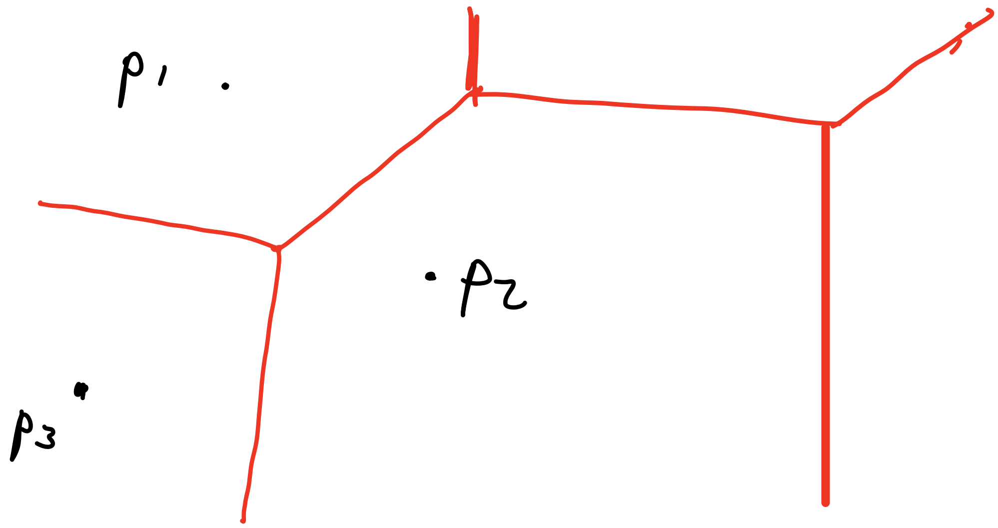
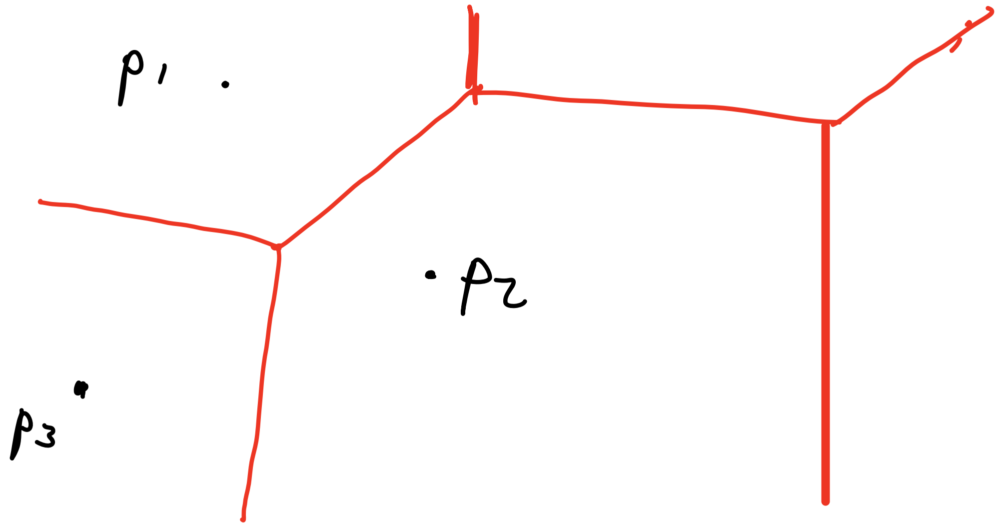
# Lecture 11 - Voronoi diagrams



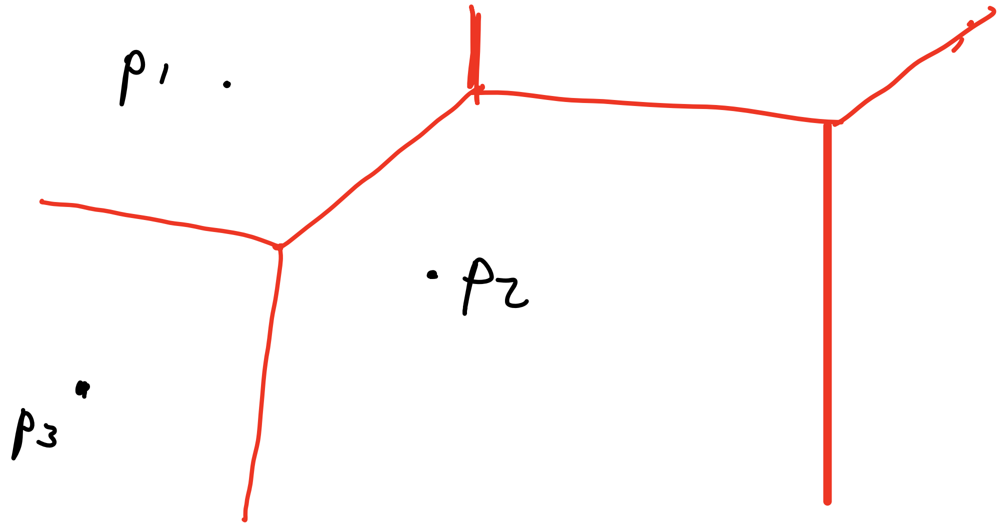
# Lecture 11 - Voronoi diagrams




## Post-office problem

- City with post-offices $P = \{p_1, \ldots, p_n\}$.

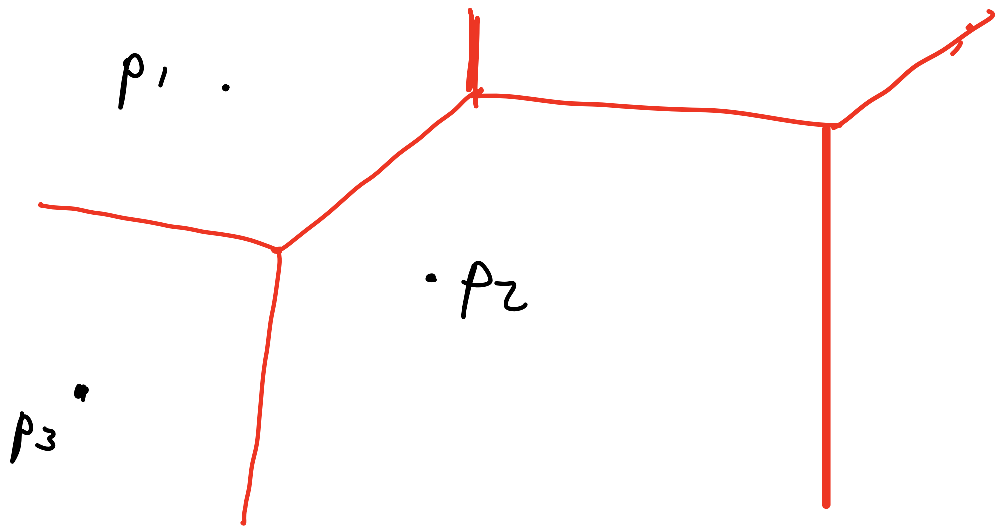
# Lecture 11 - Voronoi diagrams




## Post-office problem

- City with post-offices $P = \{p_1, \ldots, p_n\}$.
- Divide city into regions $V(p_i)$ around each post-office $p_i$ such that any point in $V(p_i)$ is at least as close to $p_i$ as it is to any other $p_j$.

# Lecture 11 - Voronoi diagrams




## Post-office problem

- City with post-offices $P = \{p_1, \ldots, p_n\}$.
- Divide city into regions $V(p_i)$ around each post-office $p_i$ such that any point in $V(p_i)$ is at least as close to $p_i$ as it is to any other $p_j$.

## Note

Examples in nature:

- giraffe's skin (areas around cells secreting melatonin)
- show picture

Given $P = \{p_1, \ldots, p_n\}$, the Voronoi diagram $V(P)$ is subdivision of plane $\mathbb{R}^2$ into $n$ regions

Given $P = \{p_1, ..., p_n\}$, the Voronoi diagram $V(P)$ is subdivision of plane $\mathbb{R}^2$ into $n$ regions:

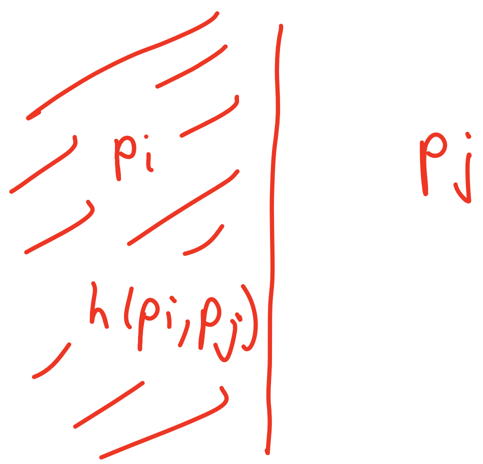
$$\underbrace{V(p_i)}_{\text{Voronoi cell}} = \{q \in \mathbb{R}^2 : d(q, p_i) \leq d(q, p_j) : j \neq i\}$$

Given $P = \{p_1, \ldots, p_n\}$, the **Voronoi diagram $V(P)$** is subdivision of plane $\mathbb{R}^2$ into $n$ regions:

$$\underbrace{V(p_i)}_{\text{Voronoi cell}} = \{q \in \mathbb{R}^2 : d(q, p_i) \leq d(q, p_j) : j \neq i\}$$

half-plane

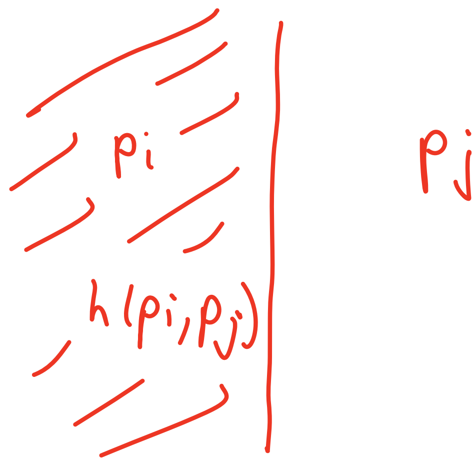
- Let $h(p_i, p_j) = \{q : d(q, p_i) \leq d(q, p_j)\}$



Given $P = \{p_1, \dots, p_n\}$, the Voronoi diagram $V(P)$ is subdivision of plane $\mathbb{R}^2$ into $n$ regions:

$$\underbrace{V(p_i)}_{\text{Voronoi cell}} = \{q \in \mathbb{R}^2 : d(q, p_i) \leq d(q, p_j) : j \neq i\}$$

- Let $h(p_i, p_j) = \{q : d(q, p_i) \leq d(q, p_j)\}$ half-plane




- Then $V(p_i) = \bigcap_{j \neq i} h(p_i, p_j)$

& as an intersection of half-planes is a convex polygon.

# Slow algorithm

Formula $$V(p_i) = \bigcap_{j \neq i} h(p_i, p_j)$$

- By Lecture 5, intersection of $n$ half-planes takes time $O(n \log n)$.

## Slow algorithm

Formula $$V(p_i) = \bigcap_{j \neq i} h(p_i, p_j)$$

- By Lecture 5, intersection of $n$ half-planes takes time $O(n \log n)$.
- Using this, to calc $n$ Voronoi cells takes time $O(n^2 \log n)$

## Slow algorithm

Formula $$V(p_i) = \bigcap_{j \neq i} h(p_i, p_j)$$

- By Lecture 5, intersection of $n$ half-planes takes time $O(n \log n)$.
- Using this, to calc $n$ Voronoi cells takes time $O(n^2 \log n)$

Today: Faster algorithm - $O(n \log n)$ (Fortune's algorithm)

# Slow algorithm

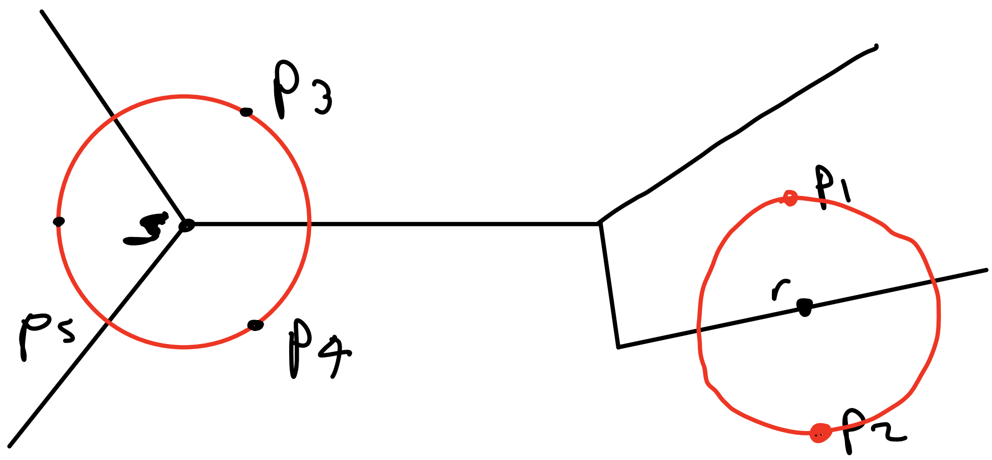
Formula $V(p_i) = \bigcap_{j \neq i} h(p_i, p_j)$

- By Lecture 5, intersection of $n$ half-planes takes time $O(n \log n)$.
- Using this, to calc $n$ Voronoi cells takes time $O(n^2 \log n)$

Today: Faster algorithm - $O(n \log n)$ ( Fortune's algorithm )

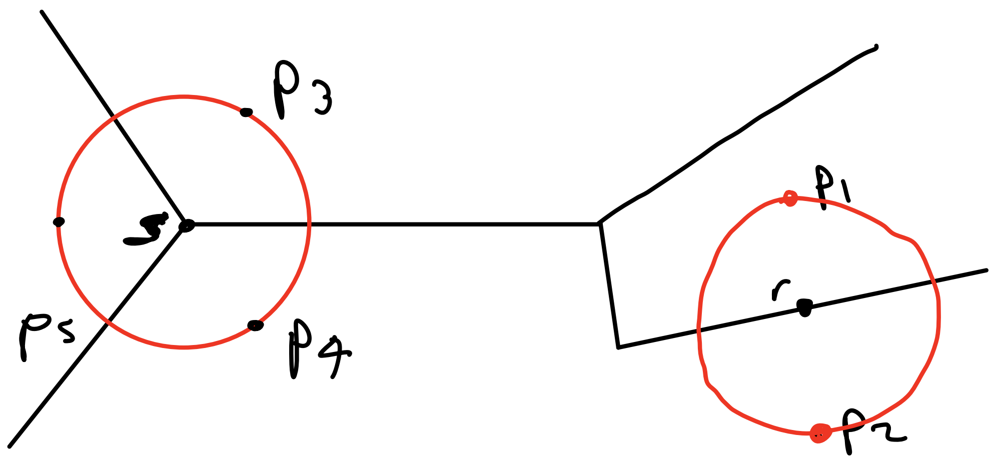
- A Sweepline algorithm

# Edges & vertices of Voronoi diagram




· It is not hard to see that :

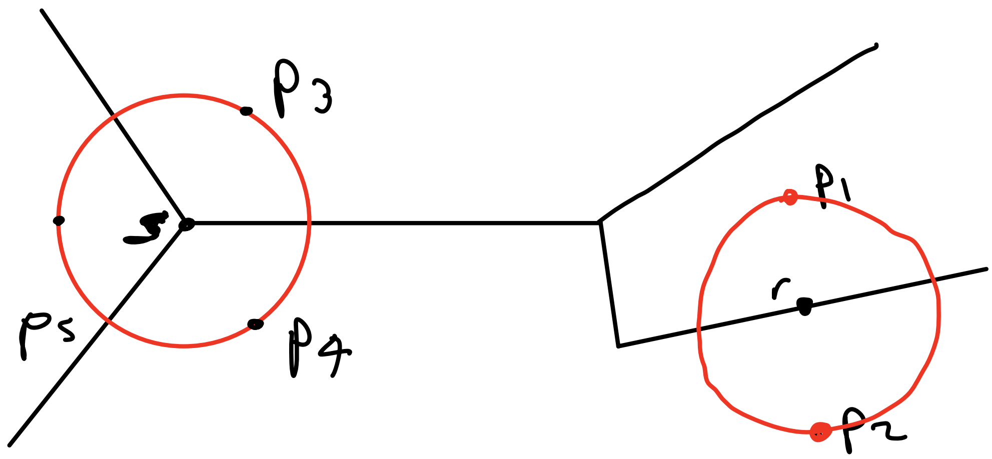
# Edges & vertices of Voronoi diagram




· It is not hard to see that :

(1) $r \in \mathbb{R}^2$ lie on an edge of V-diagram
$\Longleftrightarrow$ $r$ is equidistant from its nearest two points of $P$

# Edges & vertices of Voronoi diagram




· It is not hard to see that:

① $r \in \mathbb{R}^2$ lie on an edge of V-diagram $\iff$ $r$ is equidistant from its nearest two points of $P$

② $r$ is a vertex of V-diagram $\iff$ $r$ is equidistant from its nearest three points.

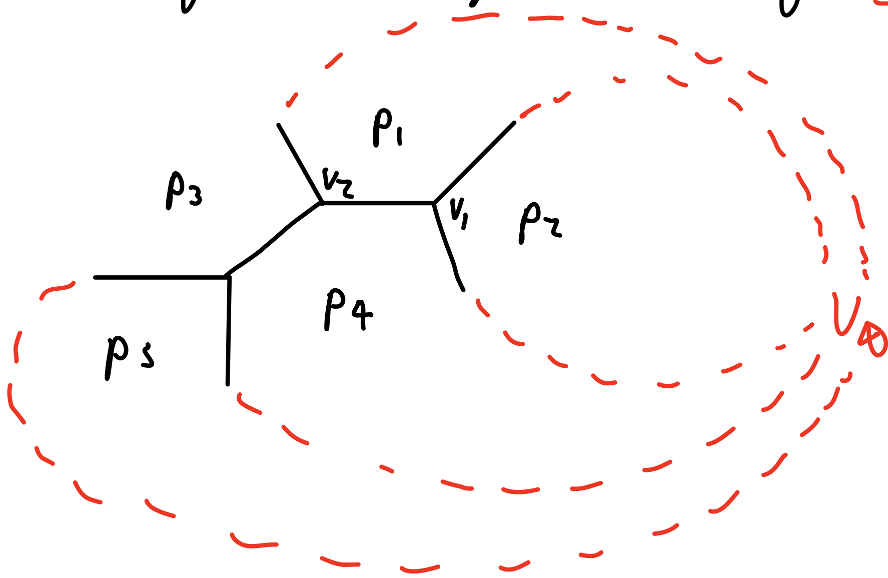
See Fig 9.3 in E-Learning.

**Theorem)** Any U-diagram for a set of $n \geq 3$ points (not on a line) has at most $2n-5$ vertices & at most $3n-6$ edges.

Proof)

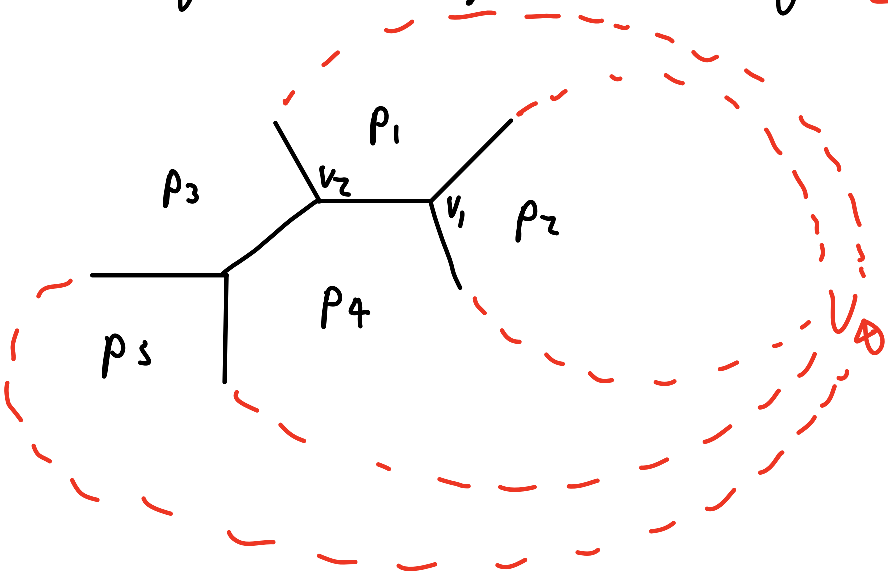
Theorem) Any V-diagram for a set of $n \geq 3$ points (not on a line) has at most $2n-5$ vertices & at most $3n-6$ edges.

Proof) $m$ = no of vertices, $h$ = no of edges



**Theorem)** Any V-diagram for a set of $n \geq 3$ points (not on a line) has at most $2n-5$ vertices & at most $3n-6$ edges.

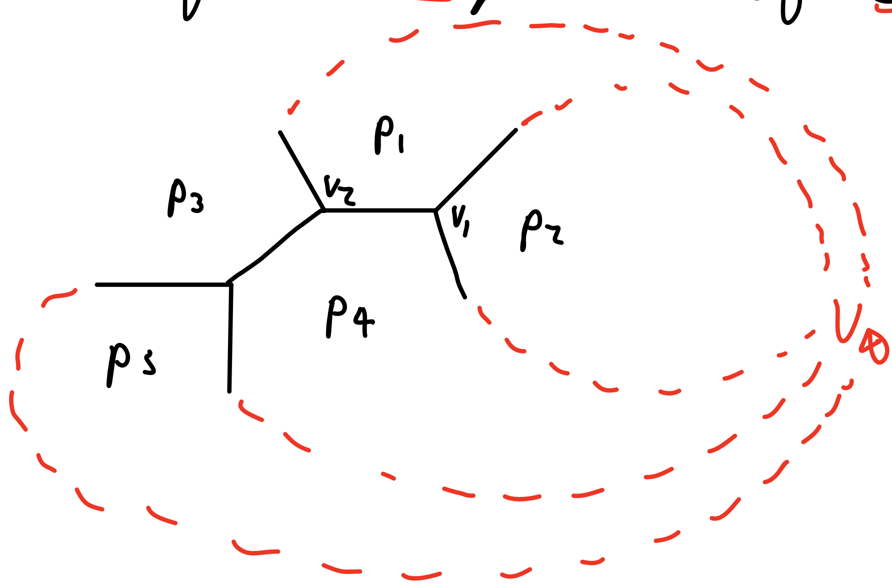
**Proof)** $m$ = no of vertices, $h$ = no of edges




- Add vertex $v_\infty$ "at infinity" as endpoint for all half-lines.

**Theorem** Any V-diagram for a set of $n \geq 3$ points (not on a line) has at most $2n-5$ vertices & at most $3n-6$ edges.

**Proof)** $m$ = no of vertices, $h$ = no of edges

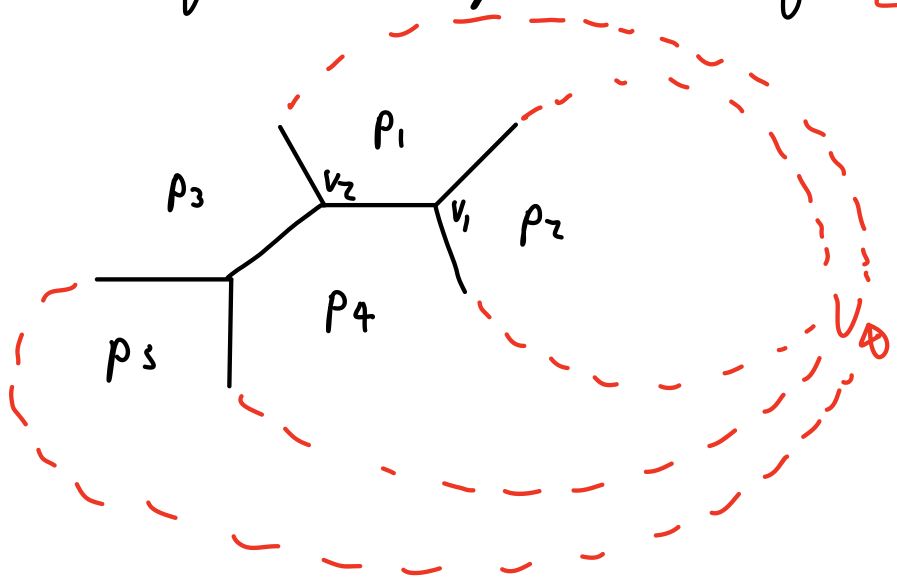



- Add vertex $v_\infty$ "at infinity" as endpoint for all half-lines.

- Obtain connected planar graph with
$n$ faces, $m+1$ vertices,
$h$ edges

**Theorem]** Any V-diagram for a set of $n \geq 3$ points (not on a line) has at most $2n-5$ vertices & at most $3n-6$ edges.

**Proof)** $m$ = no of vertices, $h$ = no of edges




- Add vertex $v_\infty$ "at infinity" as endpoint for all half-lines.

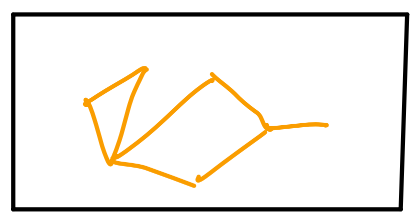
- Obtain connected planar graph with $n$ faces, $m+1$ vertices, $h$ edges

Aside: Euler Formula

$V - E + F = 2$

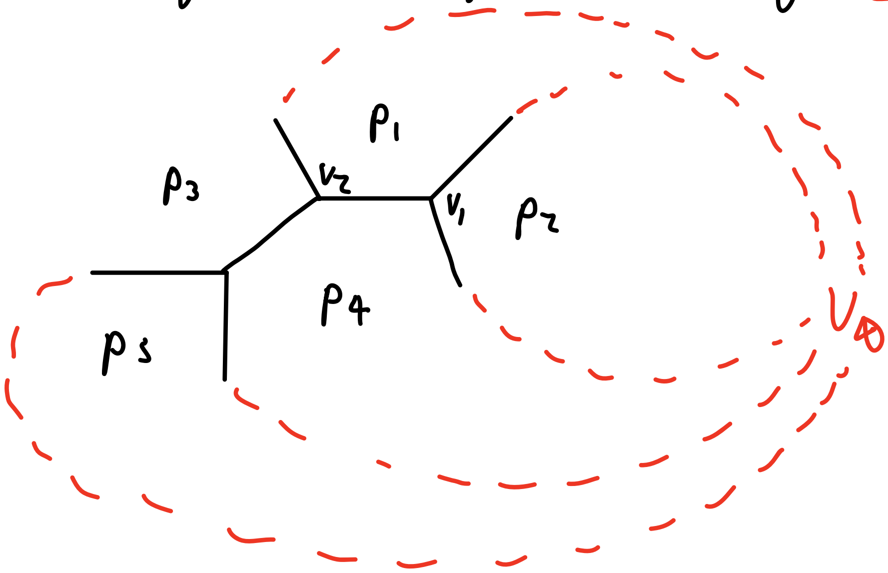
vertices edges faces

$7 - 8 + 3 = 2$



**Theorem** Any V-diagram for a set of $n \geq 3$ points (not on a line) has at most $2n-5$ vertices & at most $3n-6$ edges.

Proof) $m$ = no of vertices, $h$ = no of edges




- Add vertex $v_\infty$ "at infinity" as endpoint for all half-lines.

- Obtain connected planar graph with $n$ faces, $m+1$ vertices, $h$ edges

Euler's formula

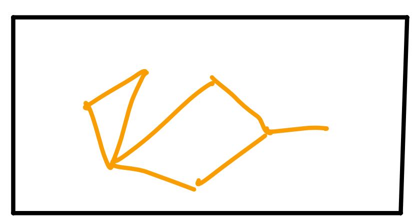
$(m+1) - h + n = 2$.

Aside: Euler formula

$V - E + F = 2$

vertices edges faces

$7 - 8 + 3 = 2$



So $\qquad (m+1) - h + n = 2$

So $$(m+1) - h + n = 2$$

- Degree of vertex of $V(P) \geq 3$ by 2 above (each vertex equidistant to nearest 3 pts)

So $\underline{(m+1) - h + n = 2}$

- Degree of vertex of $V(P) \geq 3$ by ② above (each vertex equidistant to nearest 3 pts)

- Sum of degrees $= 2$ (no of edges)

  so $2h \geq 3(m+1)$

So $\underline{(m+1) - h + n = 2}$

- Degree of vertex of $V(P) \geq 3$ by ② above (each vertex equidistant to nearest 3 pts)

- Sum of degrees $= 2$(no of edges)
  so $2h \geq 3(m+1)$

- Subsituting $h = m+n-1$ into $*$ gives
  $$2m + 2n - 2 \geq 3(m+1)$$

So $\qquad (m+1) - h + n = 2$

- Degree of vertex of $V(P) \geq 3$ by ② above (each vertex equidistant to nearest 3 pts)

- Sum of degrees = 2(no of edges)
  so $2h \geq 3(m+1)$

- Subsituting $h = m+n-1$ into $*$ gives

$$2m + 2n - 2 \geq 3(m+1)$$

$$\Rightarrow m \leq 2n - 5.$$

So $\quad (m+1)-h+n=2$

- Degree of vertex of $V(P) \geq 3$ by ② above (each vertex equidistant to nearest 3 pts)

- Sum of degrees $= 2$ (no of edges)

  so ✱ $\quad 2h \geq 3(m+1)$

- Subsituting $h = m+n-1$ into ✱ gives

$$2m+2n-2 \geq 3(m+1)$$

$$\Rightarrow m \leq 2n-5.$$

- Sim., subbing $m = h-n+1$ into ✱ gives $h \leq 3n-6$.

$\square$

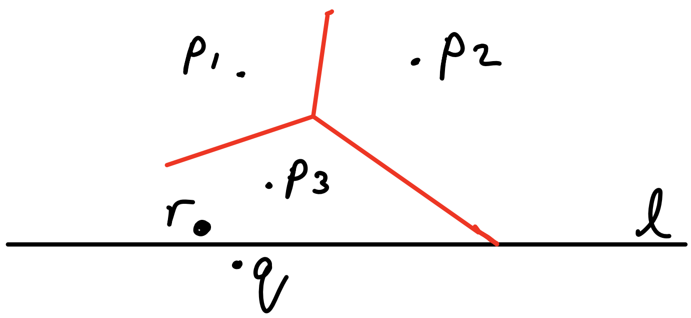
# Sweep-line algorithm - Fortune's algorithm

(See animation "Voronoi tesselation" on YouTube by K. Schaal )

# Sweep-line algorithm - Fortune's algorithm

(See animation "Voronoi tesselation" on YouTube by K. Schaal)

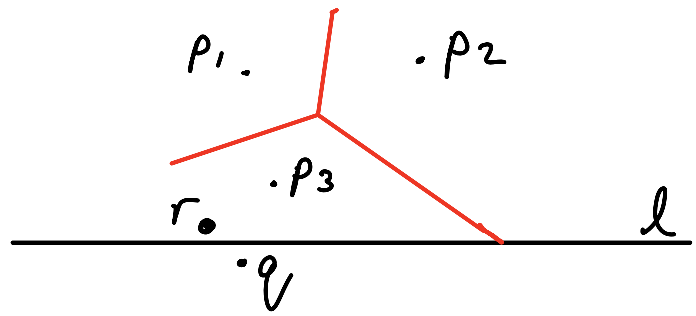
- Would like to use sweep-line approach to compute V-diag above sweep-line $\ell$.



# Sweep-line algorithm - Fortune's algorithm

(See animation "Voronoi tesselation" on YouTube by K. Schaal)

- Would like to use sweep-line approach to compute V-diag above sweep-line $\ell$.

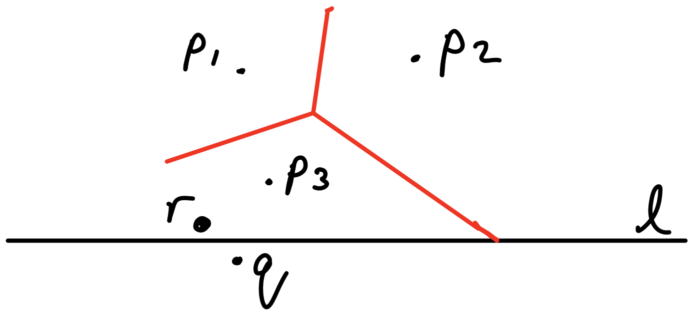



Problem: new point $q$ below sweep-line can change the Voronoi cells above.

# Sweep-line algorithm - Fortune's algorithm

(See animation "Voronoi tesselation" on YouTube by K. Schaal)

- Would like to use sweep-line approach to compute V-diag above sweep-line $\ell$.



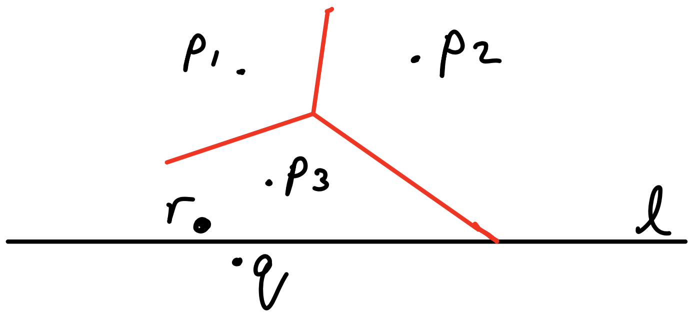

Problem: new point $q$ below sweep-line can change the Voronoi cells above.

eg. above: $r \in V(p_3)$ before $\ell$ passes $q$, but $r \in V(q)$ afterwards.

# Sweep-line algorithm - Fortune's algorithm

(See animation "Voronoi Tesselation" on YouTube by K. Schaal)

- Would like to use sweep-line approach to compute V-diag above sweep-line $\ell$.



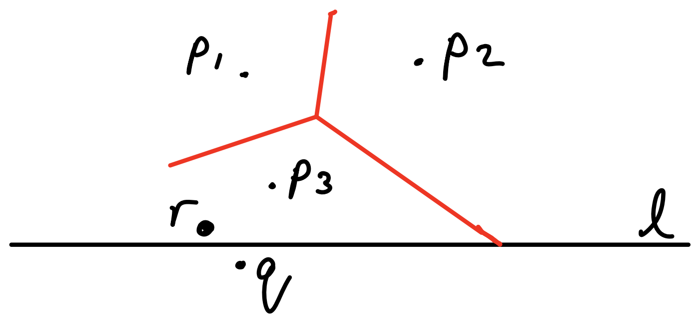

Problem: new point $q$ below sweep-line can change the Voronoi cells above -

eg. above: $r \in V(p_3)$ before $\ell$ passes $q$, but $r \in V(q)$ afterwards!

- This problem can only occur at point $r$ for which $d(r,\ell) < d(r,p)$ for each $p \in P$ above sweepline.

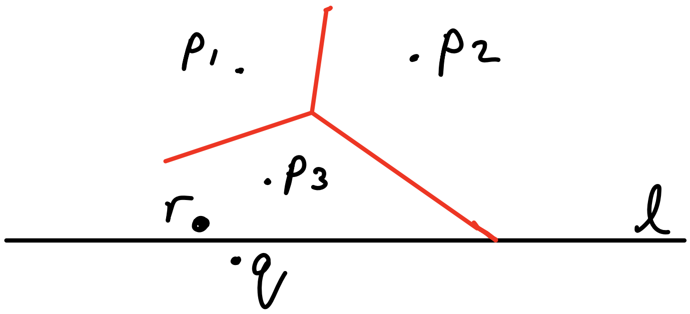
# Sweep-line algorithm - Fortune's algorithm

- Would like to use sweep-line approach to compute V-diag above sweep-line $\ell$.



# Sweep-line algorithm - Fortune's algorithm

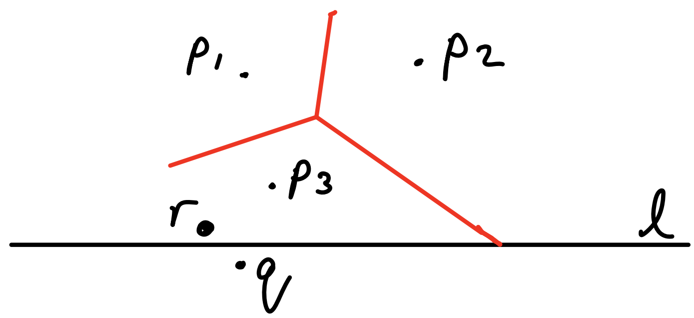
- Would like to use sweep-line approach to compute V-diag above sweep-line $\ell$.




Problem: new point $q$ below sweep-line can change the Voronoi cells above.

# Sweep-line algorithm - Fortune's algorithm

- Would like to use sweep-line approach to compute V-diag above sweep-line $\ell$.

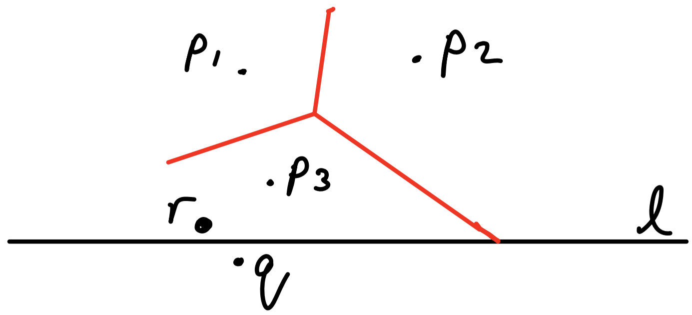



Problem: new point $q$ below sweep-line can change the Voronoi cells above -

- This problem can only occur at point $r$ for which $d(r,\ell) < d(r,p)$ for each $p \in P$ above sweepline.

# Sweep-line algorithm - Fortune's algorithm

- Would like to use sweep-line approach to compute V-diag above sweep-line $\ell$.




Problem: new point $q$ below sweep-line can change the Voronoi cells above -

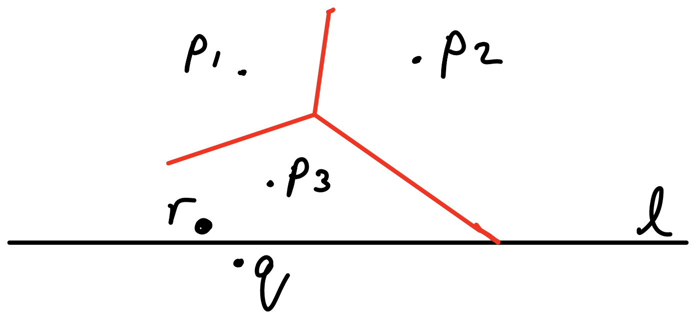
- This problem can only occur at point $r$ for which $d(r,\ell) < d(r,p)$ for each $p \in P$ above sweepline.
- For $p$ above $\ell$,

  let $\alpha^+(p,\ell) = \{ x : d(x,p) \leq d(x,\ell) \}$

  $\alpha(p,\ell) = \{ x : d(x,p) = d(x,\ell) \}$

# Sweep-line algorithm - Fortune's algorithm

- Would like to use sweep-line approach to compute V-diag above sweep-line $\ell$.




Problem: new point $q$ below sweep-line can change the Voronoi cells above.

- This problem can only occur at point $r$ for which $d(r,\ell) < d(r,p)$ for each $p \in P$ above sweepline.
- For $p$ above $\ell$,

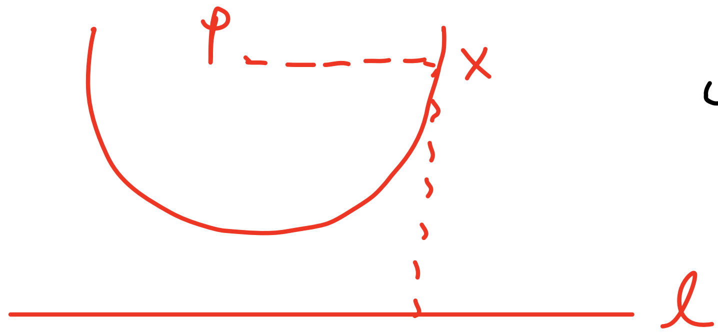
  let $\alpha^+(p,\ell) = \{ x : d(x,p) \leq d(x,\ell) \}$

  $\alpha(p,\ell) = \{ x : d(x,p) = d(x,\ell) \}$

- By the above reasoning, we can correctly compute V-diagram in the region $\bigcup_{p \text{ above } \ell} \alpha^+(p,\ell)$

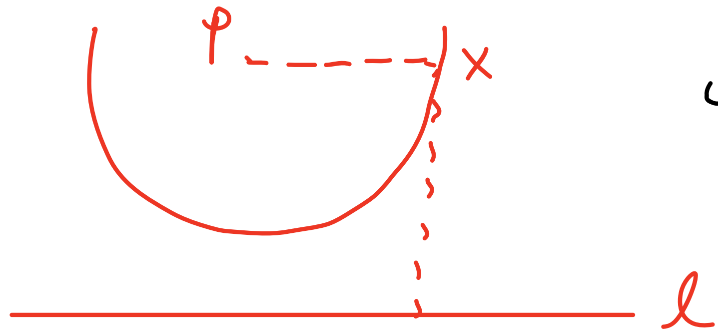
  as any point in this region is closer to some $p$ above $\ell$ than to any point below it.

Each $\alpha(p,\ell) = \{x : d(x,p) = d(x,\ell)\}$ is a parabola



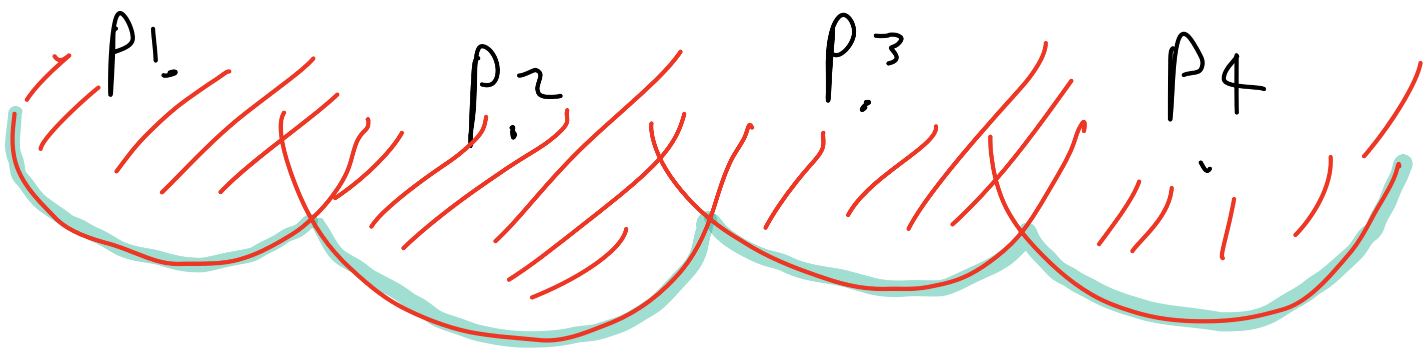

w/ focus $p$ & directrix $\ell$

Each $\alpha(p,\ell) = \{x : d(x,p) = d(x,\ell)\}$
is a parabola




w/ focus $p$
& directrix
$\ell$

The boundary of $\bigcup_{p \text{ above } \ell} \alpha^+(p,\ell)$ consists
of arcs of paraboloi

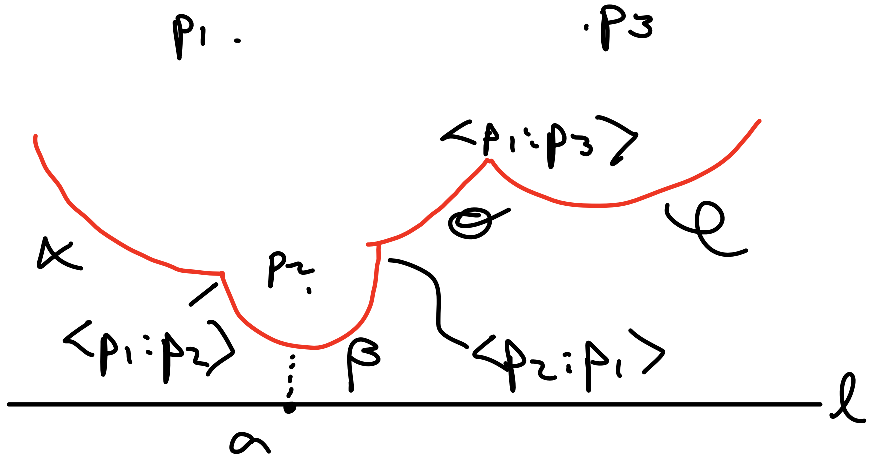



boundary of this region
is called the beach-line

- Beach-line @ $\ell$ is stored using a balanced binary tree.

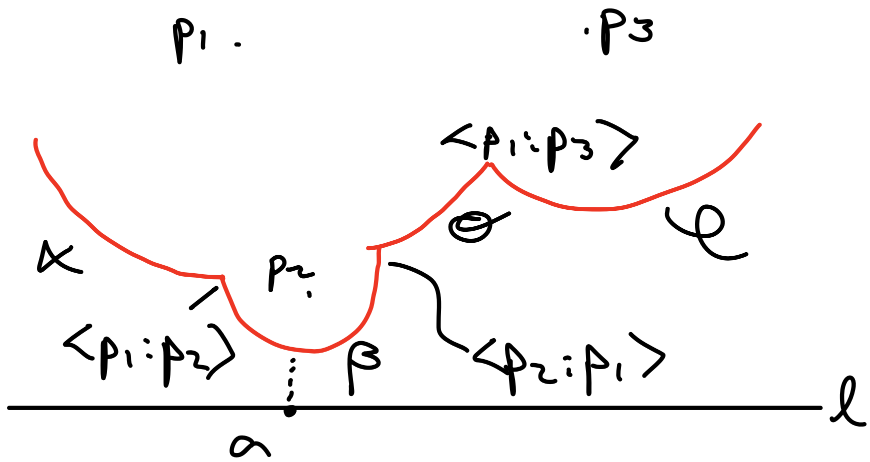
- Beach-line @ $\ell$ is stored using a <u>balanced binary tree</u>.

<u>Picture of beach-line @ $\ell$</u>:

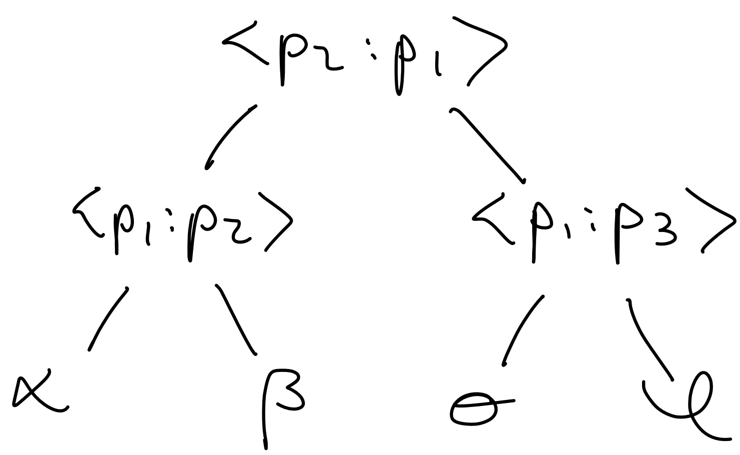


• Beach-line @ $\ell$ is stored using a balanced binary tree.

Picture of beach-line @ $\ell$ :



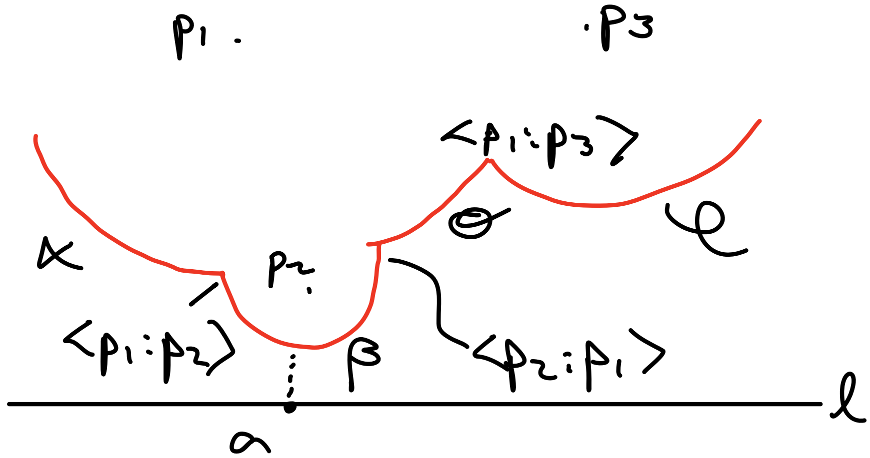

Associated tree :



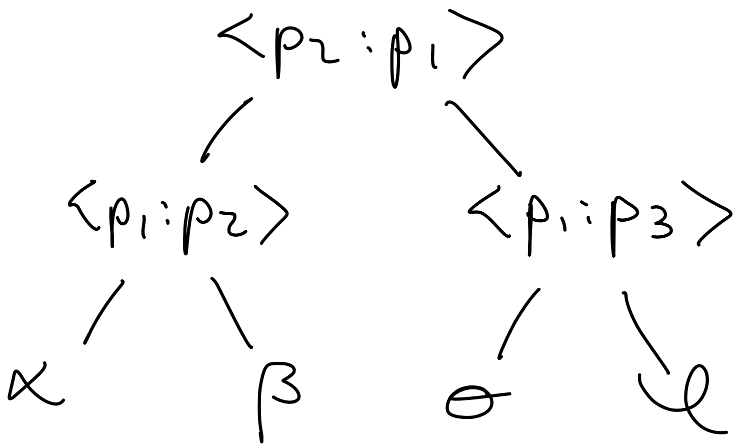

• leaves = arcs of beach-line

- Beach-line @ $\ell$ is stored using a balanced binary tree.

Picture of beach-line @ $\ell$:



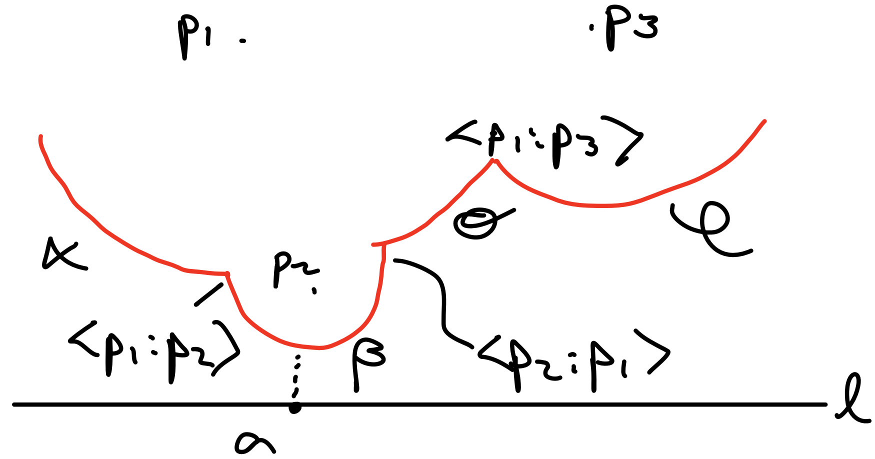

Associated tree:



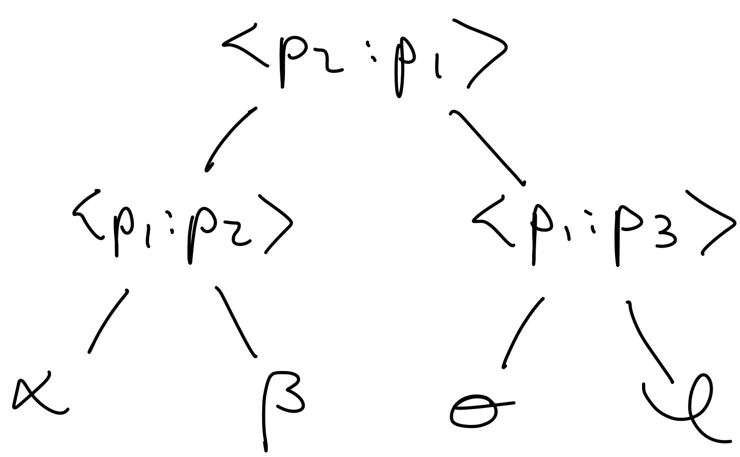

- leaves = arcs of beach-line
- Internal nodes of tree $\langle p_i : p_j \rangle$ represent "breakpoints" on beach-line, at which paraboloi around $p_i$ & $p_j$ meet with arc of $p_i$ to left & arc of $p_j$ to the right.

• Beach-line @ $\ell$ is stored using a balanced binary tree.

Picture of beach-line @ $\ell$ :




Associated tree :

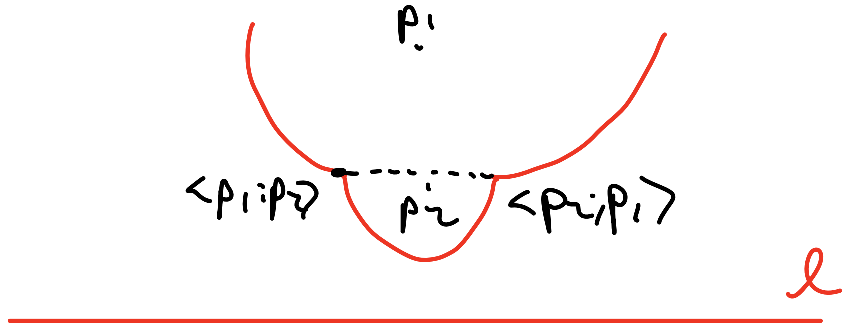



• leaves = arcs of beach-line

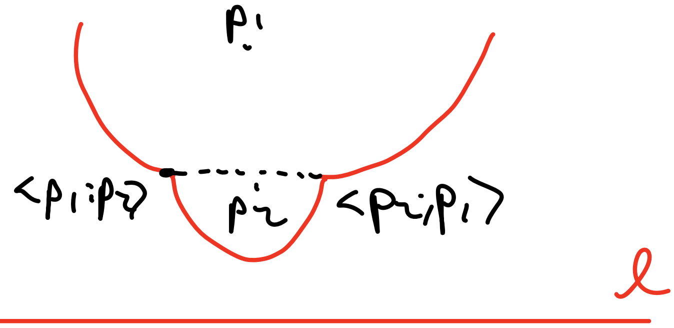
• Internal nodes of tree $\langle p_i : p_j \rangle$ represent "breakpoints" on beach-line, at which paraboloi around $p_i$ & $p_j$ meet with arc of $p_i$ to left & arc of $p_j$ to the right.

- Given a point $a$ on $\ell$, can search for arc of beach-line above $a$.

# Creating edges of Voronoi diagram 1

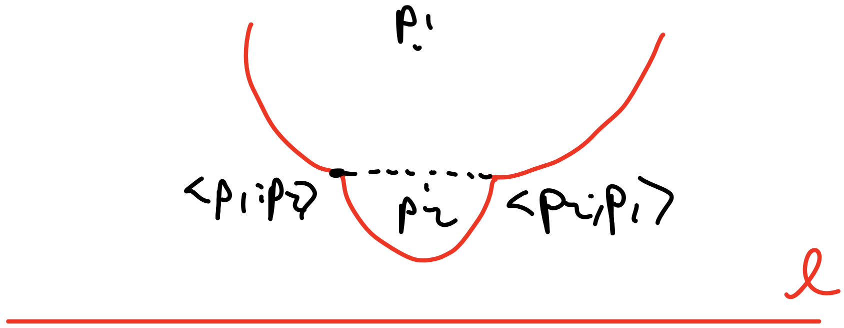


# Creating edges of Voronoi diagram I



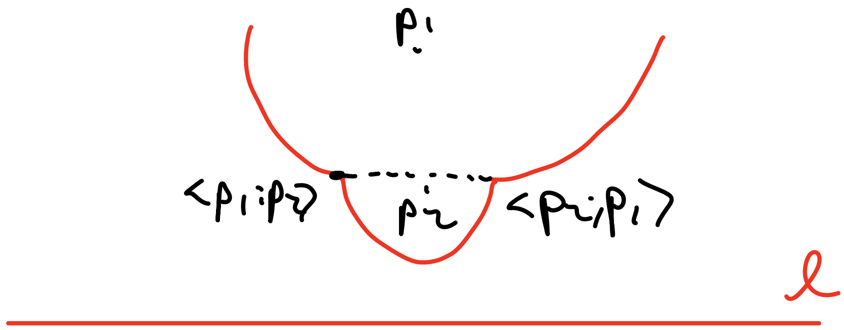

- At breakpoint $r = \langle p_1 : p_2 \rangle$, we have
  $d(r, p_1) = d(r, \ell) = d(r, p_2)$.

# Creating edges of Voronoi diagram 1




- At breakpoint $r = \langle p_1 : p_2 \rangle$, we have $d(r, p_1) = d(r, \ell) = d(r, p_2)$.
- This means that $r$ lies on an edge of the V-diagram.

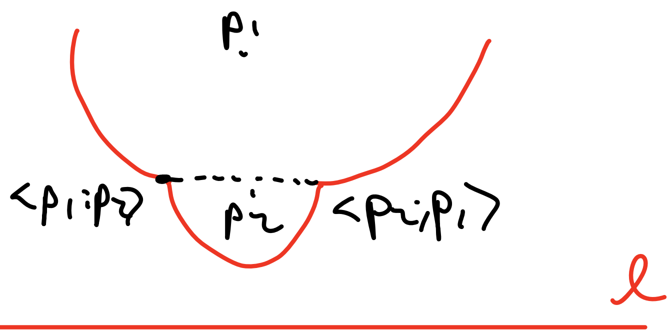
# Creating edges of Voronoi diagram 1




- At breakpoint $r = \langle p_1 : p_2 \rangle$, we have $d(r, p_1) = d(r, \ell) = d(r, p_2)$.
- This means that $r$ lies on an edge of the V-diagram.

If $\langle p_2 : p_1 \rangle$ is on beach-line, then it has same distance from $p_1$ & $p_2$.

# Creating edges of Voronoi diagram I



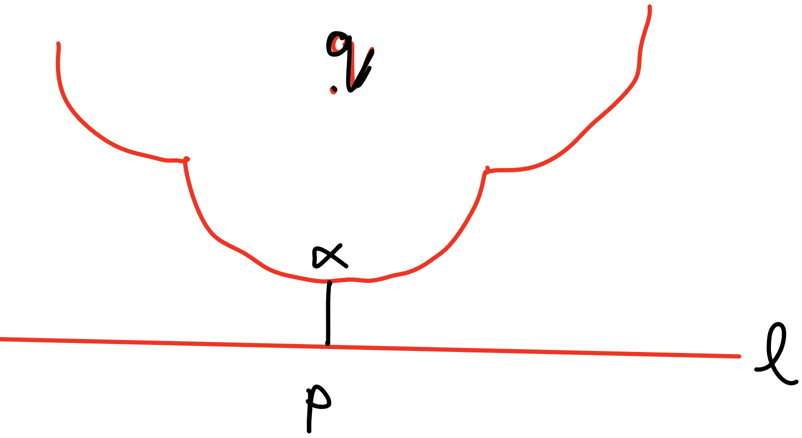

- At breakpoint $r = \langle p_1 : p_2 \rangle$, we have $d(r, p_1) = d(r, \ell) = d(r, p_2)$.
- This means that $r$ lies on an edge of the V-diagram.

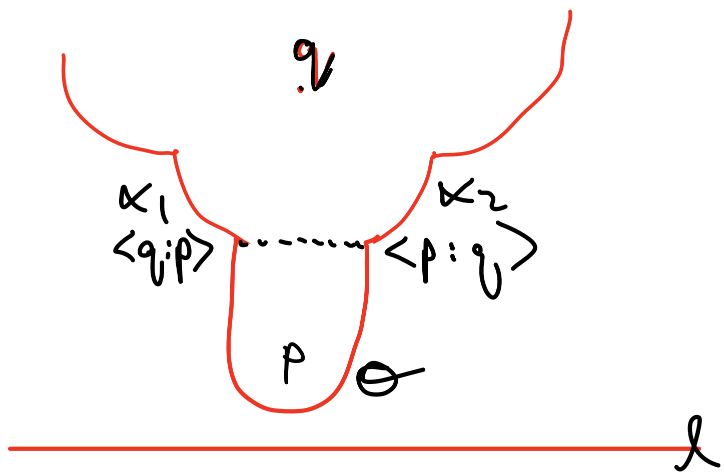
- If $\langle p_2 : p_1 \rangle$ is on beach-line, then it has same distance from $p_1$ & $p_2$.

- Therefore the edge from $\langle p_1 : p_2 \rangle$ to $\langle p_2 : p_1 \rangle$ will lie on $V(P)$

# Creating edges of Voronoi diagram 2

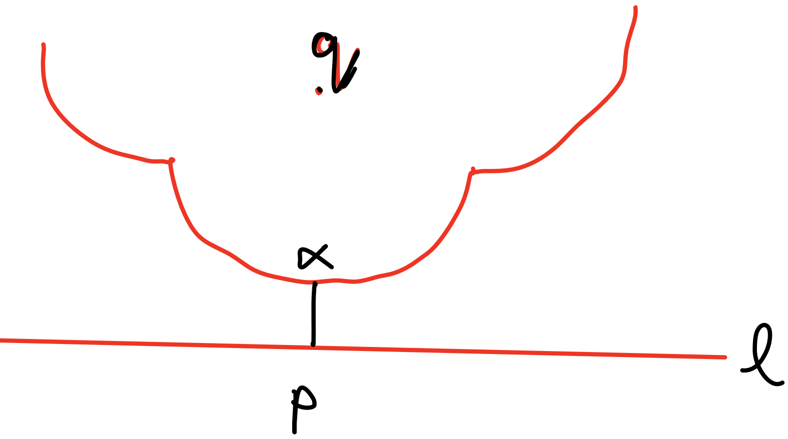
- A new arc appears just when the sweep-line passes a point of P.

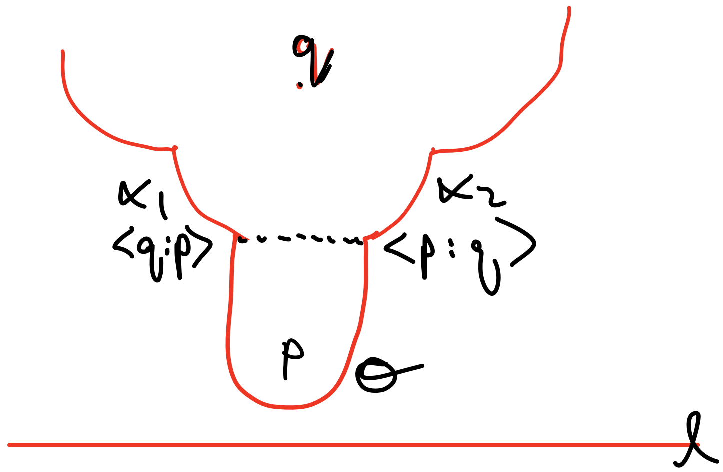


# Creating edges of Voronoi diagram 2

- A new arc appears just when the sweep-line passes a point of $P$.

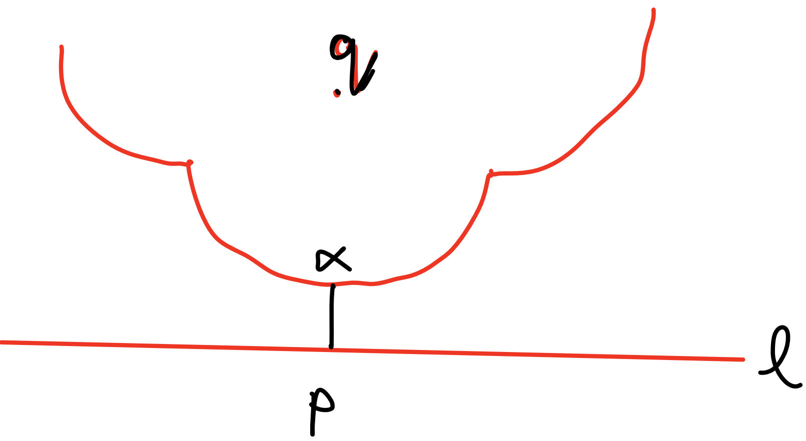


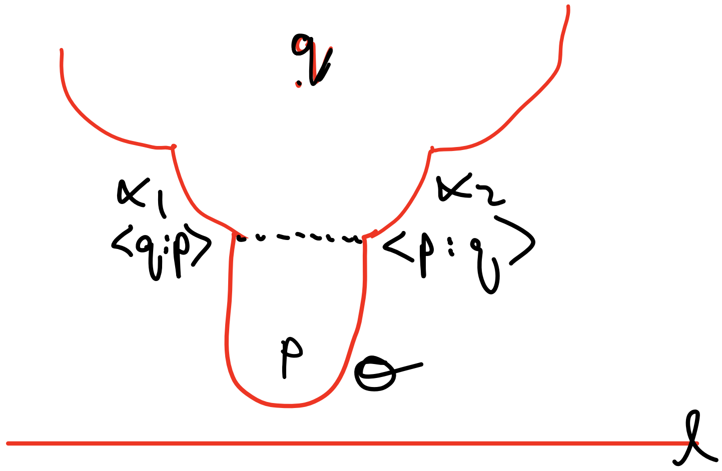




- In this case, we add edge between the new breakpoints.

# Creating edges of Voronoi diagram 2

- A new arc appears just when the sweep-line passes a point of P.



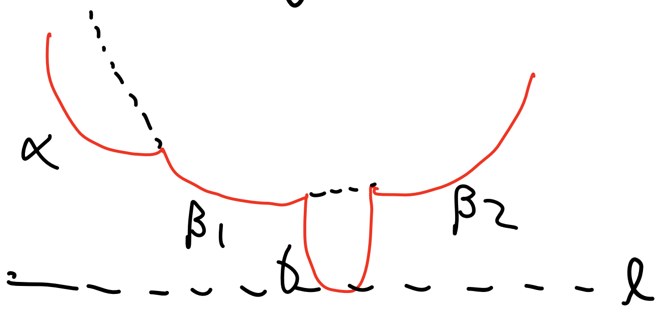




- In this case, we add edge between the new breakpoints.

- We will see that vertices of V(P) are created when an arc disappears from the beach-line.

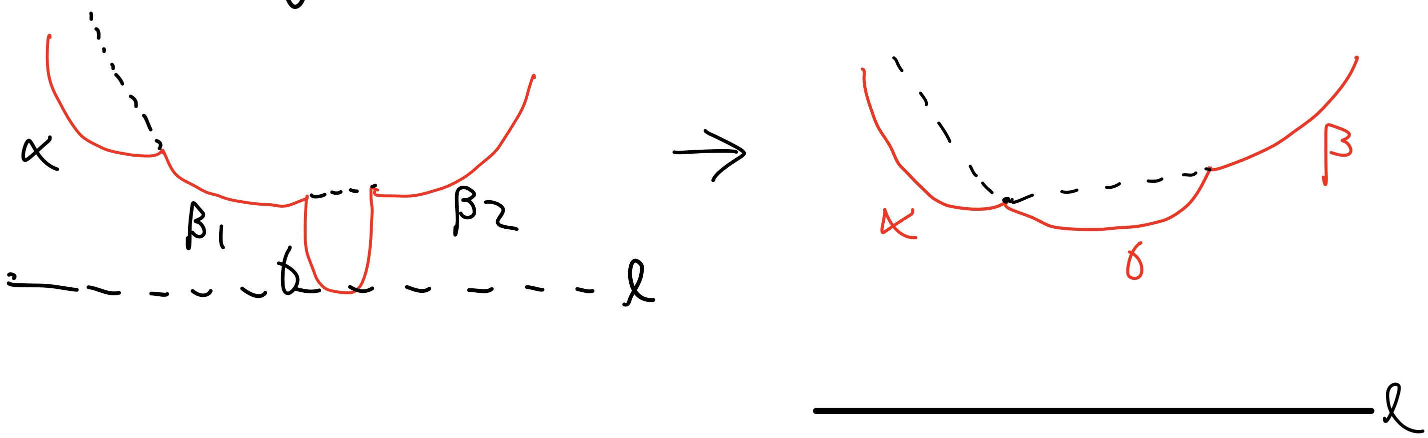
# When does an arc disappear?

- Intuitively, when squeezed out by two adjacent arcs.



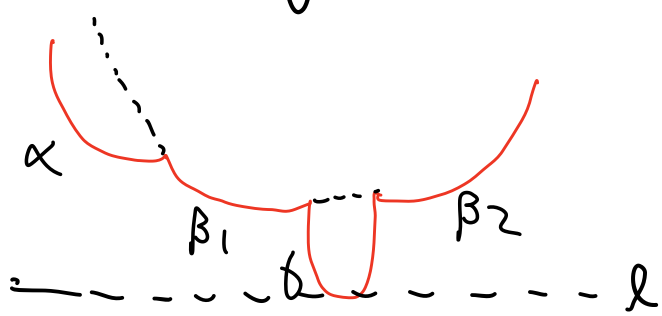
# When does an arc disappear?

- Intuitively, when squeezed out by two adjacent arcs.

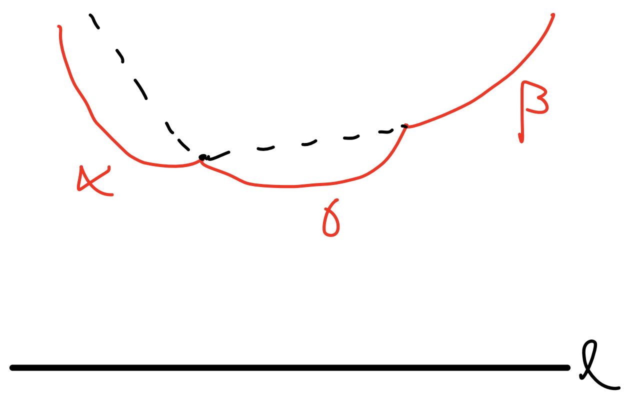


# When does an arc disappear?

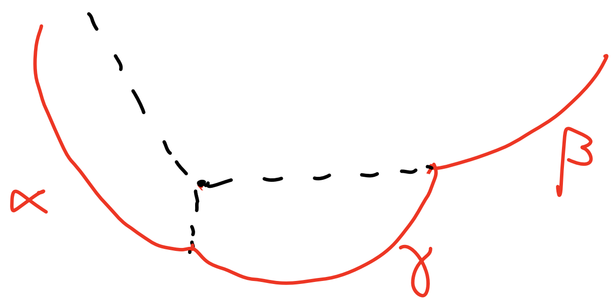
- Intuitively, when squeezed out by two adjacent arcs.




$\rightarrow$



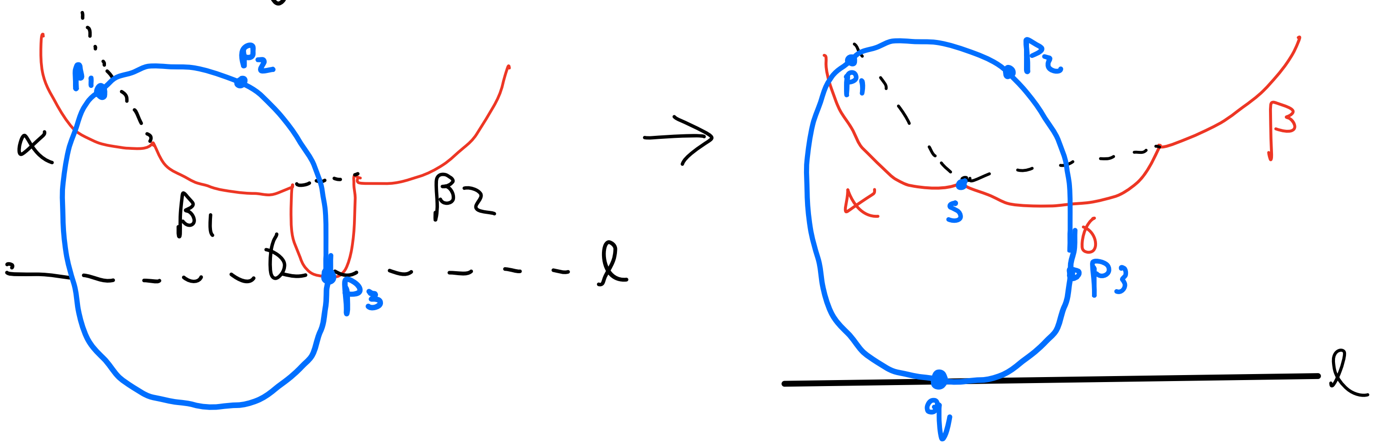

$\rightarrow$




See Fig 9.8.

# When does an arc disappear?

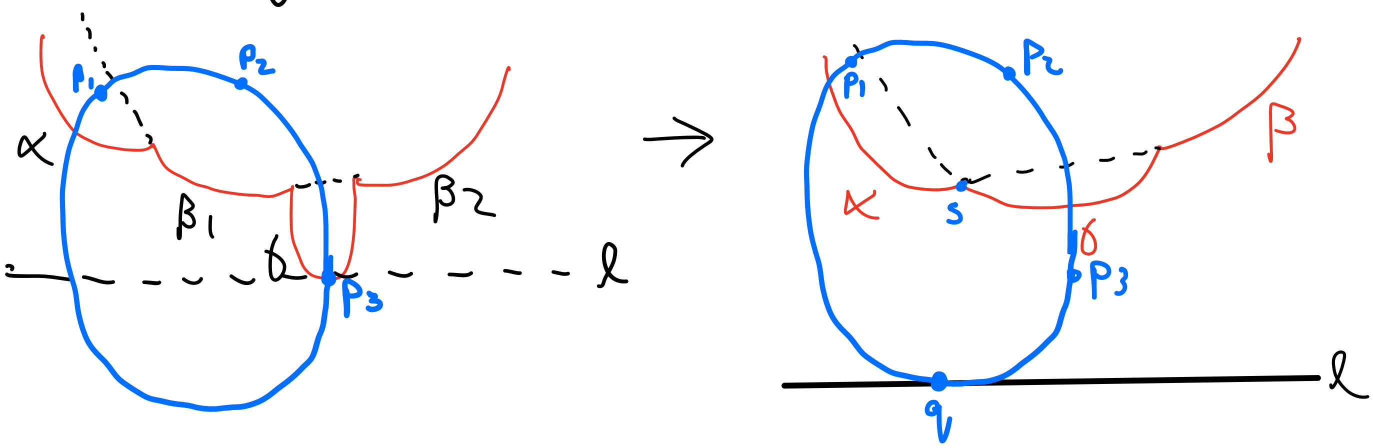
- Intuitively, when squeezed out by two adjacent arcs.




- Consider consecutive arcs $\alpha, \beta, \gamma$ with foci $p_1, p_2, p_3$.

# When does an arc disappear?

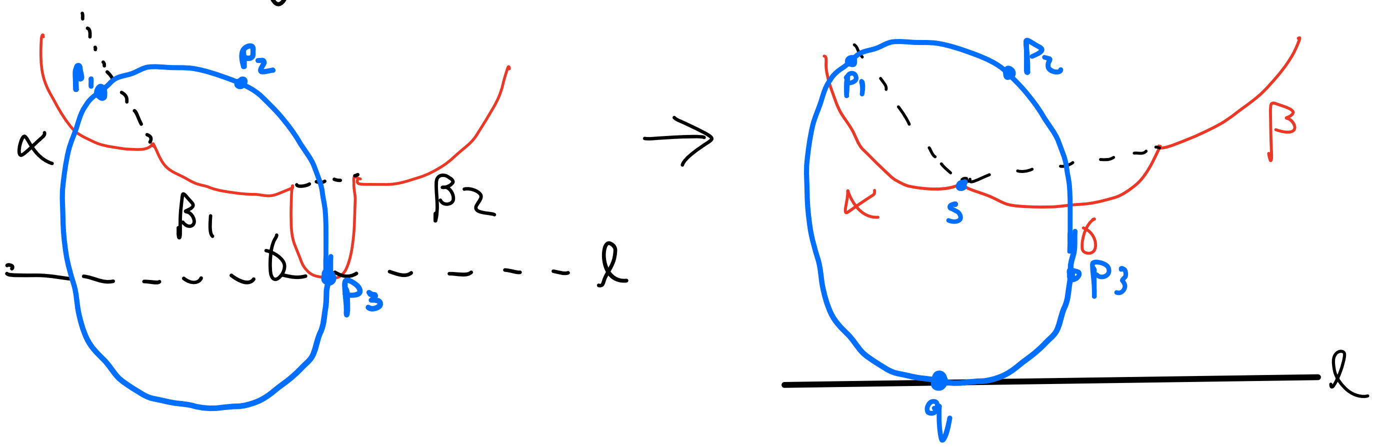
- Intuitively, when squeezed out by two adjacent arcs.




- Consider consecutive arcs $\alpha, \beta, \gamma$ with foci $p_1, p_2, p_3$.
  - Any 3 points lie on a unique circle with lowest point $q$ & centre $s$.

# When does an arc disappear?

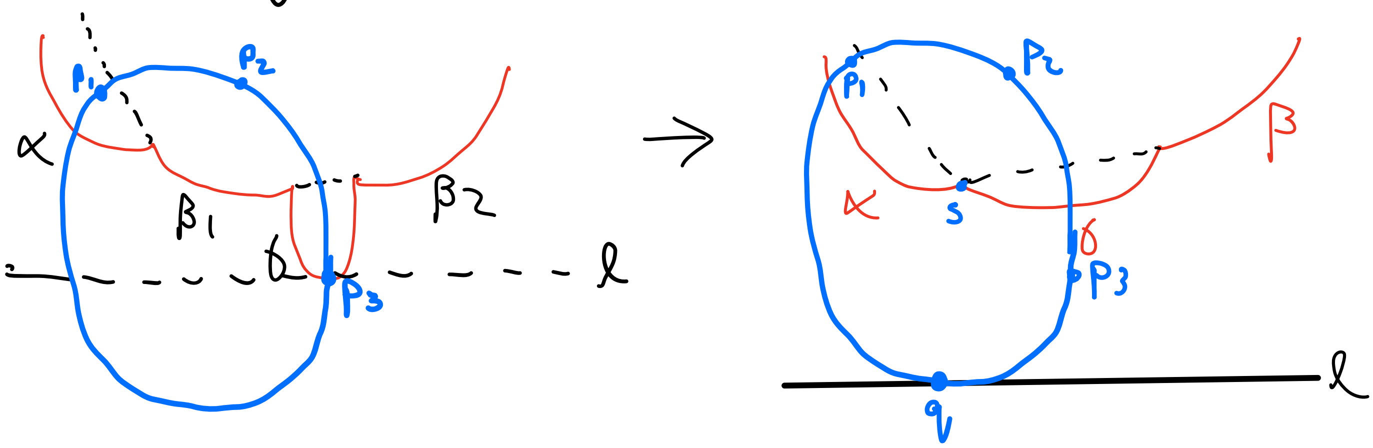
- Intuitively, when squeezed out by two adjacent arcs.




- Consider consecutive arcs $\alpha, \beta, \gamma$ with foci $p_1, p_2, p_3$.
  - Any 3 points lie on a unique circle with lowest point $q$ & centre $s$.
  - If $q$ lies below $l$, it is called a circle event for $\beta_1$.

# When does an arc disappear?

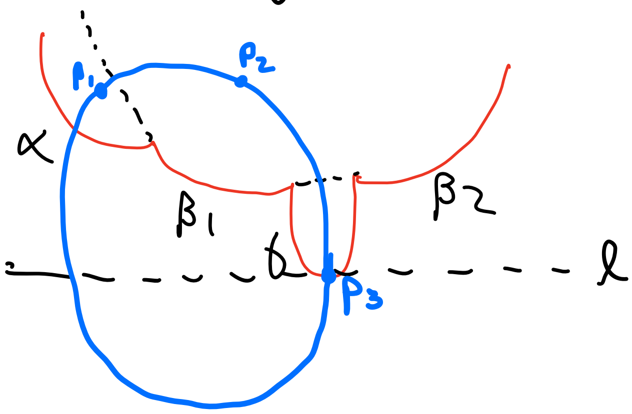
- Intuitively, when squeezed out by two adjacent arcs.



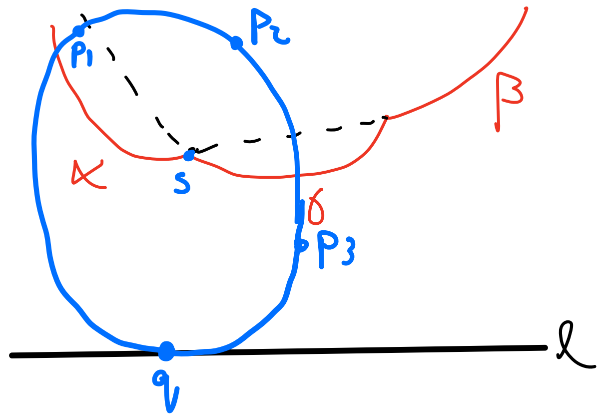

- Consider consecutive arcs $\alpha, \beta, \gamma$ with foci $p_1, p_2, p_3$.
  - Any 3 points lie on a unique circle with lowest point $q$ & centre $s$.
  - If $q$ lies below $\ell$, it is called a circle event for $\beta_1$.
- Because when $\ell$ passes $q$,
  $d(s, p_1) = d(s, p_2) = d(s, p_3) = d(s, \ell)$
  & so $s$ lies on all 3 arcs.

# When does an arc disappear?

- Intuitively, when squeezed out by two adjacent arcs.



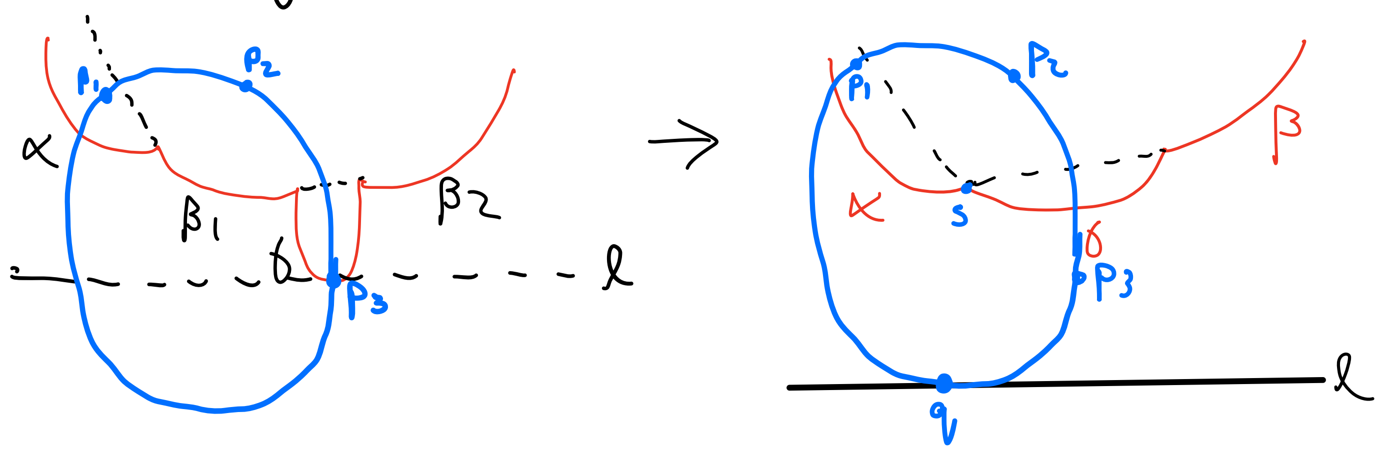

$\rightarrow$




- Consider consecutive arcs $\alpha, \beta, \gamma$ with foci $p_1, p_2, p_3$.
  - Any 3 points lie on a unique circle with lowest point $q$ & centre $s$.
    - If $q$ lies below $\ell$, it is called a circle event for $\beta_1$.
- Because when $\ell$ passes $q$,
  $d(s, p_1) = d(s, p_2) = d(s, p_3) = d(s, \ell)$
  & so $s$ lies on all 3 arcs.
- Then $\alpha$ & $\gamma$ meet at $s$ & $\beta_1$ disappears.

# When does an arc disappear?

- Intuitively, when squeezed out by two adjacent arcs.




- Consider consecutive arcs $\alpha, \beta_1, \gamma$ with foci $p_1, p_2, p_3$.
  - Any 3 points lie on a unique circle with lowest point $q$ & centre $s$.
    - If $q$ lies below $\ell$, it is called a circle event for $\beta_1$.
- Because when $\ell$ passes $q$,

$$d(s, p_1) = d(s, p_2) = d(s, p_3) = d(s, \ell)$$

& so $s$ lies on all 3 arcs.

- Then $\alpha$ & $\gamma$ meet at $s$ & $\beta_1$ disappears.
- A new vertex is created at $s$, & a new half-edge.

## Algorithm: key structures

- event queue Q (actually a binbalT)
- bbTree T for beach line
- DCEL for Voronoi diagram.

# Algorithm : Key structures

- event queue Q (actually a binbalT)
- bbTree T for beach line
- DCEL for Voronoi diagram.

- Moreover each leaf of T (arc of beach line) has a pointer to the event queue, its circle event (if it exists, else nil.).

# Algorithm : key structures

- event queue Q (actually a binbalT)
- bbtree T for beach line
- DCEL for Voronoi diagram.

- Moreover each leaf of T (arc of beach line) has a pointer to the event queue, its circle event (if it exists, else nil.).

- Each internal node (break point on beach line) has pointer to half-edge in DCEL being traced out by break point.

# Algorithm : overview

- At beginning, add all points of P to Q - called <u>site events</u>.

# Algorithm : overview

- At beginning, add all points of P to Q - called site events.
- At site event p : remove p from Q
  - create new arc $\alpha$ of beach line,
  - new edge of V. diagram

# Algorithm: overview

- At beginning, add all points of P to Q - called site events.

- At site event p: remove p from Q
  - create new arc $\alpha$ of beach line,
  - new edge of V. diagram
  - search for circle events for neighbours of $\alpha$ & add to queue.

# Algorithm : overview

- At beginning, add all points of P to Q - called site events.
- At site event p : remove p from Q
  - create new arc α of beach line,
  - new edge of V. diagram
  - search for circle events for neighbours of α & add to queue.
- At circle event q for β,
  - remove q from Q,

## Algorithm : overview

- At beginning, add all points of P to Q - called site events.
- At site event p : remove p from Q
  - create new arc $\alpha$ of beach line,
  - new edge of V. diagram
  - search for circle events for neighbours of $\alpha$ & add to queue.
- At circle event q for $\beta$,
  - remove q from Q,
  - remove $\beta$ from T

# Algorithm : overview

- At beginning, add all points of P to Q - called site events.

At site event p : remove p from Q
- create new arc α of beach line,
- new edge of V. diagram
- search for circle events for neighbours of α & add to queue.

At circle event q for β,
- remove q from Q,
- remove β from T
- create new vertex of V-diagram & half-edge.

# Algorithm : overview

- At beginning, add all points of P to Q - called site events.
- At site event p : remove p from Q
  - create new arc $\alpha$ of beach line,
  - new edge of V. diagram
  - search for circle events for neighbours of $\alpha$ & add to queue.
- At circle event q for $\beta$,
  - remove q from Q,
  - remove $\beta$ from T
  - create new vertex of V-diagram & half-edge.
  - remove circle events for neighbours of $\beta$ & search for new ones.

- When no pts left in $Q$, may be still some internal nodes in $T$, which correspond to half-infinite edges of $V(P)$.

- When no pts left in Q, may be still some internal nodes in T, which correspond to half-infinite edges of V(P).
- Compute bounding box containing all vertices of V(P) in interior, & connect half-infinite edges to bounding box.

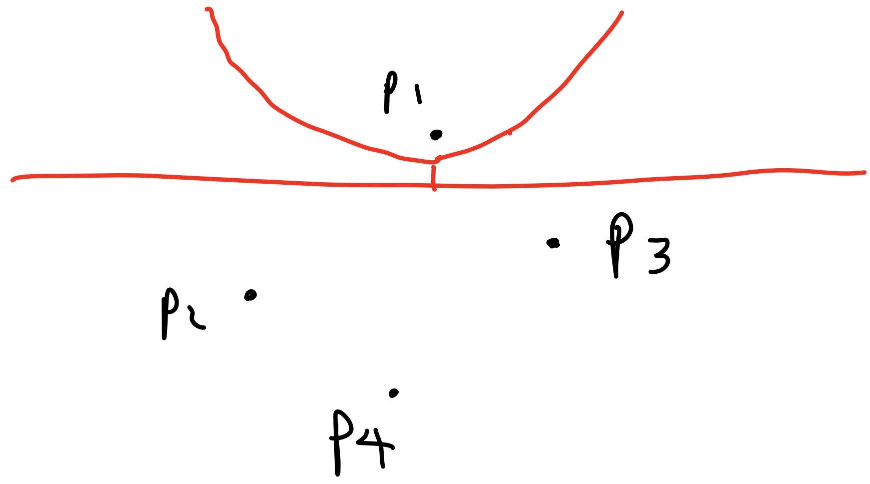
- When no pts left in Q, may be still some internal nodes in T, which correspond to half-infinite edges of V(P).
- Compute bounding box containing all vertices of V(P) in interior, & connect half-infinite edges to bounding box.

Complexity: $O(n\log n)$

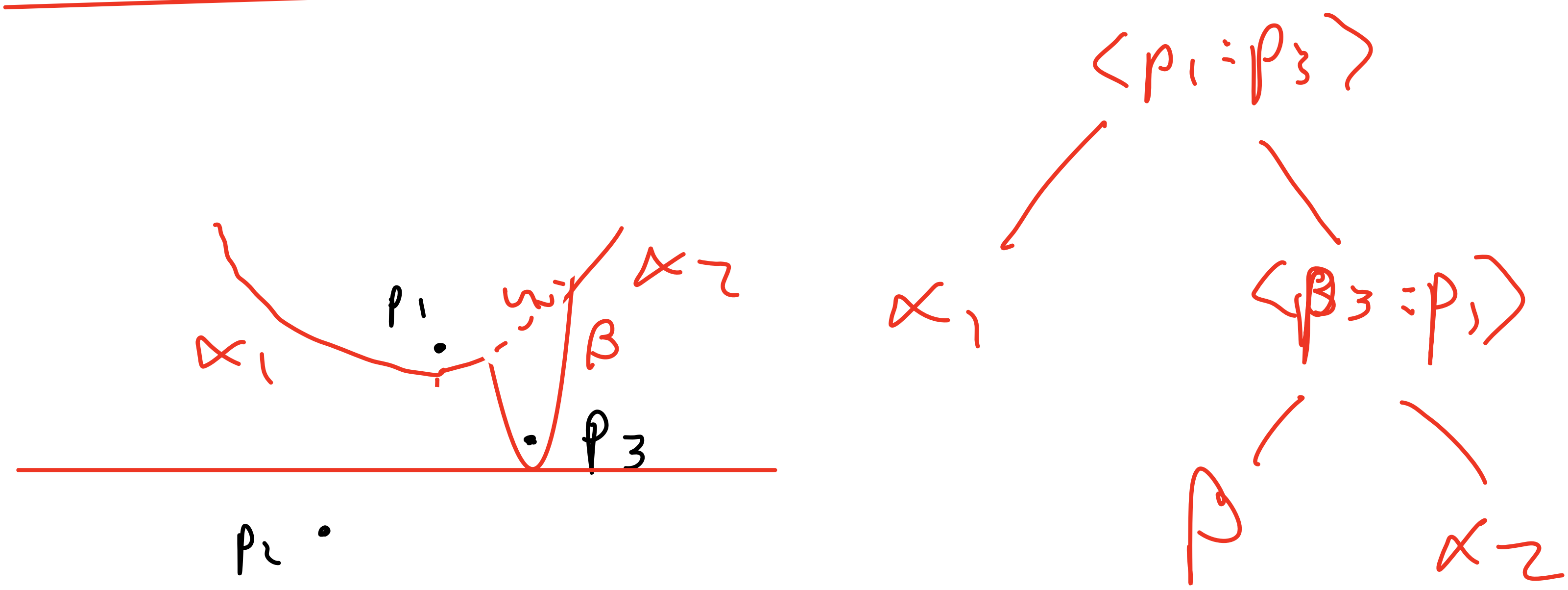
- When no pts left in Q, may be still some internal nodes in T, which correspond to half-infinite edges of V(P).
- Compute bounding box containing all vertices of V(P) in interior, & connect half-infinite edges to bounding box.

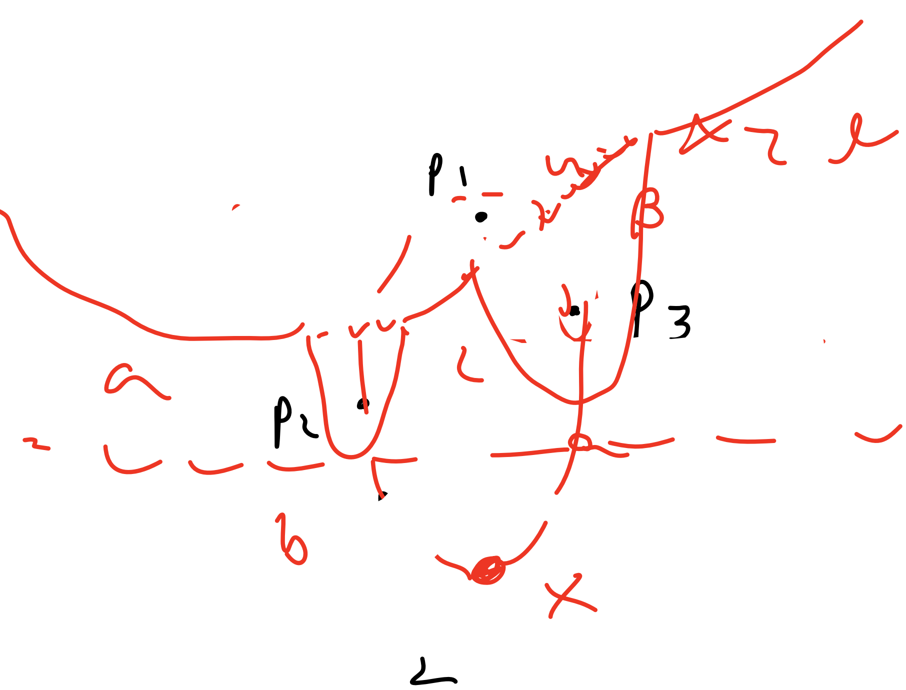
Complexity : $O(n \log n)$

- Lecture focused on geometric ideas involved in constructing V-diagram - see E-Learning for further detail on implementation, also thesis linked to at the end.

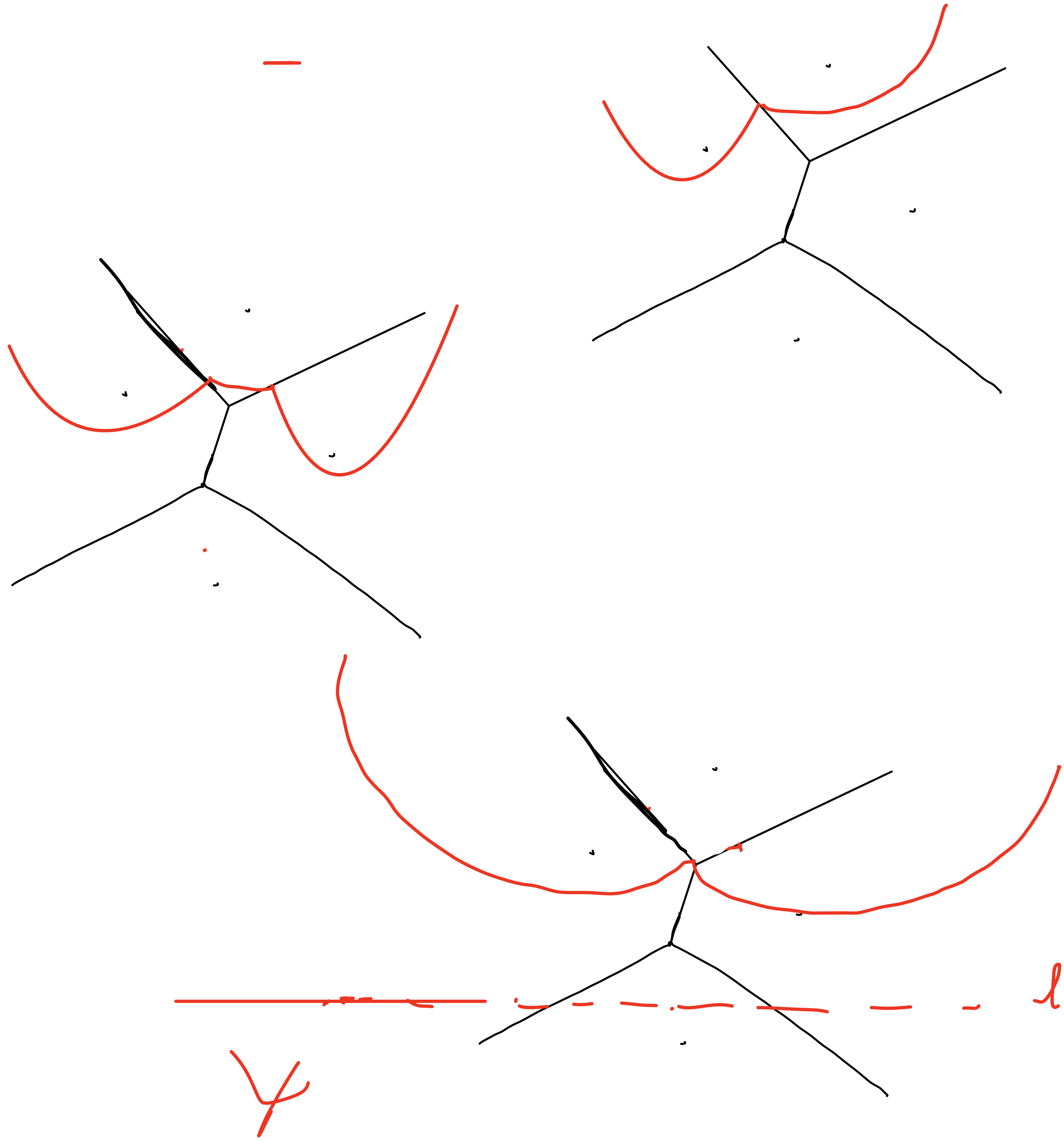



$\boxed{\alpha}$




circle event
for c


ℓ
γ