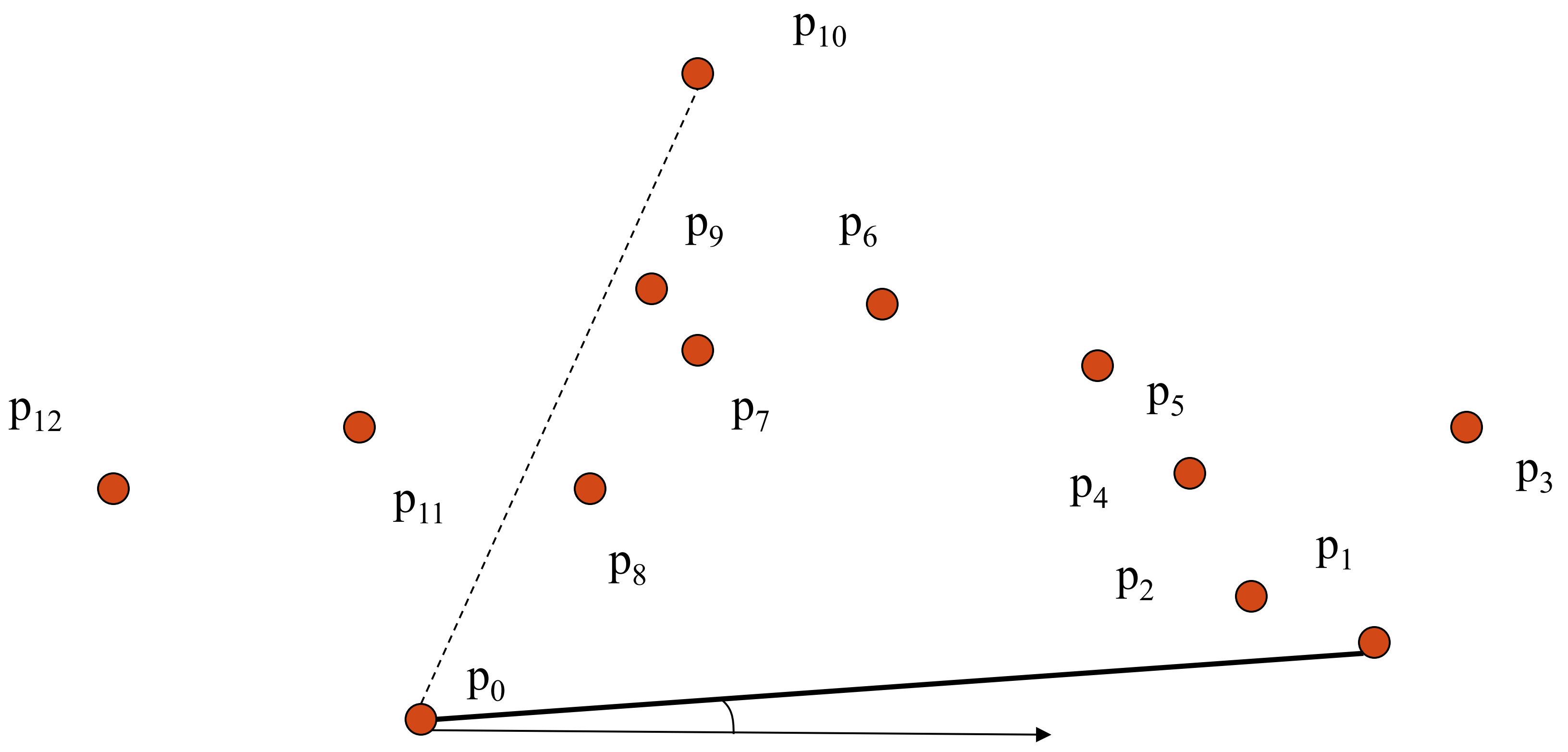
Jarvis March - Example



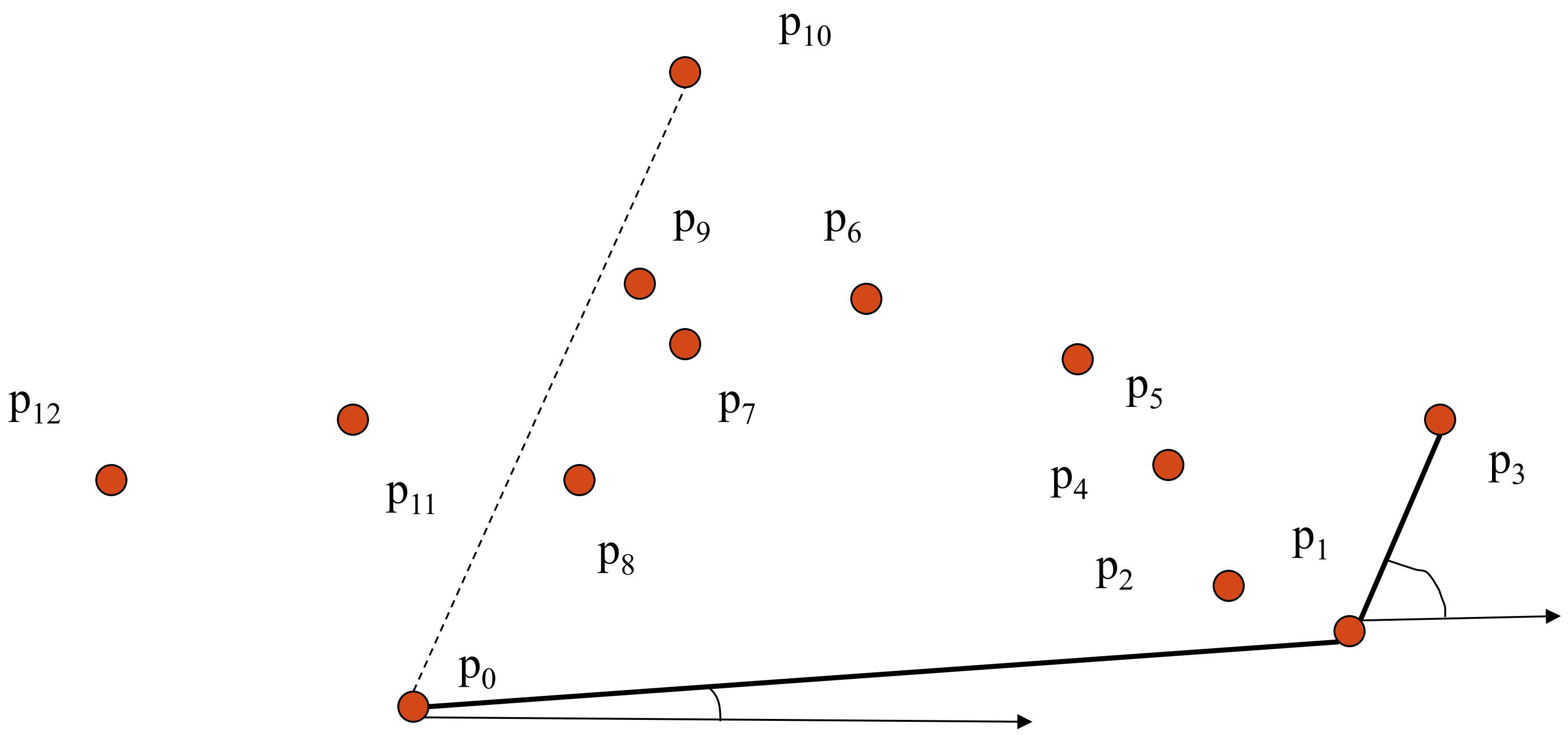
Jarvis March - Example



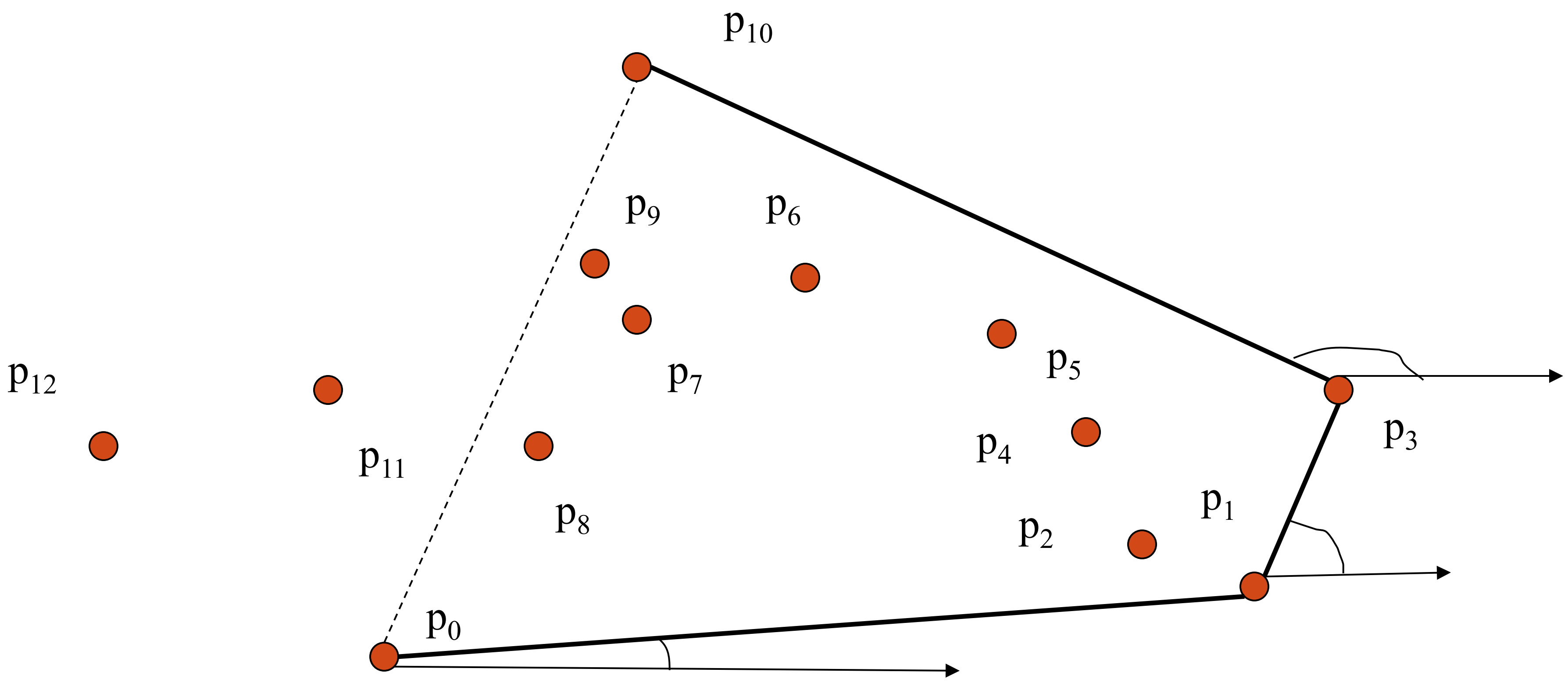
Jarvis March - Example



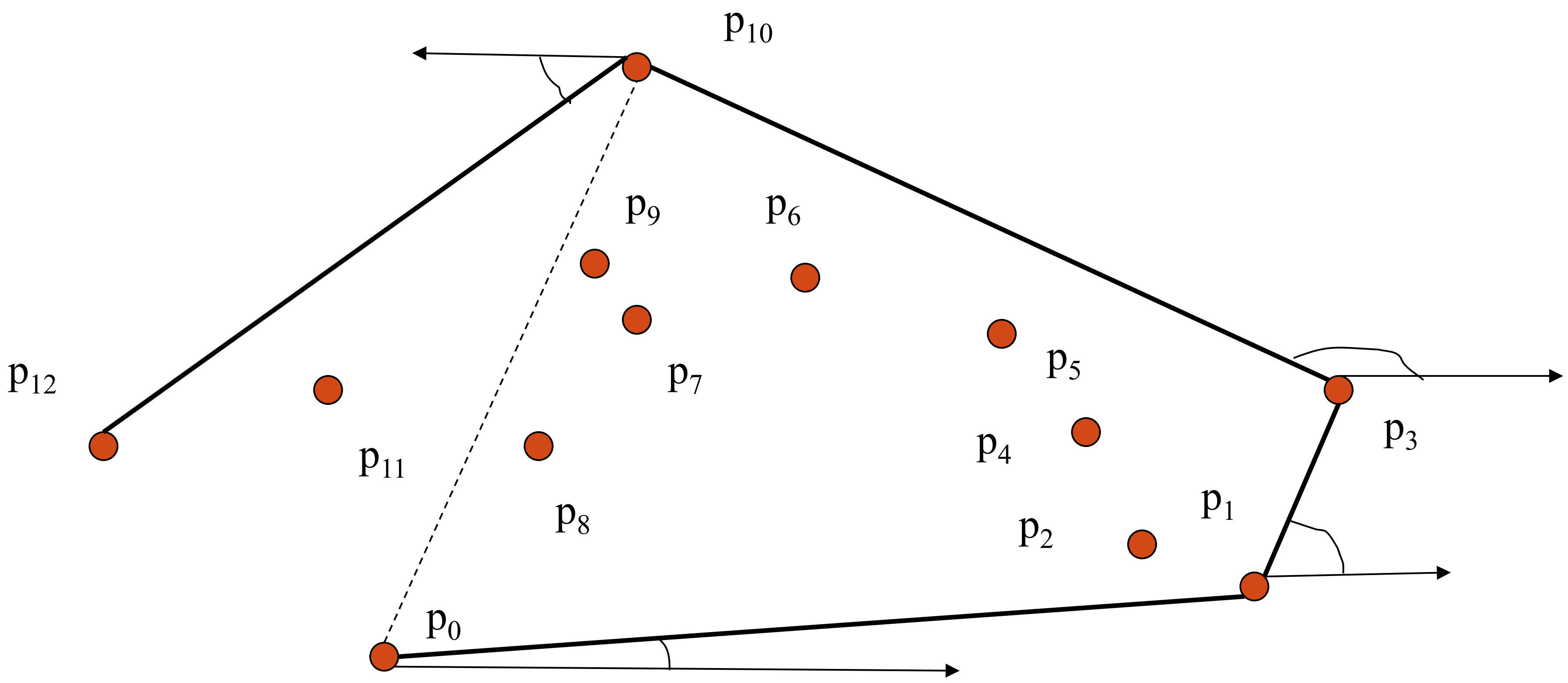
Jarvis March - Example



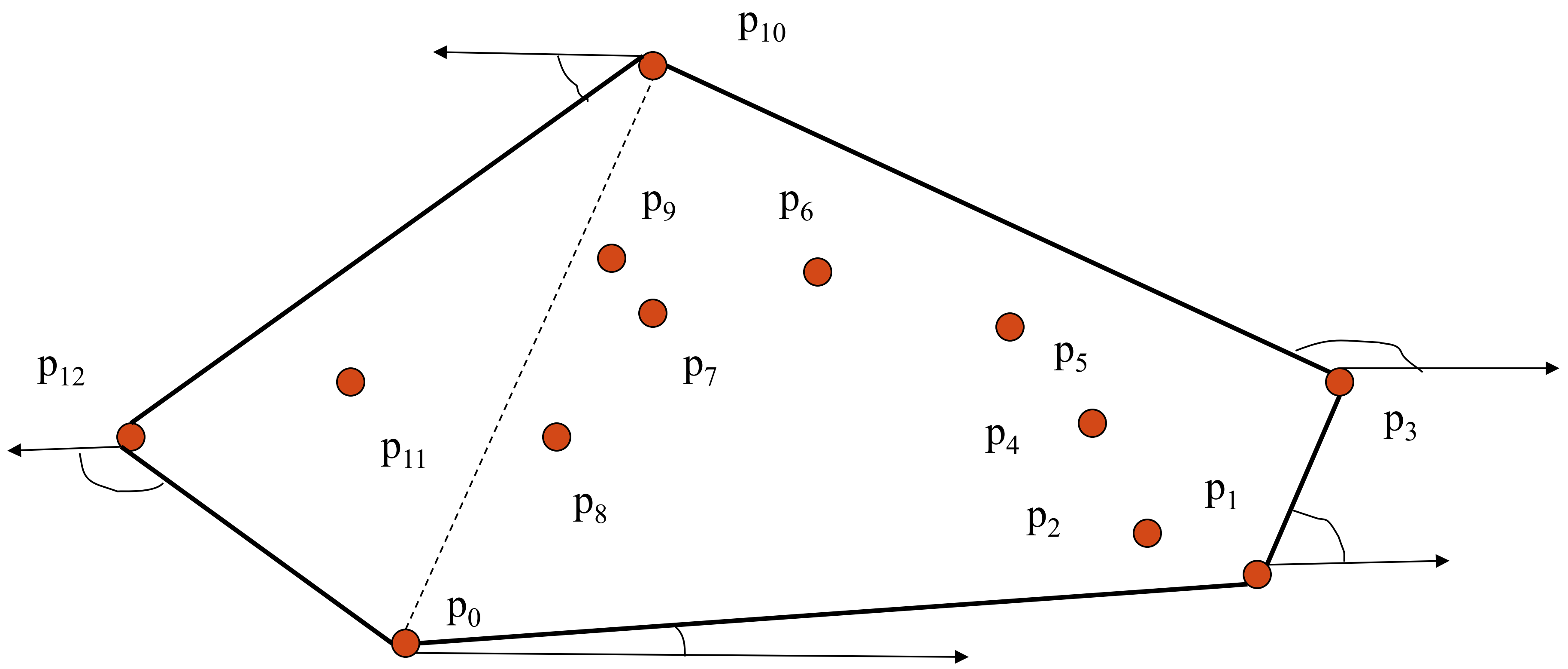
Jarvis March - Example



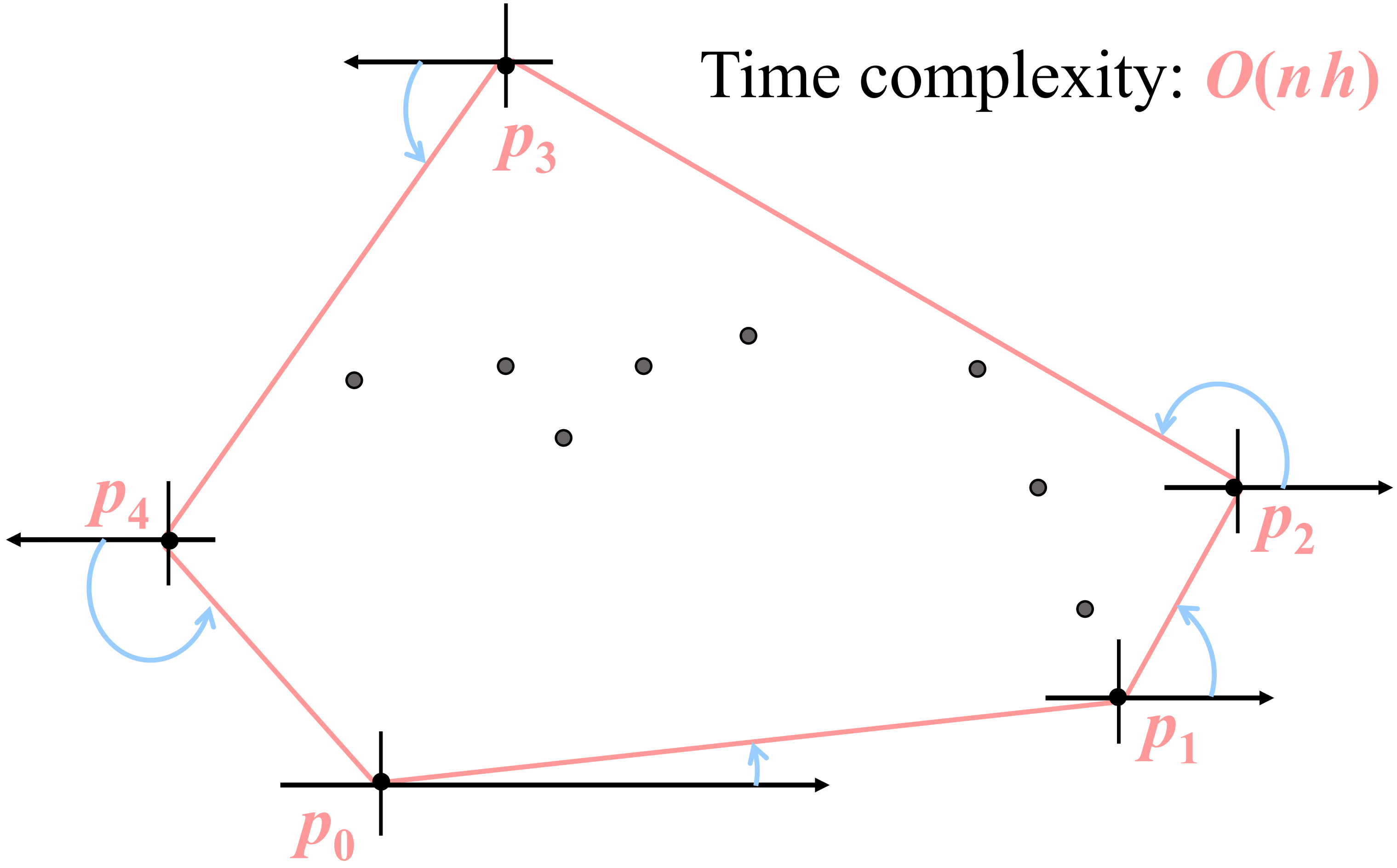
Jarvis March - Example



Jarvis March - Example



Jarvis's march for finding the Convex Hull



Time complexity: $O(nh)$

Efficiency

Proposed by R.A. Jarvis in 1973

- $\boldsymbol{O(nh)}$ complexity, with n being the total number of points in the set, and h being the number of points that lie in the convex hull.
- This implies that the time complexity for this algorithm is the number of points in the set multiplied by the number of points in the hull
- The worst case for this algorithm is denoted by $\boldsymbol{O(n^2)}$, which is not optimal.
- Favorable conditions in which to use the Jarvis march include problems with a very low number of total points, or a low number of points on the convex hull in relation to the total number of points.

Time complexity Analysis

- **Graham-Scan**
  - Sorting in step 2 needs $O(n \log n)$.
  - Time complexity of stack operation is $O(2n)$
  - The overall time complexity in **Graham-Scan** is $O(n \log n)$.
- **Jarvis's march**
  - h vertices
  - Finding smallest polar angle point cost O(n)
  - Overall time complexity: O(nh)

References

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (n.d.). Introduction to algorithms (3rd ed.). The MIT Press.
- Erickson, J. (n.d.). Convex Hulls. Lecture. Retrieved August 23, 2018, from http://jeffe.cs.illinois.edu/teaching/373/notes/x05-convexhull.pdf
- Mount, D. M. (n.d.). CMSC 754 Computational Geometry. Lecture Retrieved August 23, 2018, from https://www.cs.umd.edu/class/spring2012/cmsc754/Lects/cmsc754-lects.pdf

Excersis

1. Write an algorithm in pseudocode that implements QuickHull and test it using suitable data.

2. Code the divide-and-conquer algorithm DCHull and test it using appropriate data.

3. Run the three algorithms for convex hull discussed in this section on various random inputs and compare their performances.

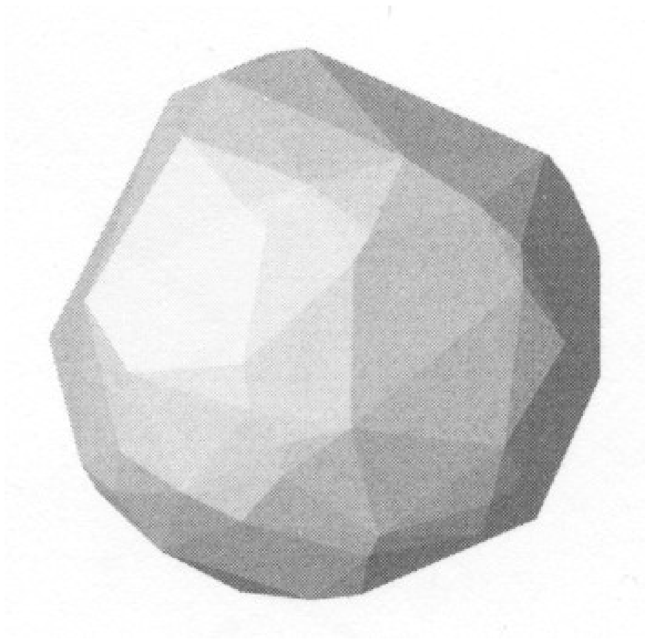
4. Algorithm DCHull can be modified as follows: Instead of using the median as the splitter, we could use a randomly chosen point as the splitter. Then X is partitioned into two around this point. The rest of the function DCHull is the same. Write code for this modified algorithm and compare it with DCHull empirically.

5. Let S be a set of n points in the plane. It is given that there is only a constant (say c) number of points on the hull of S. Can you devise a convex hull algorithm for S that runs in time $o(n \log n)$? Conceive of special algorithms for $c = 3$ and $c = 4$ first and then generalize.

Applications

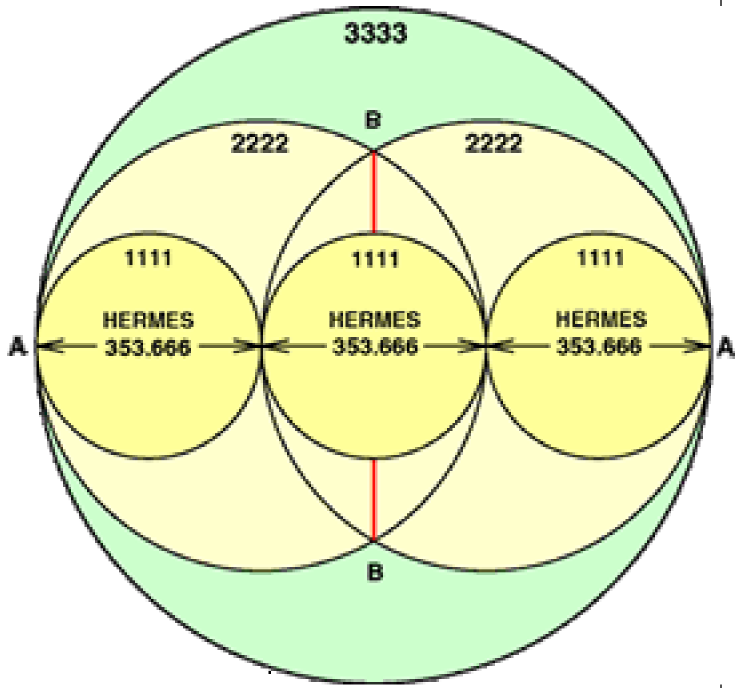
Convex Hulls in 3-space Problem Statement

- Given *P*: set of *n* points in 3-space
- Return:
  - Convex hull of P: CH(*P*)
  - Smallest polyhedron such that all elements of *P* on or in the interior of CH(*P*).



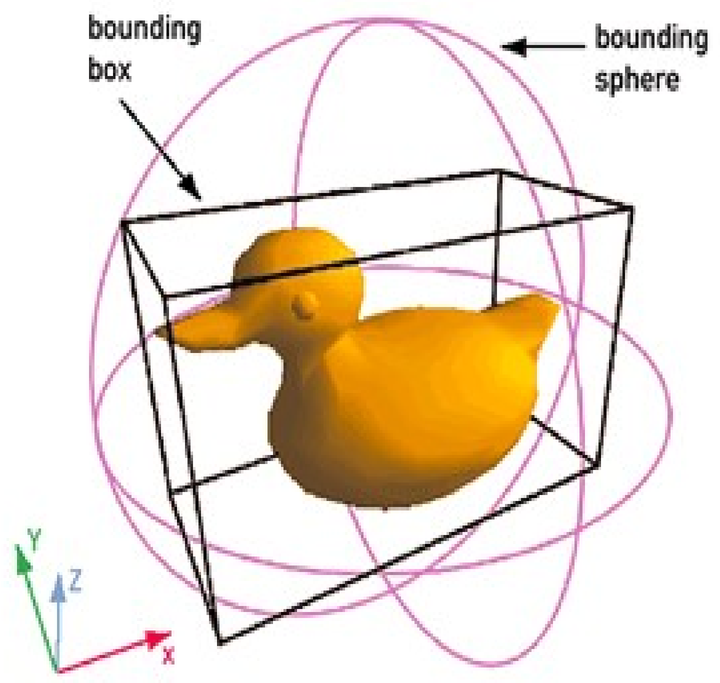
Geometry - Diameter Computation

The problem of finding the diameter of a set of points, which is the pair of points a maximum distance apart. The diameter will always be the distance between two points on the convex hull. The *O(nlogn)* algorithm for computing diameter proceeds by first constructing the convex hull, then for each hull vertex finding which other hull vertex is farthest away from it. This so-called "rotating-calipers" method can be used to move efficiently from one hull vertex to another.



Computer Visualization, Ray Tracing, Video Games, Replacement of Bounding Boxes

Bounding boxes are used to an approximate location of an object in question and as a very simple descriptor of its shape. For example, in computational geometry and its applications when it is required to find intersections in the set of objects, the initial check is the intersections between their bounding boxes. Since it is usually a much less expensive operation than the check of the actual intersection (because it only requires comparisons of coordinates), it allows to quickly exclude from checks the pairs that are far apart. Convex hull can be a candidate for replacement of bounding boxes in these situations.



Path Finding - Embedded AI of Mars mission Rovers

The ongoing unmanned Mars exploration mission, commenced in 2003 sent two robotic rovers, Spirit and Opportunity, to explore the Martian surface and geology. The mission was led by Project Manager Peter Theisinger of NASA's Jet Propulsion Laboratory and Principal Investigator Steven Squyres, professor of astronomy at Cornell University

Path Finding - Embedded AI of Mars mission Rovers

Primary among the mission's scientific goals is to search for and characterize a wide range of rocks and soils that hold clues to past water activity on Mars. In recognition of the vast amount of scientific information amassed by both rovers, two asteroids have been named in their honor: 37452 Spirit and 39382 Opportunity

Visual Pattern Matching - Detecting Car License Plates

Pattern recognition aims to classify data (patterns) based on either a priori knowledge or on statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements or observations, defining points in an appropriate multidimensional space. This is in contrast to pattern matching, where the pattern is rigidly specified.

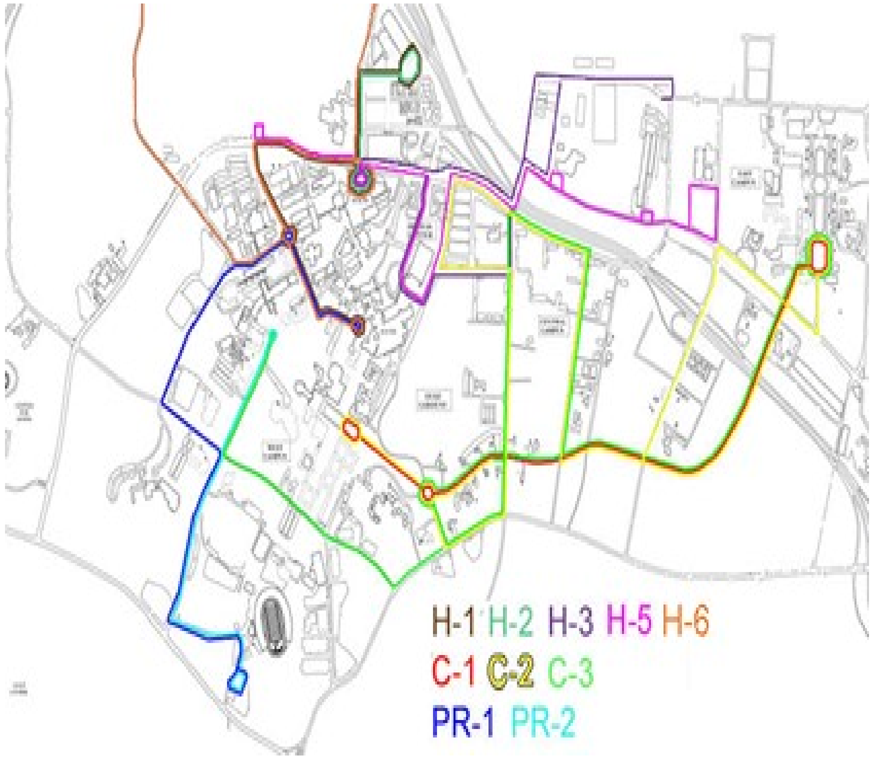
Within medical science pattern recognition creates the basis for CAD Systems (Computer Aided Diagnosis). CAD describes a procedure that supports the doctor's interpretations and findings.

Visual Pattern Matching - Detecting Car License Plates

A complete pattern recognition system consists of a sensor that gathers the observations to be classified or described; a feature extraction mechanism that computes numeric or symbolic information from the observations; and a classification or description scheme that does the actual job of classifying or describing observations, relying on the extracted features.

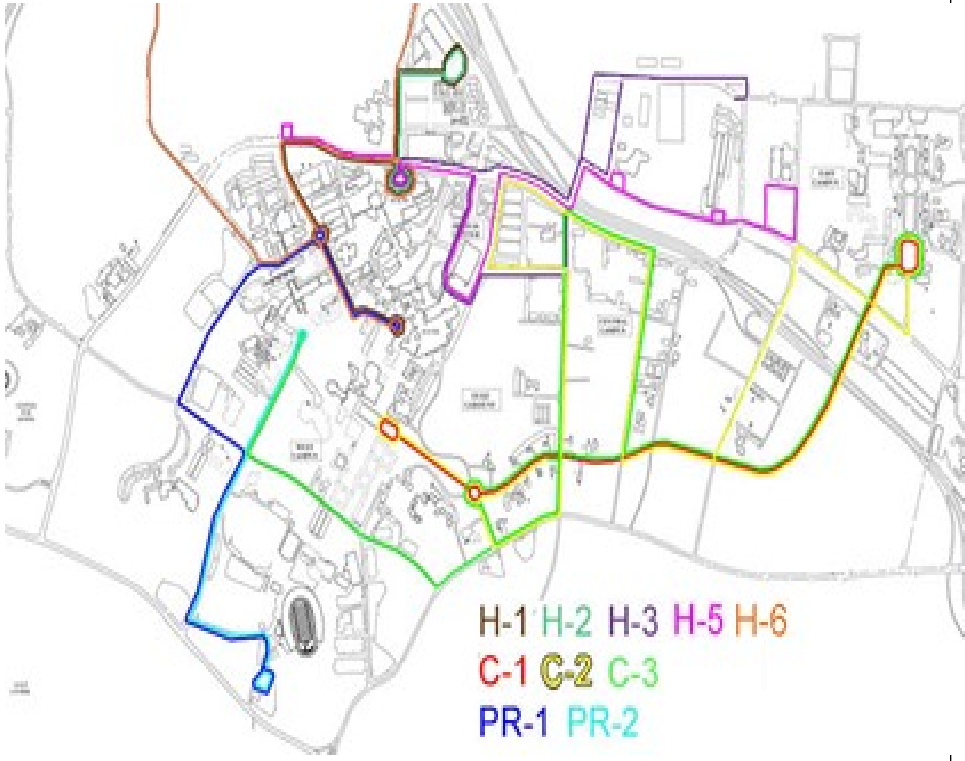
Geographical Information Systems (GIS) - Computing Accessibility Maps

A geographic information system (GIS), also known as a geographical information system or geospatial information system, is a system for capturing, storing, analyzing and managing data and associated attributes which are spatially referenced to the Earth.



Geographical Information Systems (GIS) - Computing Accessibility Maps

In the strictest sense, it is an information system capable of integrating, storing, editing, analyzing, sharing, and displaying geographically-referenced information. In a more generic sense, GIS is a tool that allows users to create interactive queries (user created searches), analyze the spatial information, edit data, maps, and present the results of all these operations. Geographic information science is the science underlying the geographic concepts, applications and systems, taught in degree and GIS Certificate programs at many universities.



Geographical Information Systems (GIS) - Computing Accessibility Maps

Geographic information system technology can be used for scientific investigations, resource management, asset management, Environmental Impact Assessment, Urban planning, cartography, criminology, history, sales, marketing, and logistics. For example, GIS might allow emergency planners to easily calculate emergency response times in the event of a natural disaster, GIS might be used to find wetlands that need protection from pollution, or GIS can be used by a company to site a new business to take advantage of a previously underserved market.

Convex Position Estimation in Wireless Sensor Networks

- Tracking through sensor network
- Hierarchical solution for large networks
- Implementing continuous distributions
- Combination of angular and radial constraints
- Erroneous data management
- Modeling uncertainty in "known" positions

Wireless Sensor Network for Structural Monitoring

Underground Structure Monitoring with Wireless Sensor Networks [Structure-Aware Self-Adaptive sensor system (SASA) in mines]

- Detecting and locating the collapse hole
- Accident reporting
- Displaced node detection and reconfiguration

- When edge nodes detect a hole, they report to the sink with the locations so that the hole can be illustrated by calculating the convex hull.

More applications

- **Collision Avoidance**:
  - If convex hull of a robot avoids collision with the obstacles, then so does the robot
  - Computation of path with convex robot is much simpler with convex robot then the non convex one
- **Smallest box:**
  - Smallest three dimensional box surrounding an object in space depends crucially on the convex hull of the object

CVX: MATLAB Software for Disciplined Convex Programming

CVX: MATLAB Software for Disciplined Convex Programming

- http://cvxr.com/cvx/
- CVX is a Matlab-based modeling system for convex optimization. CVX turns Matlab into a modeling language, allowing constraints and objectives to be specified using standard Matlab expression syntax. For example, the following code segment randomly generates a constrained norm minimization problem, and solves it:

convhull - Convex hull

Syntax

- K = convhull(X,Y)
  K = convhull(X,Y,Z)
  K = convhull(X)
  K = convhull(...,'simplify', logicalvar)
  [K,V] = convhull(...)

Definition :convhull returns the convex hull of a set of points in 2-D or 3-D space.

Description

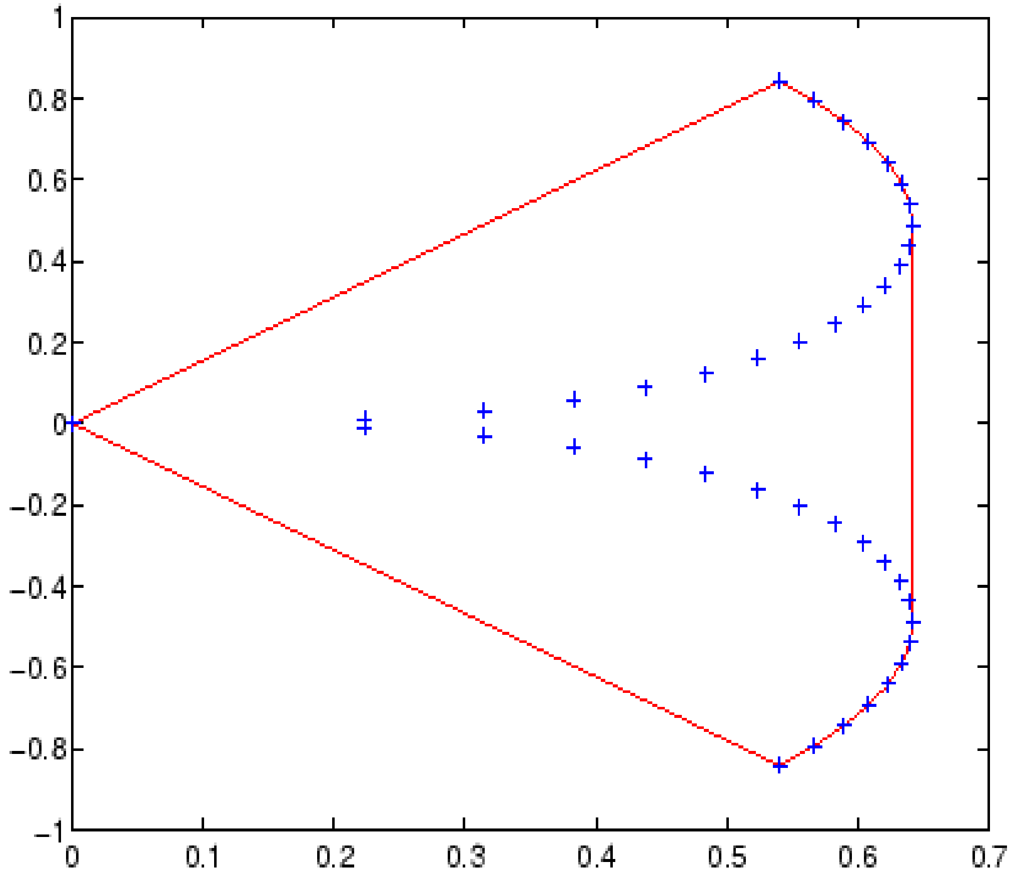
- K = convhull(X,Y) returns the 2-D convex hull of the points (X,Y), where X and Y are column vectors. The convex hull K is expressed in terms of a vector of point indices arranged in a counterclockwise cycle around the hull.
- K = convhull(X,Y,Z) returns the 3-D convex hull of the points (X,Y,Z), where X, Y, and Z are column vectors. K is a triangulation representing the boundary of the convex hull. K is of size mtri-by-3, where mtri is the number of triangular facets. That is, each row of K is a triangle defined in terms of the point indices.

convhull - Convex hull

- K = convhull(X) returns the 2-D or 3-D convex hull of the points X. This variant supports the definition of points in matrix format. X is of size mpts-by-ndim, where mpts is the number of points and ndim is the dimension of the space where the points reside, $2 \leqq \text{ndim} \leqq 3$. The output facets are equivalent to those generated by the 2-input or 3-input calling syntax.
- K = convhull(...,'simplify', logical var) provides the option of removing vertices that do not contribute to the area/volume of the convex hull, the default is false. Setting 'simplify' to true returns the topology in a more concise form.
- [K,V] = convhull(...) returns the convex hull K and the corresponding area/volume V bounded by K.

convhull : MATLAB function

- xx = -1:.05:1;
- yy = abs(sqrt(xx));
- [x,y] = pol2cart(xx,yy);
- k = convhull(x,y);
- plot(x(k),y(k),'r-',x,y,'b+')



Exercise

1. Write the full, unambiguous pseudo-code for your divide-and-conquer algorithm for finding the convex hull of a set of points Q. Be sure to label the parts of your algorithm. Also, label each part with its worst-case time efficiency.
2. Analyze the whole algorithm for its worst-case time efficiency. State the Big-O asymptotic bound. Include the recurrence relation.