Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Lehrstuhl für
INFORMATIK I
Effiziente Algorithmen und
wissensbasierte Systeme

Institut für Informatik

Computational Geometry

Winter semester 2016/17

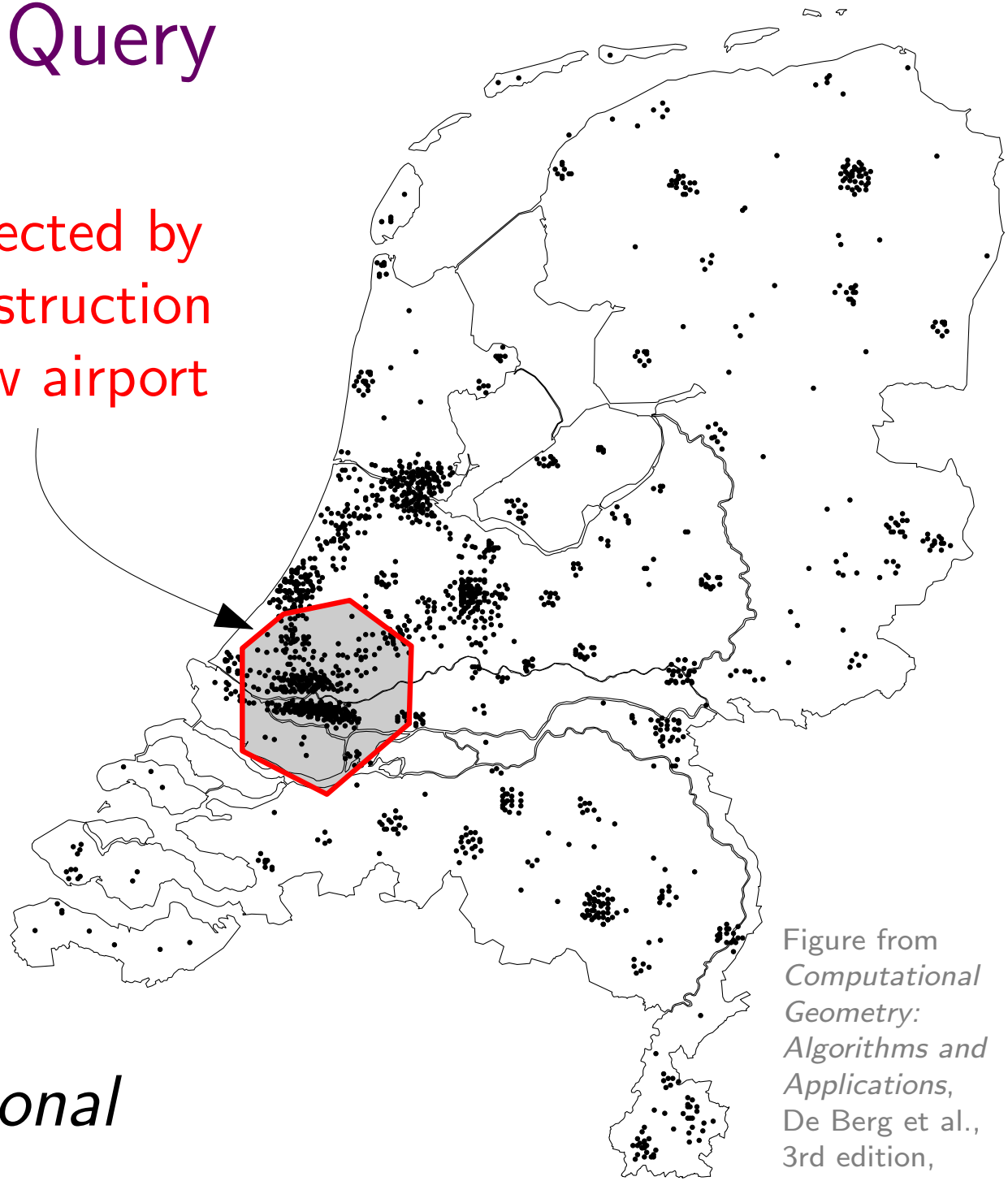
Simplex Range Searching

Lecture #11
(Chapter 16)

Joachim Spoerhase *Chair for Computer Science I*

Range-Counting Query

area affected by
the construction
of a new airport



Observation:

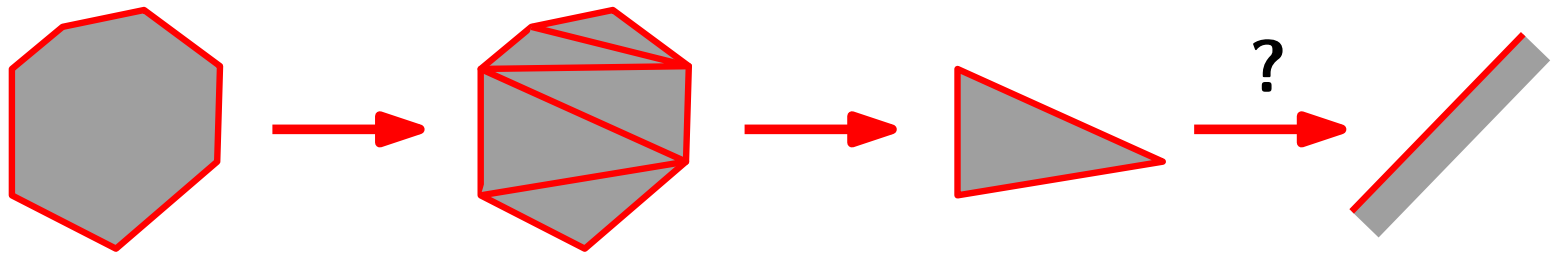
Query range depends on, e.g., dominant wind directions

$\Rightarrow$ *non-orthogonal*

Figure from *Computational Geometry: Algorithms and Applications*, De Berg et al., 3rd edition, Springer 2008.

Non-orthogonal range queries

Query range:



Problem: Given a set P of n points, preprocess P such that *half-space range-counting queries* can be answered quickly.

Task: Design a data structure for the 1-dim. case:

– Given a number x, return $|P \cap [x, \infty)|$.

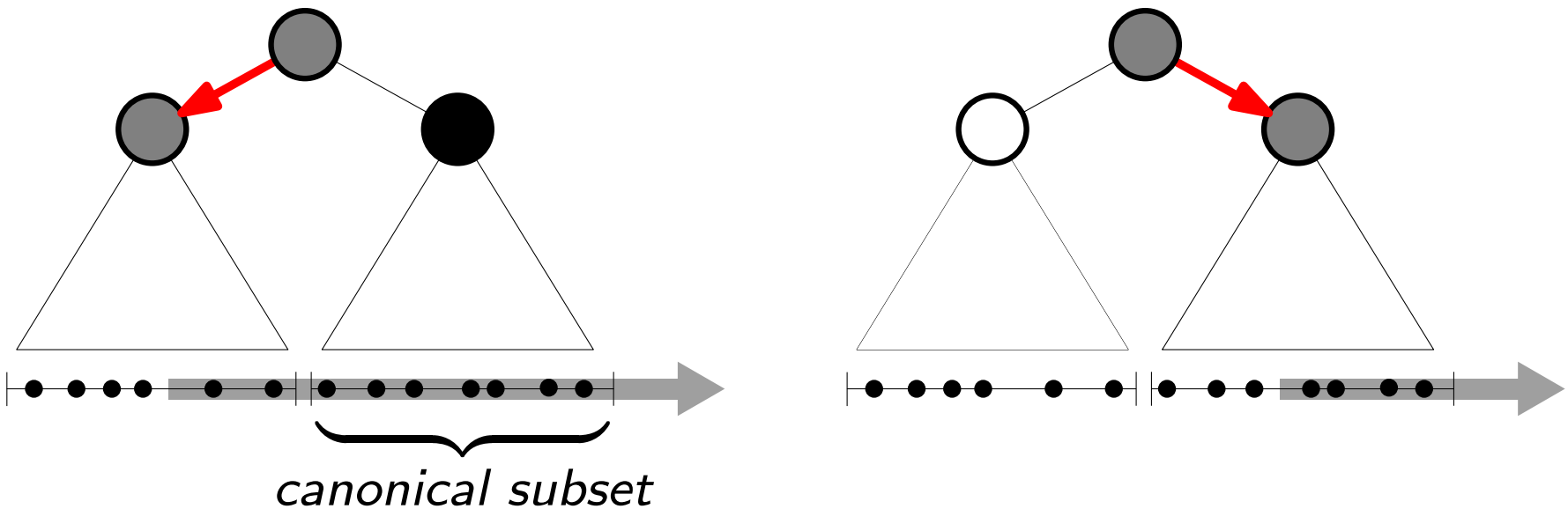
– Consider P static / dynamic!

The 1-Dimensional Case

Task: Design a data structure for the 1-dimensional case!

Solution:

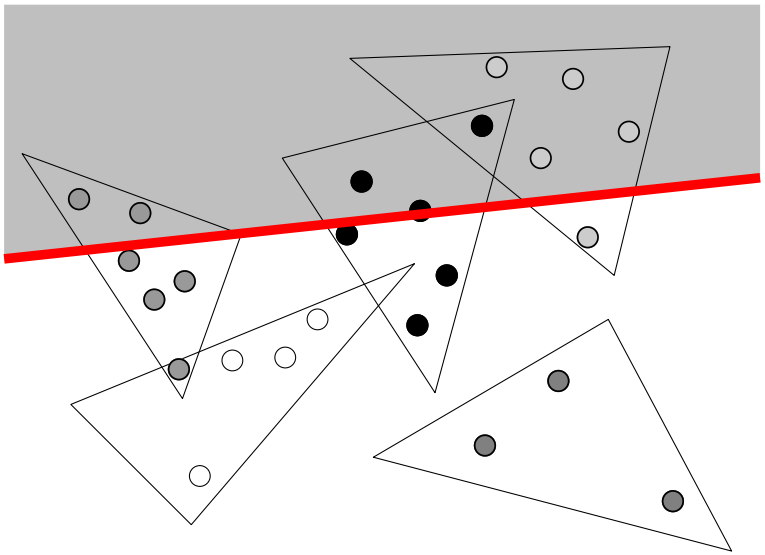
- use balanced binary search trees
- augment each node with the number of nodes in its subtree [see Cormen et al., *Introduction to Algorithms*, MIT press, 3rd ed., 2009]

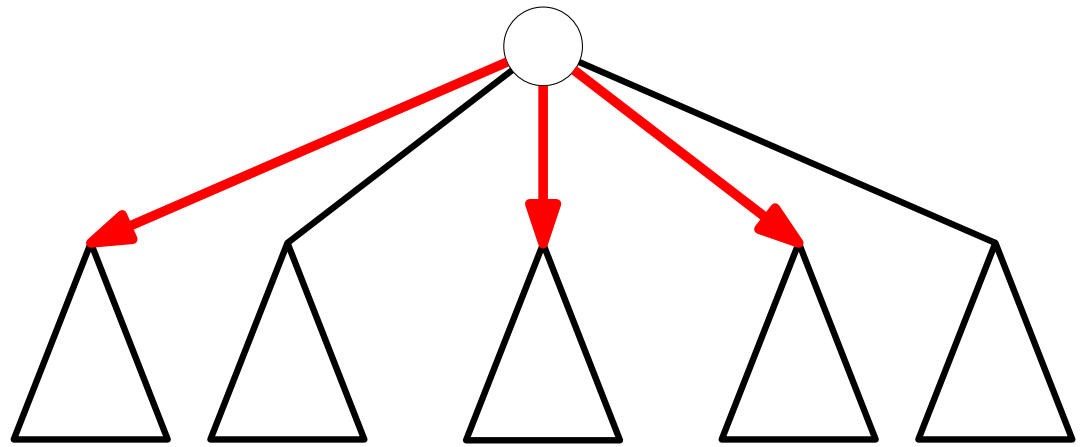


Lesson: On each level, must visit ≤ 1 subtree recursively!

Generalizing to 2 Dimensions

Partition the input! Query... in a *partition tree* ...recursively!





Definition: $\Psi(S) = \{(S_1, t_1), (S_2, t_2), \ldots, (S_r, t_r)\}$ is a *simplicial partition* (of size r) for S if

classes of S

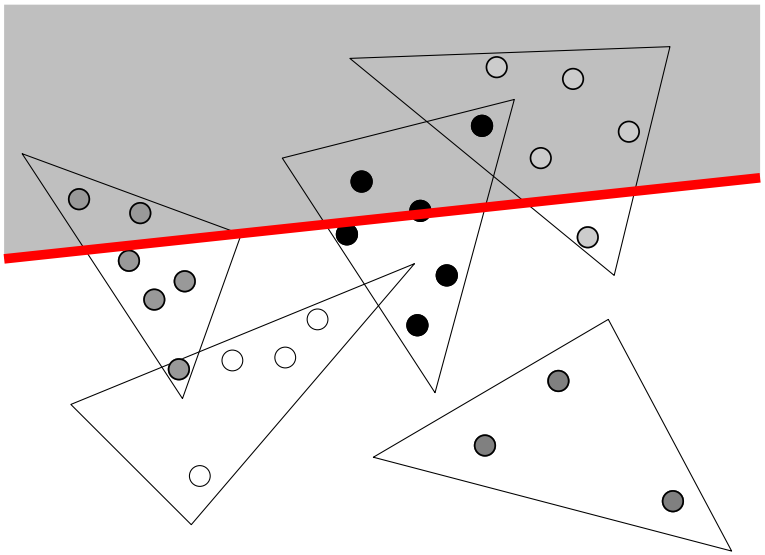
– S is partitioned by $S_1, \ldots, S_r$ and

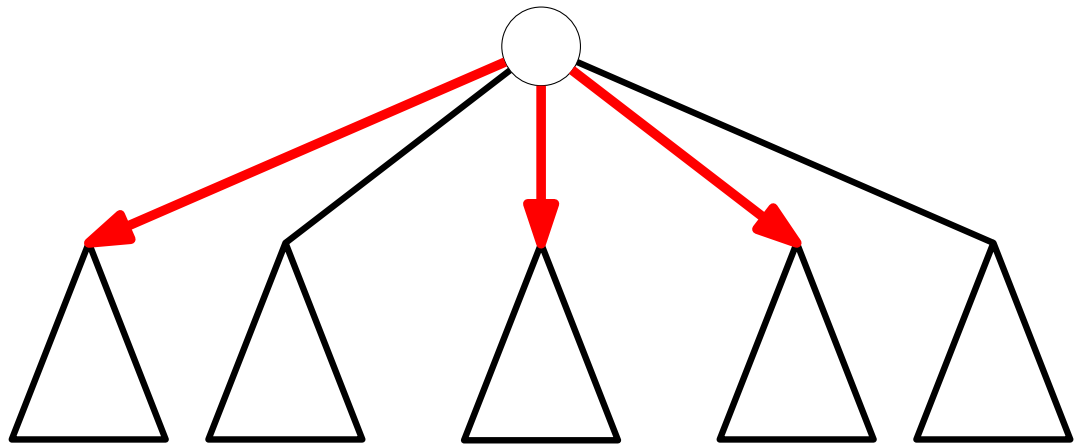
– for $1 \leq i \leq r$, t_i is a triangle and $S_i \subset t_i$.

$\Psi(S)$ is *fine* if $|S_i| \leq 2|S|/r$ for every $1 \leq i \leq r$.

Generalizing to 2 Dimensions

Partition the input! Query... in a *partition tree* ...recursively!



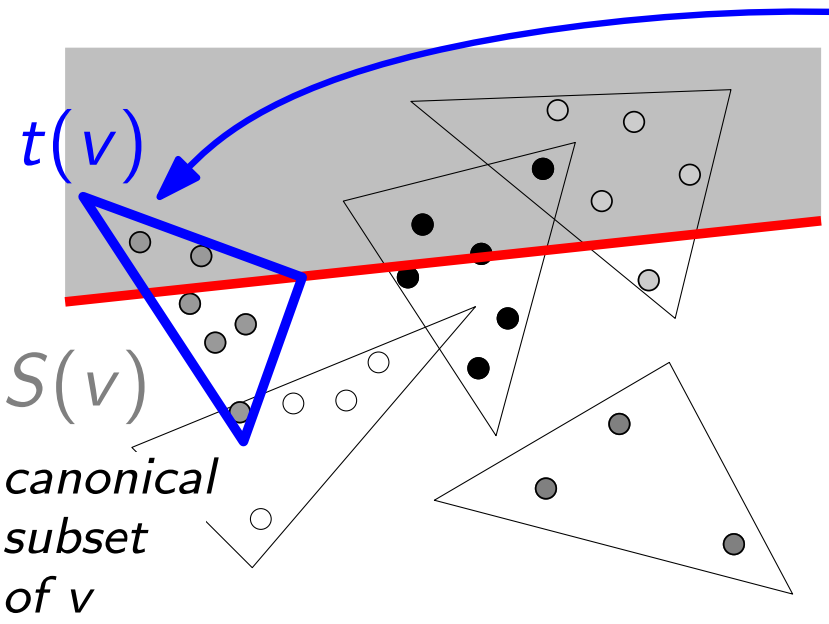


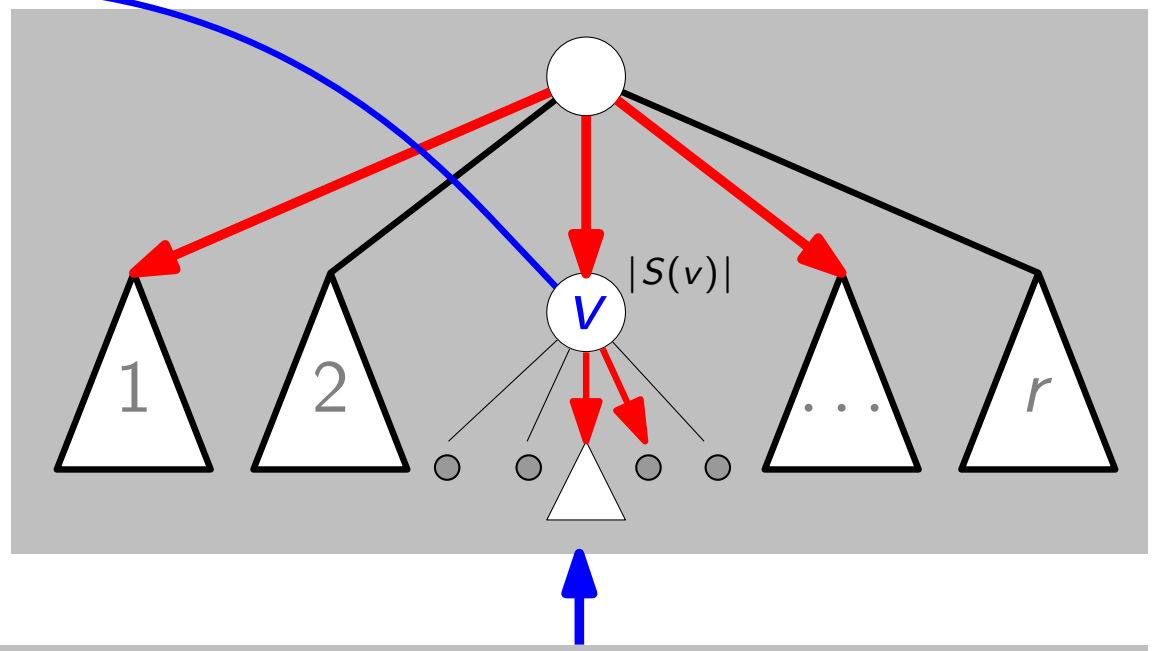
Definition: The *crossing number* of ℓ (w.r.t. $\Psi(S)$) is the number of triangles $t_1, \ldots, t_r$ *crossed* by ℓ.

The *crossing number* of $\Psi(S)$ is the maximum crossing number over all possible lines.

Generalizing to 2 Dimensions

Partition the input! Query... in a *partition tree* ...recursively!

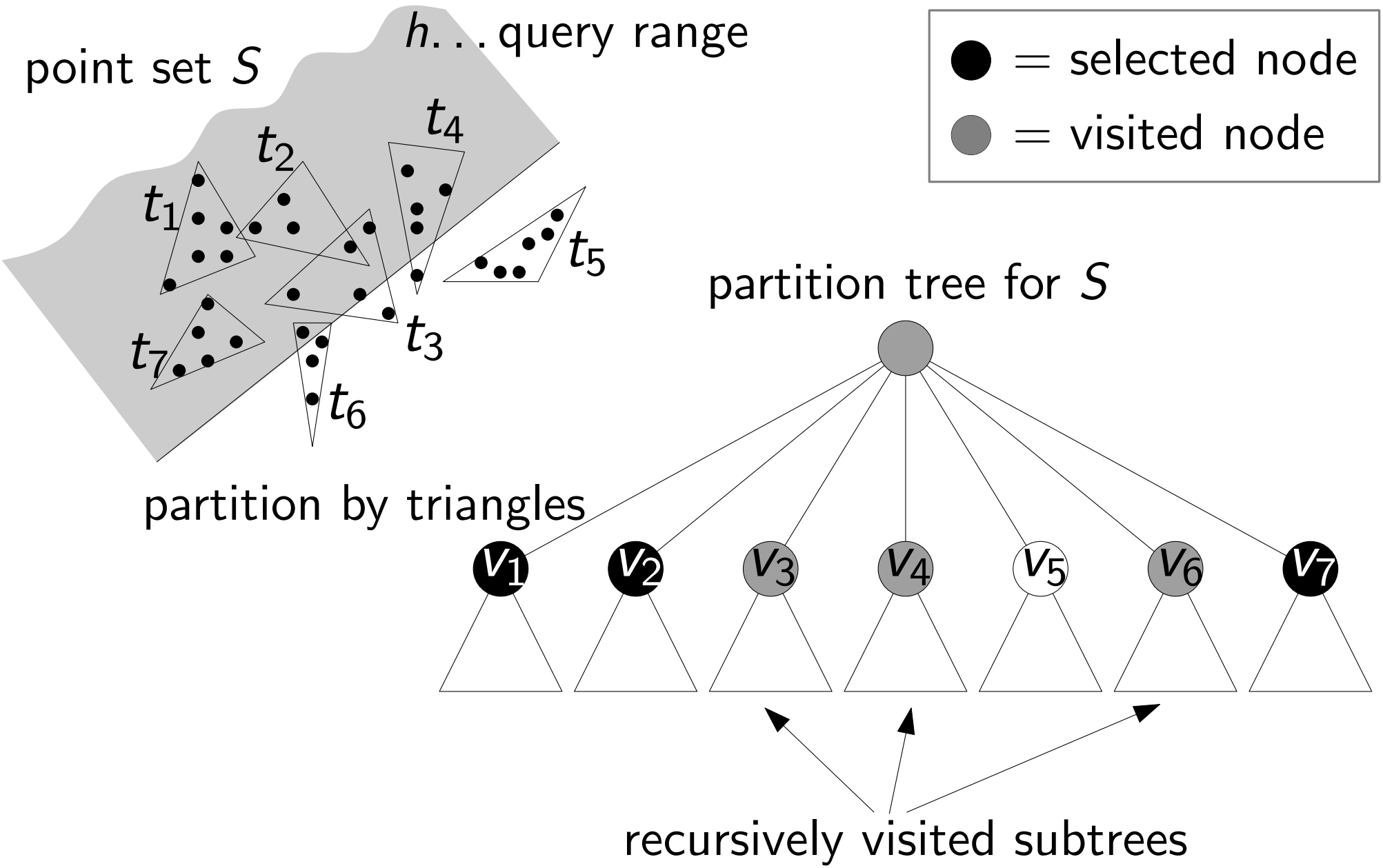




Theorem. For any set S of n pts and any $1 \leq r \leq n$, a fine simplicial partition of size r and crossing number $O(\sqrt{r})$ exists.

For any $\varepsilon > 0$, such a partition can be built in $O(n^{1+\varepsilon})$ time.

Example for a Query



Query Algorithm

SELECTINHALFPLANE(half-plane h, partit. tree $\mathcal{T}$ for pt set S)

$N \leftarrow \emptyset$ { set of *selected* nodes }

if $\mathcal{T} = \{\mu\}$ **then**
  if point stored at μ lies in h **then**
    $N \leftarrow \{\mu\}$
else
  foreach child ν of the root of $\mathcal{T}$ **do**
    if $t(\nu) \subset h$ **then**
      $N \leftarrow N \cup \{\nu\}$
    else
      if $t(\nu) \cap h \neq \emptyset$ **then**
        $N \leftarrow N \cup \text{SELECTINHALFPLANE}(h, \mathcal{T}_\nu)$
return N { with $S \cap h = \bigcup_{\nu \in N} S(\nu)$ }

Task:

Turn this into a range *counting* query algorithm!

Analysis of the Partition Tree

Let S be a set of n points in the plane. *Recall:*

Theorem. For any r with $1 \leq r \leq n$, S has a fine simplicial partition of size r and crossing number $O(\sqrt{r})$. For any $\varepsilon > 0$, such a partition can be computed in $O(n^{1+\varepsilon})$ time.

Lemma. A partition tree for S can be constructed in $O(n^{1+\varepsilon})$ time. The tree uses $O(n)$ storage.

Lemma. For any $\varepsilon > 0$, there is a partition tree $\mathcal{T}$ for S s.t.:

for a query half-plane h,
SelectInHalfplane selects in $O(n^{1/2+\varepsilon})$ time
a set N of $O(n^{1/2+\varepsilon})$ nodes of $\mathcal{T}$
with the property that $h \cap S = \bigcup_{\nu \in N} S(\nu)$.

Corollary. Half-plane range counting queries can be answered in $O(n^{1/2+\varepsilon})$ time using $O(n)$ space & $O(n^{1+\varepsilon})$ prep.

Back to *Triangular* Range Queries

Any ideas? Just use SelectInHalfplane!

Theorem: Given a set S of n pts in the plane, for any $\varepsilon > 0$, a triangular range-counting query can be answered in $O(n^{1/2+\varepsilon})$ time using a partition tree.

The tree can be built in $O(n^{1+\varepsilon})$ time and uses $O(n)$ space.

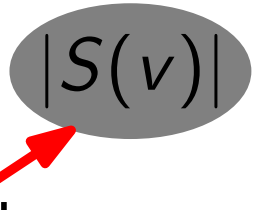
The points inside the query range can be reported in $O(k)$ additional time, where k is the number of reported pts.

Can we do better?

Use cutting trees! (Chapter 16.3)

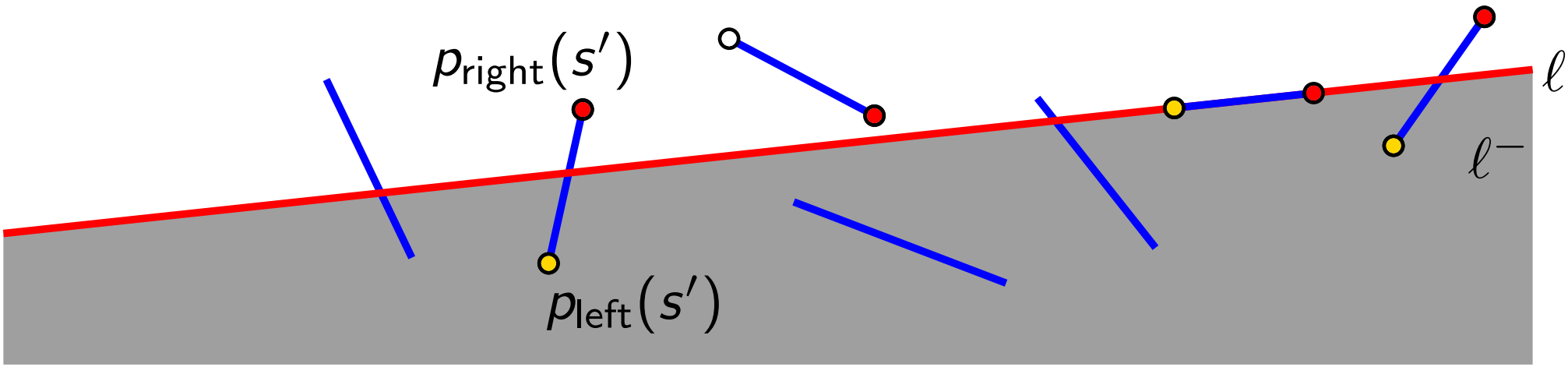
Query time $O(\log^3 n)$, prep. & storage $O(n^{2+\varepsilon})$.

Multi-Level Partition Trees



Idea: Store with each internal node not just a number, but another data structure!

Task: Design a fast data structure for line segments that counts all segments intersecting a query line ℓ.



Query Algorithm

For $S' \subseteq S$, let
$P_{\text{right}}(S') = \{p_{\text{right}}(s) \mid s \in S'\}$
left left

SelectIntSegments(line ℓ, two-level partition tree $\mathcal{T}$ for S)

– first-level tree stores $P_{\text{right}}(S)$
– second-level trees store subsets of $P_{\text{left}}(S)$

$N \leftarrow \emptyset$
if $\mathcal{T} = \{\mu\}$ **then**
  if segment stored in μ intersects ℓ **then** $N \leftarrow \{\mu\}$
else
  foreach child ν of $\mathcal{T}$'s root **do**
    if $t(\nu) \subset \ell^+$ **then**
      $N \leftarrow N \cup$ SelectInHalfplane(ℓ^-, $\mathcal{T}_\nu^{\text{assoc}}$)
    else
      if $t(\nu) \cap \ell \neq \emptyset$ **then**
        $N \leftarrow N \cup$ SelectIntSegments($\ell, \mathcal{T}_\nu$)
return N

stores $P_{\text{left}}(S_{\text{seg}}(\nu))$, where
$S_{\text{seg}}(\nu) = \{s \mid p_{\text{right}}(s) \in S(\nu)\}$

below above ?

!!! $\bigcup_{\nu \in N} S(\nu) = \{s \in S \mid p_{\text{right}}(s)$ above ℓ and $p_{\text{left}}(s)$ below $\ell\}$.

Results

Lemma: A 2-level partition tree for line-intersection queries among a set of n segments uses $O(n \log n)$ storage.

Lemma: Let S be a set of n segments in the plane. For any $\varepsilon > 0$, there is a 2-level partition tree $\mathcal{T}$ for S s.t.

- given a query line ℓ, we can select $O(n^{1/2+\varepsilon})$ nodes from $\mathcal{T}$ whose canonical subsets represent the segments intersected by ℓ.
- The selection takes $O(n^{1/2+\varepsilon})$ time.

Corollary: Let S be a set of n segments in the plane. After $O(...)$-time preprocessing, we can count the number of segments in S intersected by a query line in $O(n^{1/2+\varepsilon})$ time.