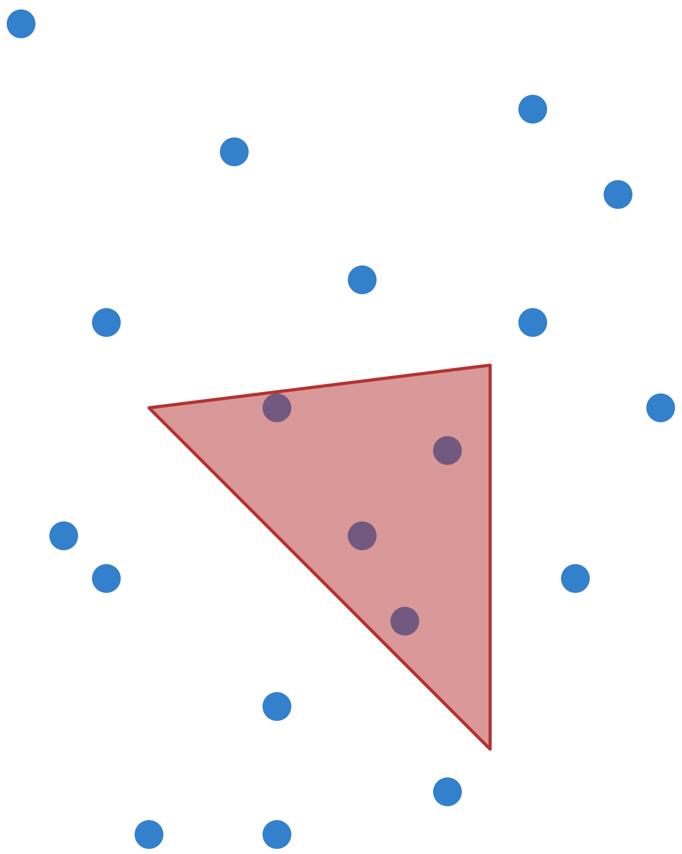
Geometric Divide & Conquer

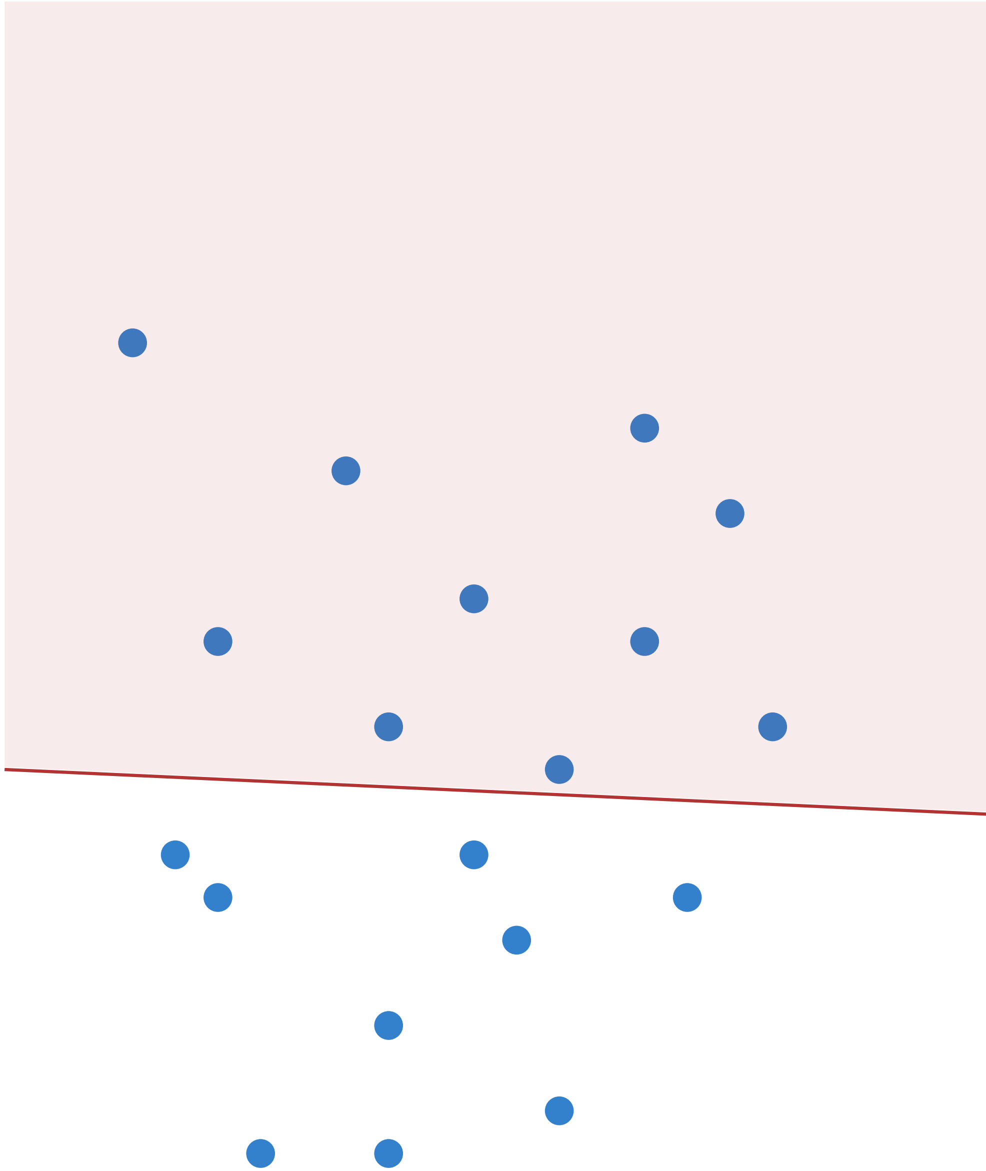
Cutting Trees

Given a set of n points P in $\mathbb{R}^2$. Store them in a data structure s.t. we can efficiently report the (number of) points from P that lie in a query triangle Q



Cutting Trees

Given a set of n points P in $\mathbb{R}^2$. Store them in a data structure s.t. we can efficiently report the (number of) points from P that lie in a query half plane h

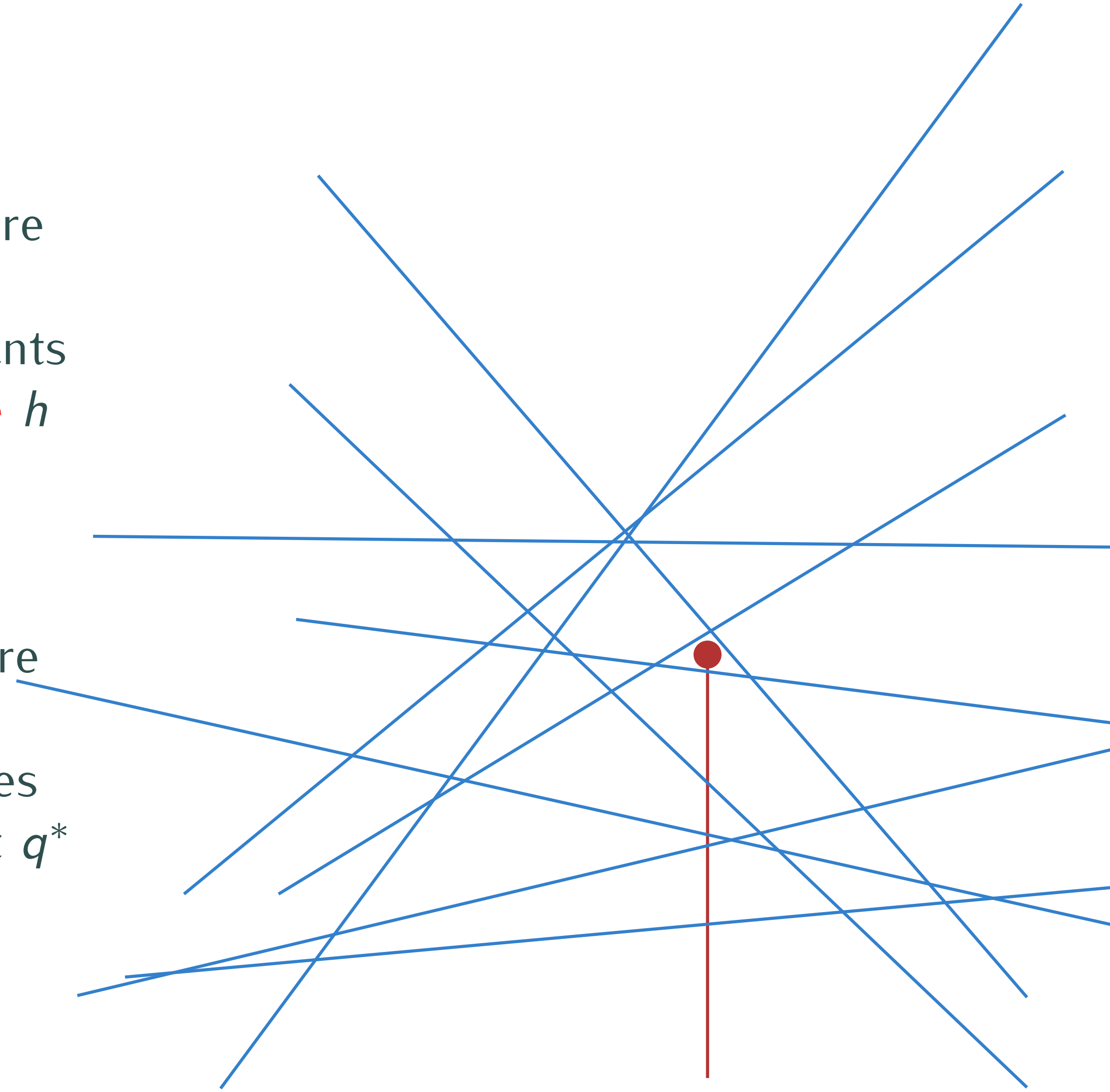


Cutting Trees

Given a set of n points P in $\mathbb{R}^2$. Store them in a data structure s.t. we can efficiently report the (number of) points from P that lie in a query half plane h

dualize the problem

Given a set of n lines P^* in $\mathbb{R}^2$. Store them in a data structure s.t. we can efficiently report the (number of) lines from P^* that lie below a query point q^*



Cutting Trees

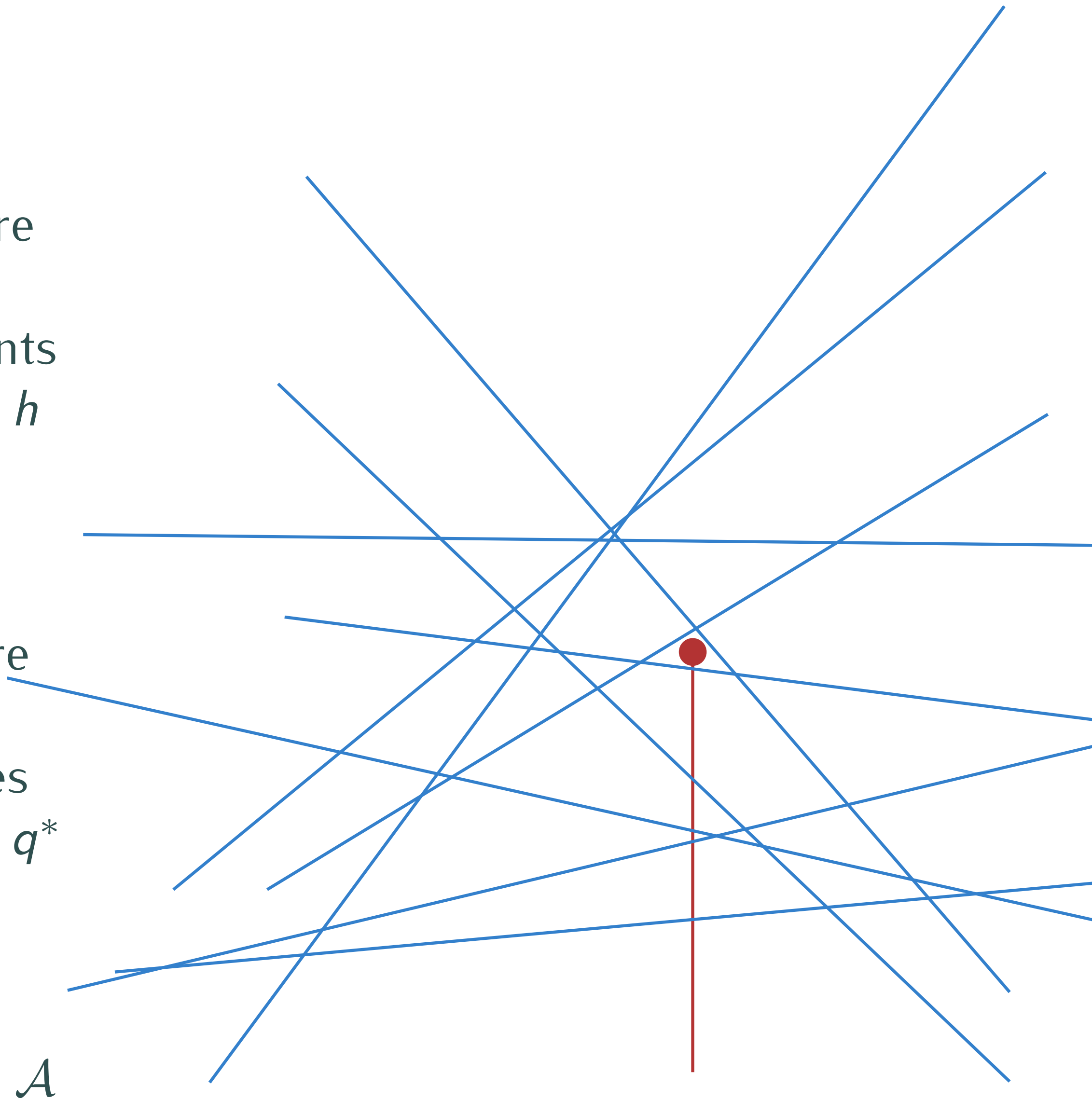
Given a set of n points P in $\mathbb{R}^2$. Store them in a data structure s.t. we can efficiently report the (number of) points from P that lie in a query half plane h

dualize the problem

Given a set of n lines P^* in $\mathbb{R}^2$. Store them in a data structure s.t. we can efficiently report the (number of) lines from P^* that lie below a query point q^*

build the arrangement $\mathcal{A}$, store the answer for every face, and prepocess $\mathcal{A}$ for point location

$O(n^2)$ space, $O(\log n)$ query.

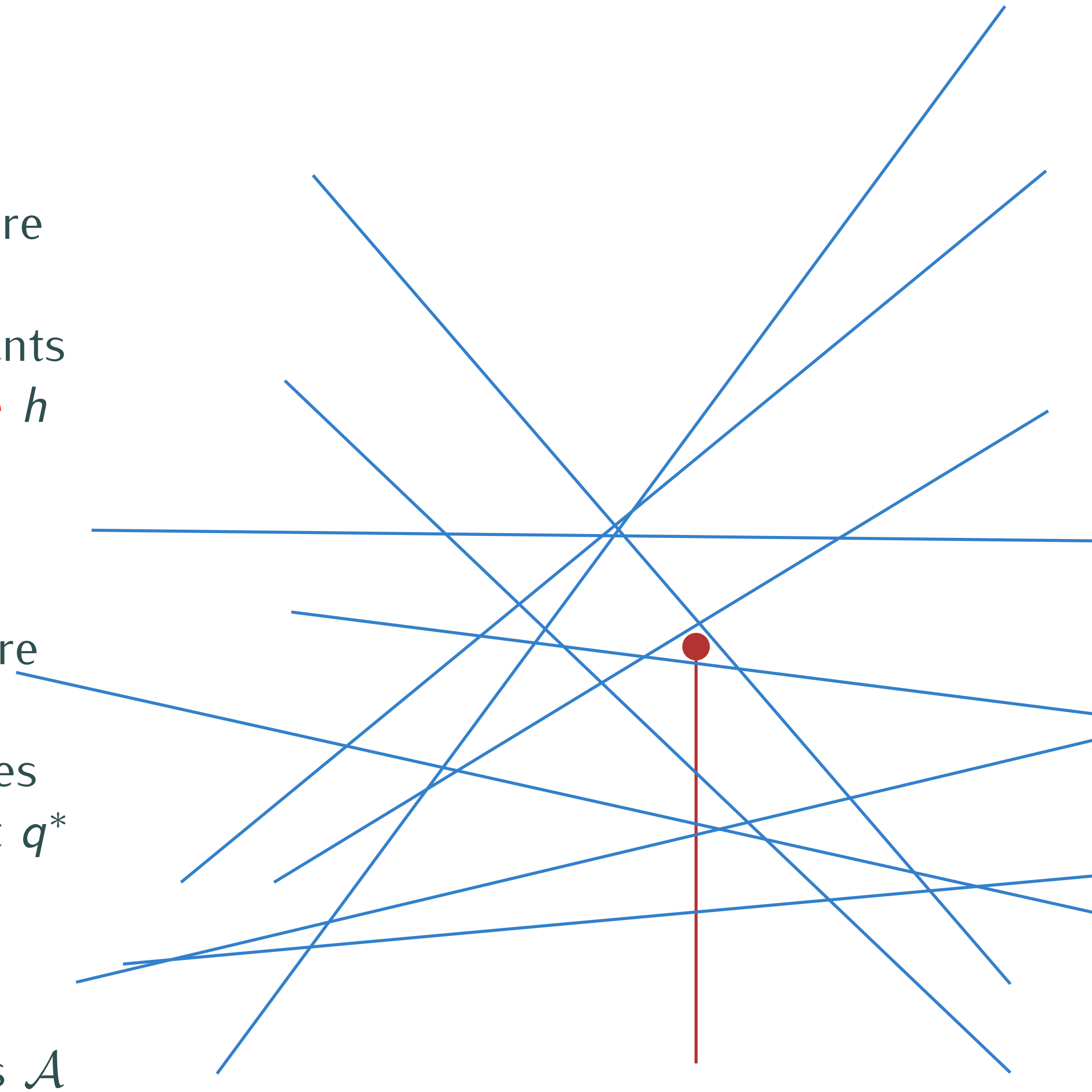


Cutting Trees

Given a set of n points P in $\mathbb{R}^2$. Store them in a data structure s.t. we can efficiently report the (number of) points from P that lie in a query half plane h

dualize the problem

Given a set of n lines P^* in $\mathbb{R}^2$. Store them in a data structure s.t. we can efficiently report the (number of) lines from P^* that lie below a query point q^*



build the arrangement $\mathcal{A}$, store the answer for every face, and prepocess $\mathcal{A}$ for point location

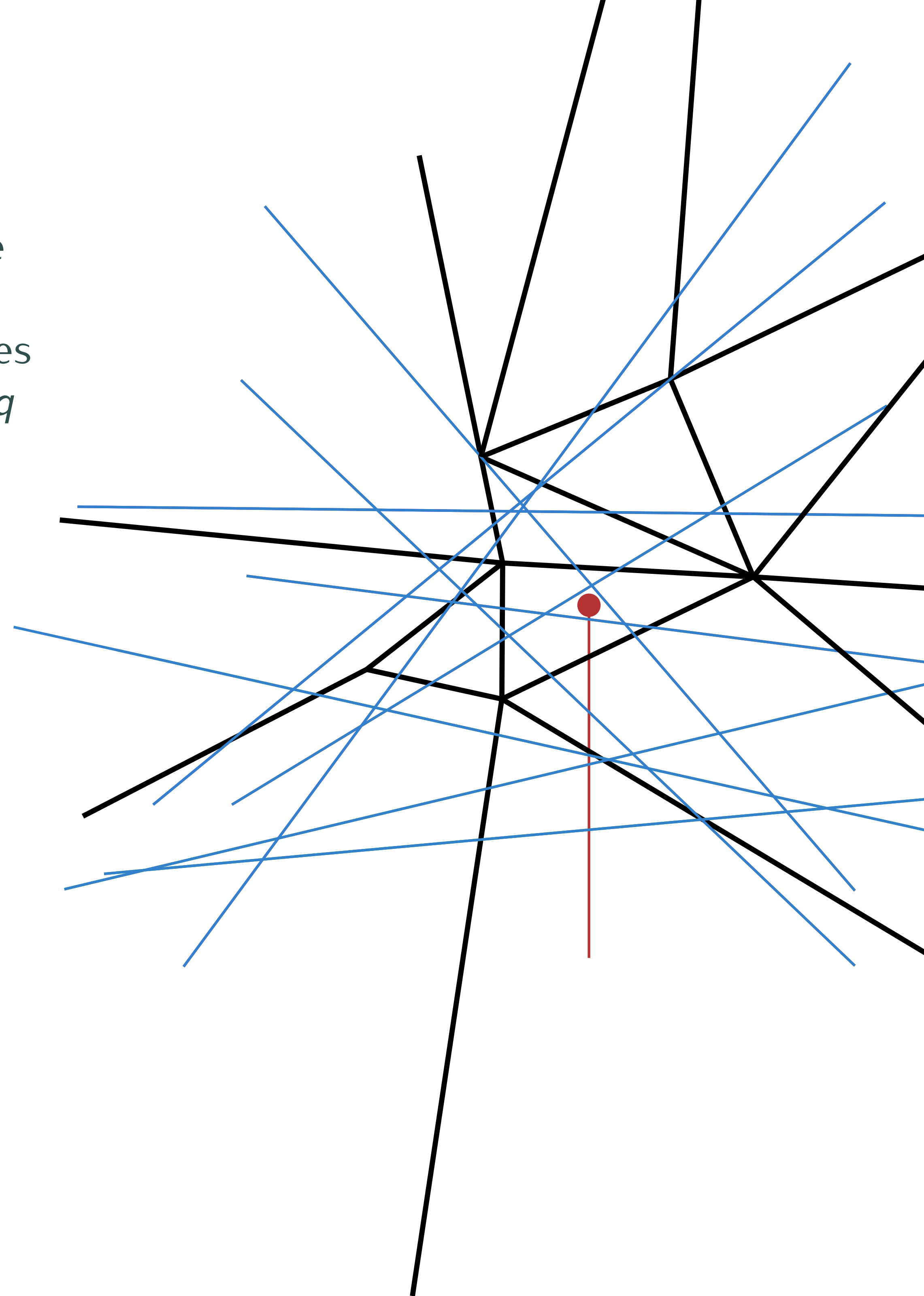
$O(n^2)$ space, $O(\log n)$ query.

Problem. Does not generalize to query triangles: too many possible answers

Cutting Trees

Given a set of n lines L in $\mathbb{R}^2$. Store them in a data structure s.t. we can efficiently report the (number of) lines from L that lie below a query point q

Main idea. partition the plane into disjoint triangles



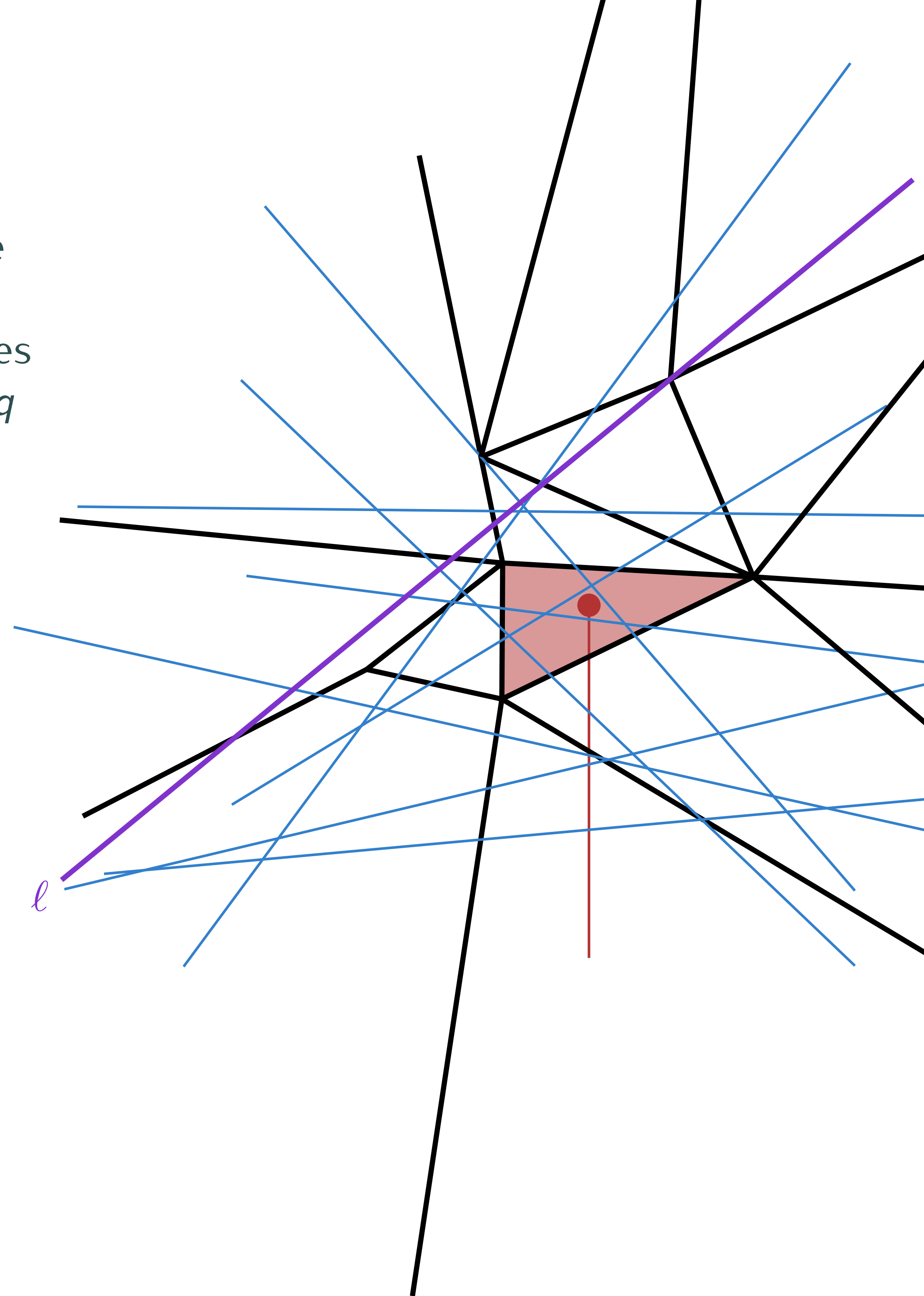
Cutting Trees

Given a set of n lines L in $\mathbb{R}^2$. Store them in a data structure s.t. we can efficiently report the (number of) lines from L that lie below a query point q

Main idea. partition the plane into disjoint triangles

consider such a triangle Δ

ℓ above $\Delta \iff \ell$ is above q for all points $q \in \Delta$



Cutting Trees

Given a set of n lines L in $\mathbb{R}^2$. Store them in a data structure s.t. we can efficiently report the (number of) lines from L that lie below a query point q

Main idea. partition the plane into disjoint triangles

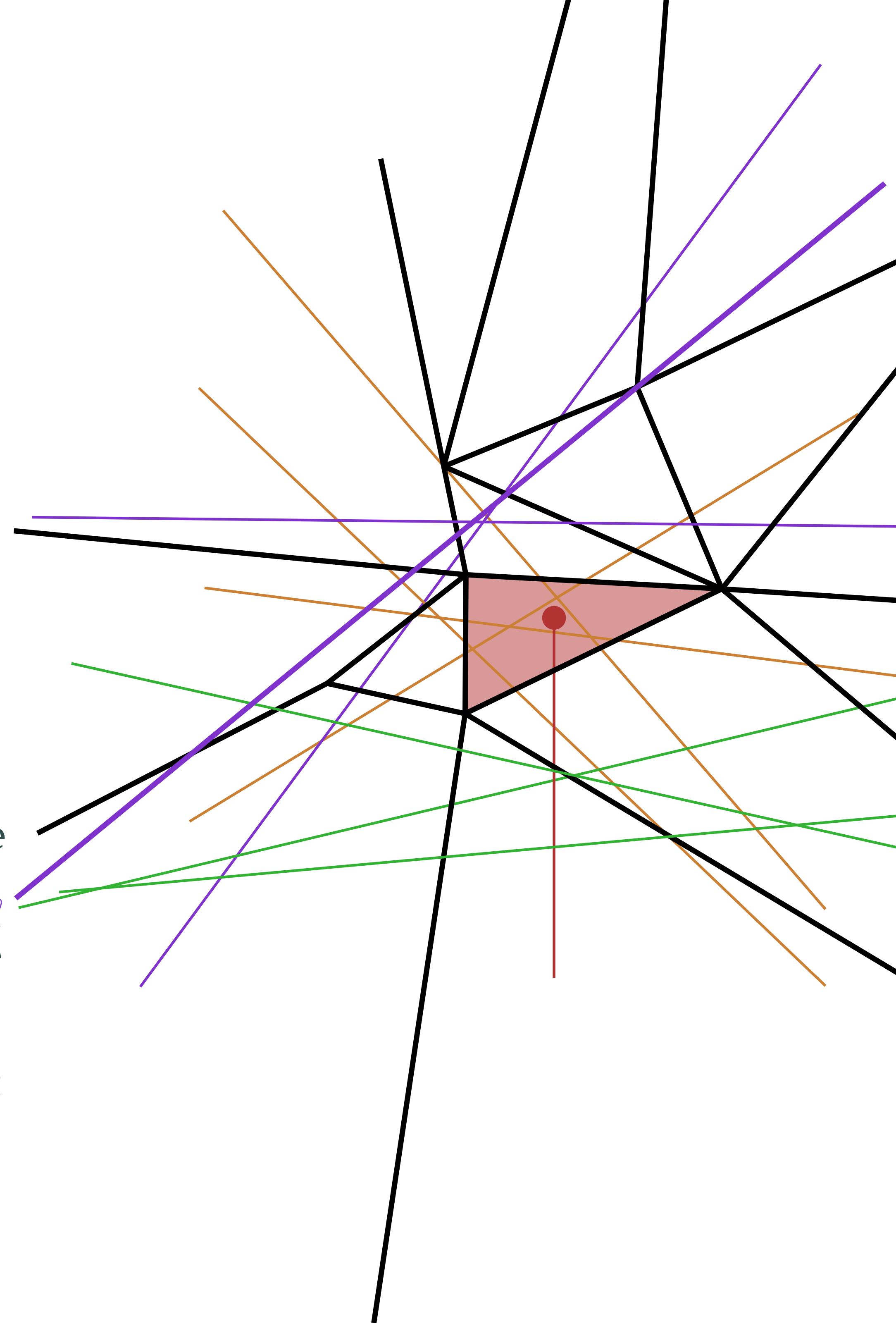
consider such a triangle Δ

ℓ above $\Delta \iff \ell$ is above q for all points $q \in \Delta$

L^+_Δ = upper canonical subset of Δ: the subset of lines that passes above Δ.

L^-_Δ = lower canonical subset of Δ: the subset of lines that passes below Δ.

C_Δ = crossing subset of Δ: the subset of lines that intersect Δ.



Cutting Trees

Given a set of n lines L in $\mathbb{R}^2$. Store them in a data structure s.t. we can efficiently report the (number of) lines from L that lie below a query point q

Main idea. partition the plane into disjoint triangles

consider such a triangle Δ

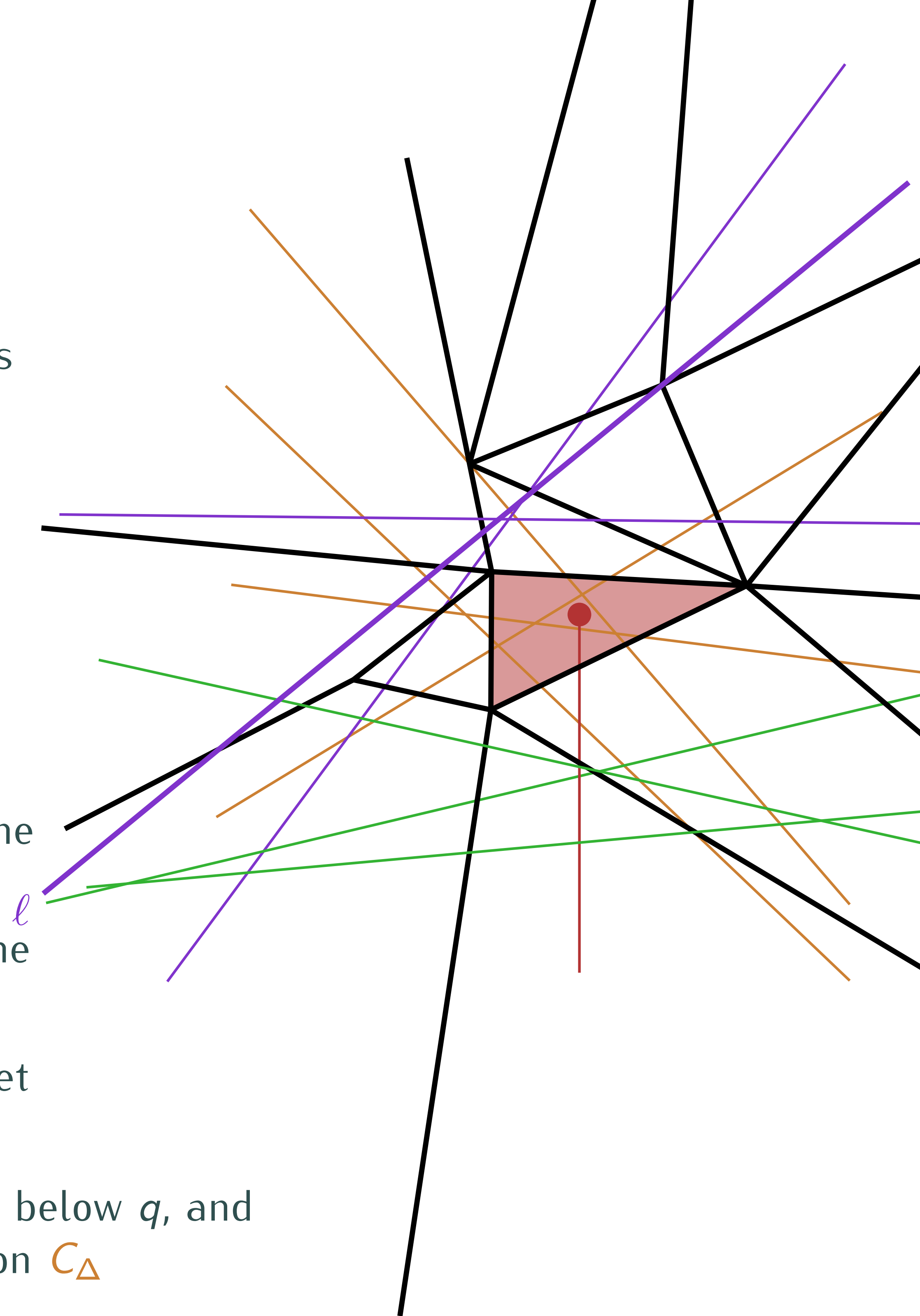
ℓ above $\Delta \iff \ell$ is above q for all points $q \in \Delta$

L^+_Δ = upper canonical subset of Δ: the subset of lines that passes above Δ.

L^-_Δ = lower canonical subset of Δ: the subset of lines that passes below Δ.

C_Δ = crossing subset of Δ: the subset of lines that intersect Δ.

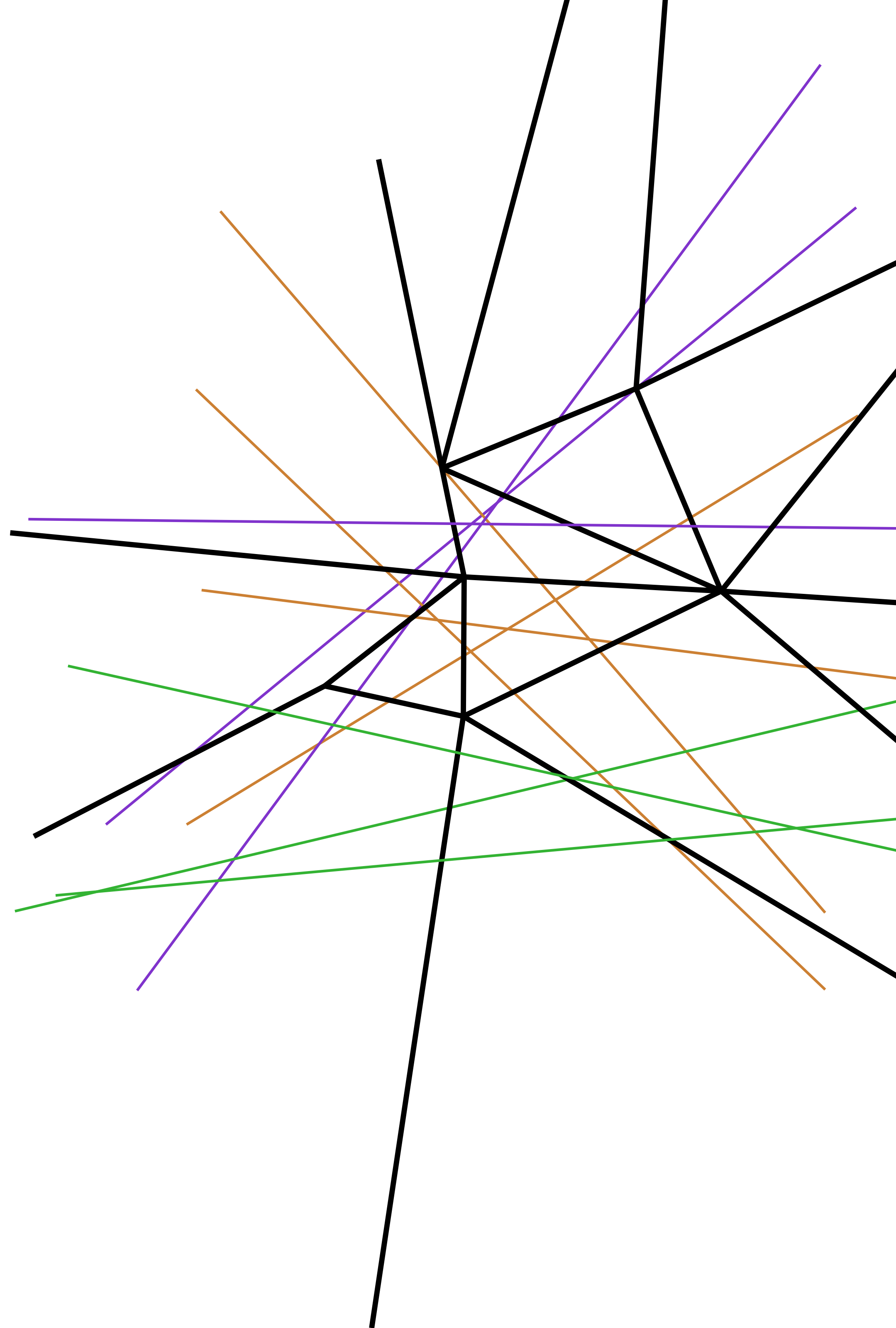
$q \in \Delta \implies$ we have found $|C^-_\Delta|$ lines below q, and we have to recurse only on C_Δ



Cutting Trees

Question. What is a good partition?

1) it should be small (i.e. low complexity)
2) every cell should intersect few lines



Cutting Trees

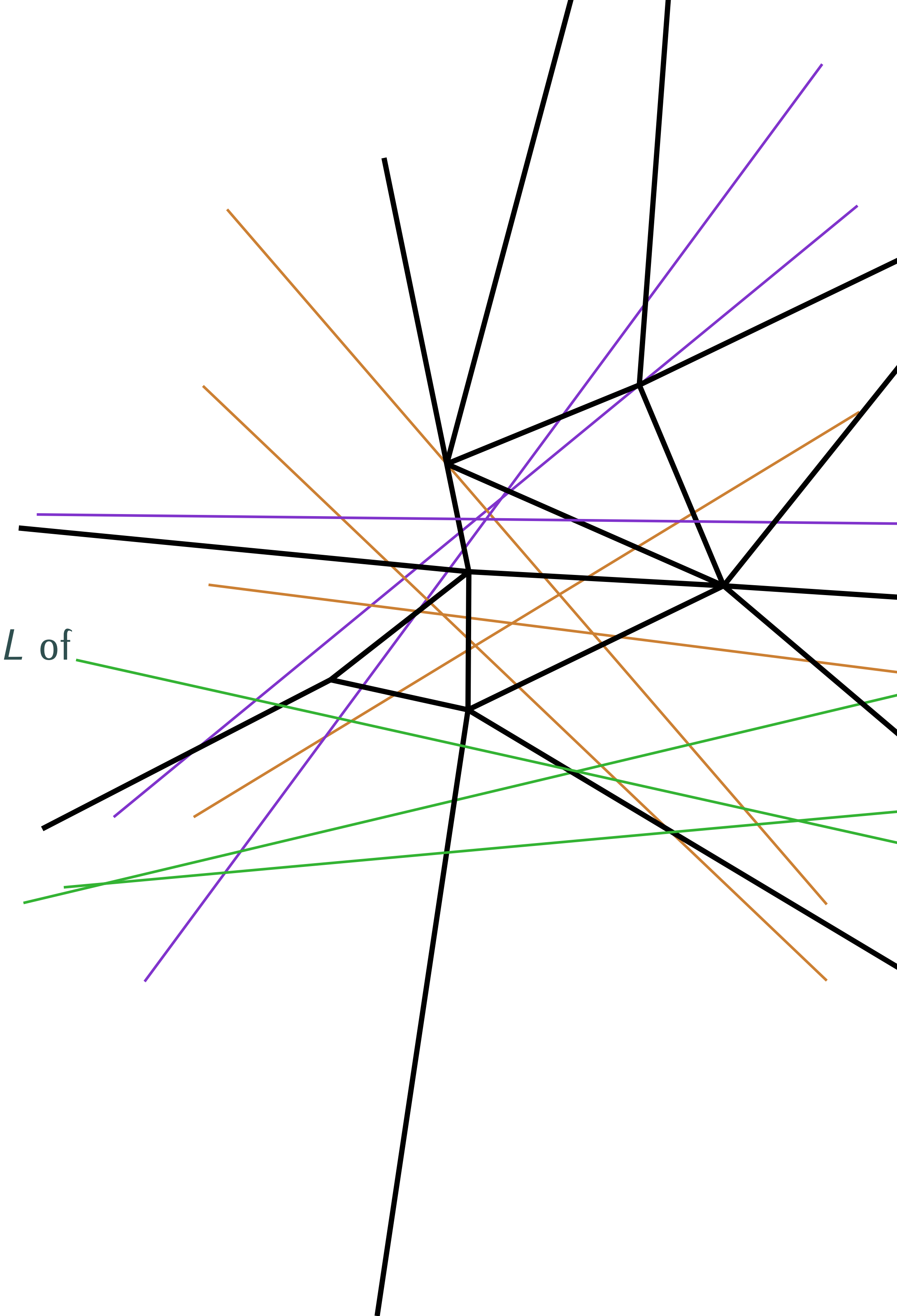
Question. What is a good partition?

1) it should be small (i.e. low complexity)
2) every cell should intersect few lines

L = a set of n lines
r = a parameter in the range $1..n$

$\Lambda(L) = \{\Delta_1, .., \Delta_m\}$ is a $(1/r)$-cutting for L of size m if and only if every triangle Δ_i is intersected by at most n/r lines from L

and all triangles Δ_i are pairwise disjoint



Cutting Trees

Question. What is a good partition?

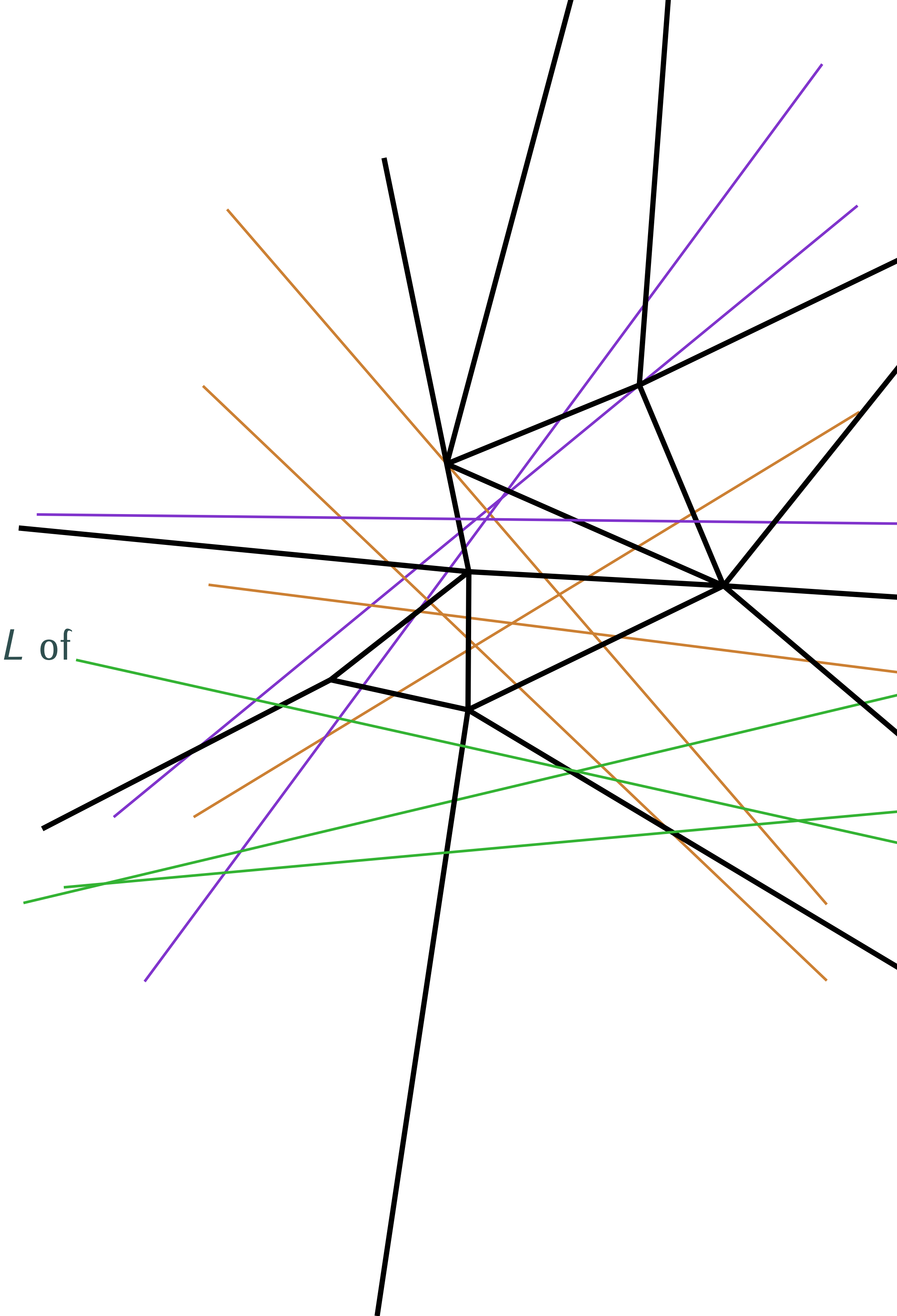
1) it should be small (i.e. low complexity)
2) every cell should intersect few lines

L = a set of n lines
r = a parameter in the range $1..n$

$\Lambda(L) = \{\Delta_1, .., \Delta_m\}$ is a $(1/r)$-cutting for L of size m if and only if every triangle Δ_i is intersected by at most n/r lines from L

and all triangles Δ_i are pairwise disjoint

Thm. For any $r \in [1..n]$ there is a $(1/r)$-cutting of size $O(r^2)$.



Cutting Trees

Question. What is a good partition?

1) it should be small (i.e. low complexity)
2) every cell should intersect few lines

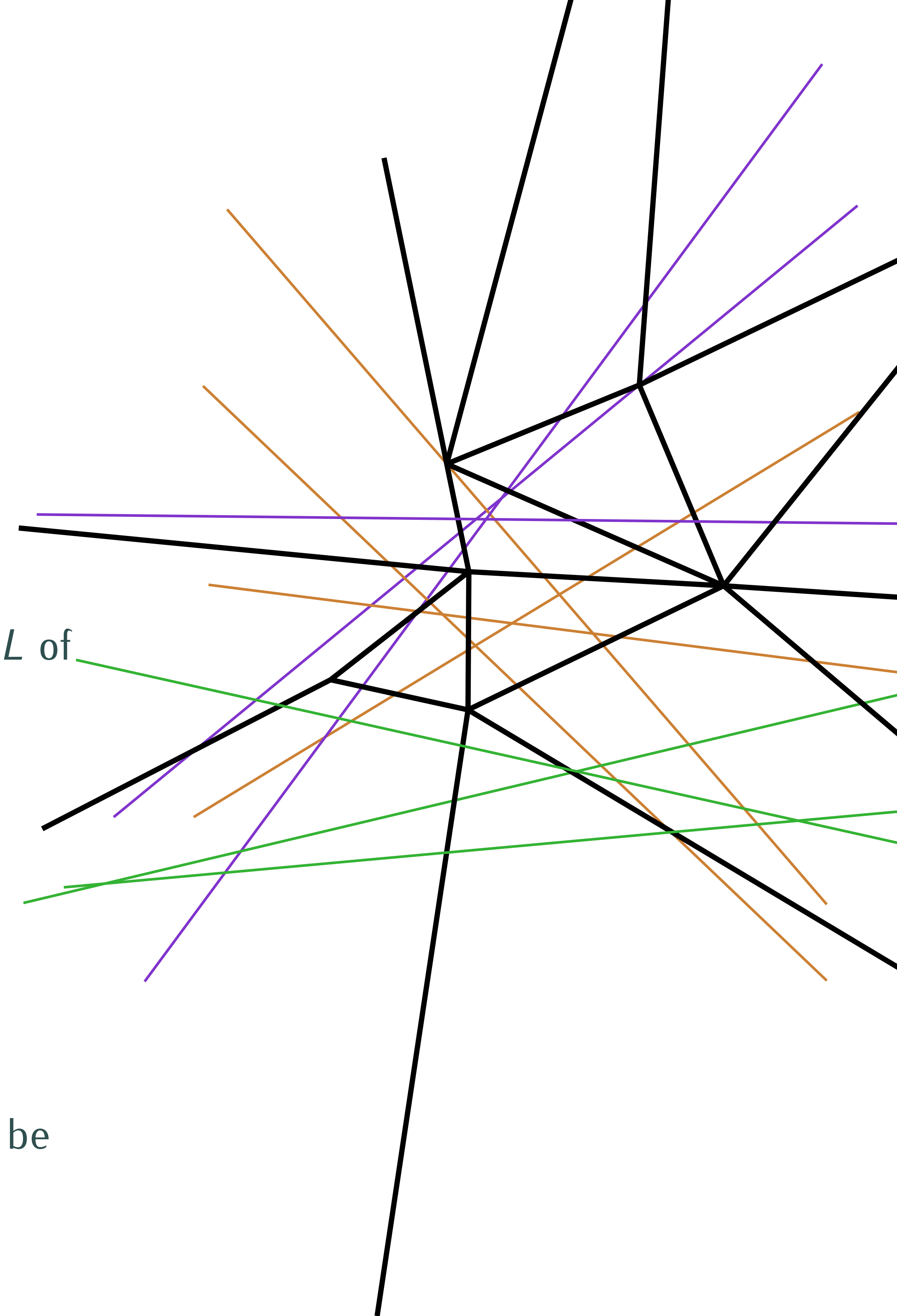
L = a set of n lines
r = a parameter in the range $1..n$

$\Lambda(L) = \{\Delta_1, .., \Delta_m\}$ is a $(1/r)$-cutting for L of size m if and only if every triangle Δ_i is intersected by at most n/r lines from L

and all triangles Δ_i are pairwise disjoint

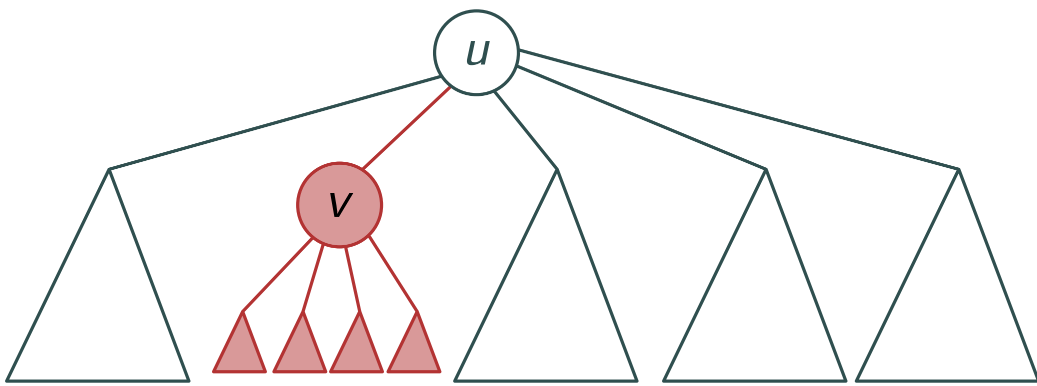
Thm. For any $r \in [1..n]$ there is a $(1/r)$-cutting of size $O(r^2)$.

Moreover, such a cutting (with for each triangle Δ the lines C_Δ that cross it), can be constructed in $O(nr)$ time.



Cutting Trees

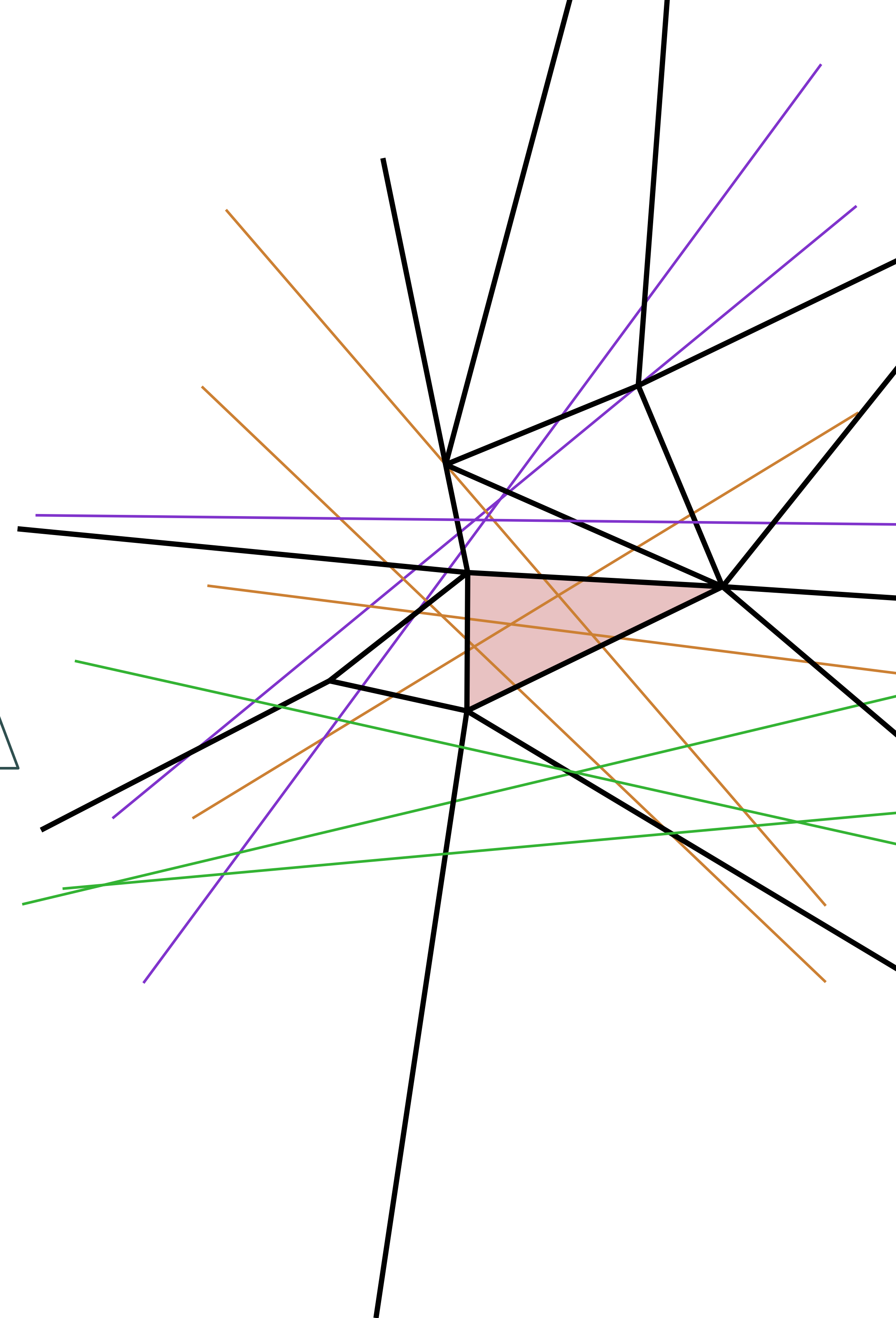
A cutting tree T is a tree with root u that has $O(r^2)$-children: each child $v = v_i$ corresponds to a triangle $\Delta_v = \Delta_i$ of a $(1/r)$-cutting $\Lambda(L)$.



Every node v stores $\Delta = \Delta_v$ and information about $L^+_\Delta = L^+_v$ and $L^-_\Delta = L^-_v$, e.g. their size.

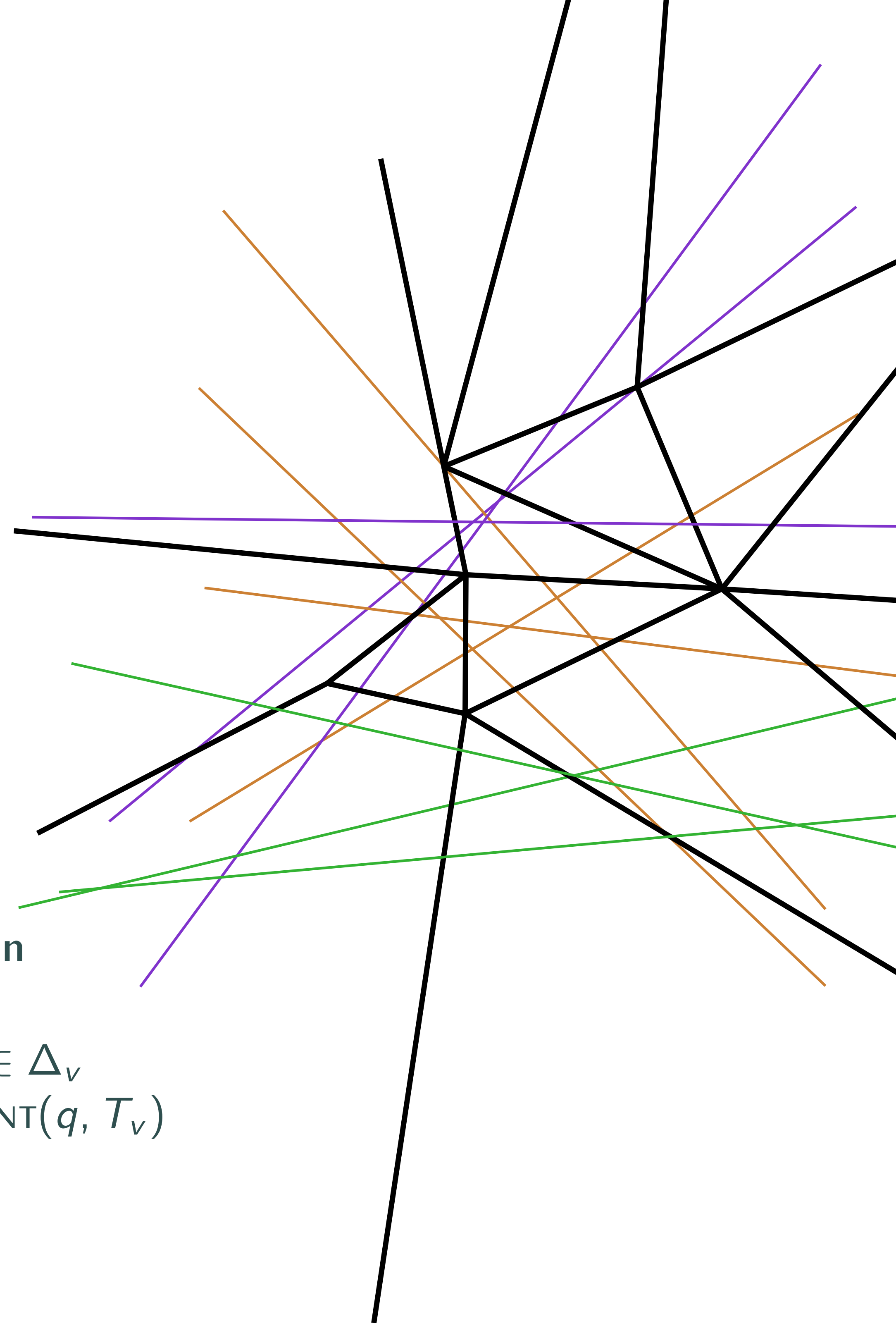
v is the root of a recursilvey defined cutting tree T_v on C_Δ

if $L = \{\ell\}$ then T is a leaf node v, with canonical subset $L_v = P$.



Cutting Trees

Given a query point q, a cutting tree T on L can report a set of nodes V such that the set of lines X below q is the disjoint union of the sets L_v^-, for $v \in V$.



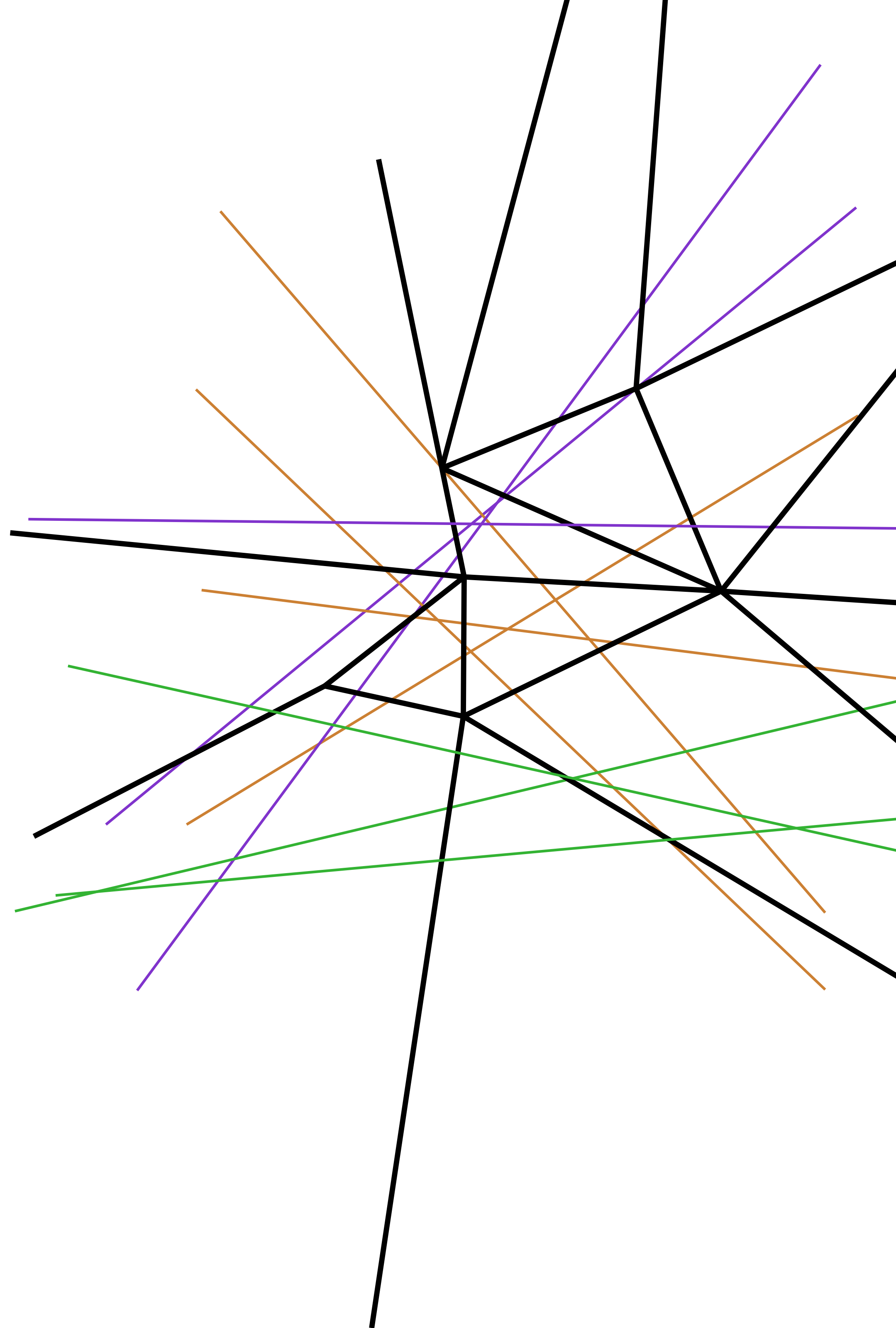
$\textsc{SelectBelowPoint}(q, T)$

1. $V \leftarrow \emptyset$
2. **if** the root u is a leaf node storing ℓ **then**
3. **if** ℓ below q **then** add u to V
4. **else** find the child v of u for which $q \in \Delta_v$
5. $V \leftarrow V \cup \{v\} \cup \textsc{SelectBelowPoint}(q, T_v)$
6. **return** V

Cutting Trees

Given a query point q, a cutting tree T on L can report a set of nodes V such that the set of lines X below q is the disjoint union of the sets L_v^-, for $v \in V$.

Lemma. X is reported as $O(\log n)$ canonical subsets, and we can find them in $O(\log n)$ time.



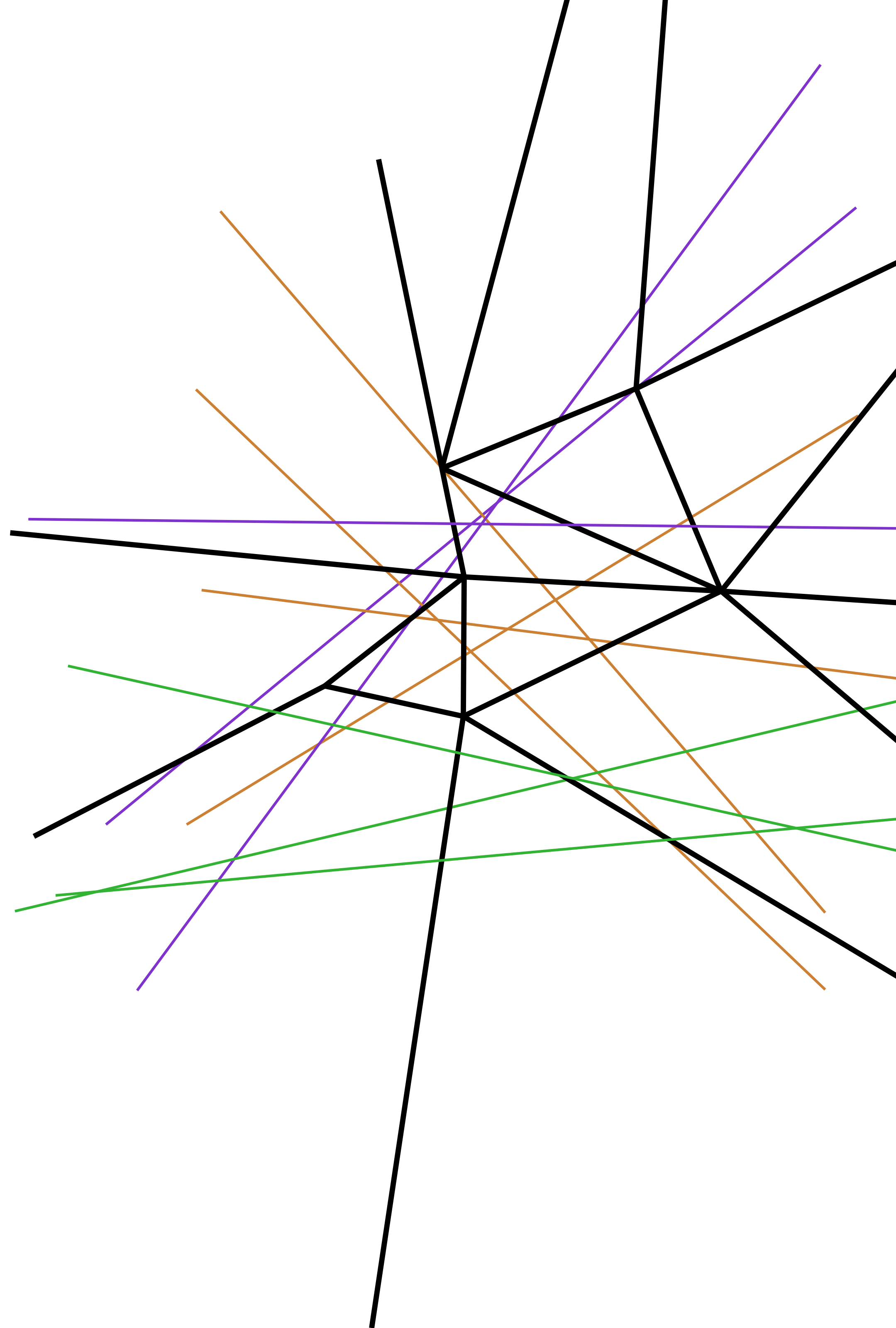
Cutting Trees

Given a query point q, a cutting tree T on L can report a set of nodes V such that the set of lines X below q is the disjoint union of the sets L_v^-, for $v \in V$.

Lemma. X is reported as $O(\log n)$ canonical subsets, and we can find them in $O(\log n)$ time.

Proof. $Q(n)$ = query time

$$Q(n) = \begin{cases} O(1) & \text{if } n = 1 \\ O(r^2) + Q(n/r) & \text{otherwise} \end{cases}$$



Cutting Trees

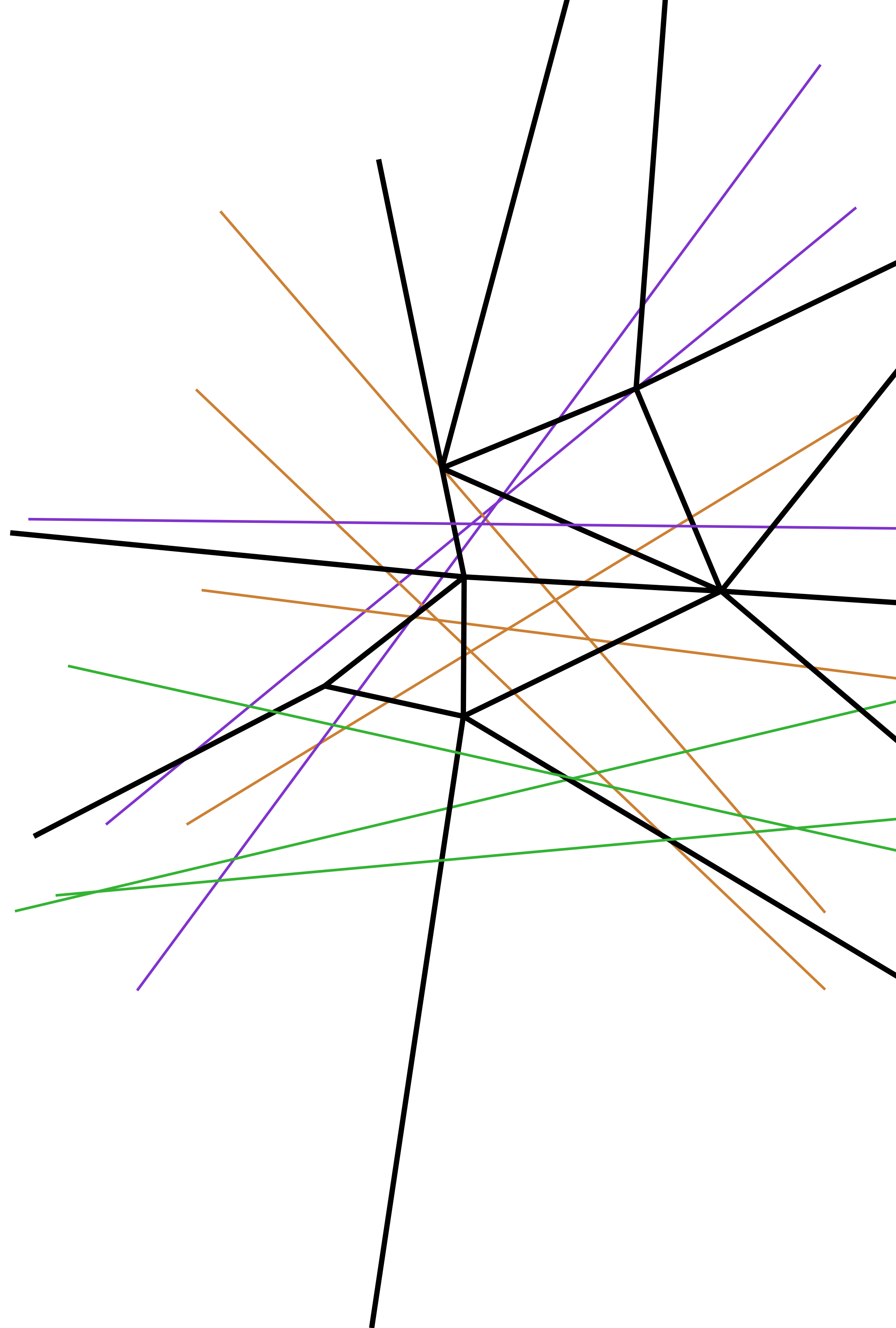
Given a query point q, a cutting tree T on L can report a set of nodes V such that the set of lines X below q is the disjoint union of the sets L_v^-, for $v \in V$.

Lemma. X is reported as $O(\log n)$ canonical subsets, and we can find them in $O(\log n)$ time.

Proof. $Q(n) =$ query time

$$Q(n) = \begin{cases} O(1) & \text{if } n = 1 \\ O(r^2) + Q(n/r) & \text{otherwise} \end{cases}$$

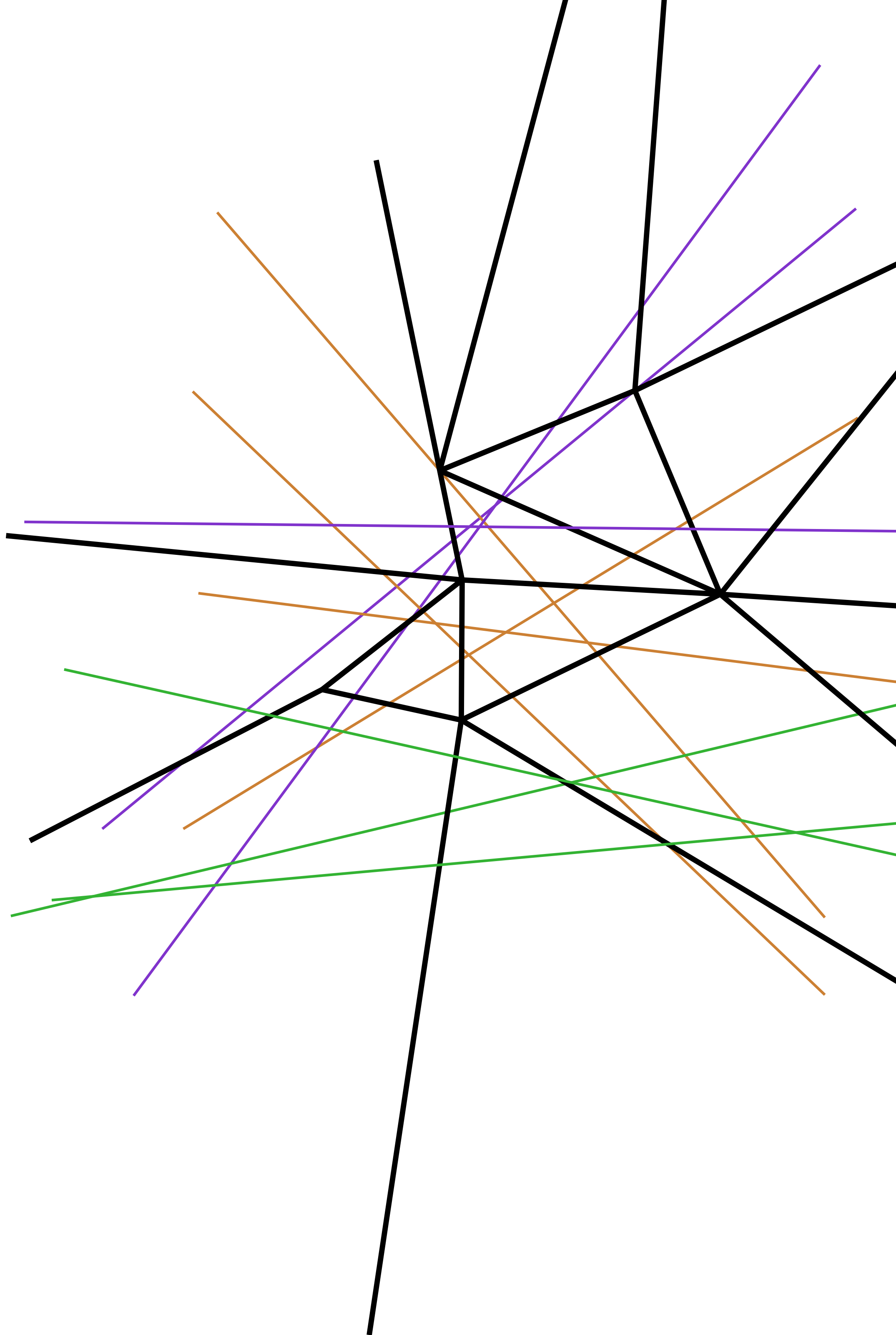
for any $r > 1$ this solves to $O(\log n)$.



Cutting Trees

Given a query point q, a cutting tree T on L can report a set of nodes V such that the set of lines X below q is the disjoint union of the sets L_v^-, for $v \in V$.

Lemma. T uses $O(n^{2+\varepsilon})$ space



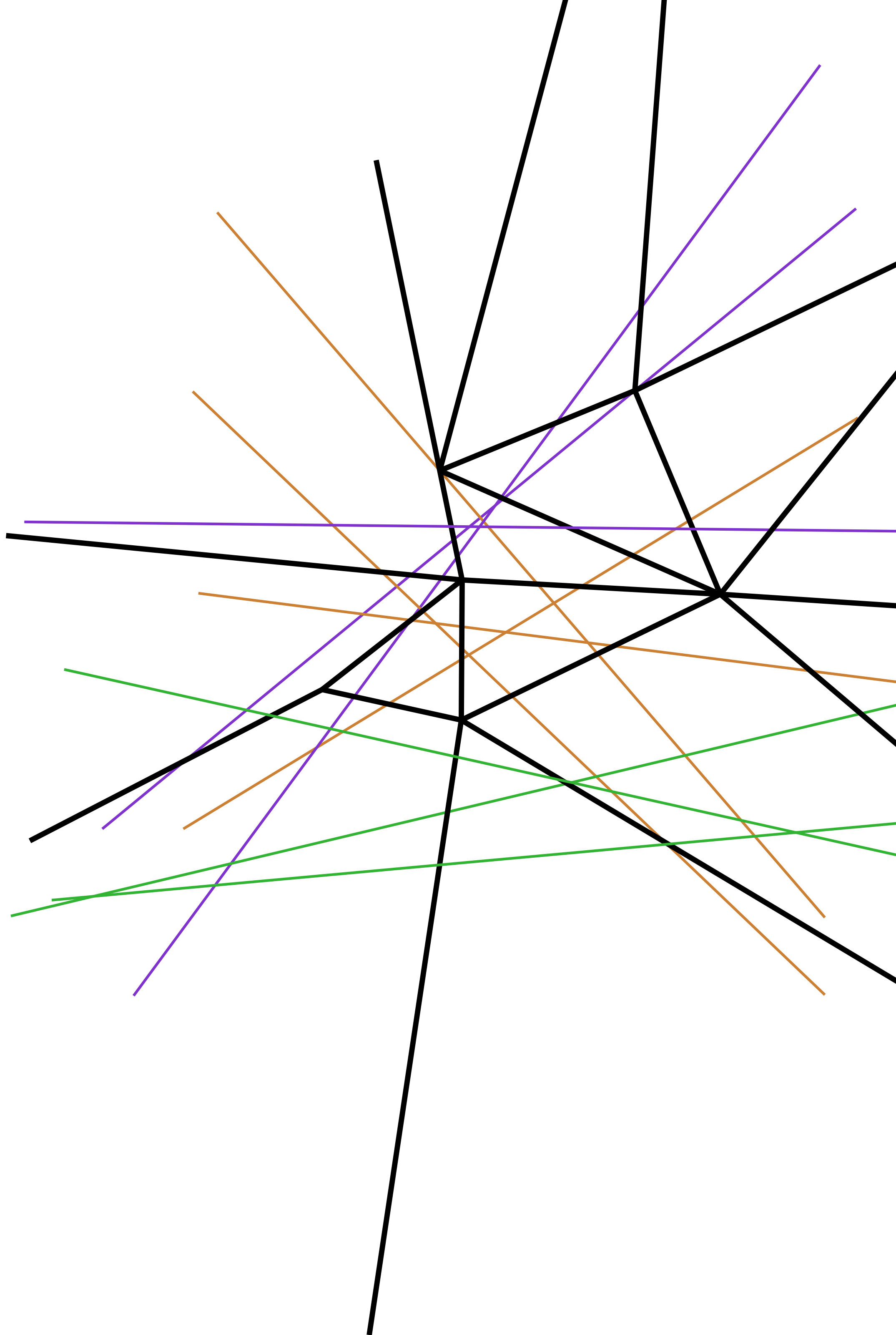
Cutting Trees

Given a query point q, a cutting tree T on L can report a set of nodes V such that the set of lines X below q is the disjoint union of the sets L_v^-, for $v \in V$.

Lemma. T uses $O(n^{2+\varepsilon})$ space

Proof. By Thm. we can construct a $(1/r)$-cutting of size at most cr^2, for some constant c.

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



Cutting Trees

Given a query point q, a cutting tree T on L can report a set of nodes V such that the set of lines X below q is the disjoint union of the sets L_v^-, for $v \in V$.

Lemma. T uses $O(n^{2+\varepsilon})$ space

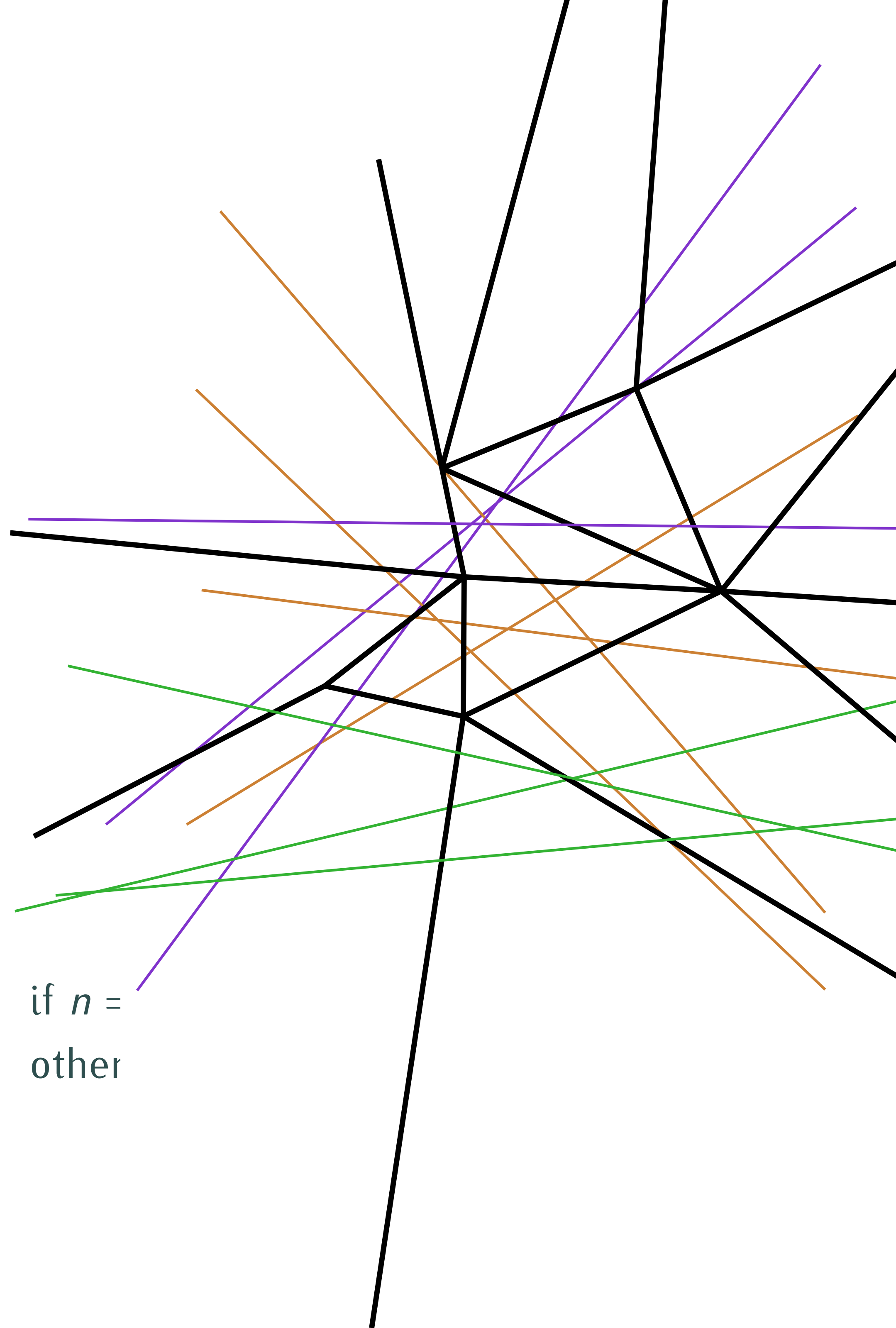
Proof. By Thm. we can construct a $(1/r)$-cutting of size at most cr^2, for some constant c.

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

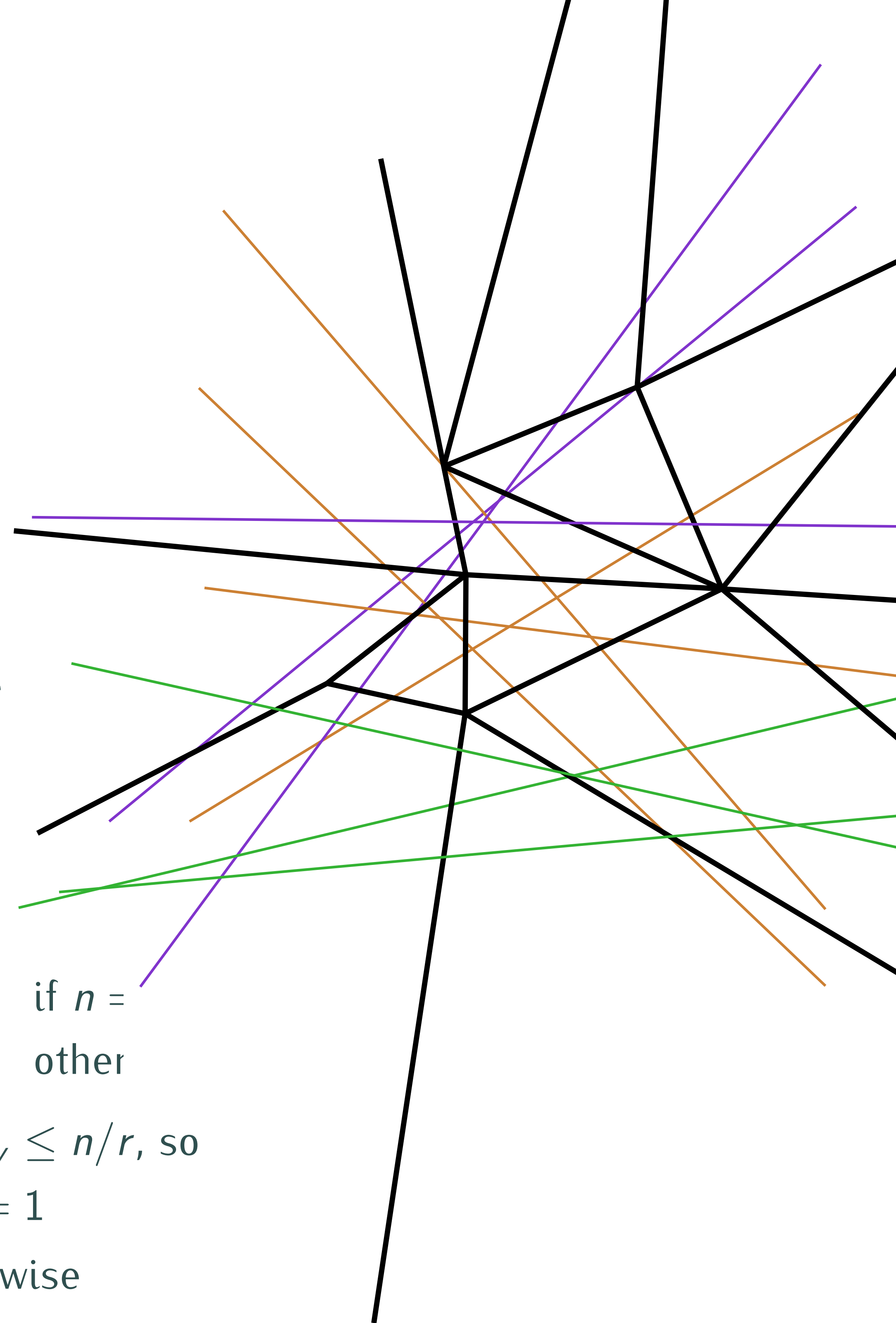
$M(n) =$ space usage

$n_v =$ number of lines in C_v

$$M(n) = \begin{cases} O(1) & \text{if } n = \\ O(r^2) + \sum_{v \text{ child of the root}} M(n_v) & \text{other} \end{cases}$$



Cutting Trees



Given a query point q, a cutting tree T on L can report a set of nodes V such that the set of lines X below q is the disjoint union of the sets L_v^-, for $v \in V$.

Lemma. T uses $O(n^{2+\varepsilon})$ space

Proof. By Thm. we can construct a $(1/r)$-cutting of size at most cr^2, for some constant c.

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

$M(n)$ = space usage

n_v = number of lines in C_v

$$M(n) = \begin{cases} O(1) & \text{if } n = \\ O(r^2) + \sum_{v \text{ child of the root}} M(n_v) & \text{other} \end{cases}$$

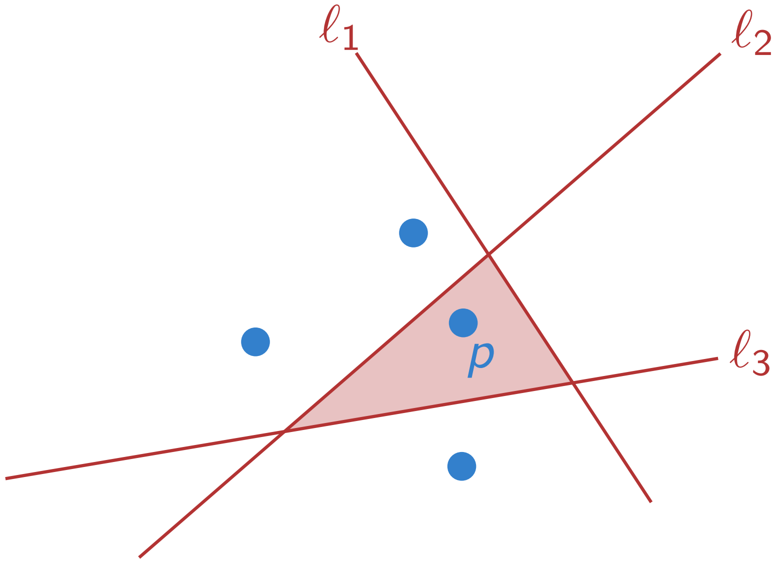
v has at most cr^2 children, each of size $n_v \leq n/r$, so

$$M(n) \leq \begin{cases} O(1) & \text{if } n = 1 \\ O(r^2) + \sum_{i=1}^{cr^2} M(n/r) & \text{otherwise} \end{cases}$$

Cutting Trees

We wanted to count (report) all points in a triangle Q.

Primal

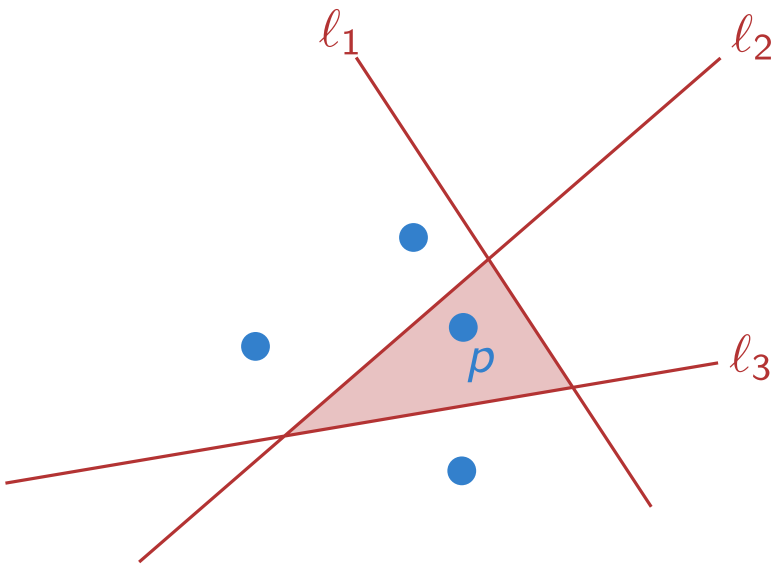


report p $\Longleftrightarrow$

Cutting Trees

We wanted to count (report) all points in a triangle Q.

Primal



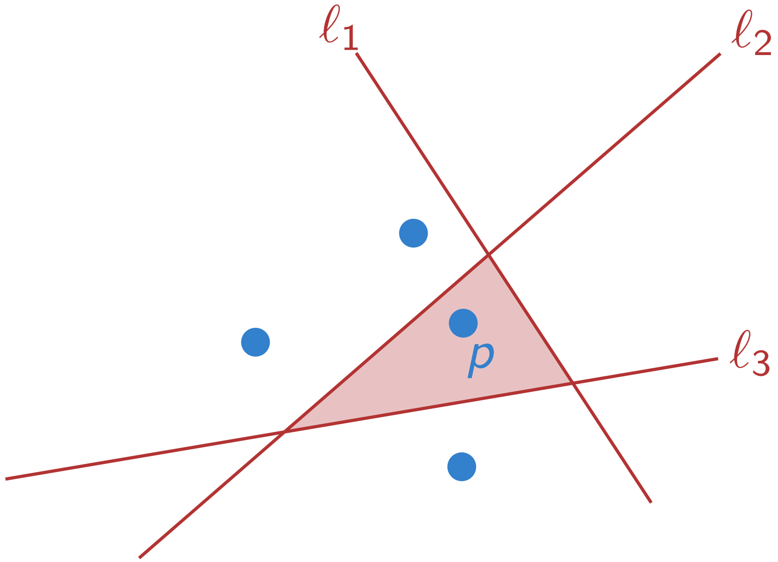
report $p \iff$ p lies below ℓ_1, and below ℓ_2, and above ℓ_3

Cutting Trees

We wanted to count (report) all points in a triangle Q.

Question. What does Q correspond to in the dual space?

Primal



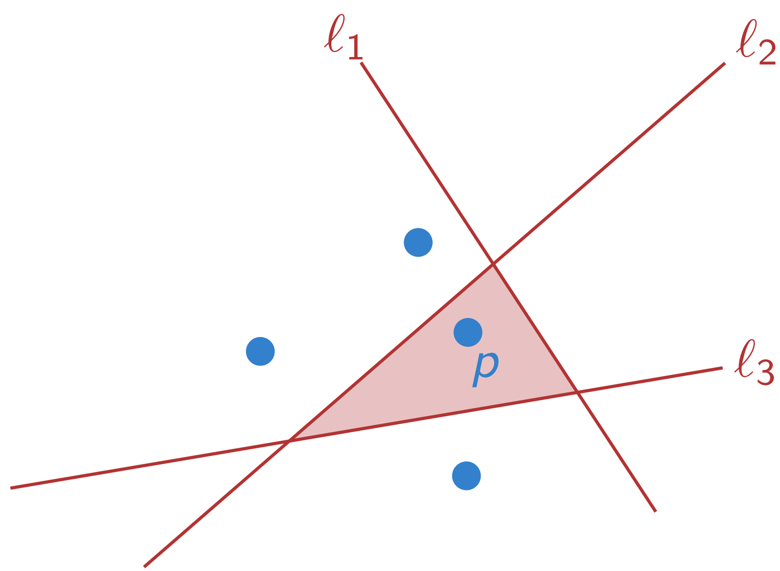
report $p \Longleftrightarrow$ p lies below ℓ_1, and below ℓ_2, and above ℓ_3

Cutting Trees

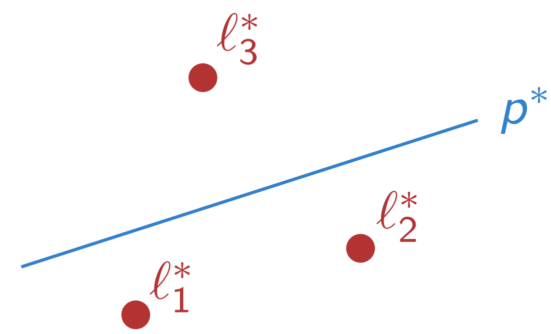
We wanted to count (report) all points in a triangle Q.

Question. What does Q correspond to in the dual space?

Primal



Dual



report $p \iff$ p lies below ℓ_1, and below ℓ_2, and above ℓ_3

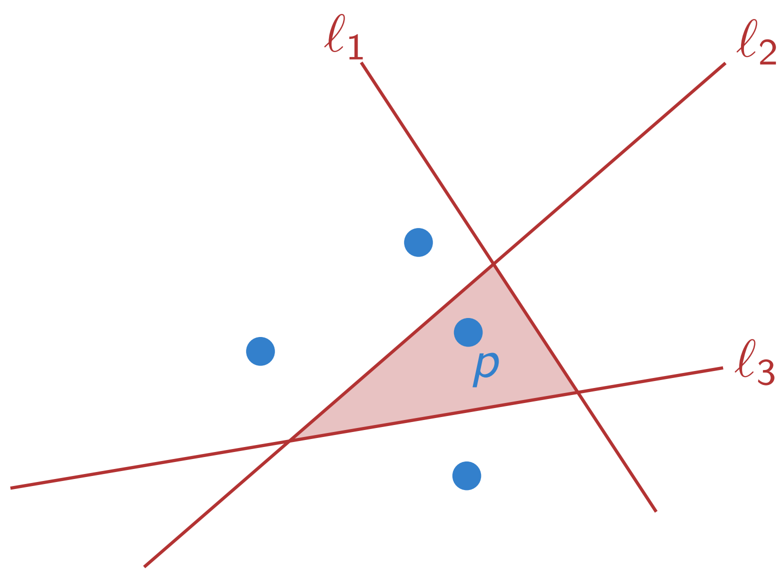
p^* lies above ℓ_1^*, and above ℓ_2^*, and below ℓ_3^*

Cutting Trees

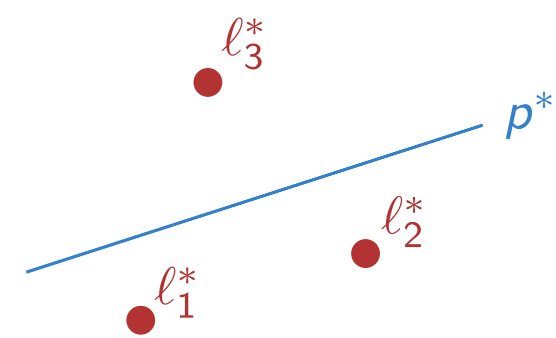
We wanted to count (report) all points in a triangle Q.

Question. What does Q correspond to in the dual space?

Primal



Dual



report $p \iff$

$p^* \in L'''$

p lies below ℓ_1, and
below ℓ_2, and
above ℓ_3

p^* lies above ℓ_1^*, and
above ℓ_2^*, and
below ℓ_3^*

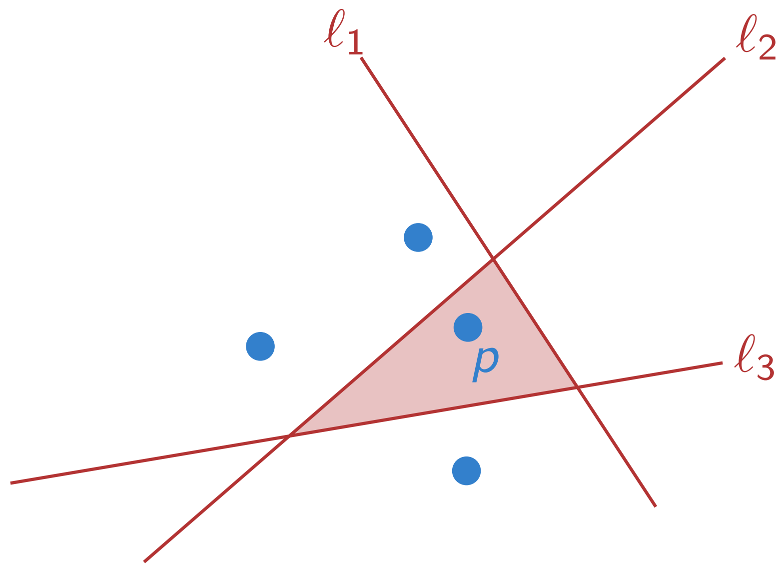
Let $L' \subseteq P^*$ of lines that lie above ℓ_1^*
Let $L'' \subseteq L'$ of lines that lie above ℓ_2^*
Let $L''' \subseteq L''$ of lines that lie below ℓ_3^*

Cutting Trees

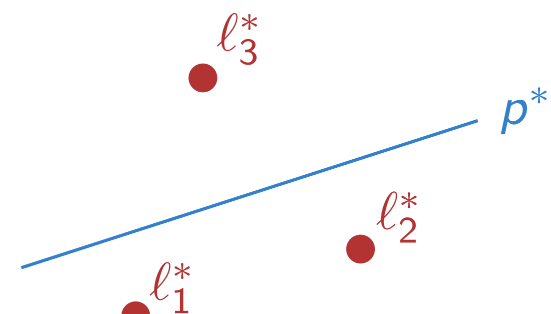
We wanted to count (report) all points in a triangle Q.

Question. What does Q correspond to in the dual space?

Primal



Dual



report $p \iff$

$p^* \in L'''$

p lies below ℓ_1, and
below ℓ_2, and
above ℓ_3

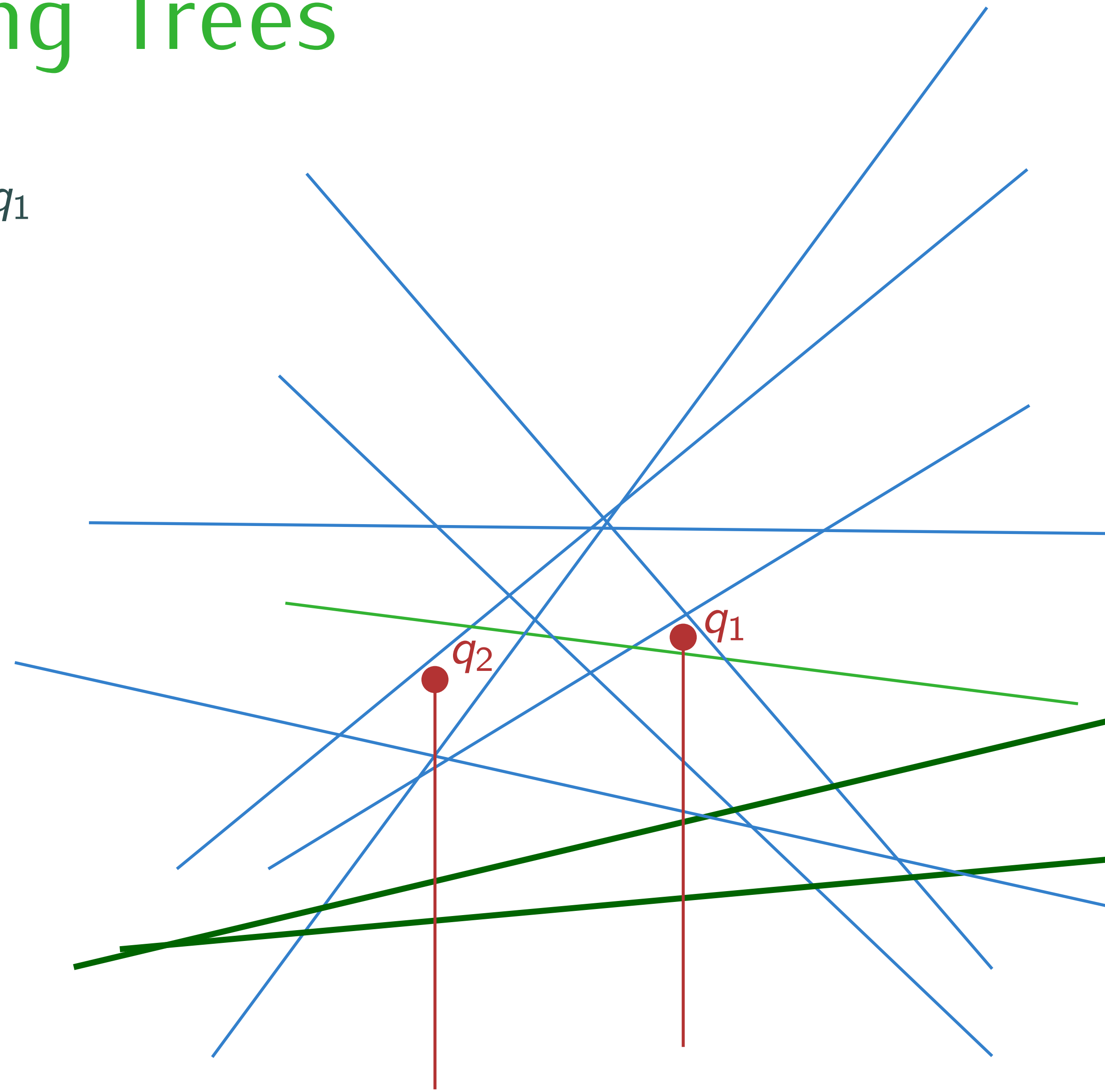
p^* lies above ℓ_1^*, and
above ℓ_2^*, and
below ℓ_3^*

Let $L' \subseteq P^*$ of lines that lie above ℓ_1^*
Let $L'' \subseteq L'$ of lines that lie above ℓ_2^*
Let $L''' \subseteq L''$ of lines that lie below ℓ_3^*

use a multilevel cutting tree.

Multilevel Cutting Trees

Count all lines from L that lie below q_1 and below q_2

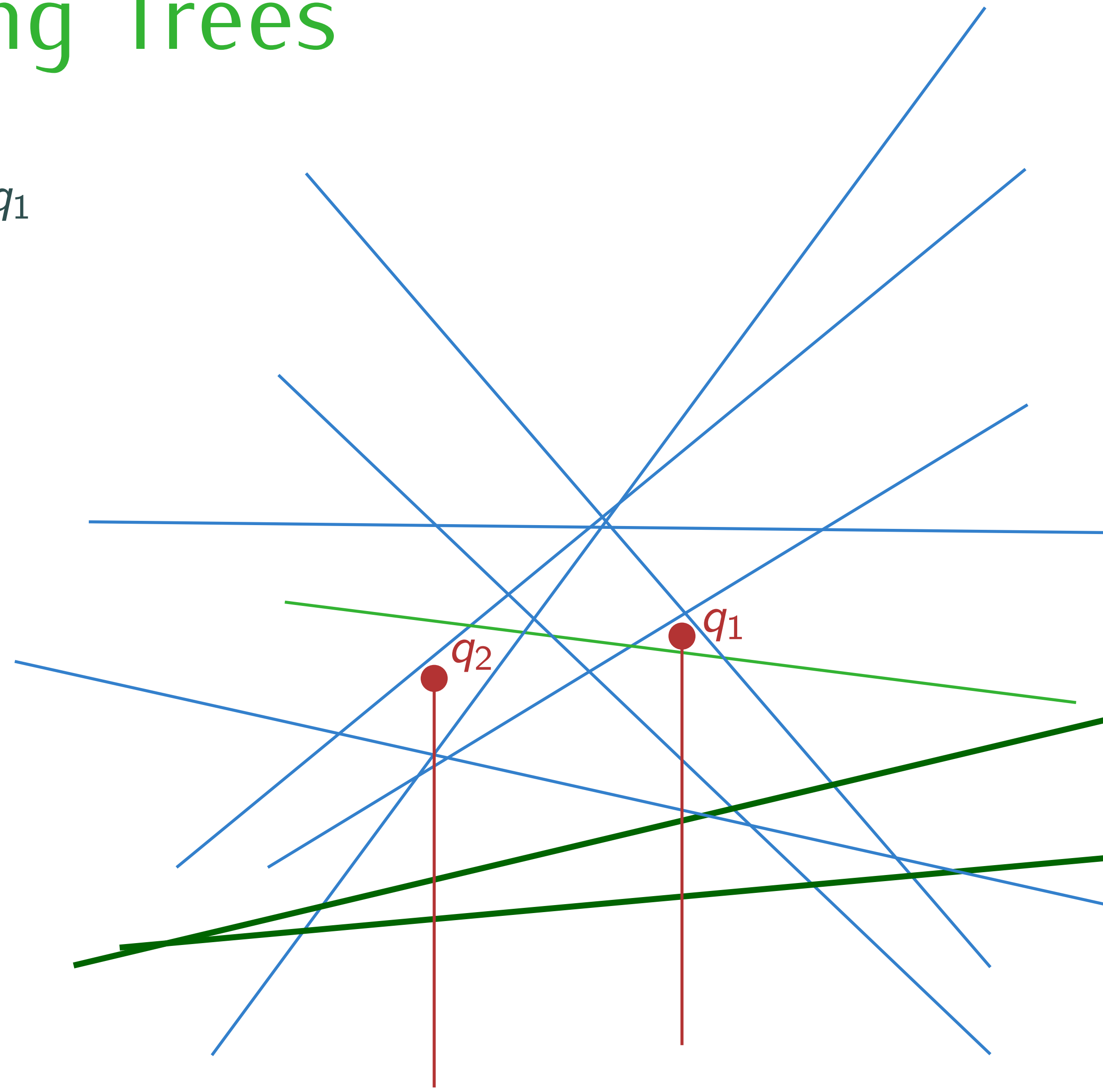


Multilevel Cutting Trees

Count all lines from L that lie below q_1 and below q_2

For every node v of the main cutting tree T:

store L_v^- in a cutting tree T_v^{assoc}



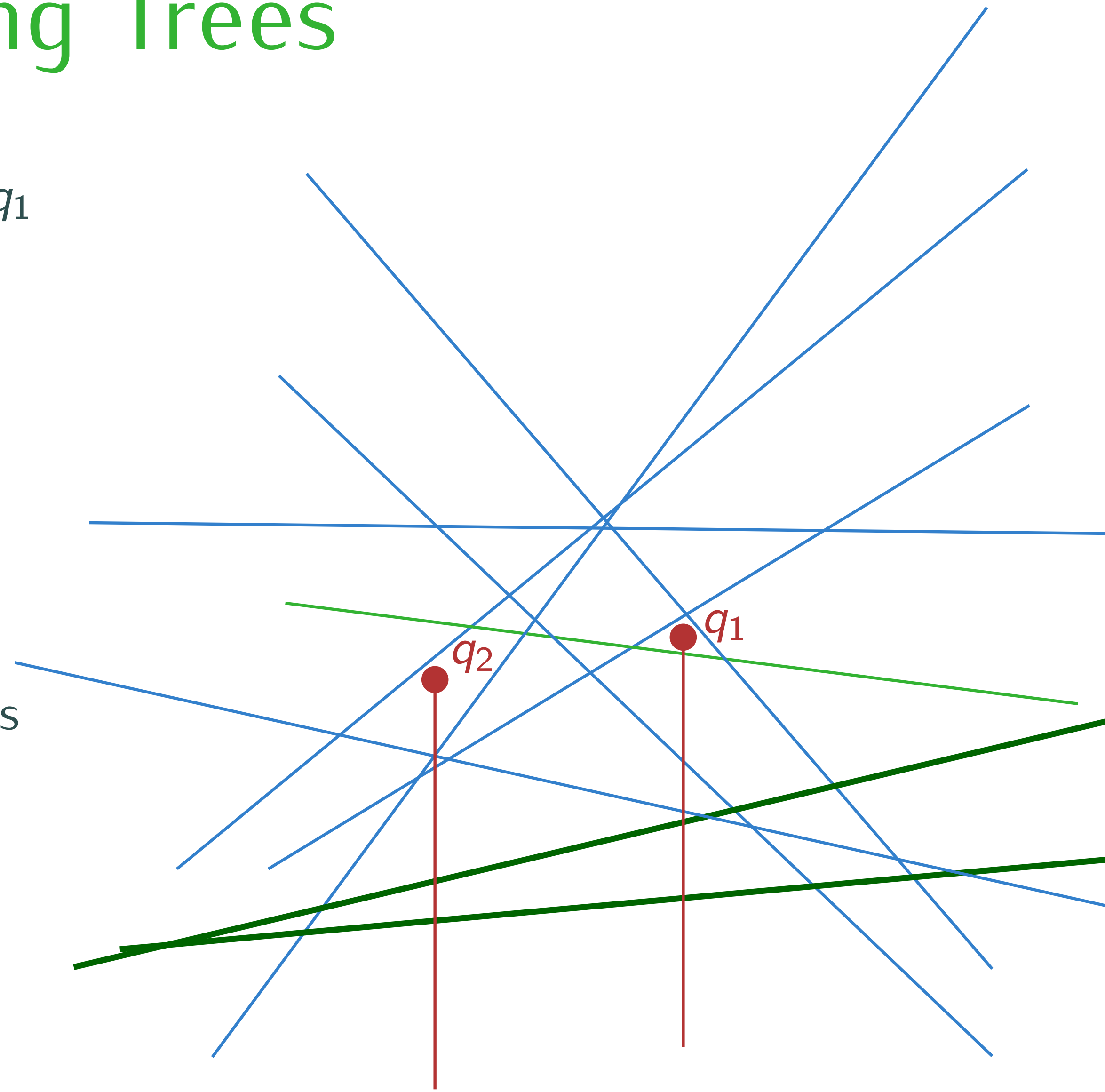
Multilevel Cutting Trees

Count all lines from L that lie below q_1 and below q_2

For every node v of the main cutting tree T:

store L_v^- in a cutting tree T_v^{assoc}

Lemma. A two-level cutting tree uses $O(n^{2+\varepsilon})$ space, and can count all lines below query points q_1 and q_2 in $O(\log^2 n)$ time.



Multilevel Cutting Trees

Count all lines from L that lie below q_1 and below q_2

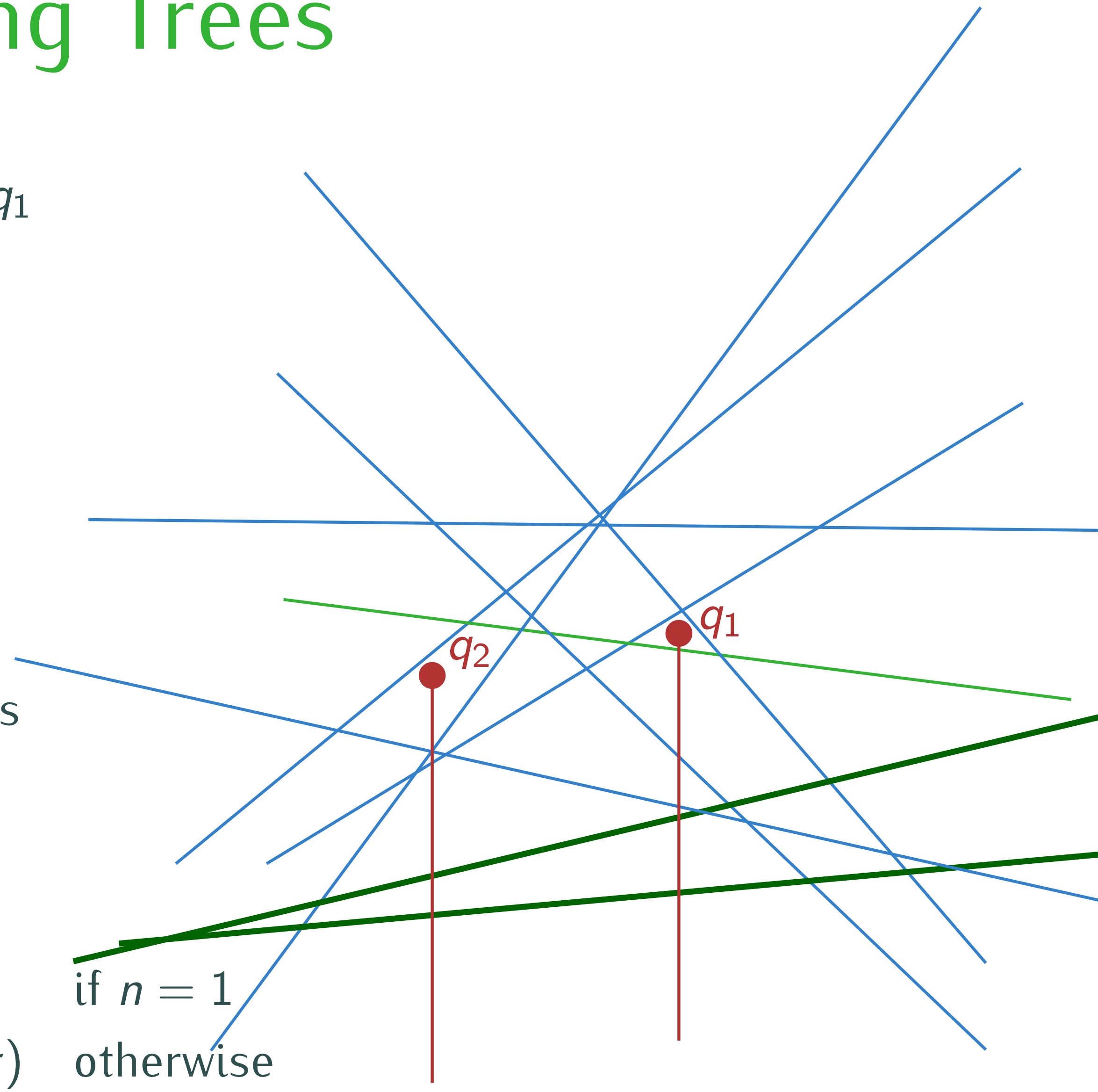
For every node v of the main cutting tree T:

store L_v^- in a cutting tree T_v^{assoc}

Lemma. A two-level cutting tree uses $O(n^{2+\varepsilon})$ space, and can count all lines below query points q_1 and q_2 in $O(\log^2 n)$ time.

$Q(n)$ = query time

$$Q(n) = \begin{cases} O(1) & \text{if } n = 1 \\ O(r^2) + O(\log n) + Q(n/r) & \text{otherwise} \end{cases}$$



Multilevel Cutting Trees

Count all lines from L that lie below q_1 and below q_2

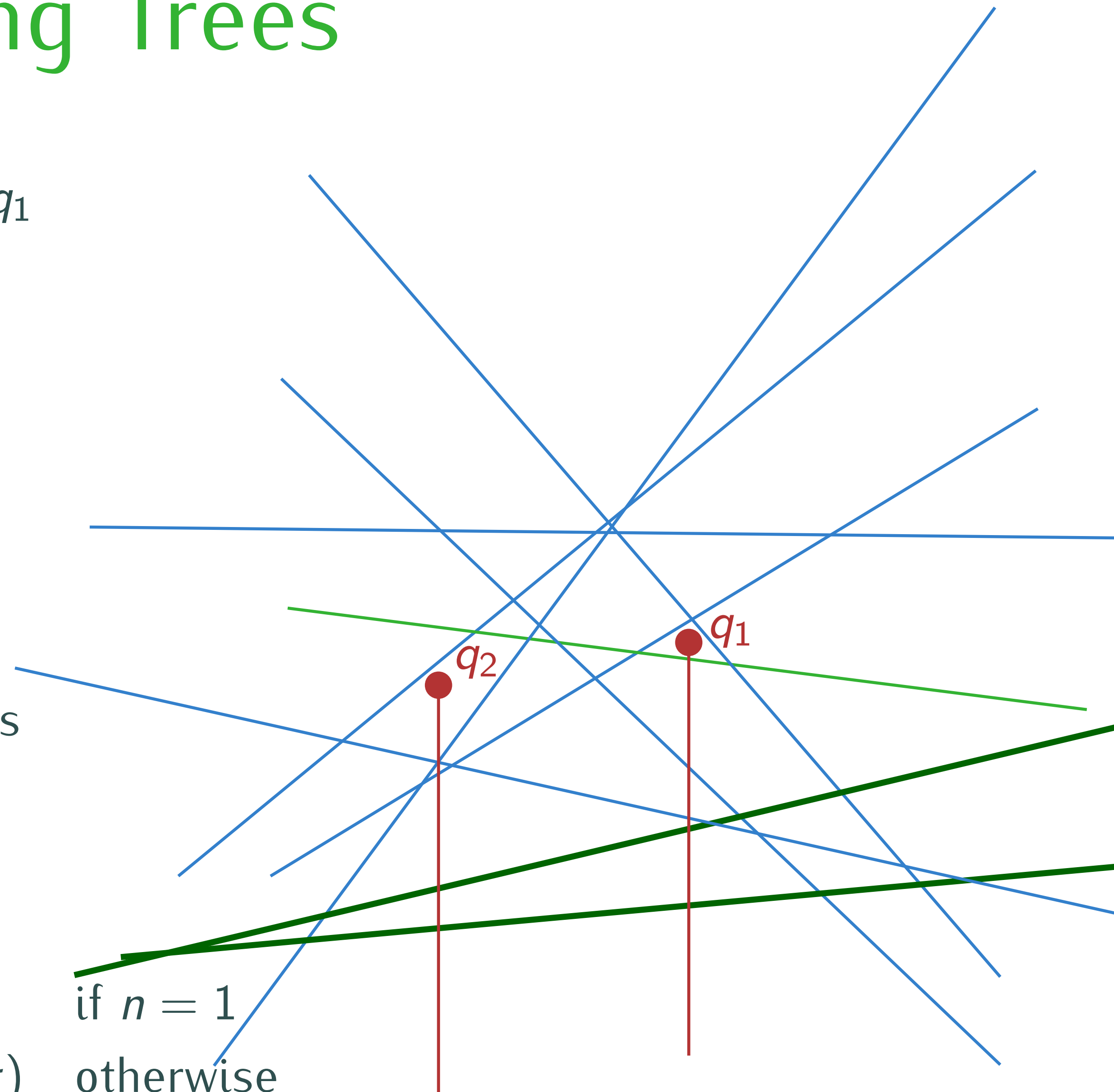
For every node v of the main cutting tree T:

store L_v^- in a cutting tree T_v^{assoc}

Lemma. A two-level cutting tree uses $O(n^{2+\varepsilon})$ space, and can count all lines below query points q_1 and q_2 in $O(\log^2 n)$ time.

$Q(n)$ = query time

$$Q(n) = \begin{cases} O(1) & \text{if } n = 1 \\ O(r^2) + O(\log n) + Q(n/r) & \text{otherwise} \end{cases}$$



$M(n)$ = space usage

$$M(n) = \begin{cases} O(1) & \text{if } n = 1 \\ \sum_{v \text{ child of the root}} (O(n^{2+\varepsilon}) + M(n_v)) & \text{otherwise} \end{cases}$$

Multilevel Cutting Trees

Count all lines from L that lie below q_1 and below q_2

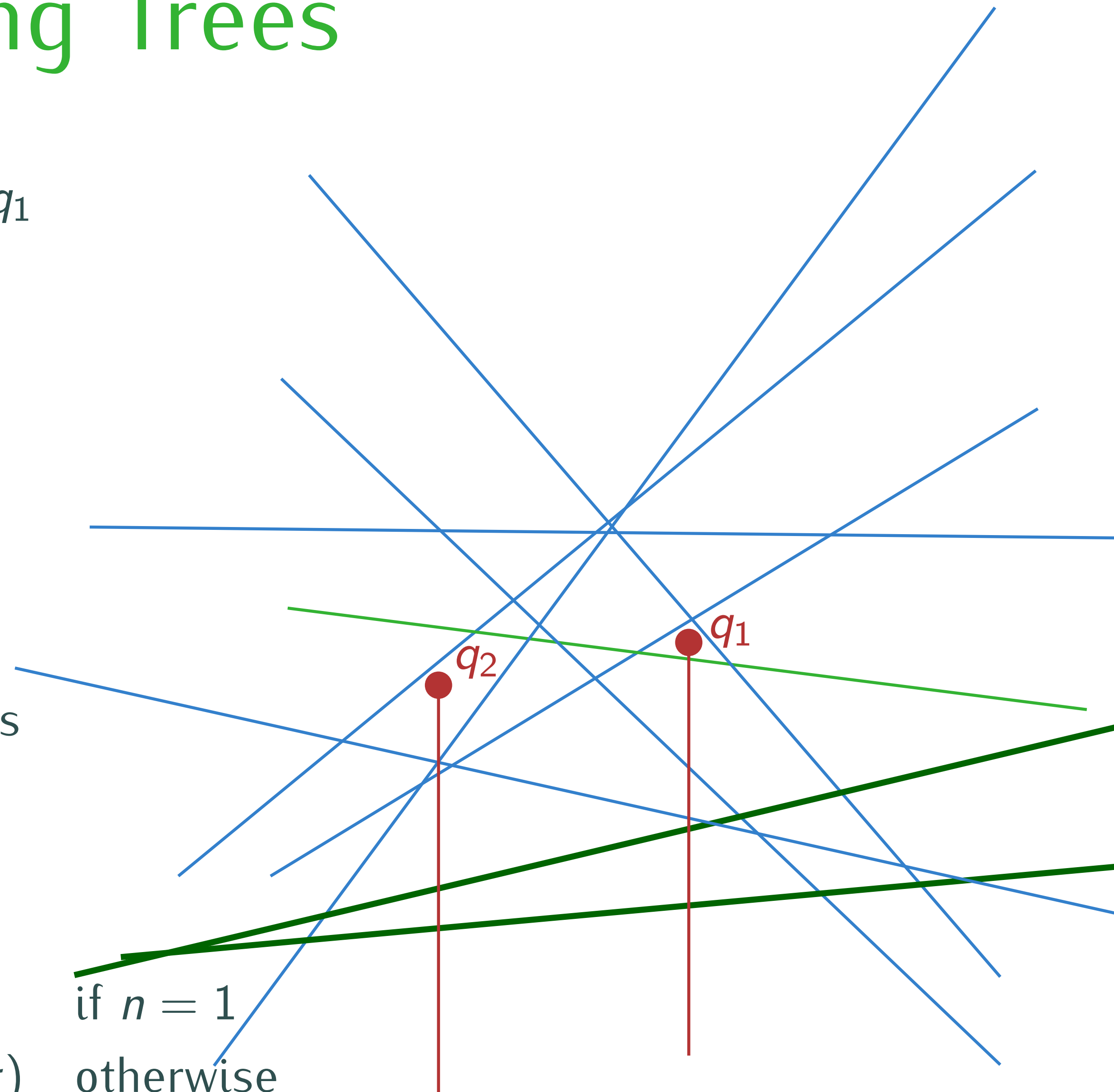
For every node v of the main cutting tree T:

store L_v^- in a cutting tree T_v^{assoc}

Lemma. A two-level cutting tree uses $O(n^{2+\varepsilon})$ space, and can count all lines below query points q_1 and q_2 in $O(\log^2 n)$ time.

$Q(n)$ = query time

$$Q(n) = \begin{cases} O(1) & \text{if } n = 1 \\ O(r^2) + O(\log n) + Q(n/r) & \text{otherwise} \end{cases}$$



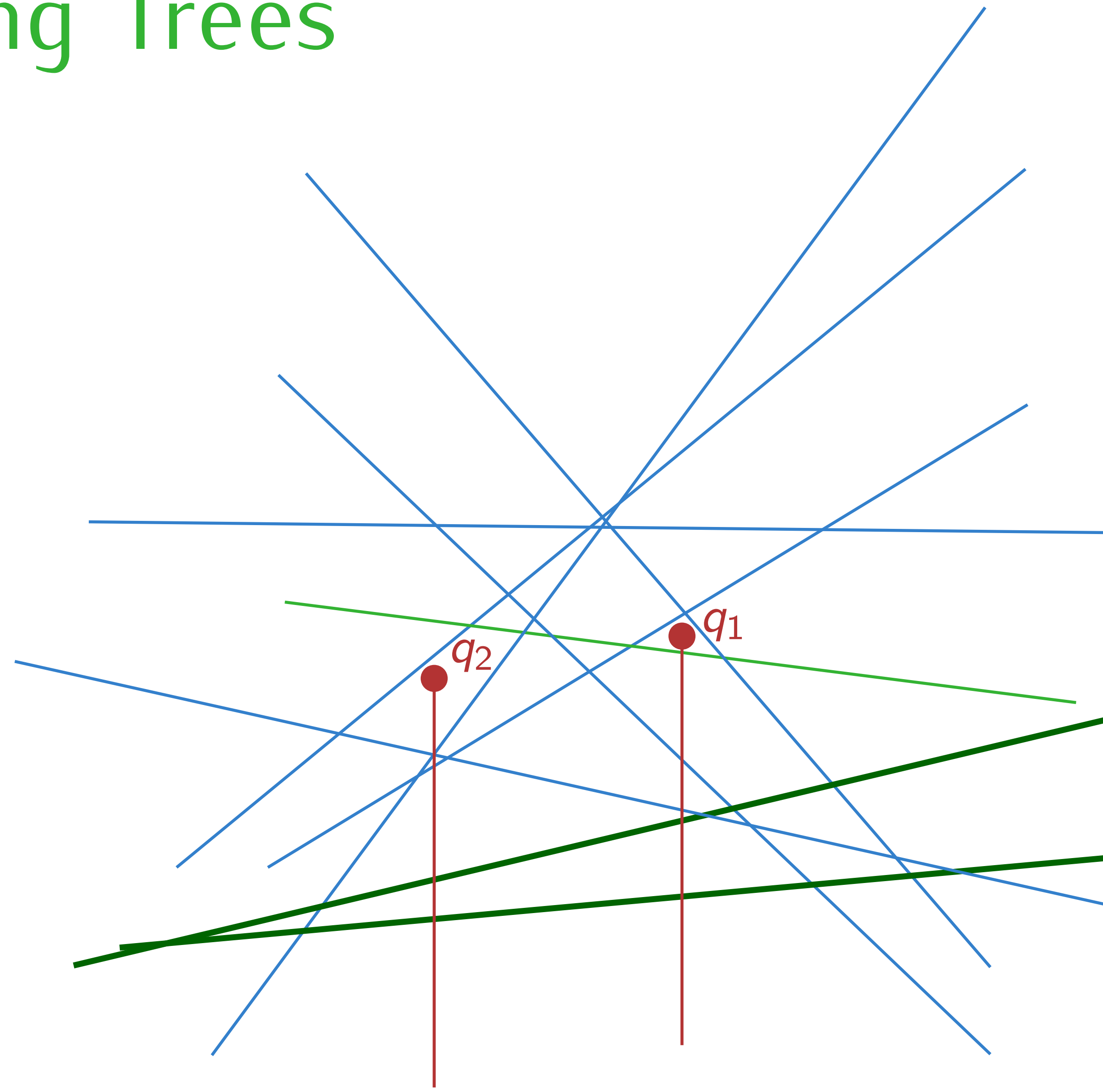
$M(n)$ = space usage

$$M(n) = \begin{cases} O(1) & \text{i} \\ O(r^2 n^{2+\varepsilon}) + \sum_{i=1}^{cr^2} M(n/r) & \text{o} \end{cases}$$

Multilevel Cutting Trees

Count all lines from P^* that lie above ℓ_1^*, above ℓ_2^*, and below ℓ_3^*

$\Longleftrightarrow$



Multilevel Cutting Trees

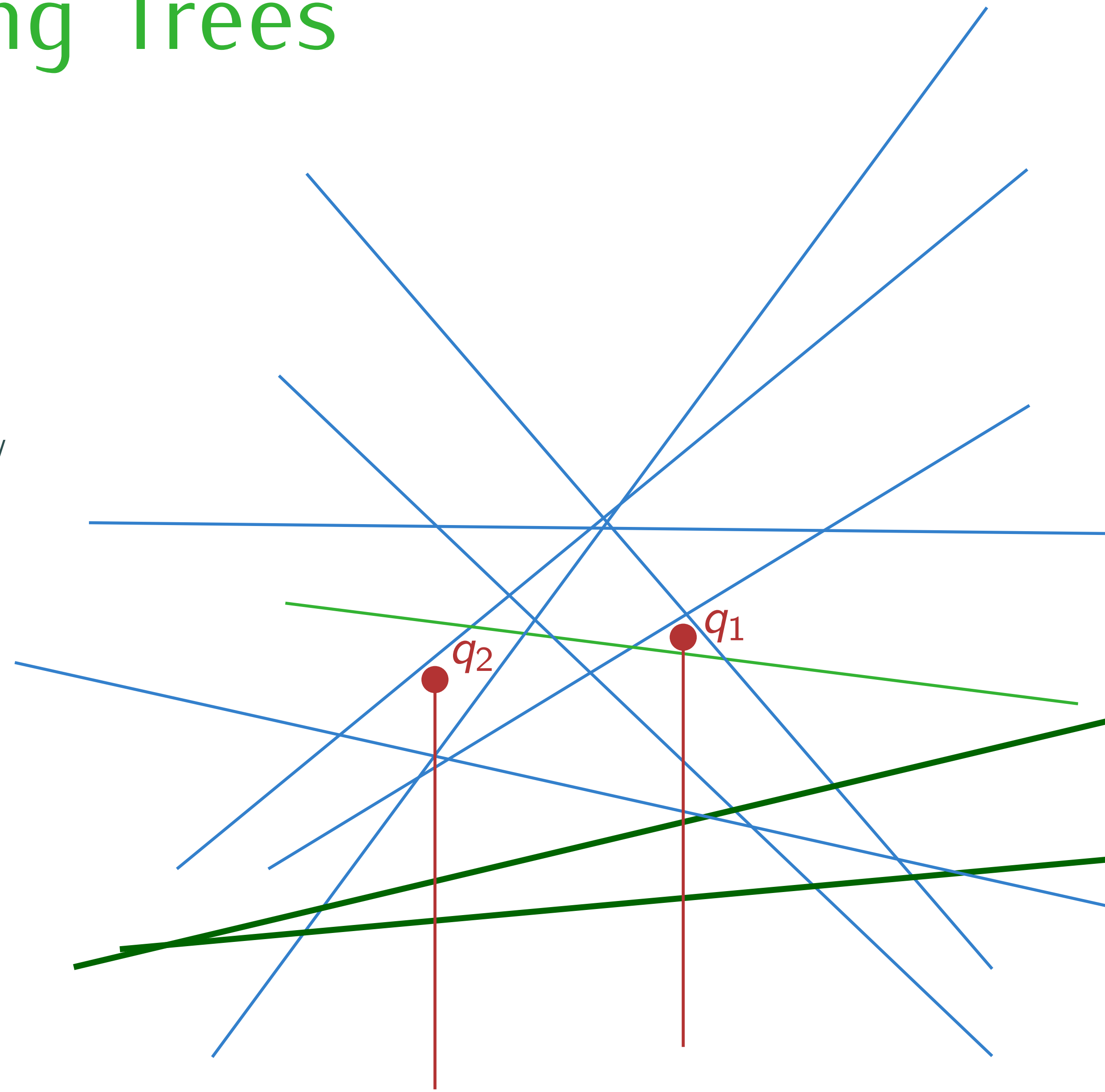
Count all lines from P^* that lie above ℓ_1^*, above ℓ_2^*, and below ℓ_3^*

$\Longleftrightarrow$

Count all points from P that lie below ℓ_1, below ℓ_2, and above ℓ_3

$\Longleftrightarrow$

Count all points from P that lie in a query triangle q (whose edges have supporting lines ℓ_1, ℓ_2, and ℓ_3)



Multilevel Cutting Trees

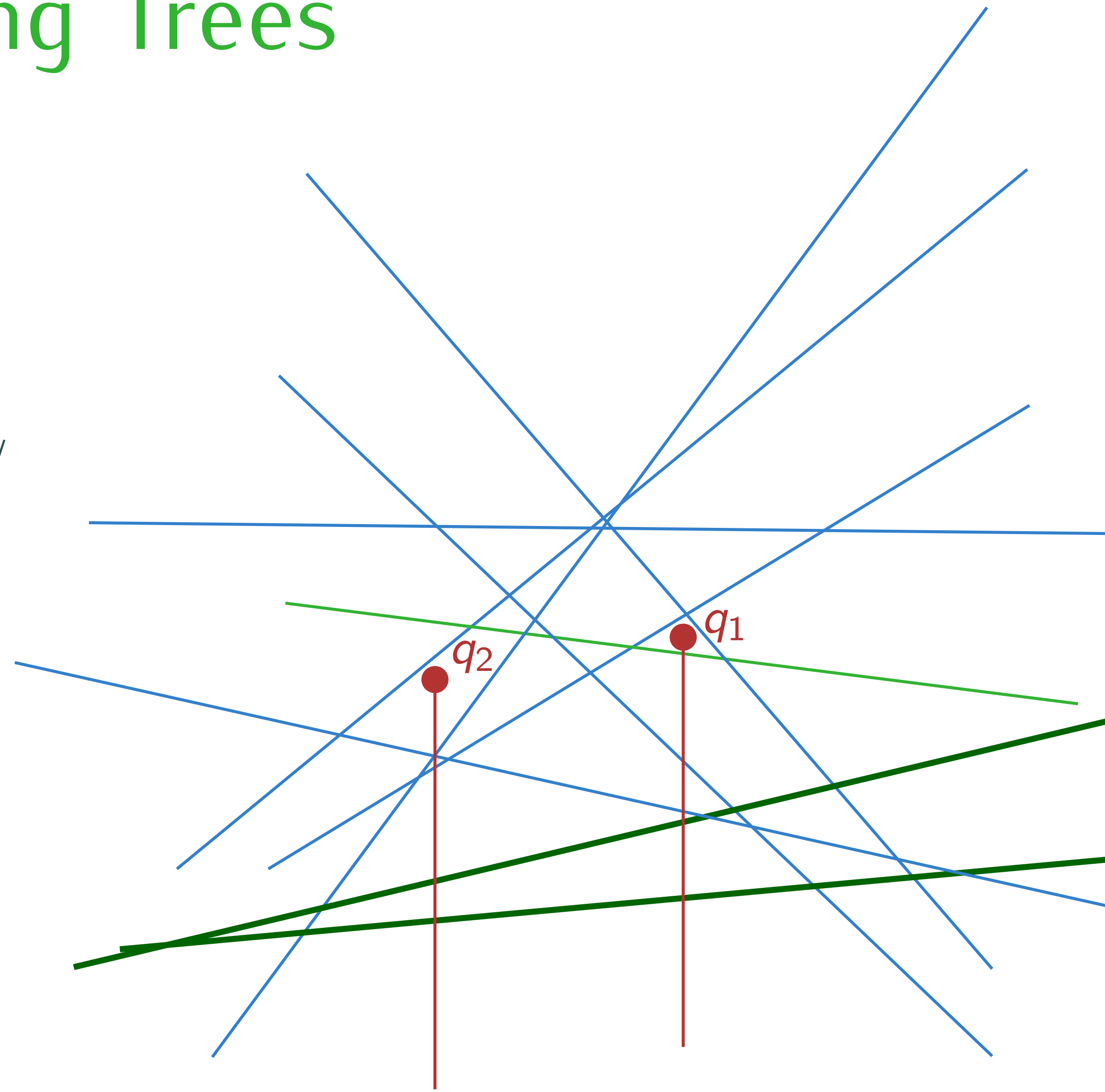
Count all lines from P^* that lie above ℓ_1^*, above ℓ_2^*, and below ℓ_3^*

$\Longleftrightarrow$

Count all points from P that lie below ℓ_1, below ℓ_2, and above ℓ_3

$\Longleftrightarrow$

Count all points from P that lie in a query triangle q (whose edges have supporting lines ℓ_1, ℓ_2, and ℓ_3)



Thm. We can count all points in q using a 3-level cutting tree in $O(\log^3 n)$ time. The data structure uses $O(n^{2+\varepsilon})$ space, and can be built in $O(n^{2+\varepsilon})$ time.

Multilevel Cutting Trees

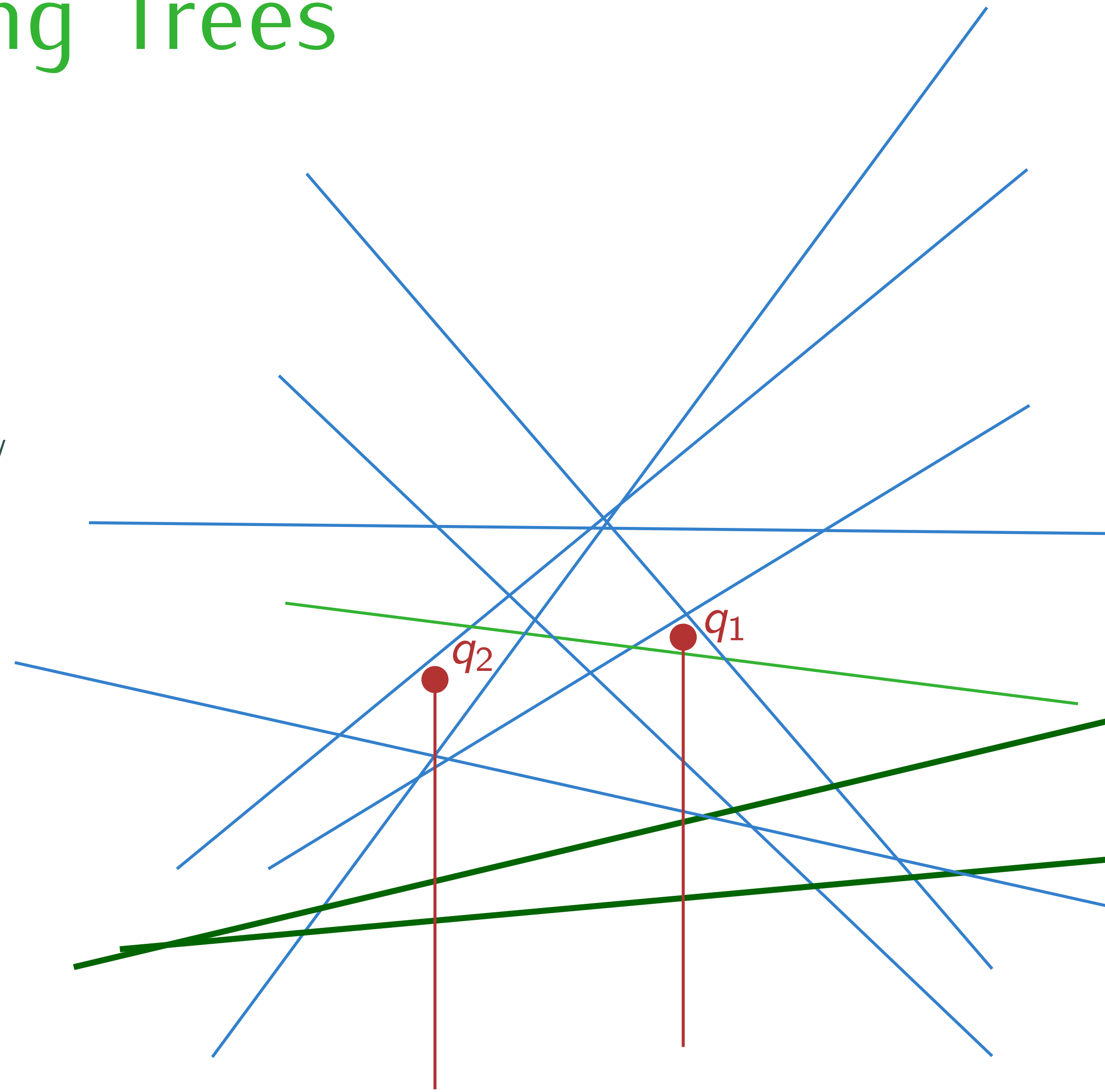
Count all lines from P^* that lie above ℓ_1^*, above ℓ_2^*, and below ℓ_3^*

$\Longleftrightarrow$

Count all points from P that lie below ℓ_1, below ℓ_2, and above ℓ_3

$\Longleftrightarrow$

Count all points from P that lie in a query triangle q (whose edges have supporting lines ℓ_1, ℓ_2, and ℓ_3)



Thm. We can count all points in q using a 3-level cutting tree in $O(\log^3 n)$ time. The data structure uses $O(n^{2+\varepsilon})$ space, and can be built in $O(n^{2+\varepsilon})$ time.

we can report those points in $O(\log^3 n + k)$ time, where k is the output size.