Computational Geometry

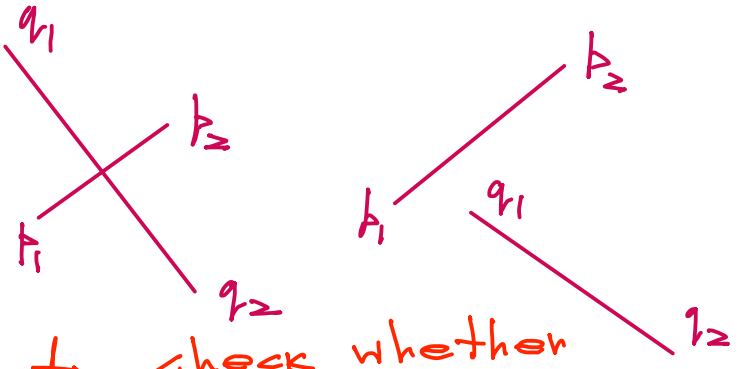
Books:-

1) BCKO:- Computational Geometry – Algorithms & Applications – Springer

⇒ David Mount's Lecture Notes (pdf)

Objects:-

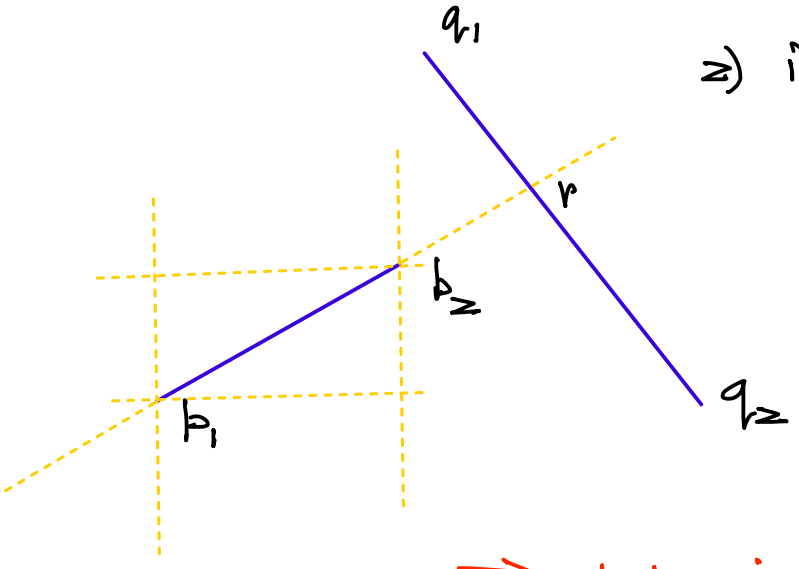
- Points
- Lines
- Line segments
- Circles
- Polygons



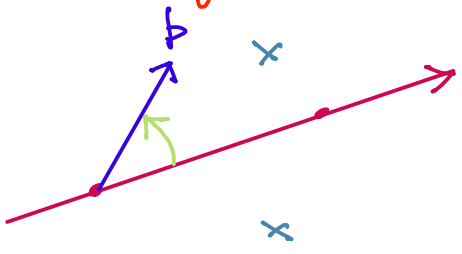
Q) Given two line segments (Non Parallel), check whether they intersect

A) 1) $r \leftarrow$ point of intersection b/w $\overleftrightarrow{p_1p_2}$ & $\overleftrightarrow{q_1q_2}$

2) if $r \in Rect(p_1, p_2)$ & $r \in Rect(q_1, q_2)$ then YES else NO



Q: Given a vector $\overrightarrow{ab}$, determine whether point p lies on or left or on the right side of ab.



A: 1) Compute $\overrightarrow{ap} \times \overrightarrow{ab}$ (cross prod)

2) If the value is 0, then $p \in \overrightarrow{ab}$. If value < 0, then $p \in Left(\overrightarrow{ab})$ else $p \in Right(\overrightarrow{ab})$

$a = (a_x, a_y)$ $b = (b_x, b_y)$

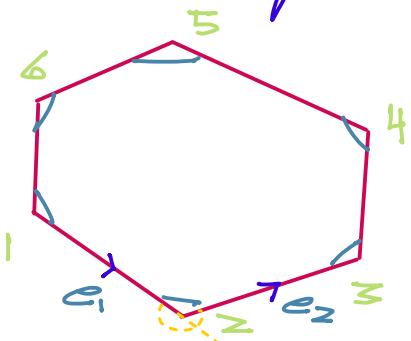
$p = (p_x, p_y)$

$\overrightarrow{ab} = (b_x - a_x)\hat{x} + (b_y - a_y)\hat{y}$

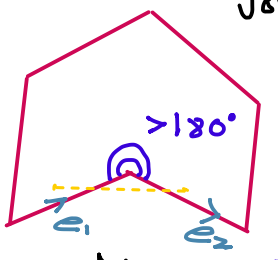
$\overrightarrow{ap} = (p_x - a_x)\hat{x} + (p_y - a_y)\hat{y}$

Convex Polygon

P is a convex polygon iff for any two points a and b in P the line segment lies entirely inside P



Yes $\vec{e_1} \times \vec{e_2} > 0$

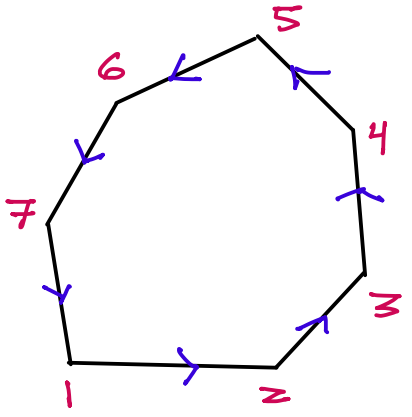


No $\vec{e_1} \times \vec{e_2} < 0$

Q) How to check whether a polygon is convex?

Each internal angle < 180°

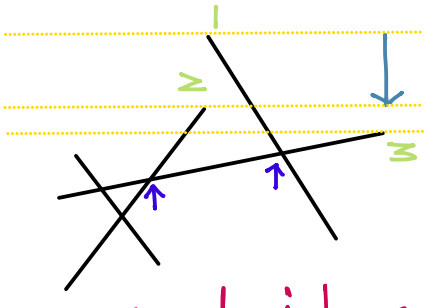
Q) Given a convex polygon P and a point q, determine whether $q \in P$ $\Rightarrow$ left of each edge



Line Segment Intersection

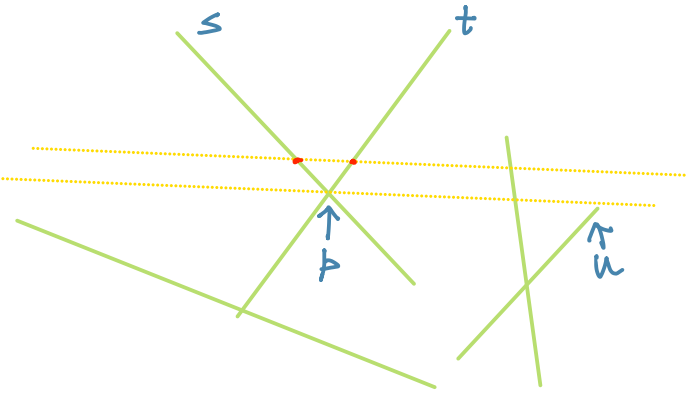
Q) Given n line segments, find their point of intersection

Brute force $\rightarrow O(n^2)$



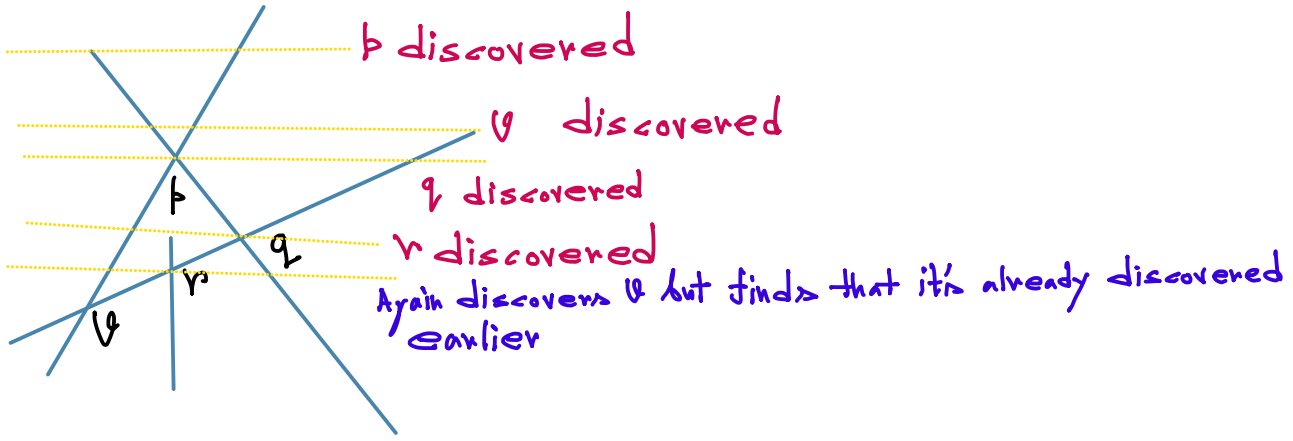
Sweep Line (λ)

If 2 segments intersect, they have to be adjacent at some time.



Observation 1:- If two segments are never adjacent to each other in the sequence of segments intersected by λ, then s and t never intersect each other.

Conversely, s and t may intersect each other if they are adjacent at some instant when λ sweeps down

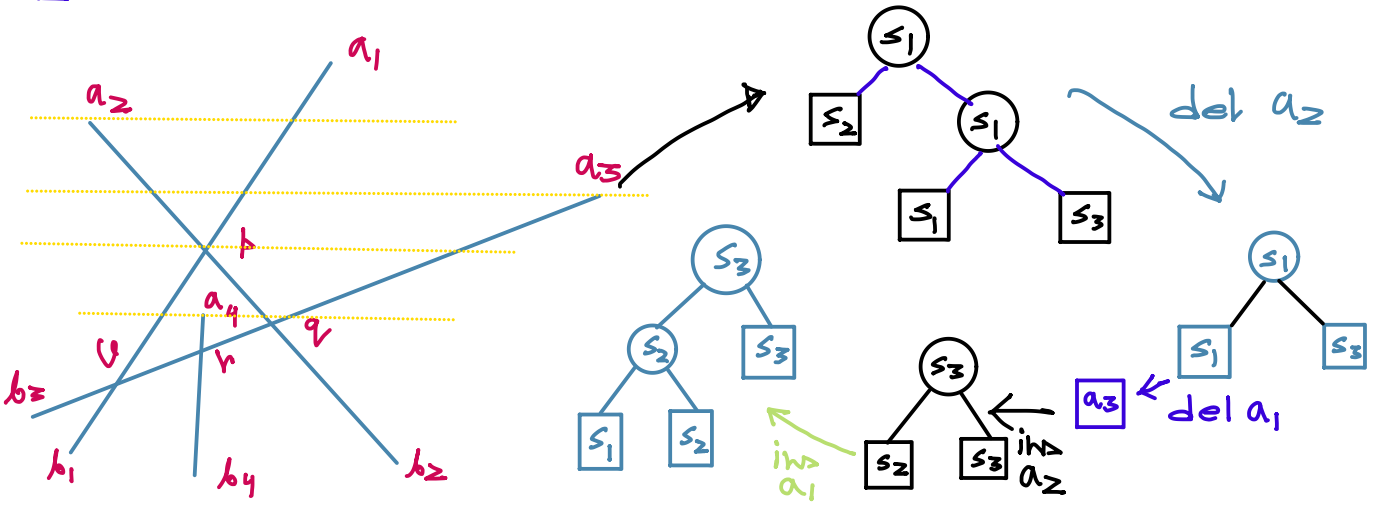


Observation 2:- Adjacency changes only when λ encounters a new event point. (= End point of a segment or an intersection point)

Data Structures:-

1) Event queue Q: stores all event points, finds and removes the topmost event p.t, searches and inserts new event points [Height balanced B.S.T. e.g. AVL Tree]

2) Sweep line λ:- stores the segments currently intersected by λ with left-to-right ordering, updates the ordering and inserts/removes segments upon encountering an event point. [AVL Tree]



$p=(s_2,s_1)$ $q=(s_2,s_3)$ $r=(s_4,s_3)$ $v=(s_3,s_4)$

$\text{Size}(\lambda)=O(n)$

$\text{Size}(Q)=O(n+K)$ [$K=$ # intersection pts]

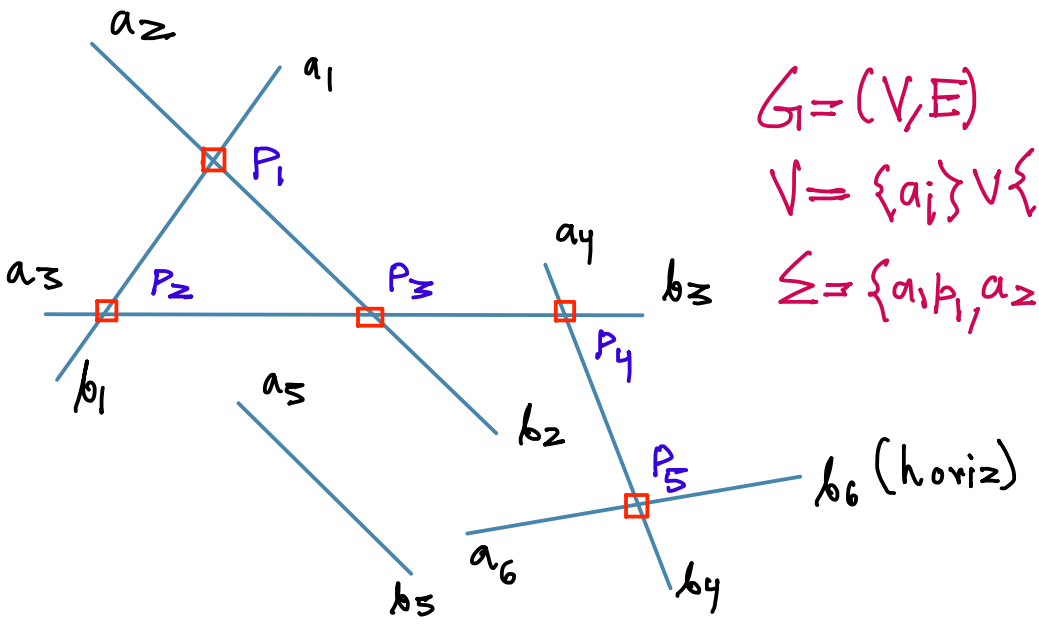
Every operation takes $O(\log n)$ time

$\Rightarrow$ The graph is equivalent to a planar graph

$G=(V,E)$, $|V|\le 2n+K$

$|V|-|E|+|F|\ge 2$ [Euler]

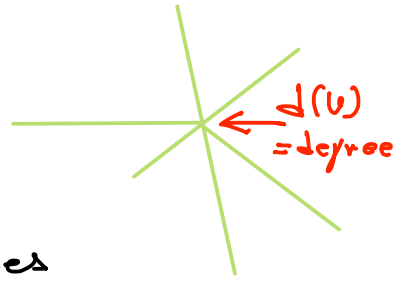
Connected (=)



$G=(V,E)$

$V=\{a_i\}\cup\{b_i\}\cup\{p_j\}$, $1\le i\le 6$, $1\le j\le 5$

$E=\{a_1p_1, a_2p_1, b_1p_2, \dots, p_5b_6\}$



Euler Formula:-

$n_v=$ # Vertices $\quad n_e=$ # edges $\quad n_f=$ # faces

① $n_v-n_e+n_f\ge 2$ ② $\sum_{v\in V} d(v)=2n_e$

$n=$ # segments

$K=$ # intersection pts

③ $n_v\le 2n+K$

④ $3(n_f-1)\le 2n_e$

Total Time

$\sum_{v\in V} d(v)\cdot O(\log(2n+K))$ $\left[\because |Q|\le 2n+K,\ |\lambda|\le n\right]$

$O(n_e \underbrace{\log(2n+K)}_{=O(\log n)})$

$=O(n_e\log n)$

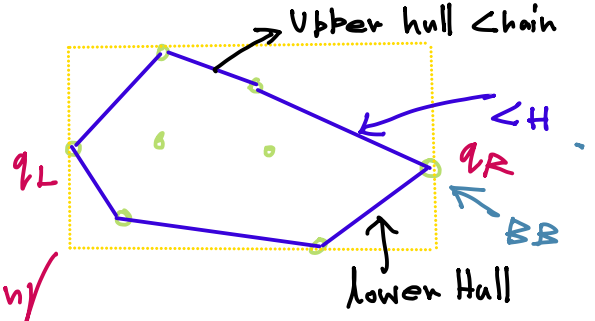
Planar Graph

$n_e=O(n_v)$

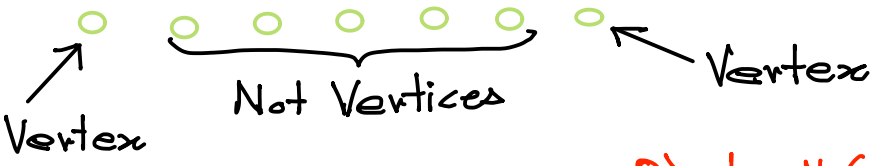
Covers

1) Bounding Box (BB) — $O(n)$ time
2) Min enclosing disk (MED)
3) Convex Hull (CH) : $(V(CH), E(CH))$

⇒ Smallest Perimeter Polygon containing all the points.

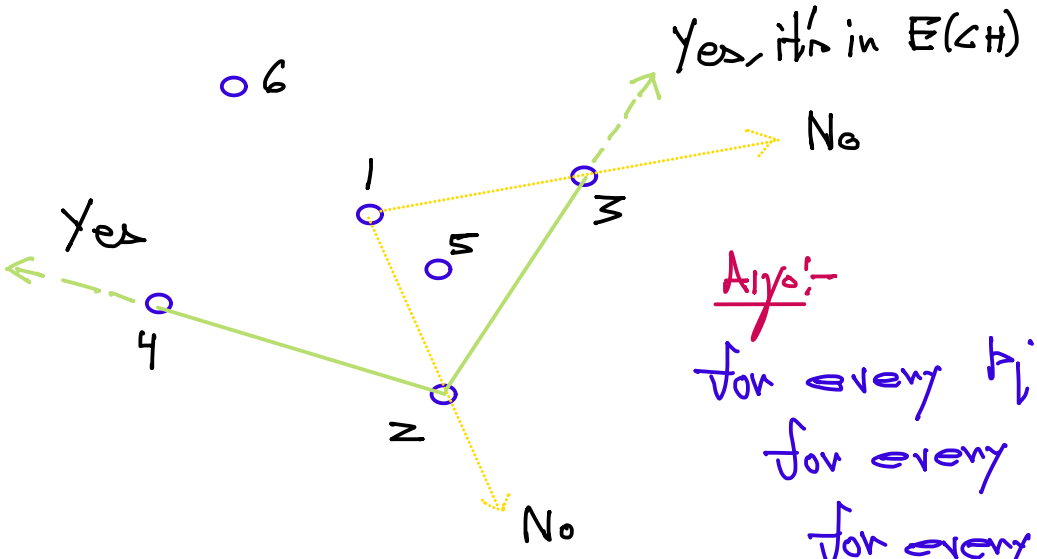


- A convex polygon with ≥ 3 vertices



$S = \{p_1, p_2, \ldots, p_n\}$

Q) Is $V(CH) \subseteq S$? ⇒ Yes?



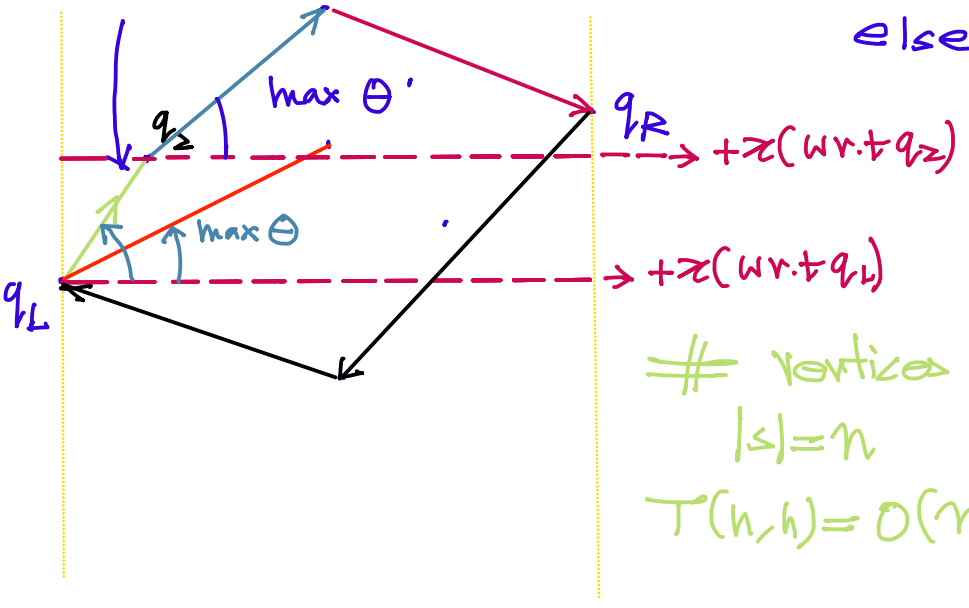
$CH \subseteq BB$

Algo:-

for every $p_i \in S$,
  for every $p_j \in S$ $(j > i)$
    for every $p_k \in S - \{p_i, p_j\}$
      if lies on the same side
      then Yes
      else No

Time $= O(n^3)$

$O(n^2)$ time



vertices of $CH(S) = h$

$|S| = n$

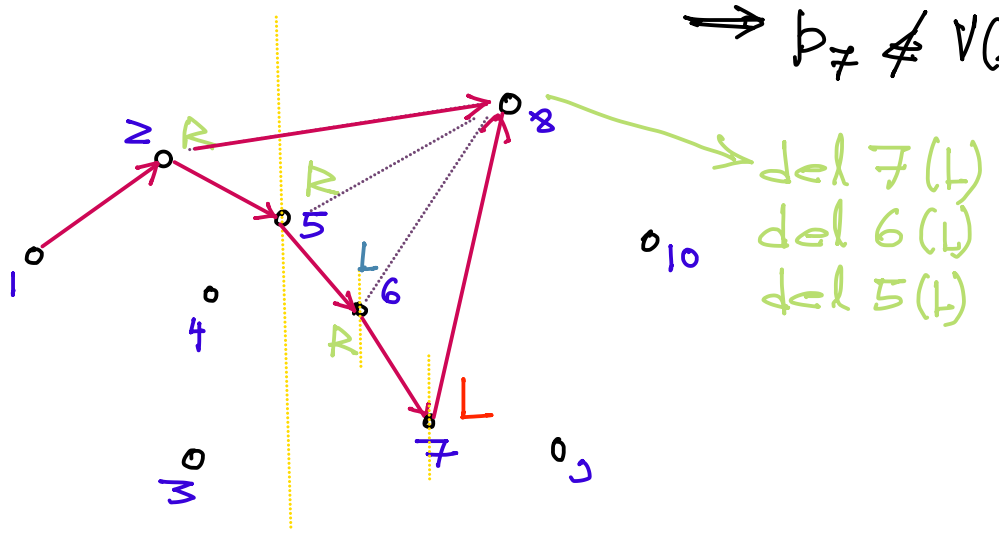
$T(n, h) = O(nh)$

Jarvis March

Graham Scan

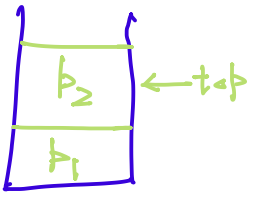
$\Rightarrow p_7 \notin V(CH(p_1, p_2, \ldots, p_8))_{upper}$

$\Rightarrow p_7 \notin V(CH(S))_{upper}$



Upper Hull

only once // for both up & lower hull chains



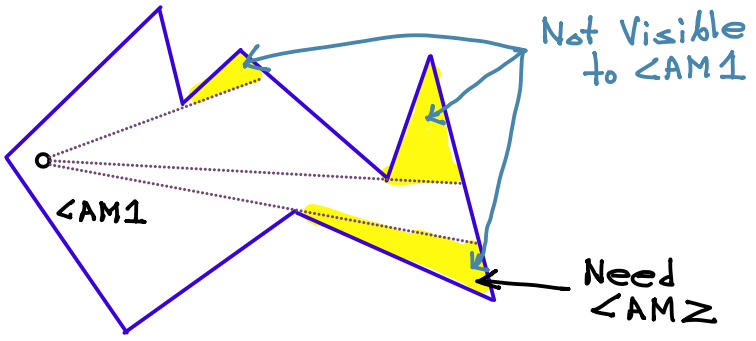
1. Sort $S \rightarrow p_1, p_2, \ldots, p_n$
2. initialize stack K
3. $i \leftarrow 3$
4. While $(i \leq n)$
5. $\quad q \leftarrow pop(K);\ r \leftarrow pop(K)$
6. $\quad$ if $turn(r, q, p_i) = 'L'$
7. $\quad\quad push(K, r)$ and goto step 5.
8. $\quad push(K, r); push(q); push(p_i)$
9. $\quad i{+}{+}$; // end while
10. Pop all from K

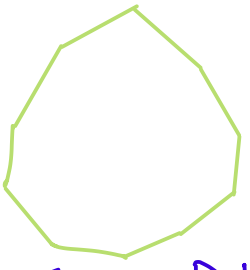
Convex Hull Algorithms:-

1) Naive Algorithm
2) Jarvis March — $O(nh)$ time, h = # vertices of CH
3) Graham Scan — $O(n \log n)$ time
4) Divide and Conquer — $O(n \log h)$ time
5) Kirpatrick–Siedel — $O(n \log h)$ time
6) T. Chan — $O(n \log h)$ time

Art Gallery Problem

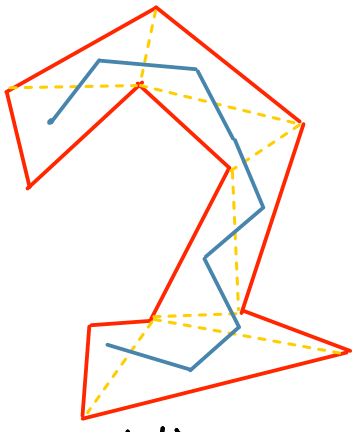
Min # Cameras for complete visibility





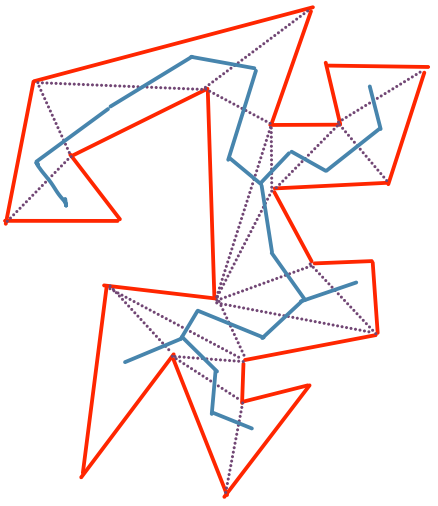
Convex Polygon

1 Cam is enough



Triangulation

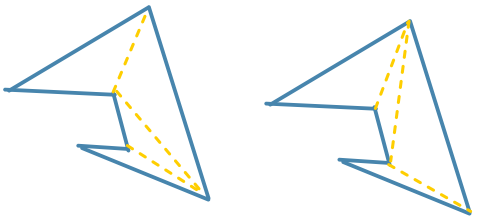
$T(P)$



Dual Graph of T

$G_T = (V_T, E_T)$

- Not Unique



Given the vertices of P in ↺ order, find a triangulation $T(P)$ of P by adding diagonals that do not intersect each other except at the vertices of P

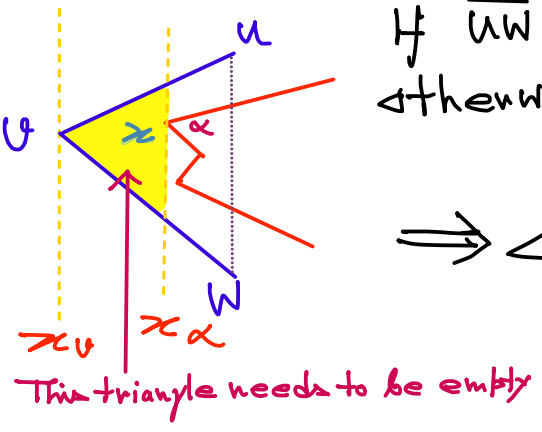
- # triangles $= n-2$

if P has n vertices

Claim: Any Polygon with $n \geq 4$ vertices has a diagonal.

Proof: Consider a leftmost vertex of P, say v.

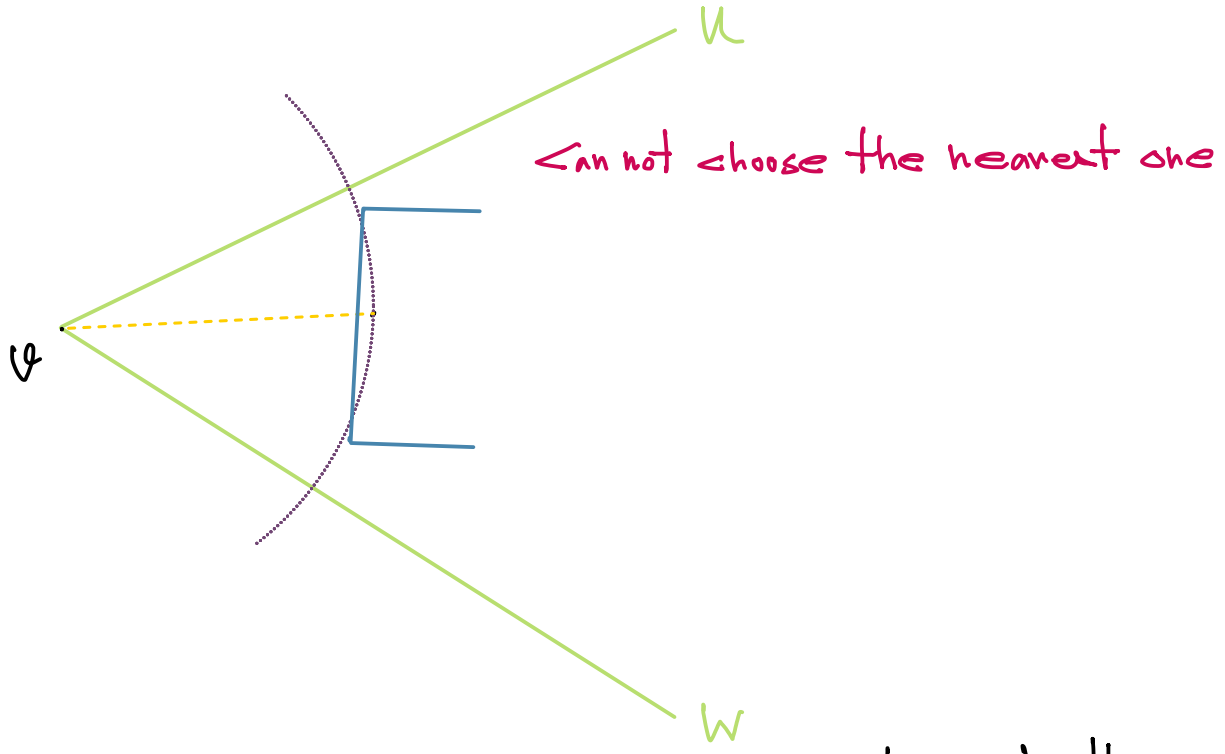
Let u and w be the adjacent vertices of v.



If $\overline{uw}$ lies in P, then we are done.

otherwise there is a chain inside Δuvw

$\Longrightarrow$ Choose the leftmost one



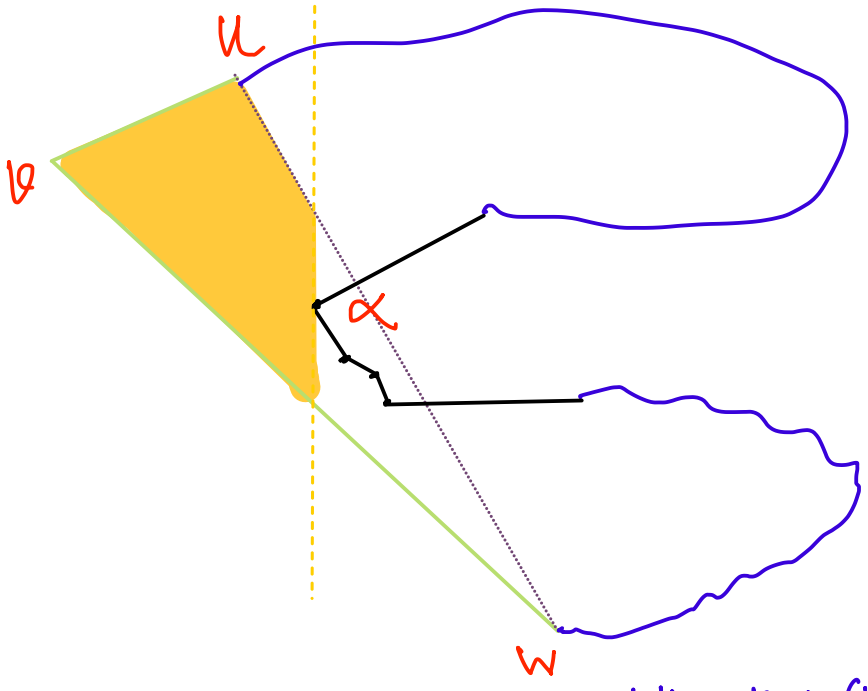
Let $\alpha_1, \ldots, \alpha_k$ be the vertices in the chain

Let $\alpha = \text{leftmost}\{\alpha_1, \ldots, \alpha_k\}$

Let R = vertical strip: $x_v \le x \le x_\alpha$

Interior of $\Delta uvw \cap R$ is empty

$\rightarrow \overline{v\alpha}$ = diag



Theorem:- # triangles in triangulation $= n-2$ $(n \geq 3)$

Proof:- Basis:- $n=3$ true

Hypothesis:- True for any # vertices $< n$

Step:- Let $\overline{uv}$ be a diagonal of P.

Use $\overline{uv}$ to break P into two smaller polygons P_1 and P_2 with n_1 and n_2 vertices respectively

$$n_1 + n_2 - 2 = n \quad \text{—— (1)}$$

$|T(P_1)| = n_1 - 2$

$|T(P_2)| = n_2 - 2$

$\Rightarrow |T(P)| = n_1 - 2 + n_2 - 2 = n - 2$

$\Longrightarrow$ To guard a polygon with n vertices, $n-2$ cameras are sufficient

$\Longrightarrow$ Can we make this bound tighter? $\longrightarrow \lfloor n/3 \rfloor$

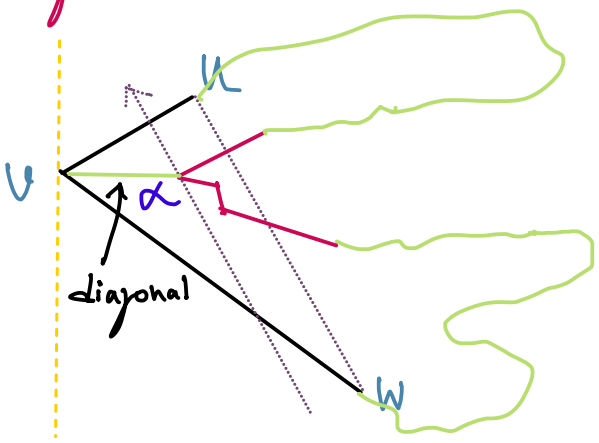
$\Rightarrow V(P)$ "properly" colourable using 3 colours [BFS on dual graph]

{RGB}

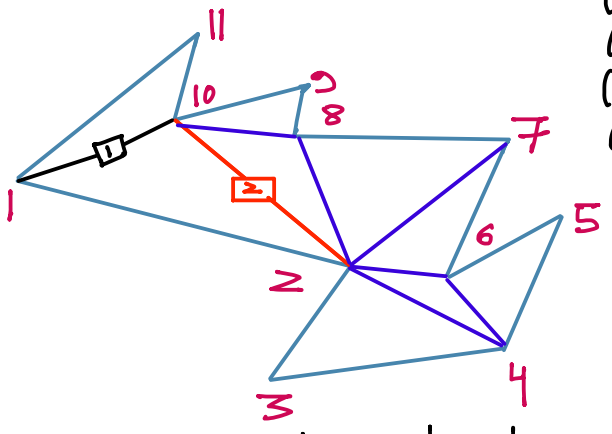
$u \quad v \qquad c(u) \neq c(v)$

edge or diagonal

Diagonal — How?



V = leftmost vector of P

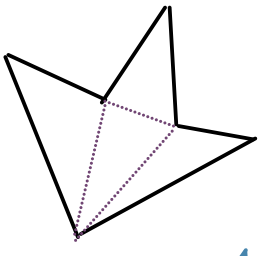


Diag

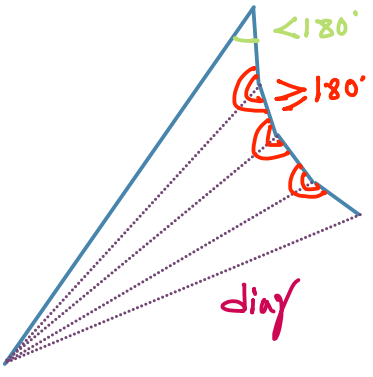
(1,10)
(2,10)
(10,8)
(2,8)
(2,4)
(2,7)
(2,6)
(6,4)

Ear decomposition

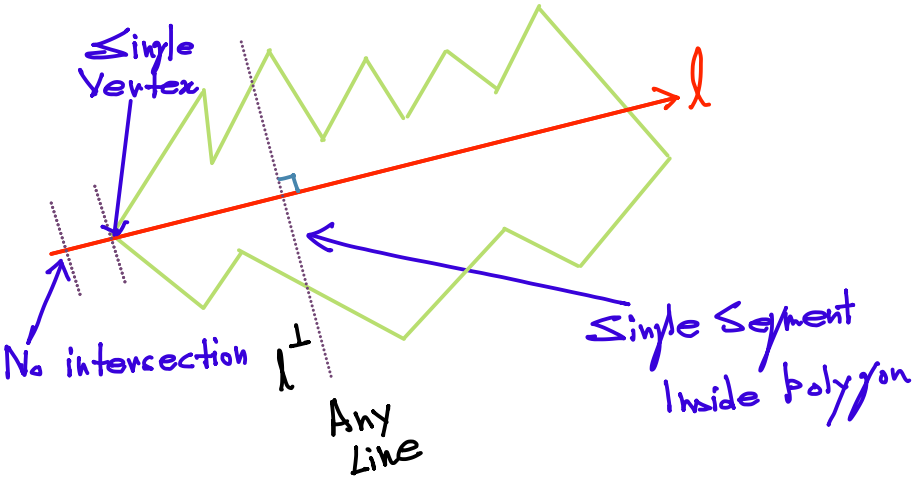
$T = O(n^2)$

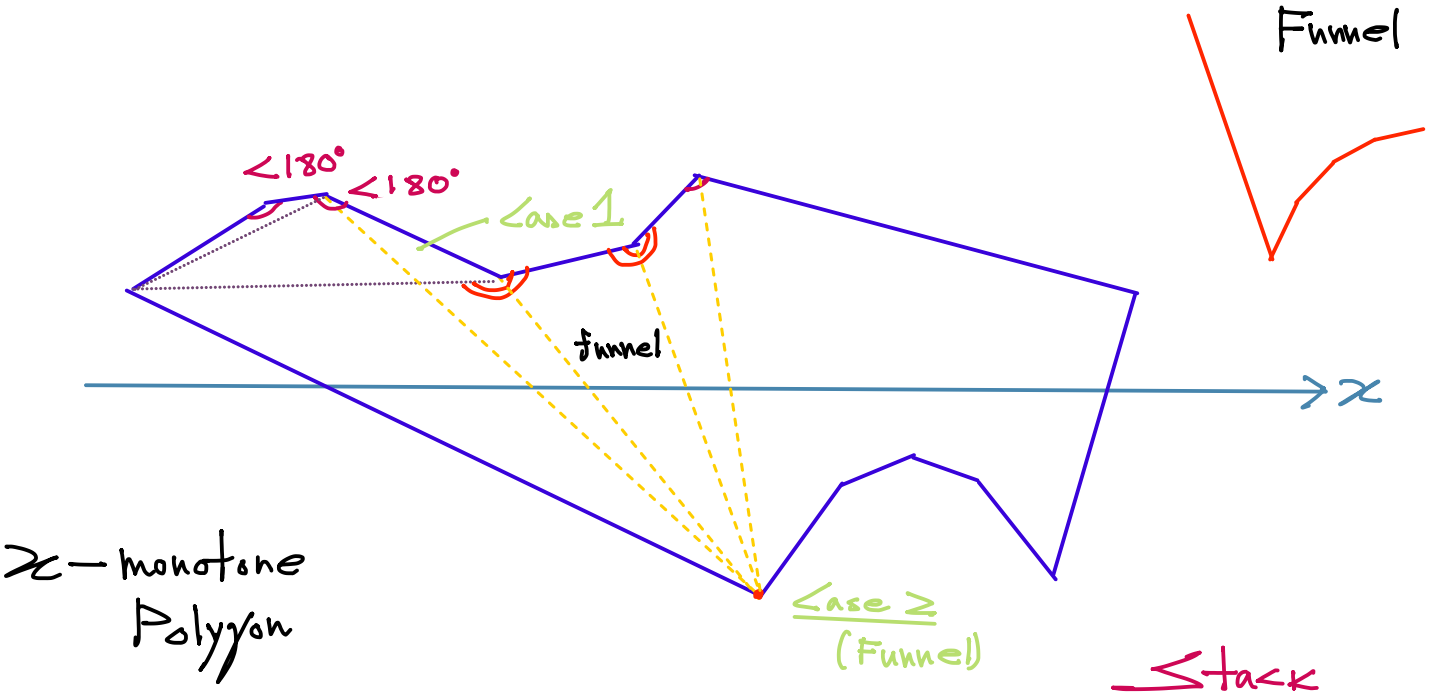


Each triangle can be discovered atmost 3 times and each diagonal will be discovered 1 time.



Monotone Polygon w.r.t l



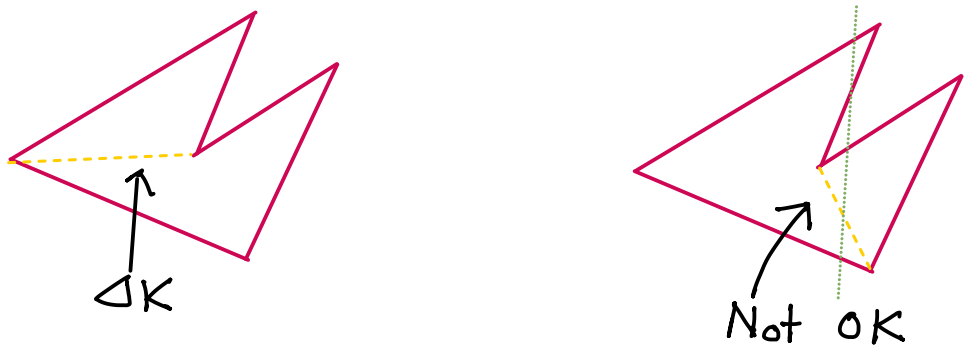


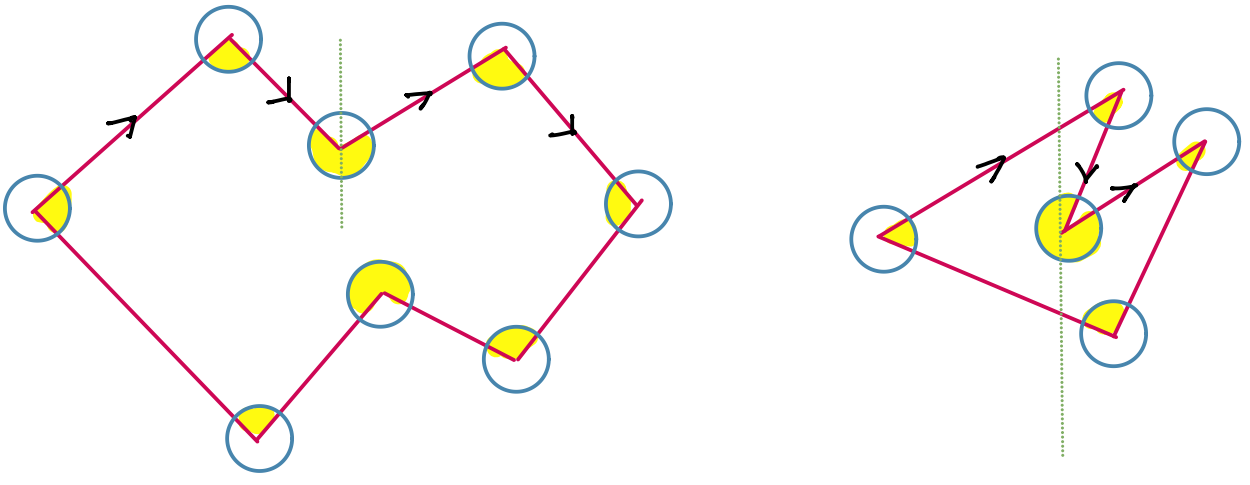
Stack

$O(n)$ time

Subdivision of a polygon into monotone polygons by adding non-intersecting diagonals

Let's consider x-monotone subdivision

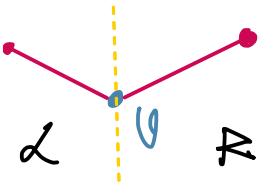


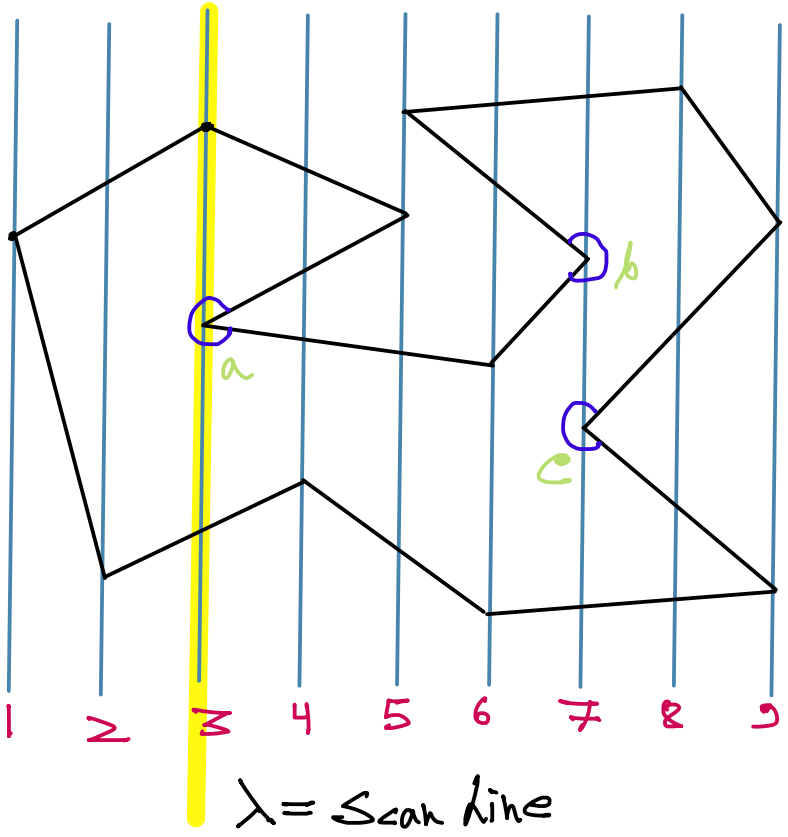


for every v, adj(v) lie on both L and R

$\Longleftrightarrow$ P is x-monotone

(or if both L or both R and internal angle at $v < 180°$)

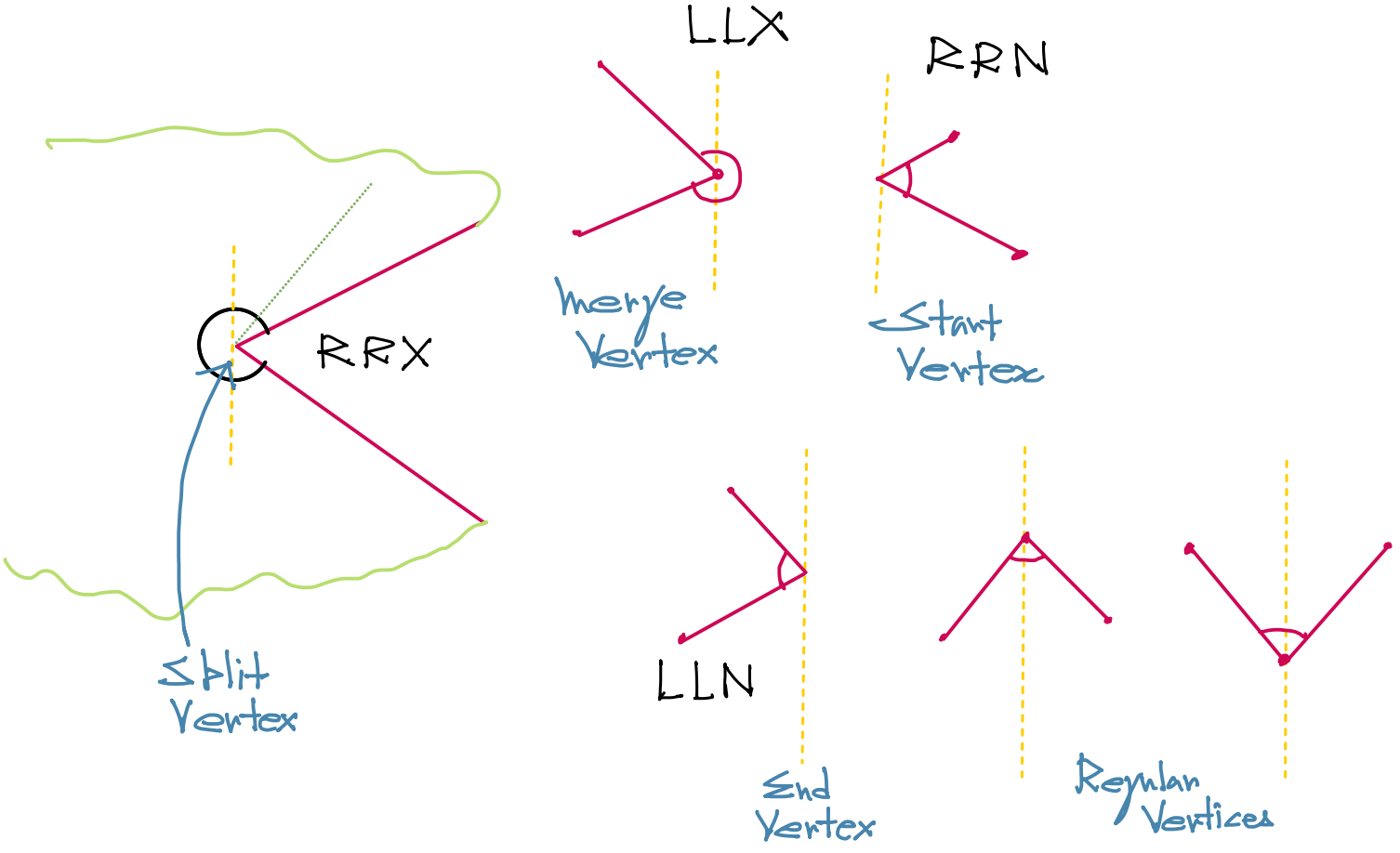


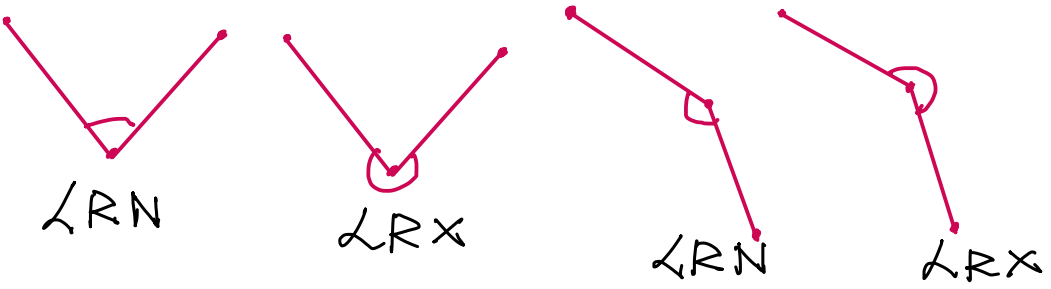


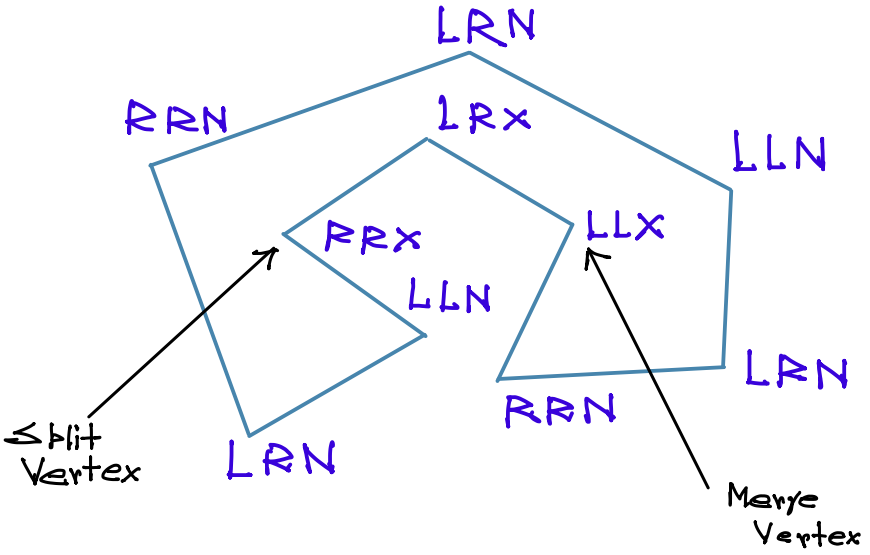
a, b, c do not satisfy the monotonicity property

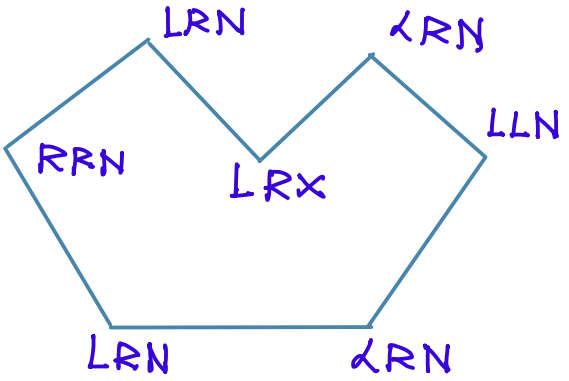
$\Rightarrow$ Diagonals endpoints include a, b, c.

$\Rightarrow$ For every diagonal, at least one endpoint = a/b/c and each of a, b, c should be the endpoint of one diag

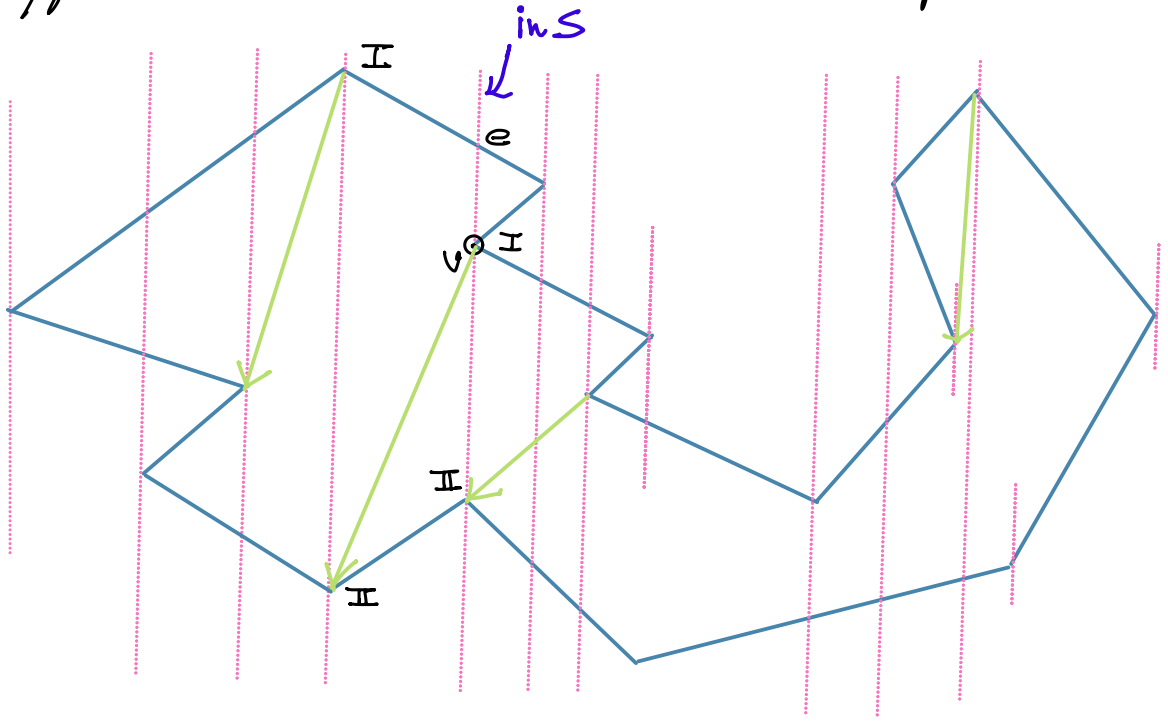








Polygon P is x-monotone $\Longleftrightarrow$ P has no merge or split vertex

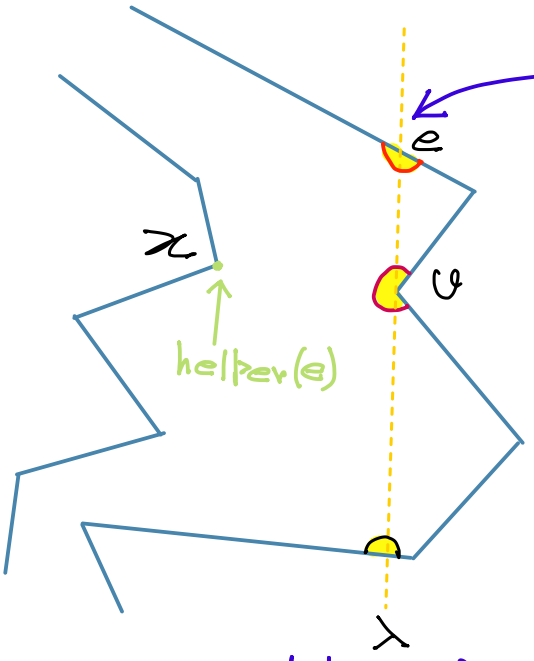


⊙ v = merge vertex

$\Longrightarrow$ diagonal vx : $x \in right(v)$ $\longrightarrow$ decision taken later

⊙ v = split vertex

$\Longrightarrow$ diagonal vx : $x \in left(v)$



found for v in $O(\log n)$ time from T_λ AVL(y)

Total time $= O(n \log n)$

type of v	type of helper(e)	diagonal?
split	any	Yes
merge	×	×
any	merge	Yes