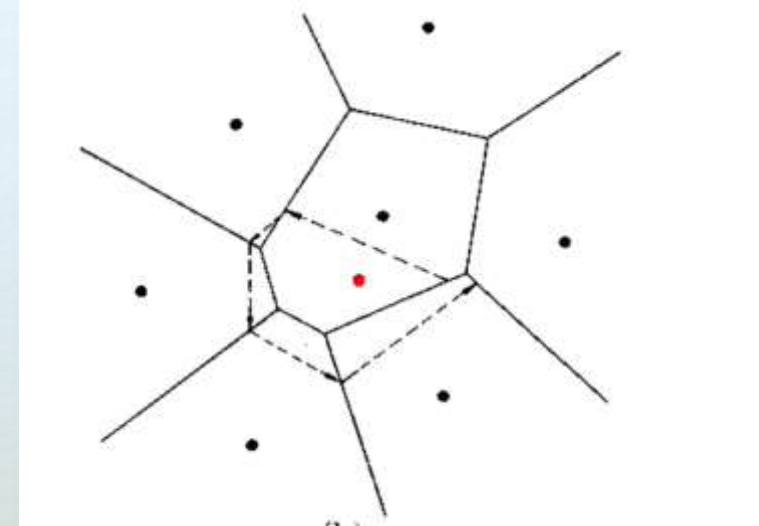
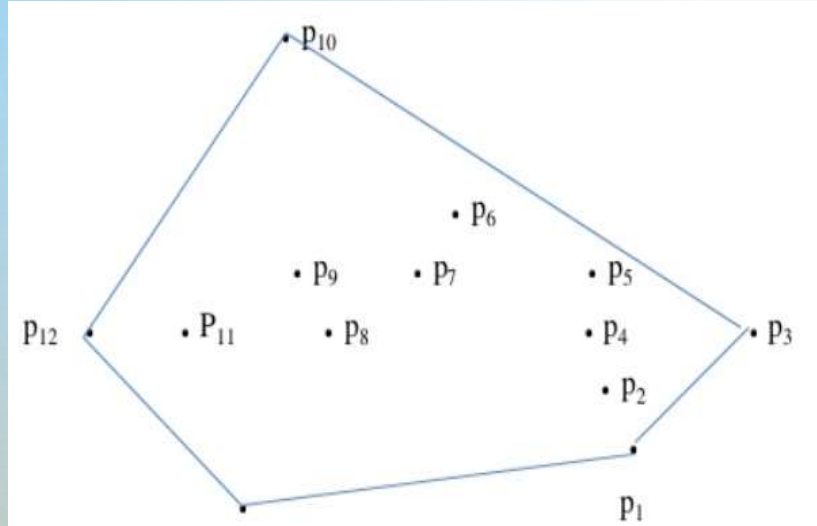


GEOMETRIC ALGORITHMS



Nhóm 3:

- Nguyễn Văn Chính – 19521287
- Nguyễn Công Đức – 19521375

KHÁI NIỆM:

Geometric Algorithms (Computational Geometry) là 1 nhánh của khoa học máy tính, gồm tập hợp các giải thuật được sử dụng để giải quyết các vấn đề hình học


Được ứng dụng rộng rãi trong các lĩnh vực:

- Computer Graphics
- Robotics
- VLSI Design
- Computer Aided Design
- Molecular Modeling...


Google cafe gần đường Kỳ Đồng

Xếp hạng ⌵ Giờ hoạt động ⌵


Unito Coffee
4,6 ★★★★★ (26) · Quán cà phê
7 Kỳ Đồng
✓ Ăn tại nhà hàng ✓ Đồ ăn mang đi




GUTA CAFE - KỶ ĐỒNG
4,4 ★★★★★ (15) · 41 · Quán cà phê
19d Kỳ Đồng
Đang mở cửa
✓ Ăn tại nhà hàng ✓ Đồ ăn mang đi




The Coffee Factory
4,0 ★★★★★ (35) · Quán cà phê
29 Kỳ Đồng
✓ Ăn tại nhà hàng ✓ Đồ ăn mang đi

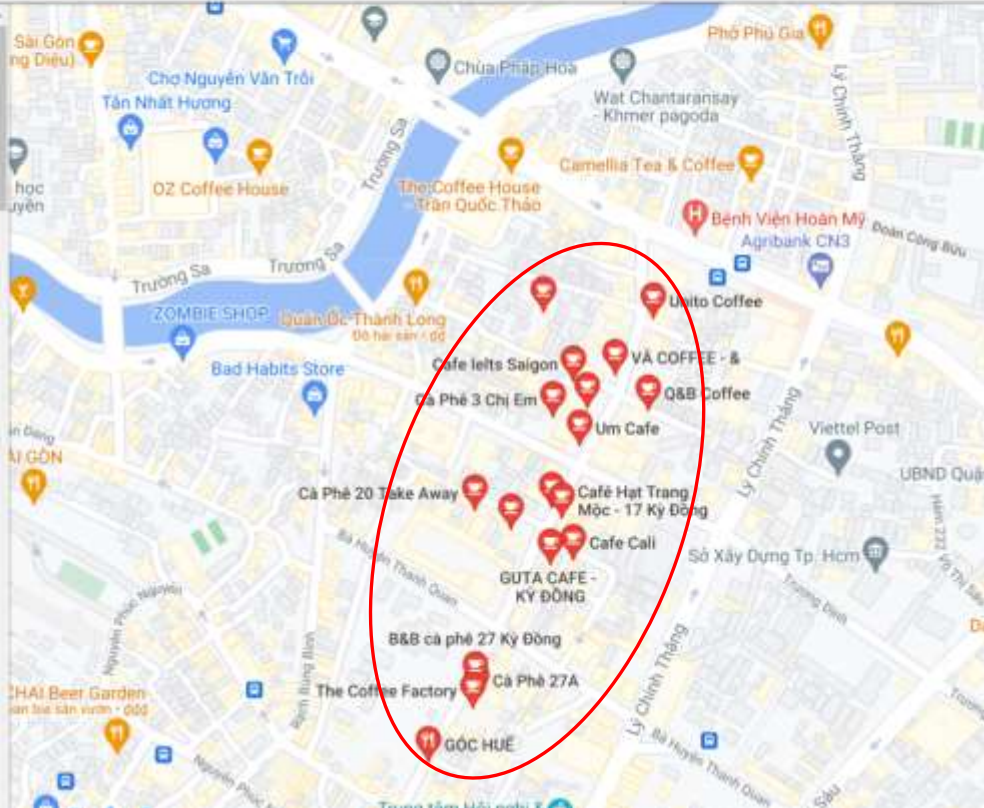


Quán Cà Phê Giải Khát Đỗ Quỳnh Nga
3,9 ★★★★★ (10) · Quán cà phê
20, Đường Kỳ Đồng
✓ Ăn tại nhà hàng



B&B cà phê 27 Kỳ Đồng
5,0 ★★★★★ (1) · Quán cà phê
27 Kỳ Đồng

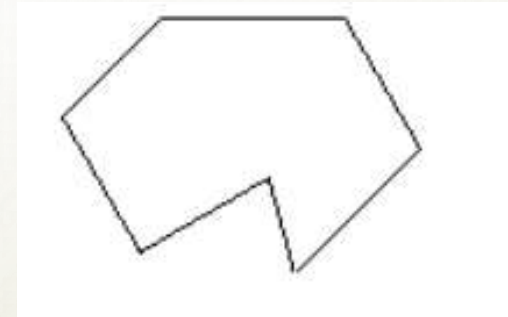
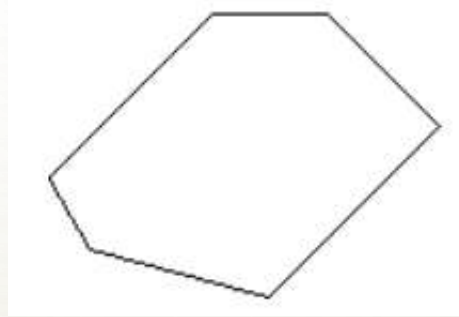
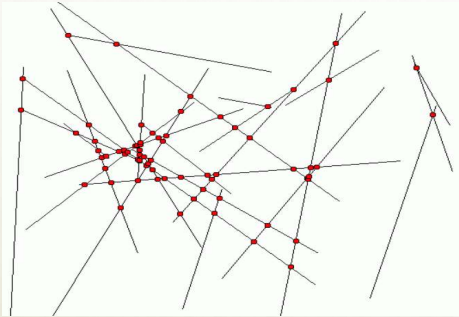




Tìm các quán cafe gần 1 địa điểm cụ thể

ĐẶC ĐIỂM BÀI TOÁN:

Input: 1 tập gồm các đối tượng hình học như điểm, đoạn thẳng, đa giác...



Output: trả lời cho các câu truy vấn về những đối tượng input (các vấn đề về đoạn thẳng giao nhau, tìm cặp điểm gần nhất, xa nhất..) hoặc các đối tượng hình học như tìm bao lồi của tập các điểm dữ liệu, tính diện tích bao lồi...

MỘT SỐ KHÁI NIỆM:

Xét phạm vi không gian 2 chiều, các đối tượng được biểu diễn trên mặt phẳng tọa độ Oxy

+Điểm: gồm 2 tọa độ theo trục hoành và trục tung

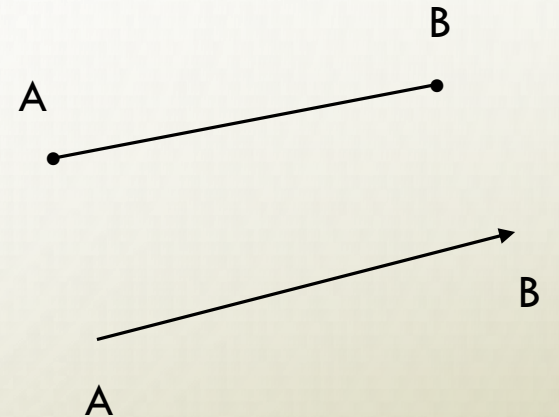
• $A(x_0, y_0)$

+Đoạn thẳng: 1 phần của đường thẳng bị giới hạn bởi 2 điểm đầu mút

+Vector: đoạn thẳng có hướng từ điểm bắt đầu đến điểm kết thúc (theo thứ tự đầu mút)

+Khoảng cách giữa 2 điểm: $d(A,B)=\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$

Trong đó $A(x_0, y_0)$, $B(x_1, y_1)$ (công thức khoảng cách Euclid)



CÔNG THỨC TÍNH DIỆN TÍCH ĐA GIÁC THEO TỌA ĐỘ:

$$S = \left| \frac{\sum (x_i - x_{i+1}) * (y_i + y_{i+1})}{2} \right|$$

Tích có hướng của 2 vector

$$\vec{a} \times \vec{b} = \hat{n} \|\vec{a}\| \|\vec{b}\| \sin\theta$$

$$\vec{a} \times \vec{b} = \begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} = x_1 y_2 - x_2 y_1$$

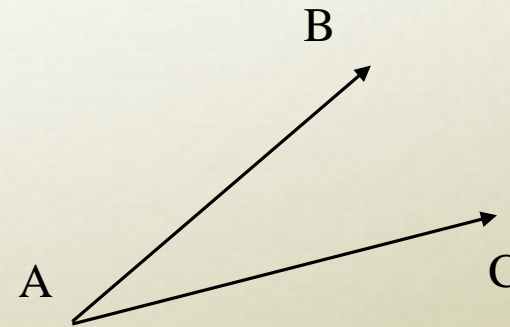
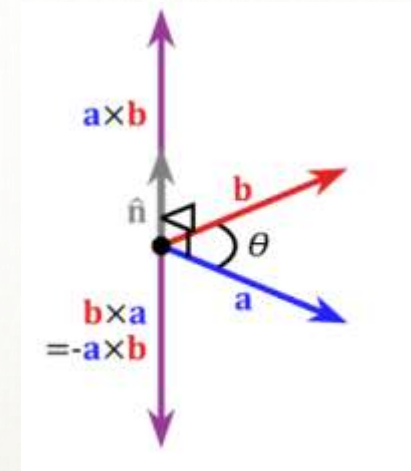
Tính

chất
 $P = \vec{a} \times \vec{b}$

- $P > 0$: a ngược chiều kim đồng hồ với b tính theo gốc tọa độ O
- $P < 0$: a cùng chiều kim đồng hồ với b tính theo gốc tọa độ O
- $P = 0$: a, b cùng phương

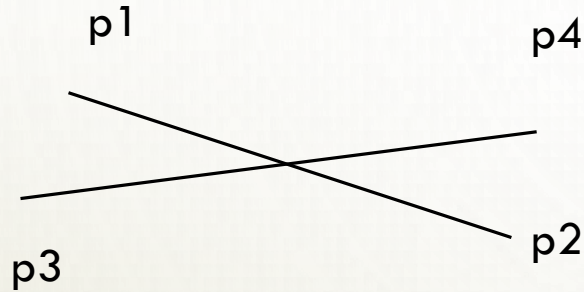
Xét 3 điểm A, B, C. Gọi $P = \text{orient}(A, B, C) = \overrightarrow{AB} \times \overrightarrow{AC}$

- $P > 0$: B nằm bên trái AC
- $P < 0$: B nằm bên phải AC
- $P = 0$: C, A, B thẳng hàng



LINE SEGMENT INTERSECTION

Kiểm tra 2 đoạn thẳng giao nhau



$$\begin{aligned}d_1 &= (p_1 - p_3) \times (p_4 - p_3), \\d_2 &= (p_2 - p_3) \times (p_4 - p_3), \\d_3 &= (p_3 - p_1) \times (p_2 - p_1), \\d_4 &= (p_4 - p_1) \times (p_2 - p_1)\end{aligned}$$

Algorithm ON-SEGMENT (p_i, p_j, p_k)

// p_i and p_j are endpoints of one line segment

// p_k is one (endpoint) of the other line segment

{ if $(\min(x_i, x_j) \leq x_k \leq \max(x_i, x_j))$

return True

else return False

}

If $((d_1 > 0 \text{ AND } d_2 < 0) \text{ OR } (d_1 < 0 \text{ AND } d_2 > 0)) \text{ AND}$ //O(1)

$((d_3 > 0 \text{ AND } d_4 < 0) \text{ OR } (d_3 < 0 \text{ AND } d_4 > 0))$

Return TRUE

elseif $d_1=0$ AND ON-SEGMENT(p_3, p_4, p_1) return TRUE; //O(1)

elseif $d_2=0$ AND ON-SEGMENT(p_3, p_4, p_2) return TRUE; //O(1)

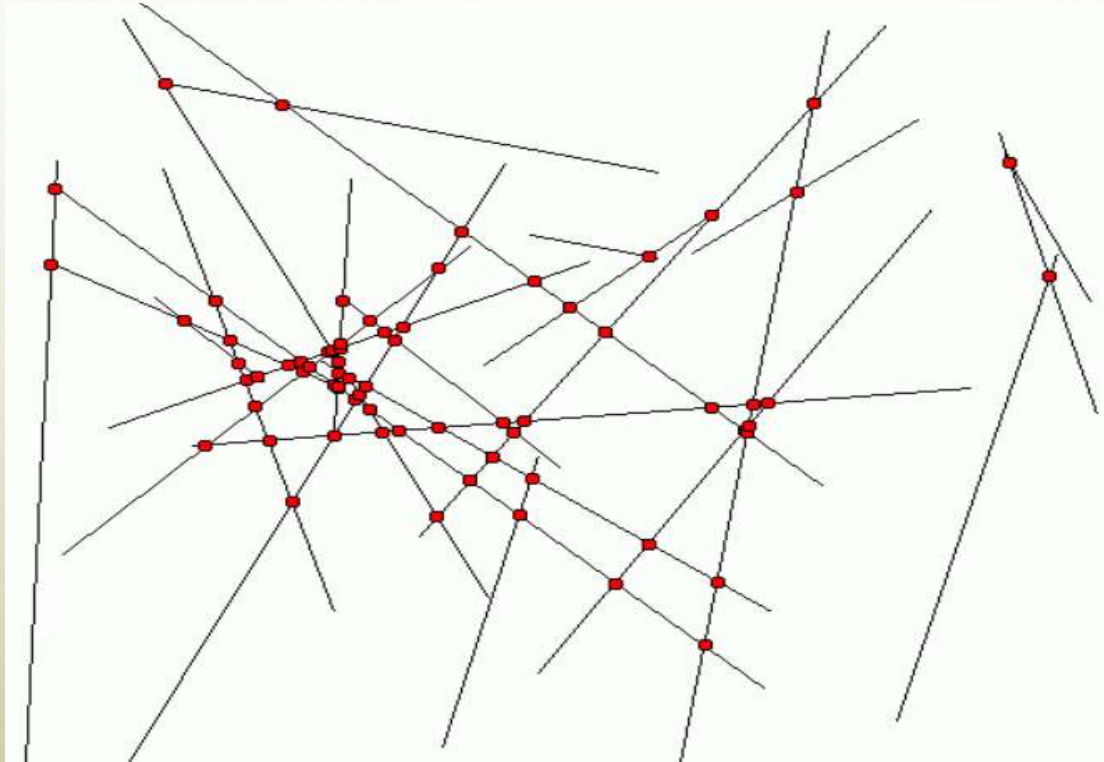
elseif $d_3=0$ AND ON-SEGMENT(p_1, p_2, p_3) return TRUE; //O(1)

elseif $d_4=0$ AND ON-SEGMENT(p_1, p_2, p_4) return TRUE; //O(1)

else return FALSE;

BÀI TOÁN:

Cho các đoạn thẳng trên mặt phẳng Oxy, mỗi đoạn thẳng gồm thông tin tọa độ của 2 điểm đầu mút. Kiểm tra tồn tại cặp đoạn thẳng giao nhau?




Sử dụng kĩ thuật plane sweep

SẮP XẾP DỮ LIỆU:

Chọn quét từ trái sang phải

 Sắp xếp các endpoint theo thứ tự hoành độ

$$A < B \Leftrightarrow (x_A < x_B) \text{ or } (x_A = x_B \text{ and } y_A < y_B)$$

 Trong 1 đoạn thẳng đầu mút có hoành độ nhỏ hơn là left endpoint, ngược lại là right endpoint

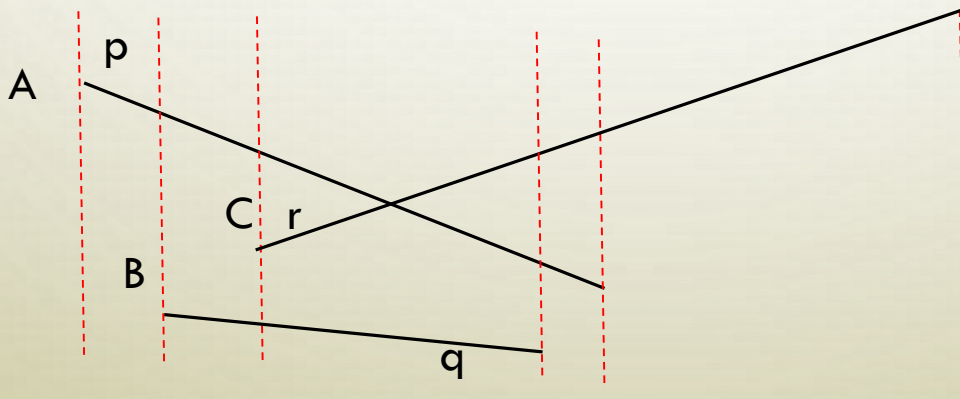
Sử dụng merge sort để sắp xếp: $O(n \log n)$

CẤU TRÚC LƯU TRỮ TRẠNG THÁI QUÉT:

Lưu trữ các đoạn thẳng đang xét và mối liên hệ giữa các đoạn thẳng

+Quét đến left endpoint của 1 đoạn thẳng => Thêm đoạn thẳng vào

+Quét đến right endpoint của 1 đoạn thẳng => Xóa đoạn thẳng

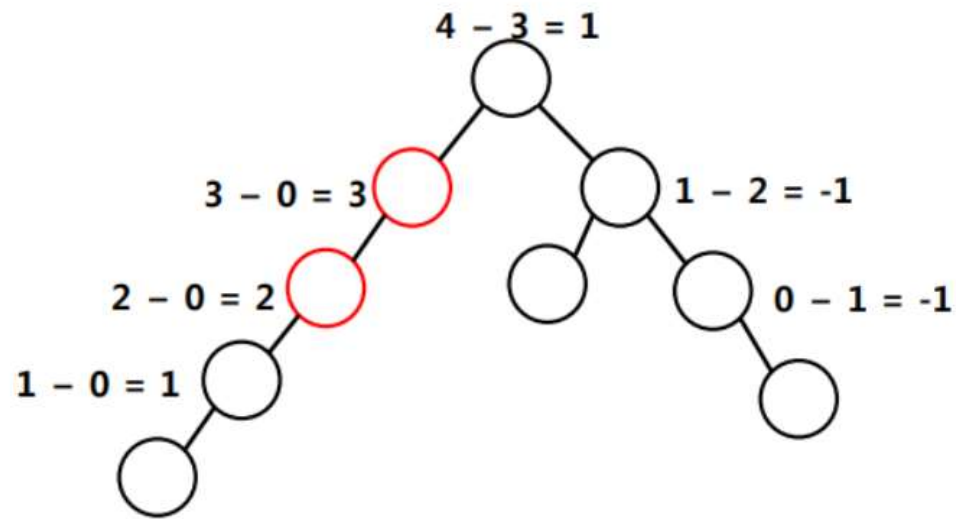


➡ **Quy tắc:** so sánh dựa trên tung độ của left endpoint

➡ $p > q \Leftrightarrow y_A > y_B$ ➡ $p > r > q$



Sử dụng cấu trúc **cây cân bằng (AVL)** để lưu trữ



Mọi nút trong v trong một cây AVL đều thỏa mãn:

$$|height(L[v]) - height(R[v])| \leq 1$$

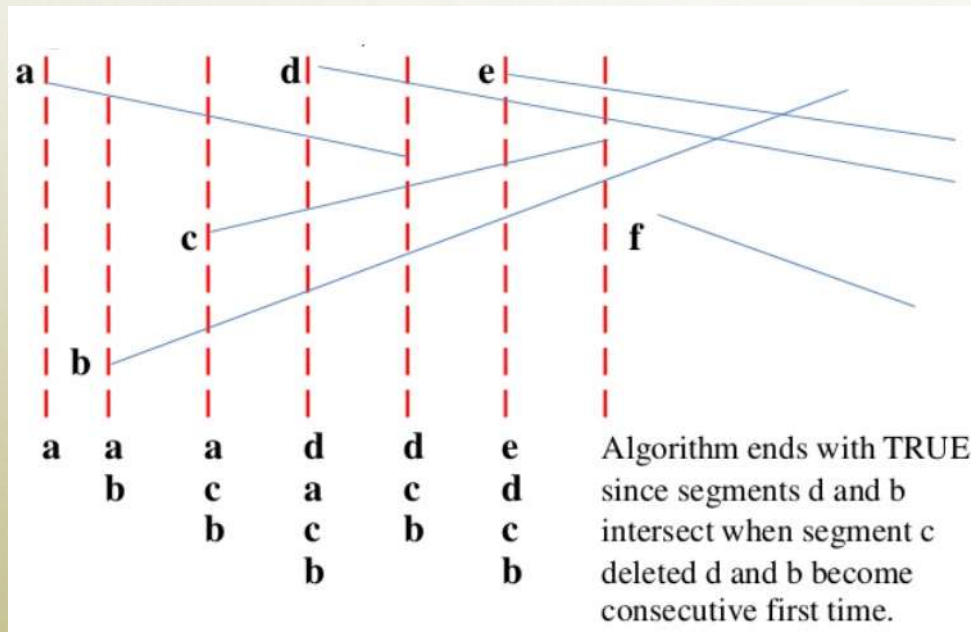
THUẬT TOÁN:

- $\text{INSERT}(T,s)$: thêm đoạn s vào cây
- $\text{DELETE}(T,s)$: xóa đoạn s khỏi cây
- $\text{ABOVE}(T,s)$: trả về đoạn thẳng phía trên gần nhất
- $\text{BELOW}(T,s)$: trả về đoạn thẳng phía dưới gần nhất

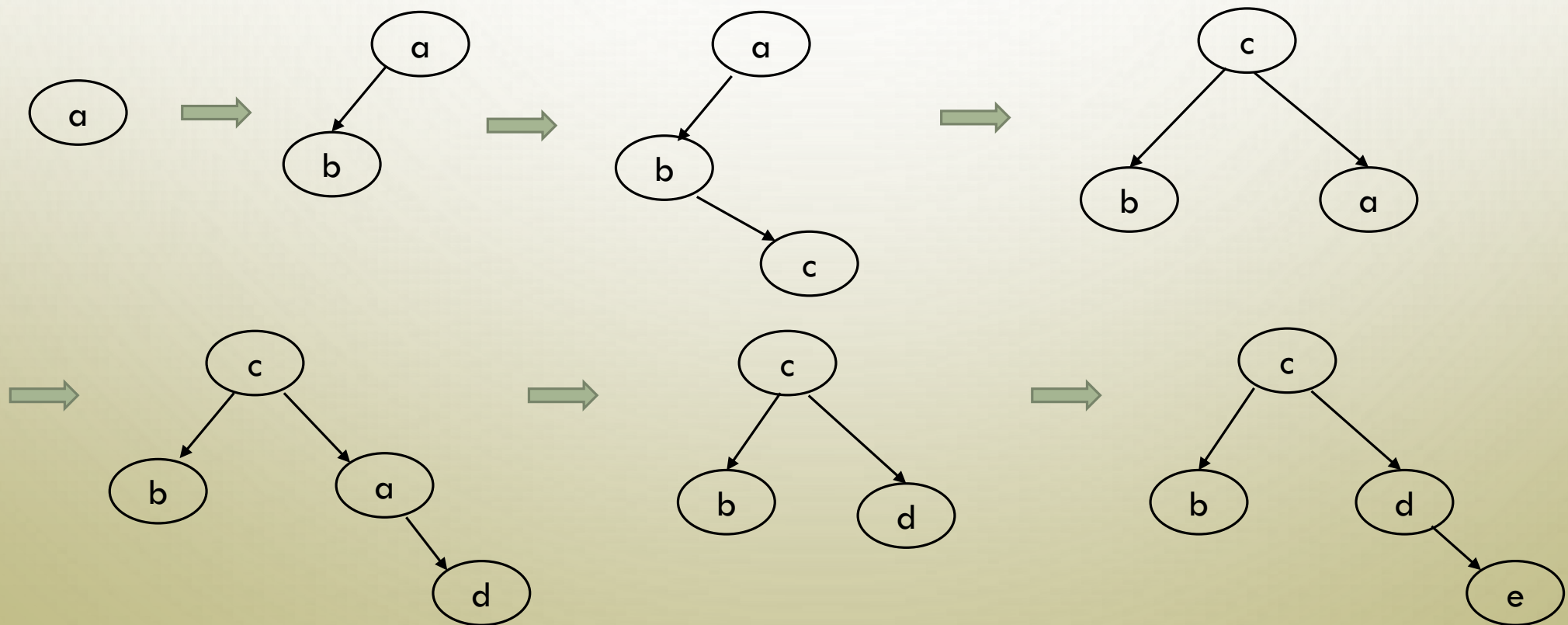
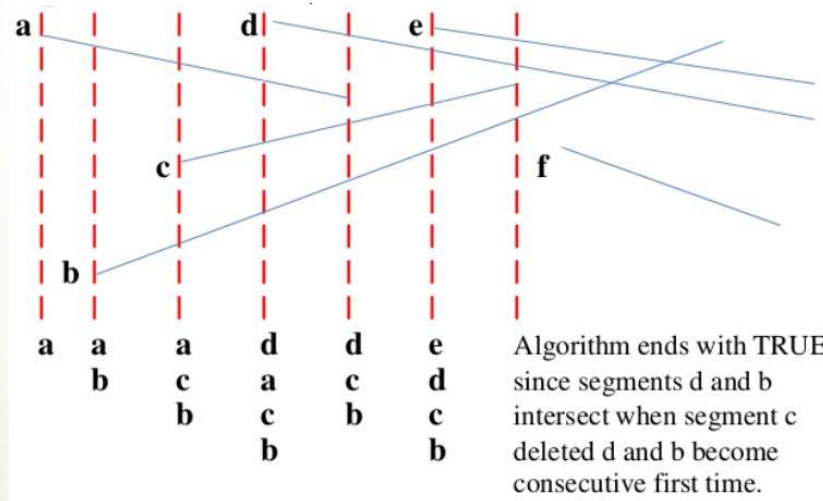
```

for each point p in the sorted list of endpoints
  if p is the left endpoint of a segment s
  {
    INSERT (T, s) // O(logn)
    if (ABOVE(T, s) exists and intersects s) OR
       (BELOW(T, s) exists and intersects s)
      return TRUE; // O(1)
  }
  if p is the right endpoint of a segment s
  {
    if both (ABOVE(T, s) and (BELOW(T, s) exists
      AND (ABOVE(T, s) intersect (BELOW(T, s)
      return TRUE; // O(1)
    DELETE (T, s) // O(logn)
  }
return FALSE;
}

```



- Add a
- Add b. Check a intersect b?
- Add c. Check c intersect a? Check c intersect b?
- Add d. Check d intersect a?
- Check d intersect c?
- Delete a
- Add e. Check e intersect d?
- Check d intersect b? => Return True



ĐỘ PHỨC TẠP THUẬT TOÁN:

- Độ phức tạp sắp xếp dữ liệu ban đầu: $O(n \log n)$
- Độ phức tạp chèn phần tử vào cây: $O(\log n)$
- Độ phức tạp xóa phần tử khỏi cây: $O(\log n)$
- Độ phức tạp kiểm tra giao nhau: $O(1)$



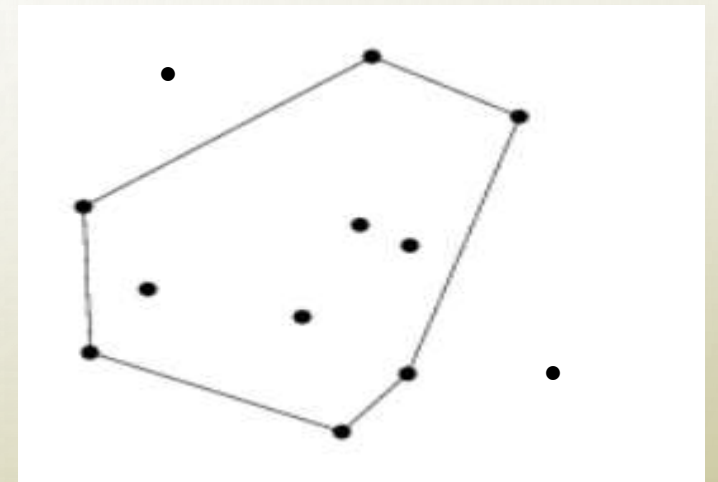
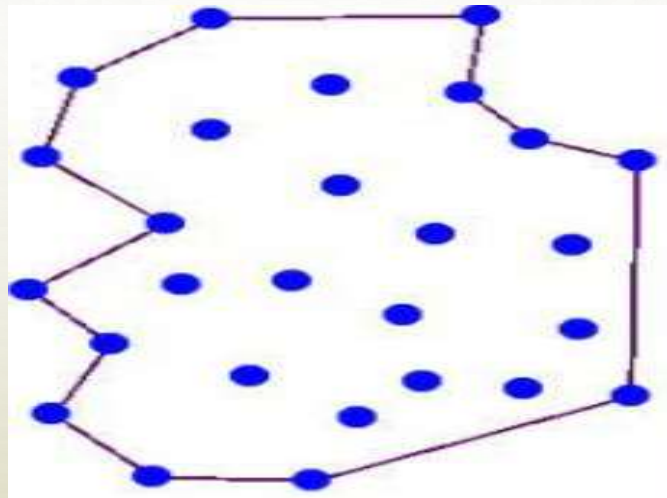
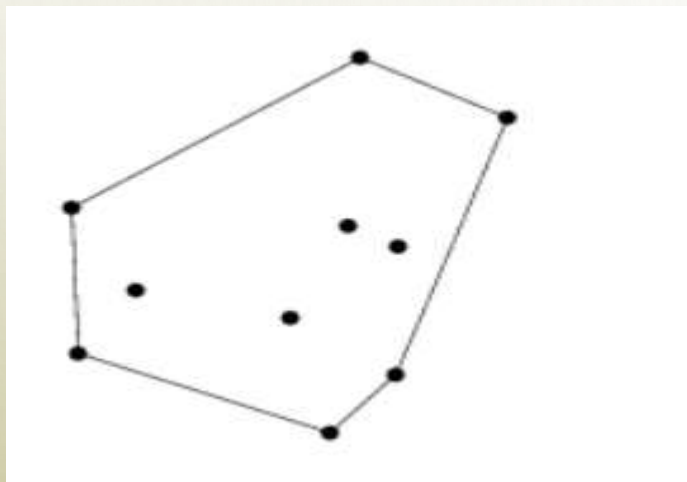
Độ phức tạp thời gian: $O(n \log n)$

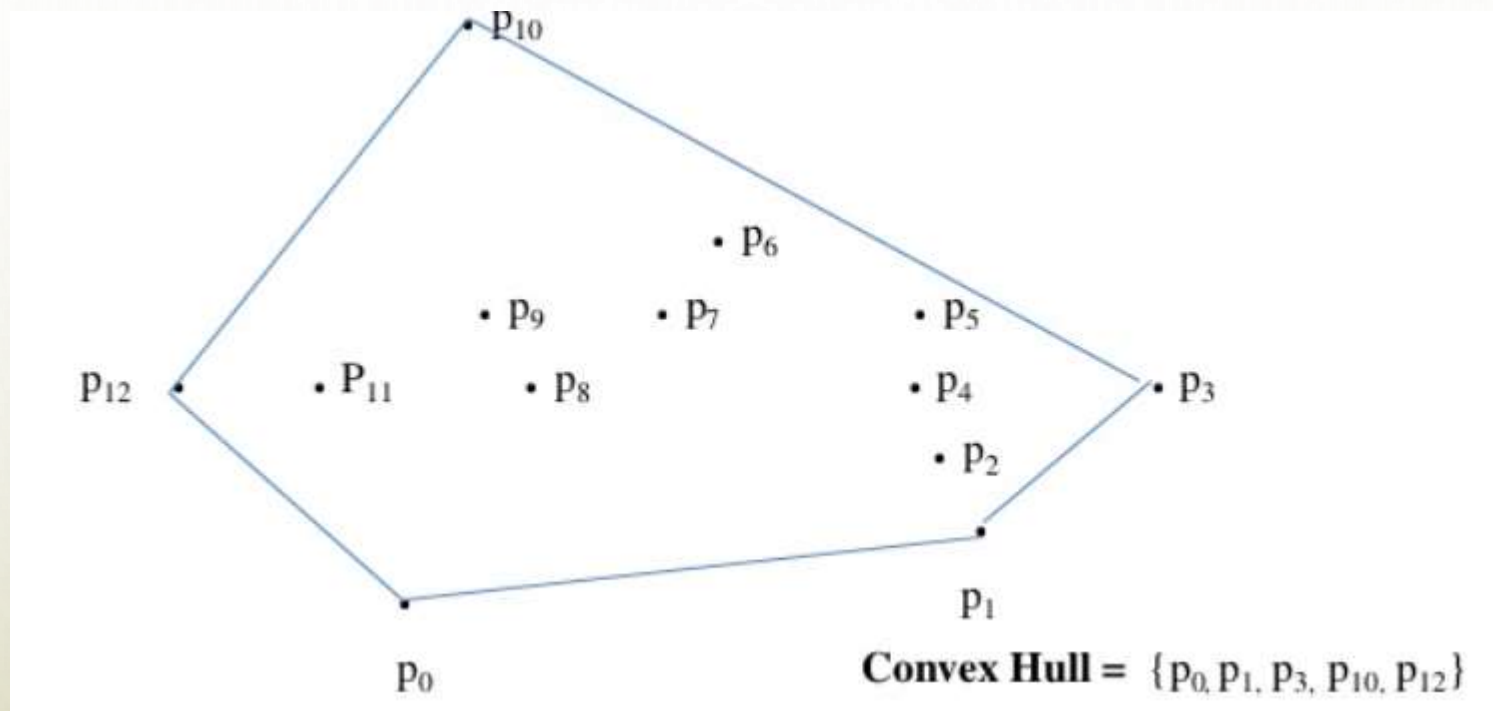


Độ phức tạp không gian: $O(n)$ (của AVL)

CONVEX HULL (BAO LÒI)

Bao lồi của 1 tập Q điểm là đa giác lồi nhỏ nhất được tạo một số điểm trong Q sao cho không có điểm nào trong Q nằm ngoài đa giác

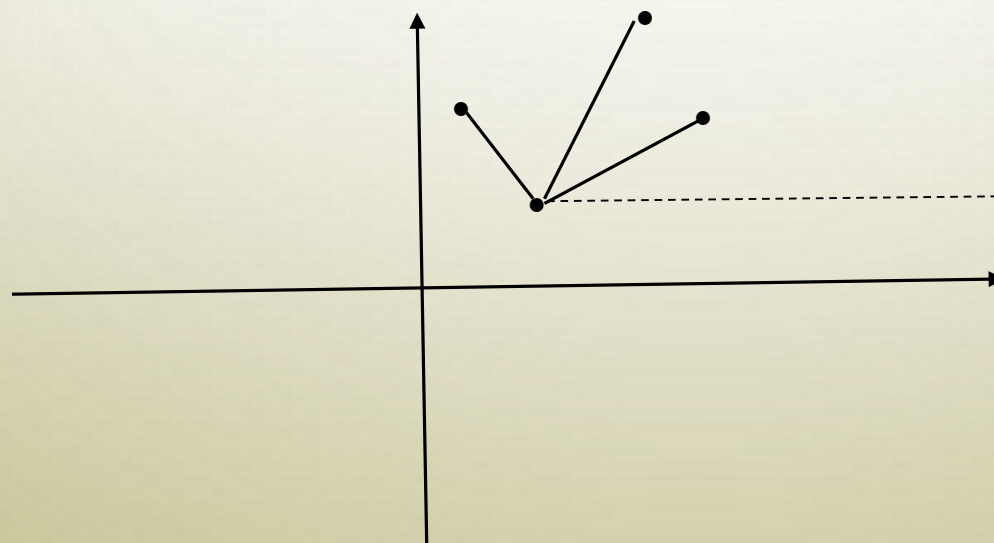




Bao lồi gồm 5 điểm: $p_0, p_1, p_3, p_{10}, p_{12}$

THUẬT TOÁN GRAHAM-SCAN

- Đầu tiên, ta xác định điểm có tung độ nhỏ nhất. Nếu có nhiều điểm cùng tung độ thì chọn điểm nằm bên trái nhất (có hoành độ nhỏ nhất). Đặt điểm này làm điểm gốc O
- Sắp xếp các điểm còn lại theo thứ tự dựa trên góc tạo bởi trục hoành và \overrightarrow{OI} với I là điểm đang xét



Sử dụng merge sort để sắp xếp thứ tự các điểm

- Gọi H là bao lồi. Điểm thứ i được thêm vào lồi sẽ được gọi là H_i
- Thêm 3 điểm đầu tiên vào bao lồi. Gán $h=3$ và $P=H_3$
- Với mỗi điểm I tiếp theo:

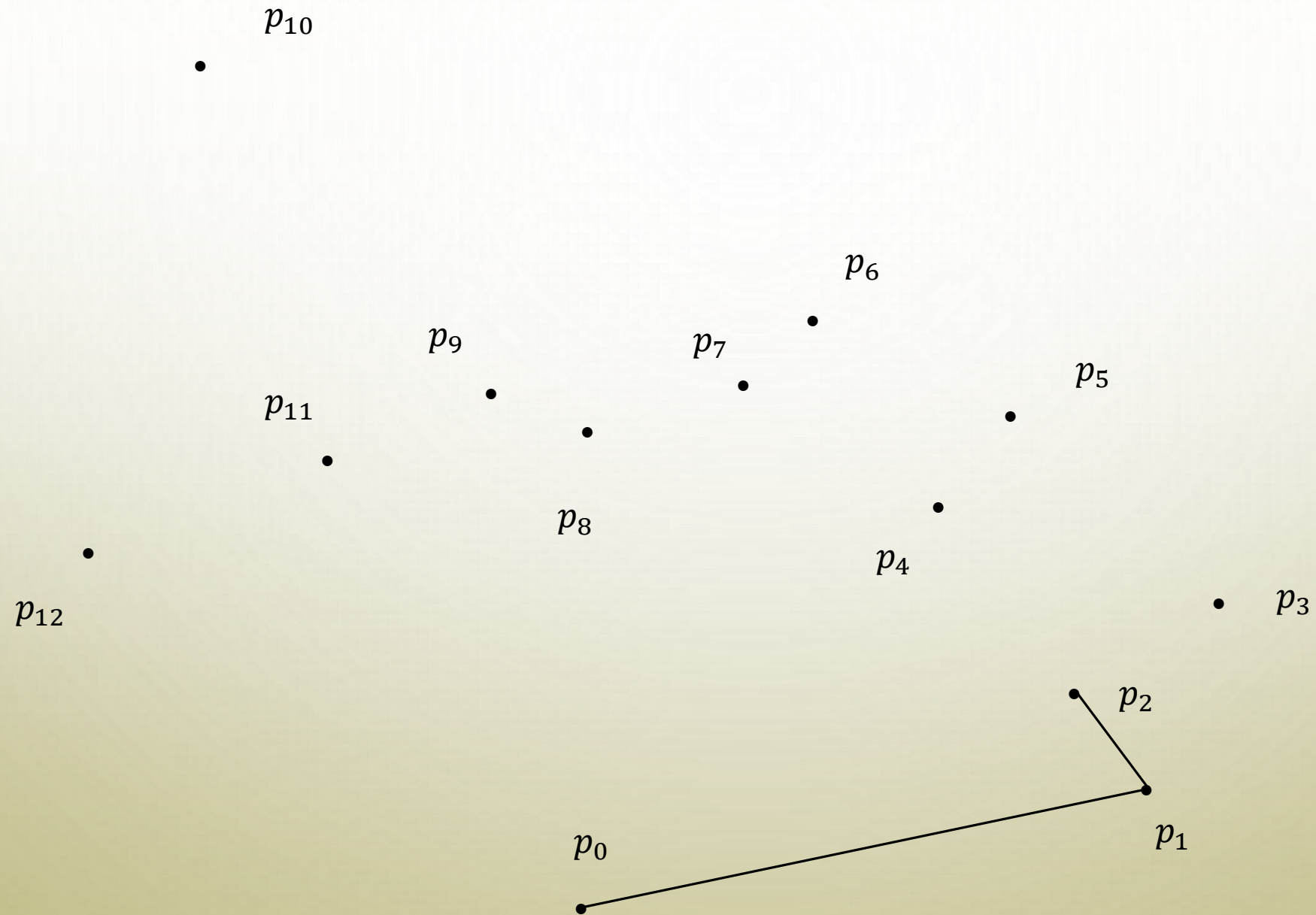
+Xét 3 điểm I, H_h, H_{h-1} . Ta gọi 2 vector $\vec{u} = \overrightarrow{H_{h-1}H_h}$ và $\vec{v} = \overrightarrow{H_hI}$.

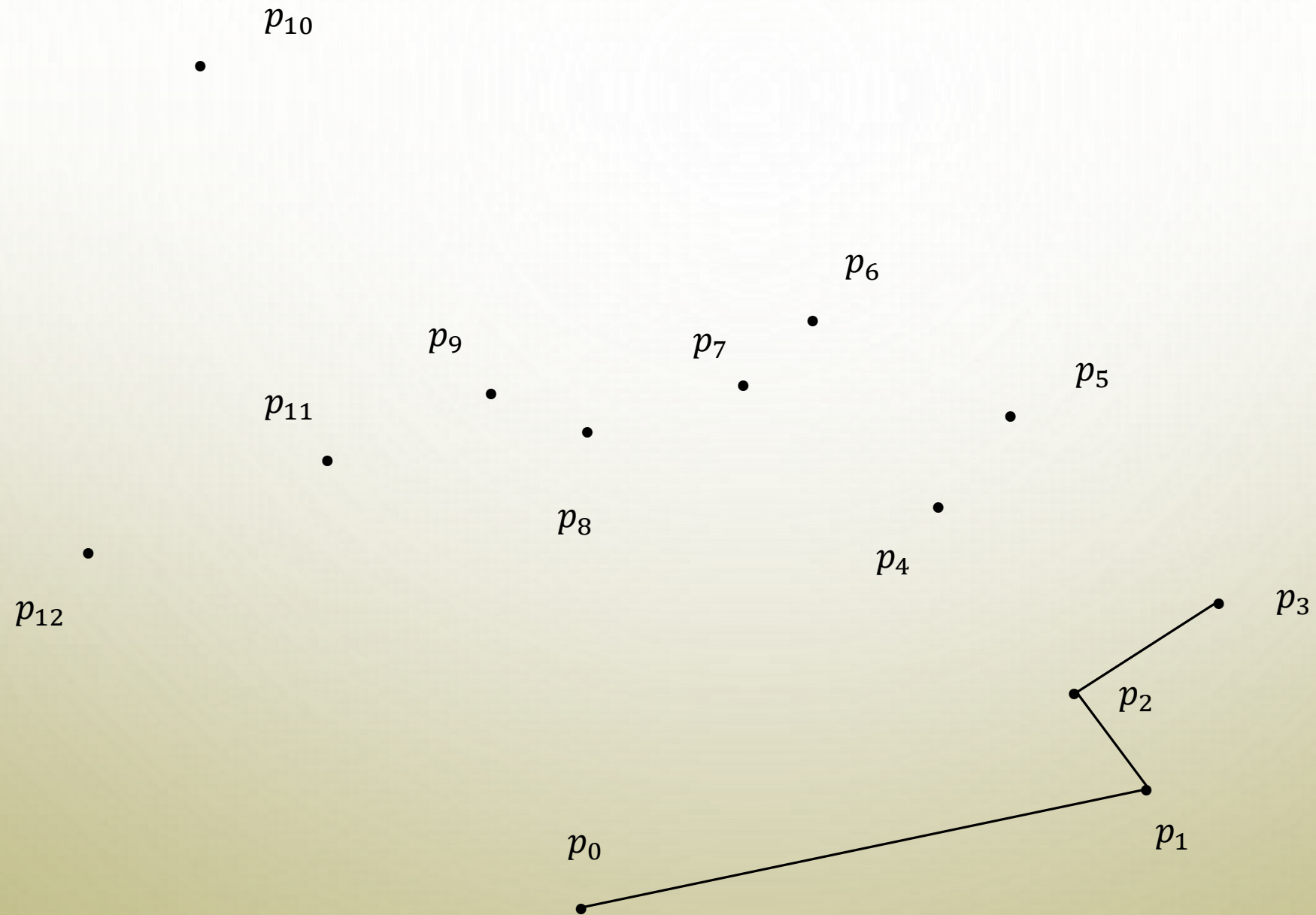
+Xét $P = \vec{u} \cdot \vec{v}$:

- Nếu $P > 0$ thì 3 điểm đều thuộc bao lồi, tăng h lên 1 và thêm I vào cuối bao lồi ($H_h = I$).
- Nếu $P < 0$ thì góc $\widehat{H_{h-1}H_hI}$ tạo ra đa giác lõm và điểm H_h phải bị loại bỏ. H_h sẽ được đặt là I .



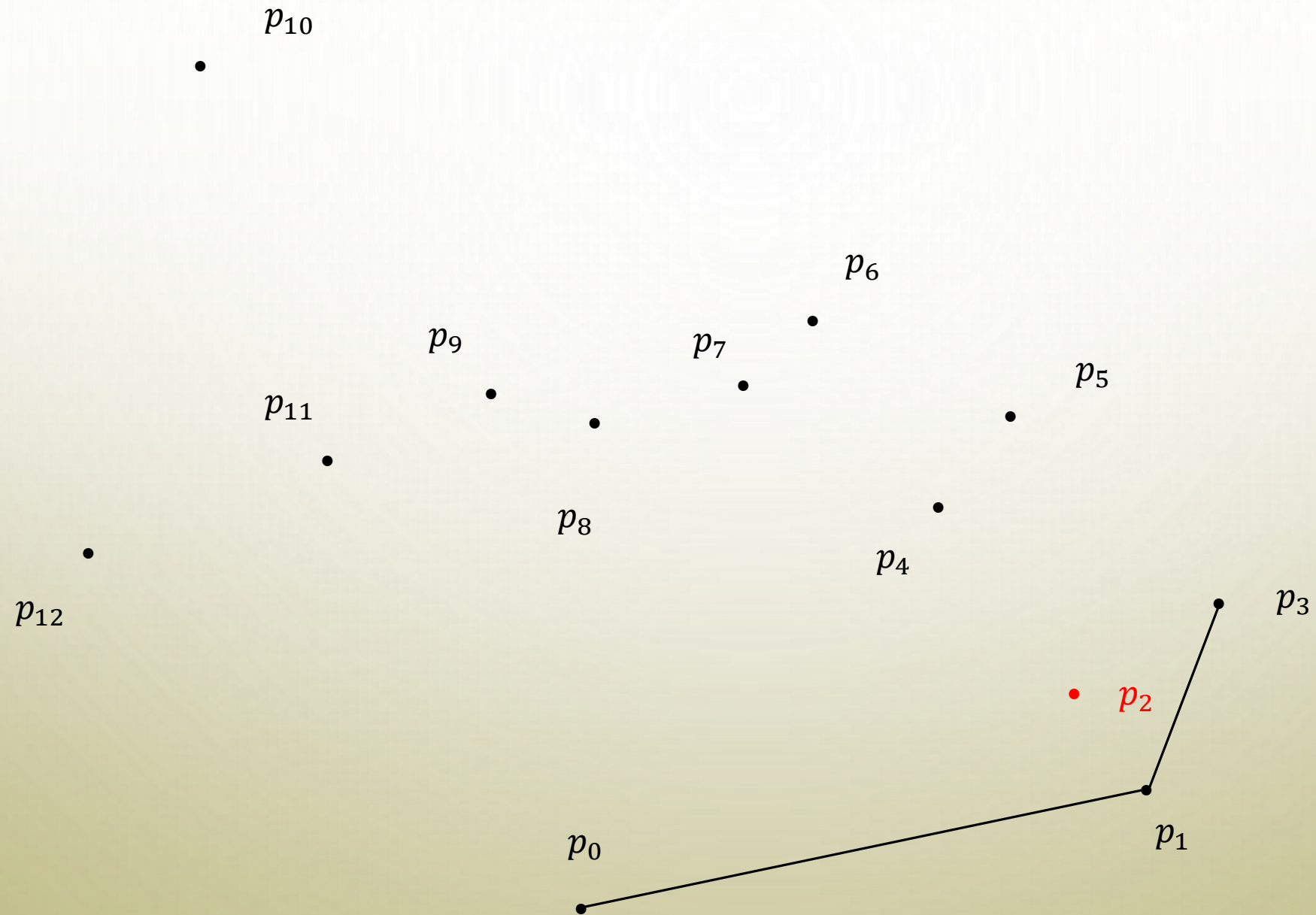
Sử dụng cấu trúc **stack** lưu trữ bao lồi



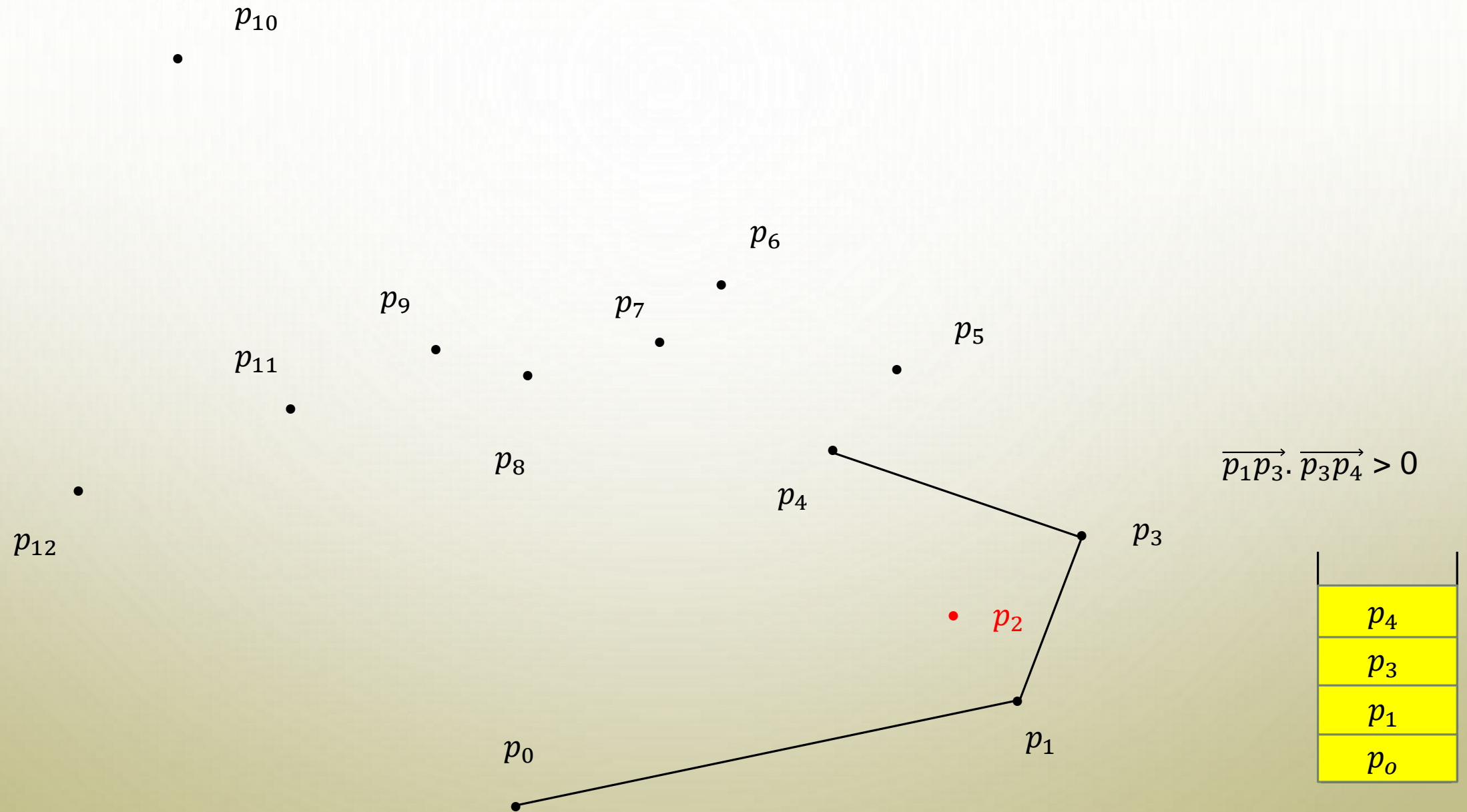


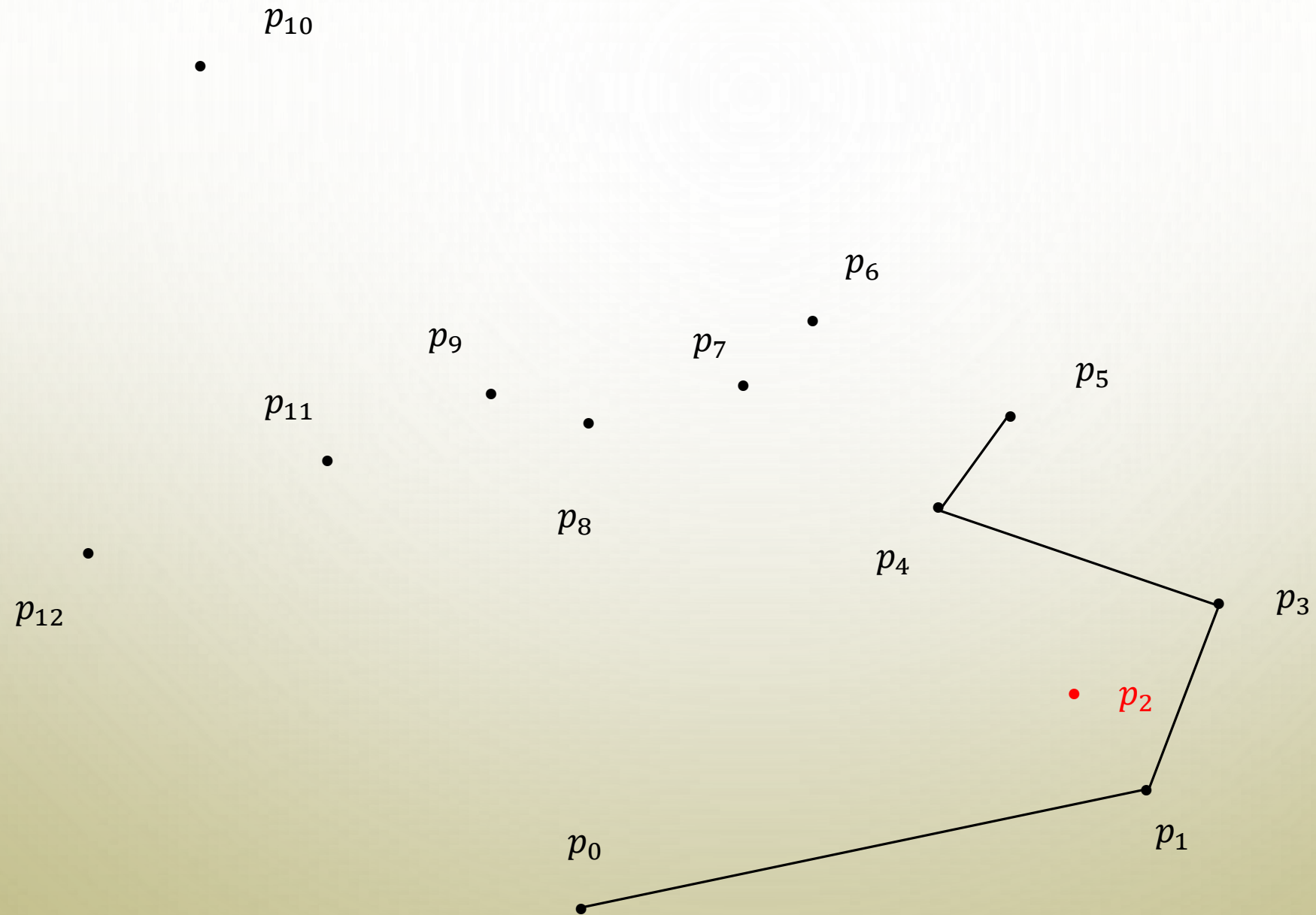
$$\overrightarrow{p_1 p_2} \cdot \overrightarrow{p_2 p_3} < 0$$

p_2
p_1
p_o

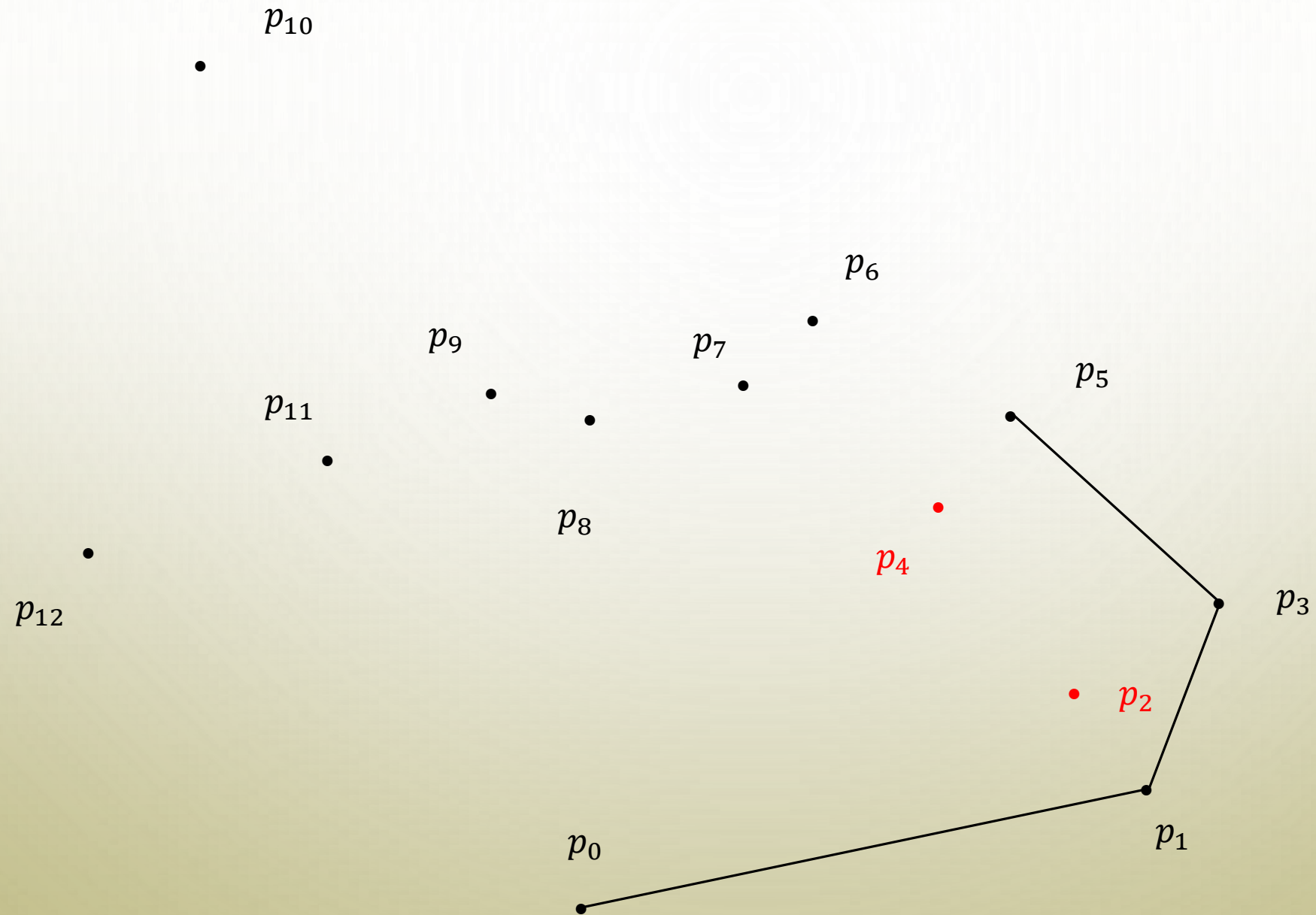


p_3
p_1
p_o

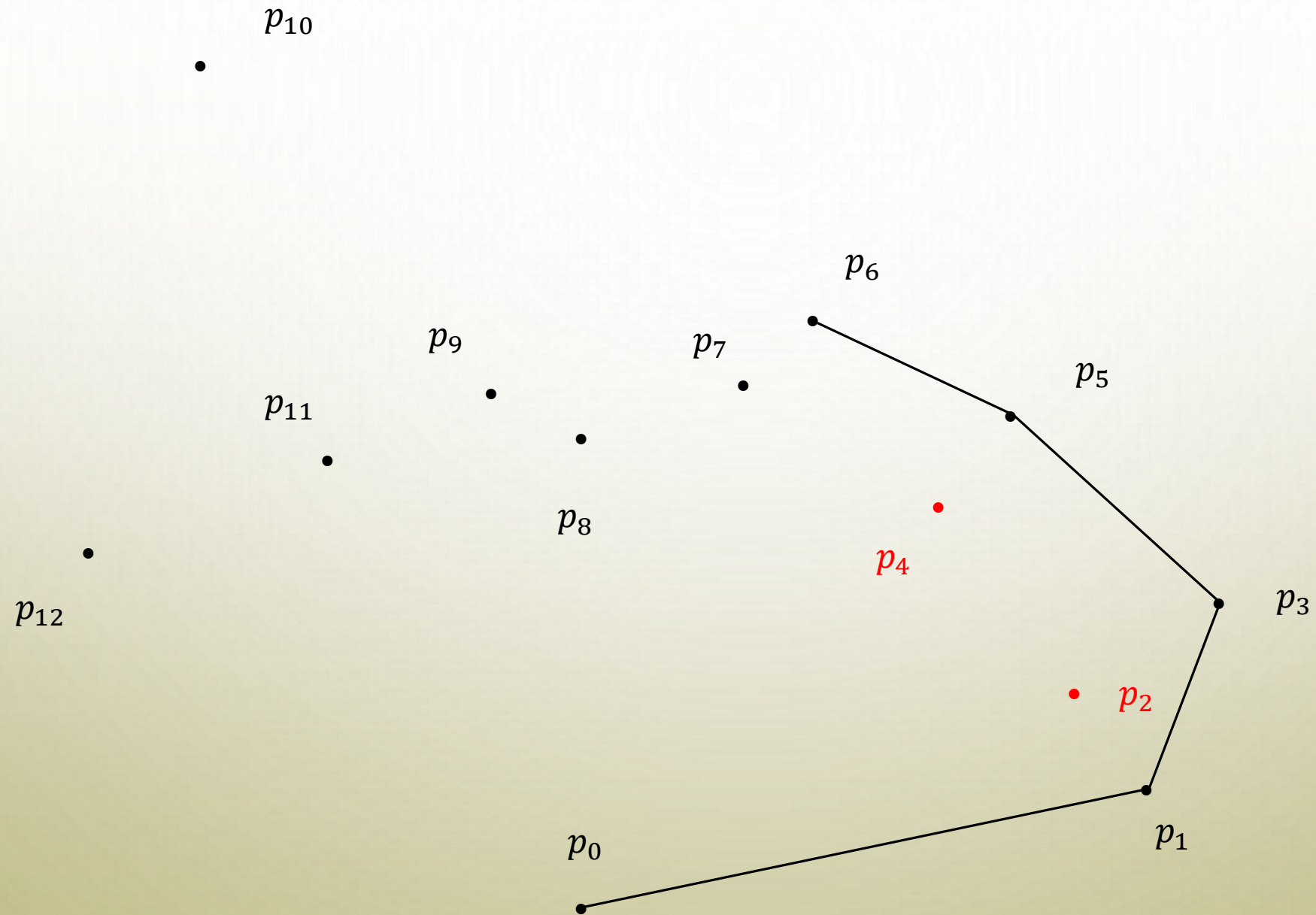


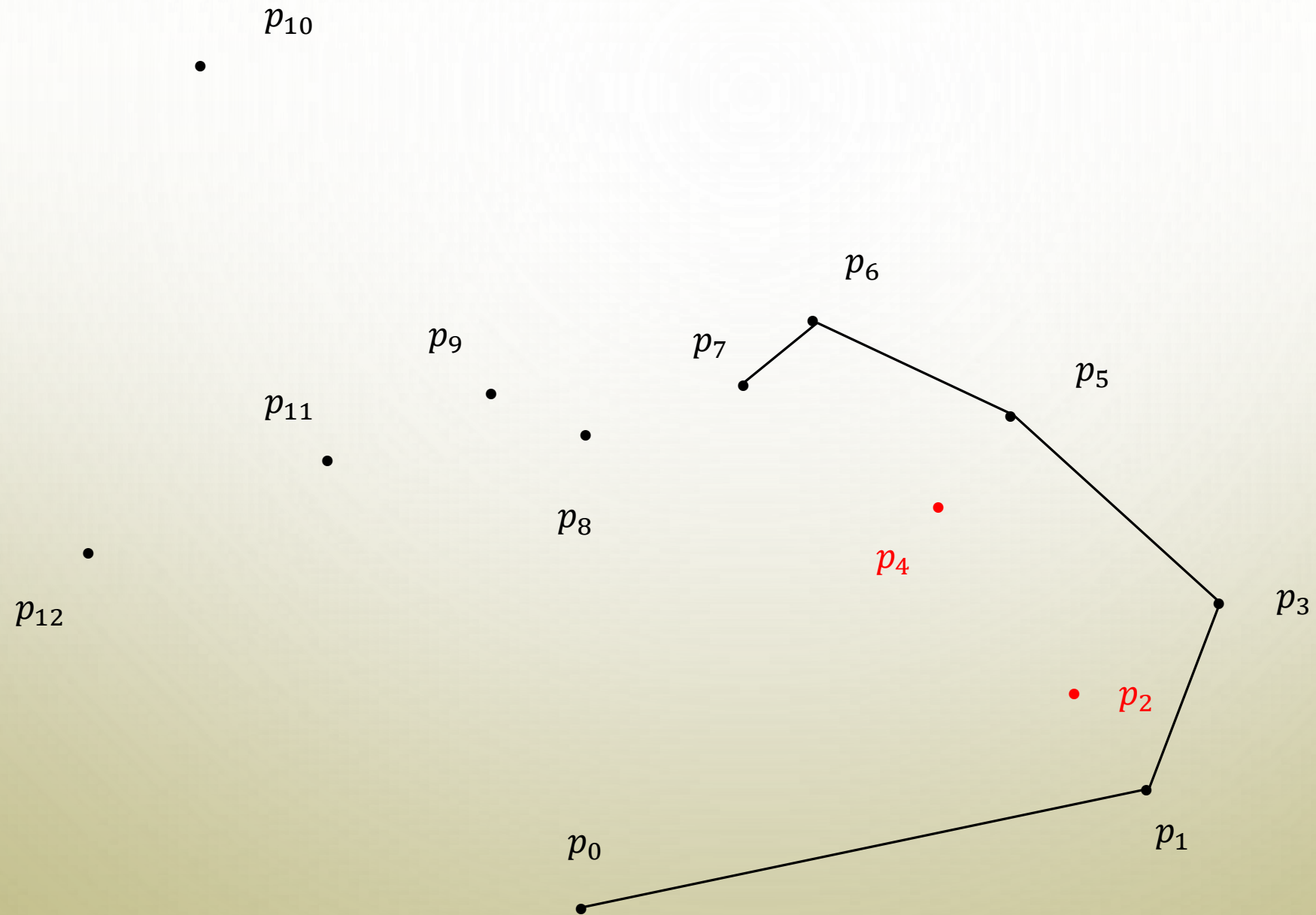


p_4
p_3
p_1
p_0

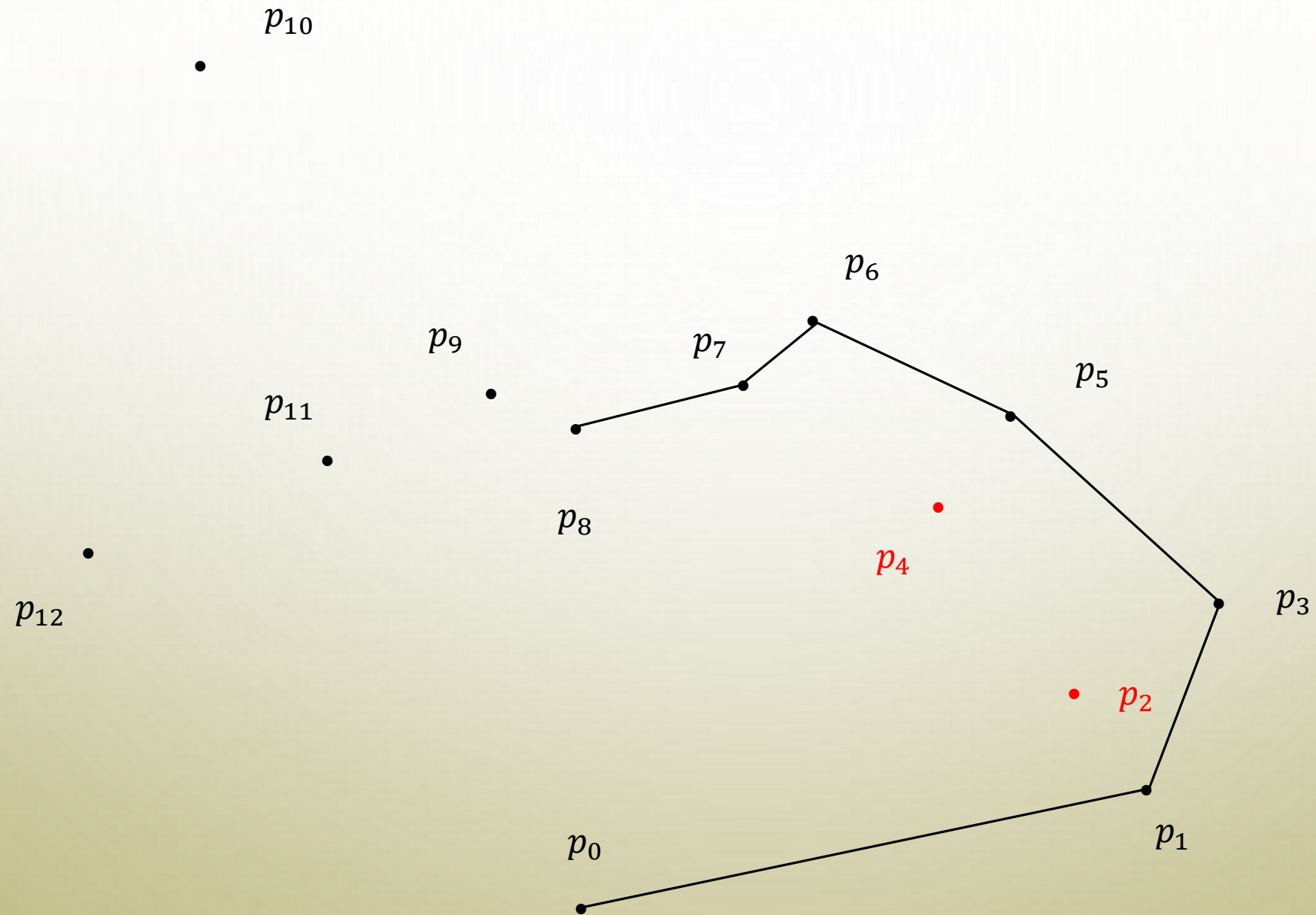


p_5
p_3
p_1
p_o

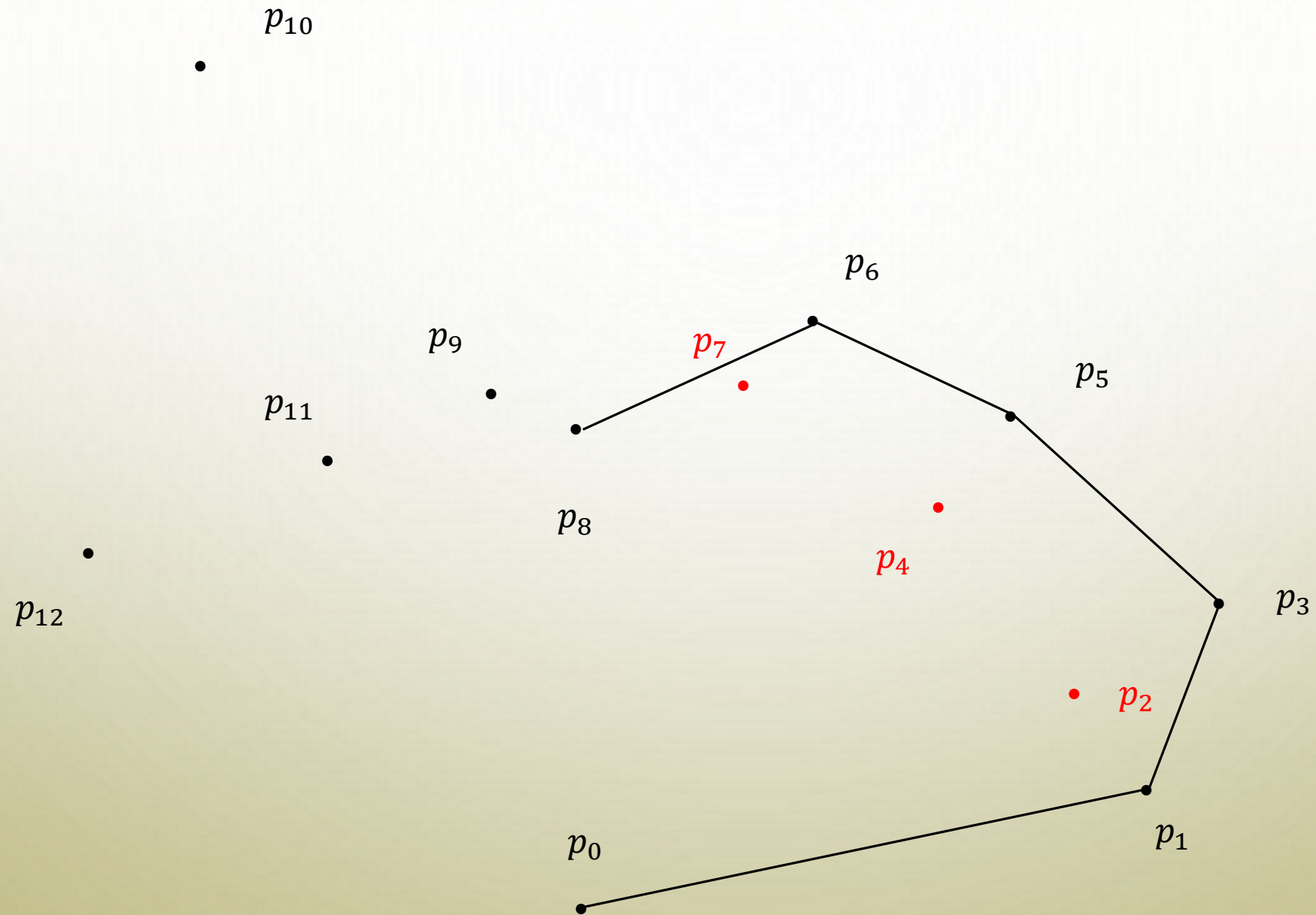


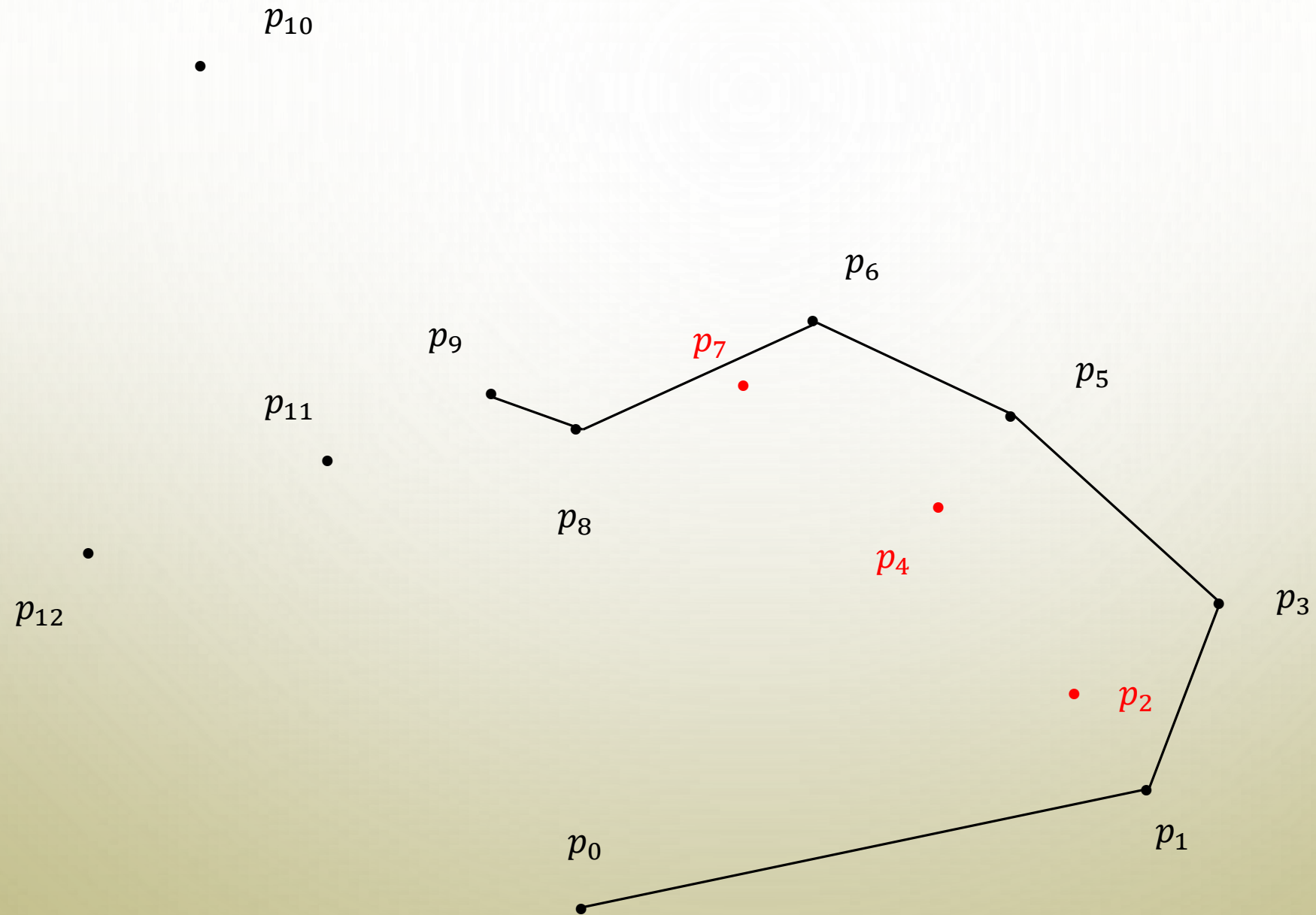


p_7
p_6
p_5
p_3
p_1
p_0

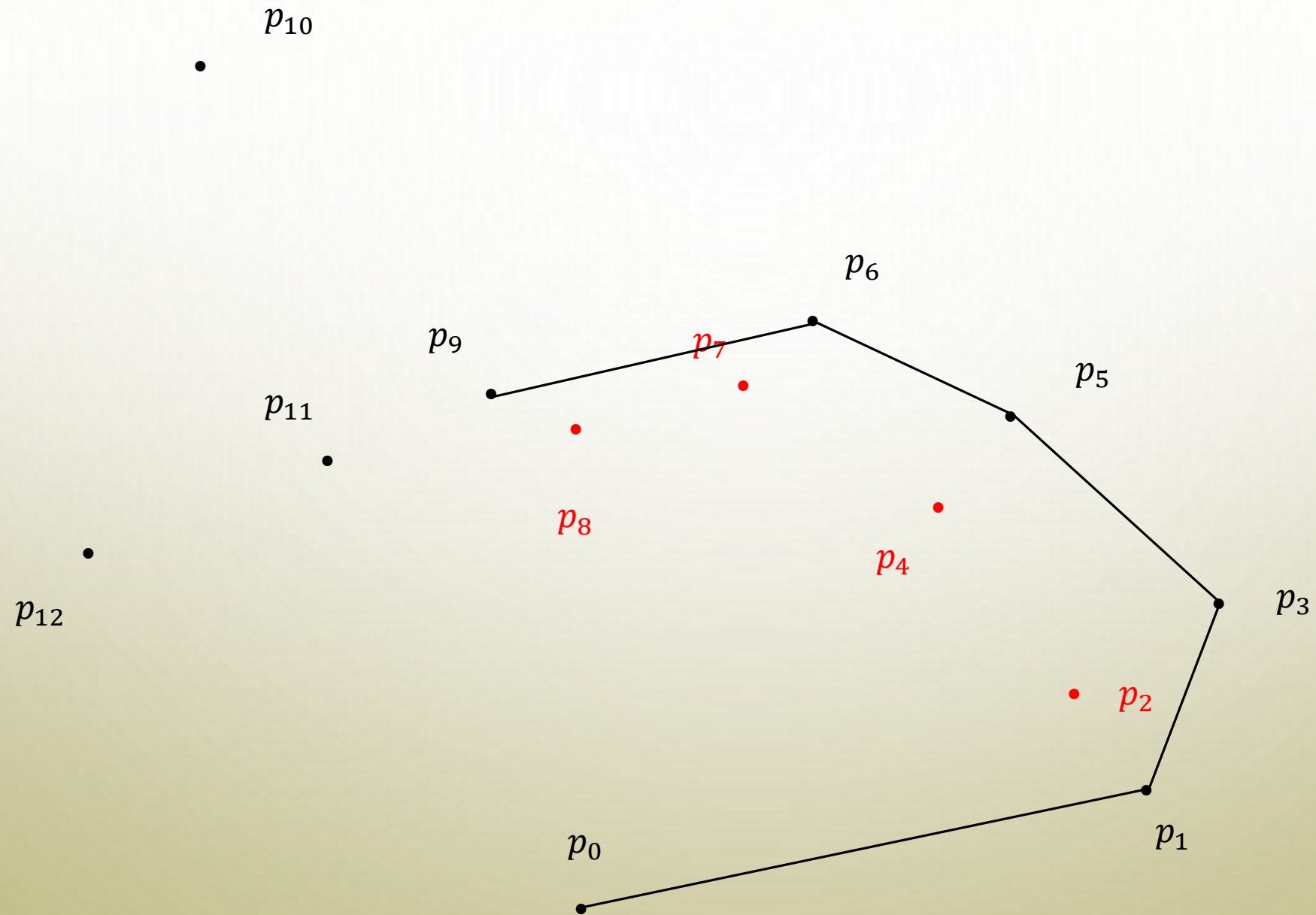


p_7
p_6
p_5
p_3
p_1
p_0

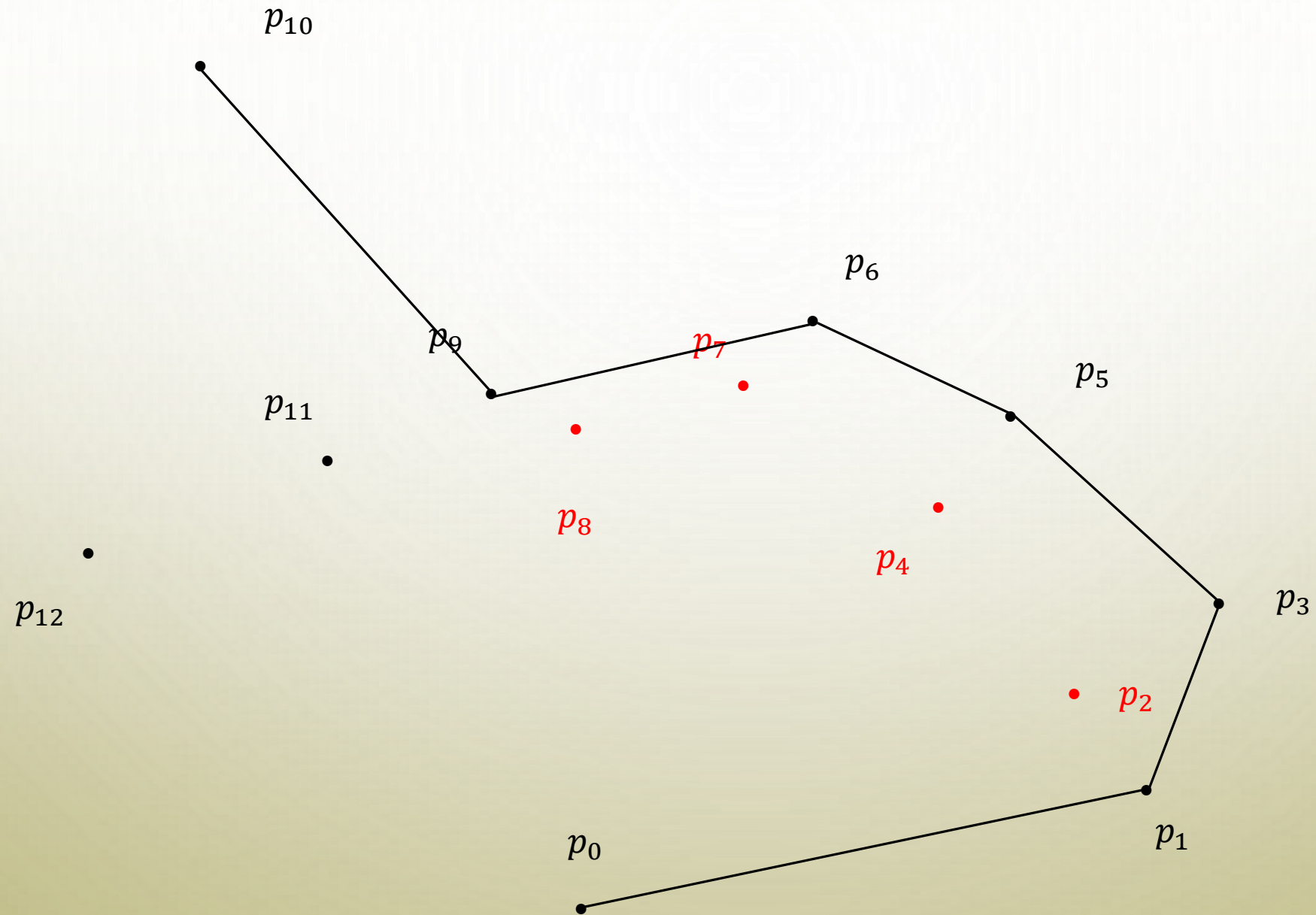




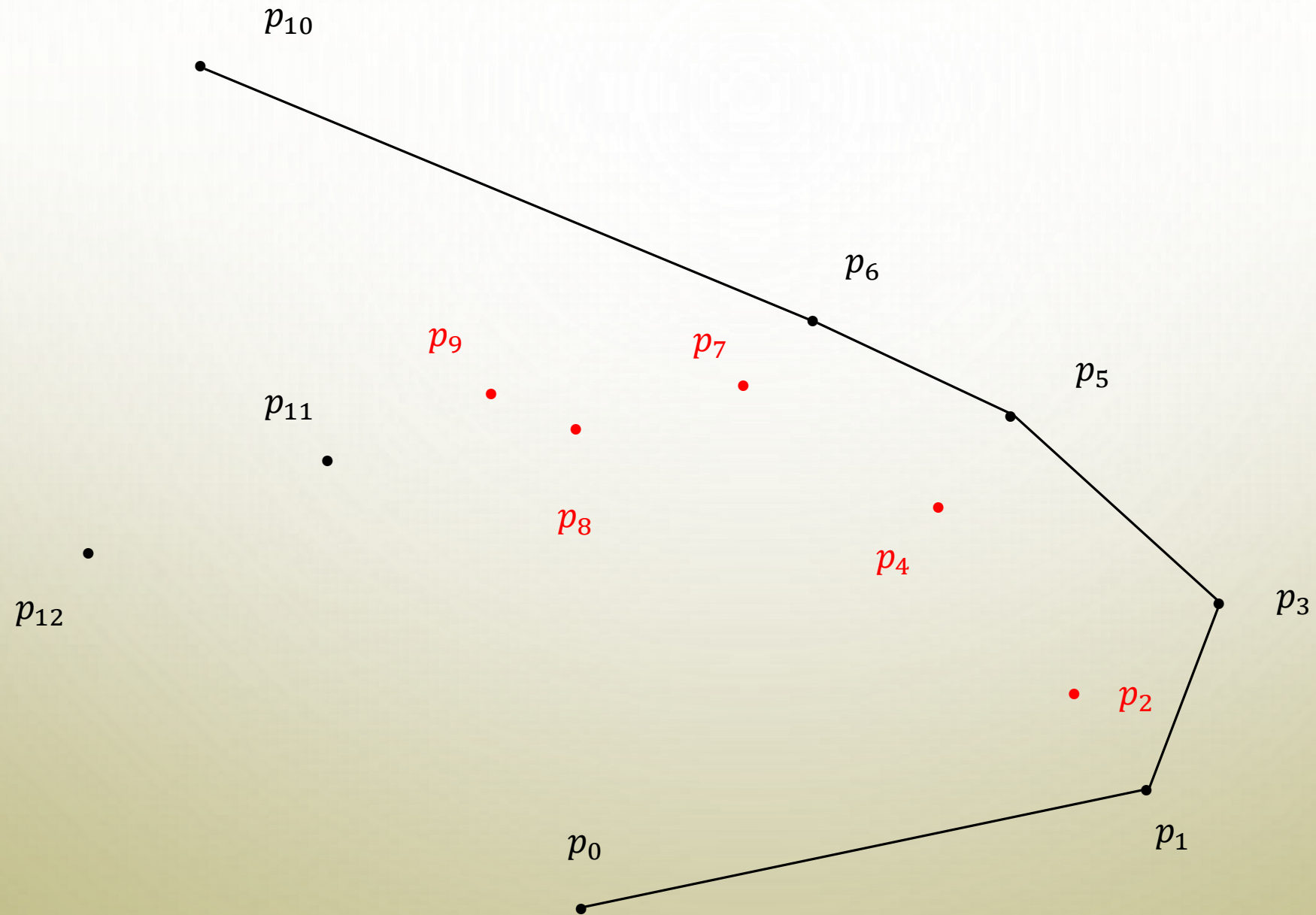
p_8
p_6
p_5
p_3
p_1
p_0



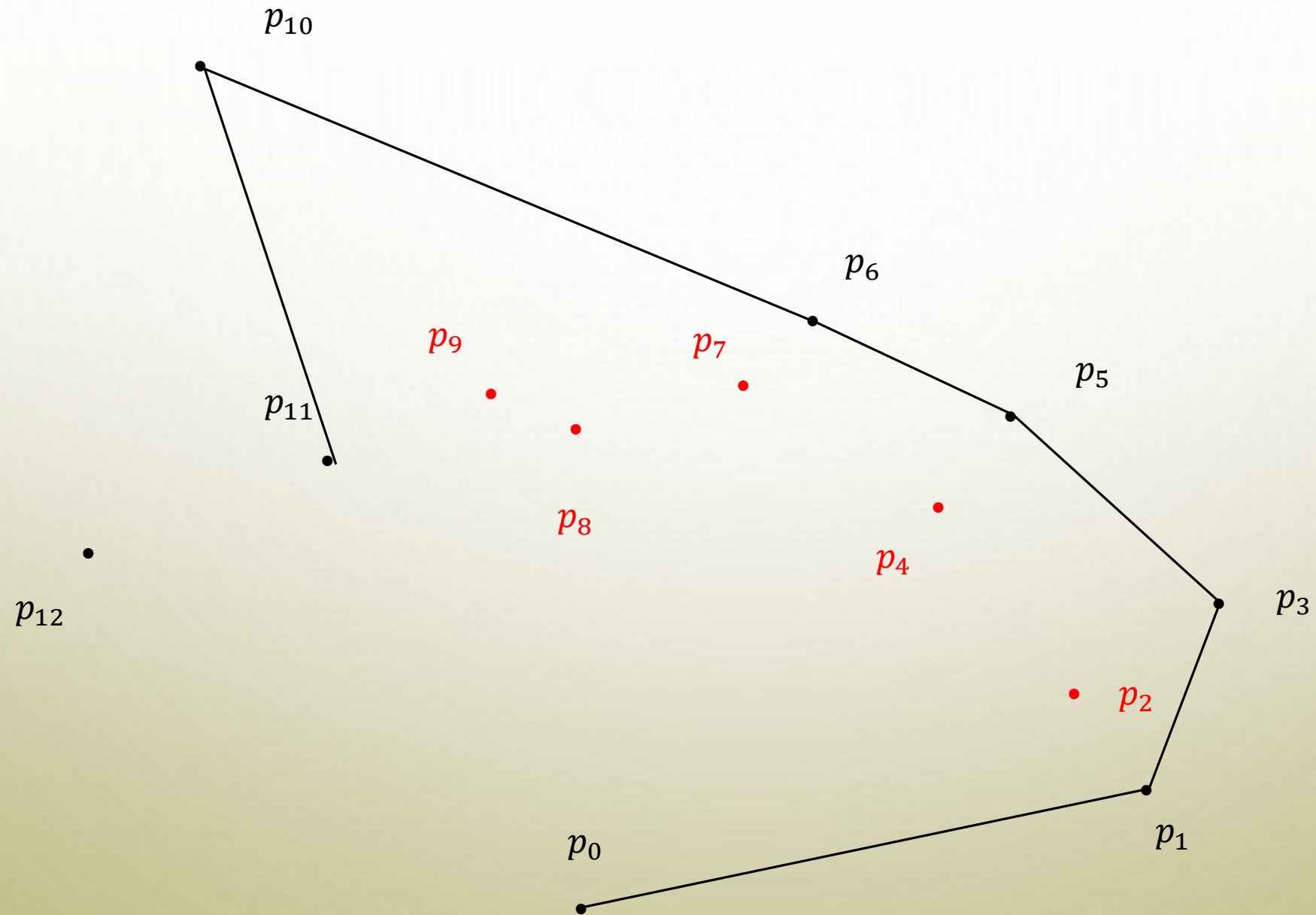
p_9
p_6
p_5
p_3
p_1
p_0



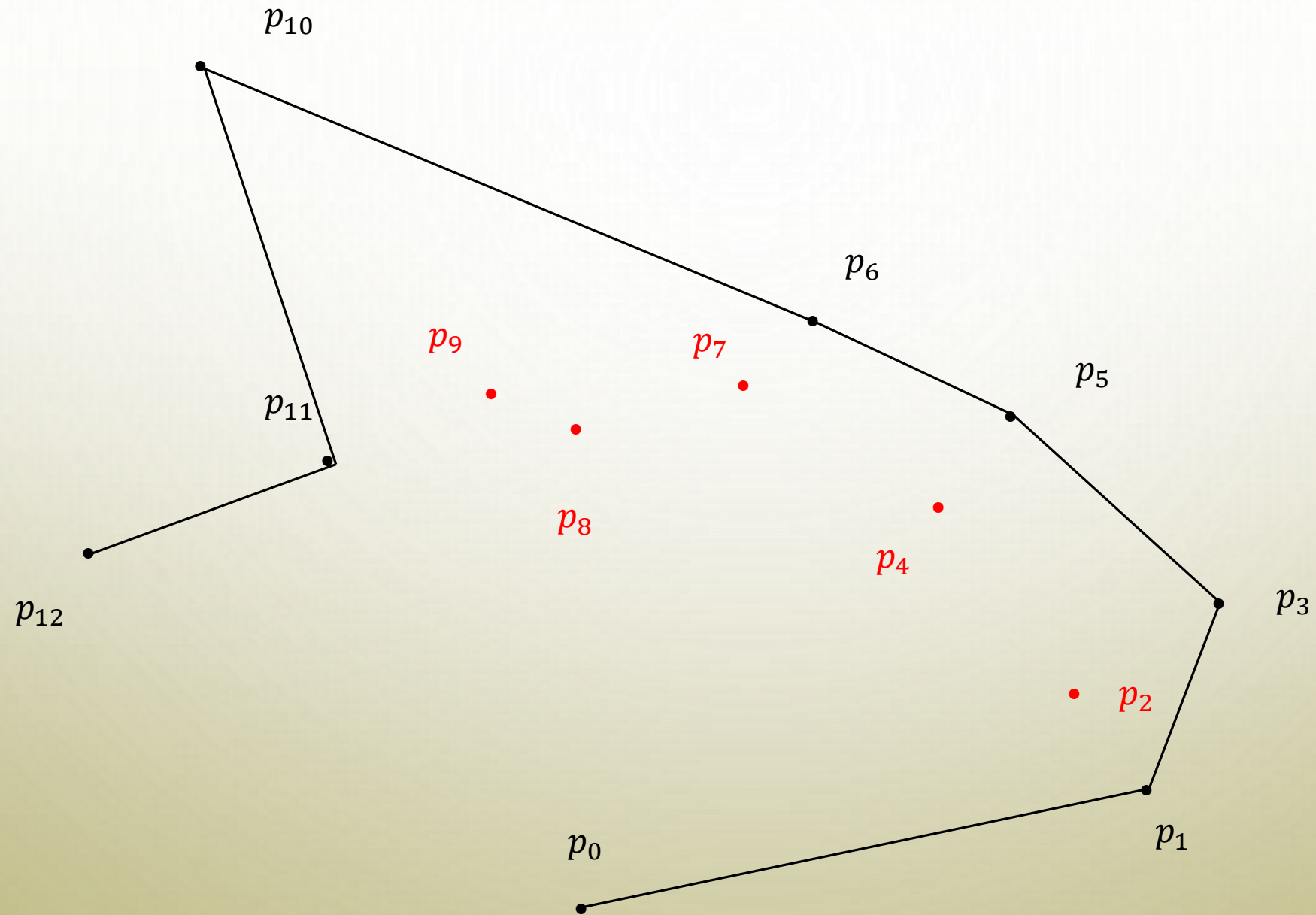
p_9
p_6
p_5
p_3
p_1
p_0



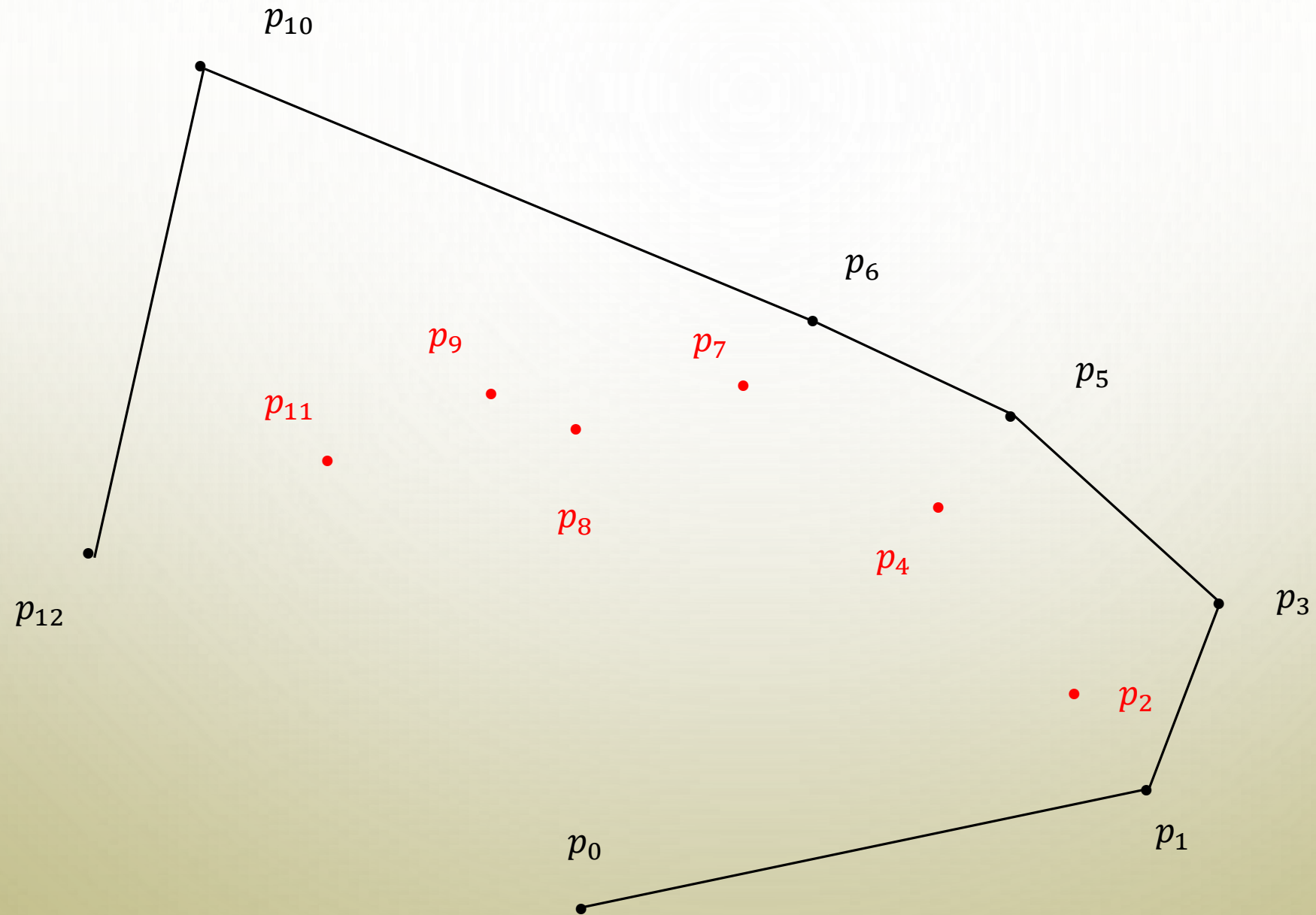
p_{10}
p_6
p_5
p_3
p_1
p_o



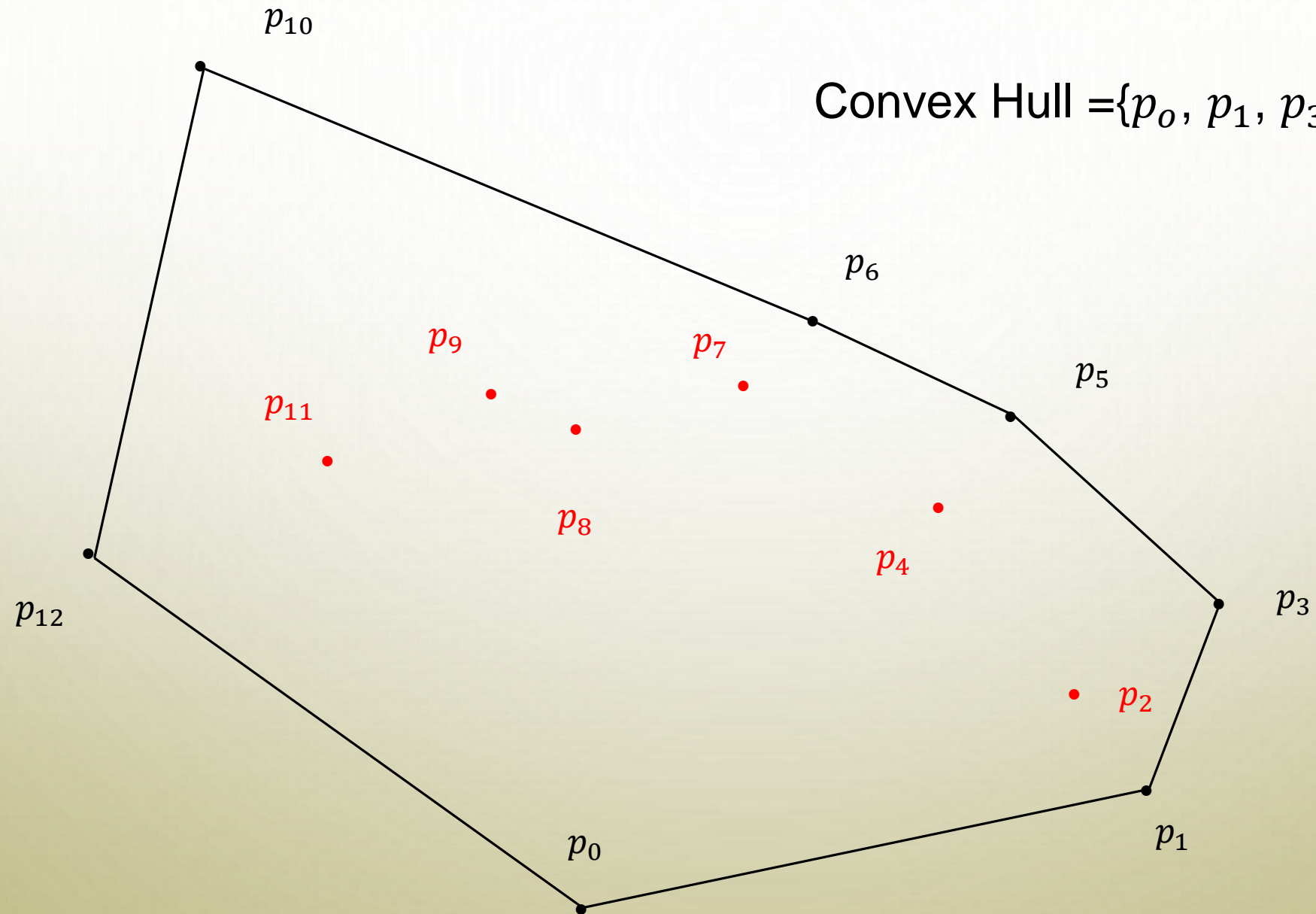
p_{11}
p_{10}
p_6
p_5
p_3
p_1
p_o



p_{11}
p_{10}
p_6
p_5
p_3
p_1
p_0



p_{12}
p_{10}
p_6
p_5
p_3
p_1
p_0



Convex Hull = $\{p_0, p_1, p_3, p_5, p_6, p_{10}, p_{12}\}$

p_{12}
p_{10}
p_6
p_5
p_3
p_1
p_0

ĐÁNH GIÁ ĐỘ PHỨC TẠP THUẬT TOÁN:

Thời gian:

- Độ phức tạp tìm điểm làm gốc O : $O(n)$
- Độ phức tạp sắp xếp các điểm theo thứ tự: $O(n \log n)$
- Độ phức tạp khi thêm các điểm vào bao lồi: $O(n)$
- Độ phức tạp khi xóa các điểm khỏi bao lồi: $O(n)$
- Độ phức tạp kiểm tra 1 điểm: $O(1)$



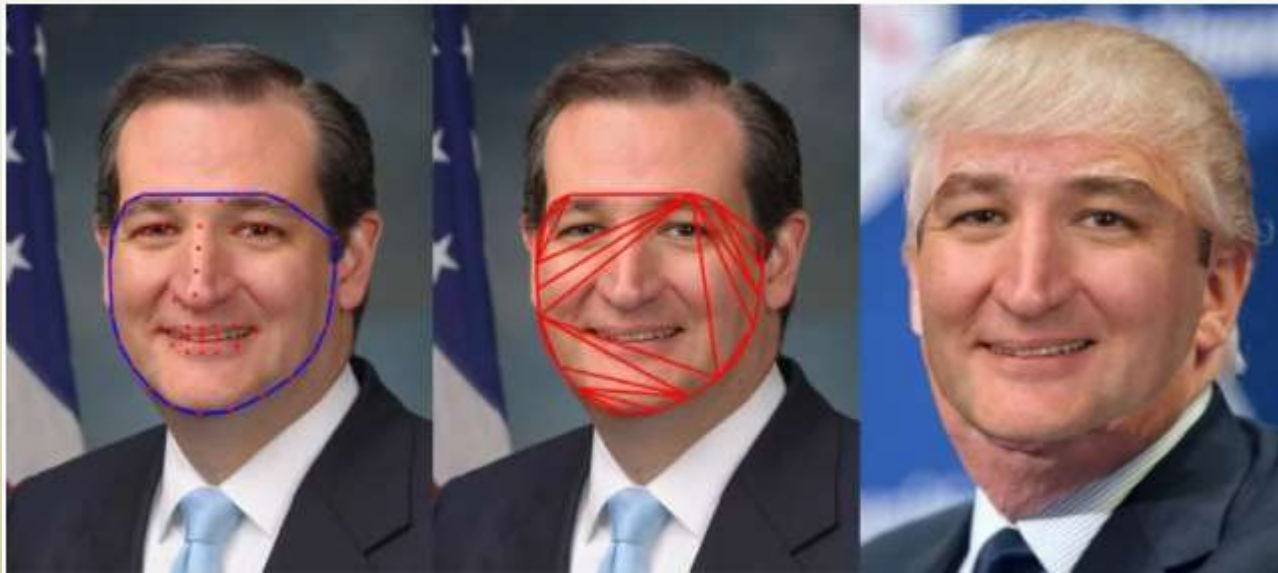
$$O(n) + O(n \log n) + O(n) + O(n) + O(1) = O(n \log n)$$

Không gian: $O(n)$

ỨNG DỤNG CỦA BAO LỖI:

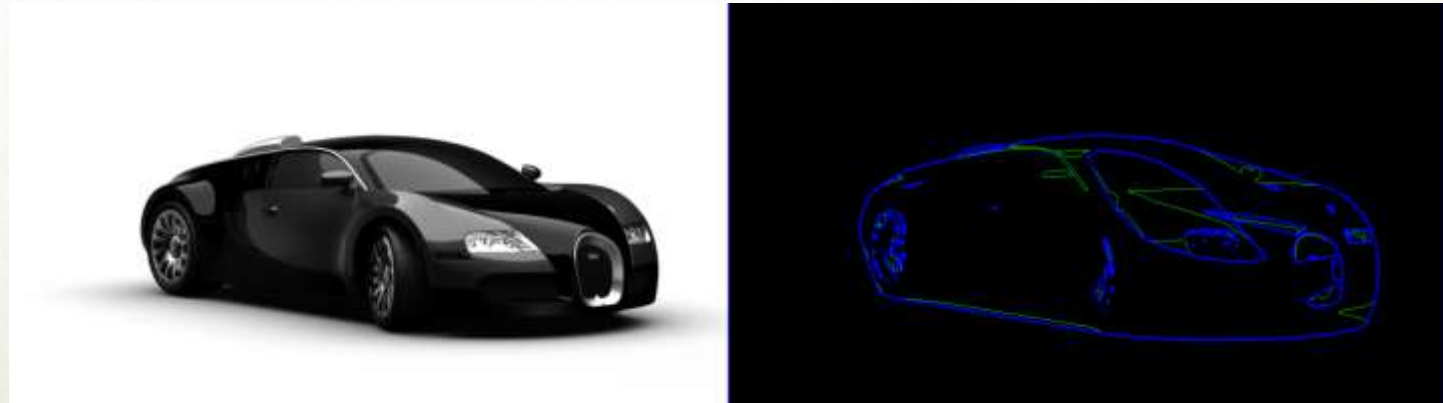
Computer Vision

Hoán đổi khuôn mặt:



ỨNG DỤNG CỦA BAO LỒI:

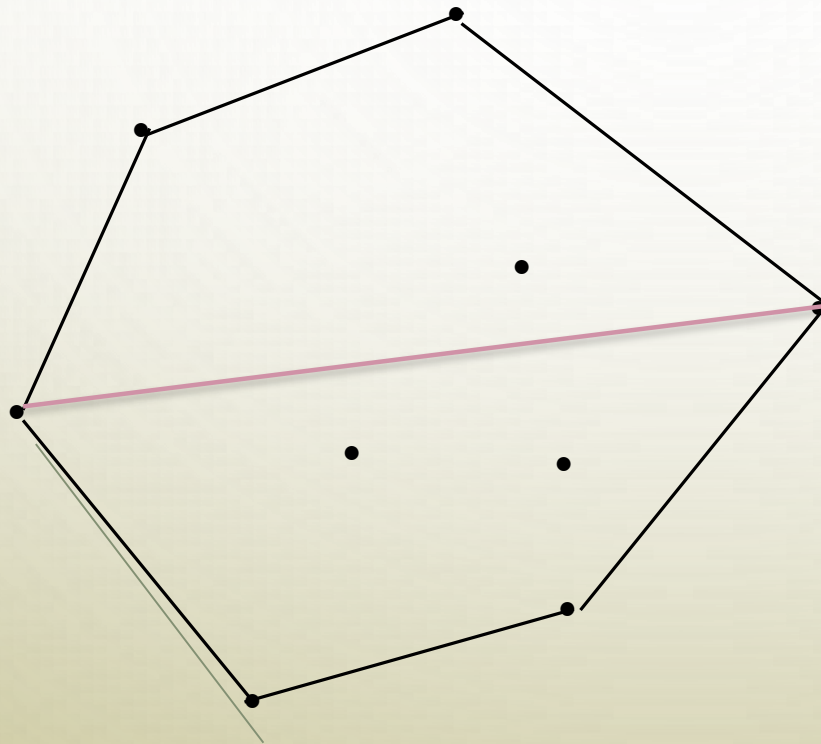
Xe ô tô tự lái:



Giúp tránh va chạm với các chướng ngại vật

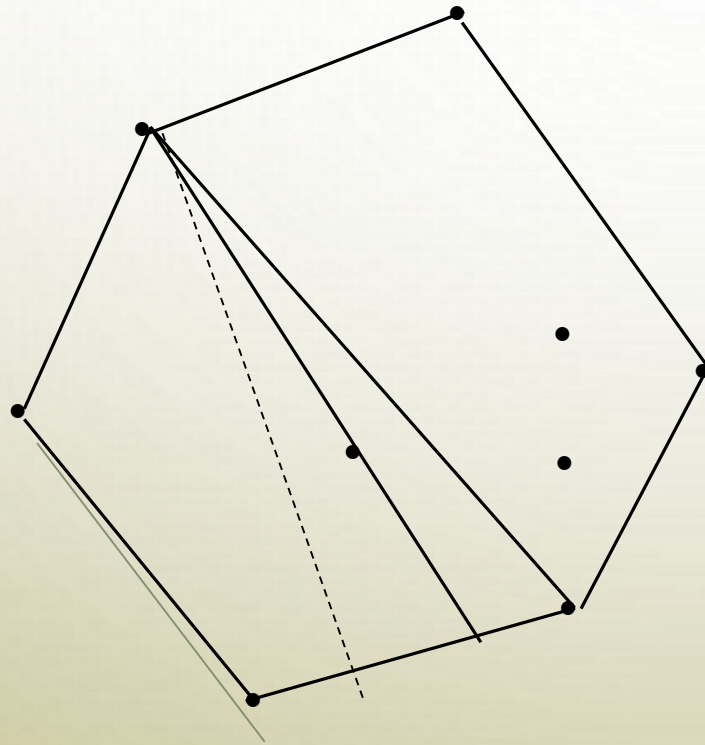
ỨNG DỤNG CỦA BAO LỒI:

Tìm cặp điểm xa nhất:



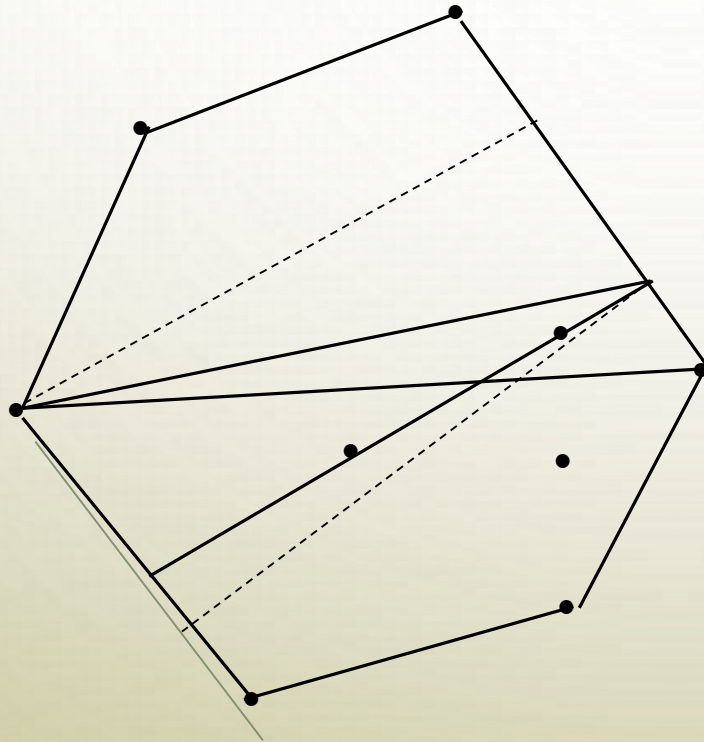
ỨNG DỤNG CỦA BAO LỒI:

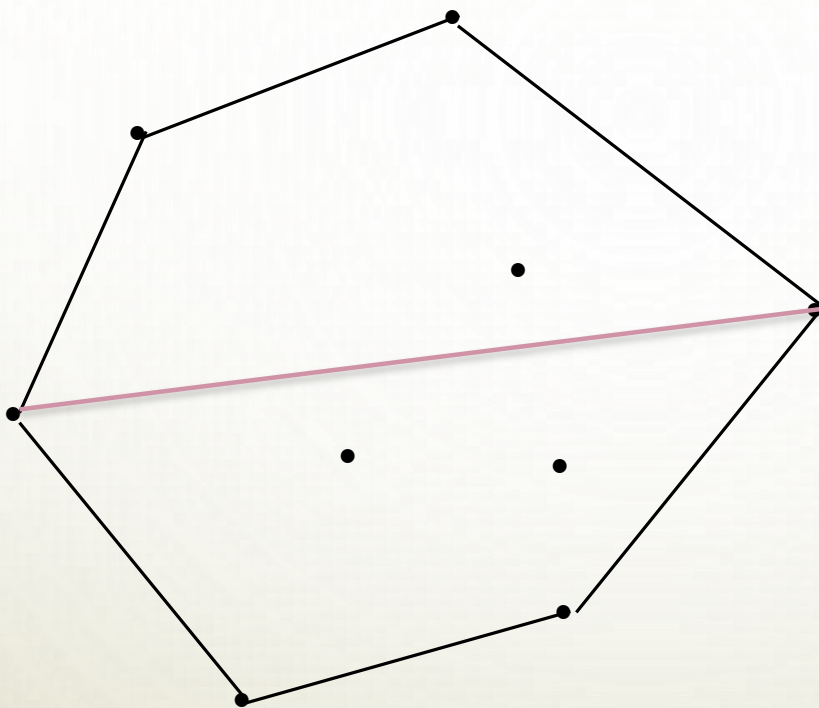
Tìm cặp điểm xa nhất:



ỨNG DỤNG CỦA BAO LỖI:

Tìm cặp điểm xa nhất:





Tìm bao lồi



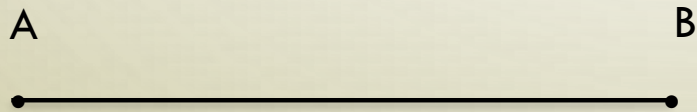
Rotating Calipers algorithm

BÀI TOÁN TRỘN MÀU SƠN

Đề bài: Bạn có 1 số hộp sơn. Mỗi hộp sơn có tỉ lệ thành phần 3 màu cơ bản **xanh:đỏ:vàng** là **$x:d:v$** với x, d, v dương. Giả sử màu của hộp sơn chỉ phụ thuộc vào tỉ lệ 3 màu cơ bản. Bằng cách pha trộn các loại sơn, liệu ta có thể thu được màu mục tiêu hay không (màu mục tiêu cũng được cho bằng tỉ lệ ba màu cơ bản), biết rằng lượng sơn trong hộp là vô tận.

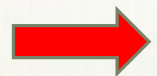
Hướng giải quyết:

- Chuẩn hóa dữ liệu bằng cách chia tỉ lệ 3 màu cho màu xanh $\Rightarrow 1:\frac{d}{x}:\frac{v}{x}$
- Khi đó tất cả giá trị x ở các thùng sơn và màu mục tiêu đều bằng 1 \Rightarrow bỏ qua màu xanh và chỉ tập trung đánh giá dựa trên 2 màu còn lại \Rightarrow Ta có thể biểu thị mỗi màu $(\frac{d}{x}:\frac{v}{x})$ thành 1 điểm trên mặt phẳng có tọa độ (a,b)
- Màu được pha trộn giữa 2 thùng sơn (biểu thị bằng điểm A và B) sẽ là điểm nằm trên đoạn thẳng AB





Nếu 1 màu được biểu thị bởi 1 điểm nằm trong bao lồi thì màu đó luôn có thể được trộn ra nhờ các điểm tạo nên bao lồi sau khi trải qua 1 số bước trộn

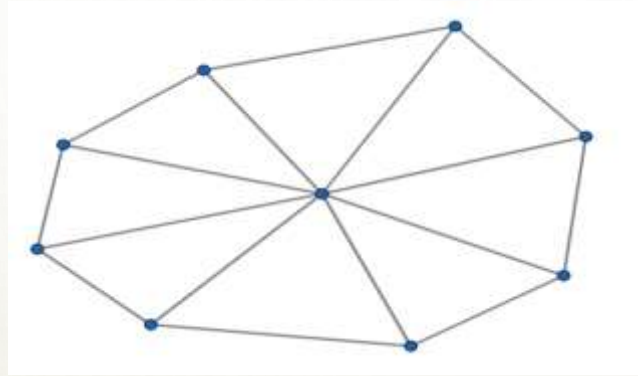


Các màu có thể trộn ra sẽ là những điểm nằm trong bao lồi được tạo ra bởi các điểm dữ liệu.



Kiểm tra điểm biểu thị cho màu mục tiêu có nằm trong bao lồi hay không?

Cách kiểm tra 1 điểm có nằm bên trong bao lồi hay không?



- Tính diện tích tạo bởi điểm đang xét và mỗi 2 đỉnh liên tiếp thuộc đa giác. Nếu tổng các diện tích bằng diện tích đa giác thì điểm sẽ nằm trong đa giác
- Công thức tính diện tích đa giác theo tọa độ:
$$S = \left| \frac{\sum (x_i - x_{i+1}) * (y_i + y_{i+1})}{2} \right|$$

ĐÁNH GIÁ ƯU, NHƯỢC ĐIỂM:

Ưu điểm:

- Cho kết quả tối ưu trong 1 mảng các bài toán mang đặc điểm hình học. So với các thuật toán khác nếu cùng áp dụng vào những bài toán thuộc mảng này, giải thuật hình học sẽ dễ thiết kế và áp dụng hơn, độ phức tạp thuật toán thường sẽ tốt hơn
- Những bài toán thuộc dạng này sẽ có những đặc điểm rất đặc trưng giúp việc tìm ra giải thuật phù hợp

NHƯỢC ĐIỂM:

- Chỉ có thể áp dụng phổ biến trong 1 mảng các bài toán. Không linh hoạt và dễ áp dụng như các thuật toán khác
- Trong 1 số trường hợp rất khó để nhận ra bản chất hình học của bài toán để áp dụng giải thuật phù hợp (áp dụng convex hull vào bài toán trộn màu sơn)
- Cài đặt thường khó và phức tạp

Tài liệu tham khảo

- <https://vnoi.info/wiki/translate/wcipeg/Convex-Hull>
- <https://www.geeksforgeeks.org/maximum-distance-between-two-points-in-coordinate-plane-using-rotating-calipers-method/>
- *<https://learnopencv.com/face-swap-using-opencv-c-python/>*

**CẢM ƠN THẦY VÀ CÁC BẠN ĐÃ LẮNG
NGHE**

