Halfplane and Simplex Range Searching

Given: A set S of n points in the plane.

Task: Process S into a data structure such that queries of one of the following types can be answered efficiently:

- **Halfplane range query:**
  Report or count all points in S that lie in a given query halfplane.
- **Triangular ("simplicial") range query:**
  Report or count all points in S that lie in a given query triangle ("simplex").

- Range counting queries should be answered in a runtime independent of the number of points in the query range
- Range reporting queries in output-sensitive runtime, depending on the number k of the points in the query range
- Consider first halfplane range queries, and construct "similar" structures for triangular range queries

- Remember our results for rectangular range queries (axis-parallel query rectangle):

	storage	construction time	query time
kd-trees	$O(n)$	$O(n \log n)$	$O(\sqrt{n} + k)$
range trees	$O(n \log n)$	$O(n \log n)$	$O(\log^2 n + k)$ (or $O(\log n + k)$ with fractional cascading)

- Triangular range queries allow to answer arbitrary polygonal range queries (triangulate query region in advance)
  → more general applications; in GIS for example

- Today we will study:

  * Halfplane range searching

	storage	constr. time	query time
partition tree	$O(n)$	$O(n^{1+\varepsilon})$	$O(n^{\frac{1}{2}+\varepsilon} + k)$
dual arrangement	$O(n^2)$	$O(n^2 \log n)$	$O(\log n + k)$

  * Triangular range searching

partition tree	$O(n)$	$O(n^{1+\varepsilon})$	$O(n^{\frac{1}{2}+\varepsilon} + k)$
cutting tree	$O(n^{2+\varepsilon})$	$O(n^{2+\varepsilon})$	$O(\log^3 n + k)$

for any fixed $\varepsilon > 0$, where the constant of proportionality in the O-notation depends on ε.

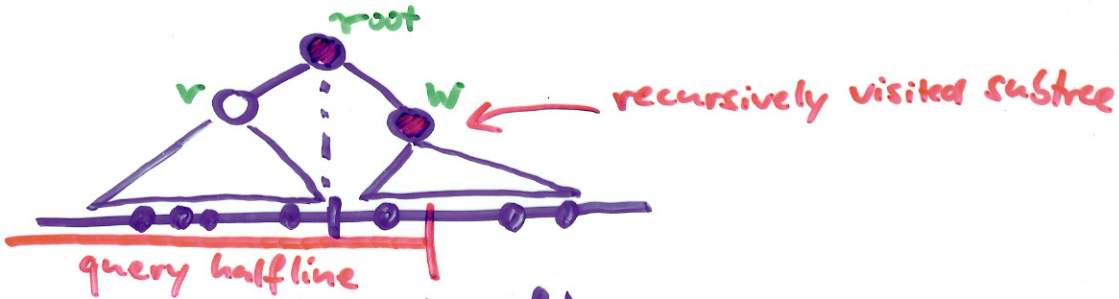
Partition Trees

Answer halfplane range counting queries efficiently for a set S of n points in the plane.

Warm-up: One-dimensional range counting for half-line queries

- balanced binary search tree
- each node stores number of points in its subtree
- " " " a split coordinate used to split the point set into the left and right subtree

↝ every node of the tree corresponds to a region on the line



- During a query, the region of one child of a node
  - is either completely contained in the query halfline (→ all points in that region are contained in the query halfline),
  - or is completely disjoint from the query halfline (→ no point is contained in the query)

  ↝ Recurse only on one subtree, and sum up counts on the path

Generalization to 2D: Partition the plane into a number of regions such that we have to search recursively only in very few regions.

A collection

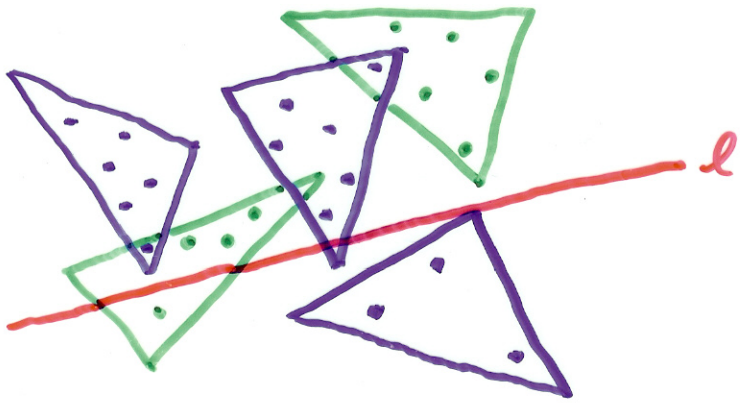
$$\Psi(S) := \{(S_1, t_1), \dots, (S_r, t_r)\}$$

where

- $S_i \subseteq S$ for all $1 \le i \le r$
- $S_i \cap S_j = \emptyset$ for all $i \neq j$; $1 \le i, j \le r$
- $\bigcup_{i=1}^{r} S_i = S$
- t_i is a triangle containing S_i; $1 \le i \le r$

is a simplicial partition of the set S of n points in $\mathbb{R}^2$?

- The S_i are called classes
- The number of triangles r is called the size of $\Psi(S)$
- The triangles are not necessarily disjoint



- Crossing # of ℓ = 2
- Crossing # of $\Psi(S)$ = 4

- A line ℓ intersects a triangle t_i if ℓ intersects the interior of t_i
- The number of triangles of $\Psi(S)$ crossed by a line ℓ is the crossing number of the line ℓ.
- Crossing number of $\Psi(S)$:= $\max_{\text{lines } \ell}$ (crossing number of ℓ)
- $\Psi(S)$ is fine $:\Leftrightarrow |S_i| \le \frac{2n}{r}$ for all $1 \le i \le r$

Use simplicial partition in half-plane range queries:

- Let ℓ^+ be the query half-plane with bounding line ℓ.
- If t_i is not crossed by ℓ then S_i lies either completely in ℓ^+ or is disjoint from ℓ^+.
- Recurse only on classes S_i for which t_i is crossed by ℓ.

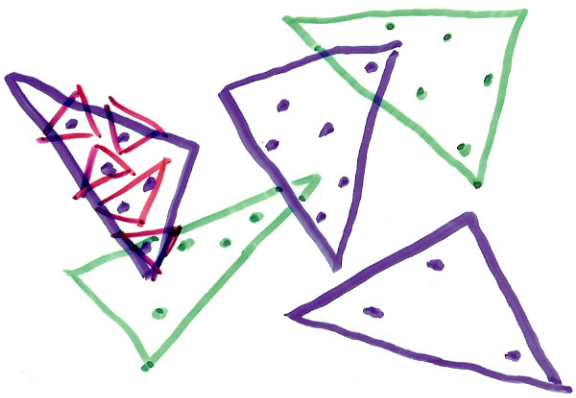
$\Rightarrow$ Query time depends on crossing number of $\psi(S)$

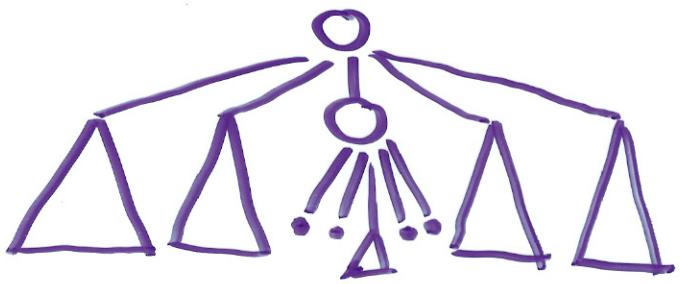
Theorem 1 (Matoušek): For any set S of n points in the plane and any parameter r with $1 \le r \le n$ there exists a simplicial partition of size r and crossing number $O(\sqrt{r})$, and it can be constructed in $O(n^{1+\varepsilon})$ time, for any $\varepsilon > 0$.

- Fix any $\varepsilon > 0$, e.g. 0.01, and the construction time is $O(n^{1+\varepsilon})$, so e.g. $O(n^{1.01})$
- The constant of proportionality in the O-notation depends on ε (and grows with decreasing ε)
- $O(n^{1+\varepsilon})$ is worse than $O(n)$ or $O(n \log n)$ or even $O(n \log^c n)$ for a constant $c > 0$.

Partition tree for S (recursively defined):

- $|S|=1$: Partition tree consists of single leaf, storing the single point as the canonical subset of the leaf
- Pick r to be a sufficiently large constant (exact choice shown below)
- $|S|>1$: Partition tree is a tree $\mathcal{T}$ of branching degree r;
  Let $\Psi(S)$ be a fine simplicial partition of size r of S.
  Then the root of $\mathcal{T}$ has a child for every triangle in $\Psi(S)$.
  - Denote $t(v)$ to be the triangle of $\Psi(S)$ corresponding to child node v
  - Denote $S(v)$ to be the class of $\Psi(S)$ corresponding to child node v. $S(v)$ is called the canonical subset of v
  - Recursively define every child v to be the root of a partition tree $\mathcal{T}_v$ on the set $S(v)$
- With each child v store the triangle $t(v)$, and other relevant information about $S(v)$ → for range counting store $|S(v)|$





- Order on the children is irrelevant

Query algorithm for a query half-plane h:

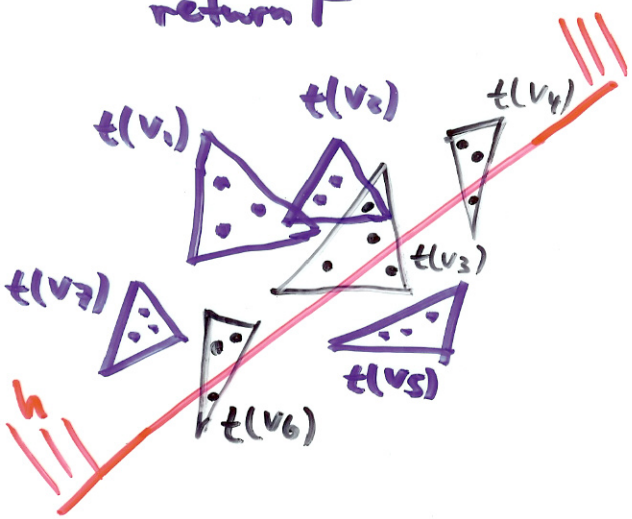
- Return a set Γ of nodes from $\mathcal{T}$ such that

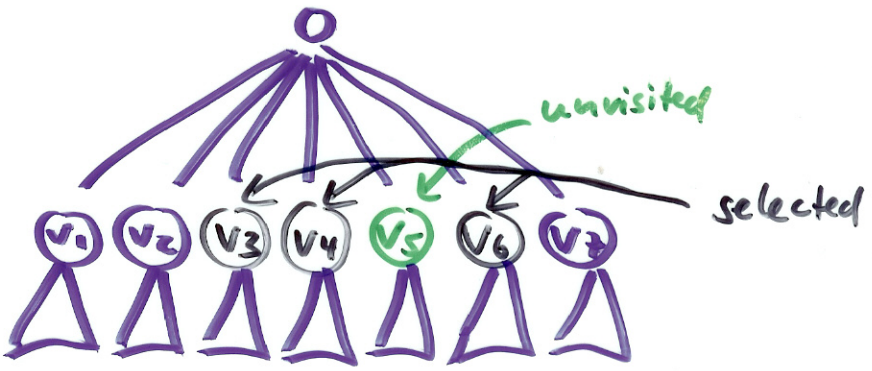
$$S \cap h = \bigcup_{v \in \Gamma} S(v)$$

- Call Γ the set of **selected nodes**
- $\Gamma = \{ v \text{ node in } \mathcal{T} \mid t(v) \subseteq h \text{ and there is no ancestor } \mu \text{ of } v \text{ such that } t(\mu) \subseteq h \}$

Algorithm Select_Nodes_in_Halfplane $(h, \mathcal{T})$:

$\Gamma := \emptyset$
if $\mathcal{T}$ consists of a single leaf μ **then**
  if point stored at μ lies in h **then** $\Gamma := \{\mu\}$
else for each child v of the root of $\mathcal{T}$ **do**
    if $t(v) \subseteq h$ **then** $\Gamma := \Gamma \cup \{v\}$
    else if $t(v) \cap h \neq \emptyset$ **then**
      $\Gamma := \Gamma \cup$ Select_Nodes_in_Halfplane $(h, \mathcal{T}_v)$
return Γ





Lemma 2: A partition tree of n points uses $O(n)$ storage.

Proof: Let $M(n) :=$ max number of nodes a partition tree on n points can have. Let $n_v := |S(v)|$

$$\Rightarrow M(n) \leq \begin{cases} 1 & , n = 1 \\ 1 + \sum_{v \text{ children of root}} M(n_v) & , n > 1 \end{cases}$$

Since $\sum_v n_v = n$, the recurrence solves to $M(n) = O(n)$ for $r > 2$.
Storage for single node of tree is $O(r)$. □

Lemma 3: For any $\varepsilon > 0$ there is a partition tree for S such that for any half-plane h the selected nodes Γ can be computed in $O(n^{\frac{1}{2}+\varepsilon})$ time; and $|\Gamma| = O(n^{\frac{1}{2}+\varepsilon})$. Thus half-plane range counting queries can be answered in $O(n^{\frac{1}{2}+\varepsilon})$ time.

Proof: Let $\varepsilon > 0$ fixed.

- Theorem 1 gives a constant c
  $\rightsquigarrow$ Simplicial partition of size r with crossing # $\leq c \cdot \sqrt{r}$
- Choose $r := \lceil 2(c \cdot \sqrt{2})^{1/\varepsilon} \rceil$, and construct partition tree
- Let $Q(n) :=$ max query time for any tree for n points
  Let h a query half-plane

$$\rightarrow \quad Q(n) \leq \begin{cases} 1 & , n = 1 \\ r + \sum_{v \in C(h)} Q(n_v) & , n > 1 \end{cases}$$

where $C(h) := \{\text{children } v \text{ of root} \mid \text{boundary of } h \text{ crosses } t(v)\}$

a) $|C(h)| \leq c \cdot \sqrt{r}$ cutting number

b) $n_v \leq 2n/r$ fine partition

$\rightsquigarrow$ Plugging a) and b) into the recurrence one can show that $Q(n) = O(n^{1/2+\varepsilon})$ □

Triangular range queries:

- Same approach as for half-plane queries
- A triangle t in the partition is crossed by the boundary of a query triangle Δ iff t is crossed by one of the three lines bounding Δ
  $\rightsquigarrow$ crossing number of Δ is $\leq 3 \cdot c\sqrt{r}$

Theorem 4: Let S be a set of n points in the plane. For any $\varepsilon > 0$ there is a partition tree for S that uses $O(n)$ storage and can be constructed in $O(n^{1+\varepsilon})$ time. It allows to answer triangular or half-plane range counting queries in $O(n^{\frac{1}{2}+\varepsilon})$ time, and range reporting queries in $O(n^{\frac{1}{2}+\varepsilon}+k)$ time, where k is the number of reported points.

Proof: Let $T(n)$ be the construction time. Let $\varepsilon > 0$. Construct a fine simplicial partition for S of size r with crossing # $O(\sqrt{r})$ in time $O(n^{1+\varepsilon'})$ with $\varepsilon' = \varepsilon/2$.

$$\rightsquigarrow T(n) = \begin{cases} O(1) & , n = 1 \\ O(n^{1+\varepsilon/2}) + \sum_{v \text{ children of root}} T(n_v) & , n > 1 \end{cases}$$

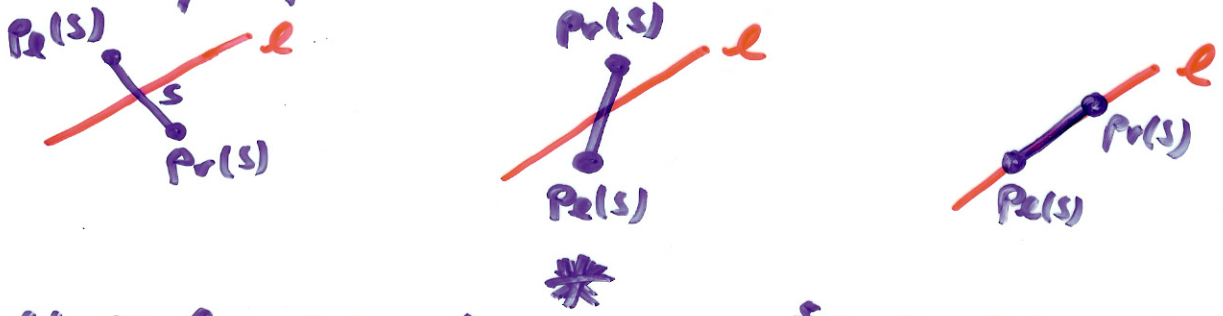
With $\sum_v n_v = n$ the recurrence solves to $T(n) = O(n^{1+\varepsilon})$.

For range reporting visit all leaves in the subtrees rooted at the selected nodes. Since the size of the subtrees is linear in the number of leaves (degree ≥ 2) $\rightsquigarrow$ time linear in # of reported points

Multi-Level Partition Trees:

Let S be a set of n line segments.

Task: Count the number of segments in S that intersect a query line ℓ.



Subtask: Count number of segments s in S with $p_r(s)$ lying above ℓ and $p_\ell(s)$ lying below ℓ.

⇝ Construct two-level partition tree:

- Construct partition tree to find all $s \in S$ with $p_r(s)$ lying above ℓ ⇝ canonical subsets
- Interested in # of segments s in canonical subsets such that $p_\ell(s)$ is below ℓ
  ⇝ halfplane range counting on canonical subset

Let $P_r(S) := \{p_r(s) \mid s \in S\}$ and $P_\ell(S) := \{p_\ell(s) \mid s \in S\}$

- Store $P_r(S)$ in partition tree $\mathcal{T}$
  Canonical subset of node v is denoted $P_r(v)$
  Let $S(v) := \{s \in S \mid p_r(s) \in P_r(v)\}$
- In each node v of the 1st-level tree $\mathcal{T}$ store $P_\ell(S(v))$ in a second-level partition tree $\mathcal{T}_v$ as used for half-plane range counting.

Query algorithm: Sum counters of selected canonical sets

Algorithm Select_Intersected_Segments($\ell, \mathcal{T}$):

Output: A set of nodes for all segments in the tree that are intersected by ℓ

$\Gamma := \emptyset$
if $\mathcal{T}$ consists of a single leaf μ **then**
  if the segment stored at μ intersects ℓ **then** $\Gamma := \{\mu\}$
else for each child v of the root of $\mathcal{T}$ **do**
    if $t(v) \subseteq \ell^+$ **then**
      $\Gamma := \Gamma \cup$ Select_Nodes_In_Halfplane$(\ell^-, \mathcal{T}_v^{assoc})$
    else if $t(v) \cap \ell \neq \emptyset$ **then**
      $\Gamma := \Gamma \cup$ Select_Nodes_In_Halfplane$(\ell, \mathcal{T}_v)$
return Γ

- In order to find segments with $p(s)$ above ℓ, simply exchange ℓ^+ and ℓ^-.

Lemma 5: The two-level partition tree uses $O(n \log n)$ storage.

Proof: Let $n_v := |S(v)|$. Let $M(n)$ be the storage complexity.

$$\rightarrow M(n) = \begin{cases} O(1) & , n = 1 \\ \sum_{v \text{ children}} (O(n_v) + M(n_v)) & , n > 1 \end{cases}$$

With $\sum_v n_v = n$ and $n_v \leq 2n/r$ this solves to $M(n) = O(n \log n)$. □

Lemma 6: For any $\varepsilon > 0$ there is a two-level partition tree that allows to select $O(n^{1/2+\varepsilon})$ nodes Γ for a query line, in $O(n^{1/2+\varepsilon})$ time.

Proof: From lemma 3 follows that the query time in $\mathcal{T}_v^{assoc}$ is $O(n_v^{1/2+\varepsilon})$. Let $Q(n)$ be the overall query time.

$$Q(n) = \begin{cases} O(1) & , n = 1 \\ O(r \cdot n^{1/2+\varepsilon}) + \sum_{i=1}^{c\cdot\sqrt{r}} Q(2n/r) & , n > 1 \end{cases}$$

□

Cutting Trees

Construct a data structure for planar simplicial range queries that has quadratic storage but logarithmic query time.

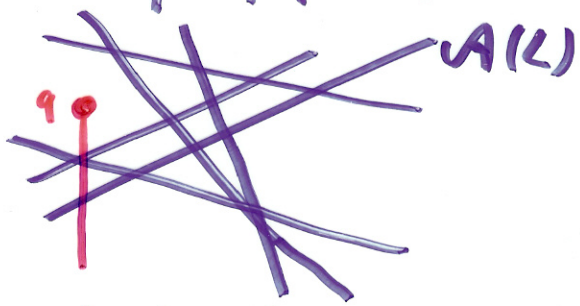
Halfplane range searching:

Primal problem

Given a set S of n points, count the number of points lying in a query half-plane (say above a line)

Dual problem

Given a set L of n lines, count the number of lines below a query point q

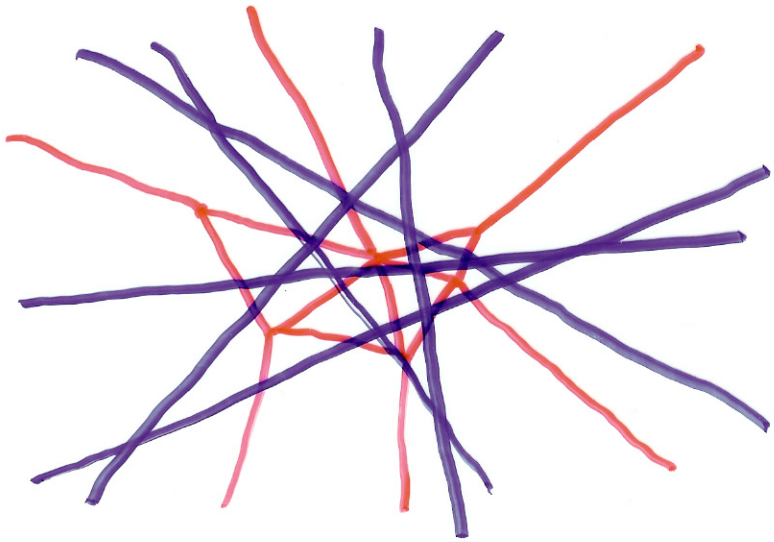


→ Construct $A(L)$, process for point location queries

- Store counter in each face
- $O(n^2)$ storage, $O(n^2 \log n)$ construction time, $O(\log n)$ query time

- Each face in the arrangement corresponds to a "precomputed" canonical subset of lines in L
- Does not immediately work for triangular range queries, however.

→ partition plane into disjoint triangles



- Set of lines $L = \{l_1, \dots, l_n\}$
- Partioning into disjoint (probably unbounded) triangles
  $\rightsquigarrow (\frac{1}{2})$-cutting of size 10

- Let t be a triangle in the partitioning, and l_i a line not intersecting t
  - → If l_i lies below (above) t then l_i lies below (above) any query point inside t
  - → If query point lies inside t the only lines to check are those that intersect t
- Store each triangle of the partitioning with a counter of how many lines are above / below
- Recursively define data structure for lines intersecting the triangle

Let r be a parameter with $1 \leq r \leq n$; $|L| = n$.

A $(\frac{1}{r})$-cutting for L is a set $\Xi(L) := \{t_1, \dots, t_m\}$ such that

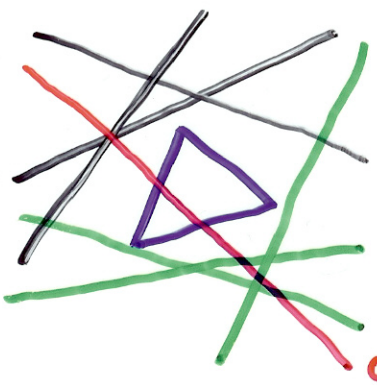
- each t_i is a, possibly unbounded, triangle ; $1 \leq i \leq m$
- all triangles in $\Xi(L)$ have disjoint interiors
- $\bigcup_{i=1}^{m} t_i = \mathbb{R}^2$ the triangles cover the plane
- no triangle in $\Xi(L)$ is crossed by more than n/r lines from L

The size of $\Xi(L)$ is the number m of triangles it consists of

Theorem 7: For any set L of n lines in the plane and any parameter $1 \le r \le n$ there exists a $(1/r)$-cutting of size $O(r^2)$. It can be constructed (together with the subset of lines in L that cross each triangle) in $O(nr)$ time.

Cutting Tree:

- If $|L|=1$, the tree consists of a single leaf where L is stored.
- If $|L|>1$ the structure is a tree T. The children of the root correspond to the triangles of a $(1/r)$-cutting for L
  - Choice of r shown below
  - Denote $t(v)$ to be the triangle of the cutting corresponding to child node v
  - $L^-(v) := \{\ell \in L \mid \ell \text{ lies below } t(v)\}$ lower canonical subset of v
  - $L^+(v) := \{\ell \in L \mid \ell \text{ lies above } t(v)\}$ upper canonical subset of v
  - $L^c(v) := \{\ell \in L \mid \ell \text{ crosses } t(v)\}$ Crossing subset of $t(v)$
  - child v is the root of a recursively defined cutting tree on its crossing subset $\rightsquigarrow$ tree T_v
- With each child v store $t(v)$, and other relevant information on $L^-(v)$, $L^+(v)$ such as $|L^-(v)|$



Query: Compute a set of selected nodes Γ, and sum the cardinalities of the canonical subsets

Algorithm Select_Below_Point($q, \mathcal{T}$):

Output: A set Γ of canonical nodes for all lines in the tree that lie below q

$\Gamma := \emptyset$
if $\mathcal{T}$ consists of a single leaf μ then
  if the line stored at μ lies below q then $\Gamma := \{\mu\}$
← else for each child v of the root of $\mathcal{T}$ do
    Check if q lies in $t(v)$
    Let v_q such that $q \in t(v_q)$
    $\Gamma := \{v_q\} \cup$ Select_Below_Point($q, \mathcal{T}_{v_q}$)
return Γ

Lemma 8: For any $\varepsilon > 0$ a cutting tree on L can be constructed that has $O(n^{2+\varepsilon})$ storage. A query runs in $O(\log n)$ time.

Proof: Query time $Q(n) = \begin{cases} O(1) & , n = 1 \\ Q(r^2) + Q(n/r) & , n > 1 \end{cases}$

Let c be the constant from Theorem 7 for a $(1/r)$-cutting of size $c \cdot r^2$.

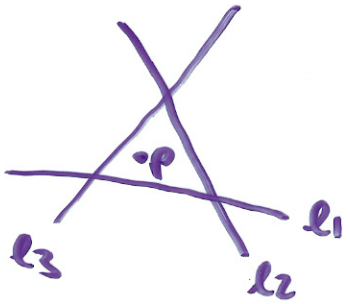
$\rightsquigarrow$ Set $r := \lceil (2c)^{1/\varepsilon} \rceil$

Storage $M(n) = \begin{cases} O(1) & , n = 1 \\ O(r^2) + \sum_v M(n_v) & , n > 1 \end{cases}$

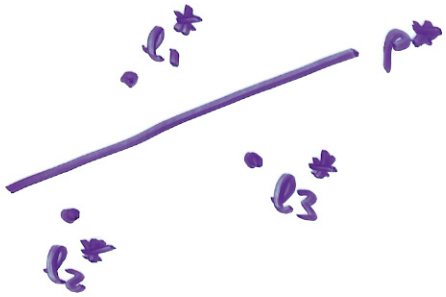
- # children of root $= c \cdot r^2$
- $n_v \leq n/r$

$\rightsquigarrow M(n) = O(n^{2+\varepsilon})$ □

Primal plane



Dual plane



$p \in l_1^+ \cap l_2^- \cap l_3^-$

$\Leftrightarrow$ p ~~below~~ above l_1, ~~above~~ below l_2, ~~below~~ above l_3

$\Leftrightarrow$ l_1^* is ~~below~~ above p^*, l_2^* ~~above~~ below p^*, l_3^* ~~above~~ below p^*

Thus triangular range queries in the primal plane translate to queries of the type: Count # of lines that lie on prespecified sides of three query points

$\rightsquigarrow$ 3-level cutting tree

Easier variant: Select lines that lie below a pair of query points $\rightsquigarrow$ 2-level cutting tree

- Store L in a cutting tree $\mathcal{T}$
- At each node v in $\mathcal{T}$ store $L^-(v)$ in a second-level cutting tree $\mathcal{T}_v^{assoc}$

Algorithm Select_Below_Pair($q_1, q_2, \mathcal{T}$):

$\Gamma := \emptyset$

if $\mathcal{T}$ consists of a single leaf μ then

  if line stored at μ lies below q_1 and q_2 then $\Gamma := \{\mu\}$

else for each child v of root of $\mathcal{T}$ do Check if q_1 lies in $t(v)$

  Let v_{q_1} be such that $q_1 \in t(v_{q_1})$

  $\Gamma_1 := $ Select_Below_Point$(q_2, \mathcal{T}_{v_{q_1}}^{assoc})$

  $\Gamma_2 := $ Select_Below_Pair$(q_1, q_2, \mathcal{T}_{v_{q_1}})$

  $\Gamma := \Gamma_1 \cup \Gamma_2$

return Γ

Lemma 9: Storage $O(n^{2+\varepsilon})$, query time $O(\log^2 n)$.

Proof: $Q(n) = \begin{cases} O(1) & , n=1 \\ O(r^2) + O(\log n) + Q(n/r) & , n>1 \end{cases}$

Storage $M(n) = \begin{cases} O(1) & , n=1 \\ \sum_v (O(n_v^{2+\varepsilon}) + M(n_v)) & , n>1 \end{cases}$

- # children of root $= O(r^2)$
- $n_v \leq n/r$

□

3-level cutting trees for triangular range searching are built analogously:

Theorem 10: Let S be a set of n points in the plane. For any $\varepsilon > 0$ a 3-level cutting tree can be constructed in $O(n^{2+\varepsilon})$ time that uses $O(n^{2+\varepsilon})$ storage. The points lying inside a query triangle can be counted in $O(\log^3 n)$ time and reported in $O(\log^3 n + k)$ time.