Convex Hull

CS-332

Prof. Bibhudatta Sahoo

NIT Rourkela, INDIA

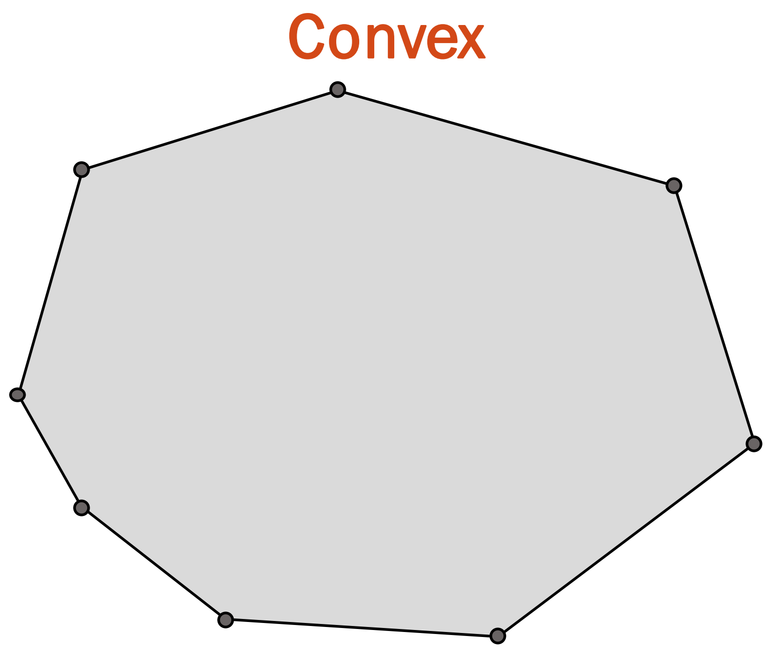
Convex Hull

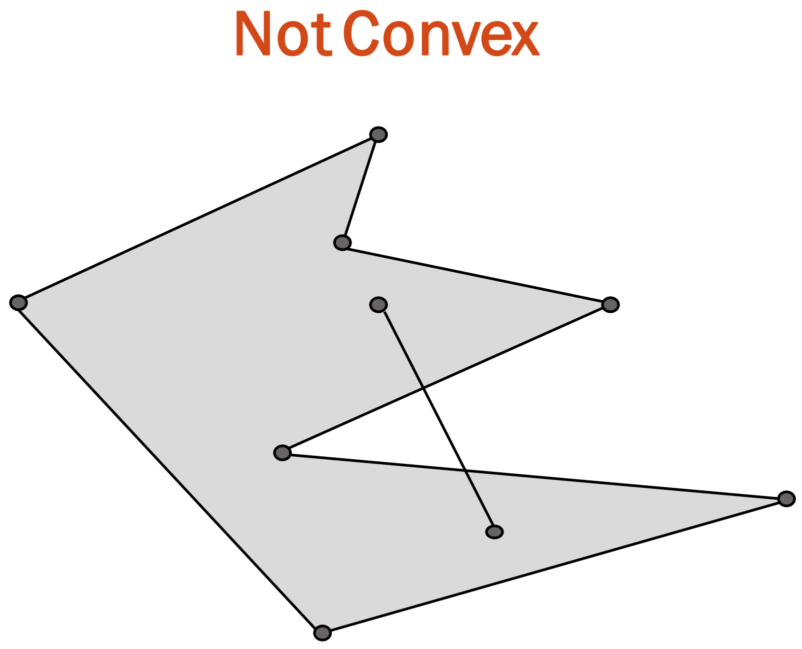
- most ubiquitous structure in computational geometry
- useful to construct other structures
- many applications: robot motion planning, shape analysis etc.
- a beautiful object, one of the early success stories in computational geometry that sparked interest among Computer Scientists by the invention 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

- 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.

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.

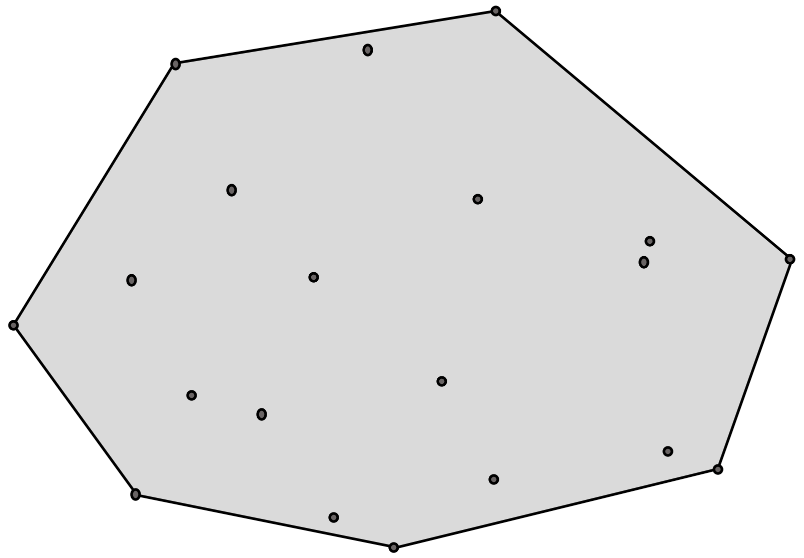
Basic Concepts

- 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



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.

Graham scan algorithm

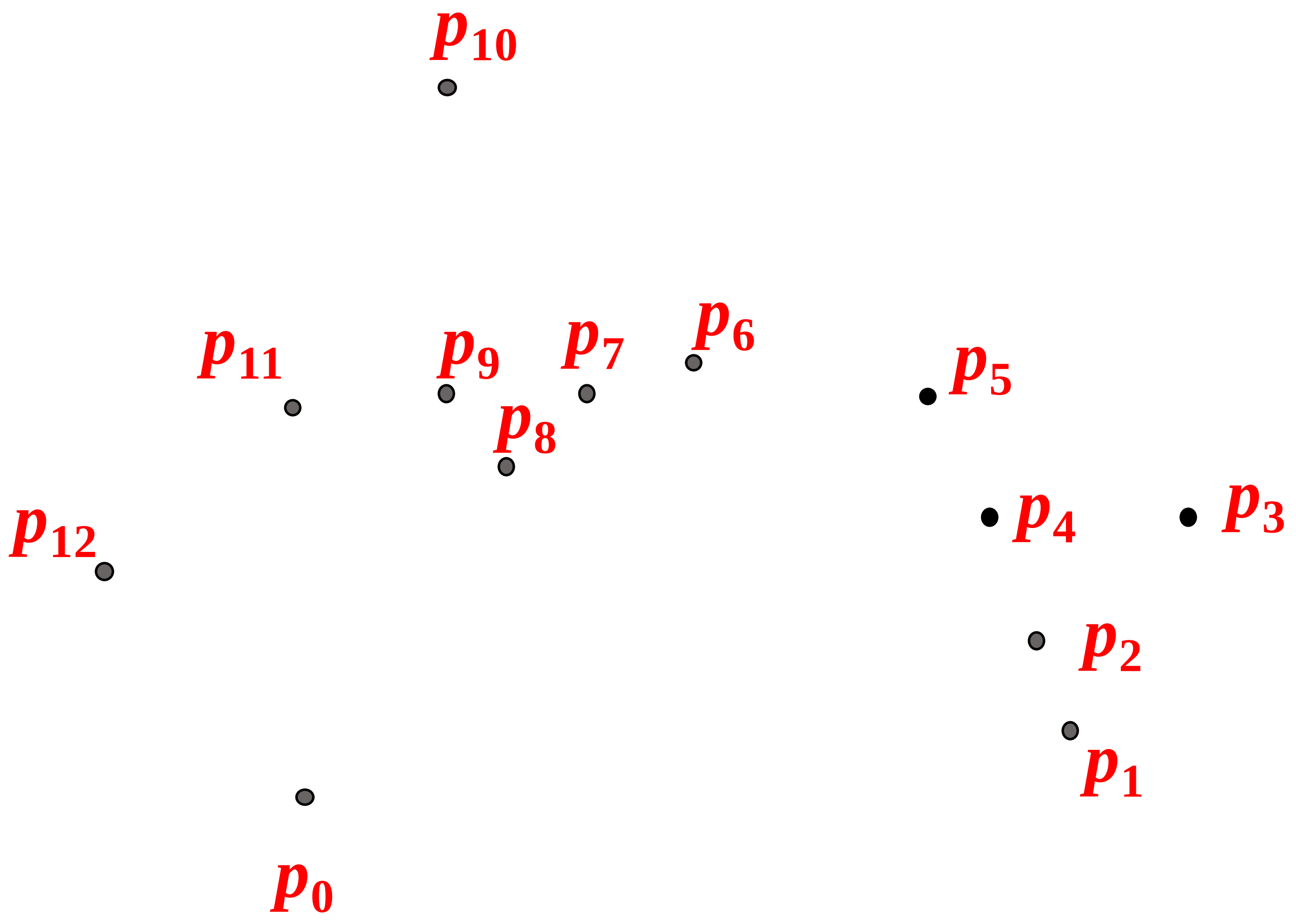
```
Input: a set of points S = {P = (P.x ,P.y)}
        Select the rightmost lowest point $P_0$ in S.
        Sort S angularly about $P_0$ as a center.
           For ties, discard the closer points.
        Let P[N] be the sorted array of points.

        Push P[0]=$P_0$ and P[1] onto a stack W.

        while i < N
        {
           Let $P_{T1}$ = the top point on W
           Let $P_{T2}$ = the second top point on W
           if (P[i] is strictly left of the line $P_{T2}$ to $P_{T1}$) {
              Push P[i] onto W
              i++    // increment i
           }
           else
              Pop the top point $P_{T1}$ off the stack
            }
        Output:  W = the convex hull of S.
```

Algorithms and Complexities

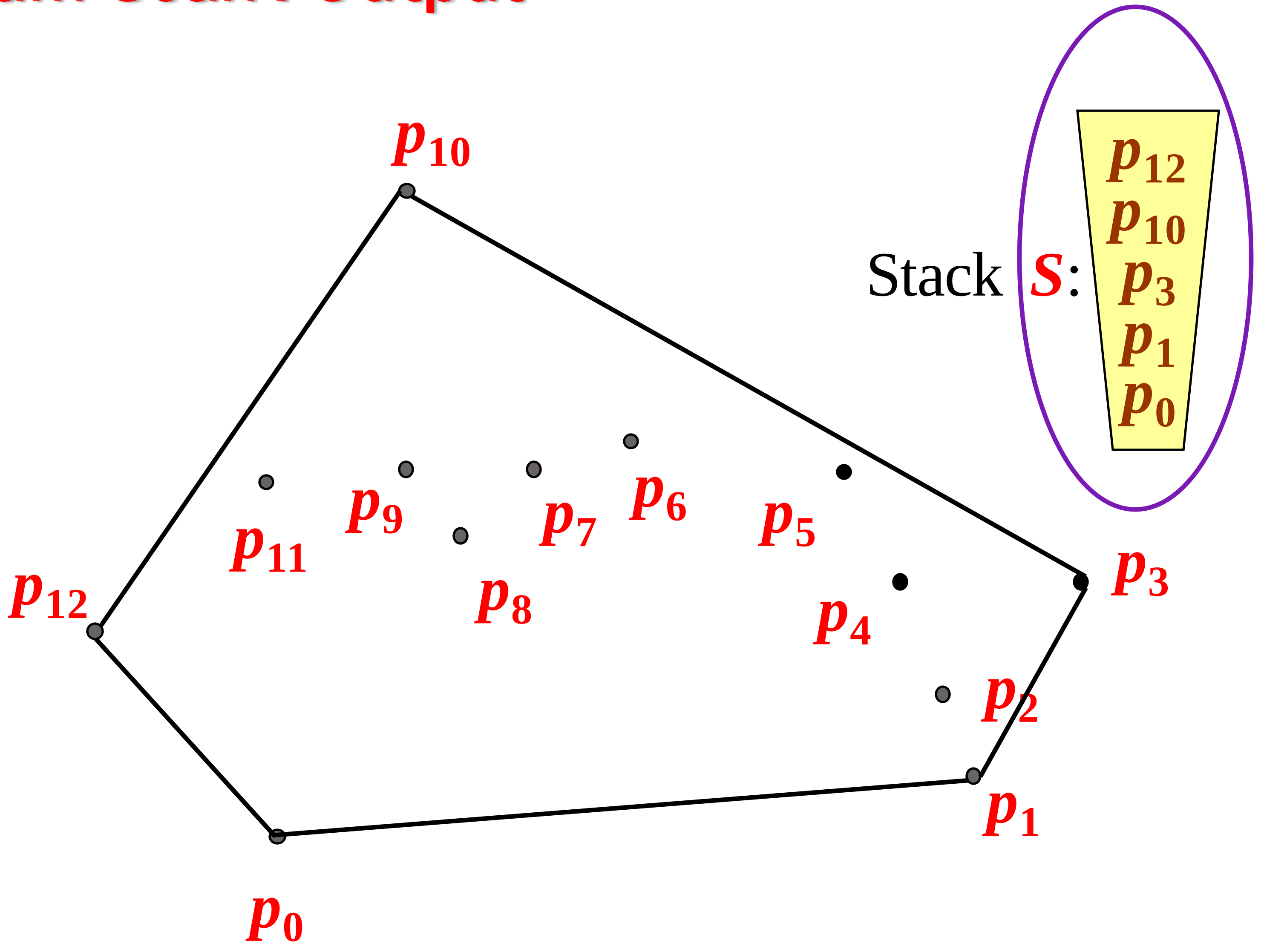
Any convex hull algorithm have the lower bound of $\theta(n\log n)$ through a reduction from Sorting, but it gives rises to Output Sensitive Algorithms, where the Complexity of the algorithm depends on the size of the output. Time complexity of each algorithm is stated in terms of the number of inputs points n and the number of points on the hull h. Note that in the worst case h may be as large as n

Algorithm	Complexity
Naive Bruteforce	$O(n^4)$
Better Bruteforce	$O(n^3)$
Gift Wrapping (Jarvis March)	$O(nh)$
QuickHull	$O(nh)$
Graham Scan	$O(n\log n)$
Divide and Conquer	$O(n\log n)$
Monotone Chain	$O(n\log n)$
Incremental	$O(n\log n)$
Marriage before Conquest	$O(n\log n)$
Chan's	$O(n\log n)$
Shatterhull	$O(n\log n)$

Graham-Scan : Input

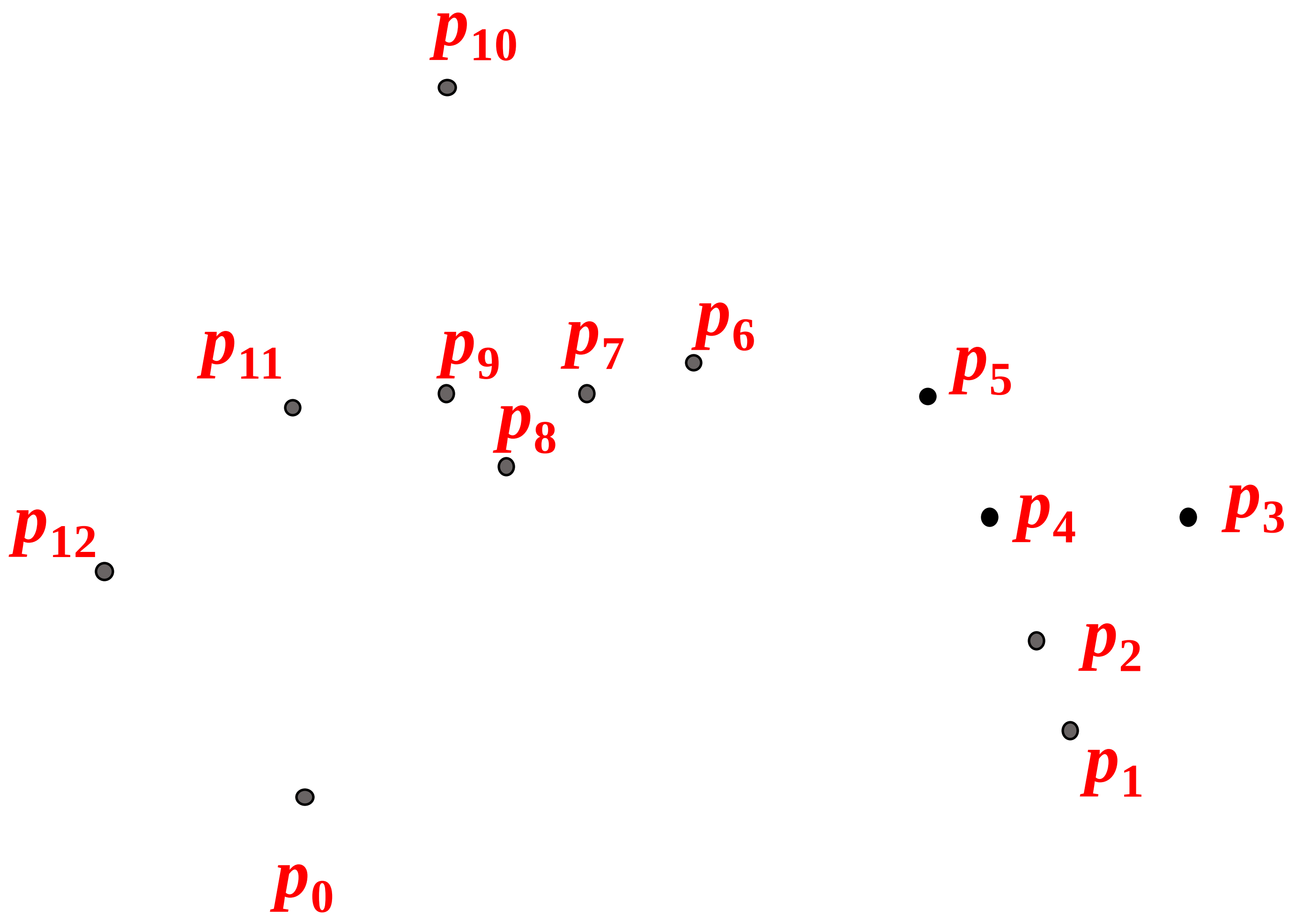


Graham-Scan : Output



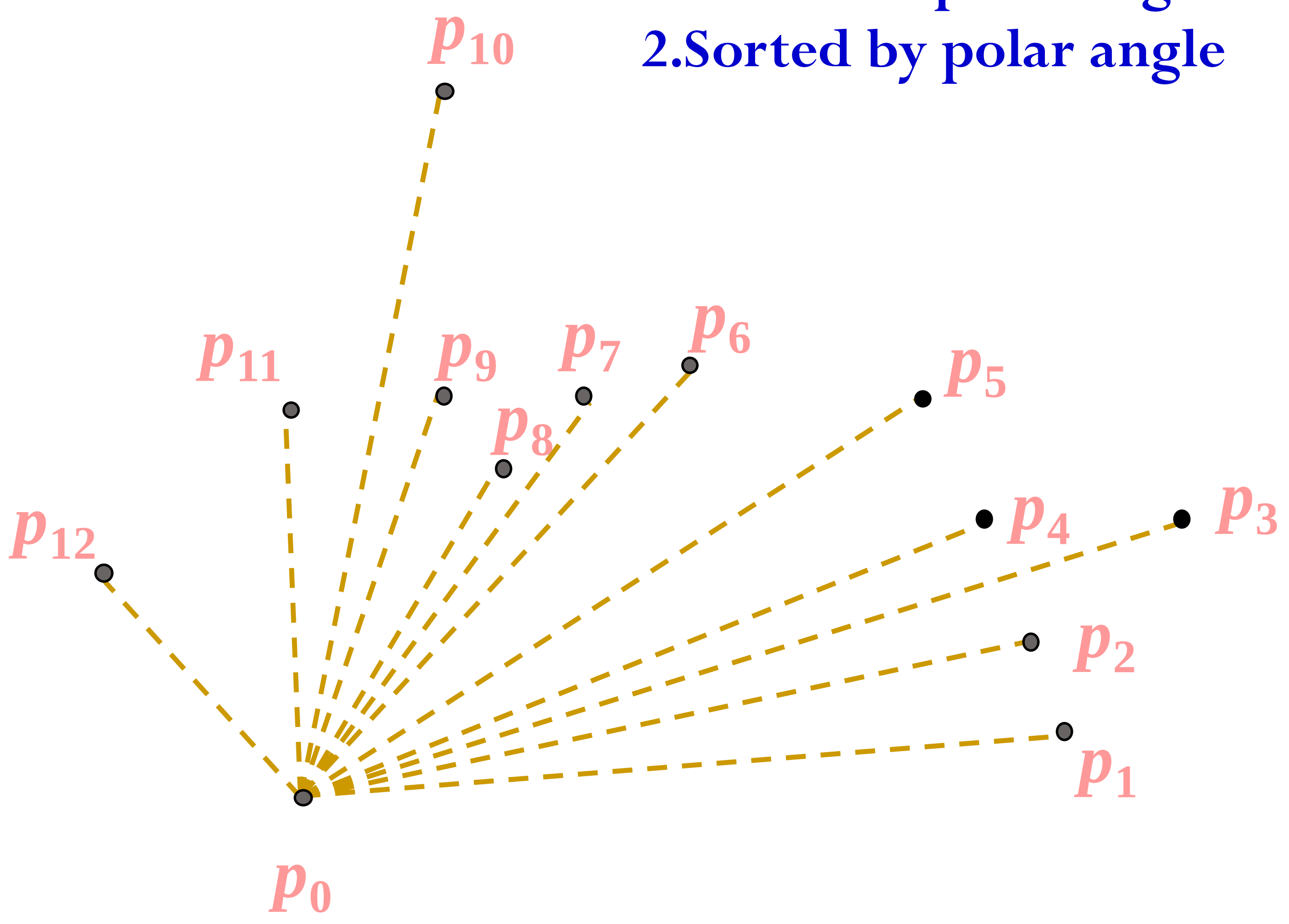
Graham-Scan for finding the CH

Graham-Scan : Input

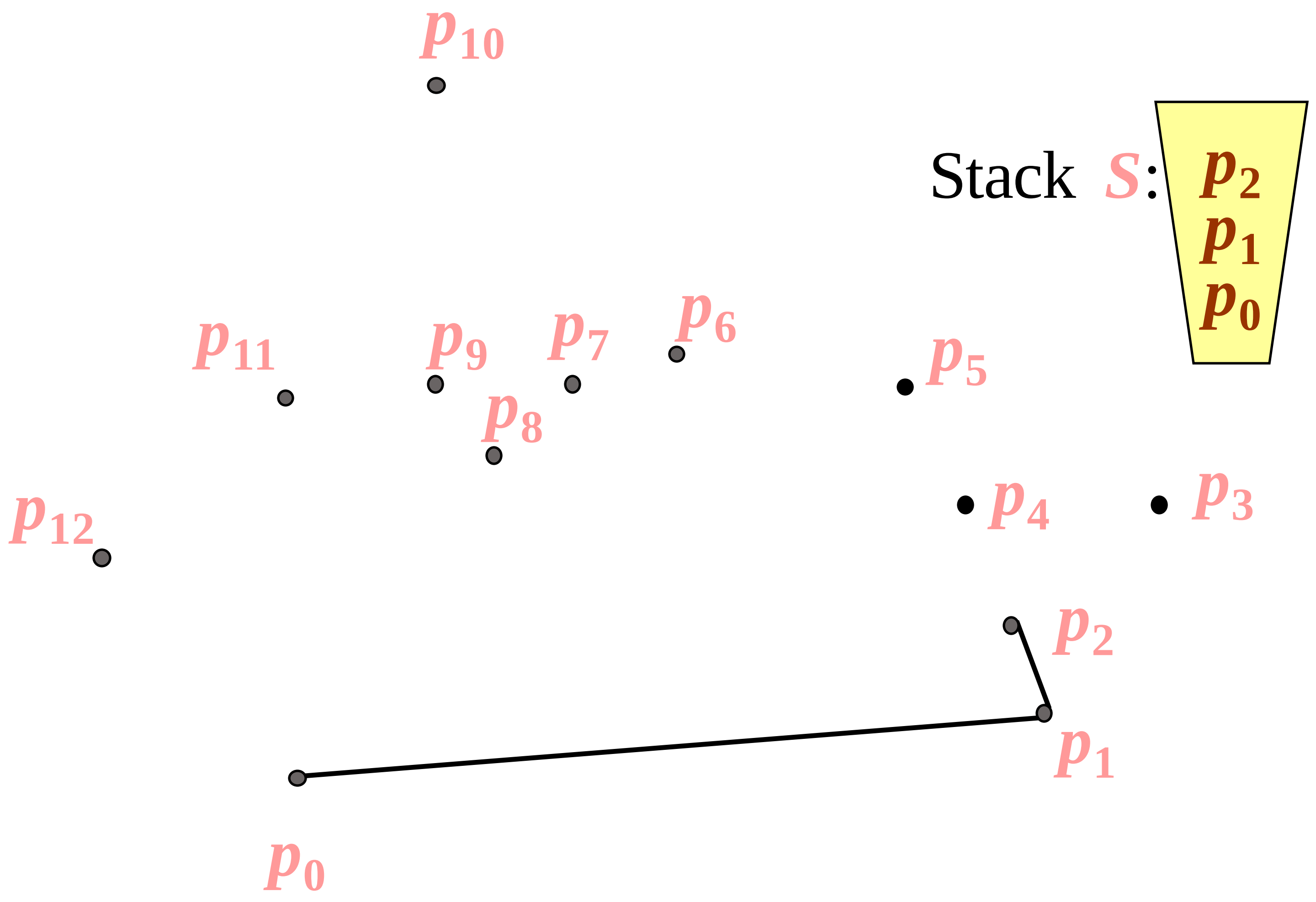


Graham-Scan :(1/11)

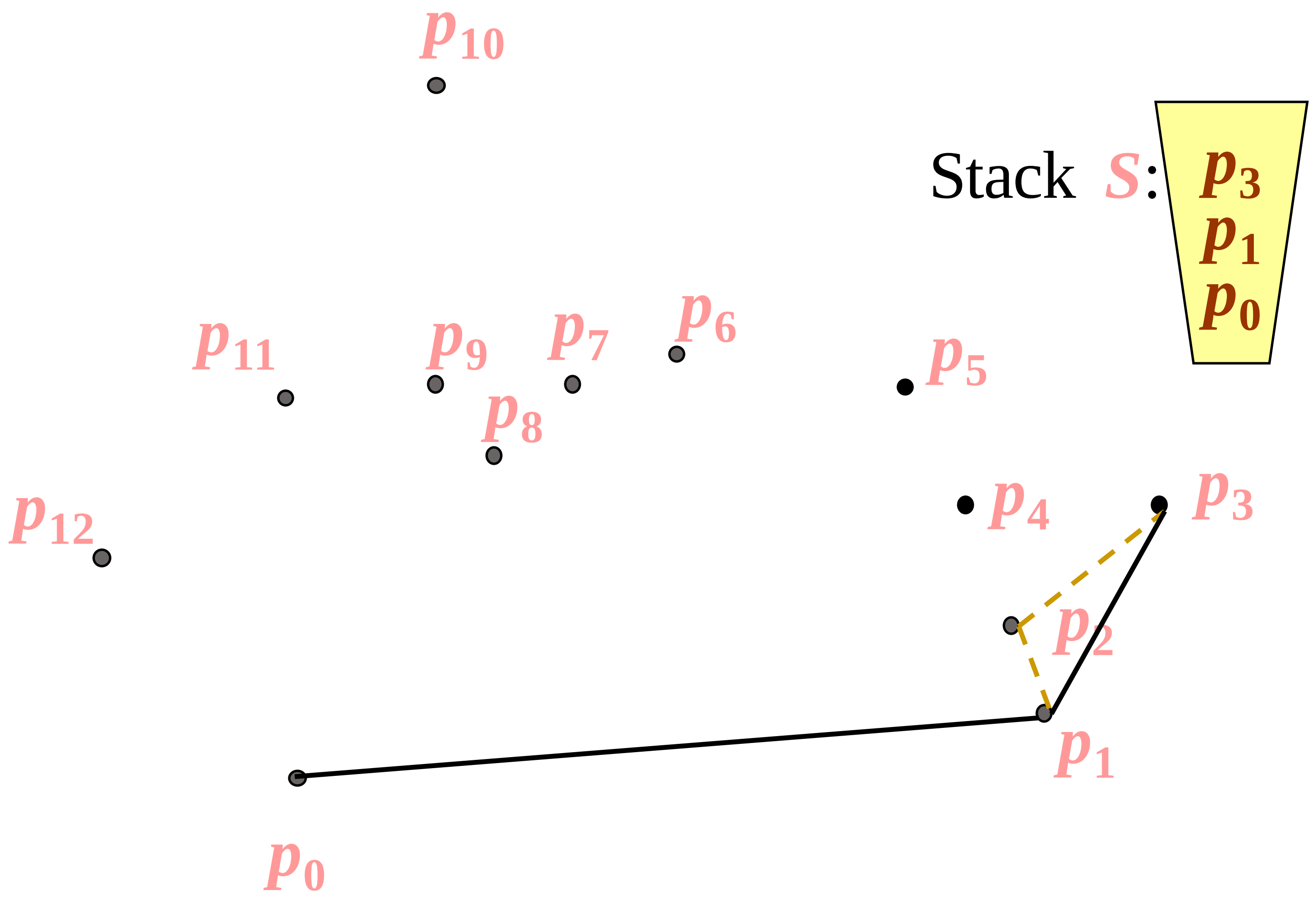
1.Calculate polar angle
2.Sorted by polar angle



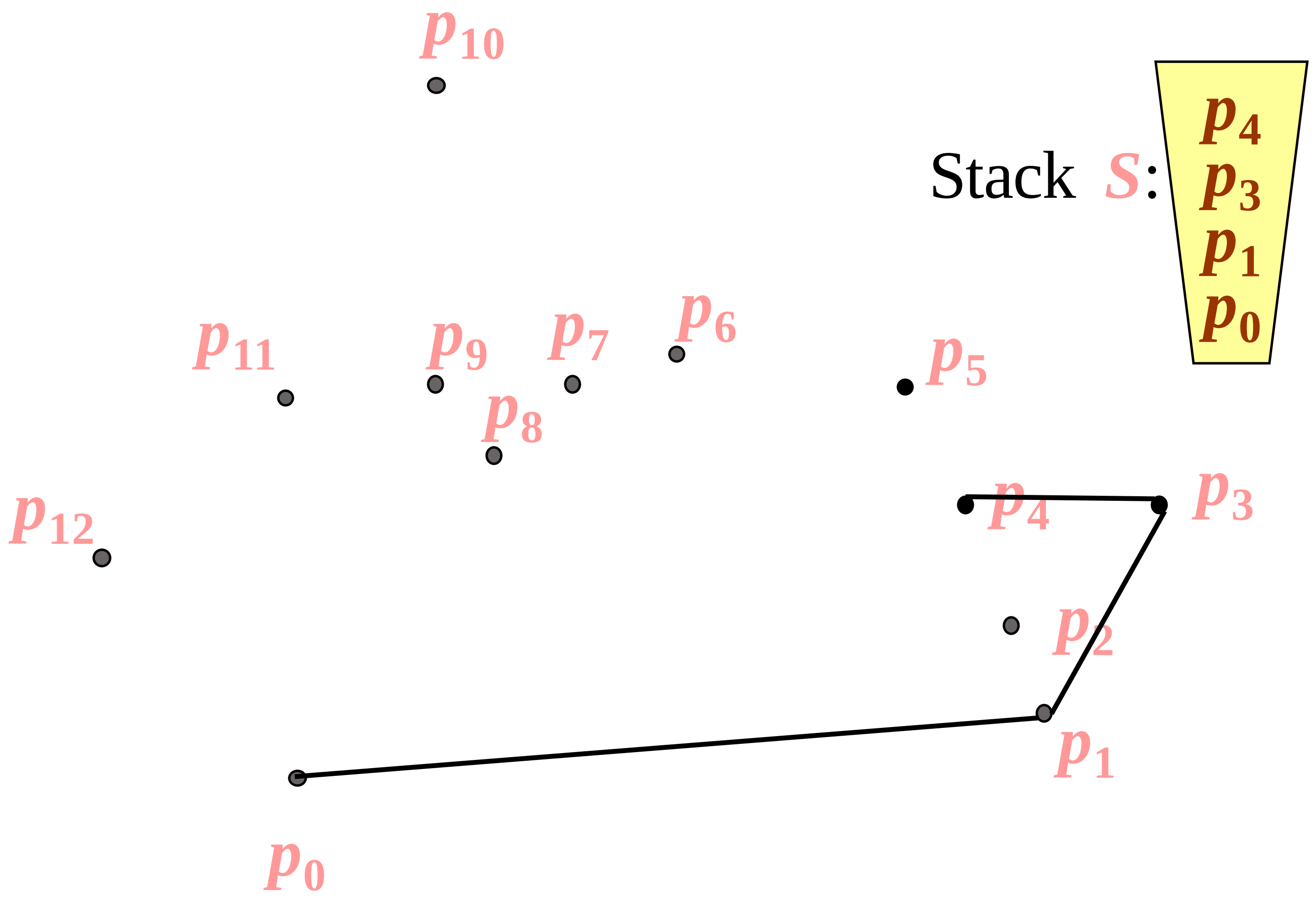
Graham-Scan : (2/11)



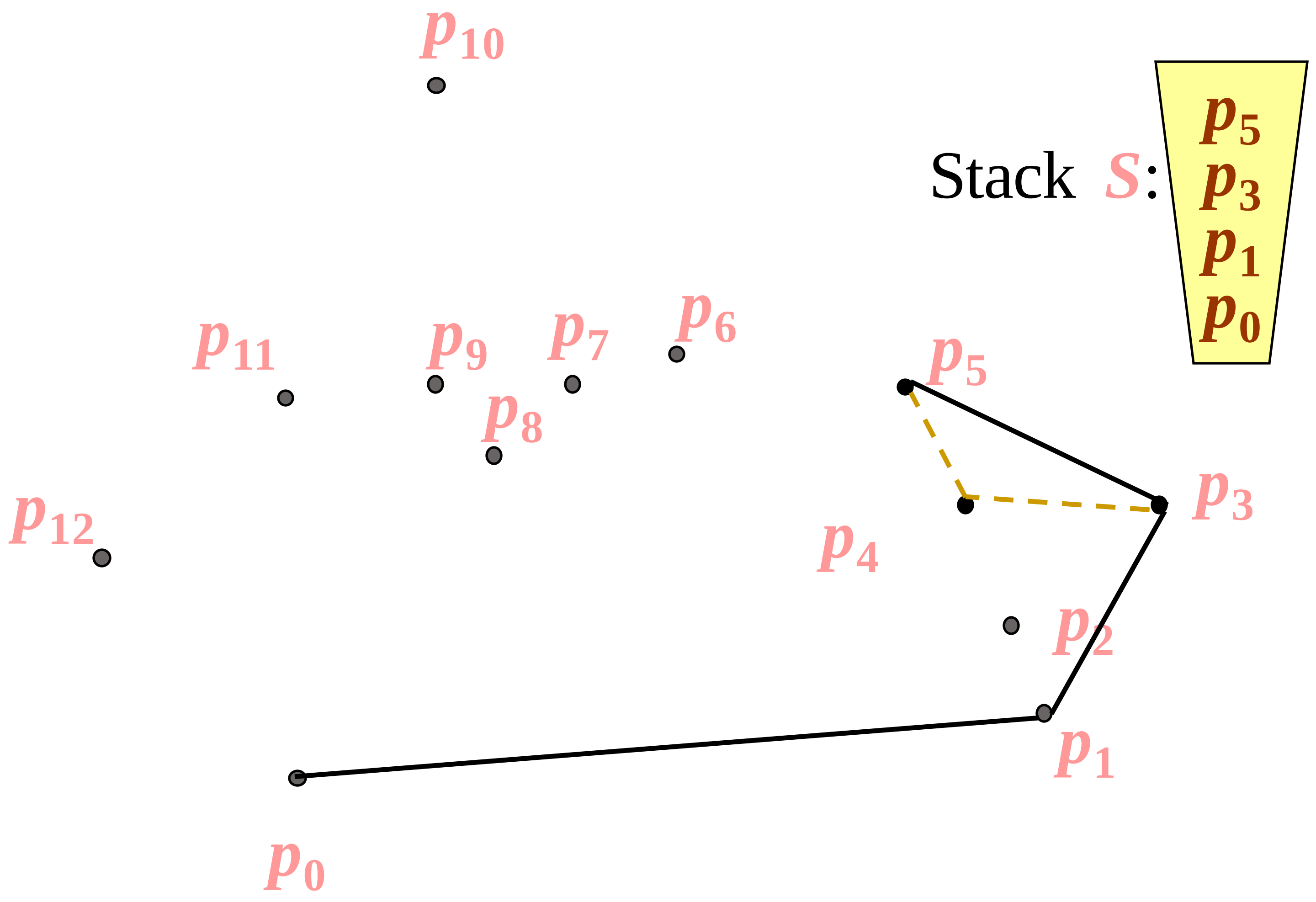
Graham-Scan : (3/11)



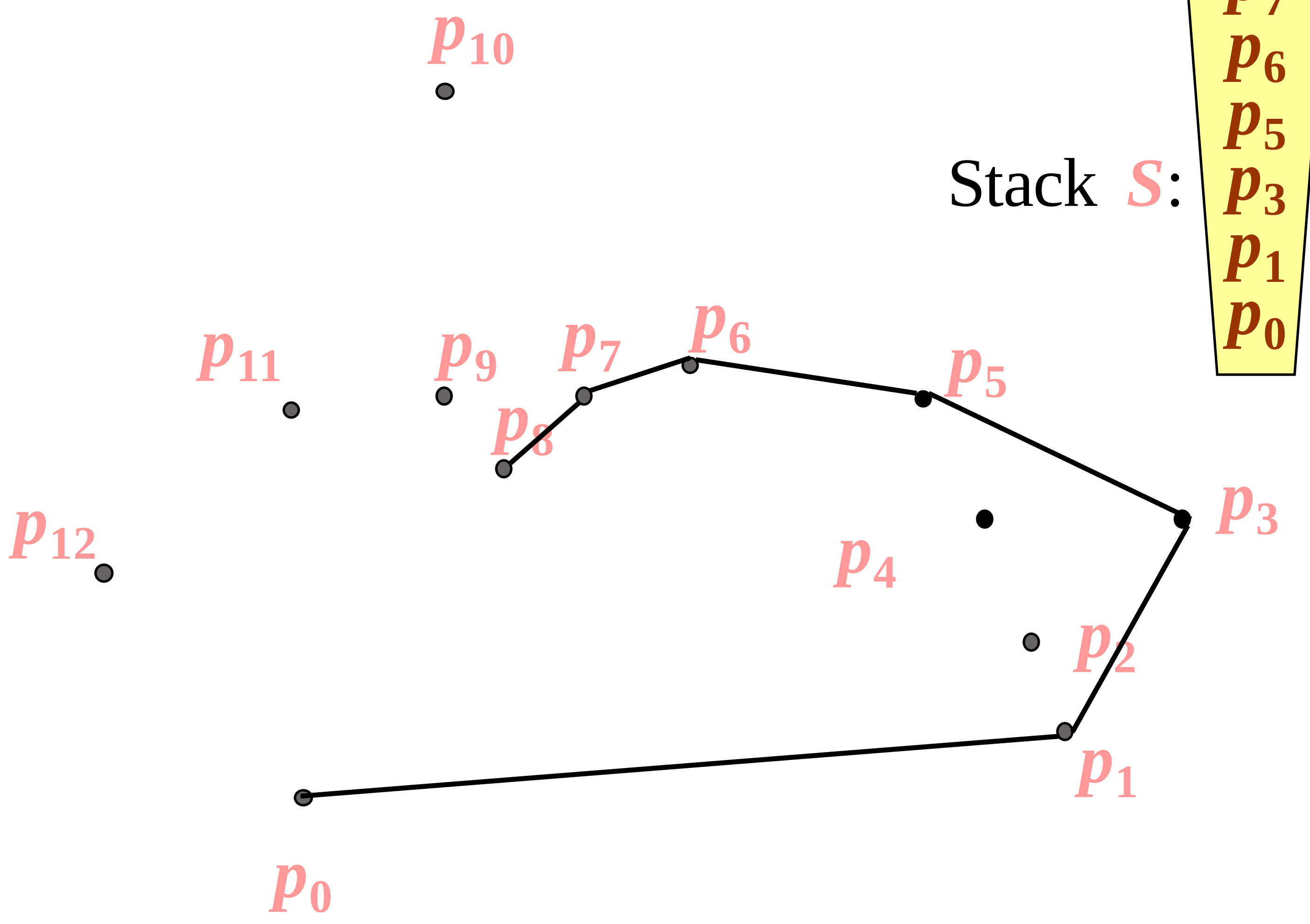
Graham-Scan : (4/11)



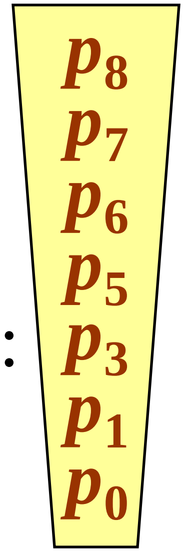
Graham-Scan (5/11)



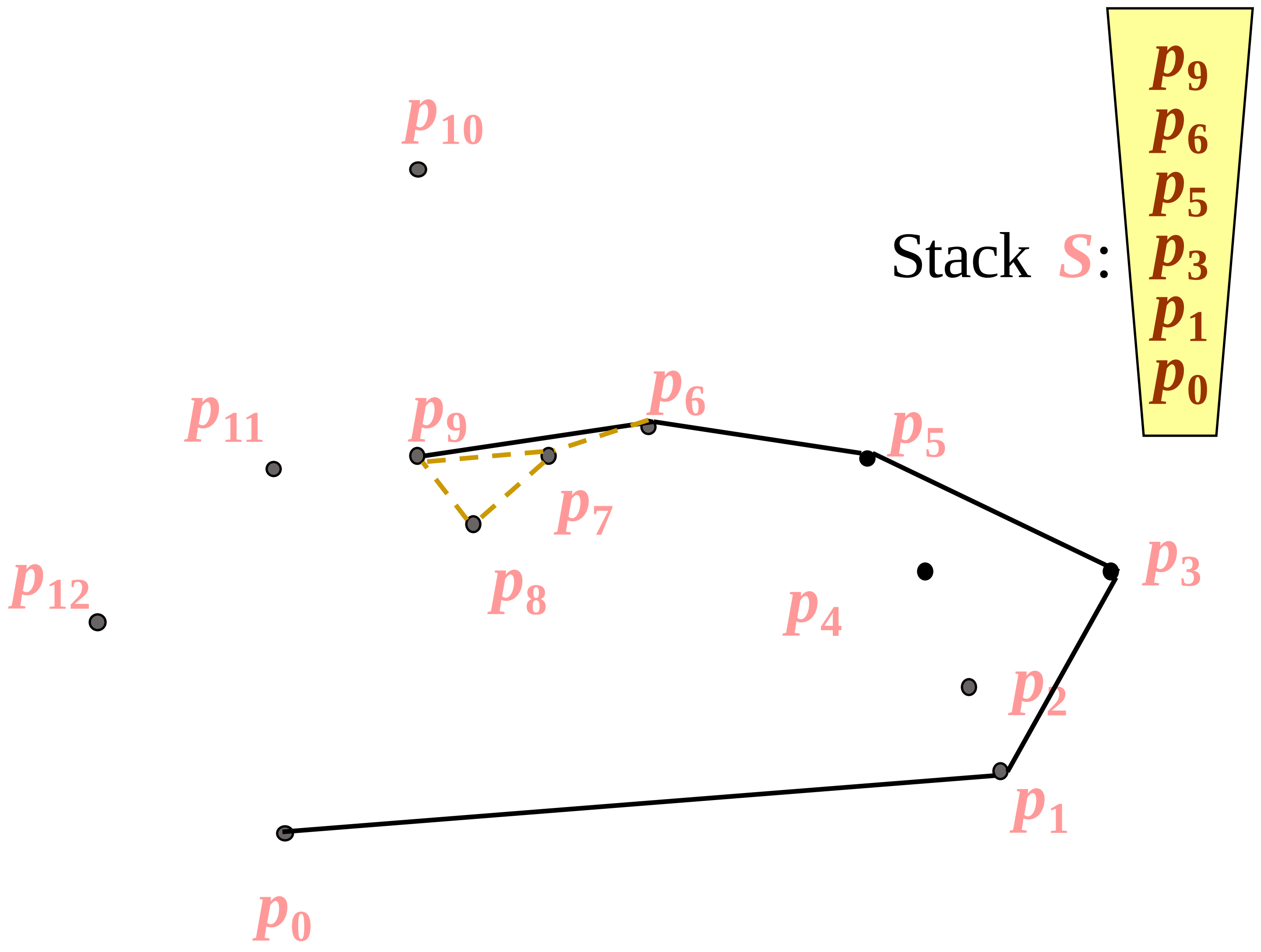
Graham-Scan (6/11)



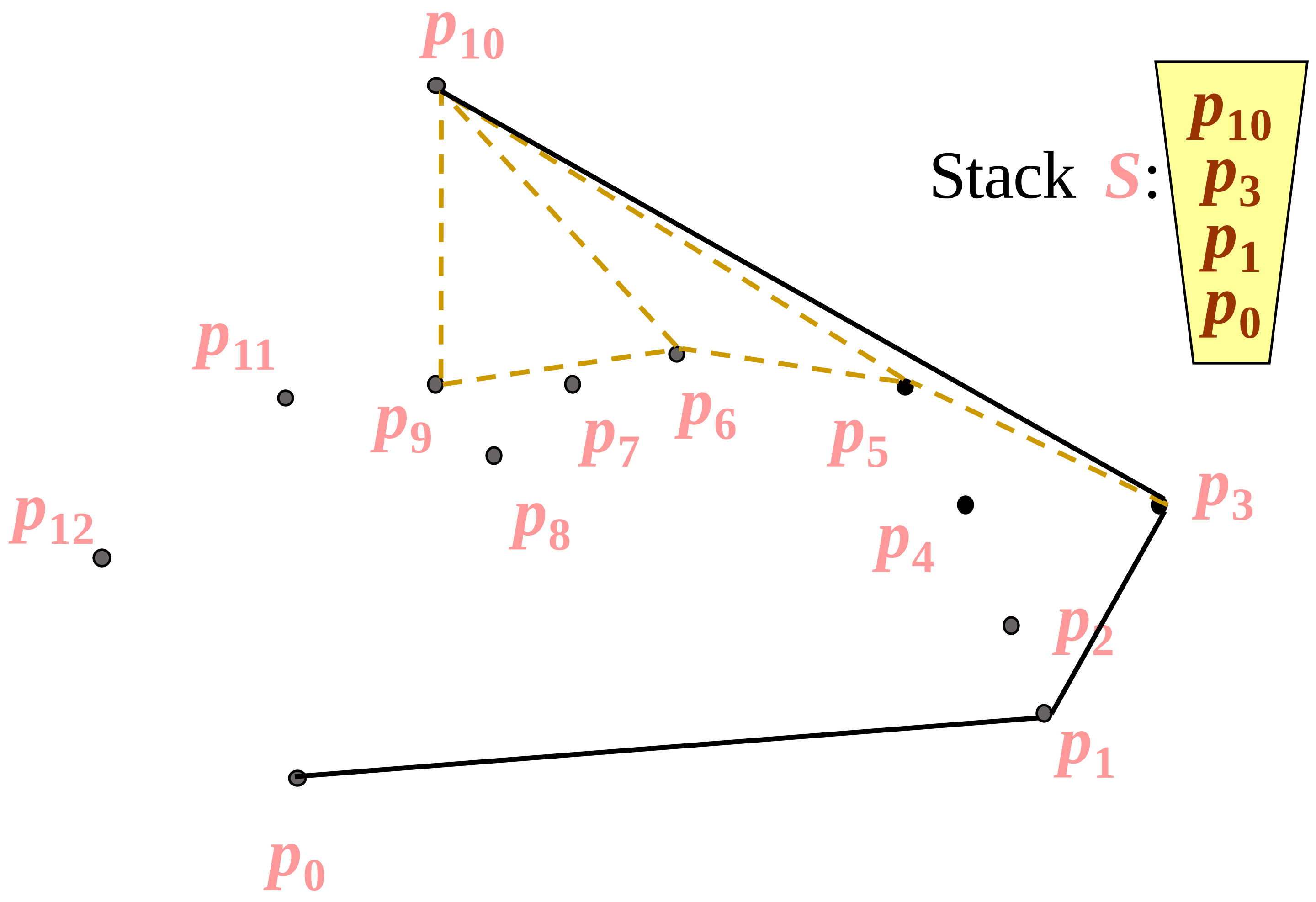
Stack S:



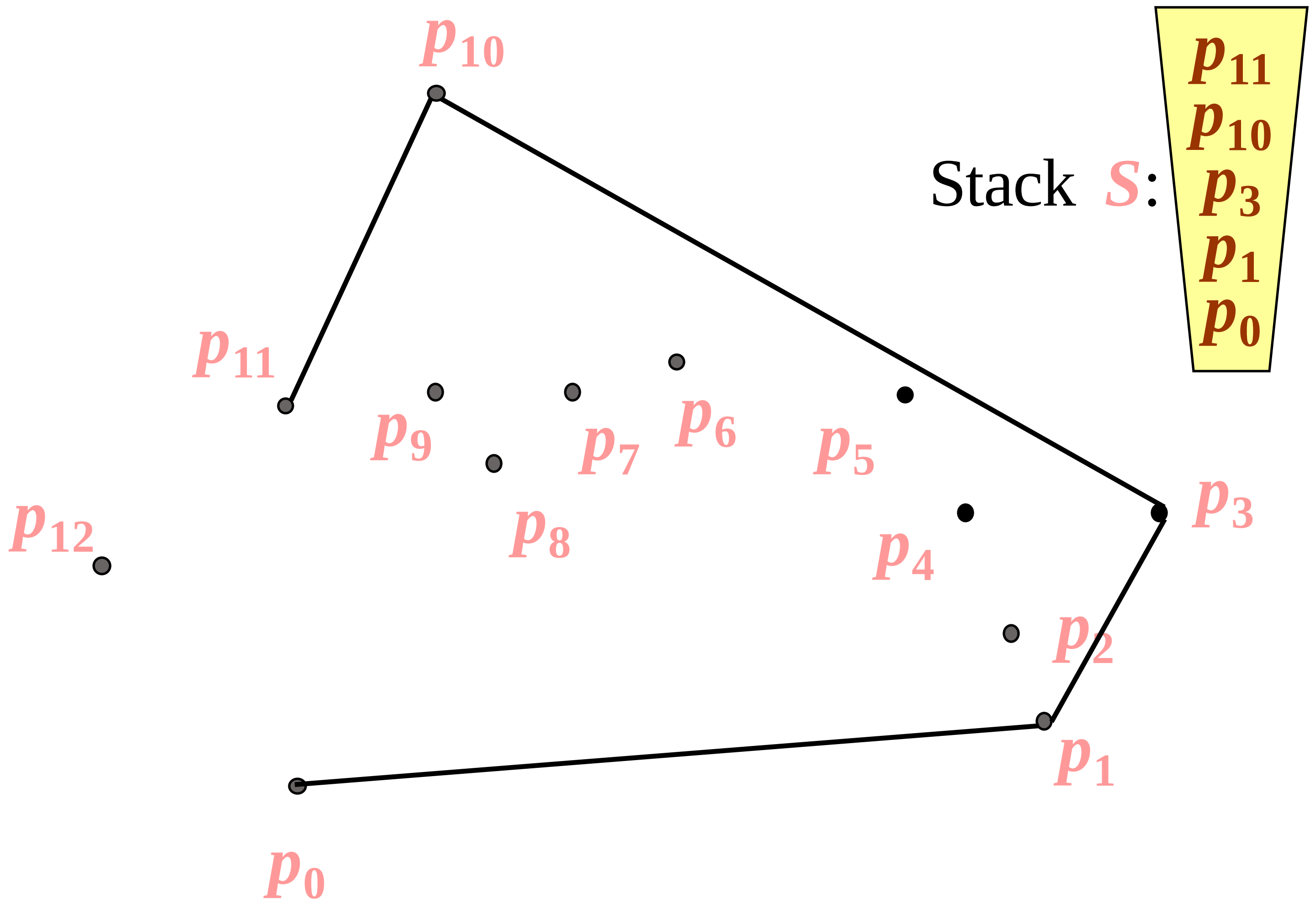
Graham-Scan (7/11)



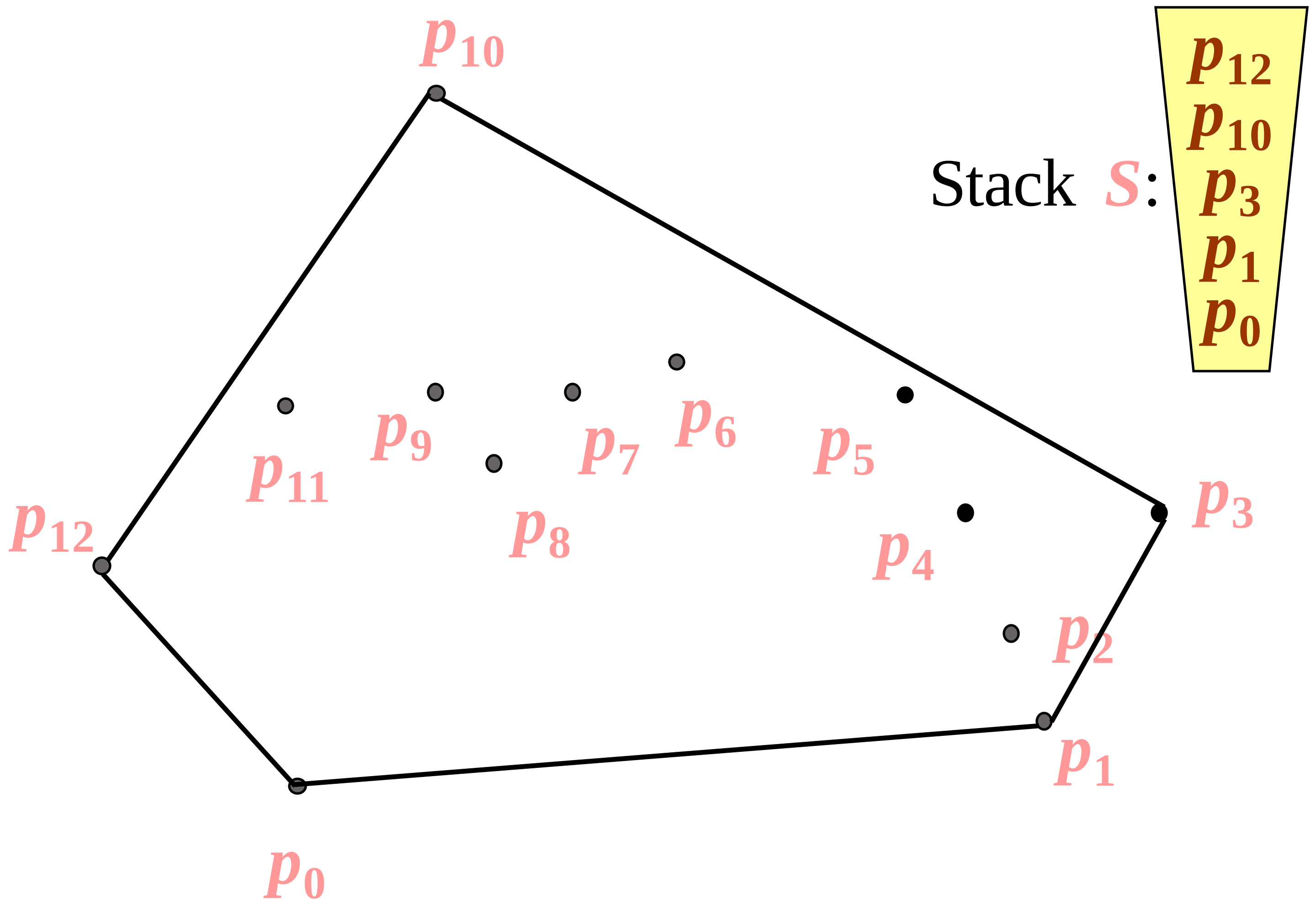
Graham-Scan (8/11)



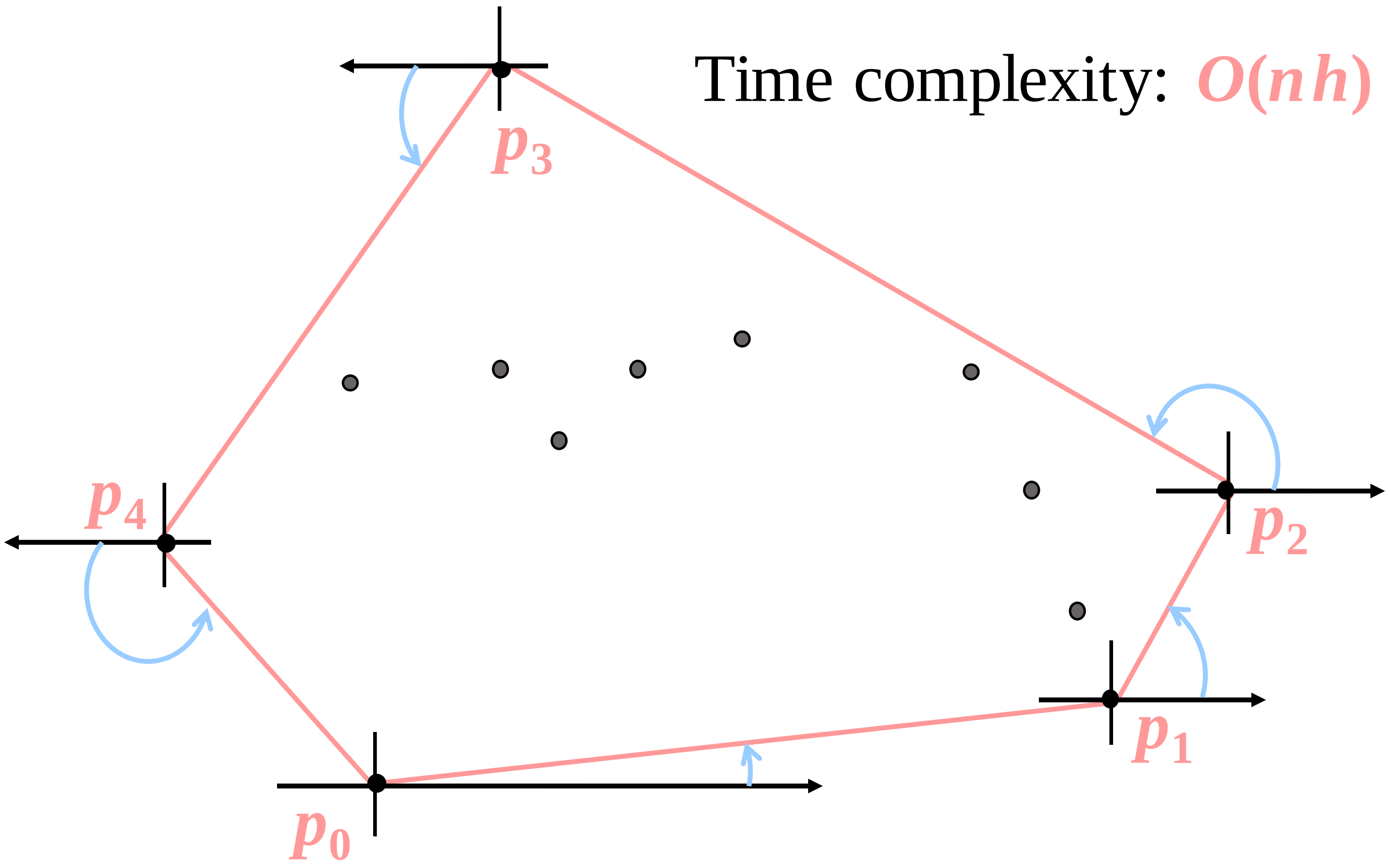
Graham-Scan (9/11)



Graham-Scan (10/11)



Jarvis's march for finding the CH

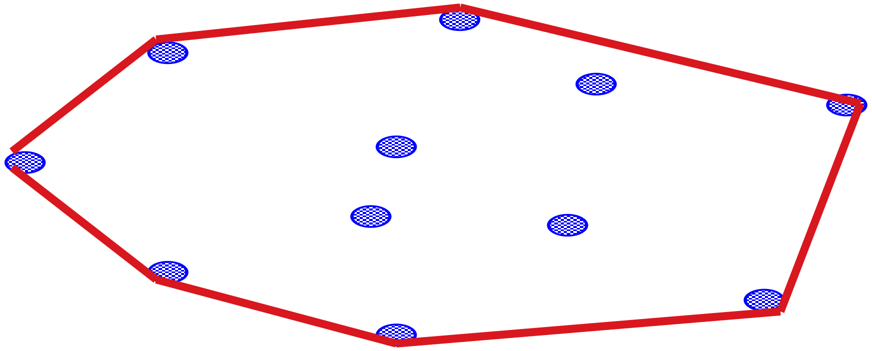


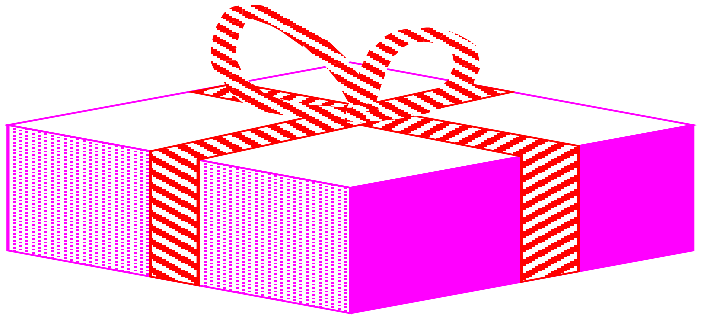
Time complexity: $O(nh)$

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)

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.



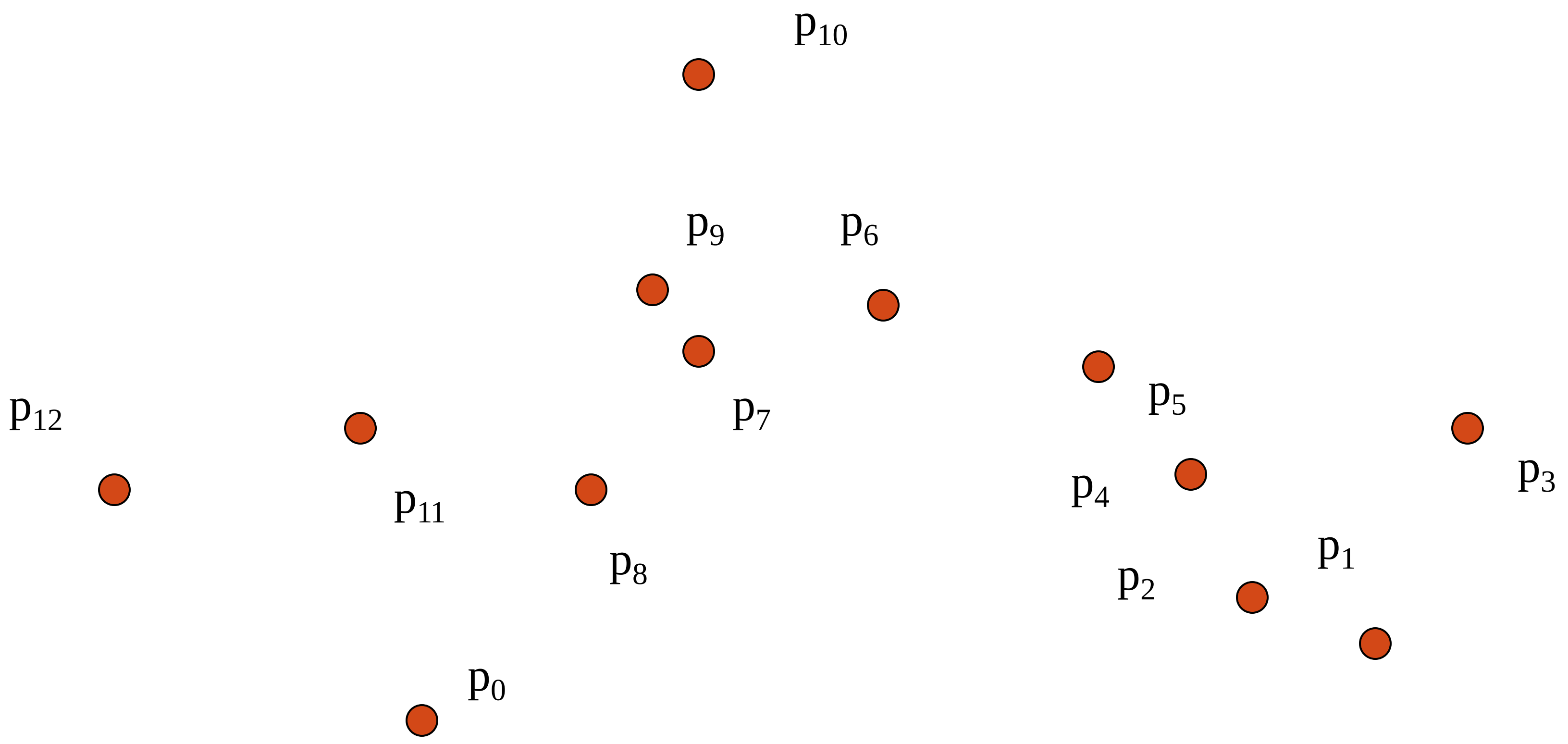


Algorithm: Gift Wrapping

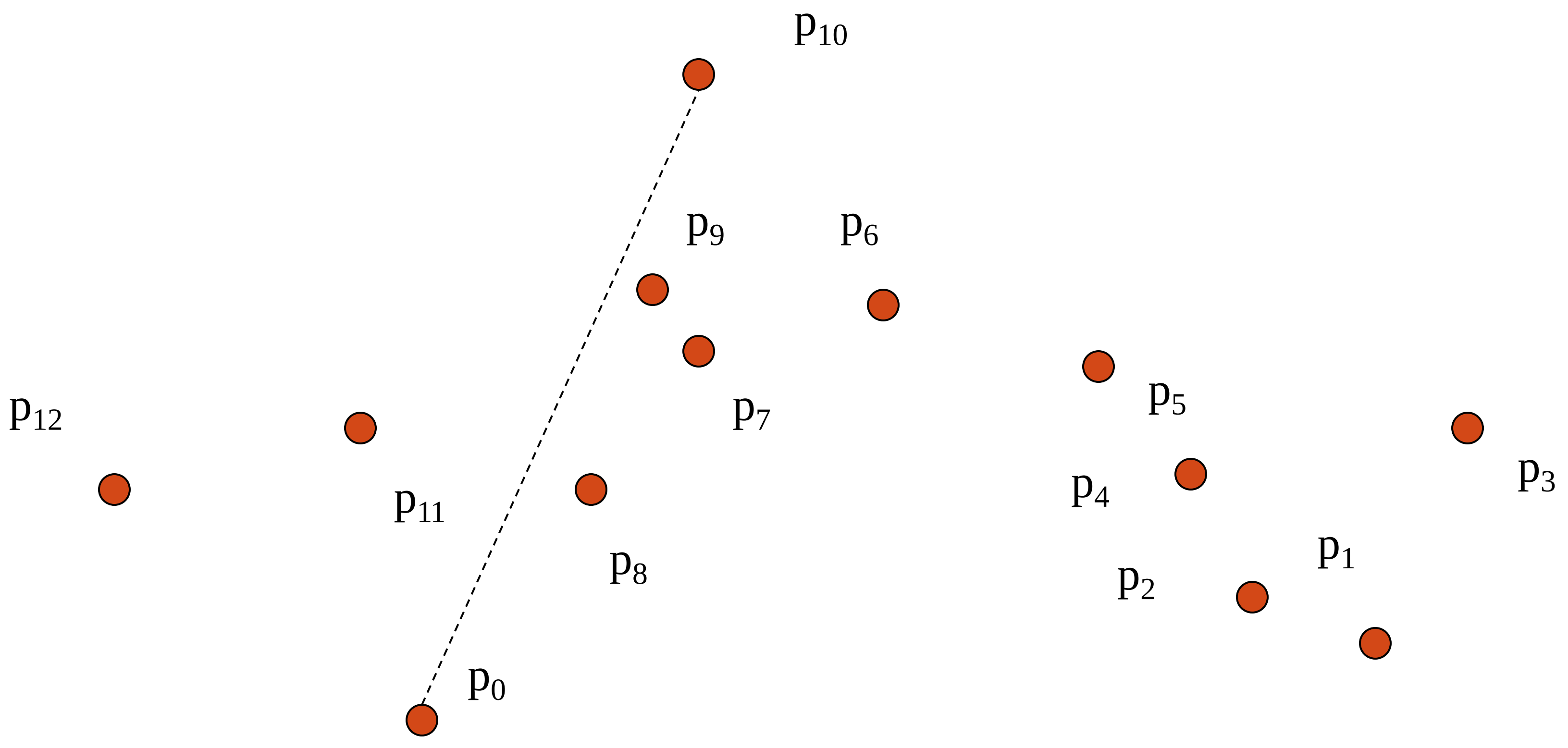
```
Find the lowest point (smallest y coordinate)
Let $i_0$ be its index, and set $i \leftarrow i_0$
repeat
 for each $j \neq i$ do

   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(n²)*
- 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

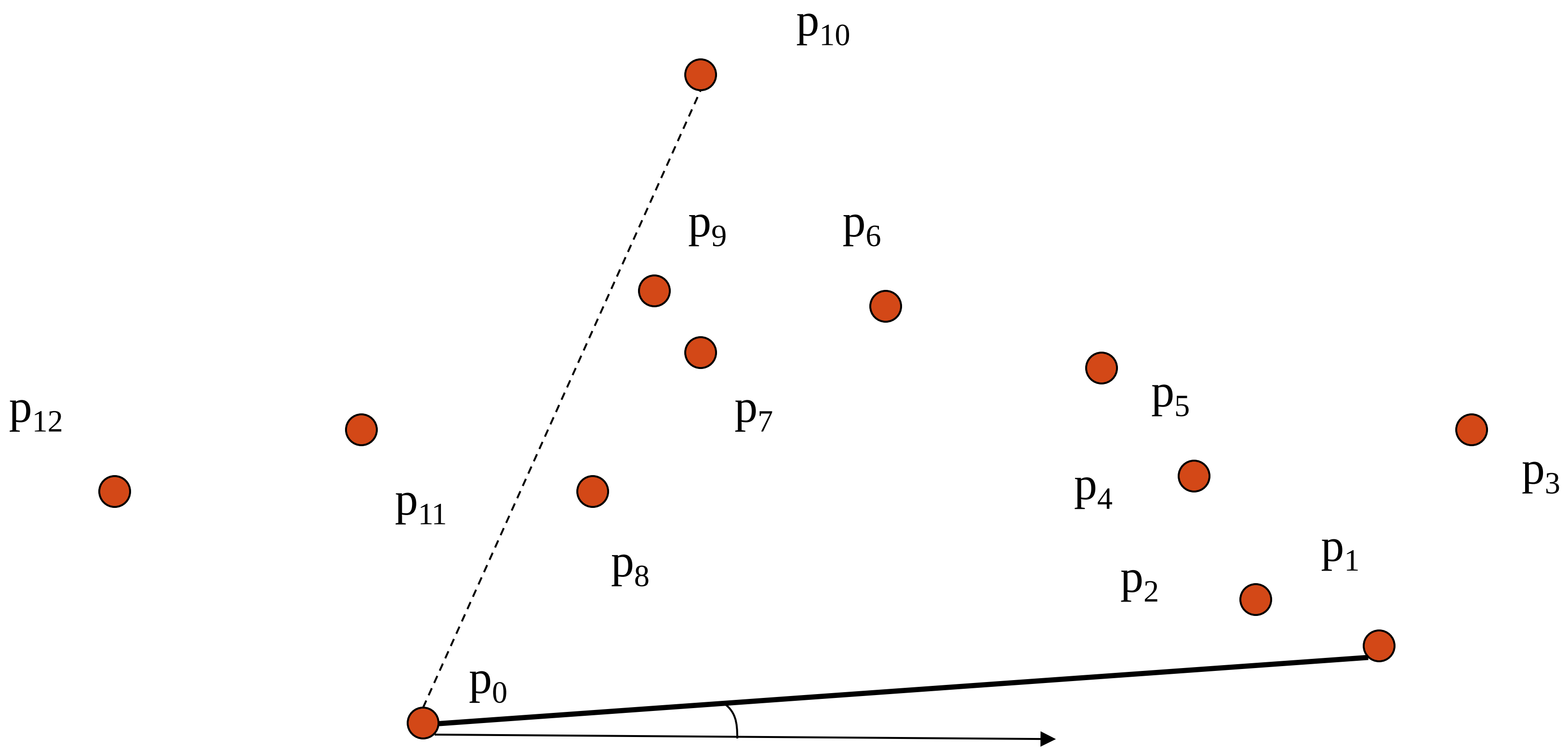
Jarvis March - Example



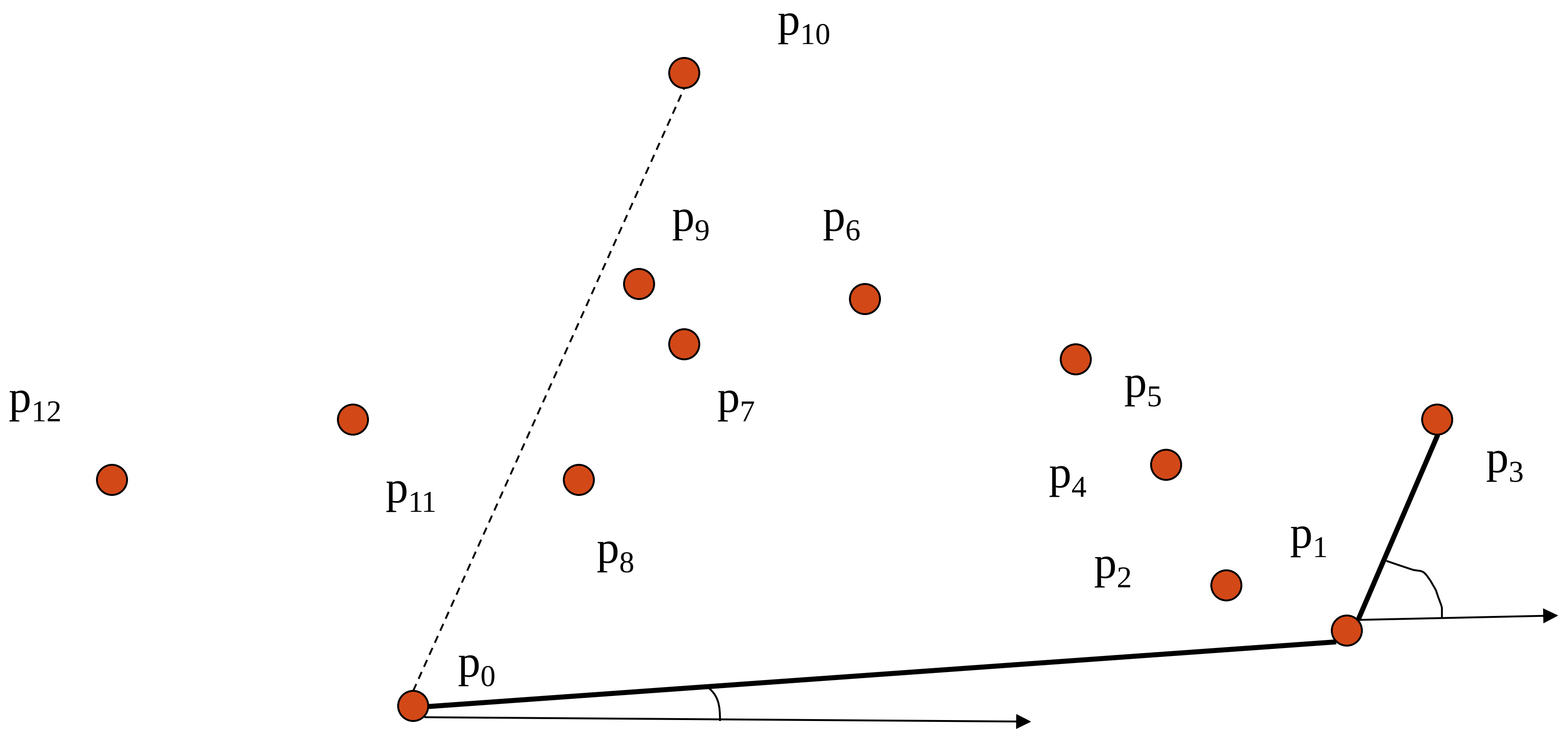
Jarvis March - Example



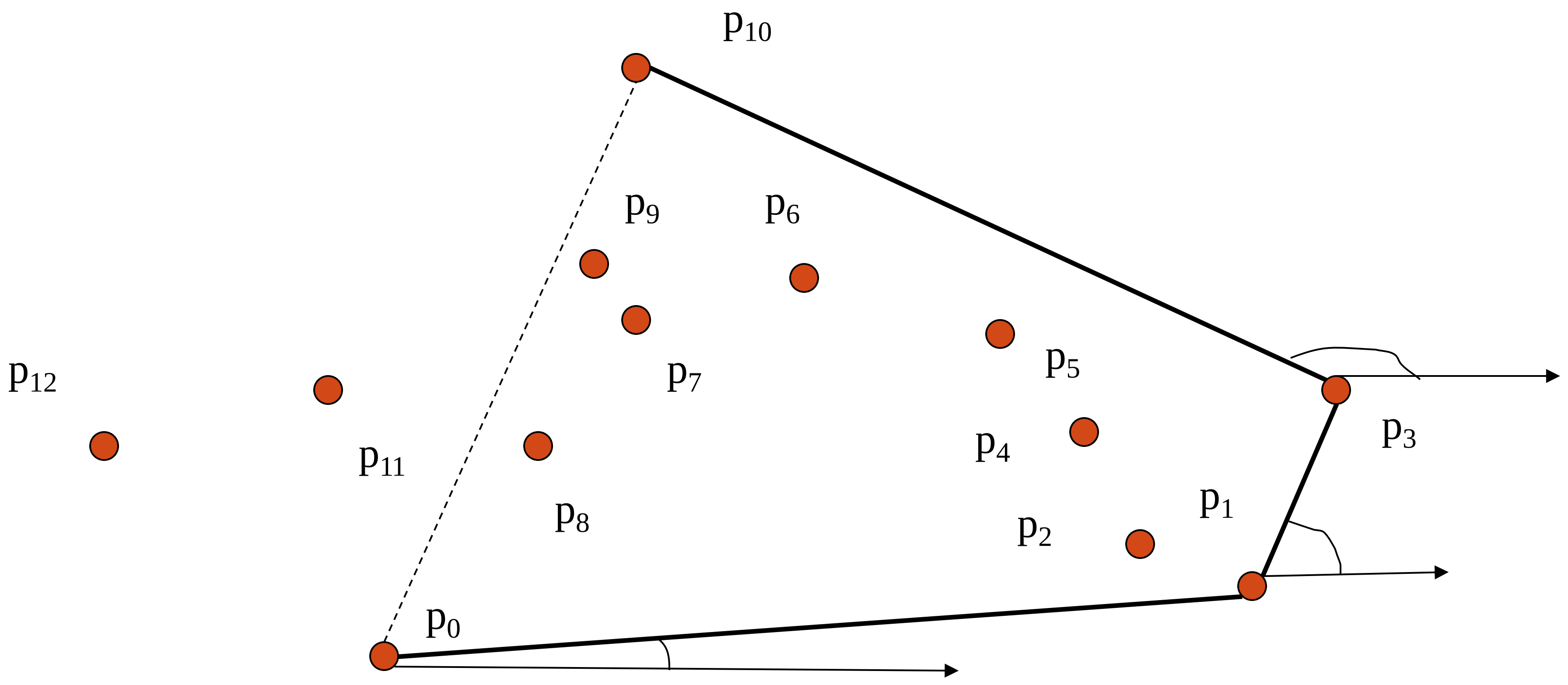
Jarvis March - Example



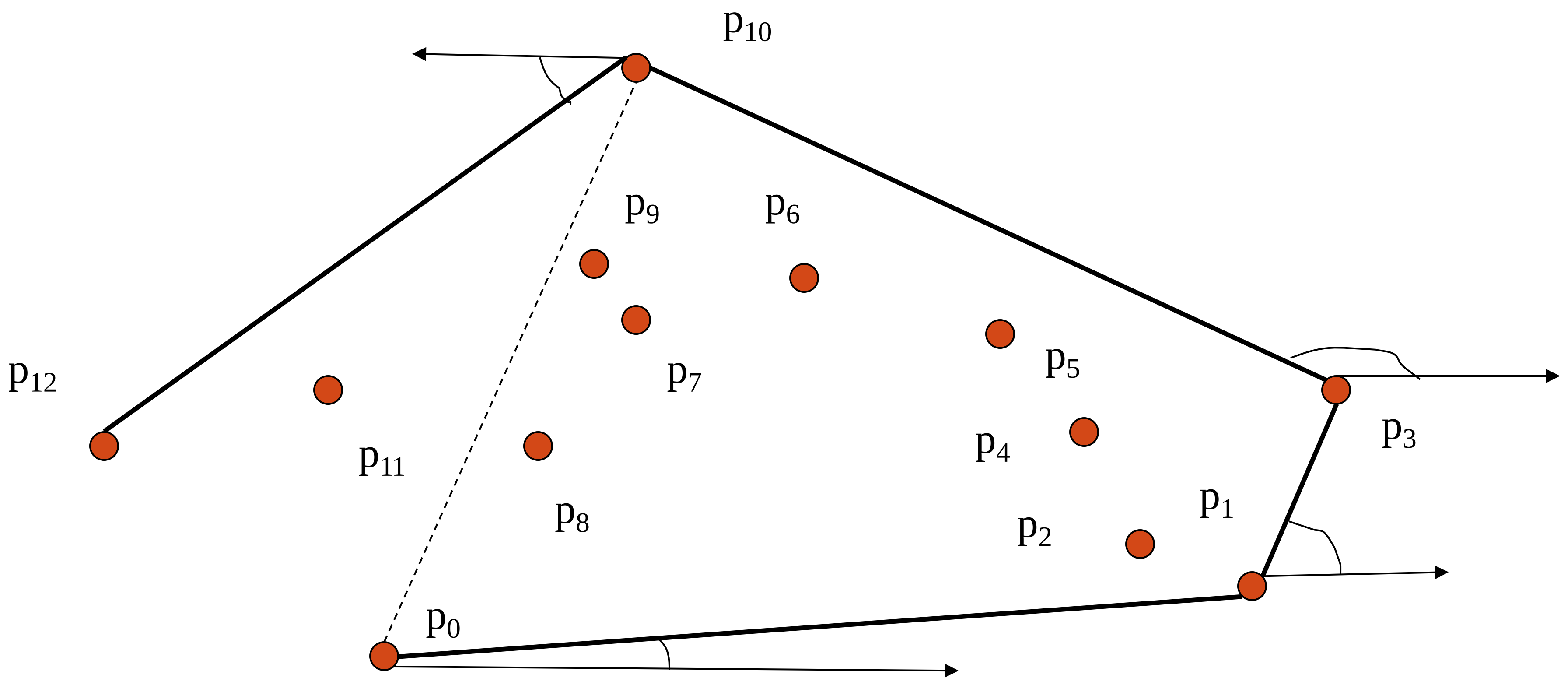
Jarvis March - Example



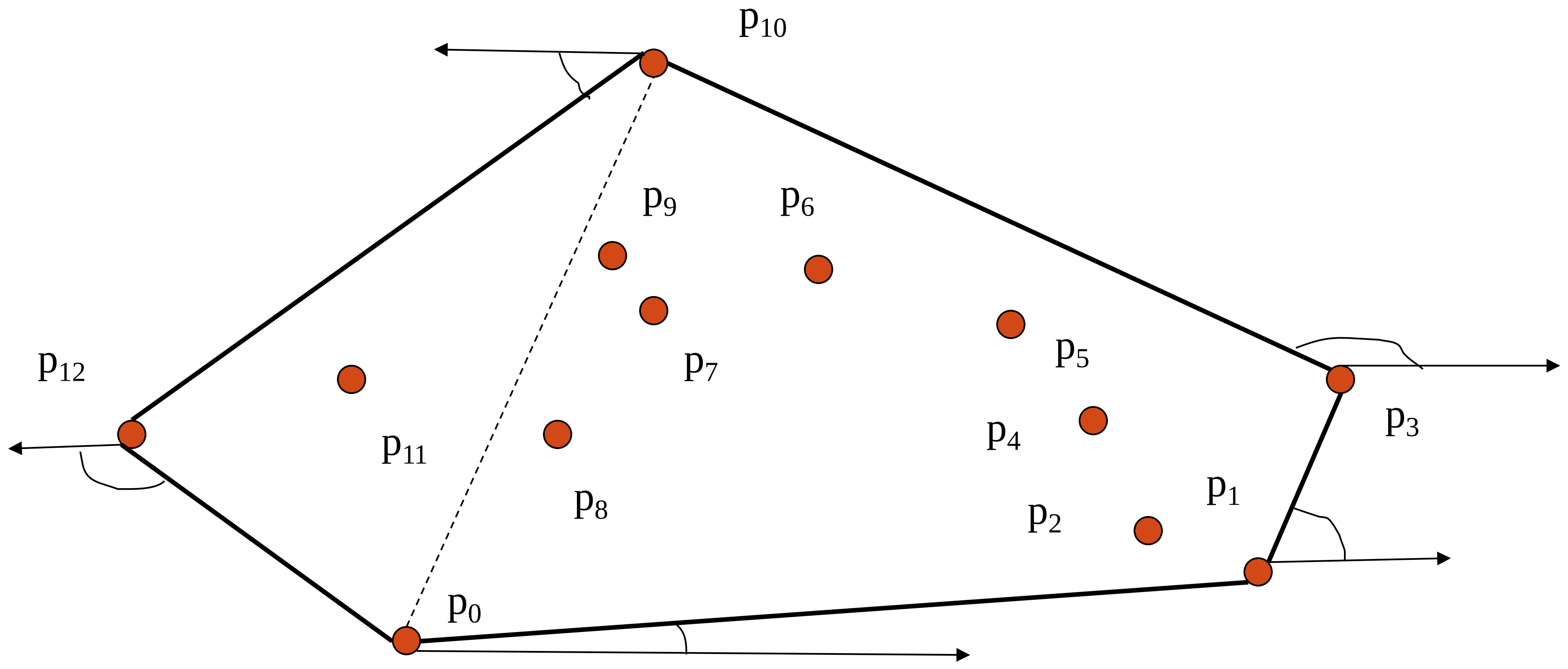
Jarvis March - Example



Jarvis March - Example



Jarvis March - Example



Gift Wrapping

- $O(n\ |H(S)|\ )$

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 .

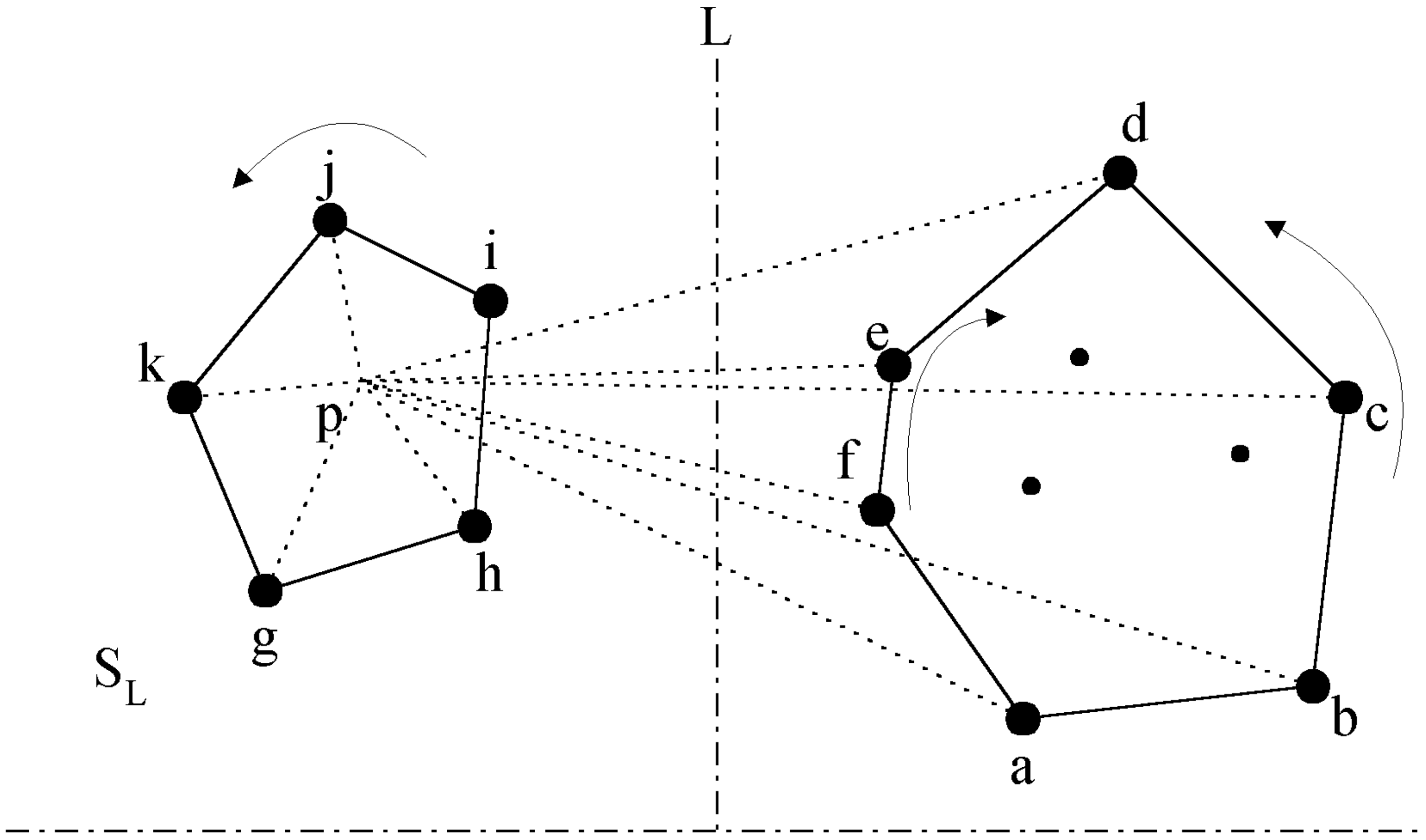
Step 3: Recursively construct convex hulls for S_L and S_R. Denote these convex hulls by Hull(S_L) and Hull(S_R) respectively.

Step 4: Apply the merging procedure to merge Hull(S_L) and Hull(S_R) together to form a convex hull.

- **Time complexity**:

$$T(n) = 2T(n/2) + O(n) \quad = O(n \log n)$$

- The divide-and-conquer strategy to solve the problem:



- **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

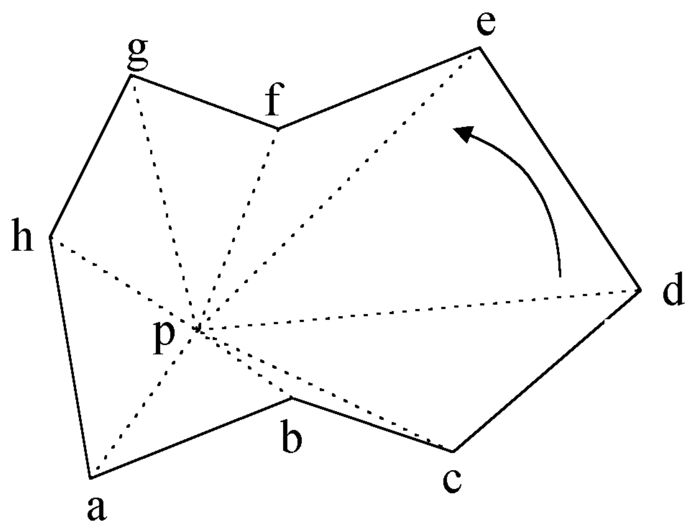
   (2) a, b, c, d

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

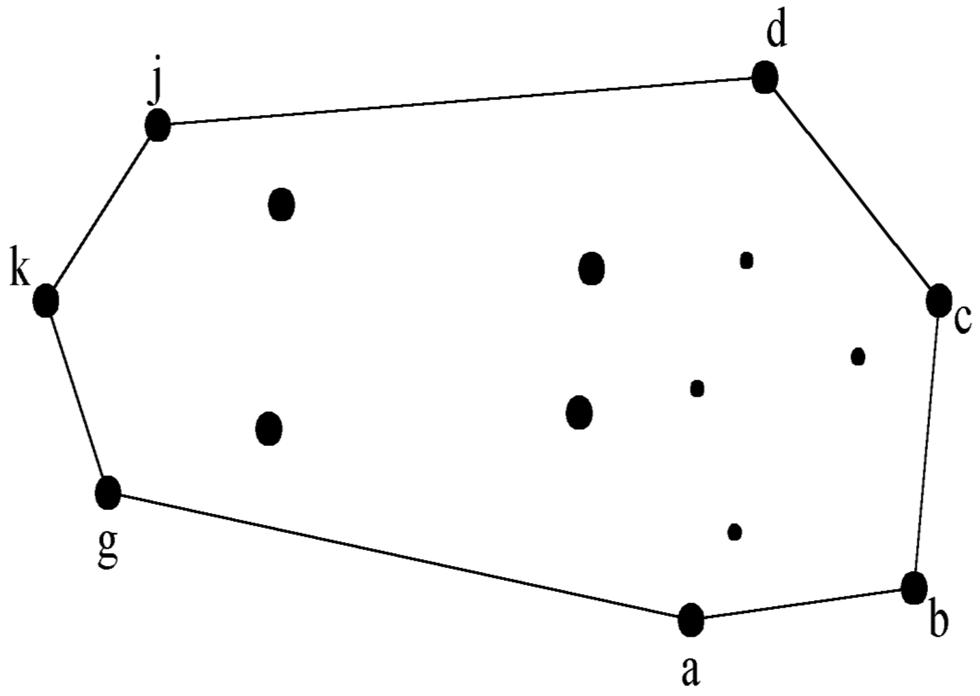
   g, h, a, b, f, c, e, d, i, j, k.
4. Apply Graham scan to examine the points one by one and eliminate the points which cause reflexive angles.

(See the example on the next page.)

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



Final result:



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) and Hull(S_R) respectively.

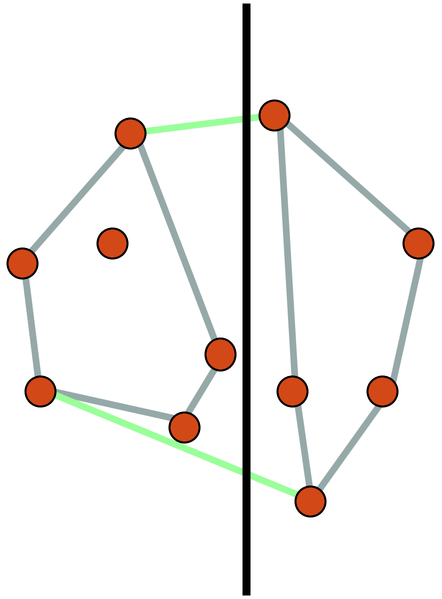
Step 4: Apply the merging procedure to merge Hull(S_L) and Hull(S_R) together to form a convex hull.

- **Time complexity**:

$$T(n) = 2T(n/2) + O(n) \quad = O(n \log n)$$

Convex Hull - Divide and Conquer

- Algorithm:
  - Find a point with a median x coordinate (time: O(n))
  - Compute the convex hull of each half (recursive execution)
  - Combine the two convex hulls by finding common tangents.
    Can be done in O(n)

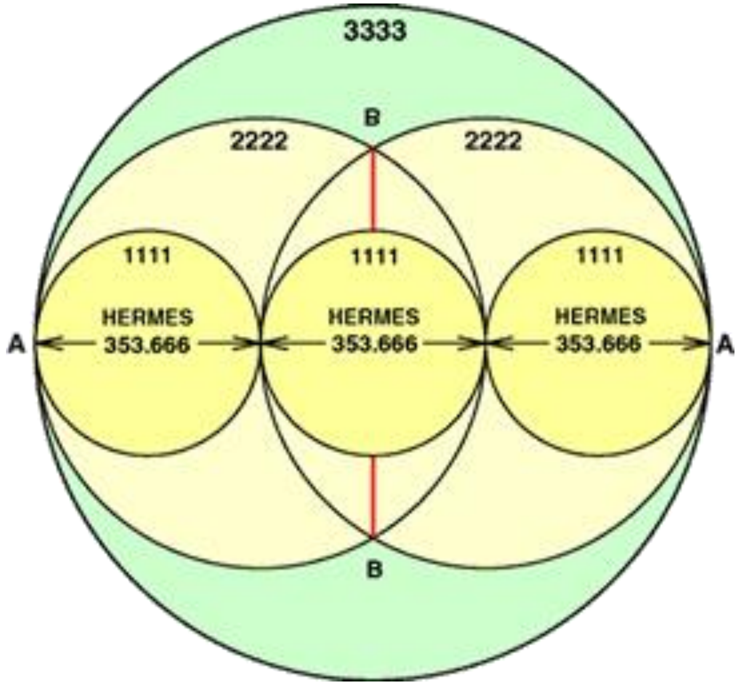


◆ Complexity: O($n$log$n$)

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

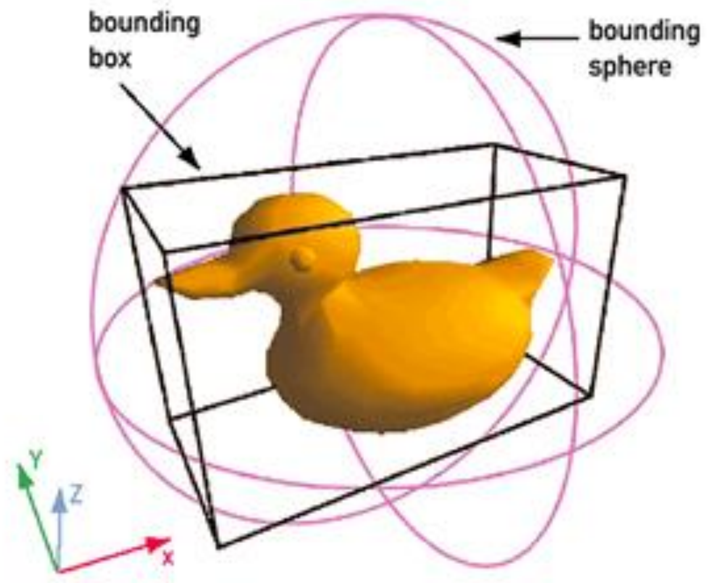
Geometry - Diameter Computation

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.

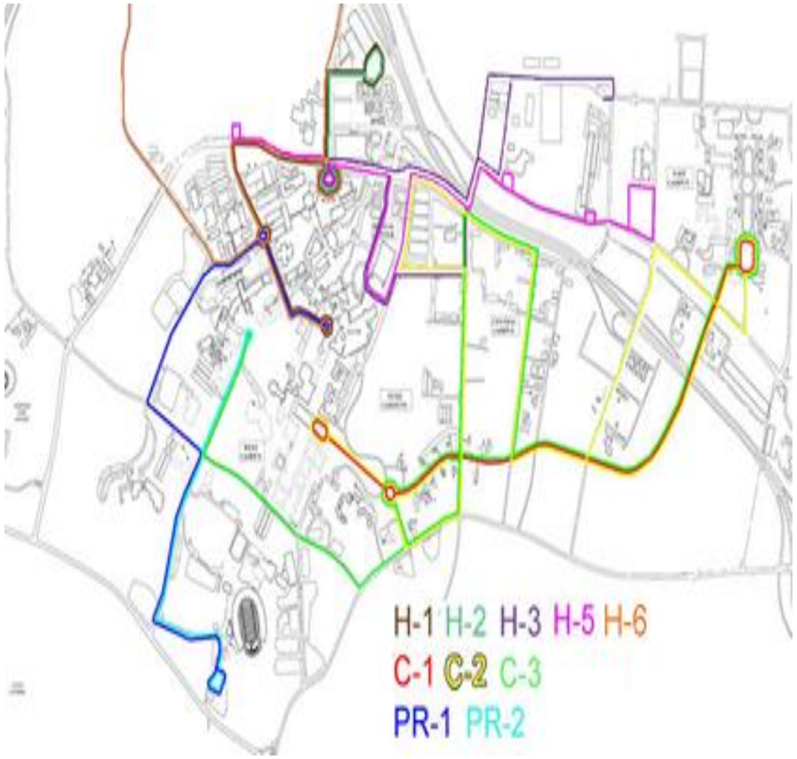
Within medical science pattern recognition creates the basis for CAD Systems (Computer Aided Diagnosis). CAD describes a procedure that supports the doctor's interpretations and findings.

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.

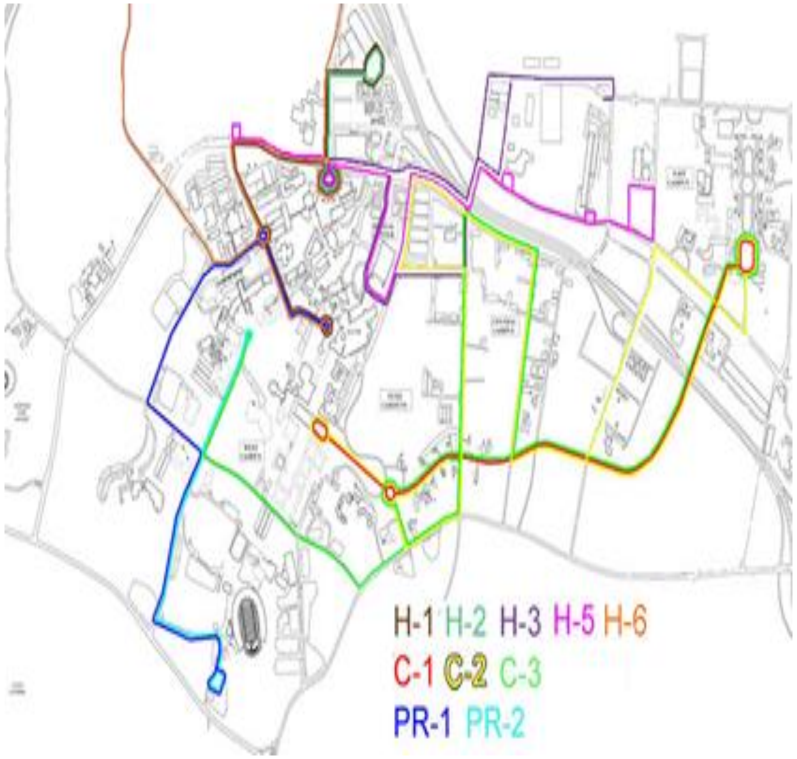
Geographical Information Systems (GIS) - Computing Accessibility Maps

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.

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

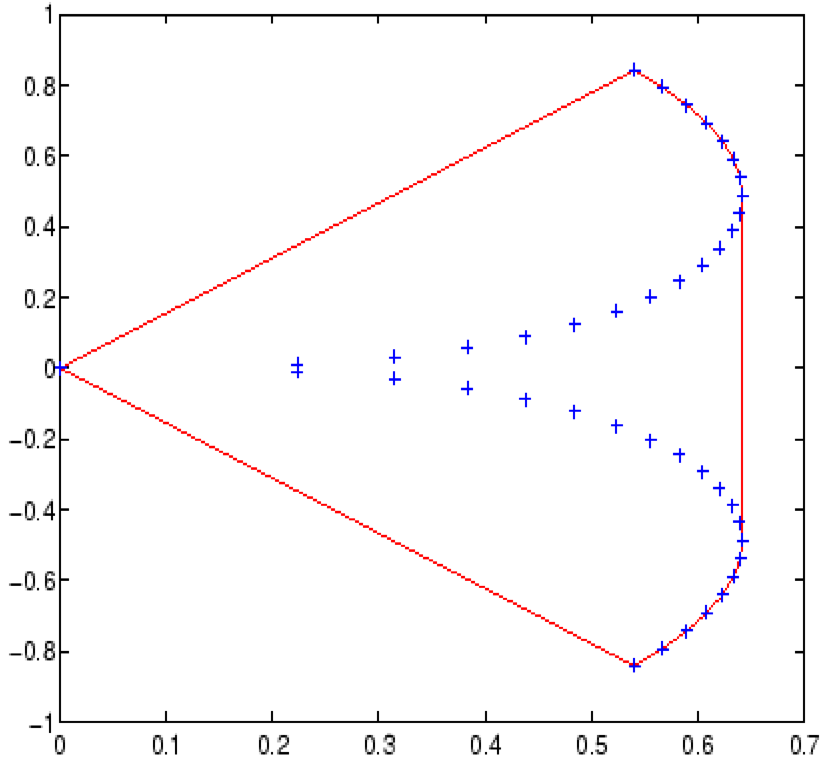
- **Syntax**
- K = convhull(X,Y)
  K = convhull(X,Y,Z)
  K = convhull(X)
  K = convhull(...,'simplify', logicalvar)
  [K,V] = convhull(...)

- **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.

- 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', logicalvar) 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+')
```


1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7