Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Chair for
INFORMATICS I
Efficient Algorithms and
Knowledge-Based Systems

Institute for Informatics

Computational Geometry

Winter term 2014/15

Point Localization

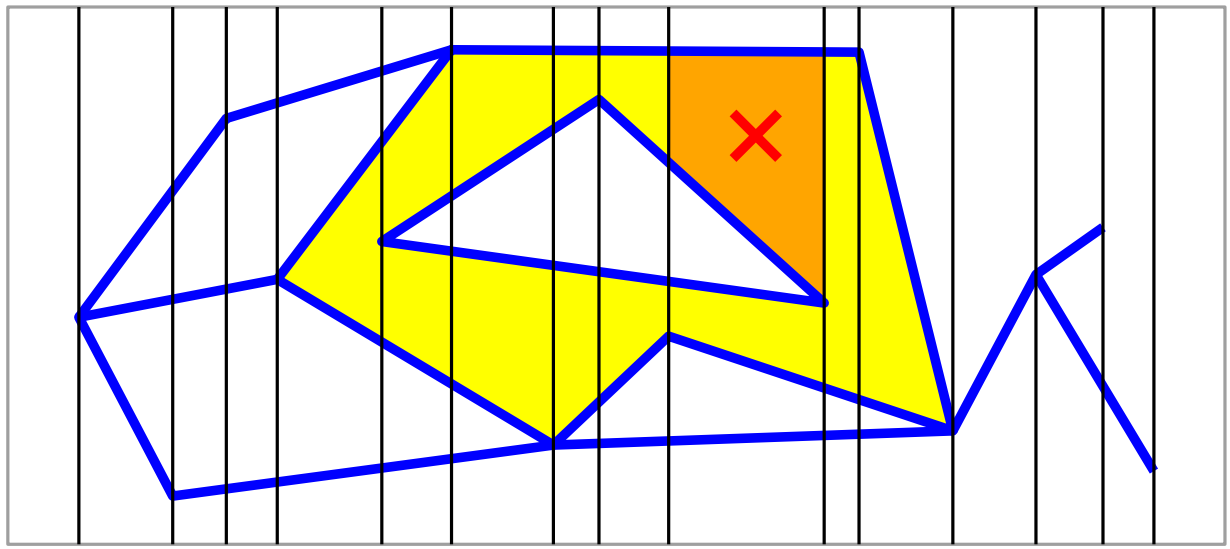
or

Where am I?

Lecture #6

Prof. Dr. Alexander Wolff *Chair for Computer Science I*

What's the Problem?



Task: Given a planar subdivision $\mathcal{S}$ with n segments, preprocess $\mathcal{S}$ to allow for fast point location queries!

Solution: Pre-proc: Partition $\mathcal{S}$ into slabs induced by vertices.

Query: – find right slab
– search slab } 2 bin. searches! $O(\log n)$ time!

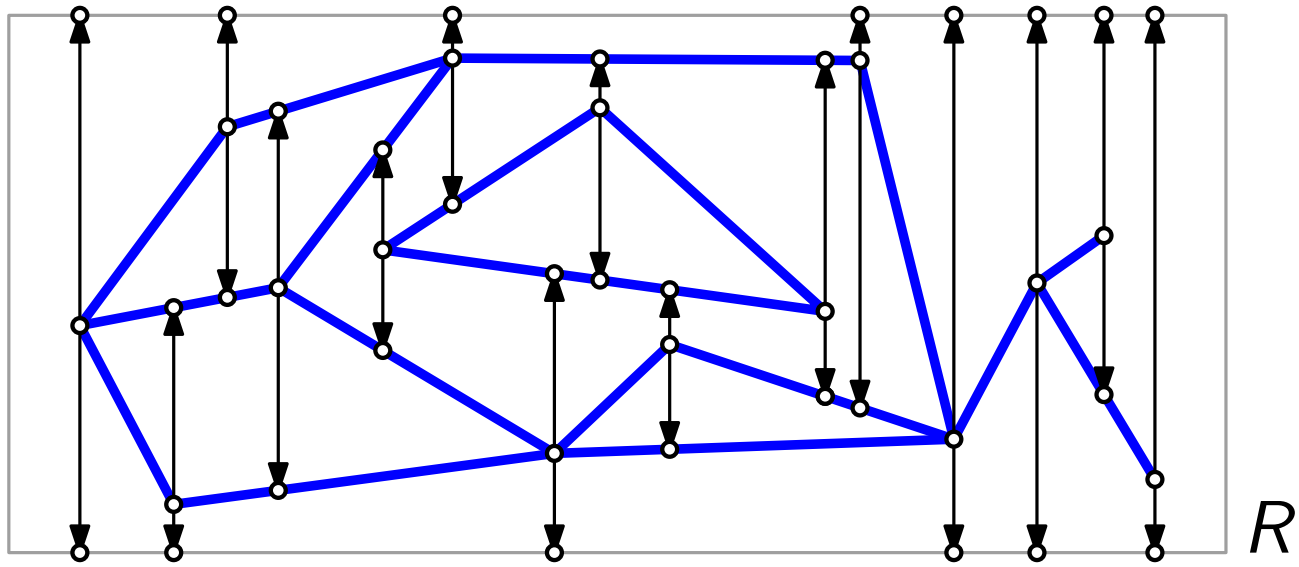
But: Space? $\Theta(n^2)$ Pre-proc? $O(n^2 \log n)$

Decreasing the Complexity

Observation: The slab partition of $\mathcal{S}$ is a *refinement* $\mathcal{S}'$ of $\mathcal{S}$ that consists of (possibly degenerate) trapezoids.

Task: Find "good" refinement of $\mathcal{S}$ of low complexity!

Solution: *Trapezoidal map* $\mathcal{T}(\mathcal{S})$

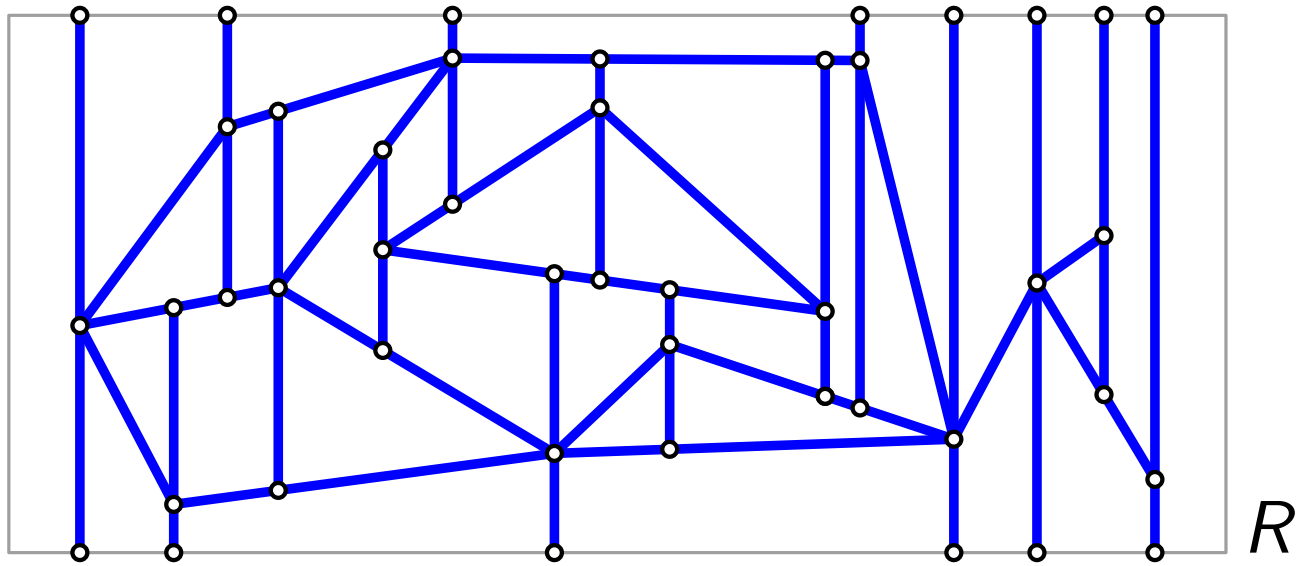


Decreasing the Complexity

Observation: The slab partition of $\mathcal{S}$ is a *refinement* $\mathcal{S}'$ of $\mathcal{S}$ that consists of (possibly degenerate) trapezoids.

Task: Find "good" refinement of $\mathcal{S}$ of low complexity!

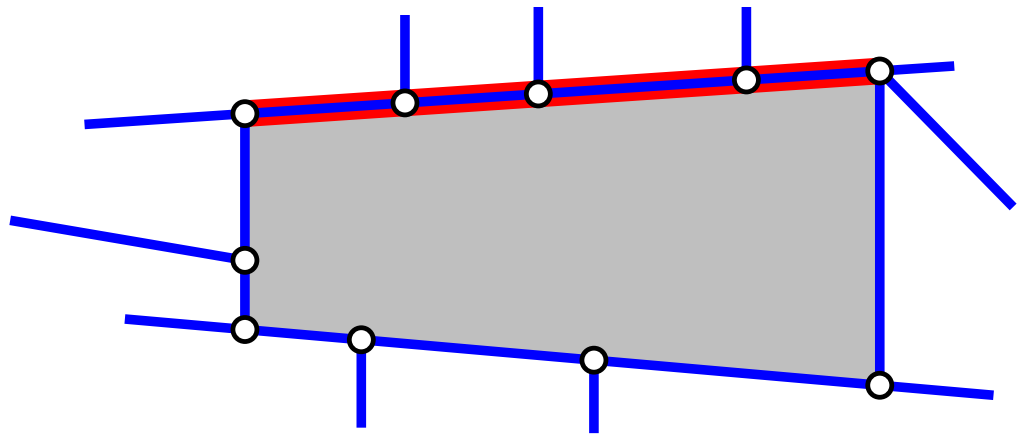
Solution: *Trapezoidal map* $\mathcal{T}(\mathcal{S})$



Assumption: $\mathcal{S}$ is in *general position*, that is, no two vertices have the same x-coordinates.

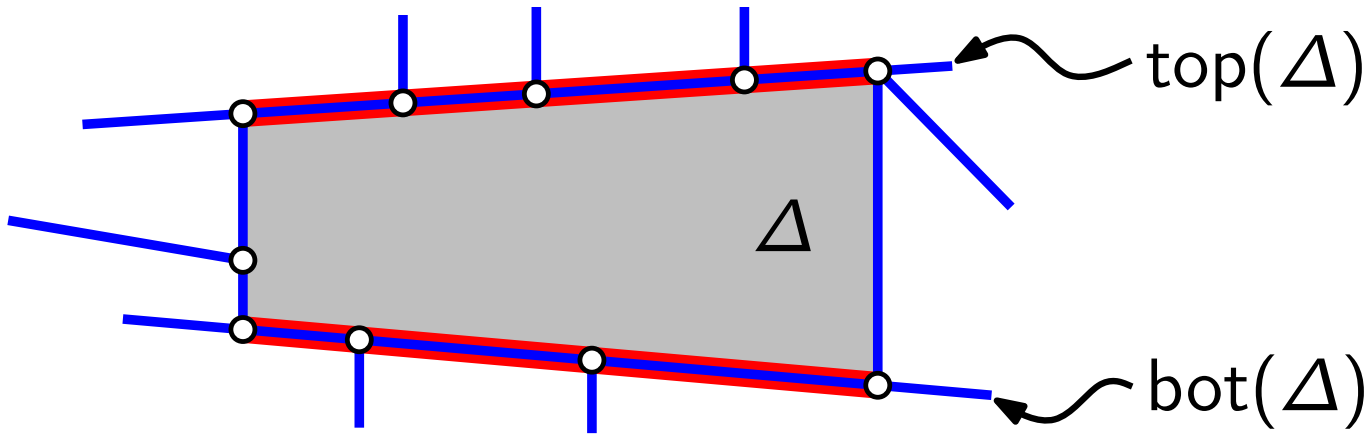
Notation

Definition: A *side* of a face of $\mathcal{T}(\mathcal{S})$ is a segment of maximum length contained in the boundary of the face.



Notation

Definition: A *side* of a face of $\mathcal{T}(\mathcal{S})$ is a segment of maximum length contained in the boundary of the face.

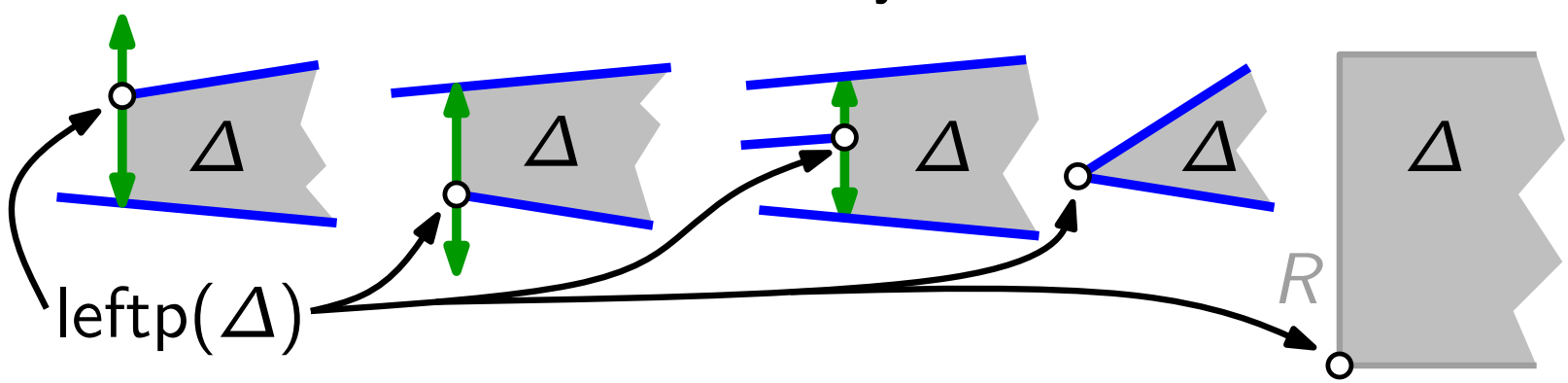


Observation: $\mathcal{S}$ in gen. pos. $\Rightarrow$ each face Δ of $\mathcal{T}(\mathcal{S})$ has:

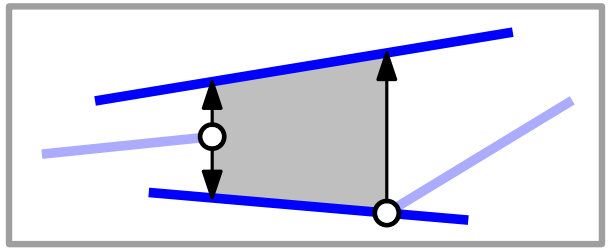
– one or two vertical sides

– exactly 2 non-vertical sides

Left side:



Complexity of $\mathcal{T}(\mathcal{S})$



Observe: A face Δ of $\mathcal{T}(\mathcal{S})$ is uniquely defined by $\text{top}(\Delta)$, $\text{bot}(\Delta)$, $\text{leftp}(\Delta)$, and $\text{rightp}(\Delta)$.

Lemma. $\mathcal{S}$ planar subdivision in gen. pos., with n segments $\Rightarrow \mathcal{T}(\mathcal{S})$ has $\leq 6n+4$ vtc and $\leq 3n+1$ trapezoids.

Proof. The vertices of $\mathcal{T}(S)$ are

– endpts of segments in $\mathcal{S}$	$\leq 2n$
– endpts of vertical extensions	$\leq 2 \cdot 2n$
– vertices of R	4

$\Big\} \leq 6n+4$

Bound #trapezoids via Euler or directly (segments/leftp).

Approach: Construct tapezoidal map $\mathcal{T}(\mathcal{S})$ and point-location data structure $\mathcal{D}(\mathcal{S})$ for $\mathcal{T}(\mathcal{S})$ *incrementally*!

algorithm-design paradigm!

The 1d-Problem

Given a set S of n real numbers... $i \in \{1, \ldots, n\}$

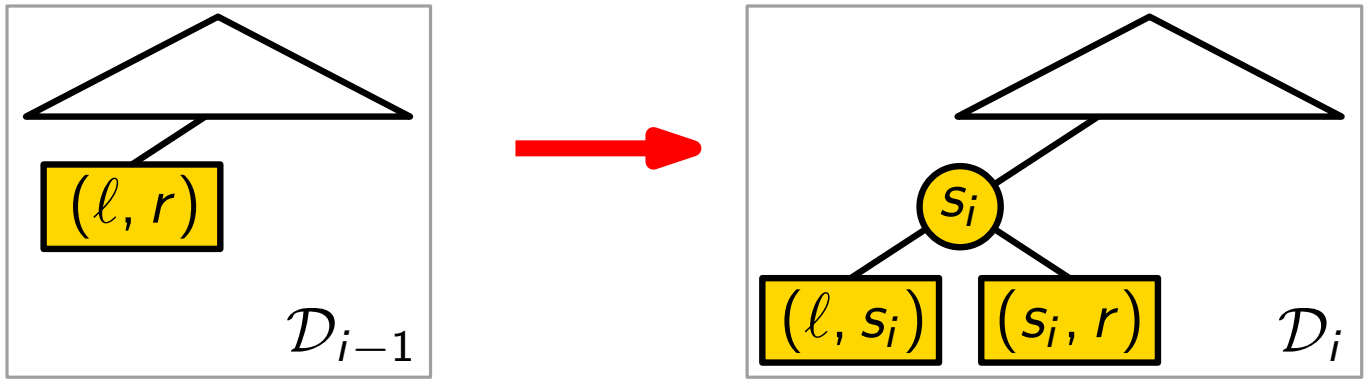
$S_{i-1} := \{s_1, \ldots, s_{i-1}\}$, $I_{i-1} :=$ set of conn. comp. of $\mathbb{R} \setminus S_{i-1}$

– pick an arbitrary point s_i from $S \setminus S_{i-1}$

– locate s_i in the search structure $\mathcal{D}_{i-1}$ of S_{i-1}

– split interval (ℓ, r) of I_{i-1} containing s_i

– build $\mathcal{D}_i$:



Problem: *looong* search paths!

The 1d-Problem

Given a set S of n real numbers... $i \in \{1, \ldots, n\}$

$S_{i-1} := \{s_1, \ldots, s_{i-1}\}$, $I_{i-1} :=$ set of conn. comp. of $\mathbb{R} \setminus S_{i-1}$

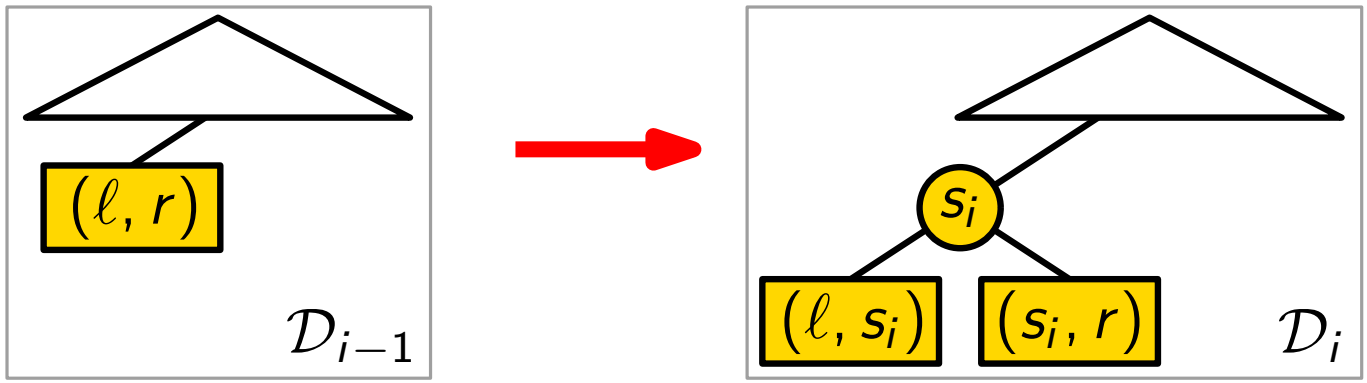
Solution: *random!*

– pick an ~~arbitrary~~ point s_i from $S \setminus S_{i-1}$

– locate s_i in the search structure $\mathcal{D}_{i-1}$ of S_{i-1}

– split interval (ℓ, r) of I_{i-1} containing s_i

– build $\mathcal{D}_i$:



~~**Problem:** *looong* search paths!~~

1d Result

Given a set S of n real numbers... $i \in \{1, \ldots, n+1\}$

$S_{i-1} := \{s_1, \ldots, s_{i-1}\}$, $I_{i-1} :=$ set of conn. comp. of $\mathbb{R} \setminus S_{i-1}$

Thm. The randomized-incremental algorithm preprocesses a set S of n reals in $O(n \log n)$ expected time such that a query takes $O(\log n)$ expected time.

Proof. Let $q \in \mathbb{R}$ (wlog. $q \notin S$) and $I_i(q) = \arg\{I \in I_i : q \in I\}$.

Define random variable $X_i = \begin{cases} 1 & \text{if } I_i(q) \neq I_{i-1}(q), \\ 0 & \text{else.} \end{cases}$

$E[\text{query time in } \mathcal{D}_n] = E[\text{length search path in } \mathcal{D}_n] =$

$= E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} E[X_i] =$ **?**

Expected Query Time of $\mathcal{D}_n$

$E[X_i] = P[X_i = 1] = 2/i$

$=$ probability that $l_i(q) \neq l_{i-1}(q)$, i.e., $s_i \in l_{i-1}(q)$.

Backwards analysis: Consider S_i fixed.
If we *remove* a randomly chosen pt from S_i, what's the probability that the interval containing q changes?
– we have i choices, identically distributed
– at most two of these change the interval

Define random variable $X_i = \begin{cases} 1 & \text{if } l_i(q) \neq l_{i-1}(q), \\ 0 & \text{else.} \end{cases}$

$E[\text{query time in } \mathcal{D}_n] = E[\text{length search path in } \mathcal{D}_n] =$

$= E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} E[X_i] =$ **?**

$O(\log n)$

The 1d-Result

Thm. The randomized-incremental algorithm preprocesses a set S of n reals in $O(n \log n)$ expected time such that a query takes $O(\log n)$ expected time.

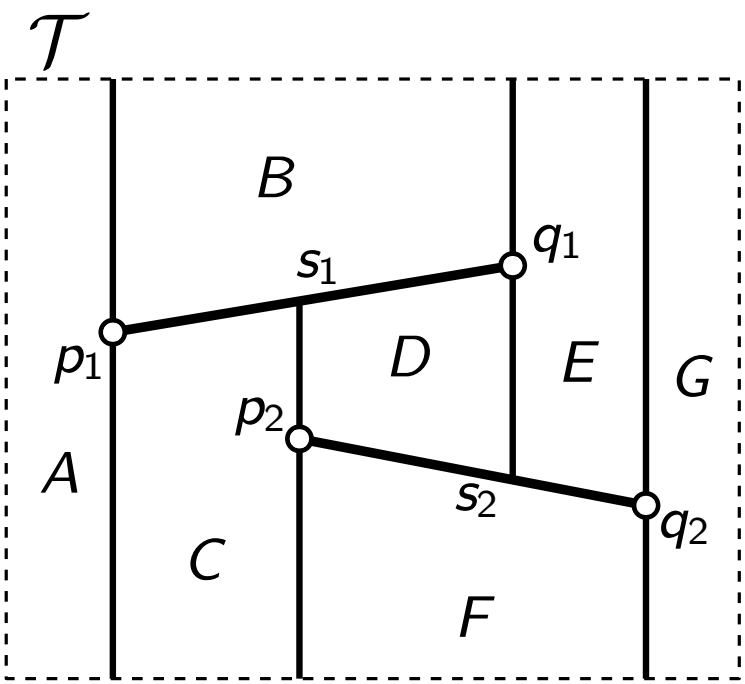
The 2d-Problem

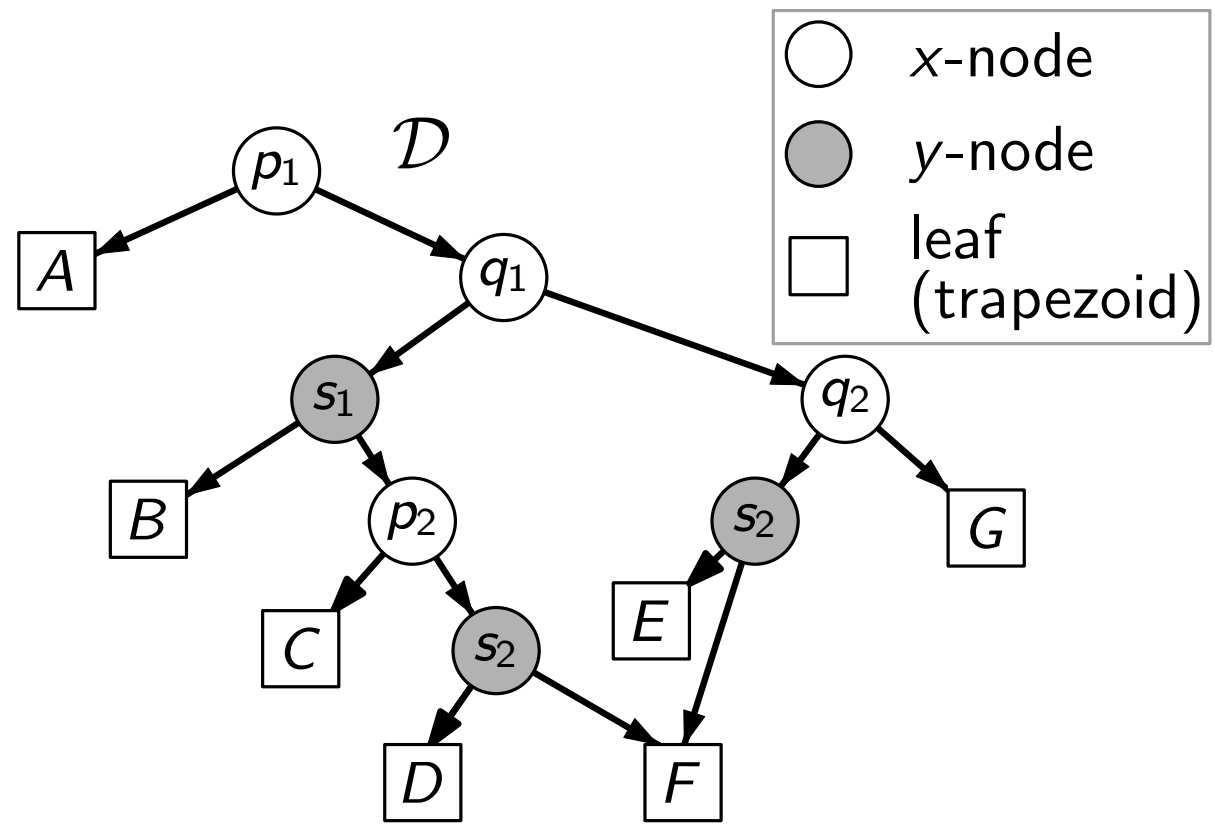
point-location data structure (DAG)

trapezoidal map

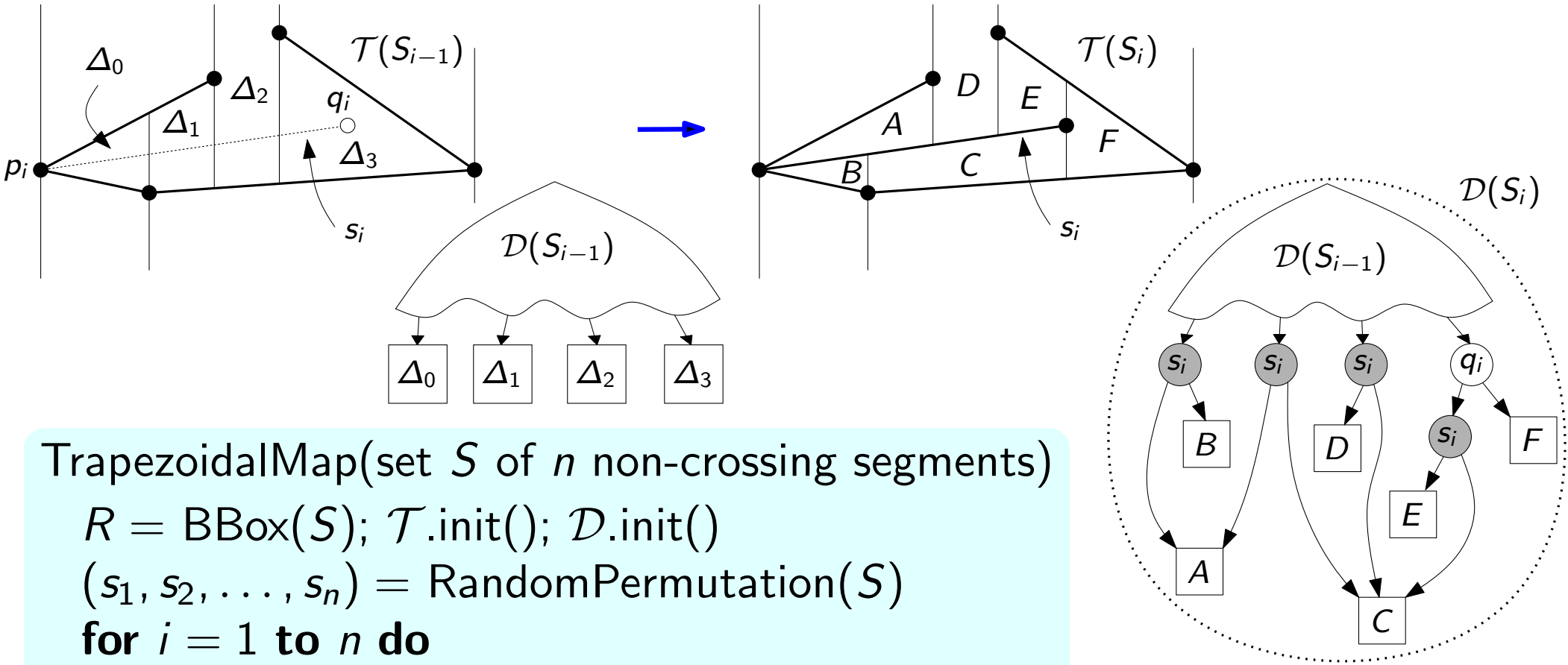
Approach: randomized-incremental construction of $\mathcal{T}$ and $\mathcal{D}$

- use $\mathcal{D}$ to locate left endpoint of next segment s
- "walk" along s through $\mathcal{T}$
- destroy all trapezoids of $\mathcal{T}$ intersecting s
- construct new trapezoids of $\mathcal{T}$ (adjacent to s)
- update $\mathcal{D}$





Walking through $\mathcal{T}$ and Updating $\mathcal{D}$



TrapezoidalMap(set S of n non-crossing segments)

$R = \text{BBox}(S)$; $\mathcal{T}$.init(); $\mathcal{D}$.init()

$(s_1, s_2, \ldots, s_n) = \text{RandomPermutation}(S)$

for $i = 1$ **to** n **do**

- $(\Delta_0, \ldots, \Delta_k) = \text{FollowSegment}(\mathcal{T}, \mathcal{D}, s_i)$
- $\mathcal{T}$.remove$(\Delta_0, \ldots, \Delta_k)$
- $\mathcal{T}$.add(new trapezoids incident to s_i)
- $\mathcal{D}$.remove_leaves$(\Delta_0, \ldots, \Delta_k)$
- $\mathcal{D}$.add_leaves(new trapezoids incident to s_i)
- $\mathcal{D}$.add_new_inner_nodes()

The 2d-Result

Theorem. trapezoidalMap(S) computes $\mathcal{T}(S)$ for a set of n line segments in general position and a search structure $\mathcal{D}$ for $\mathcal{T}(S)$ in $O(n \log n)$ expected time. The expected size of $\mathcal{D}$ is $O(n)$ and the expected query time is $O(\log n)$.

Invariant: Before step i, $\mathcal{T}$ is a trapezoidal map for S_{i-1} and $\mathcal{D}$ is a valid search structure for $\mathcal{T}$.

Proof.
- Correctness by loop invariant.
- Query time similar to 1d analysis.

  $\Rightarrow$ construction time