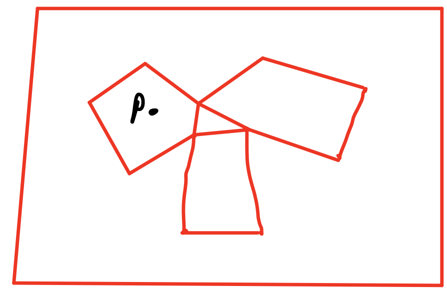
Lecture 9 - Point location

Given planar subdivision (map) give an algorithm that finds in which face a given point lies.

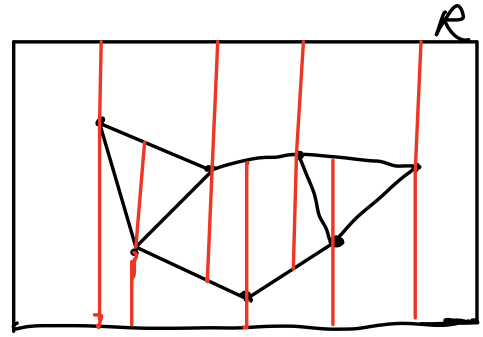


Idea: construct refinement of the map which is easier to search (& not much larger).

- the faces will be trapezoids (with two vertical sides) or triangles (degenerate trapezoids).

This will be called a trapezoidal map.

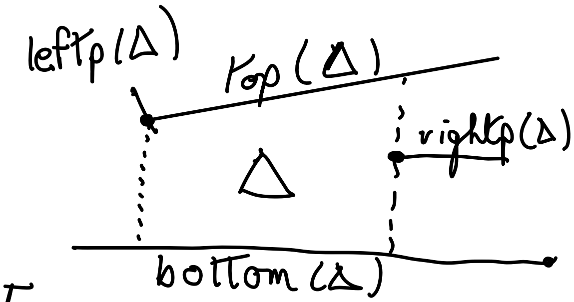
– Trapeze artist!



Assumptions:

- set $S = \{s_1, \ldots, s_n\}$ of segments, which do not intersect except at endpoints.
- enclosed in a box R.
- Distinct endpoints do not have same x-coord (remove this assumption later.)
- From map S create Trapezoidal map $T(S)$ by drawing a vertical line from each endpoint of a segment to nearest upper & lower segments (or to the boundary of R).

- For a trapezoid Δ, top(Δ) segment of S or edge of R bounding Δ from above.



- Similarl bottom(Δ) is segment of S or edge of R bounding Δ from below.
- Left and right sides determined by endpoints of segments or corners of R, leftp(Δ) & rightp(Δ).
- For unique trapezoid whose left side is left boundary of R,

define leftp to be bottom left corner of R.
Sim. for unique trapezoid on right.
- In case of triangle, left & right sides degenerate to a point.

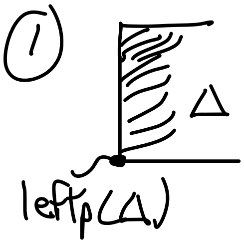
Trapezoid Δ is specified by (topΔ, bottomΔ, leftpΔ, rightpΔ).

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

$\boxed{\text{Proof}}$ No. of vertices : 4 corners of R.
- At most $2n$ endpoints, For each endpoint create 2 new endpoints;

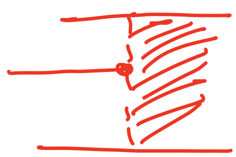
Total no. of vertices $\leq 4 + 2n + 2(2n) = 6n+4$.

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

(1) 

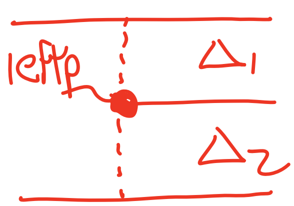
~ 1 trapezoid with leftp(Δ) = bottom corner.

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



each such right endpoint specifies exactly 1 trapezoid.

(3) leftp(Δ) is left endpoint of segment

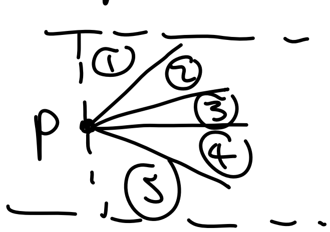


~ determines two trapezoids.

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

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

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

$$= 2k_1 + 3k_2 + 4k_3 + \dots$$

$$\leq 2(\underbrace{k_1 + k_2 + k_3 + \dots}_{= 2n})$$

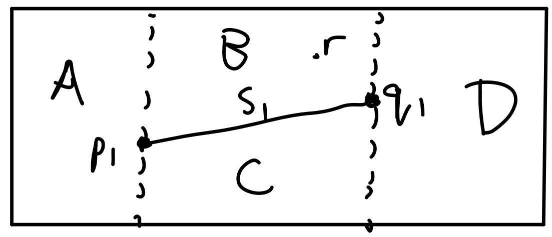
- So total number of trapezoids $\leq$

$$\underset{\text{case 1}}{1} + \underset{\text{case 2}}{n} + \underset{\text{case 3}}{2n} = 3n+1.$$

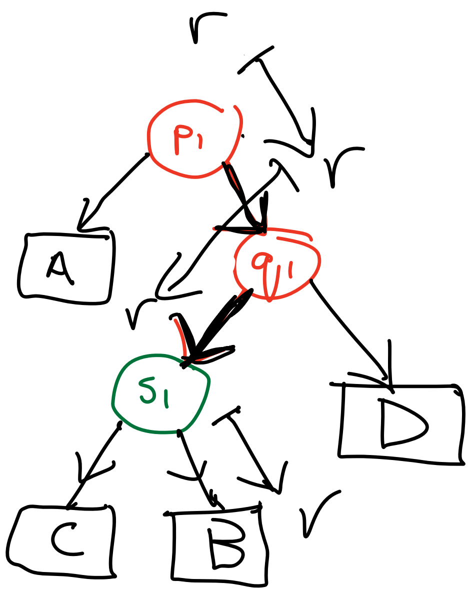
$\square$

Search structure

Oriented graph $D(S)$ associated to $T(S)$:



- leaves are trapezoids of $T(S)$.
- Inner nodes are endpoints of segments & segments: two edges from each inner node.



- Given r, find trapezoid in which it lies by:
  - if a node is an endpoint, go left if r lies to its left & right if r lies to its right.
  - if a node is segment, go left if r lies below & right if r lies above.

Assume: r has distinct x-coord to endpoints & also does not lie on a segment.

Will remove the assumptions.

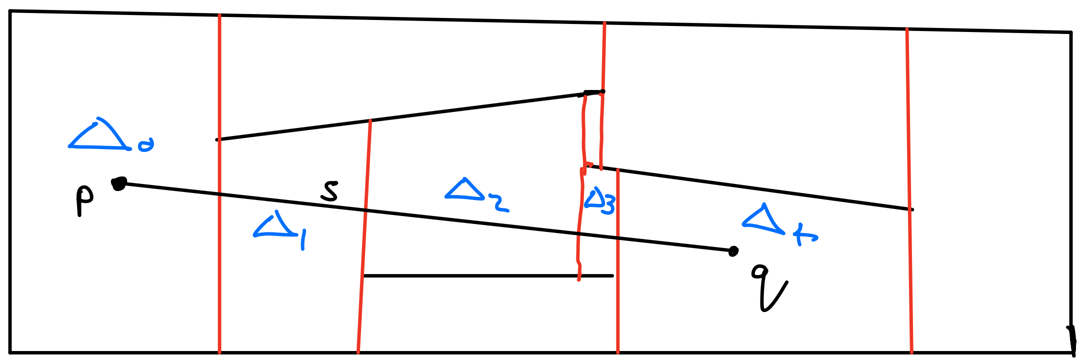
Randomised incremental algorithm

- Input $S = \{s_1, \dots, s_n\}$ set of segments.
- Randomise order.
- For $i = 1, \dots, n$ construct trapezoidal map T_i and search structure D_i from T_{i-1} & D_{i-1} by adding segment s_i.

Steps

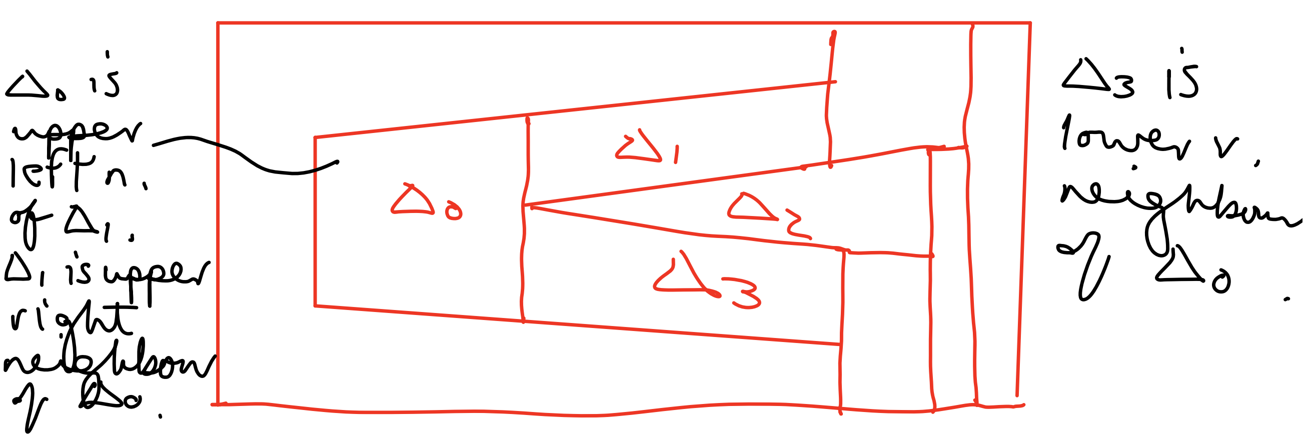
1. Find set $\Delta_0, \dots, \Delta_k$ of trapezoids in T_{i-1} properly intersected by s_i.
2. Remove $\Delta_0, \dots, \Delta_k$ from T_{i-1} & replace by new trapezoids appearing because of s_i.
3. Remove leaves $\Delta_0, \dots, \Delta_k$ from D_{i-1} & replace these by subgraphs to create D_i.

① Following segment algorithm

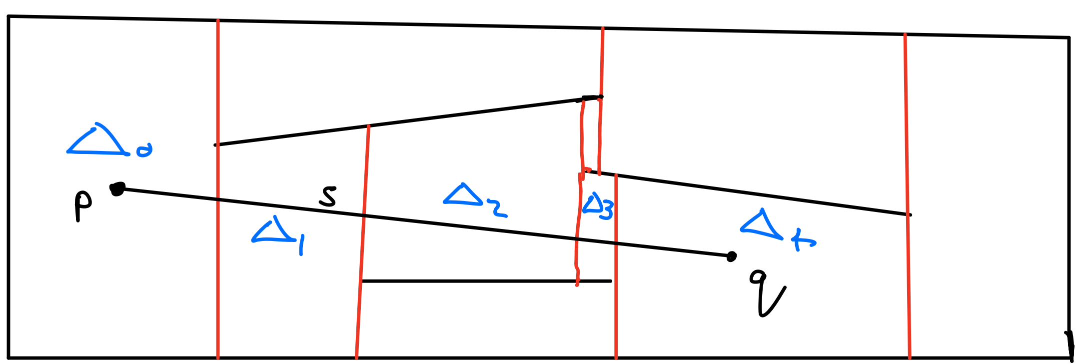


Firstly, find trapezoid Δ_0 containing left endpoint p of new segment s: several cases

- If p not an endpoint of $\{s_1, \ldots, s_{i-1}\}$ 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 an endpoint, see E-Learning: Figure 8.10.
- Having found Δ_0, need to find $\Delta_1, \ldots \Delta_k$. For this, need notion of neighbour of trapezoid.



- Two trapezoids are adjacent if they share a vertical line, not merely a point.
- Δ_0, Δ_1 are adj, Δ_0, Δ_3 are adj., Δ_0 & Δ_0 are not adj.
- If 2 adjacent trapezoids have common bottom, say one is lower left neighbour & one is lower right neighbour.
- If 2 adjacent trapezoids have common top, say one is upper left neighbour & one is upper right neighbour.
- Neighbours should be stored with trapezoidal map.



- Back to algorithm:
  - If right endpoint q of s belongs to Δ_0, then s lies completely within Δ_0 & we stop.
  - Otherwise, s intersects upper neighbour of Δ_0

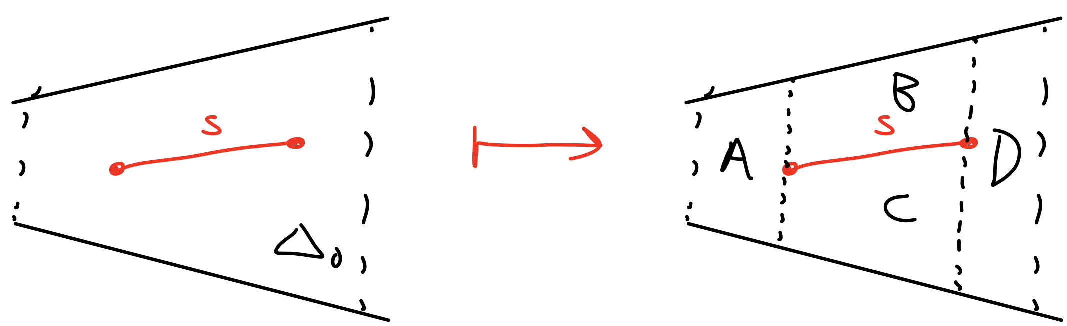
or lower right neighbour of Δ_0 -
- if rightp(Δ_0) is above s, then s intersects lower right neighbour
- otherwise s intersects upper right neighbour.

Continue in this way to find $\Delta_0, \Delta_1, \ldots, \Delta_k$.

- See E-learning for pseudocode for algorithm.

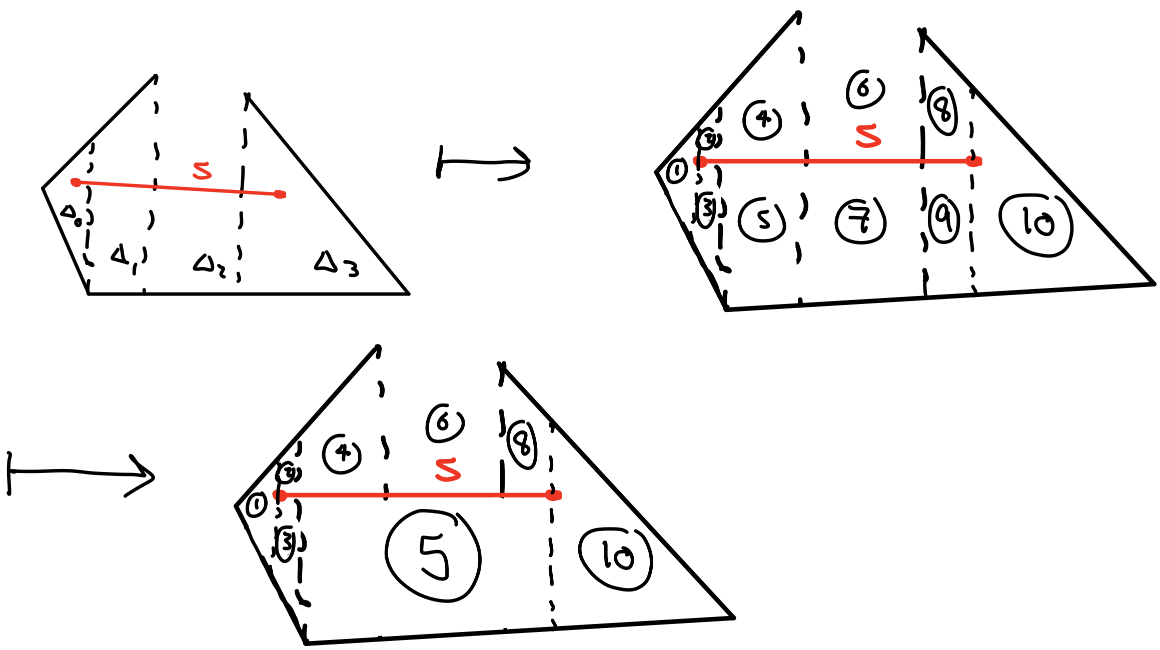
Steps 2 & 3 - brief geometric outline

- If s is contained in Δ_0:



See E-Learning Fig 8.13 & 8.14 for update of search structure.

- 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 $\Delta_0, \ldots, \Delta_k$ into smaller trapezoids.
- Merge adjacent trapezoids with same top & bottom.

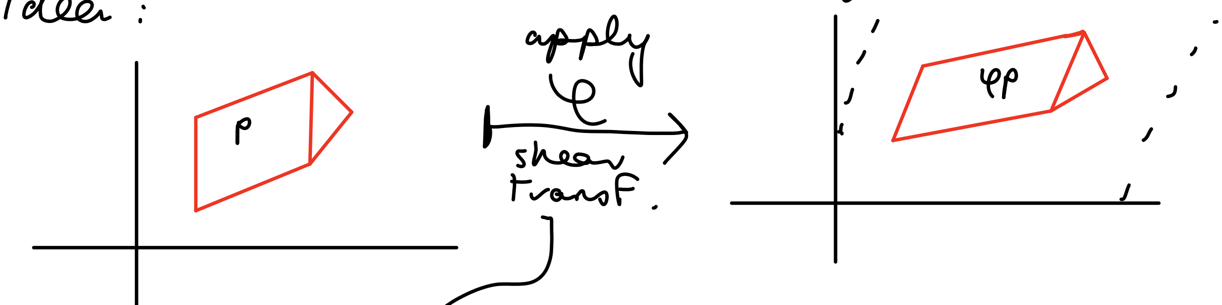


See Fig. 8.15 & 8.16 in E-Learning for update of search structure.

Remove restrictive assumptions

- We assumed:
  - no endpoints of segments have same 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, no 2 segments in φS have same x-coord as endpoints. Sim φp has diff x-coord to endpoints.
- Therefore searching for φp in φS will produce the desired trapezoid containing φp.
- So run same alg. on φS & φp.

- In fact, do not need shear transformation at all.

For ℓp lies to left of ℓq $\iff$ $p_x < q_x$ or ($p_x = q_x$ & $p_y < q_y$) $\iff$ $p < q$ in lexicog. ordering.

- So it suffices to modify algorithm by using lex. order instead of checking if points lie to left or right.
- This gives same result but removes restrictive assumption.

Complexity

Search time to locate a point - $O(\log n)$

- Search structure is created in n steps.
- Consider point r.
- X_i : no. of nodes added to search point for r at i'th stage.
- From E-Learning 8.14, 8.16, we see $X_i \leq 3$.
- Length of search path for r = $\sum_{i=1}^{n} X_i$.

Expected search time :

$\sum_{i=1}^{n} E(X_i)$ where $E(X_i)$ = expected no. of nodes added @ i'th stage to search path for r.

$$E(X_i) = 0.p(X_i=0) + 1.p(X_i=1) + 2.p(X_i=2) + 3.p(X_i=3)$$
$$\leq 3.p(X_i \neq 0).$$

- let $\Delta_i \in T_i$ be trapezoid containing r in i'th trapezoidal map.
- $p(X_i \neq 0) \Leftrightarrow p(\Delta_i \neq \Delta_{i-1})$.

Now $\Delta_i \neq \Delta_{i-1} \Leftrightarrow$ top $\Delta_i \neq$ top Δ_{i-1} or bottom $\Delta_i \neq$ bottom Δ_{i-1} or leftp $\Delta_i \neq$ leftp Δ_{i-1} or rightp $\Delta_i \neq$ rightp Δ_{i-1}.

Therefore

$$p(\Delta_i \neq \Delta_{i-1}) \leq p(\text{top}\,\Delta_i \neq \text{top}\,\Delta_{i-1}) + p(\text{bottom}\,\Delta_i \neq \text{bottom}\,\Delta_{i-1}) +$$

$p(\text{leftp } \Delta_i \neq \text{leftp } \Delta_{i-1}) +$
$p(\text{rightp } \Delta_i \neq \text{rightp } \Delta_{i-1})$.

- $p(\text{top } \Delta_i \neq \text{top } \Delta_{i-1}) = p(\text{top } \Delta_i = s_i)$

$1 = p(\text{top } \Delta_i = s_1) + \dots + p(\text{top } \Delta_i = s_i) + p(\text{top } \Delta_i = \text{top} R)$

$$\Rightarrow \quad \sum_{j=1}^{i} p(\text{top } \Delta_i = s_j) \leq 1$$

& these on left all equal as $s_1, \dots, s_i$ are in random order

$$\Rightarrow \quad p(\text{top } \Delta_i = s_i) \leq 1/i .$$

- Sim for bottoms.
- $p(\text{leftp } \Delta_i \neq \text{leftp } \Delta_{i-1})$:

$\text{leftp } \Delta_i \neq \text{leftp } \Delta_{i-1} \iff$

$\{$ it is endpoint of s_i & none of $s_1, \dots, s_{i-1}$ or corner(R). $\}$

$p(\text{leftp } \Delta_i$ is endpoint of s_1 & none of others$)$
$+ p(\text{leftp } \Delta_i$ is endpoint of s_2 & none of others$)$
$+$
$\vdots$
$p(\text{leftp } \Delta_i$ is endpoint of s_i & none of others$)$
≤ 1

$\Rightarrow \quad p(\text{leftp } \Delta_i$ is endpoint of s_i & none of others$) \leq 1/i$

Hence $p(\text{leftp}\,\Delta_i \neq \text{leftp}\,\Delta_{i-1}) \leq 1/i$ &

$p(X_i \neq 0) \leq 4/i$.

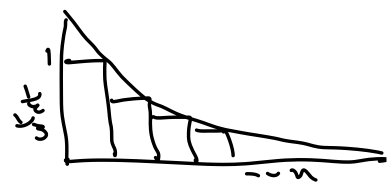
- So $\displaystyle\sum_{i=1}^{n} E(X_i) \leq 3 \sum_{i=1}^{n} p(X_i \neq 0)$

$$\leq 3 \sum_{i=1}^{n} 4/i$$

$$\leq 12\left(1 + \sum_{i=2}^{n} 1/i\right)$$

$$\leq 12\left(1 + \int_1^n 1/x \, dx\right)$$

$$= 12(1 + \log n) = O(\log n)$$



as claimed.

Expected size of search structure O(n)

Proof: size = no. of leaves + no. of internal nodes

$$\leq 3n+1 + \sum_{i=1}^{n} (\text{no of nodes arising at } i\text{'th step})$$

last week

$$= 3n+1 + \sum_{i=1}^{n} u_i - 1$$ where

u_i = no of new trapezoids at i'th step.

Eg. see examples in E-Learning (not proven)

So randomized complexity $\leq O(n) + \sum_{i=1}^{n} E(u_i)$.

We must show that $E(u_i) = O(1)$.

- For $\Delta \in T(S_i)$ & a segment $s \in S_i$,

let $\lambda(\Delta, s) = \begin{cases} 1 & \text{if } \Delta \text{ disappears from } T(S_i) \text{ when we remove } s \text{ from } S_i \\ 0 & \text{otherwise.} \end{cases}$

- Since Δ determined by at most 4 segments

$$\sum_{s \in S_i} \lambda(\Delta, s) \leq 4.$$

- Now $u_i = \sum_{\Delta \in T(S_i)} \lambda(\Delta, s_i)$

Also $\sum_{\Delta \in T(S_i)} \sum_{s \in S_i} \lambda(\Delta, s) \leq \sum_{\Delta \in T(S_i)} 4 \leq 4 \underbrace{|T(S_i)|}_{\text{no. of Traps in } T(S_i)}$

$|| \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \geq 4(3i+1) = O(i)$

$\sum_{s \in S_i} \sum_{\Delta \in T(S_i)} \lambda(\Delta, s)$

=

no. of trapezoids disappearing when remove s_1

+

$u_i \sim$ no. of trapezoid disapp. - - - remove s_i

so since $s_1, \ldots, s_n$ are in random order

$$E(u_i) \leq \frac{1}{i} \left(\sum_{s} \sum_{\Delta} \lambda(\Delta, s) \right)$$

$$\leq \frac{1}{i} O(i) = O(1).$$

Algorithm constructs search str. in expected time $O(n \log n)$

Time for creating $T(S_i)$ & $D(S_i)$ from $T(S_{i-1})$ & $D(S_{i-1})$

is

$E(u_i)$ + time searching in $D(S_{i-1})$ for trap. containing left endpoint of s_i

exp. no. of trapezoids

So Time ≤

$$\sum_{i=1}^{n} \left(E(u_i) + O(\log i) \right)$$

$$\le n\,O(1) + n \log n$$

$$= O(n \log n).$$