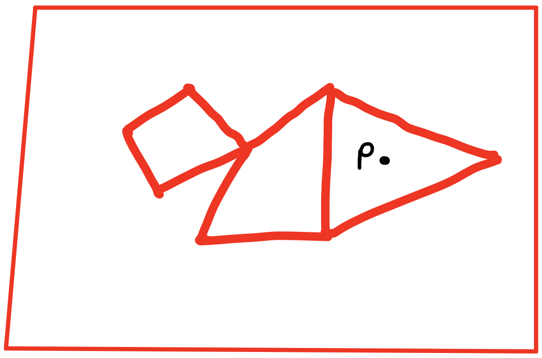
Lecture 9 - Point location

- Given planar subdivision (map) Find the face in which a given point lies.

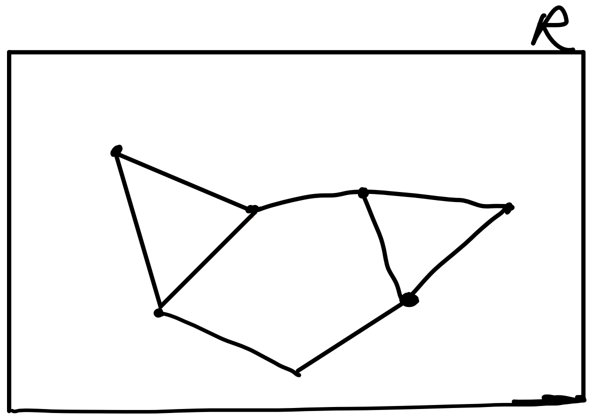


- Idea: construct refinement of the map which is easier to search (and not much larger).
- Faces will be:
  - Trapezoids (with 2 vertical sides)

    or
  - Triangles (with 1 vertical side)
- Called a trapezoidal map.

Assumptions :

- $S = \{ s_1, \ldots\ldots, s_n \}$ a set of segments
- Enclosed in a box R.

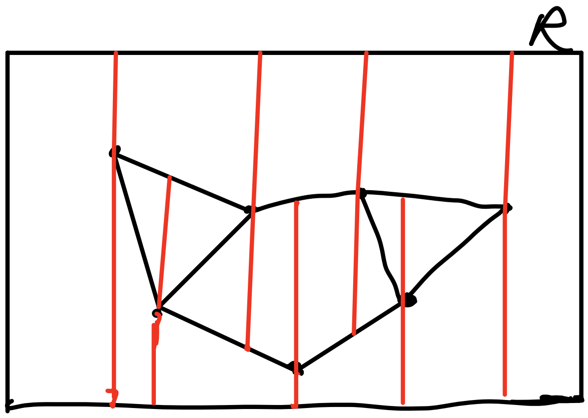


- Distinct endpoints have different x-coord
  (remove this assumption later)

- From map S, create trapezoidal map $T(S)$ by drawing a vertical line from each endpoint to nearest upper or lower segment / boundary of R.

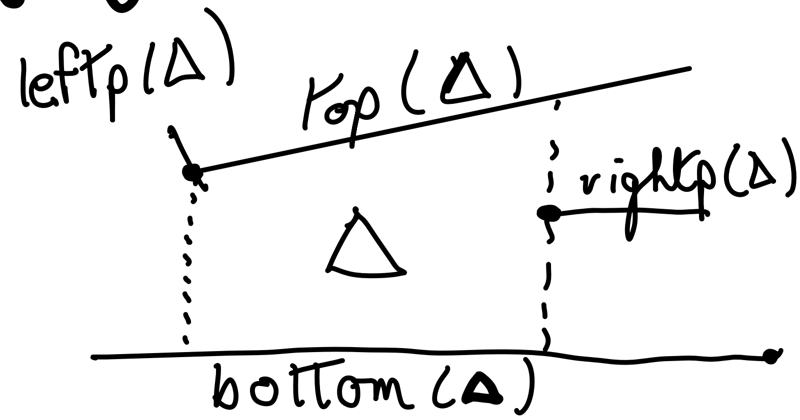
Assumptions :

- $S = \{ s_1, \ldots\ldots, s_n \}$ a set of segments
- Enclosed in a box R.



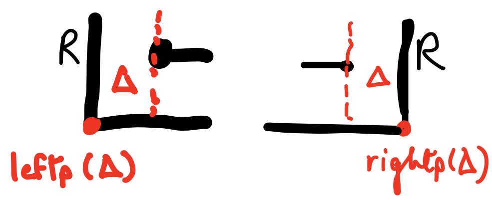
- Distinct endpoints have different x-coord
  (remove this assumption later)
- From map S, create trapezoidal map $T(S)$ by drawing a vertical line from each endpoint to nearest upper or lower segment / boundary of R.

Specifying a trapezoid



- For trapezoid Δ,
  - top(Δ) is segment of S / edge of R bounding Δ from above;
  - bottom(Δ) is segment of S / edge of R bounding Δ from below;
  - left & right sides determined by endpoints of segments / corners of R, leftp(Δ) & rightp(Δ).

Note: • unique trapezoid Δ whose left side is left boundary of R. Define leftp(Δ) = bottom left point of R.



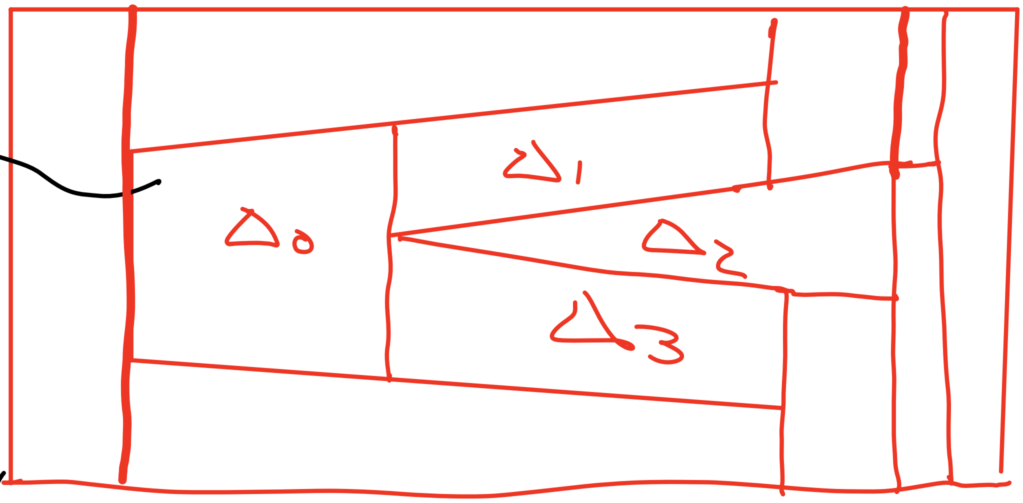
• likewise on right side.

Trapezoid Δ is specified by (top Δ, bottom Δ, leftp Δ, rightp Δ)

Neighbours of a trapezoid

Δ_0 is upper left n. of Δ_1.

Δ_1 is upper right neighbour of Δ_0.



Δ_3 is lower r. neighbour of Δ_0.

- Two trapezoids are adjacent if they share a vertical line (not a point).

  (Eg. Δ_0 & Δ_1 ✓ Δ_0 & Δ_3 ✓ not Δ_0 & Δ_2 ✗)

- If 2 adjacent trapezoids have common bottom, one is lower left neighbour & one is lower right neighbour.

- Sim., upper left / upper right neighbours.

Storing the trapezoidal map

T(S) stores :

- segments of S & edges of R
- endpoints of S & corners of R
- set of trapezoids Δ,
  which have pointers
  top Δ, bottom Δ, leftp Δ, rightp Δ

This enables us to reconstruct planar subdivision.

- Also, it contains pointers to the at most 4 neighbours of a trapezoid Δ,
  upper/lower, left/right

Theorem) Trapezoidal map for n segments has at most 6n+4 vertices & 3n+1 Trapezoids.

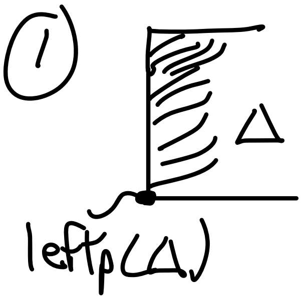
Proof

No. of vertices :- 4 corners of R.

- At most 2n endpoints, For each endpoint create 2 new endpoints with vertical lines.

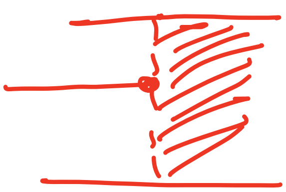
No. of vertices $\leq 4 + 2n + 2(2n) = 6n+4.$

- No of Trapezoids : count by no. with a given leftpoint.

(1) 

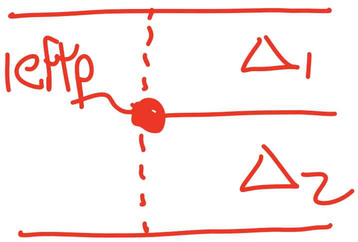
$\sim$ 1 Trapezoid with leftp(Δ) = bottom corner.

(2) leftp(Δ) is right endpoint of segment



each such right endpoint specifies exactly 1 Trapezoid.

③ $\text{leftp}(\Delta)$ is left endpoint of segment

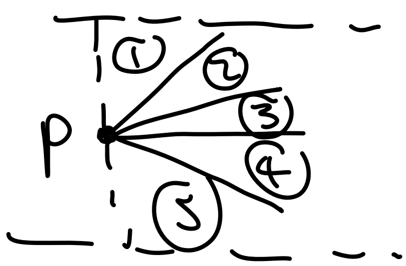


$\sim$ determines two trapezoids.

K_i = no. of leftpoints which are common left endpoint of exactly i segments.

Then $n = K_1 + 2K_2 + 3K_3 + \ldots$

Example:



4 segments with common left endpoint p, these determine 5 trapezoids with leftpoint p

• More gen. K segments $\to$ $K+1$ trapezoids

No. of trapezoids of type ③

$$= 2K_1 + 3K_2 + 4K_3 + \ldots$$

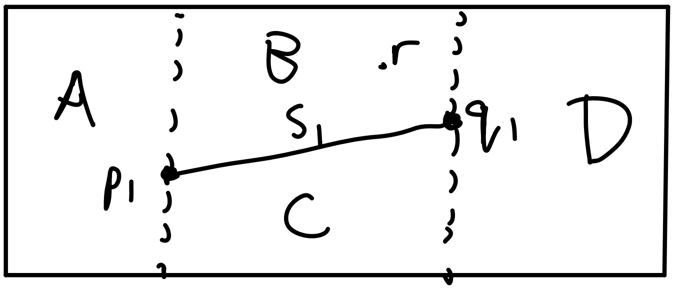
$$\leq 2(K_1 + 2K_2 + 3K_3 + \ldots) = 2n$$

• So total $\leq \underset{\text{case 1}}{1} + \underset{\text{case 2}}{n} + \underset{\text{case 3}}{2n} = 3n+1$ $\square$

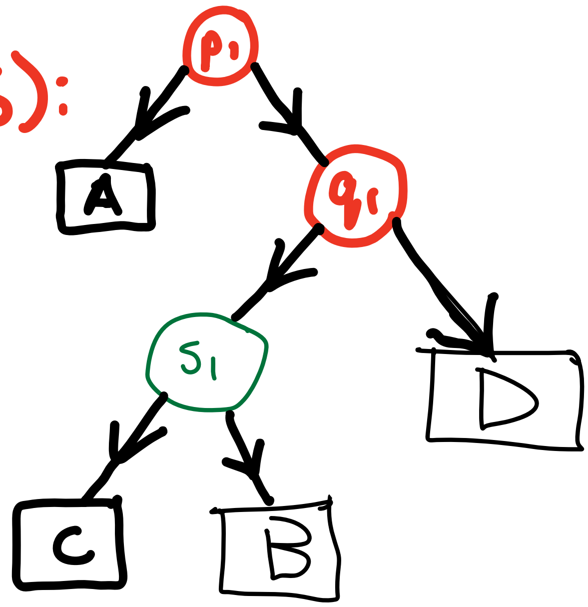
Search structure

- Oriented graph $D(S)$ associated to $T(S)$.

T(S) :



D(S):



- Leaves are trapezoids.
- Inner nodes are segments / endpoints of segments
- Two edges from each inner node.

Given r, find trapezoid in which it lies by:

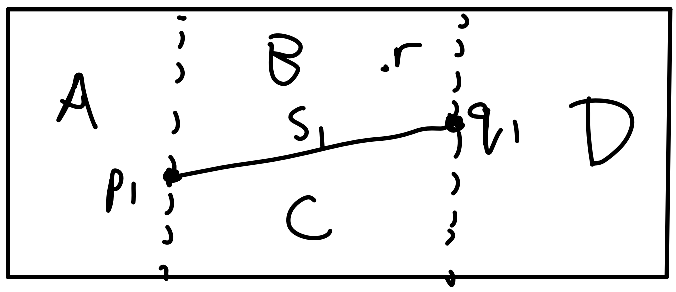
- @ endpoint, go left/right if r lies to left/right.
- @ segment, go left/right if r lies below / above.

(Assume r has diff x-coord to all endpoints & does not lie on segment — remove these assumptions later)

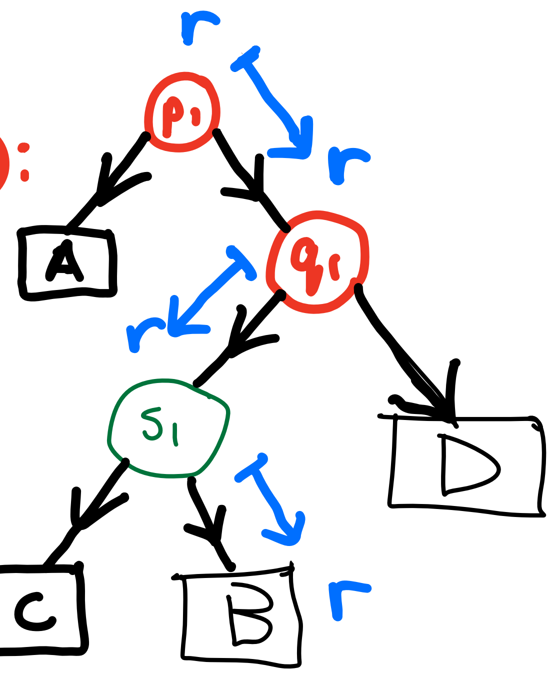
Search structure

- Oriented graph D(S) associated to T(S).

T(S):



D(S):



- Leaves are trapezoids.
- Inner nodes are segments/endpoints of segments
- Two edges from each inner node.

Given r, find trapezoid in which it lies by:

- @ endpoint, go left/right if r lies to left/right.
- @ segment, go left/right if r lies below/above.

(Assume r has diff x-coord to all endpoints & does not lie on segment — remove these assumptions later)

Randomised incremental algorithm

- Input $S = \{s_1, \ldots, s_n\}$ set of segments.
- Randomise order.
- For $i = 1, \ldots, n$:
  given trapezoidal map T_{i-1} & search str. D_{i-1}
  construct T_i & D_i.

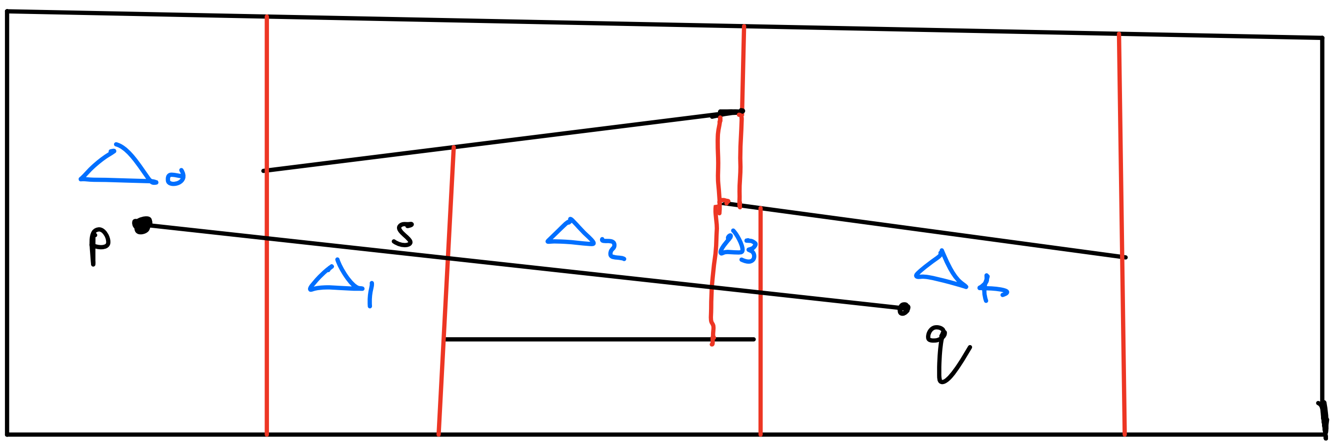
Steps

1. Find set $\Delta_0, \ldots, \Delta_k$ of trapezoids in T_{i-1} properly intersected by s_i.

2. Remove $\Delta_0, \ldots, \Delta_k$ from T_{i-1} & replace by new trapezoids appearing because of s_i.

3. Remove leaves $\Delta_0, \ldots, \Delta_k$ from D_{i-1} & replace these by subgraphs to create D_i.

① Following segment algorithm



Firstly find trapezoid Δ_0 containing left endpoint p of new segment s:

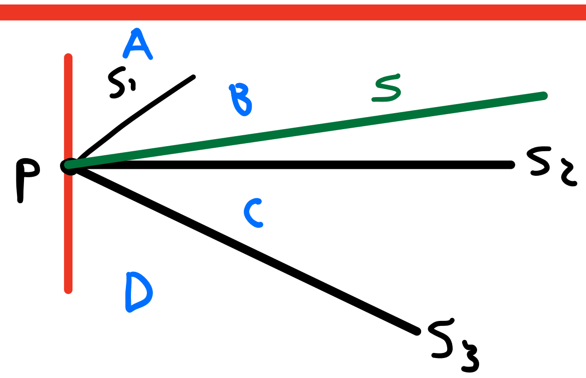
several cases:

- If p not the endpoint of any of $\{S_1, \dots, S_{i-1}\}$

then it lies in interior of Δ_0.

In this case, find Δ_0 by searching for p in $D(S_{i-1})$.

- If p is endpoint of $\{s_1, \dots, s_{i-1}\}$, it is slightly delicate:

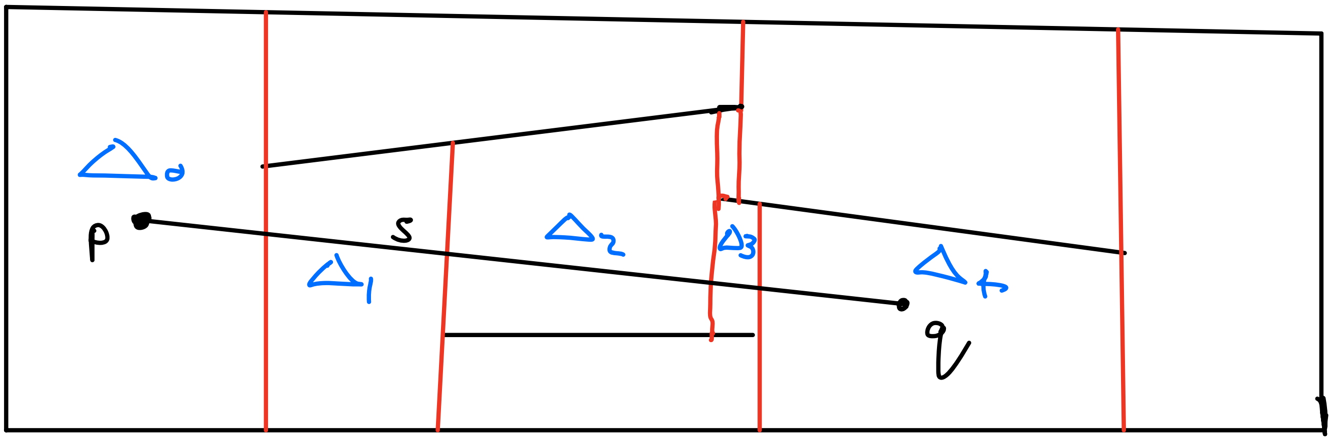


compare slope of those s_i with slope of s to find trapezoid through which s runs ~ this will be Δ_0

can still use search structure as follows:

- at a node for an endpoint, go right if x-coord of $p \geq$ x-coord of endpoint.
- at a node for a segment s_i, p lies on s_i (only happens if p is left endpoint of s_i)

compare slope of s & s_i: if slope of s is larger, decide s is larger & go left, else right.



Having found Δ_0:

- If right endpoint q of s belongs to Δ_0, then s lies completely within Δ_0 & we stop.

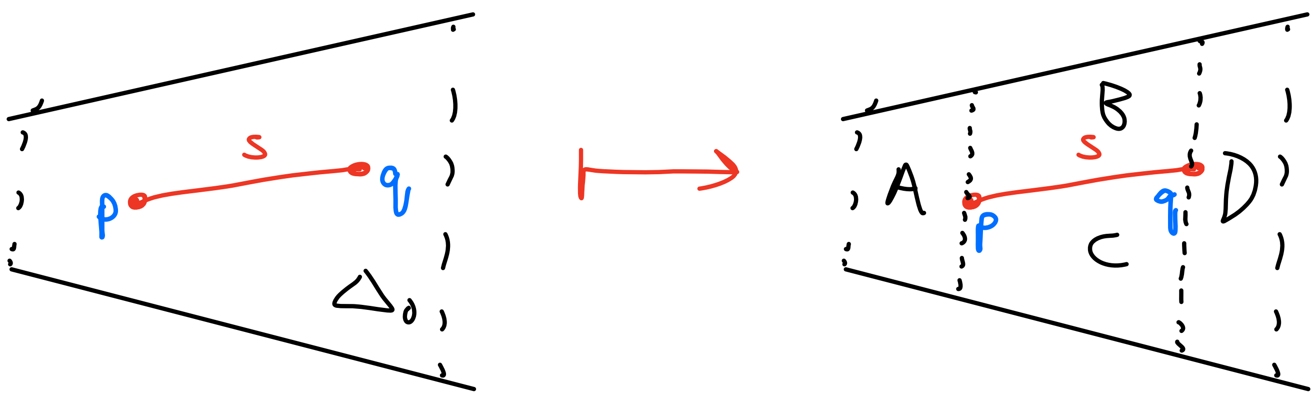
- Otherwise s intersects upper or lower right neighbour of Δ_0
  (if $right_p(\Delta_0)$ above s, s intersects lower right neighbour, else upper right neighbour.)

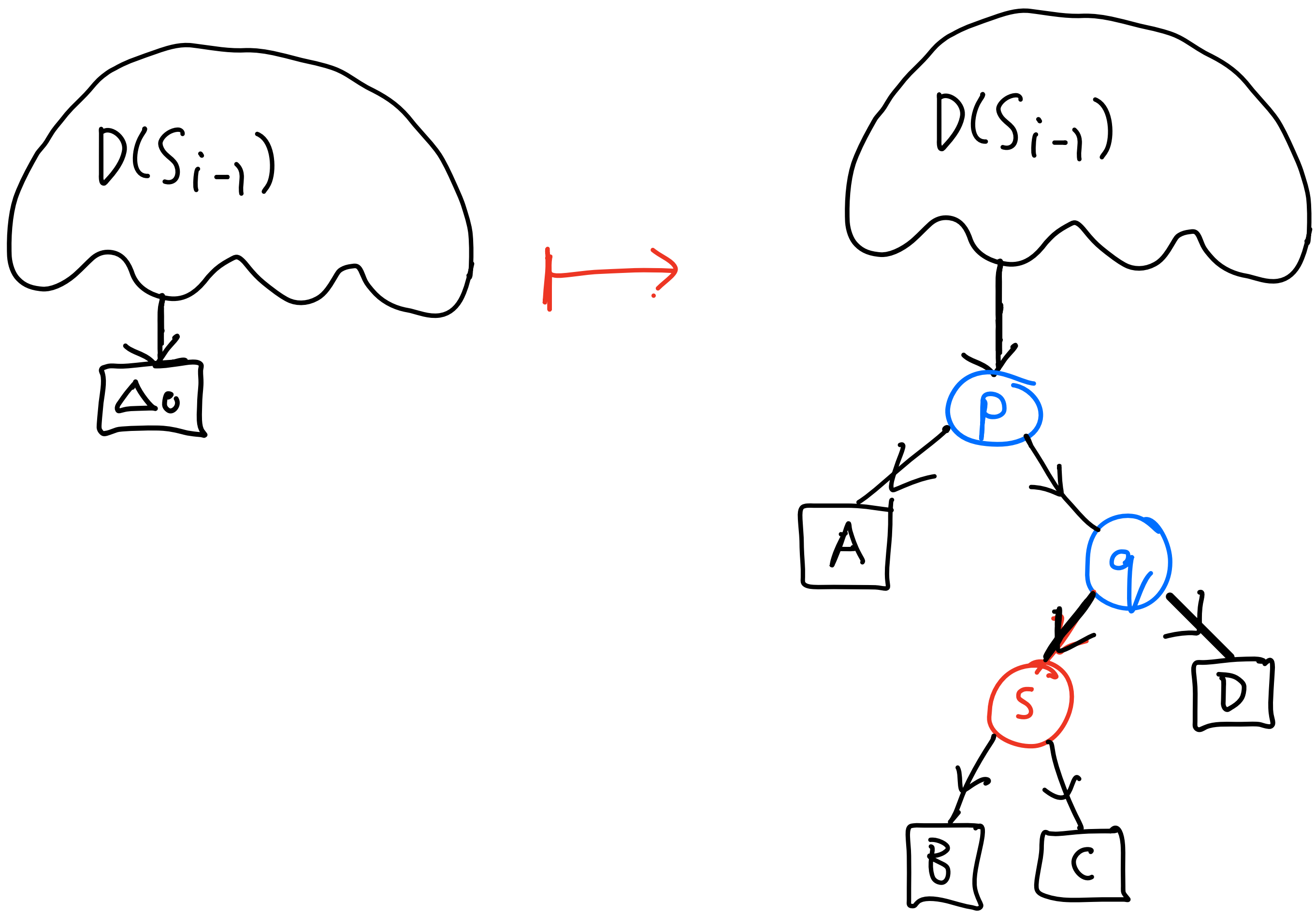
- In this way, we find Δ_1 & continuing in the same way, find $\Delta_0, \Delta_1, \ldots, \Delta_k$.

See E-Learning for pseudocode.

Steps 2 & 3 - brief geometric outline

- If s is contained in Δ_0:

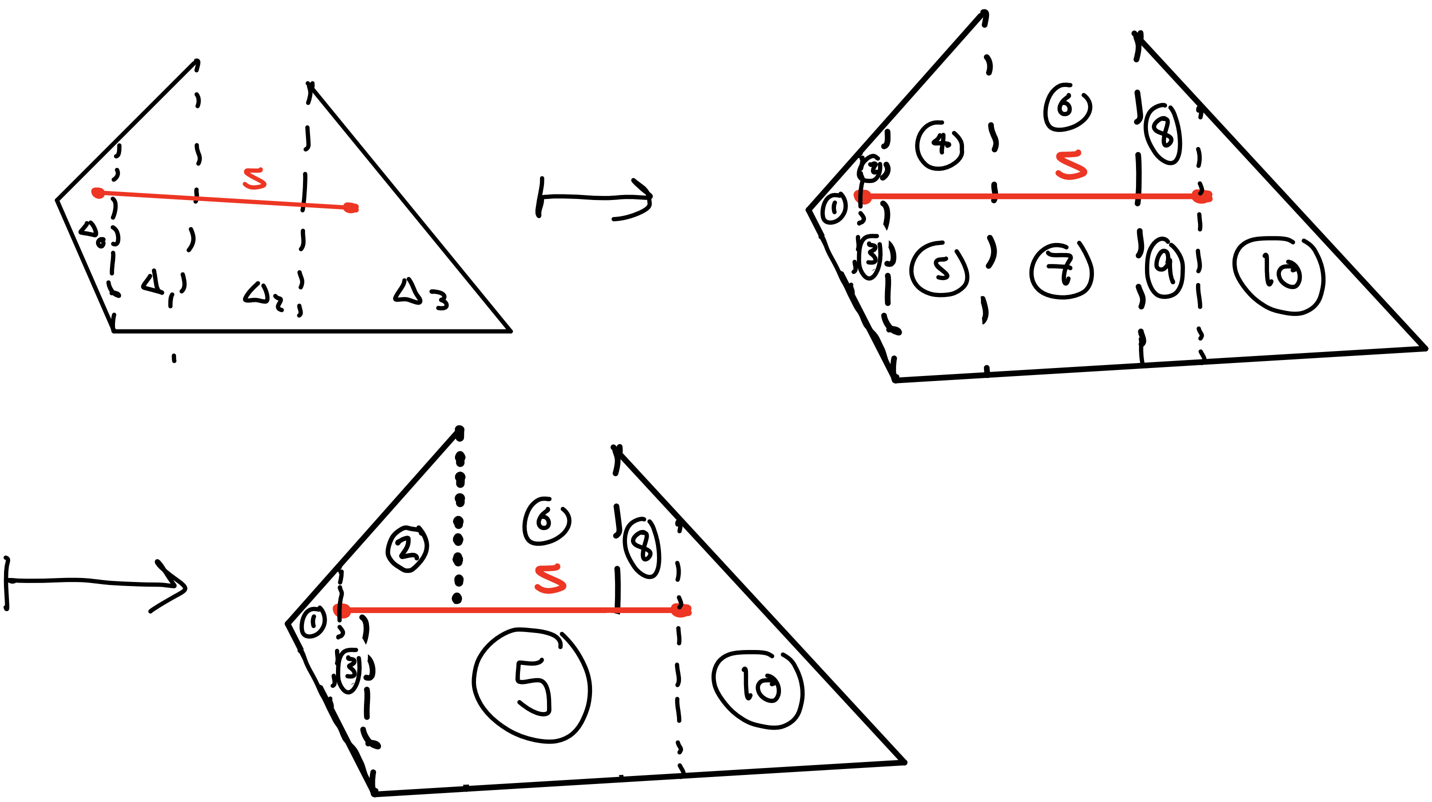




- Otherwise, draw vertical lines from endpoints p, q of s to nearest segments (only if endpoints weren't already present).

- New vertical lines together with s naturally split $\Delta_0, \dots, \Delta_k$ into smaller trapezoids.

- Merge adjacent trapezoids with same top & bottom.



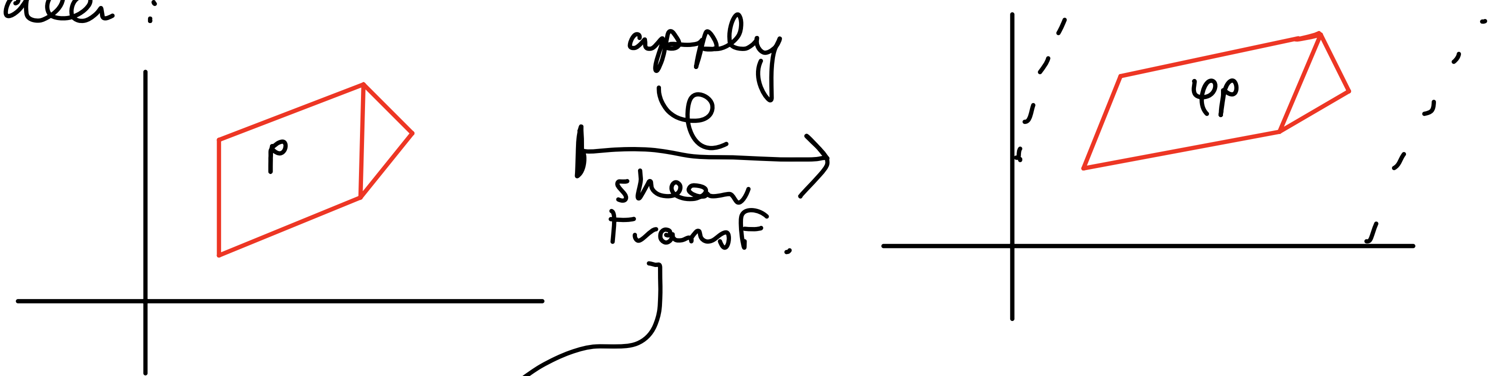
See Fig. 8.15 & 8.16 in E-Learning for update of search structure.

Remove restrictive assumptions

- We assumed :
  - distinct endpoints have different x-coord
  - searchpoint p has different x-coord to all endpoints of segments &

  (*) p does not lie on a segment (this is not really problematic - just find trapezoid containing s)

Idea :



$\varphi(x,y) = (x + \varepsilon y, y)$ for very small ε

- Applying φ to S & p gets rid of these bad cases :

  so searching for φp in φS will produce the desired trapezoid.

- So just run same alg. for φS & φp.

- In Fact, don't need shear transformation:

  ℓ_p lies to left of ℓ_q $\Leftrightarrow$

  $p_x < q_x$ or ($p_x = q_x$ & $p_y < q_y$)

  $\Leftrightarrow$ $p < q$ in lex order

- So just modify original algorithm by using lex order instead of checking if points lie to left or right.

- Gives same result but gets rid of restrictive assumptions.

Complexity

- Exp. search time - $O(\log n)$
- Exp. time to constr search str - $O(n \log n)$
- Exp. size of search str - $O(n)$