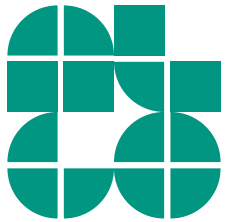
Computational Geometry – Exercise

Range Searching

LEHRSTUHL FÜR ALGORITHMIK I · INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Guido Brückner

06.07.2018



Orthogonal Range Queries for $d = 2$

Given: Set P of n points in $\mathbb{R}^2$

Find: A data structure that efficiently answers queries of the form $[a_1, b_1] \times \cdots \times [a_d, b_d]$

Orthogonal Range Queries for $d = 2$

Karlsruhe Institute of Technology

Given: Set P of n points in $\mathbb{R}^2$

Find: A data structure that efficiently answers queries of the form $[a_1, b_1] \times \cdots \times [a_d, b_d]$

Solutions:

- *one* search tree, alternate search for x and y coordinates
  $\rightarrow$ kd-**Tree**
- *primary* search tree on x-coordinates,
  several *secondary* search trees on y-coordinates
  $\rightarrow$ **Range Tree**

Orthogonal Range Queries for $d = 2$

Karlsruhe Institute of Technology

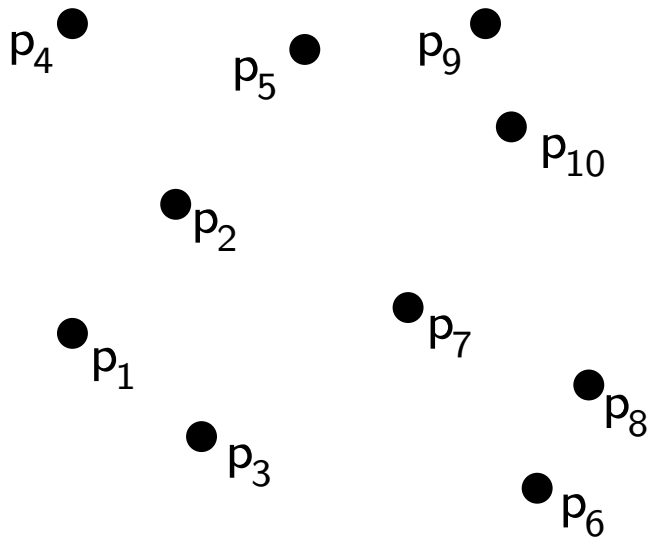
Given: Set P of n points in $\mathbb{R}^2$

Find: A data structure that efficiently answers queries of the form $[a_1, b_1] \times \cdots \times [a_d, b_d]$

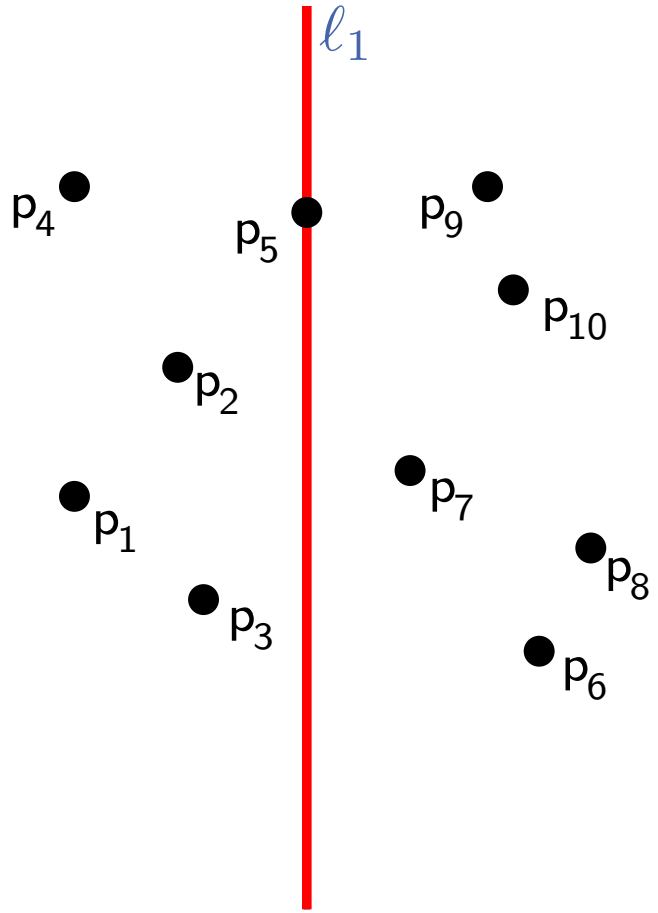
Solutions:

- *one* search tree, alternate search for x and y coordinates
  $\rightarrow$ kd-**Tree**
- *primary* search tree on x-coordinates,
  several *secondary* search trees on y-coordinates
  $\rightarrow$ **Range Tree**

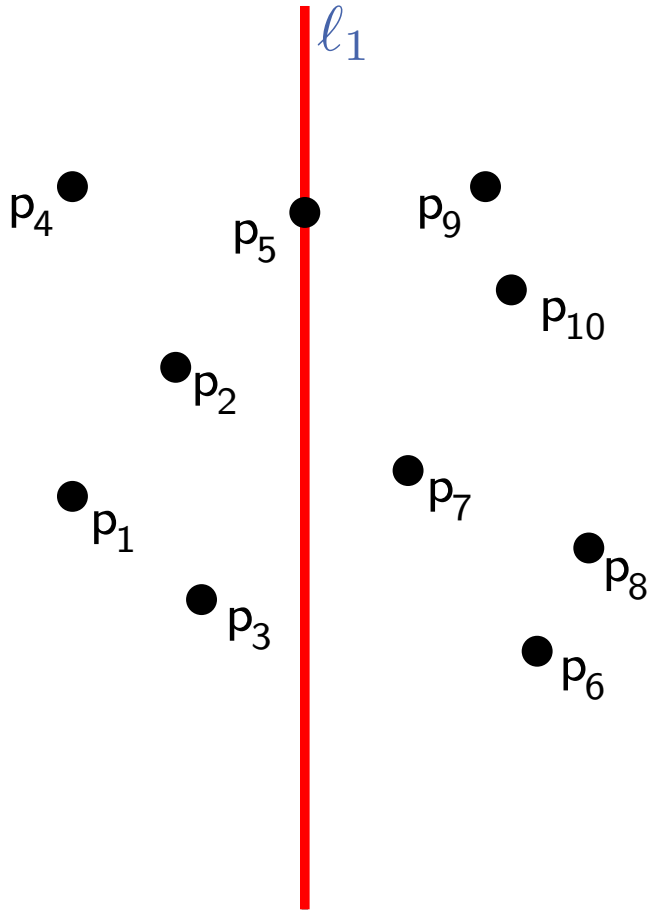
kd-Trees: Example

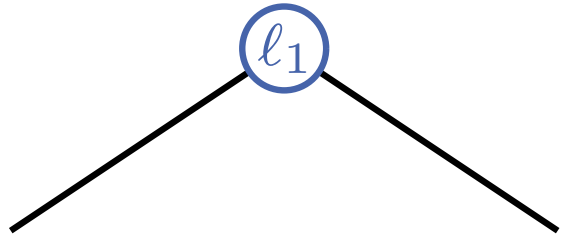


kd-Trees: Example

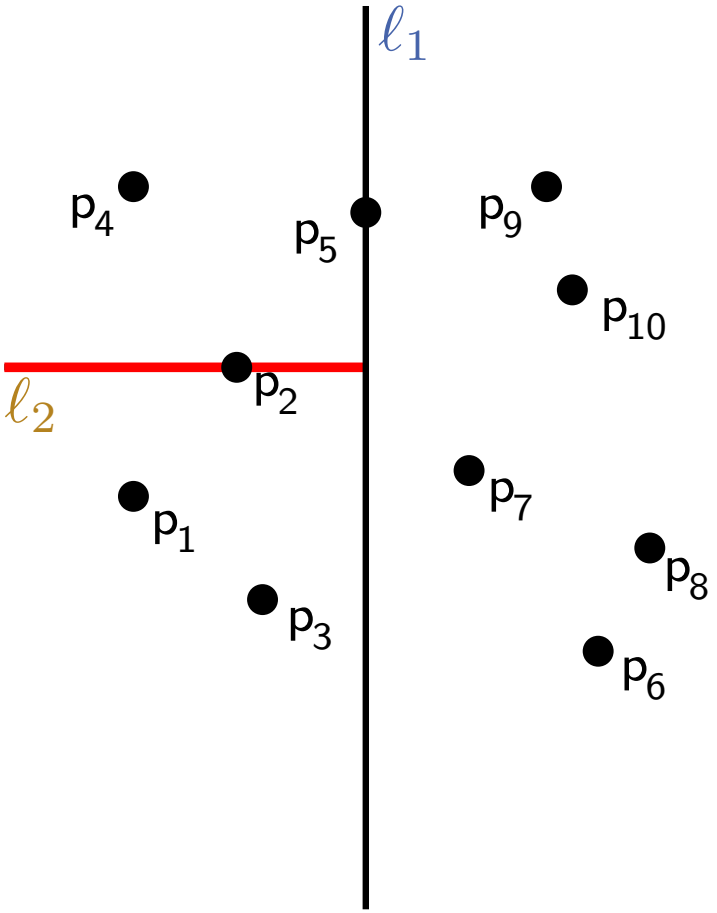


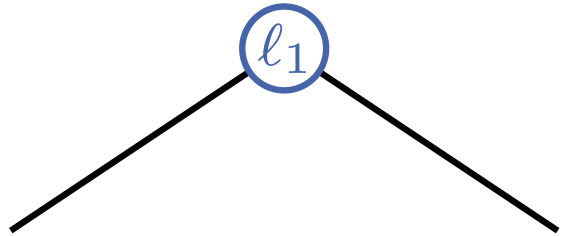
kd-Trees: Example



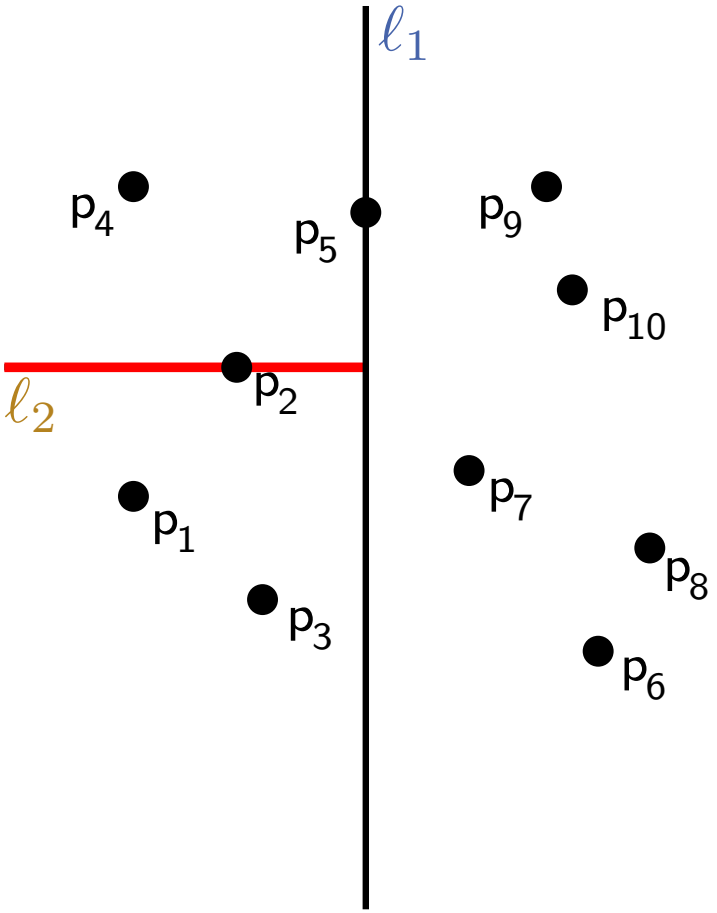


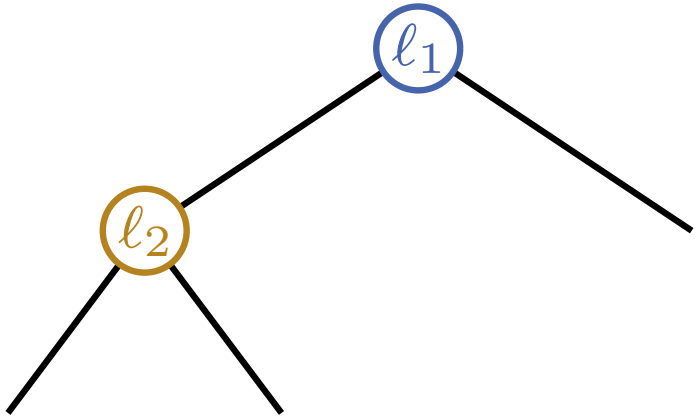
kd-Trees: Example



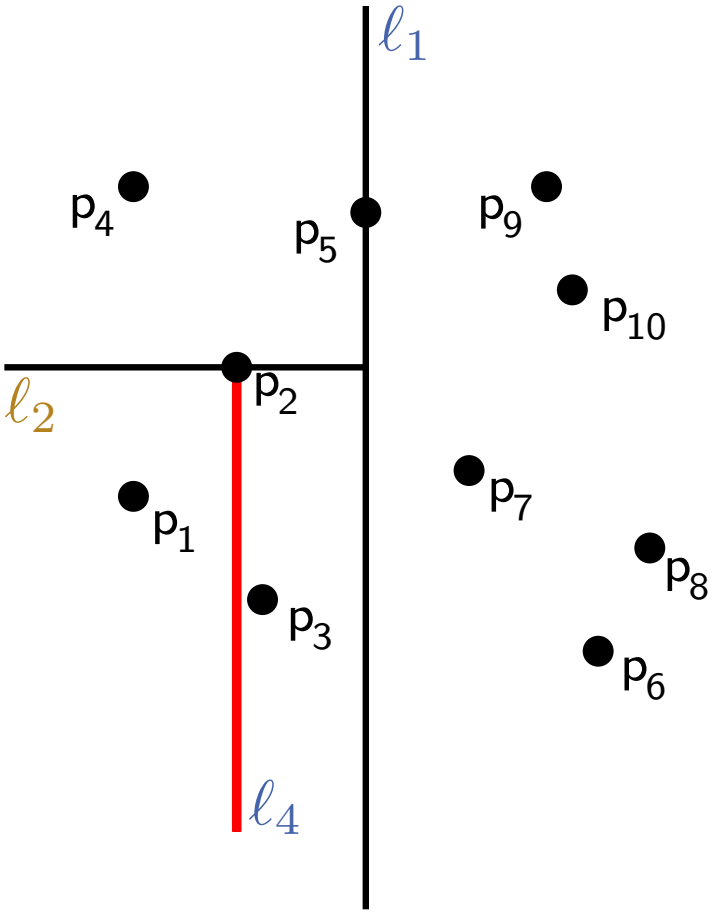


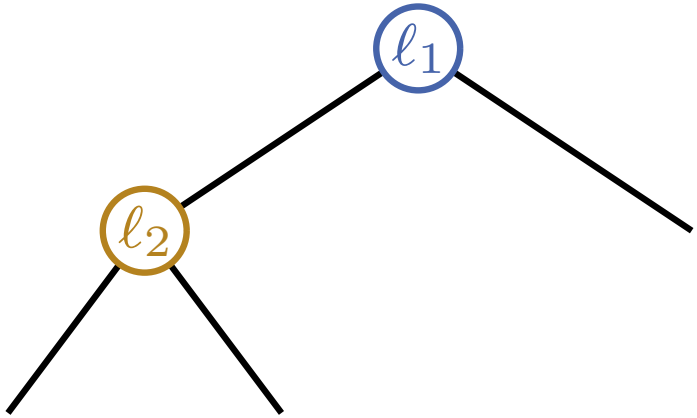
kd-Trees: Example



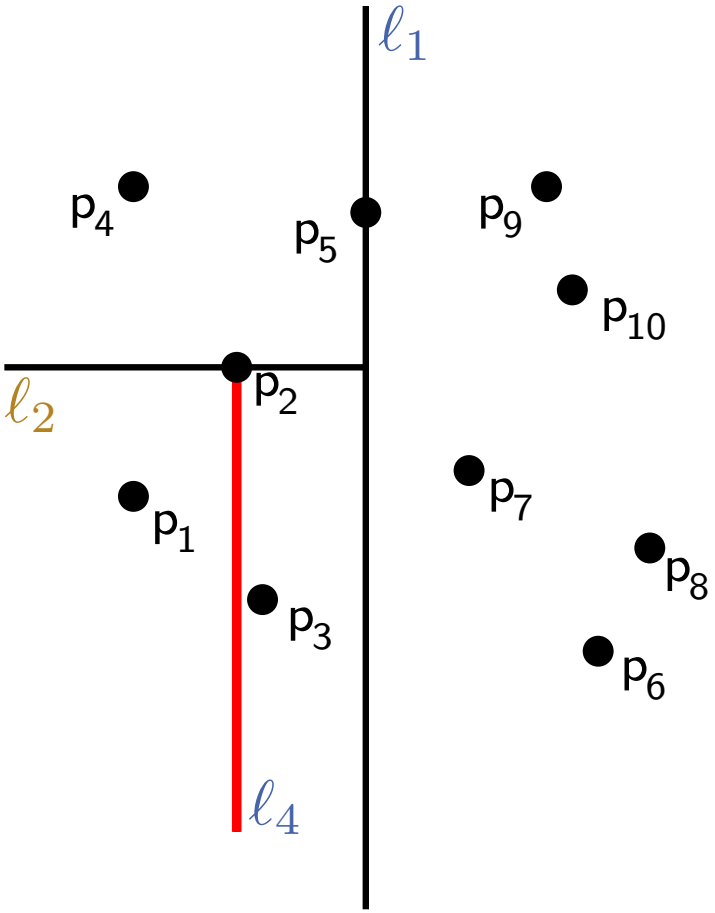


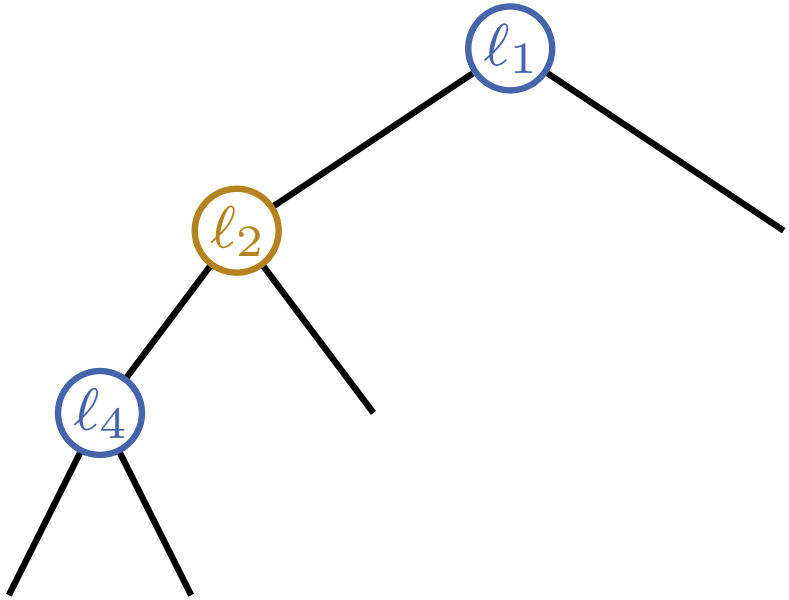
kd-Trees: Example



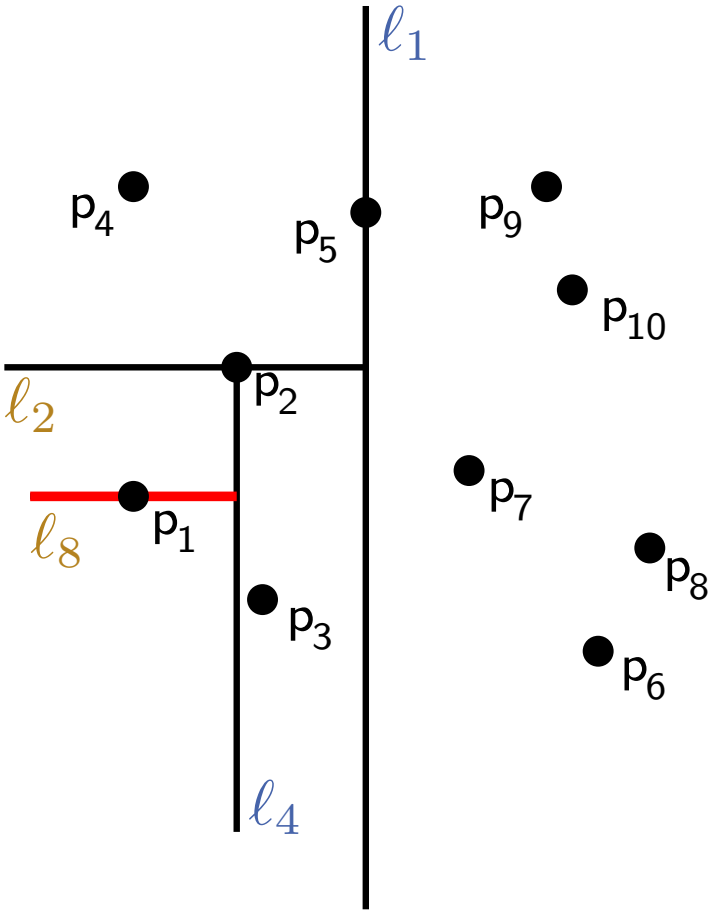


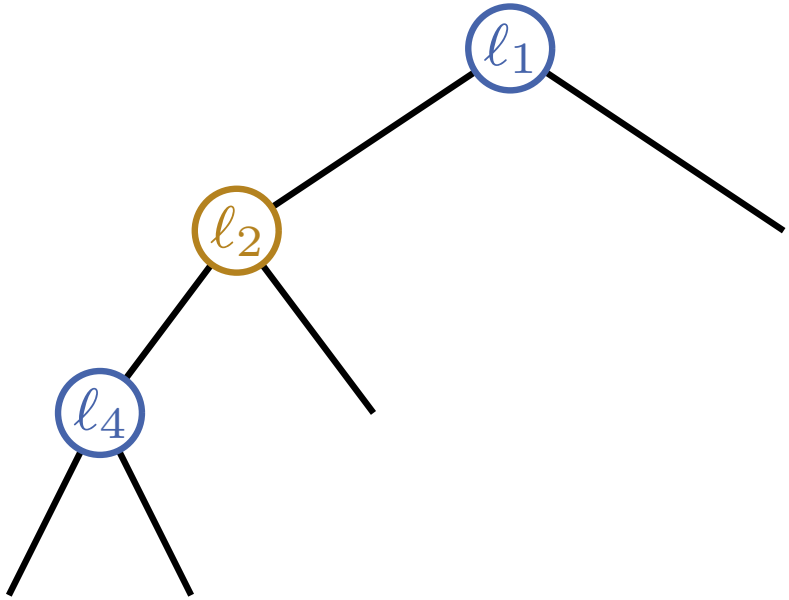
kd-Trees: Example



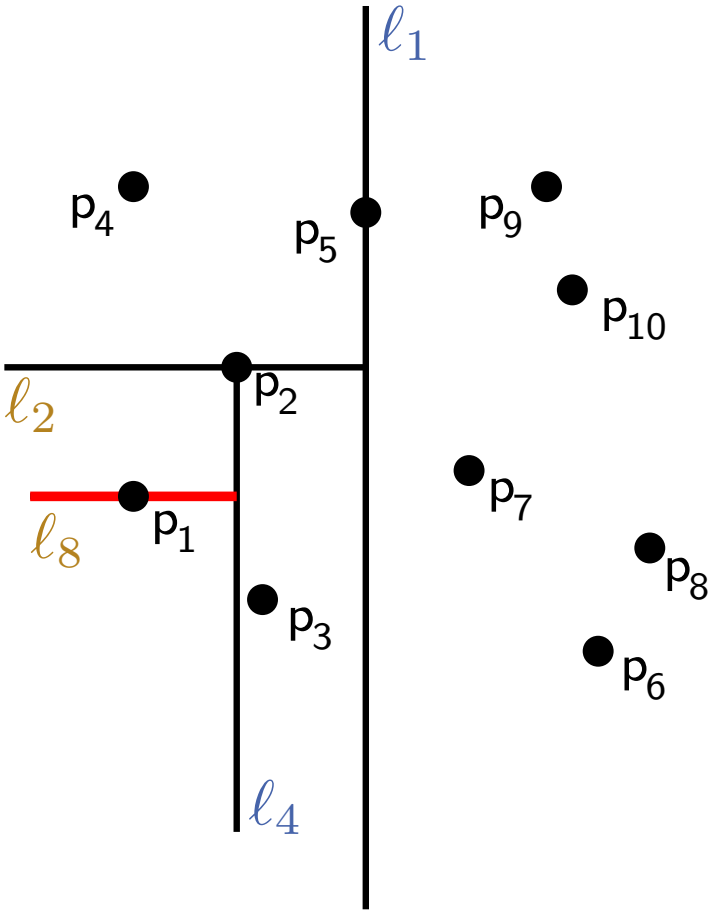


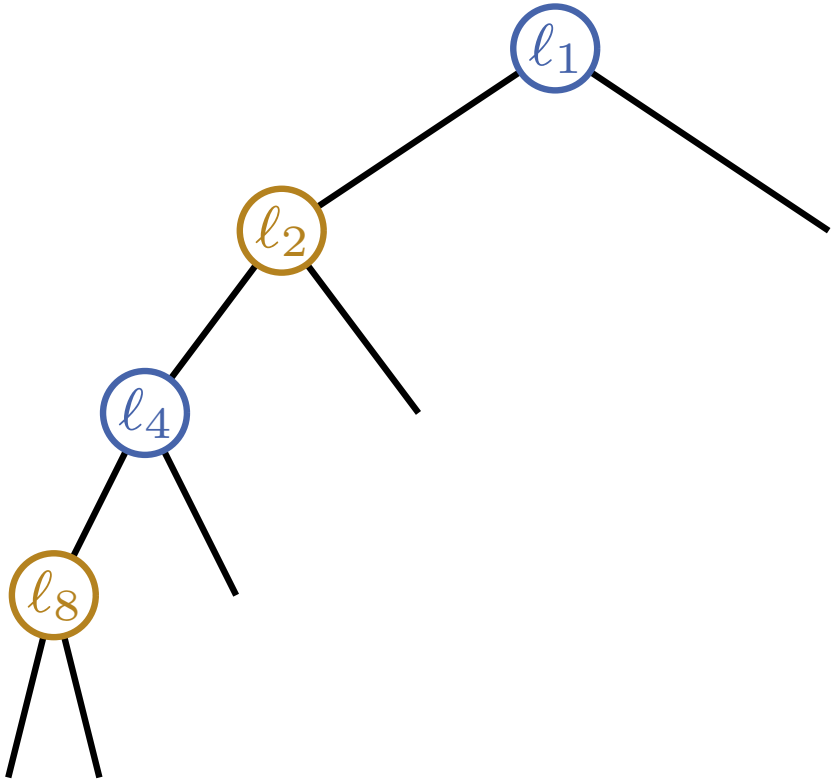
kd-Trees: Example



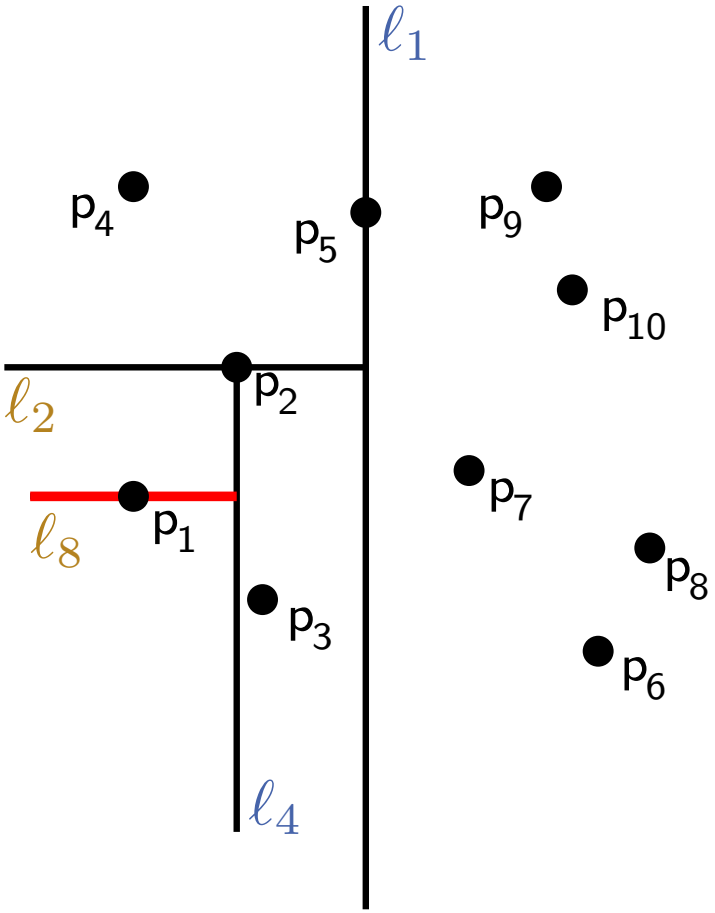


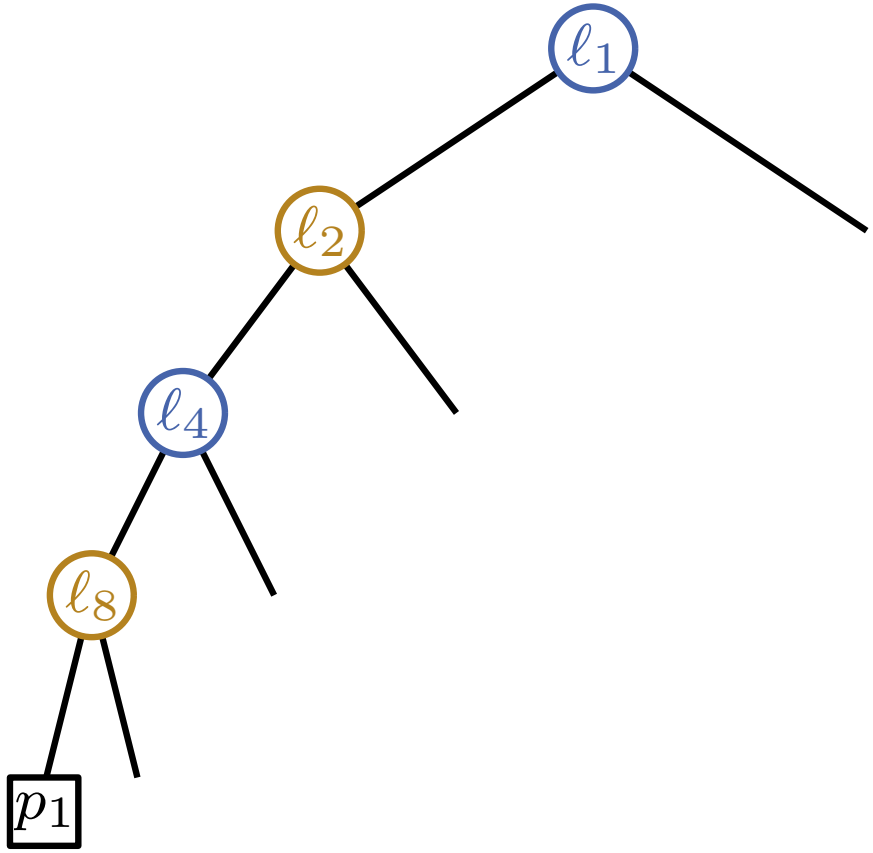
kd-Trees: Example



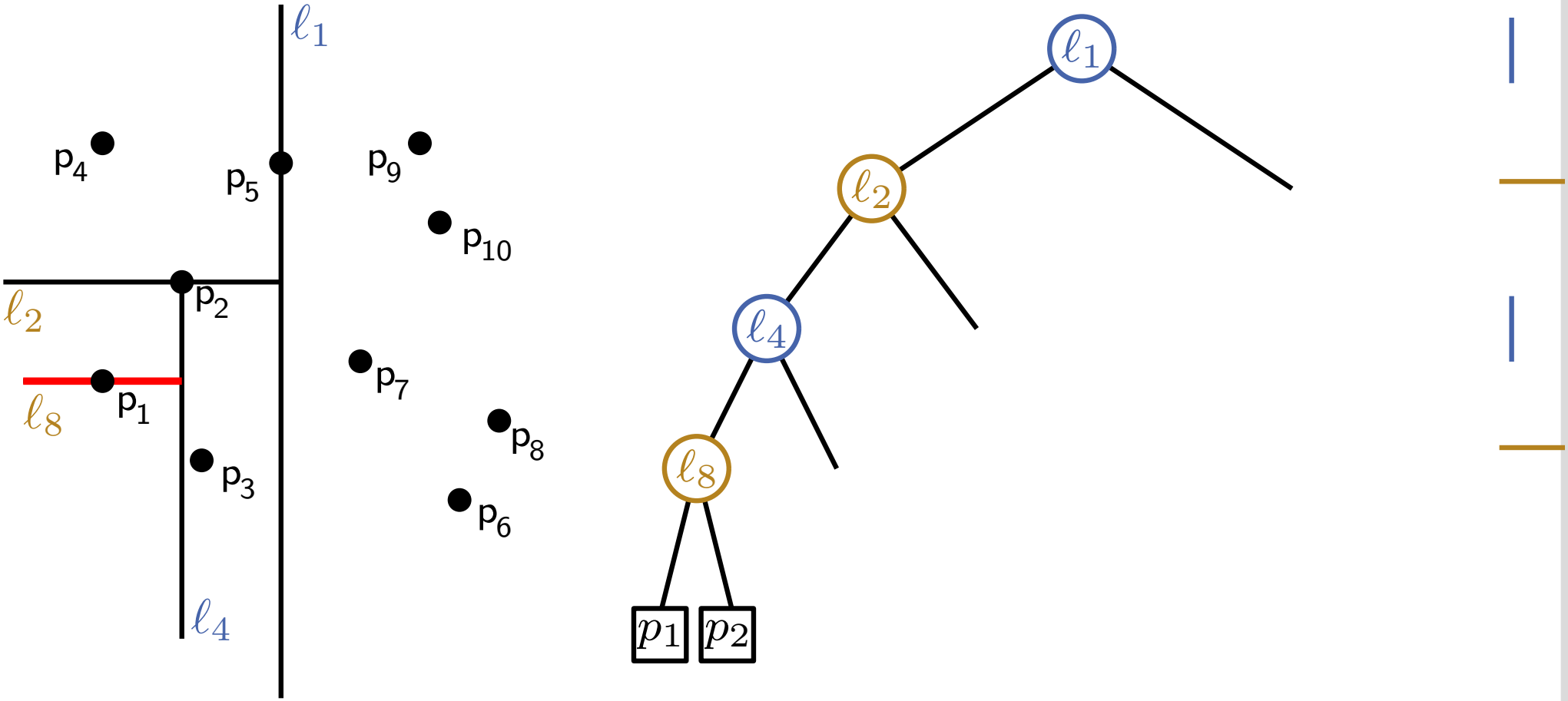


kd-Trees: Example

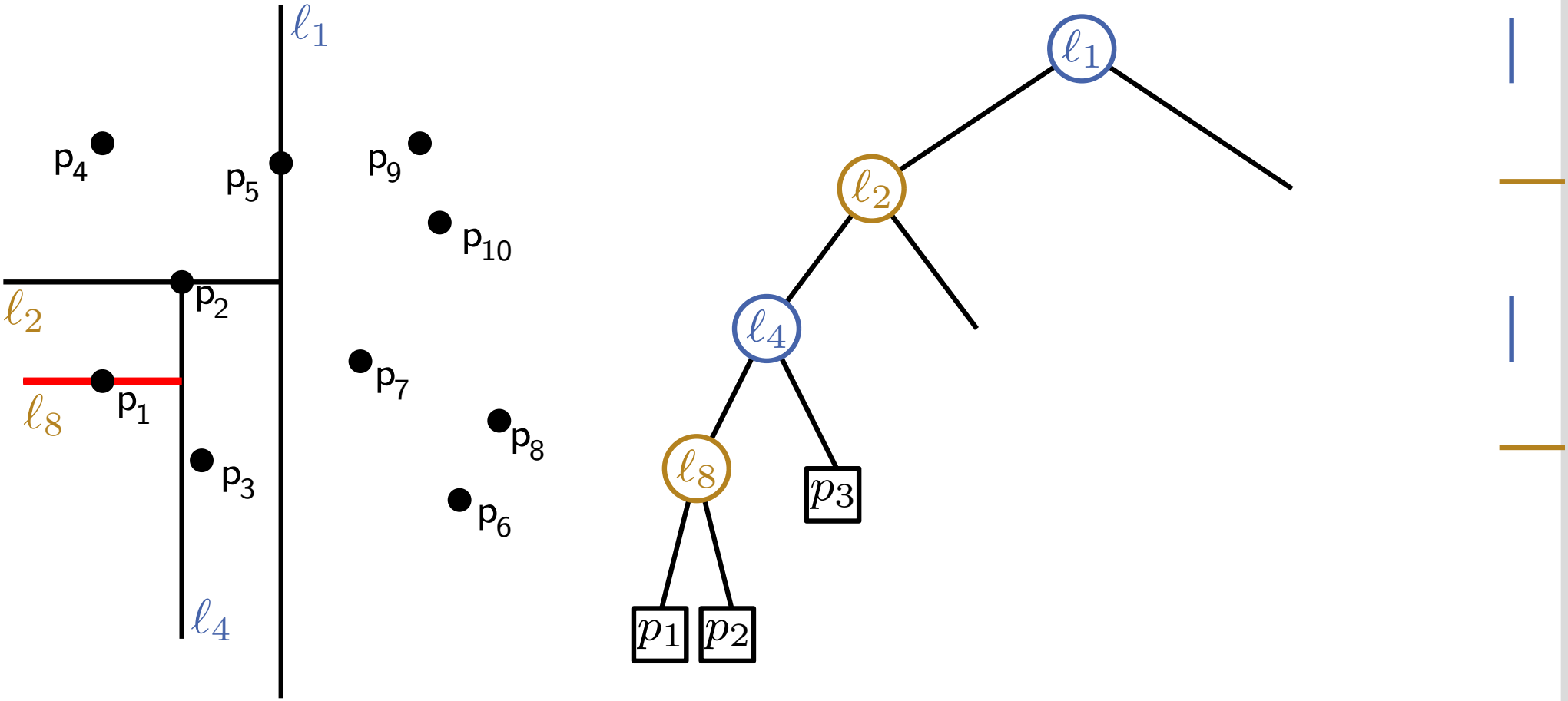




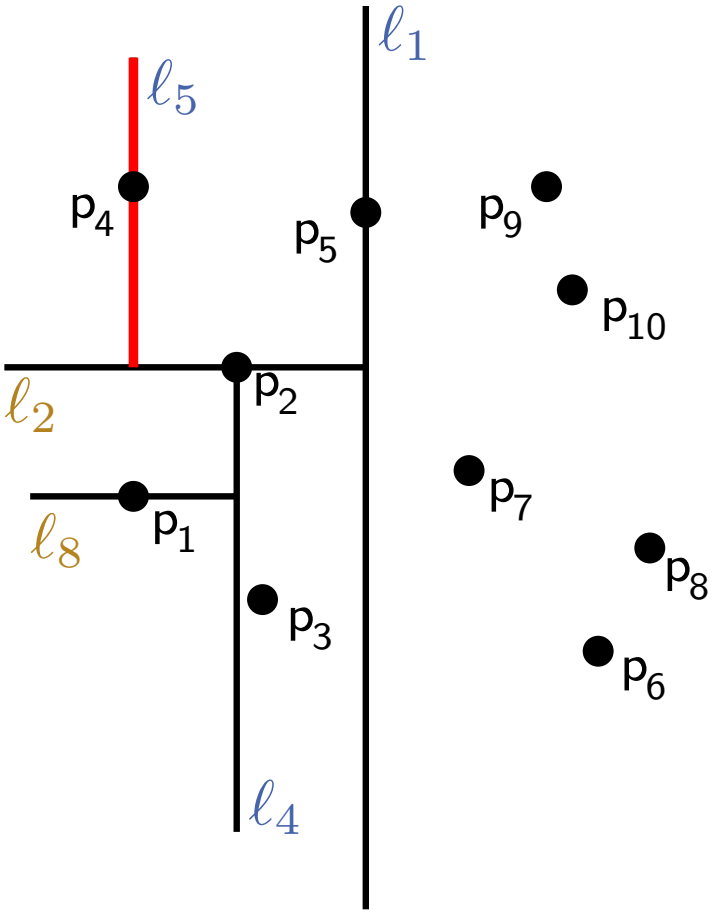
kd-Trees: Example

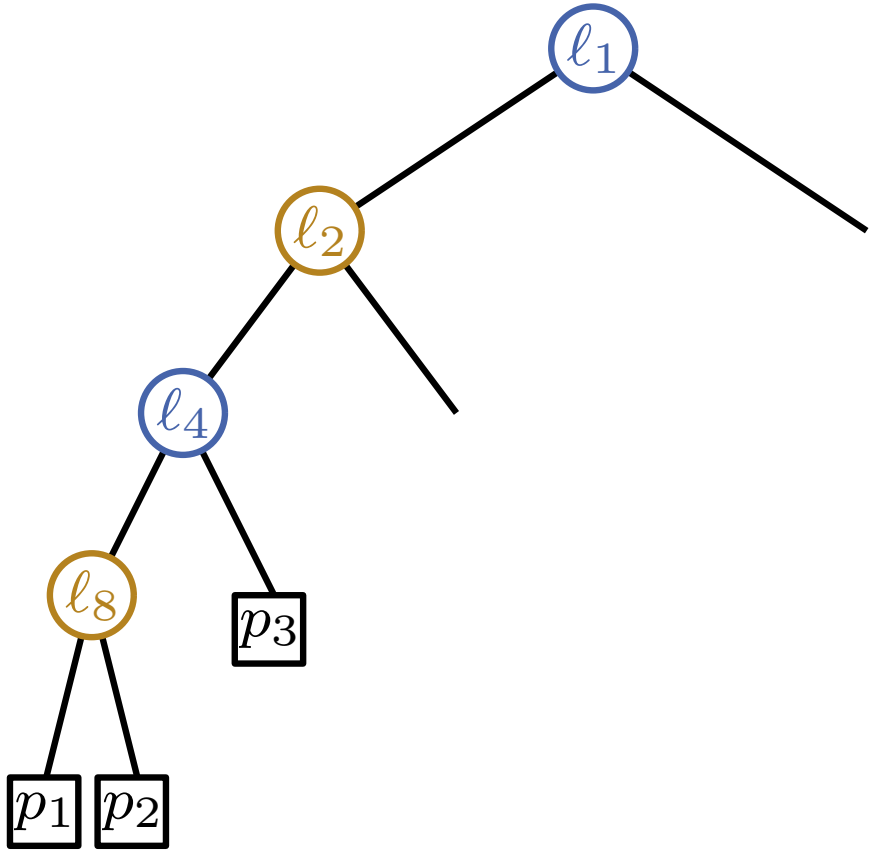


kd-Trees: Example

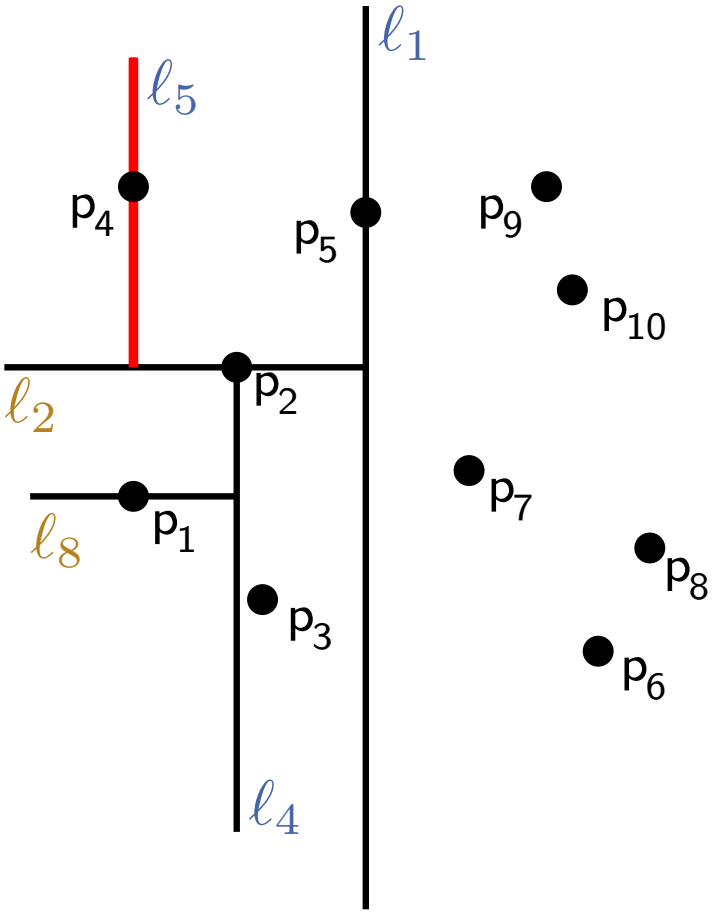


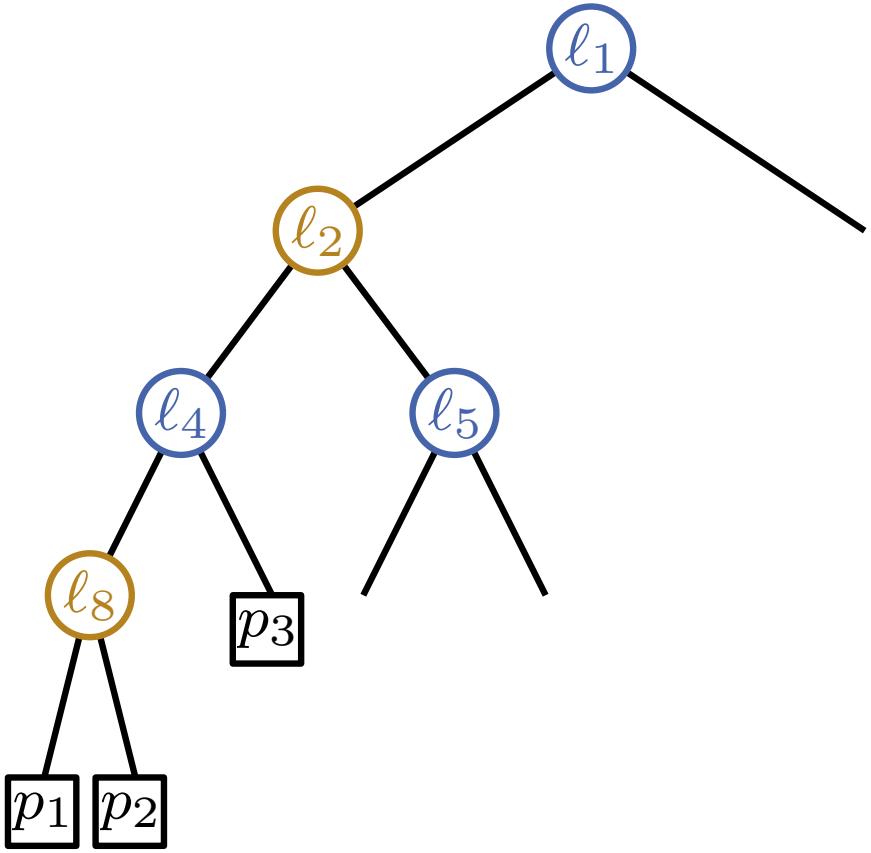
kd-Trees: Example



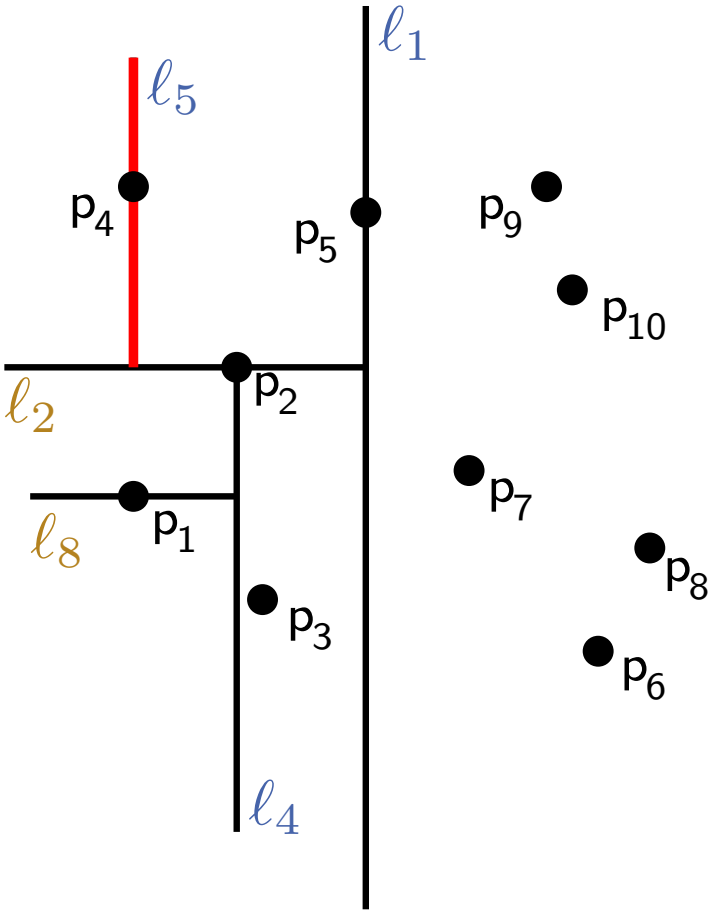


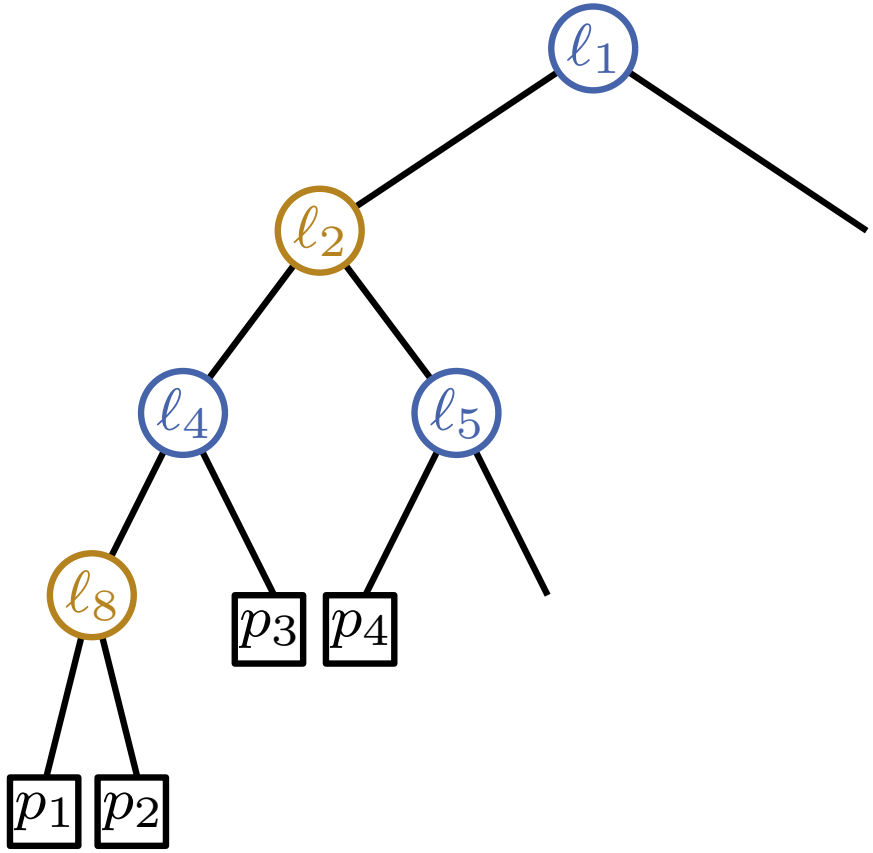
kd-Trees: Example



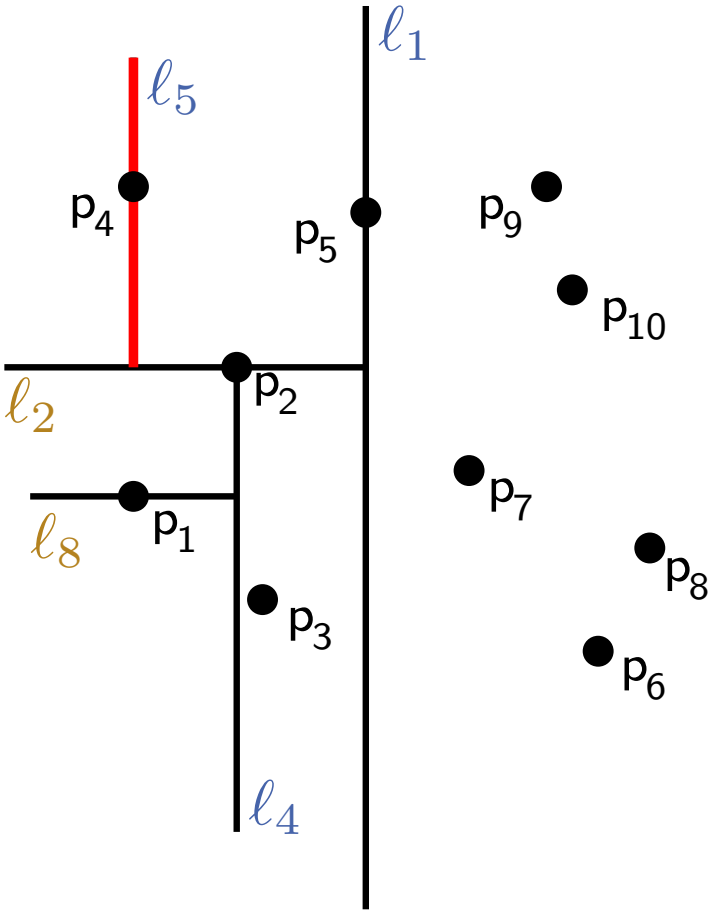


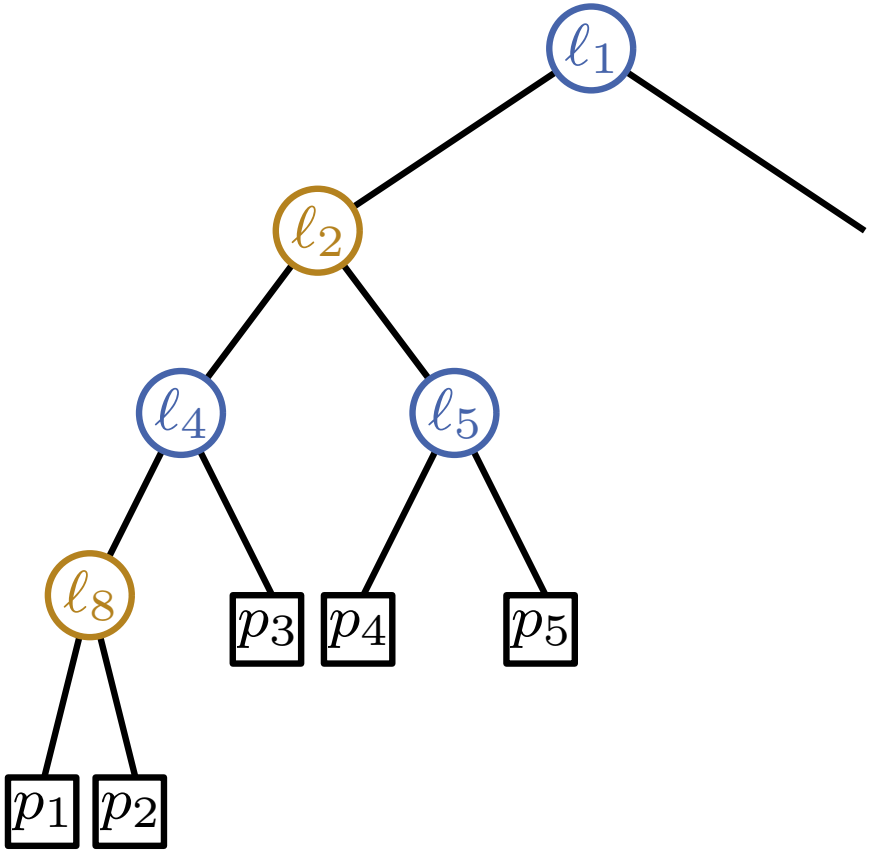
kd-Trees: Example



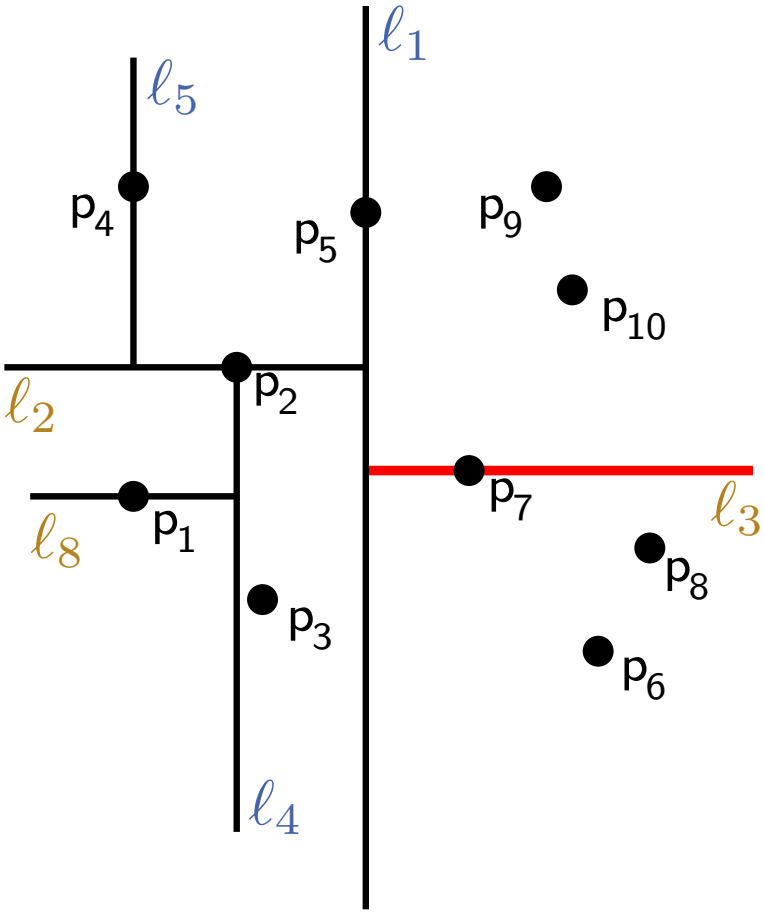


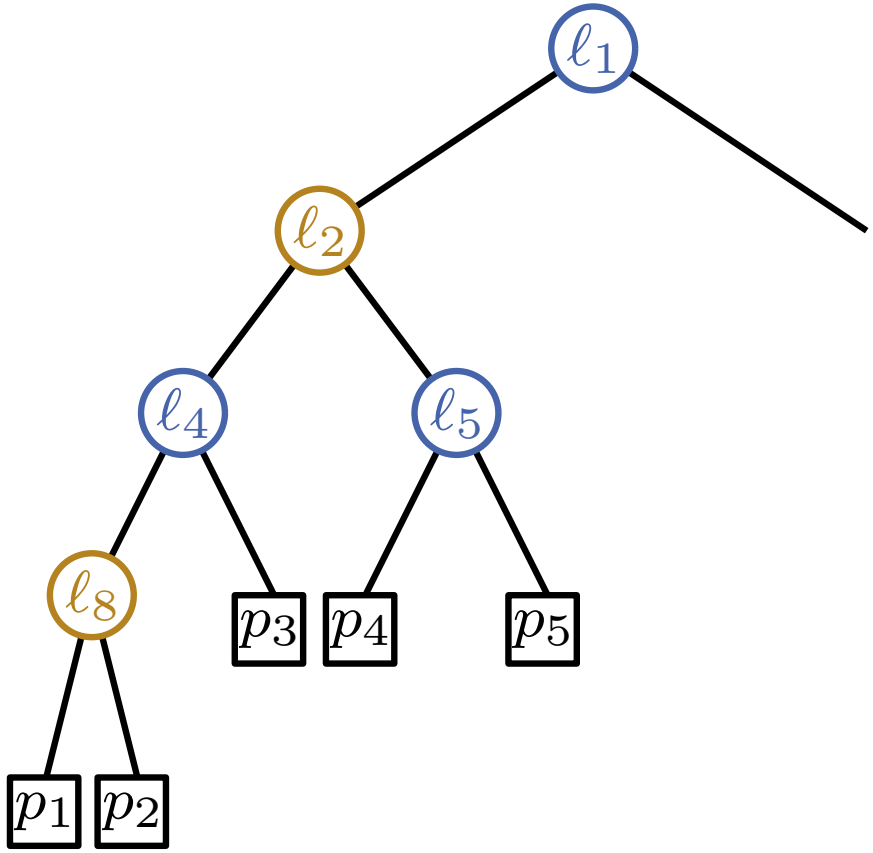
kd-Trees: Example



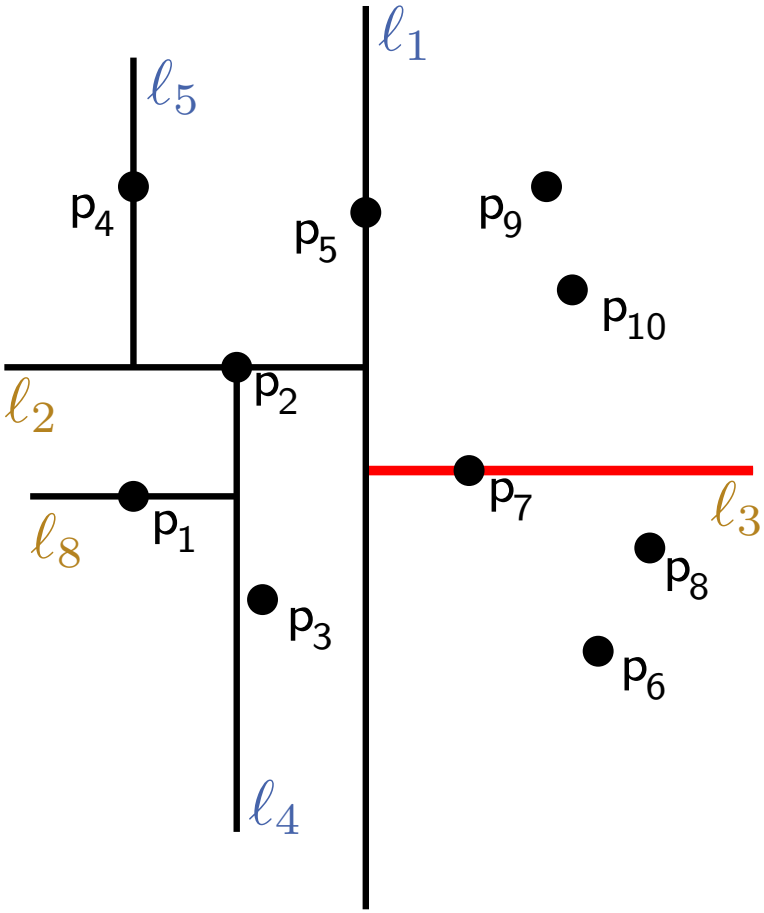


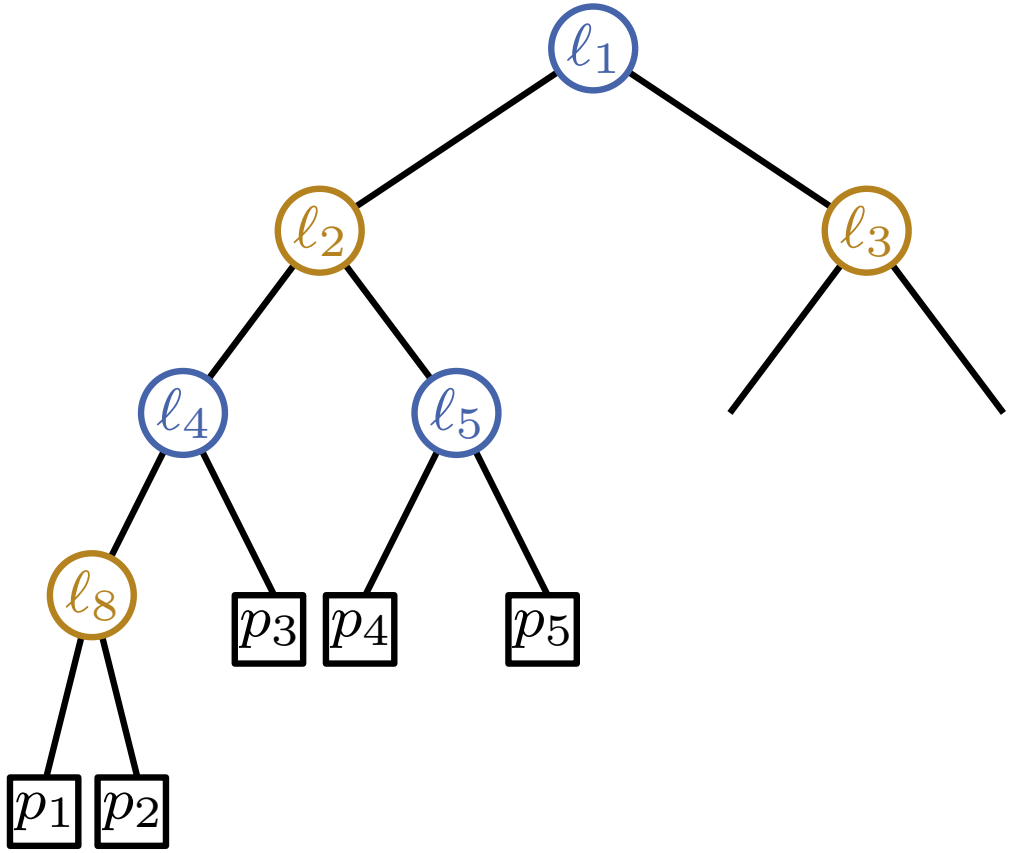
kd-Trees: Example



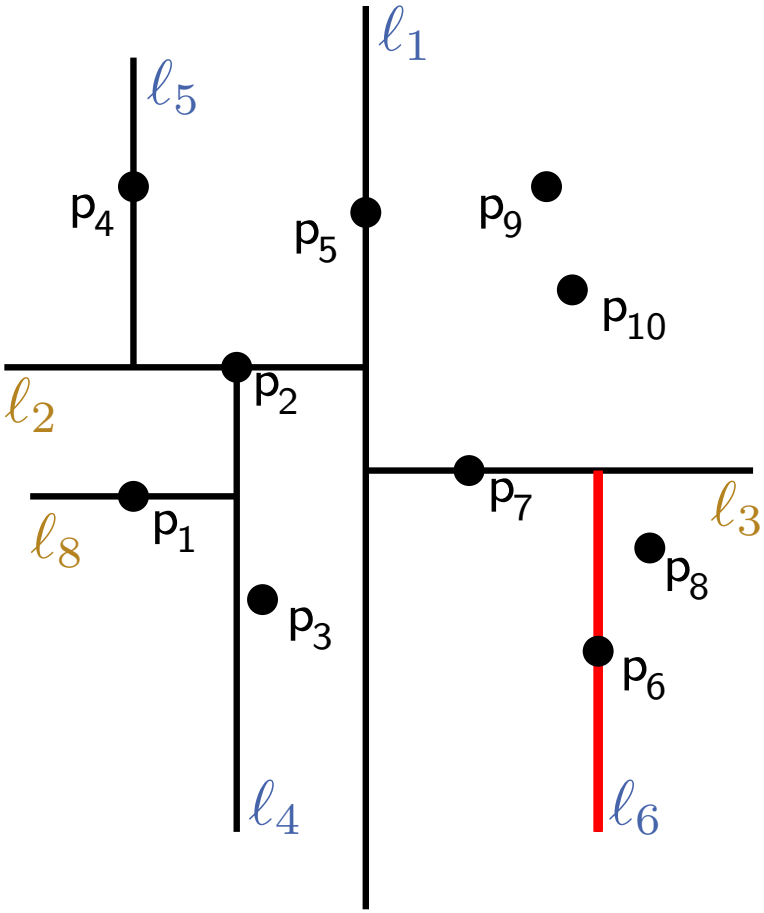


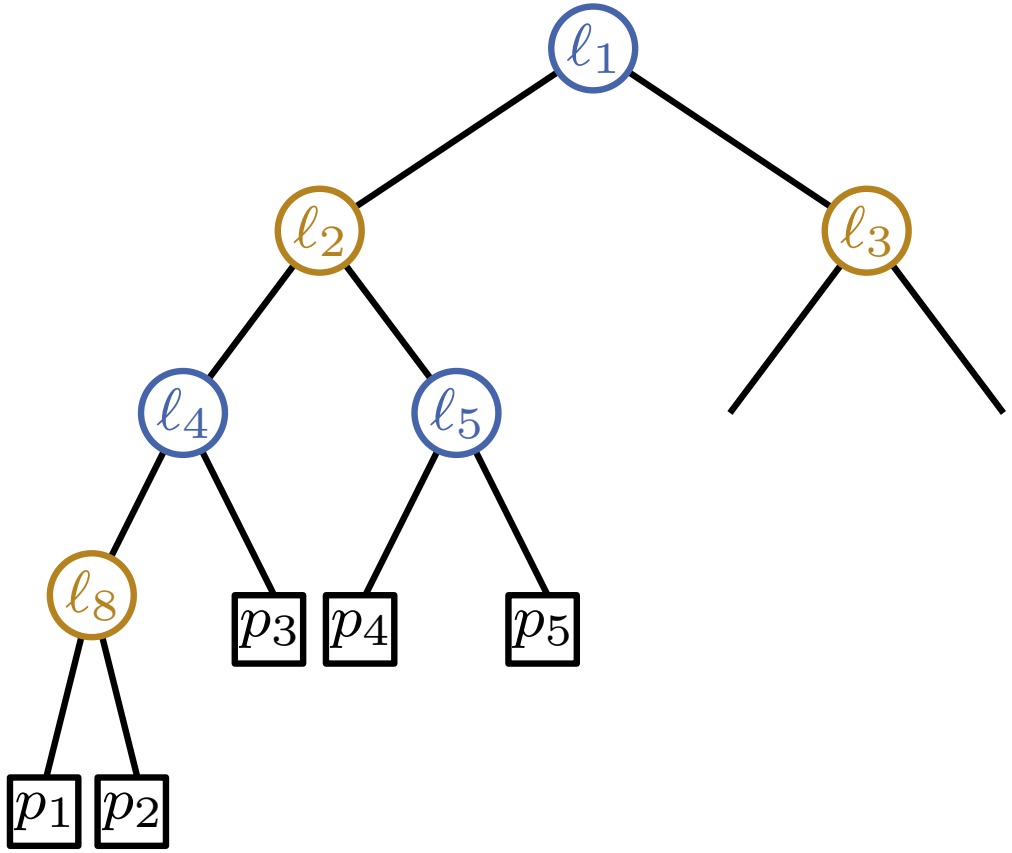
kd-Trees: Example



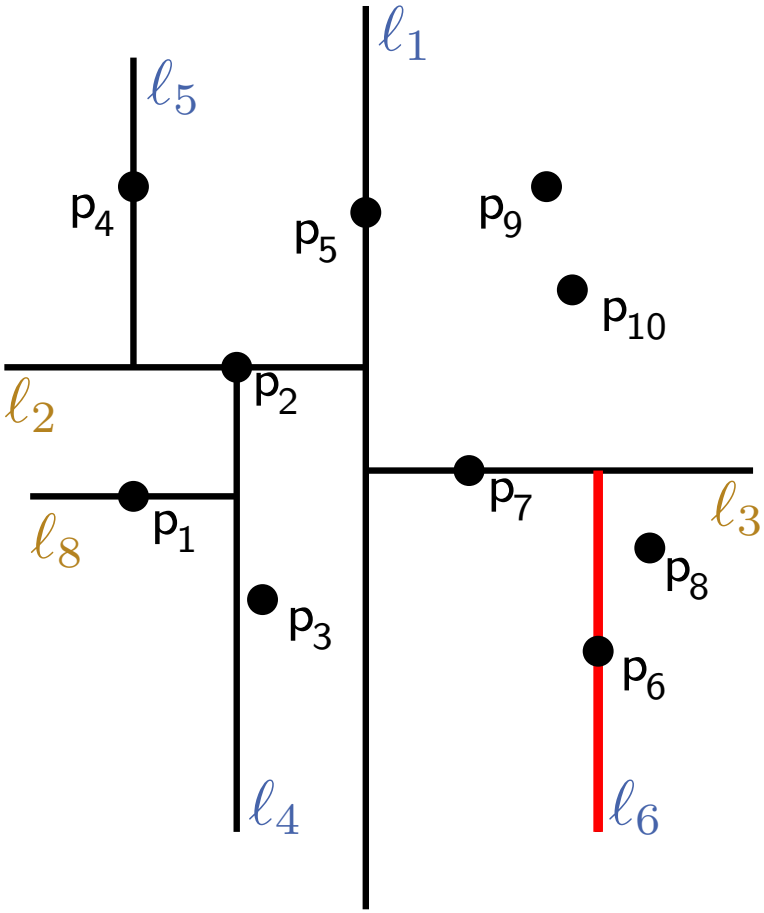


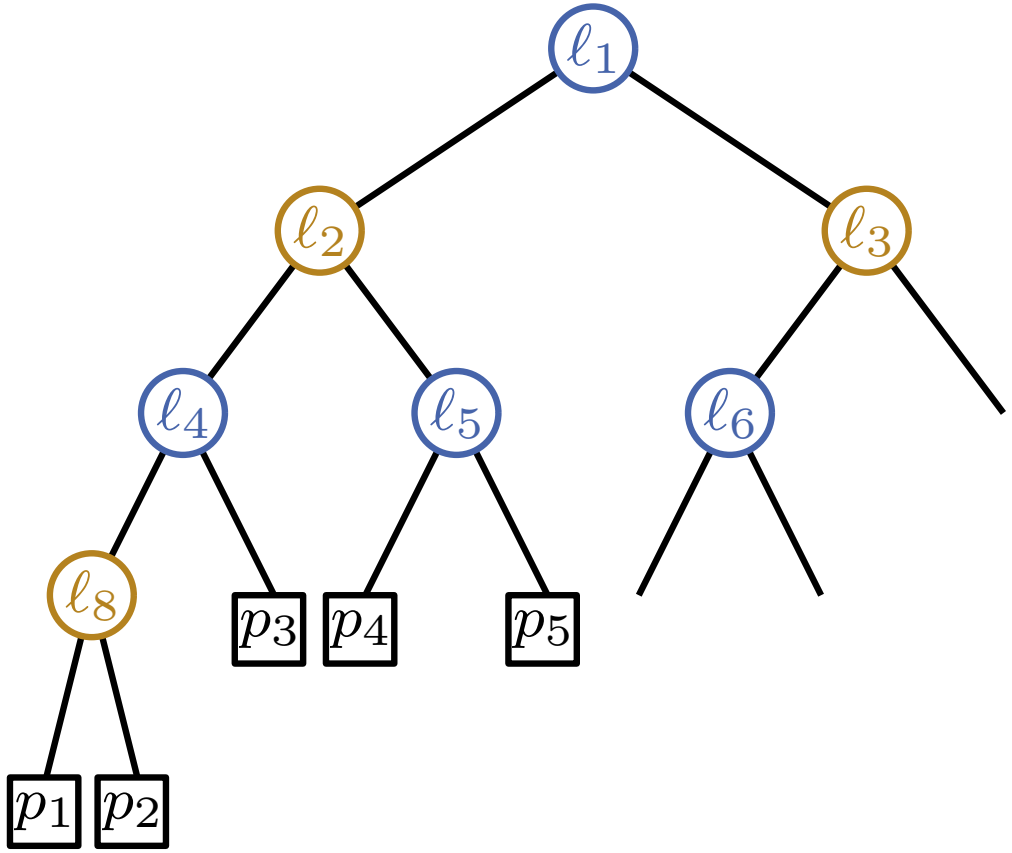
kd-Trees: Example



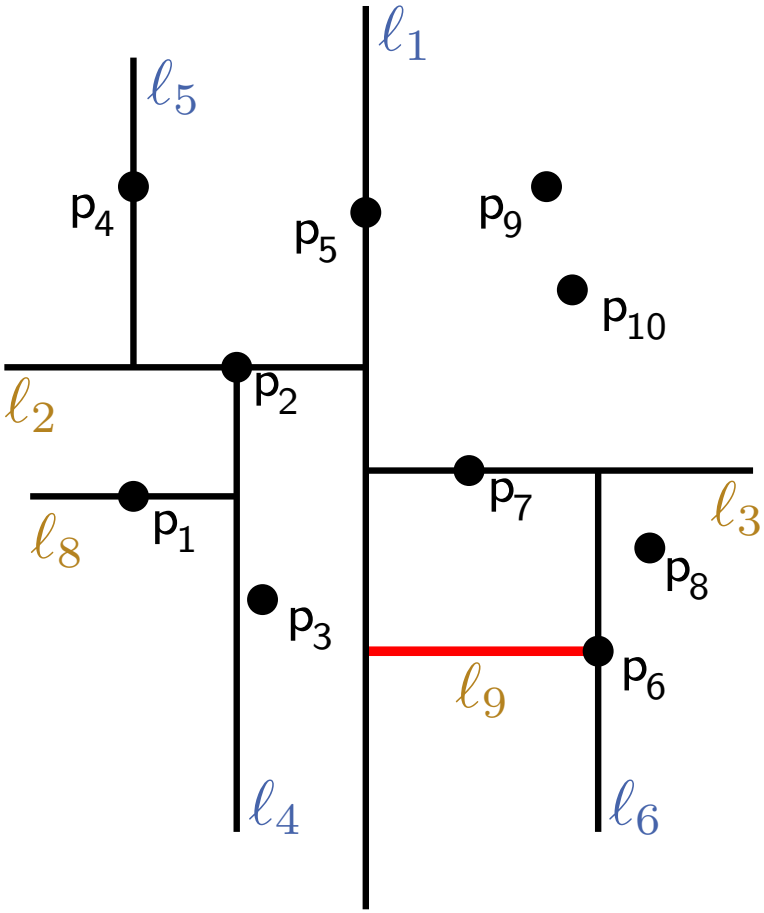


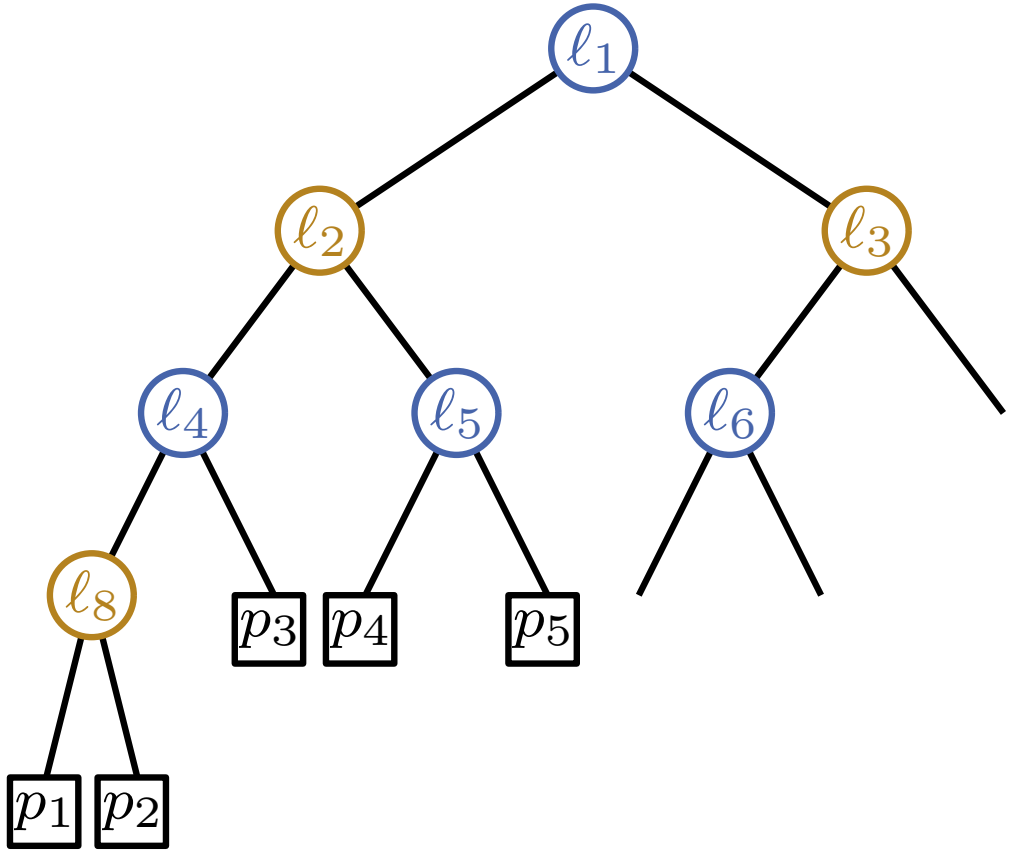
kd-Trees: Example



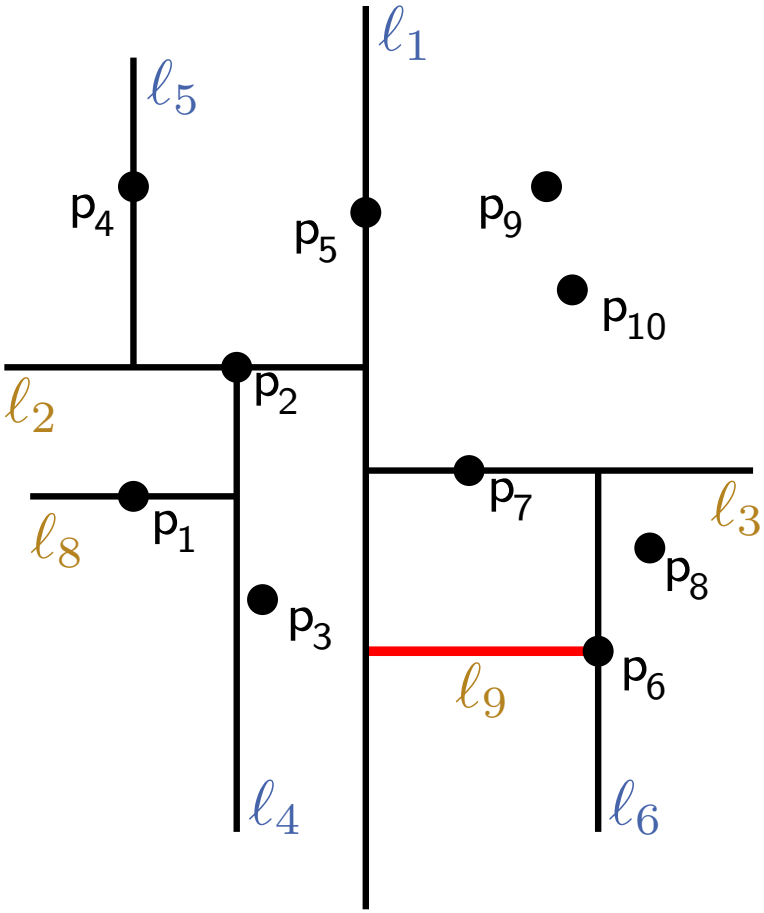


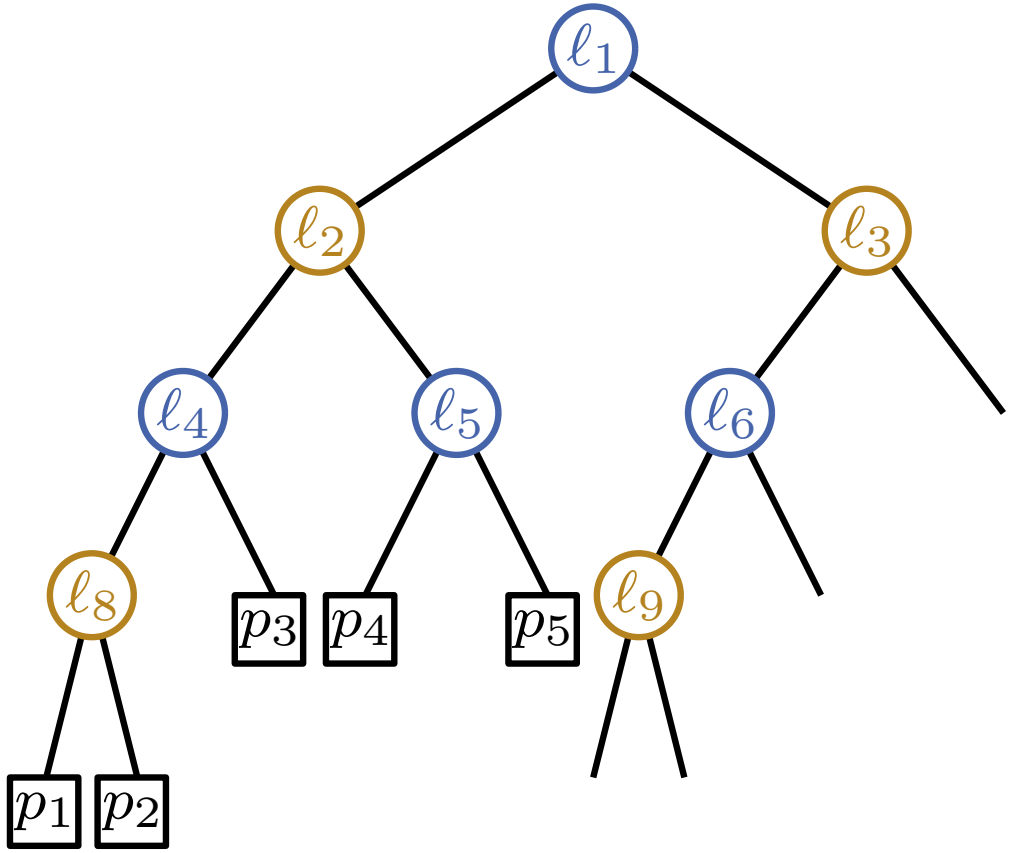
kd-Trees: Example



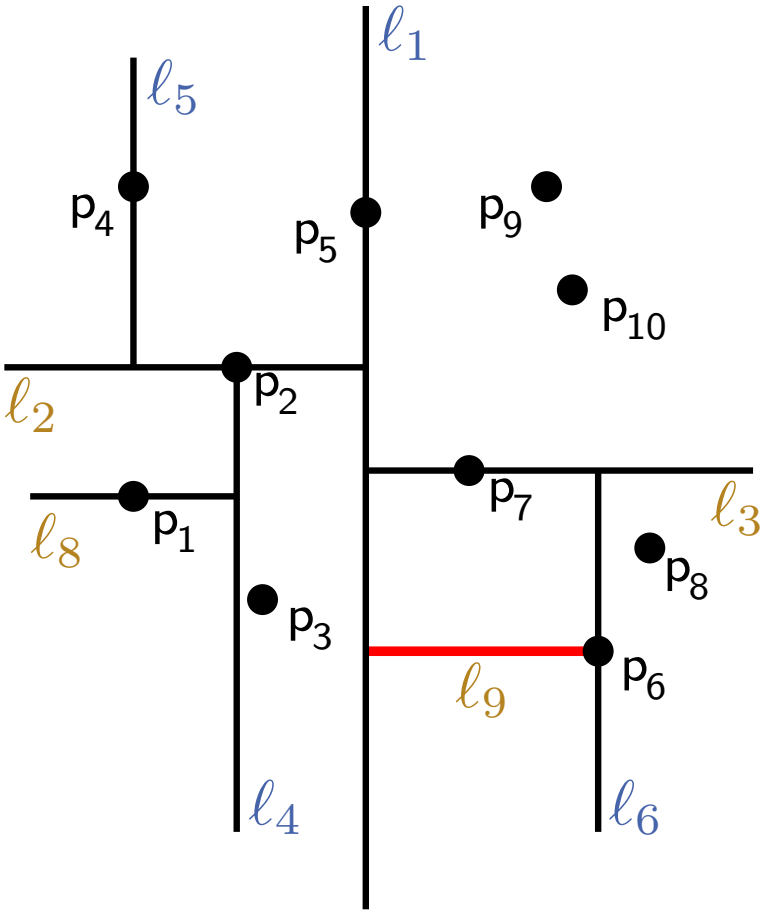


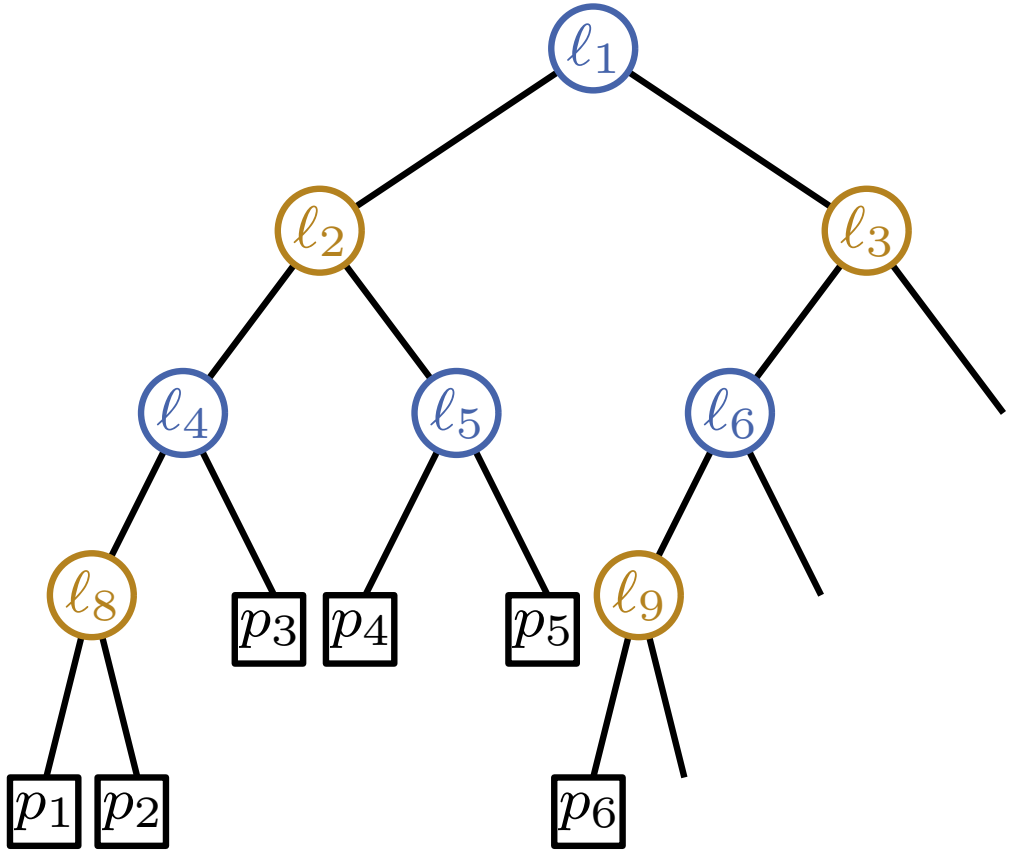
kd-Trees: Example



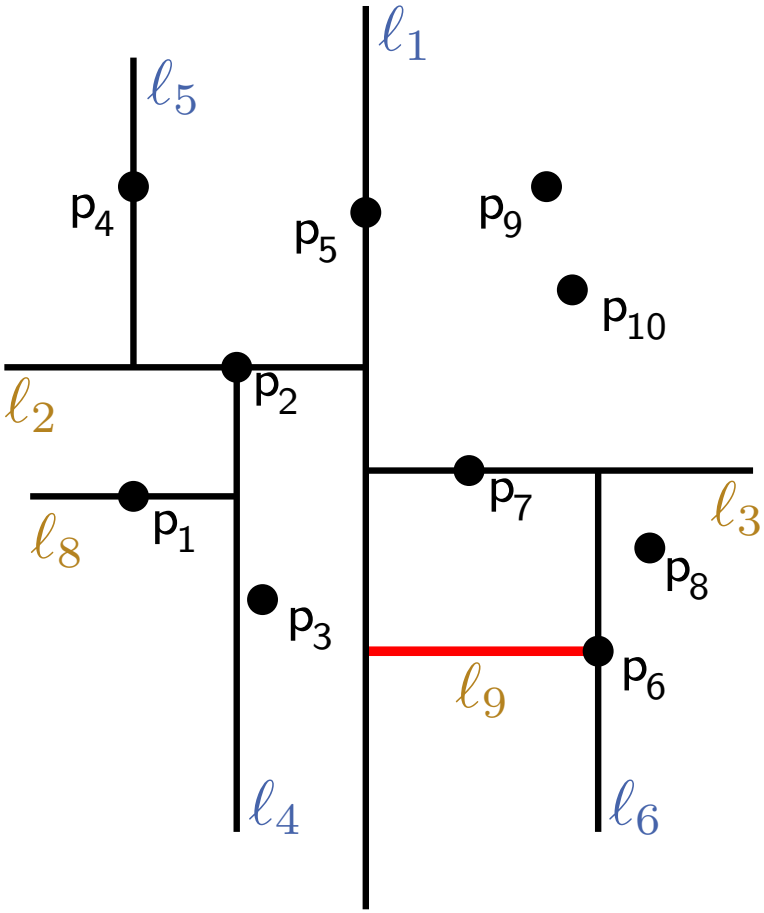


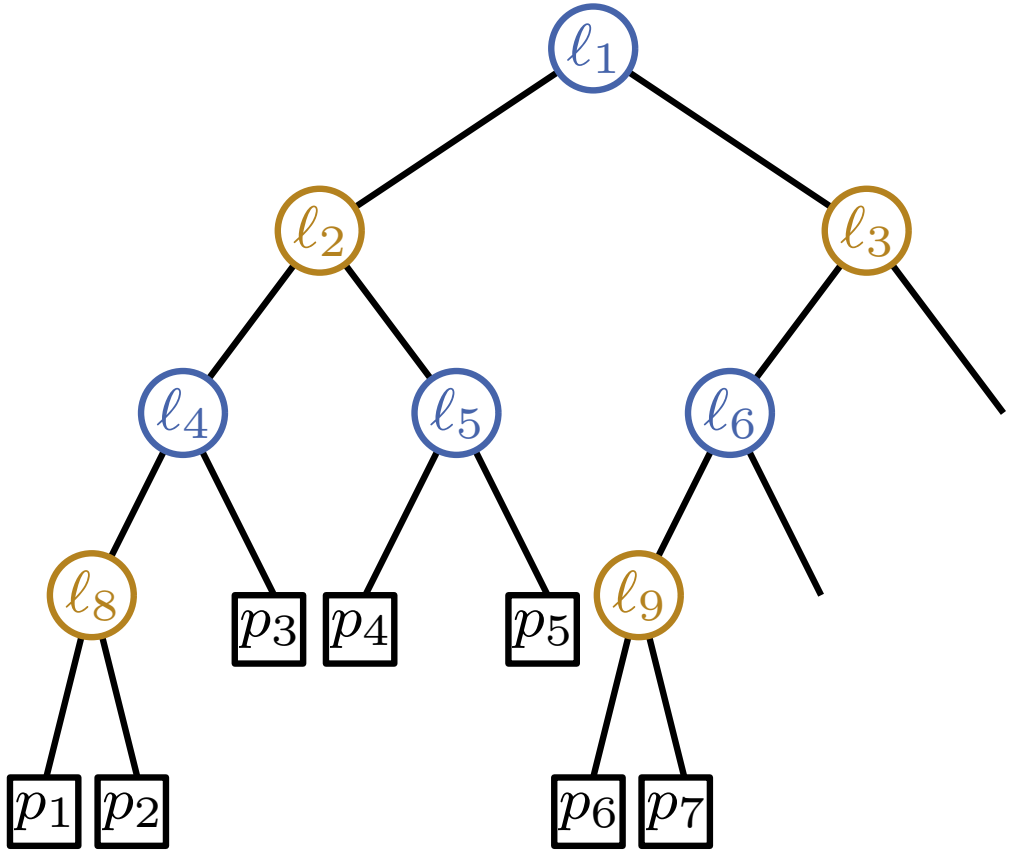
kd-Trees: Example



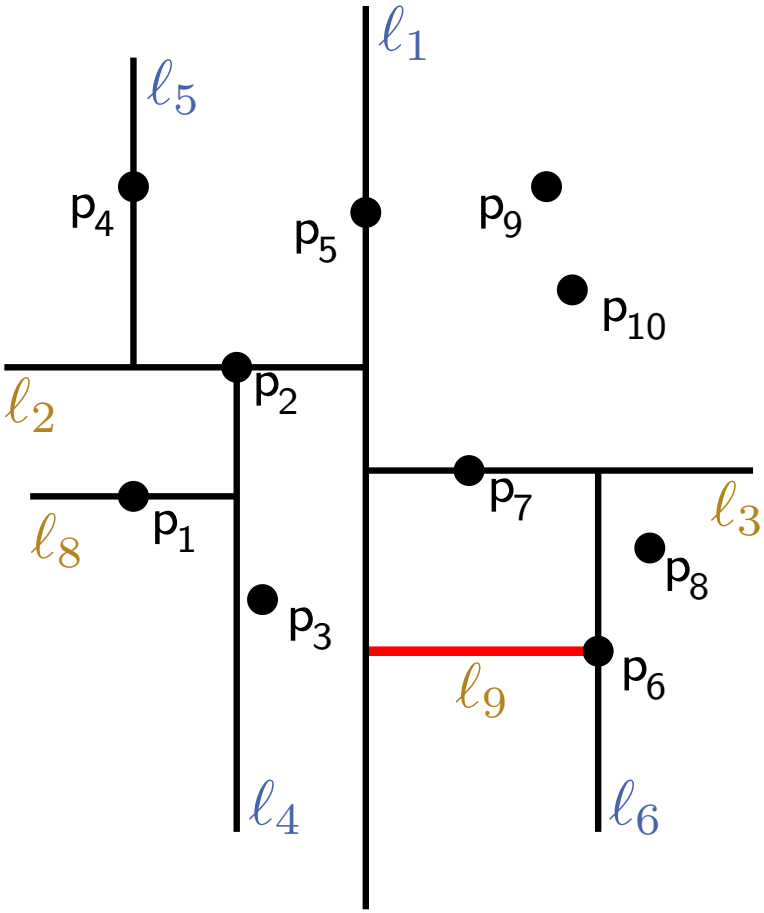


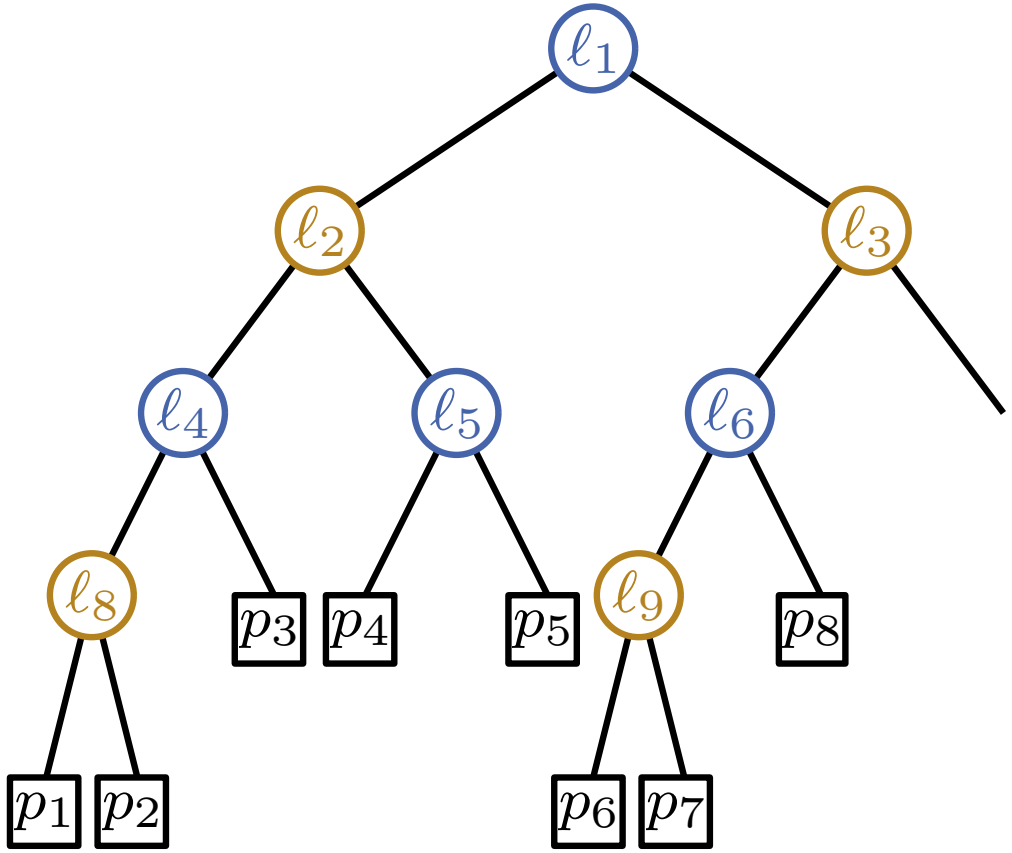
kd-Trees: Example



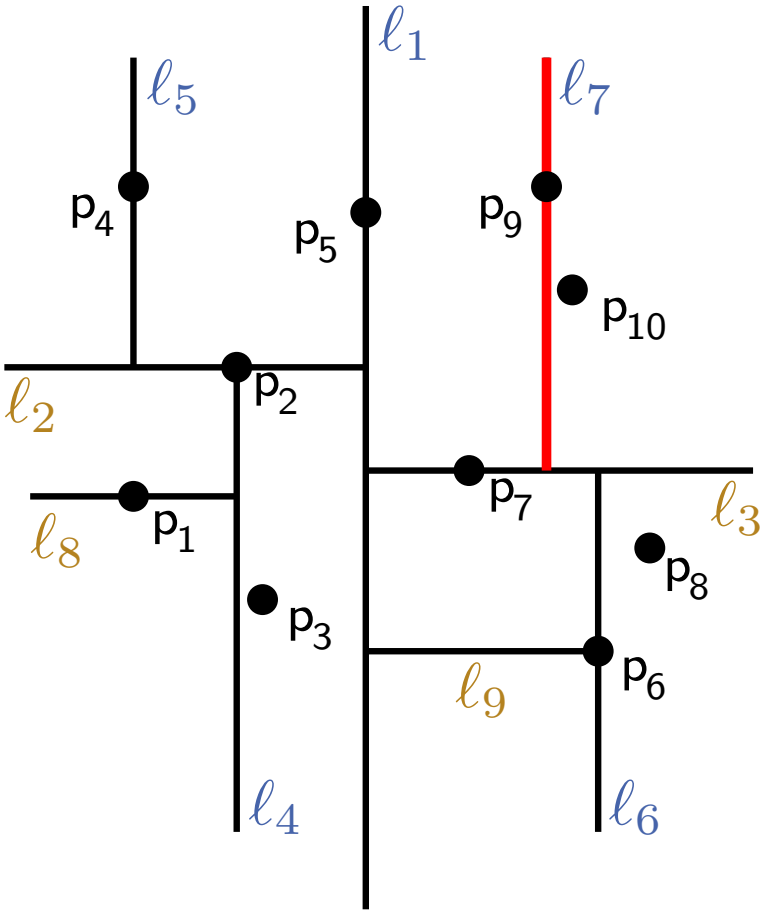


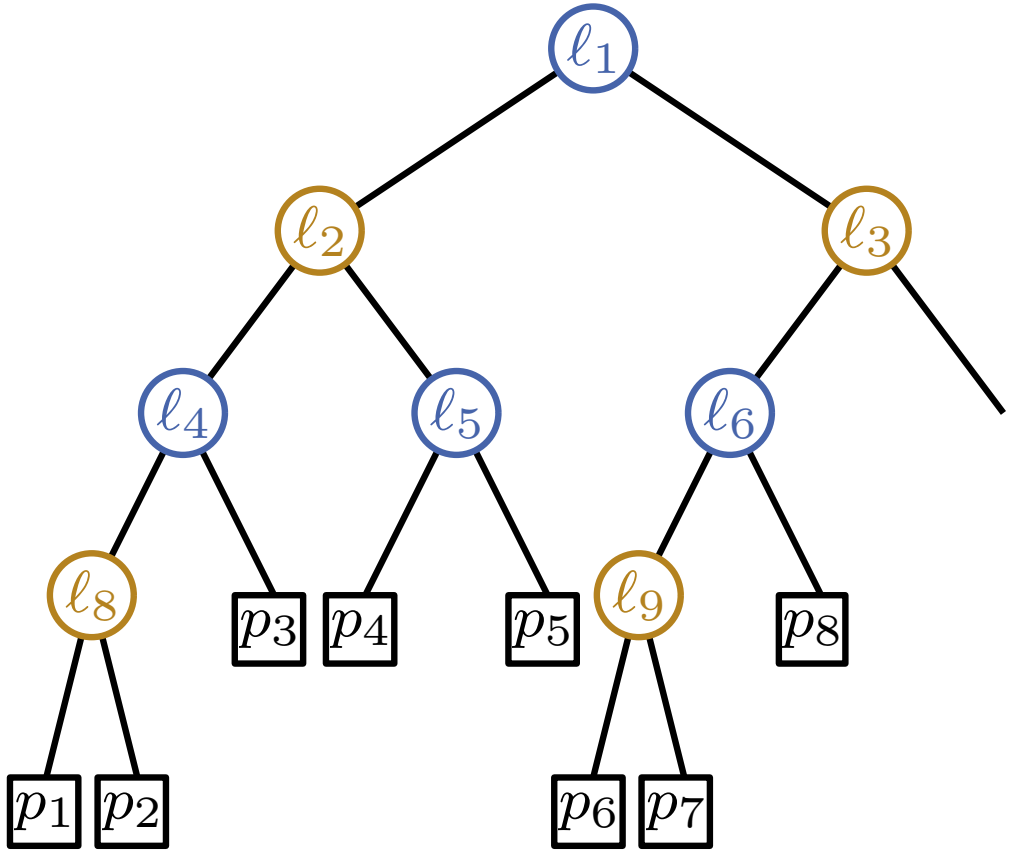
kd-Trees: Example



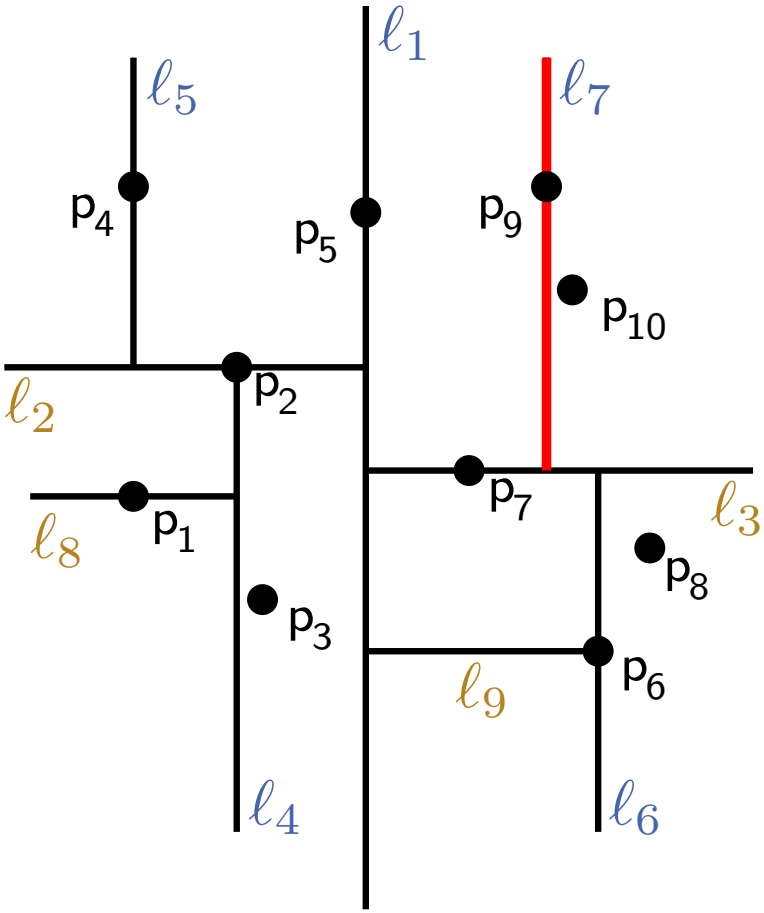


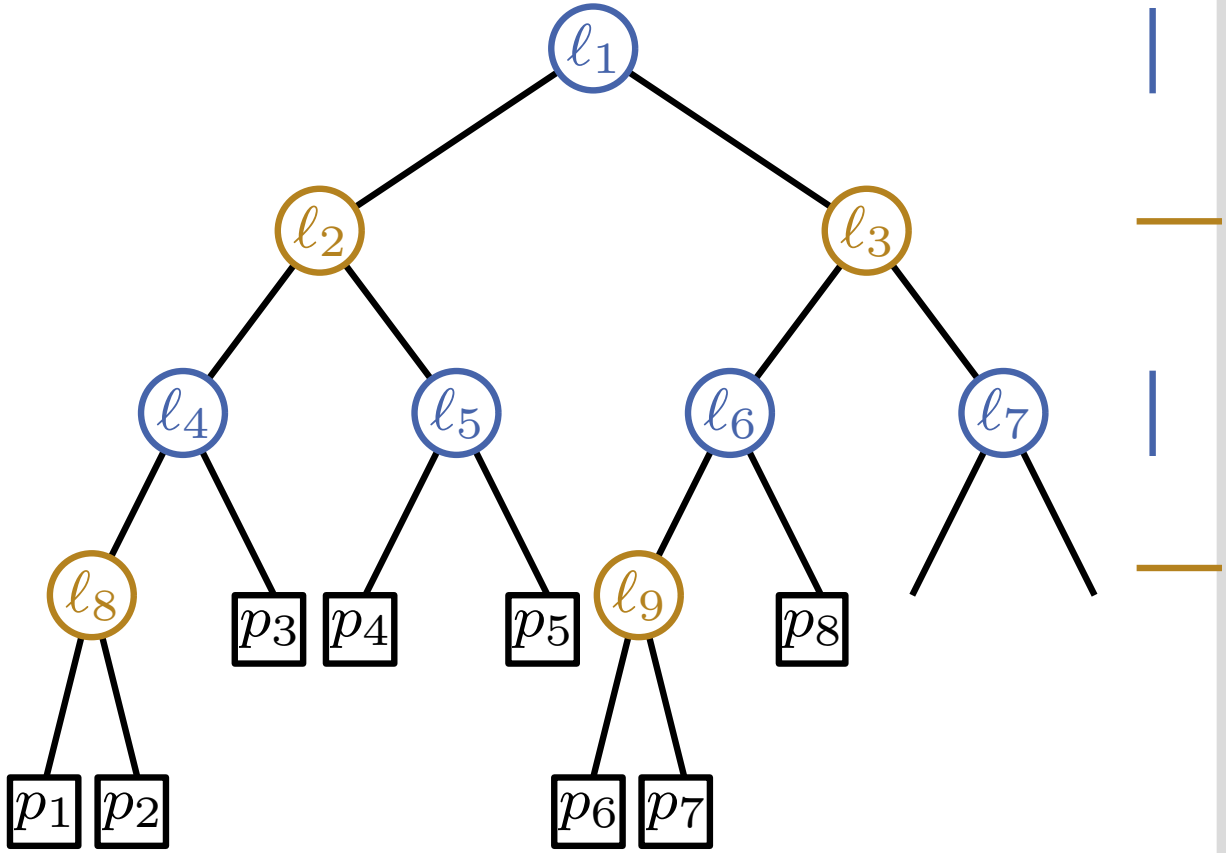
kd-Trees: Example



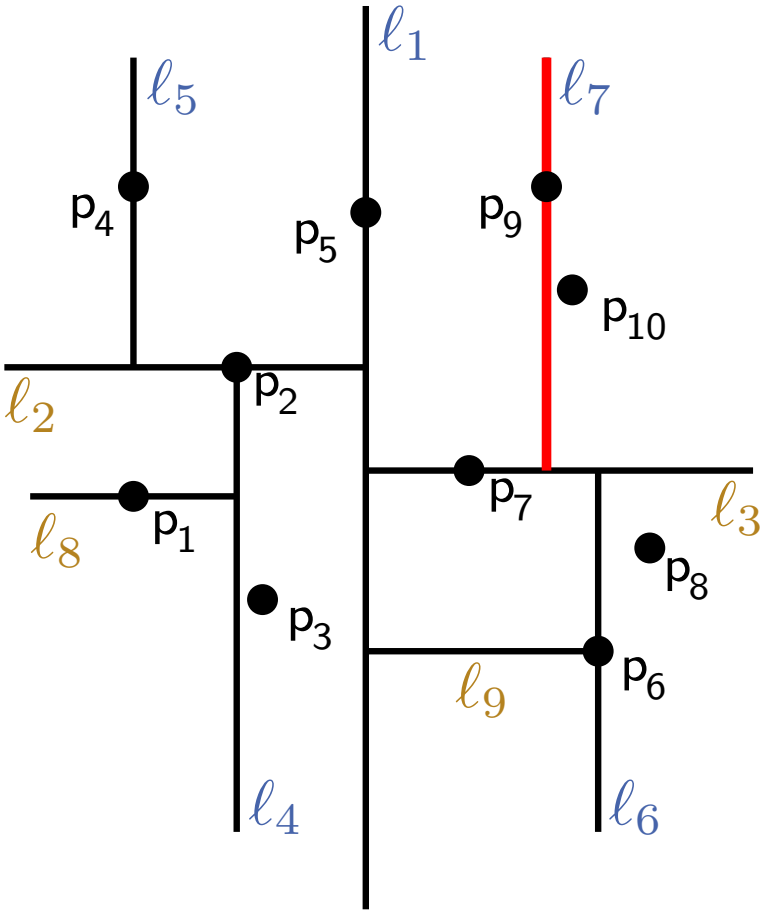


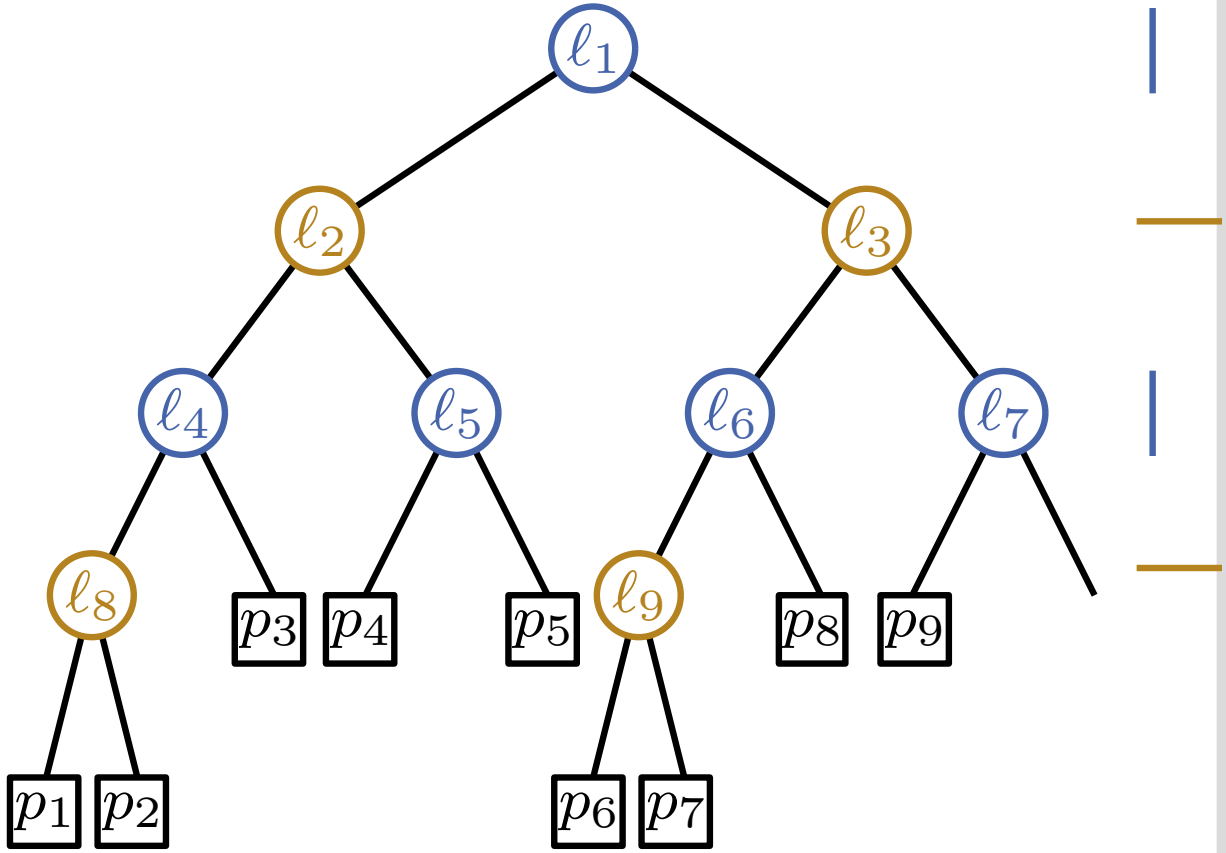
kd-Trees: Example



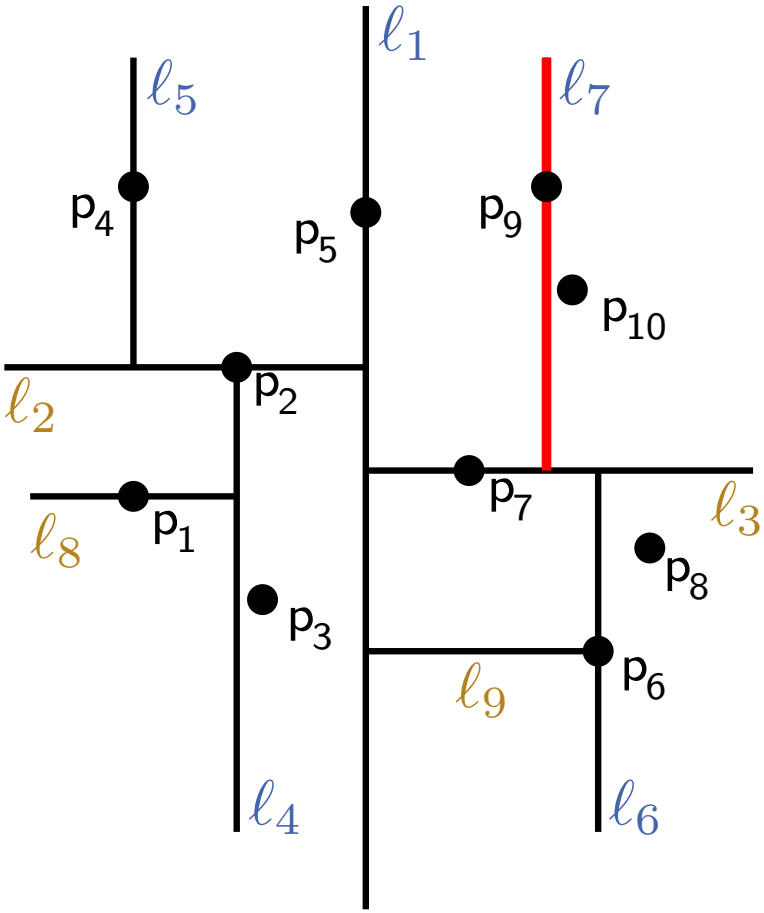


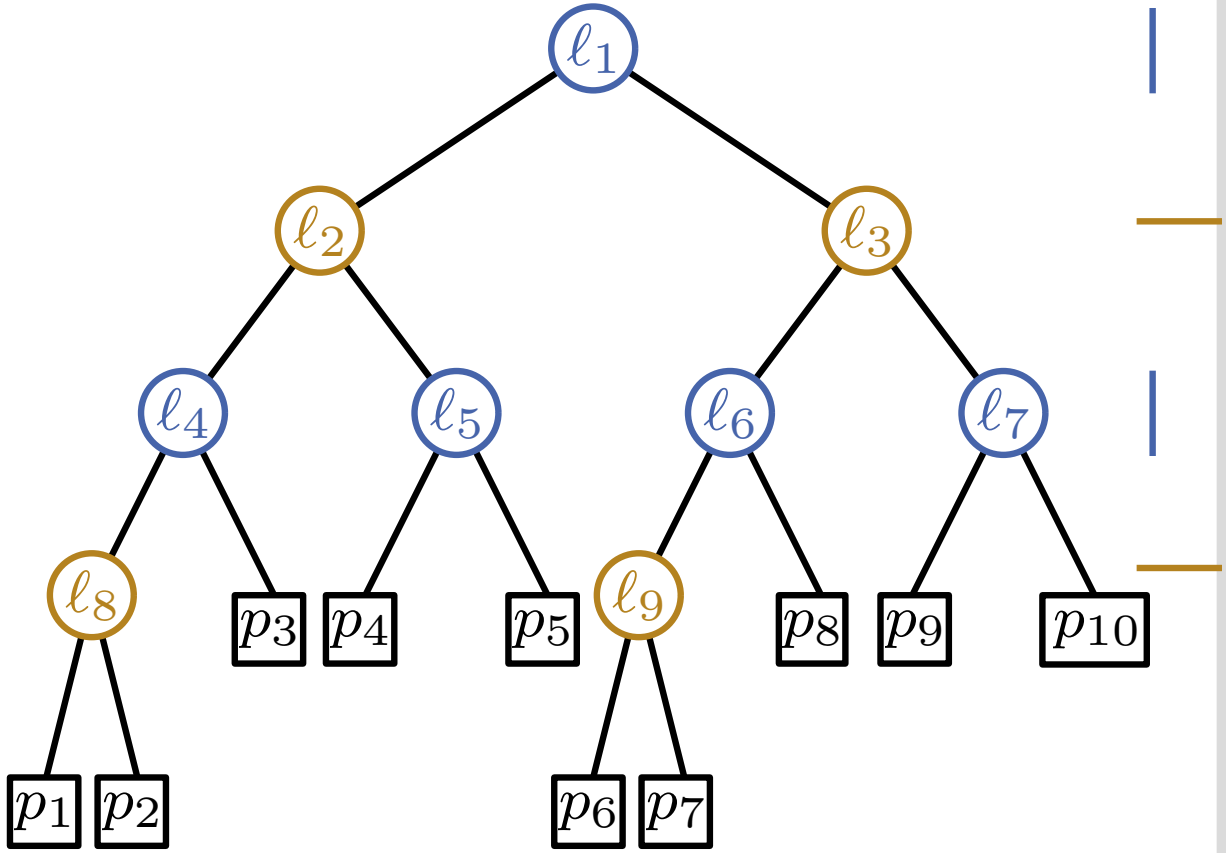
kd-Trees: Example



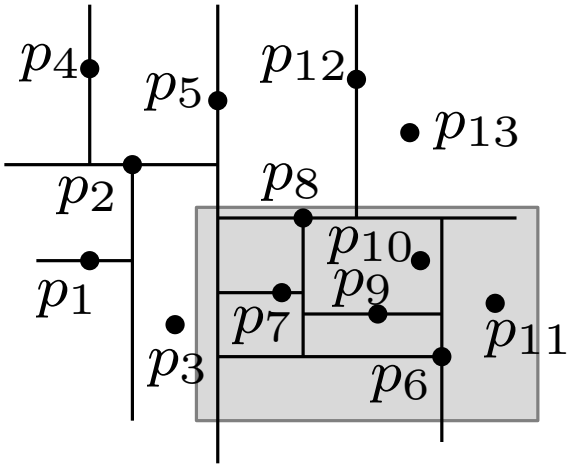


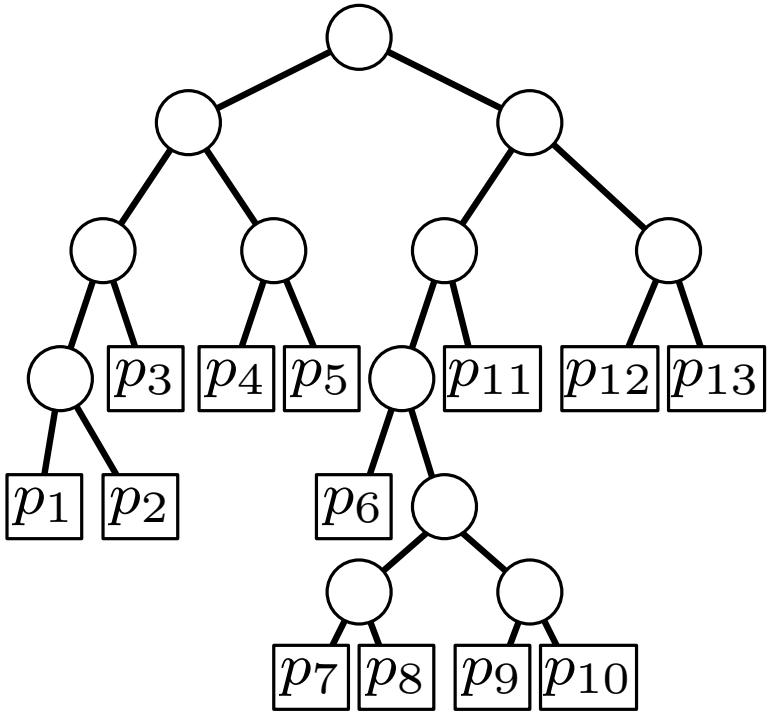
kd-Trees: Example



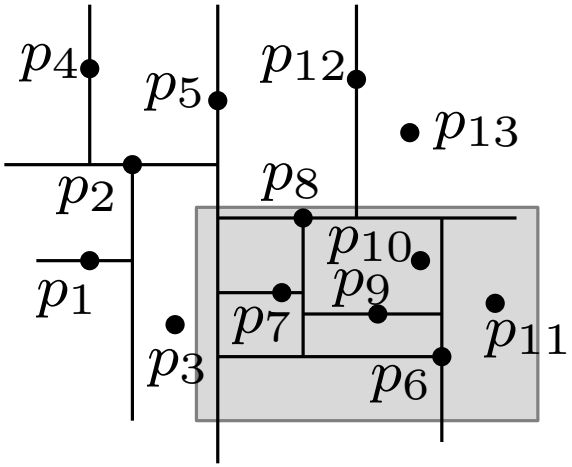


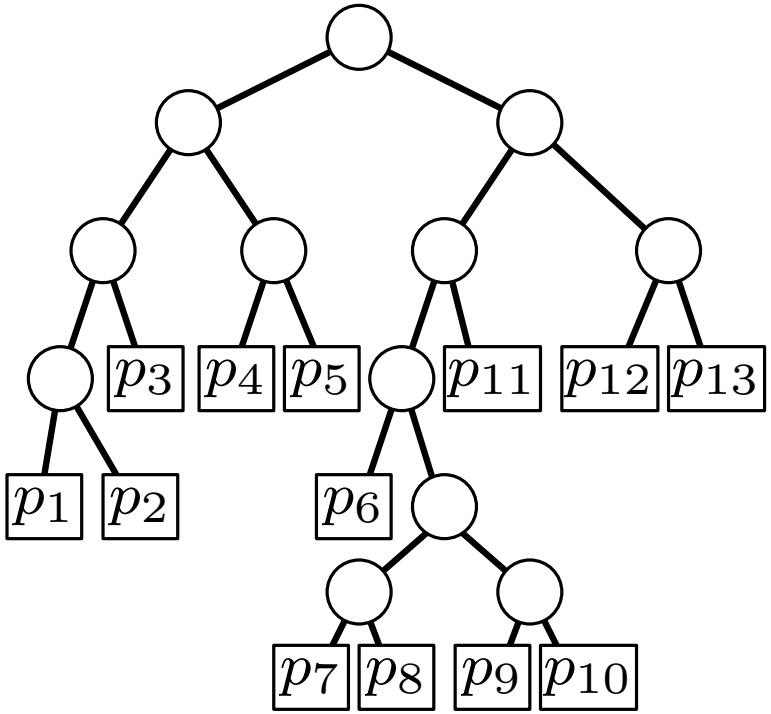
Range Queries in a kd-Tree





Range Queries in a kd-Tree

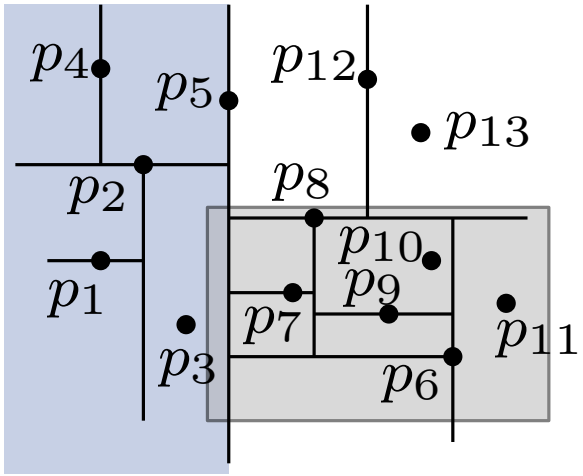


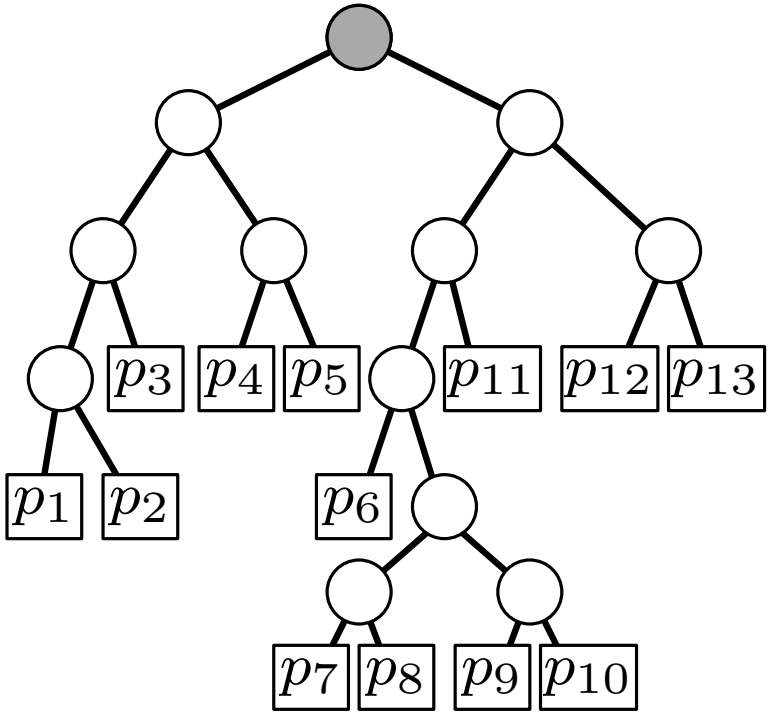


SearchKdTree(v, R)

if v leaf **then**
  report point p in v when $p \in R$
else
  if region(lc(v)) $\subseteq R$ **then**
    ReportSubtree(lc(v))
  else
    if region(lc(v))$\cap R \neq \emptyset$ **then**
      SearchKdTree(lc(v), R)
  if region(rc(v)) $\subseteq R$ **then**
    ReportSubtree(rc(v))
  else
    if region(rc(v))$\cap R \neq \emptyset$ **then**
      SearchKdTree(rc(v), R)

Range Queries in a kd-Tree

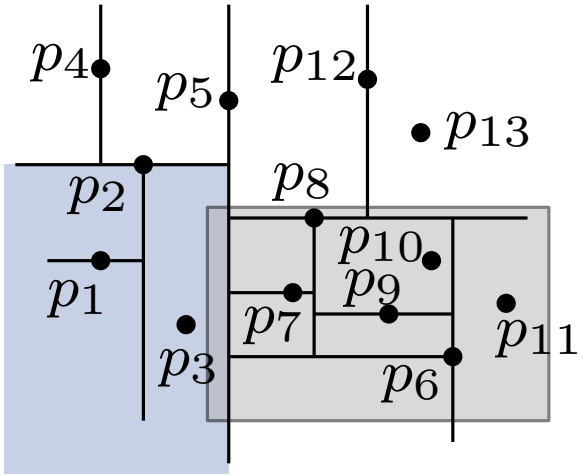


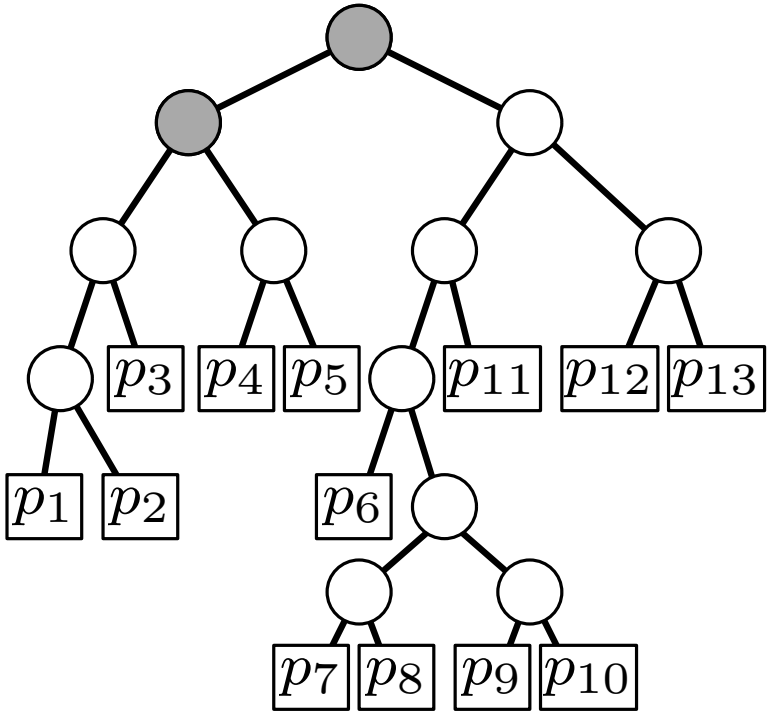


SearchKdTree(v, R)

if v leaf **then**
  report point p in v when $p \in R$
else
  if region(lc(v)) $\subseteq R$ **then**
    ReportSubtree(lc(v))
  else
    if region(lc(v))$\cap R \neq \emptyset$ **then**
      SearchKdTree(lc(v), R)
  if region(rc(v)) $\subseteq R$ **then**
    ReportSubtree(rc(v))
  else
    if region(rc(v))$\cap R \neq \emptyset$ **then**
      SearchKdTree(rc(v), R)

Range Queries in a kd-Tree

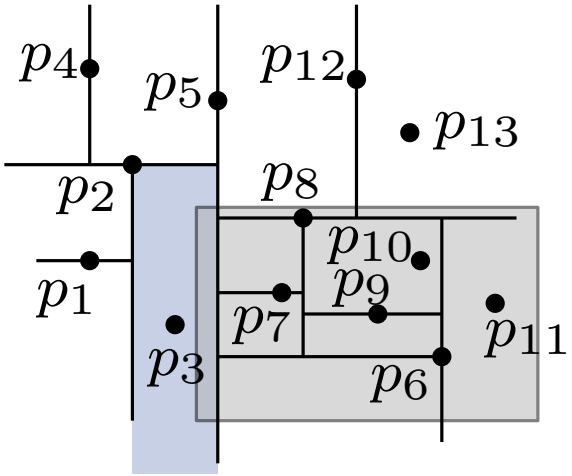


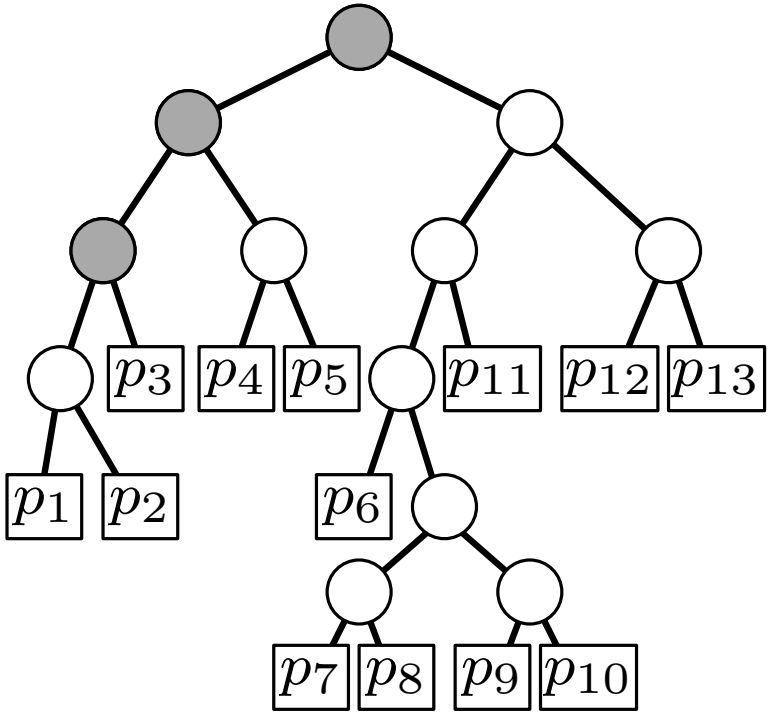


SearchKdTree(v, R)

if v leaf **then**
  report point p in v when $p \in R$
else
  if region(lc(v)) $\subseteq R$ **then**
    ReportSubtree(lc(v))
  else
    if region(lc(v))$\cap R \neq \emptyset$ **then**
      SearchKdTree(lc(v), R)
  if region(rc(v)) $\subseteq R$ **then**
    ReportSubtree(rc(v))
  else
    if region(rc(v))$\cap R \neq \emptyset$ **then**
      SearchKdTree(rc(v), R)

Range Queries in a kd-Tree

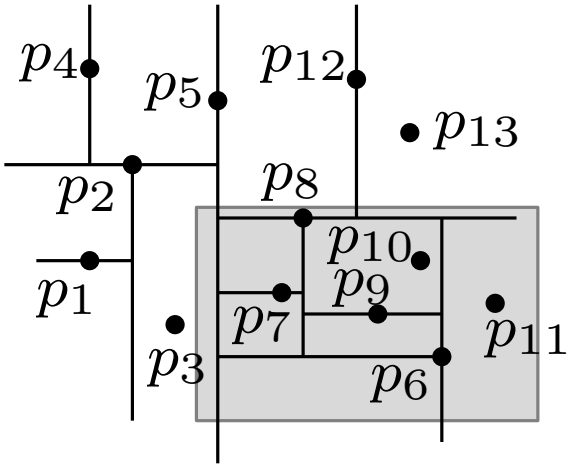


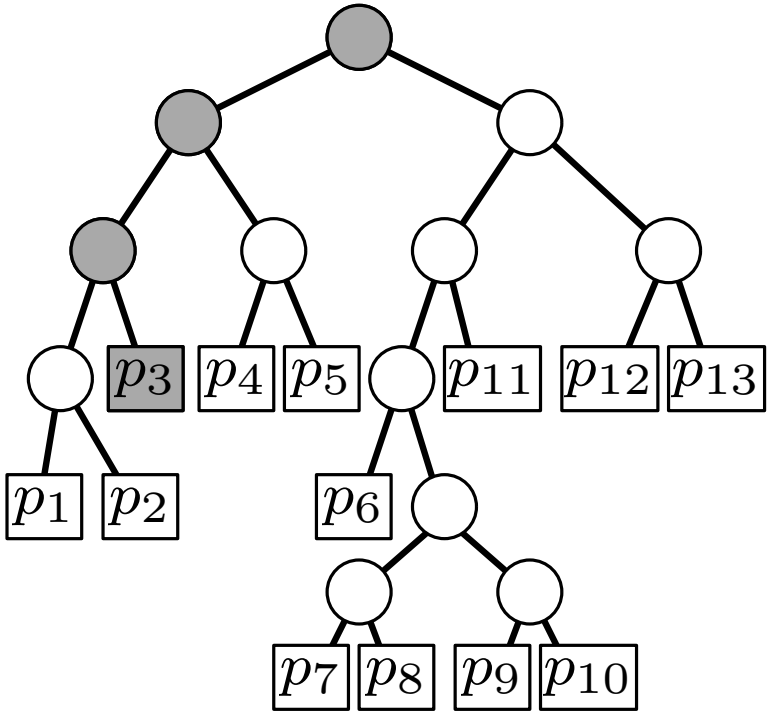


SearchKdTree(v, R)

- **if** v leaf **then**
  - report point p in v when $p \in R$
- **else**
  - **if** region(lc(v)) $\subseteq R$ **then**
    - ReportSubtree(lc(v))
  - **else**
    - **if** region(lc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(lc(v), R)
  - **if** region(rc(v)) $\subseteq R$ **then**
    - ReportSubtree(rc(v))
  - **else**
    - **if** region(rc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(rc(v), R)

Range Queries in a kd-Tree

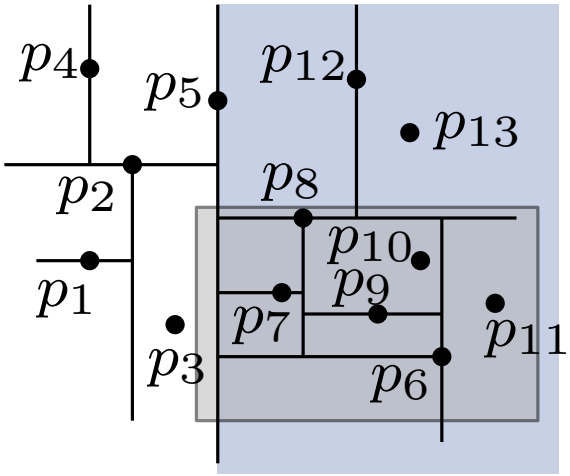


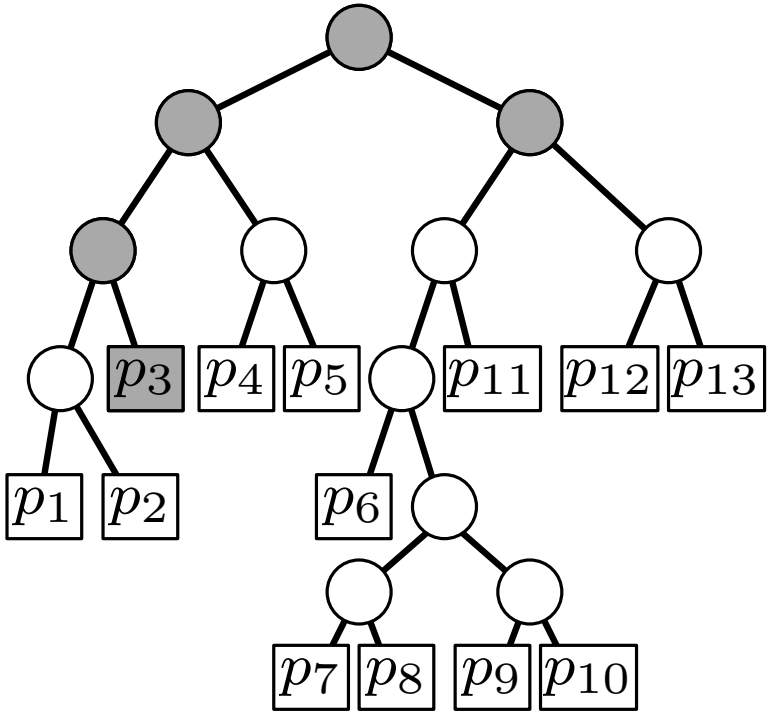


SearchKdTree(v, R)

- **if** v leaf **then**
  - report point p in v when $p \in R$
- **else**
  - **if** region(lc(v)) $\subseteq R$ **then**
    - ReportSubtree(lc(v))
  - **else**
    - **if** region(lc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(lc(v), R)
  - **if** region(rc(v)) $\subseteq R$ **then**
    - ReportSubtree(rc(v))
  - **else**
    - **if** region(rc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(rc(v), R)

Range Queries in a kd-Tree

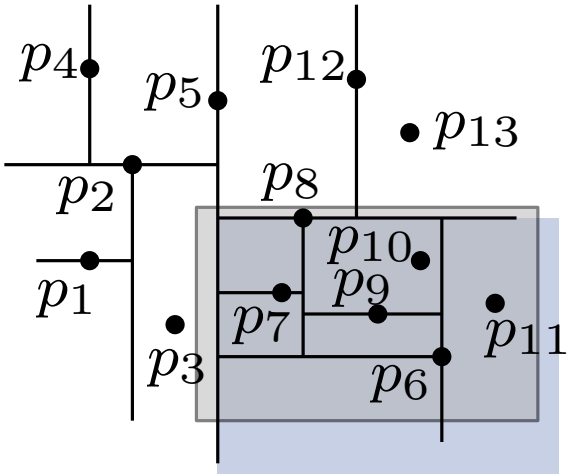


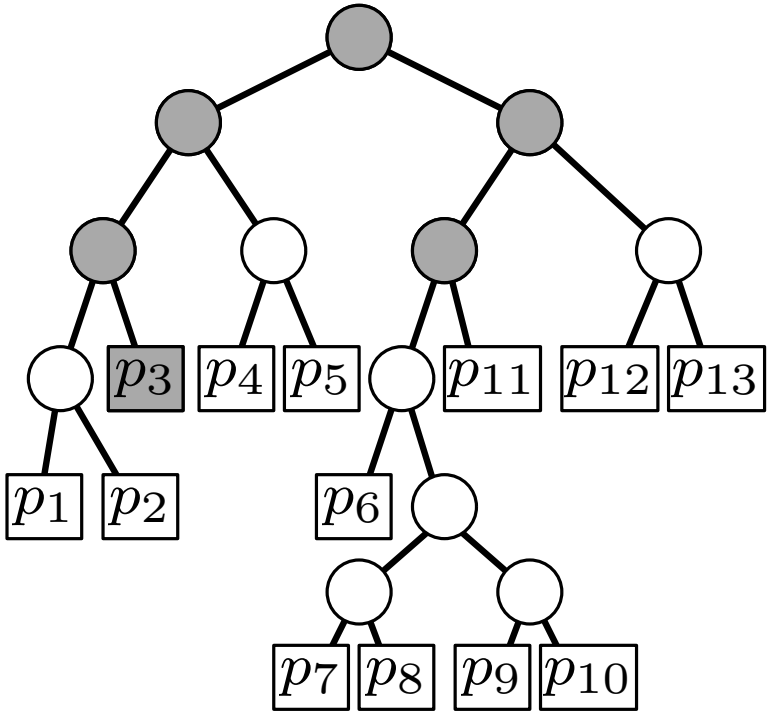


SearchKdTree(v, R)

- **if** v leaf **then**
  - report point p in v when $p \in R$
- **else**
  - **if** region(lc(v)) $\subseteq R$ **then**
    - ReportSubtree(lc(v))
  - **else**
    - **if** region(lc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(lc(v), R)
  - **if** region(rc(v)) $\subseteq R$ **then**
    - ReportSubtree(rc(v))
  - **else**
    - **if** region(rc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(rc(v), R)

Range Queries in a kd-Tree

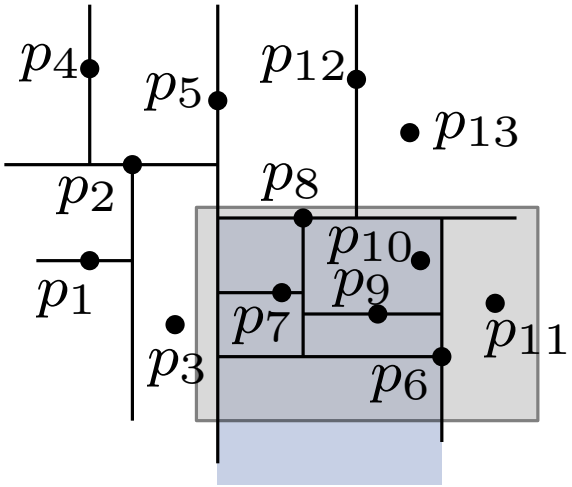


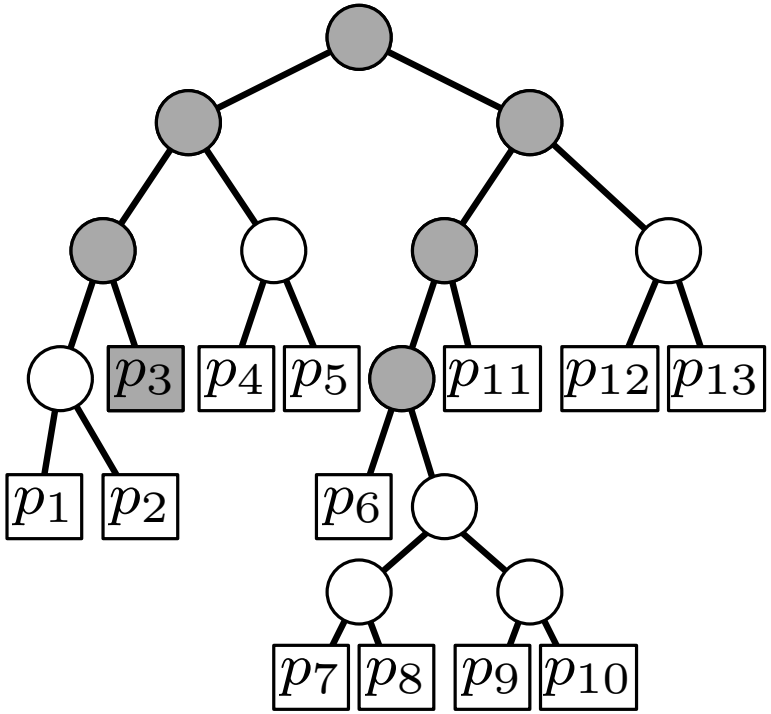


SearchKdTree(v, R)

- **if** v leaf **then**
  - report point p in v when $p \in R$
- **else**
  - **if** region(lc(v)) $\subseteq R$ **then**
    - ReportSubtree(lc(v))
  - **else**
    - **if** region(lc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(lc(v), R)
  - **if** region(rc(v)) $\subseteq R$ **then**
    - ReportSubtree(rc(v))
  - **else**
    - **if** region(rc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(rc(v), R)

Range Queries in a kd-Tree

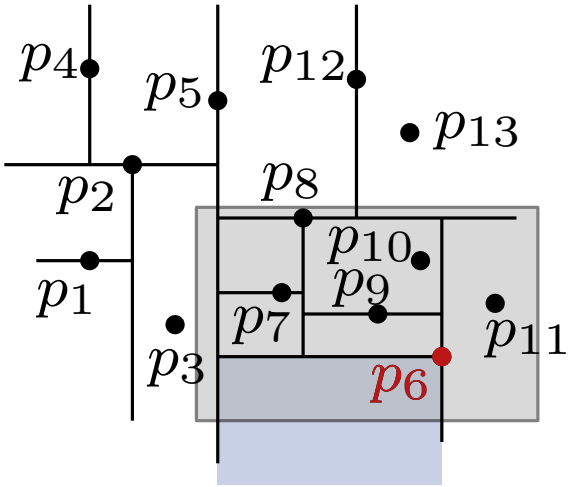


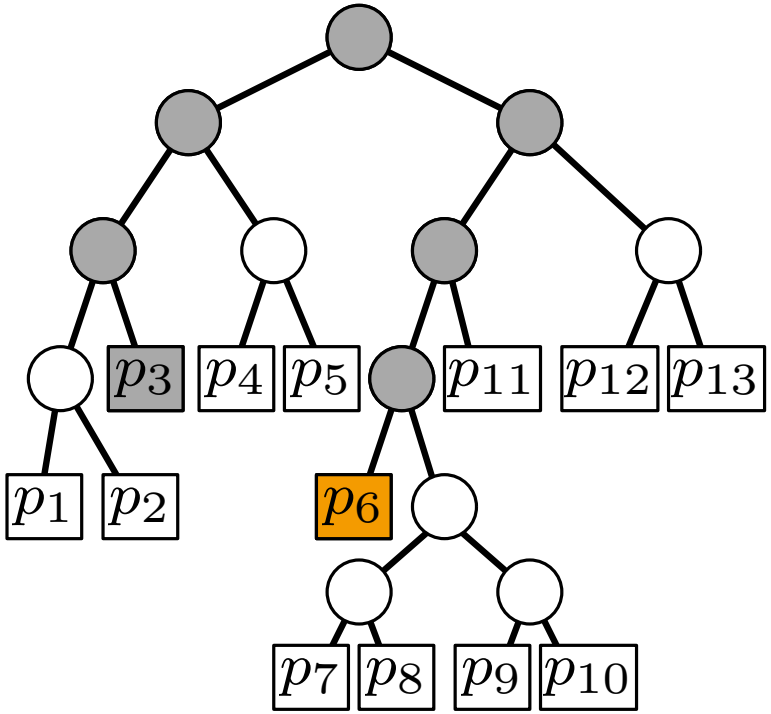


SearchKdTree(v, R)

- **if** v leaf **then**
  - report point p in v when $p \in R$
- **else**
  - **if** region(lc(v)) $\subseteq R$ **then**
    - ReportSubtree(lc(v))
  - **else**
    - **if** region(lc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(lc(v), R)
  - **if** region(rc(v)) $\subseteq R$ **then**
    - ReportSubtree(rc(v))
  - **else**
    - **if** region(rc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(rc(v), R)

Range Queries in a kd-Tree

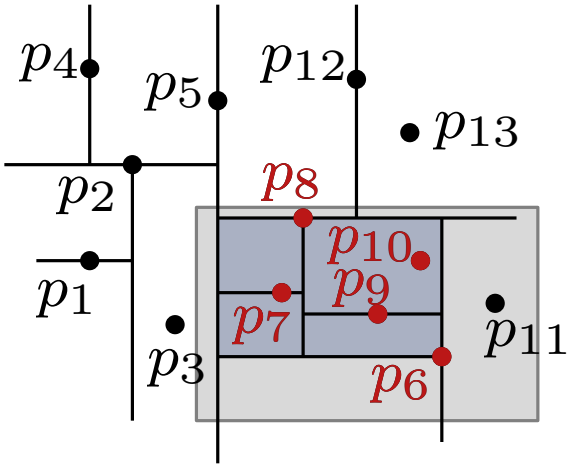


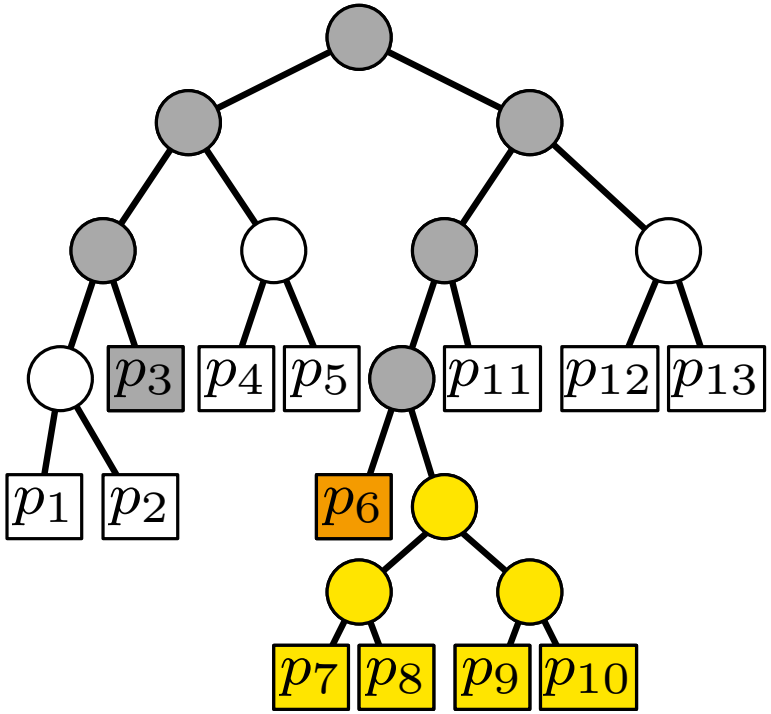


SearchKdTree(v, R)

- **if** v leaf **then**
  - report point p in v when $p \in R$
- **else**
  - **if** region(lc(v)) $\subseteq R$ **then**
    - ReportSubtree(lc(v))
  - **else**
    - **if** region(lc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(lc(v), R)
  - **if** region(rc(v)) $\subseteq R$ **then**
    - ReportSubtree(rc(v))
  - **else**
    - **if** region(rc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(rc(v), R)

Range Queries in a kd-Tree

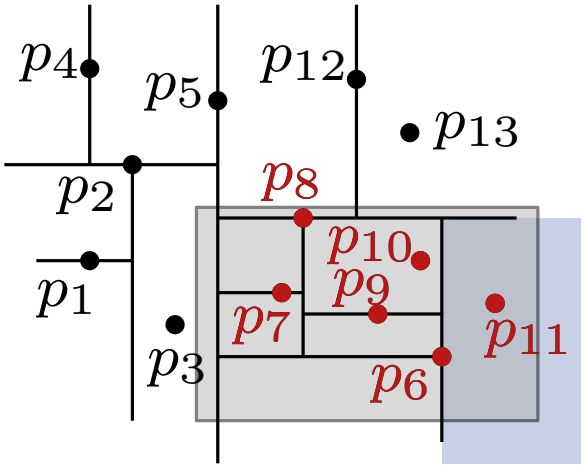


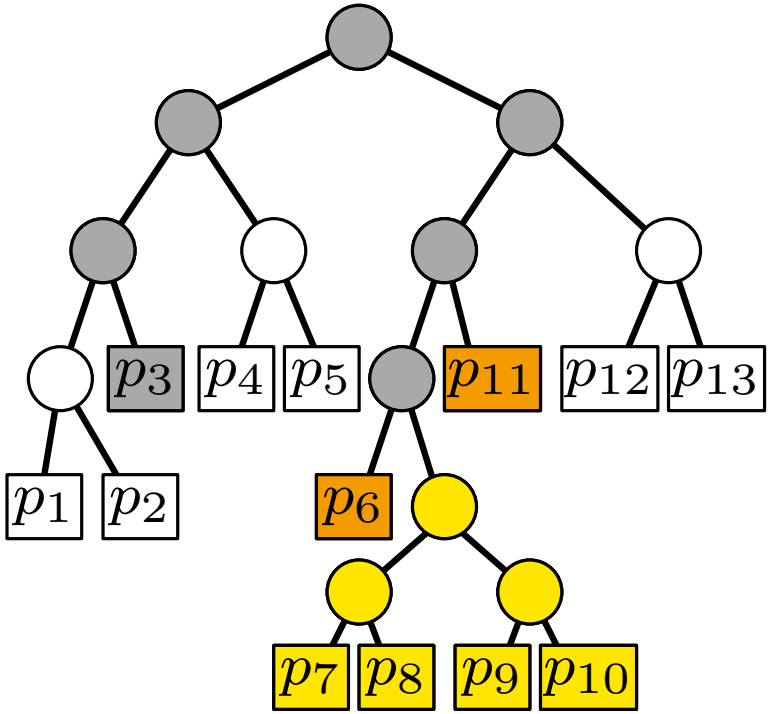


SearchKdTree(v, R)

if v leaf **then**
  report point p in v when $p \in R$
else
  if region(lc(v)) $\subseteq R$ **then**
    ReportSubtree(lc(v))
  else
    if region(lc(v))$\cap R \neq \emptyset$ **then**
      SearchKdTree(lc(v), R)
  if region(rc(v)) $\subseteq R$ **then**
    ReportSubtree(rc(v))
  else
    if region(rc(v))$\cap R \neq \emptyset$ **then**
      SearchKdTree(rc(v), R)

Range Queries in a kd-Tree

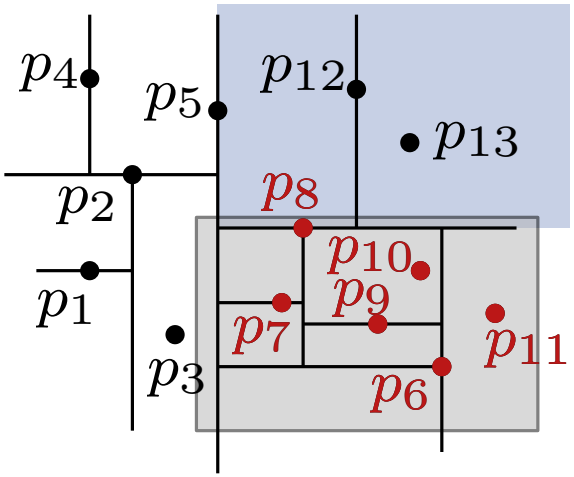


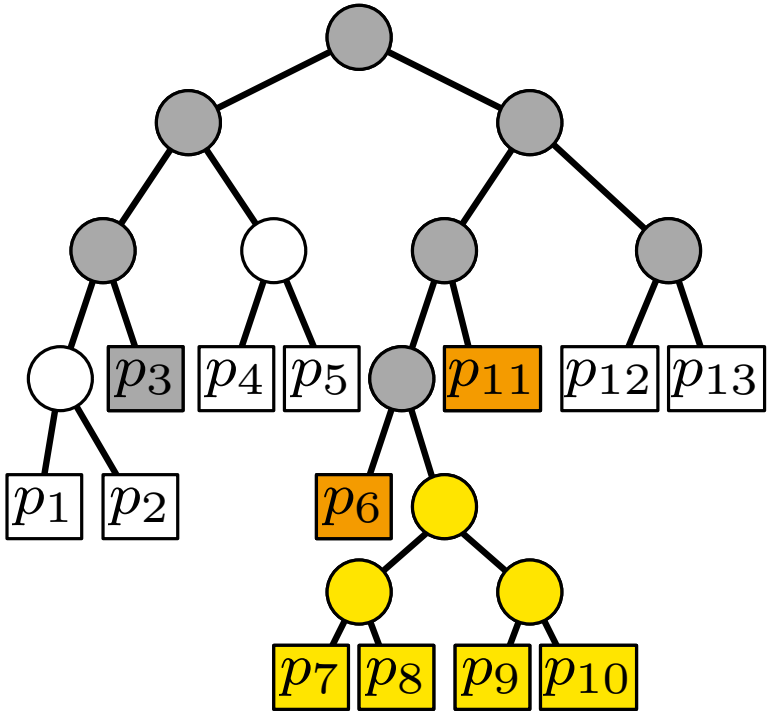


SearchKdTree(v, R)

- **if** v leaf **then**
  - report point p in v when $p \in R$
- **else**
  - **if** region(lc(v)) $\subseteq R$ **then**
    - ReportSubtree(lc(v))
  - **else**
    - **if** region(lc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(lc(v), R)
  - **if** region(rc(v)) $\subseteq R$ **then**
    - ReportSubtree(rc(v))
  - **else**
    - **if** region(rc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(rc(v), R)

Range Queries in a kd-Tree

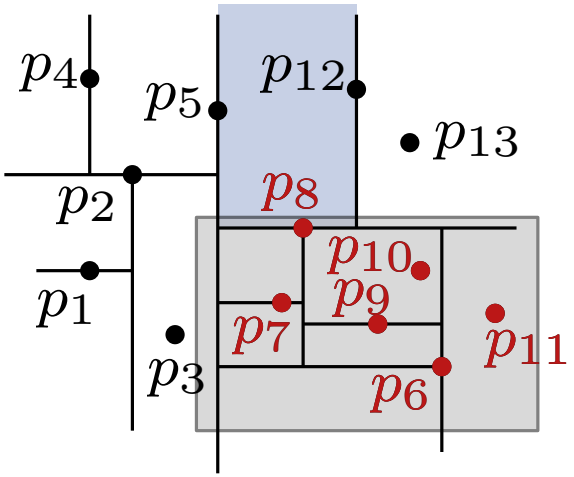


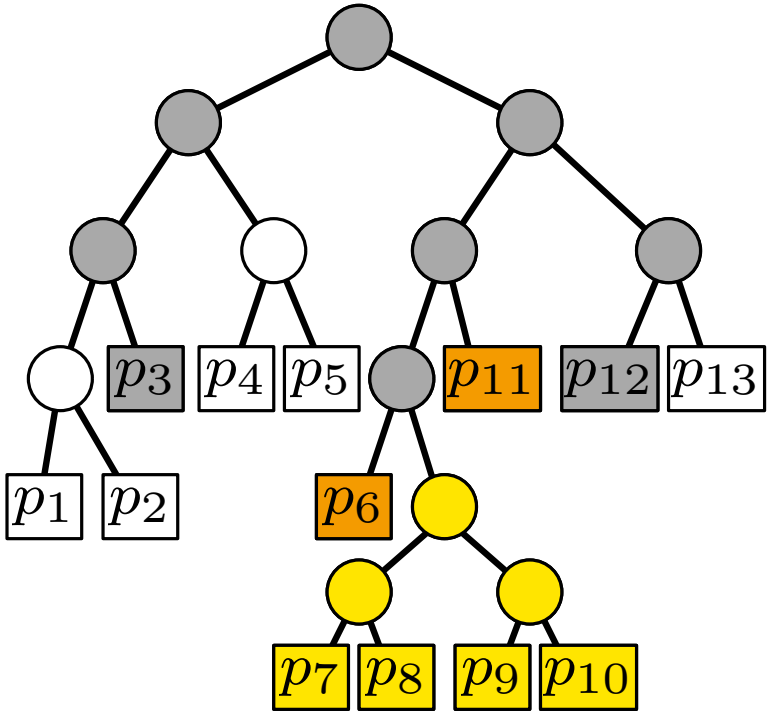


SearchKdTree(v, R)

- **if** v leaf **then**
  - report point p in v when $p \in R$
- **else**
  - **if** region(lc(v)) $\subseteq R$ **then**
    - ReportSubtree(lc(v))
  - **else**
    - **if** region(lc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(lc(v), R)
  - **if** region(rc(v)) $\subseteq R$ **then**
    - ReportSubtree(rc(v))
  - **else**
    - **if** region(rc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(rc(v), R)

Range Queries in a kd-Tree

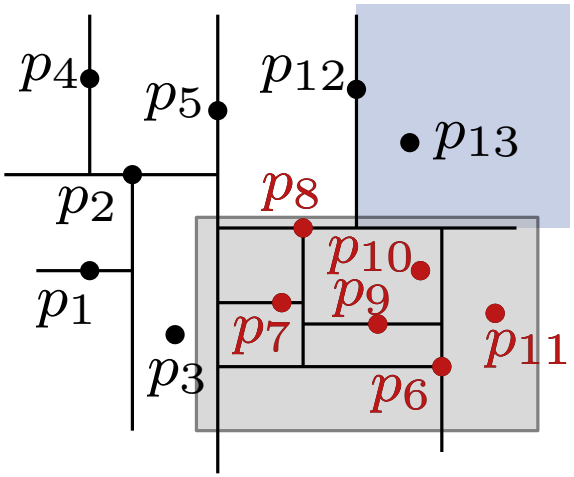


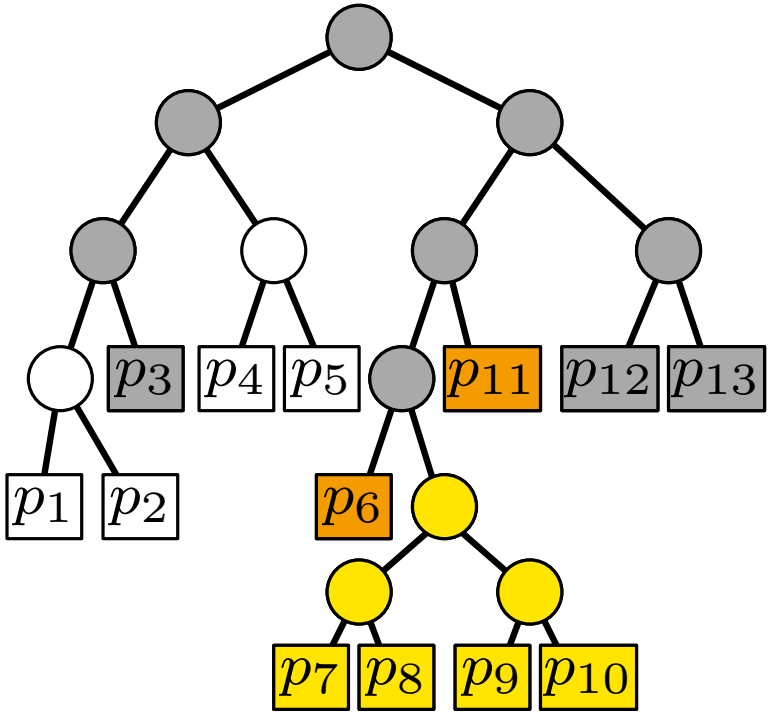


SearchKdTree(v, R)

if v leaf **then**
  report point p in v when $p \in R$
else
  if region(lc(v)) $\subseteq R$ **then**
    ReportSubtree(lc(v))
  else
    if region(lc(v))$\cap R \neq \emptyset$ **then**
      SearchKdTree(lc(v), R)
  if region(rc(v)) $\subseteq R$ **then**
    ReportSubtree(rc(v))
  else
    if region(rc(v))$\cap R \neq \emptyset$ **then**
      SearchKdTree(rc(v), R)

Range Queries in a kd-Tree





SearchKdTree(v, R)

- **if** v leaf **then**
  - report point p in v when $p \in R$
- **else**
  - **if** region(lc(v)) $\subseteq R$ **then**
    - ReportSubtree(lc(v))
  - **else**
    - **if** region(lc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(lc(v), R)
  - **if** region(rc(v)) $\subseteq R$ **then**
    - ReportSubtree(rc(v))
  - **else**
    - **if** region(rc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(rc(v), R)

Exercise 1

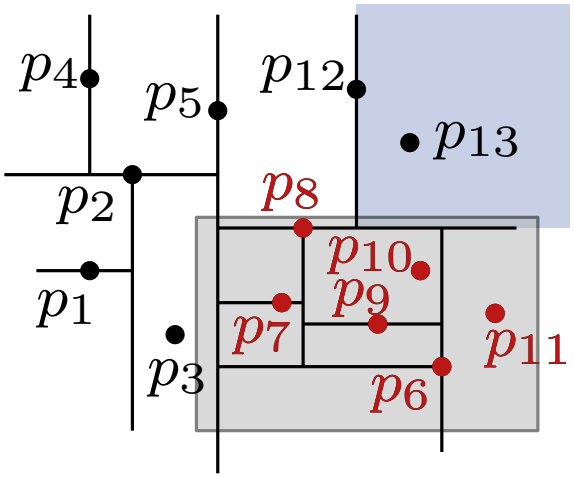
Claim: Queries in $O(\sqrt{n} + k)$ time

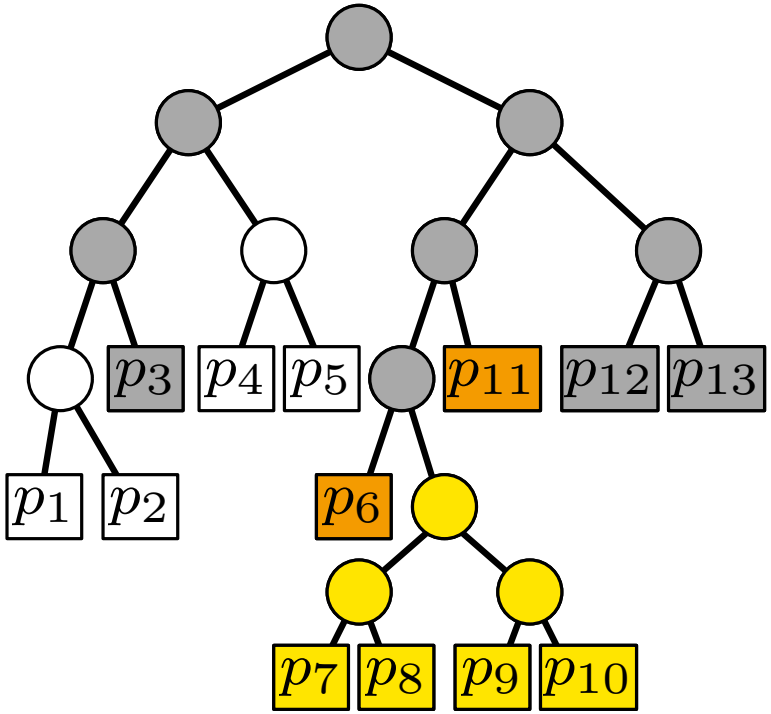
$Q(n) =$ Number of checked regions.

a) Show the following recurrence.

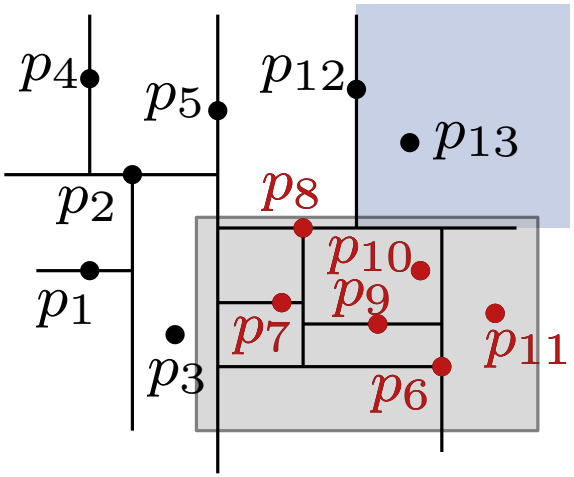
$$Q(n) = \begin{cases} \mathcal{O}(1) & , \text{ for } n = 1 \\ 2 + 2Q(n/4) & , \text{ for } n > 1 \end{cases}$$

Range Queries in kd-Trees

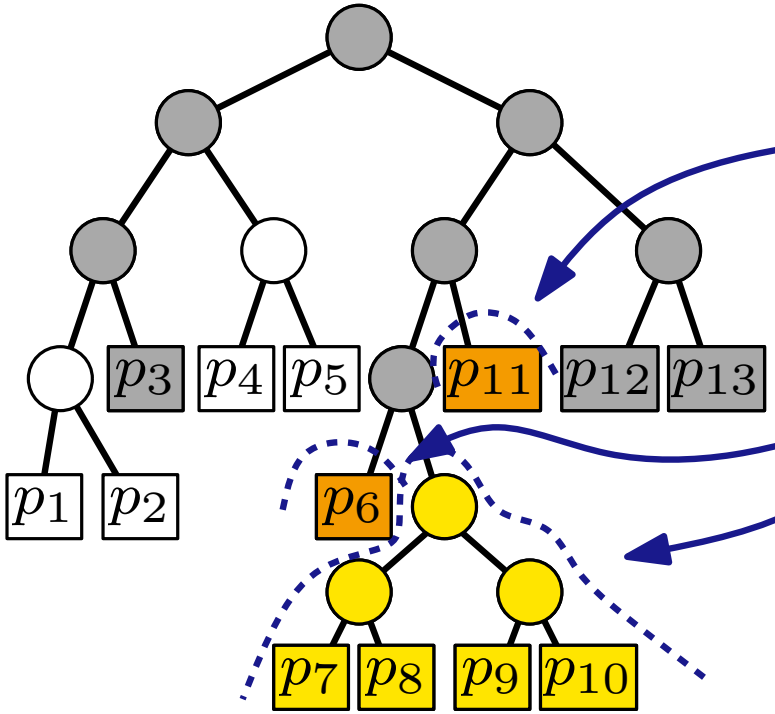




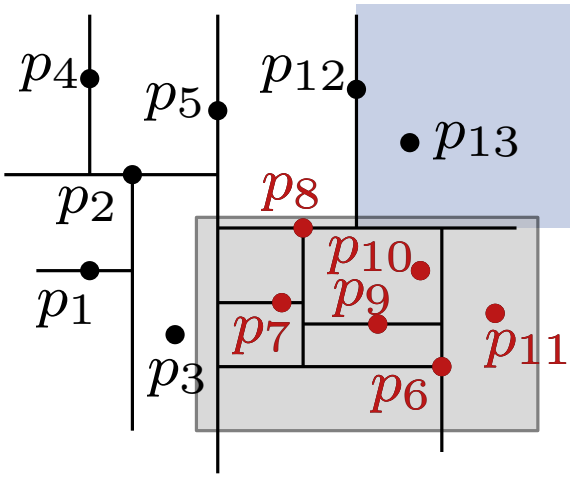
Range Queries in kd-Trees



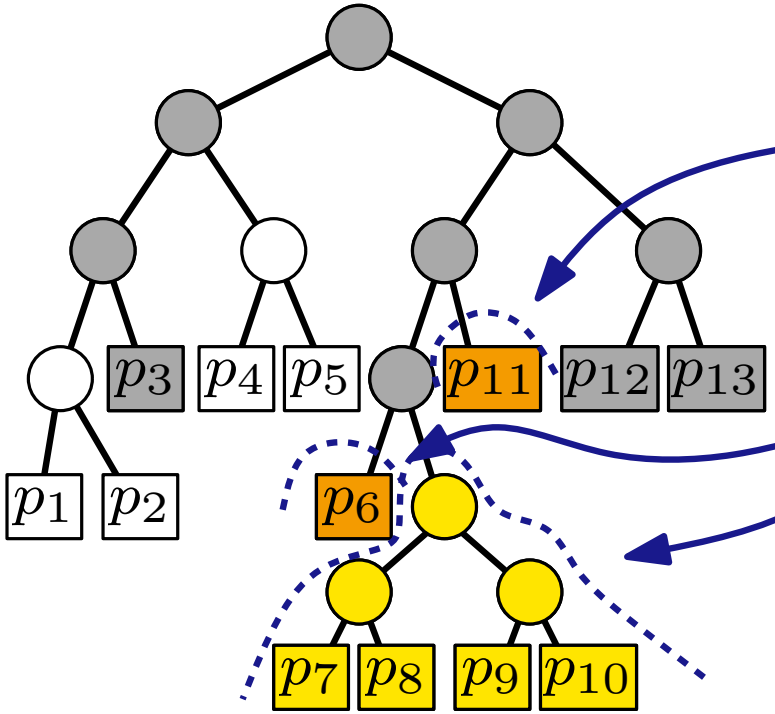
ReportSubtree in $\mathcal{O}(k)$



Range Queries in kd-Trees

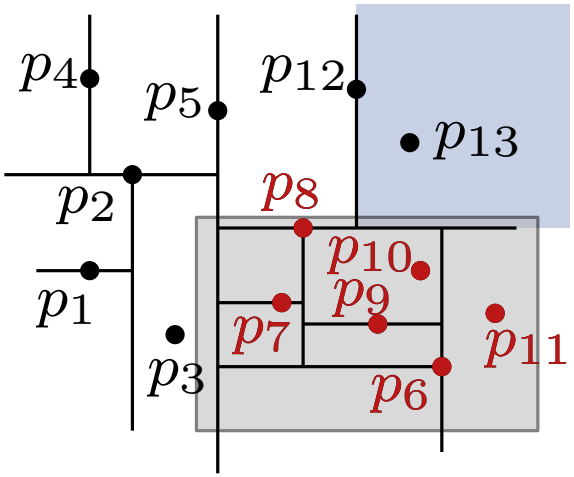


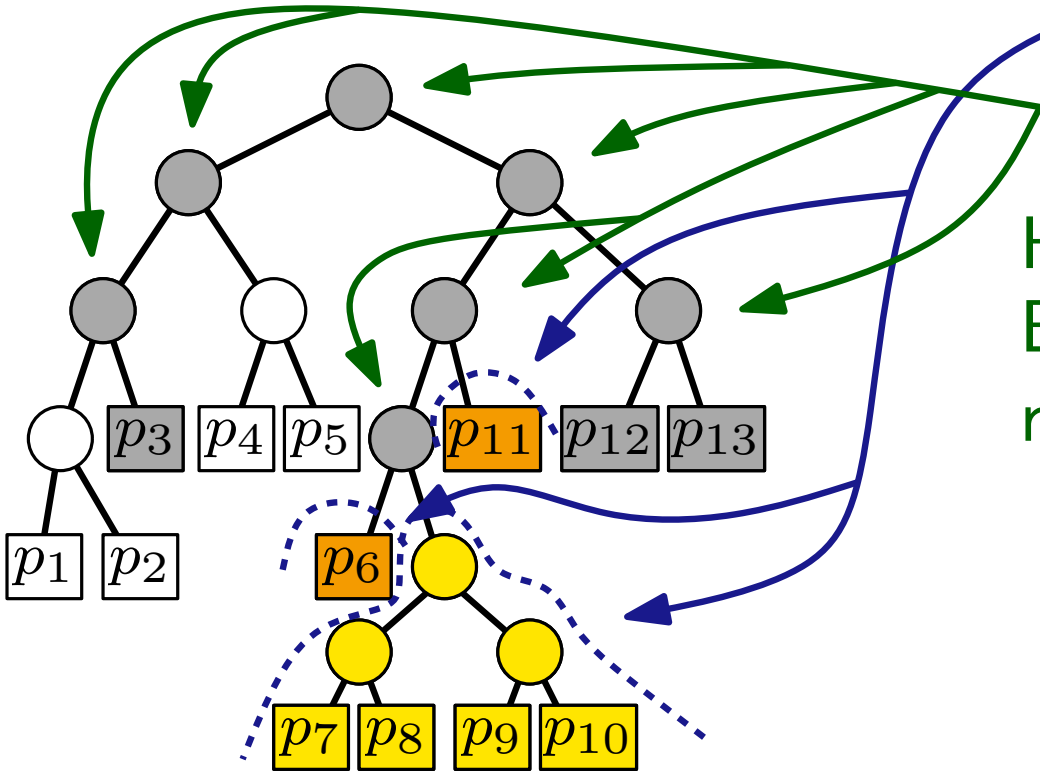
ReportSubtree in $\mathcal{O}(k)$



How many nodes do we consider?
Each node corresponds with one recursion step.

Range Queries in kd-Trees

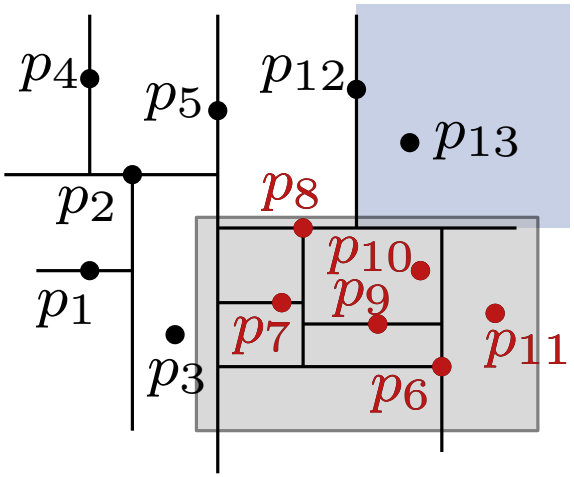




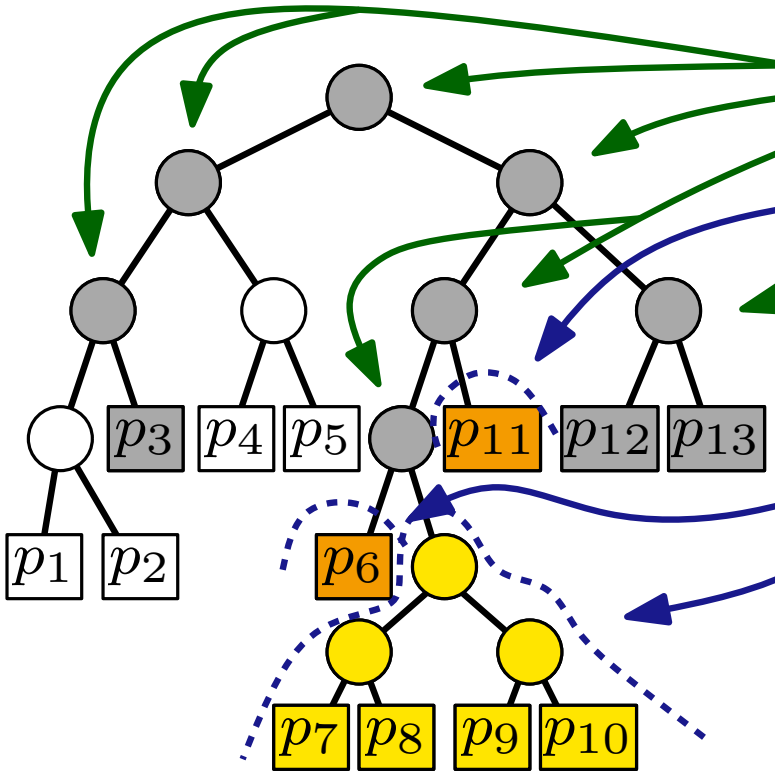
ReportSubtree in $\mathcal{O}(k)$

How many nodes do we consider? Each node corresponds with one recursion step.

Range Queries in kd-Trees



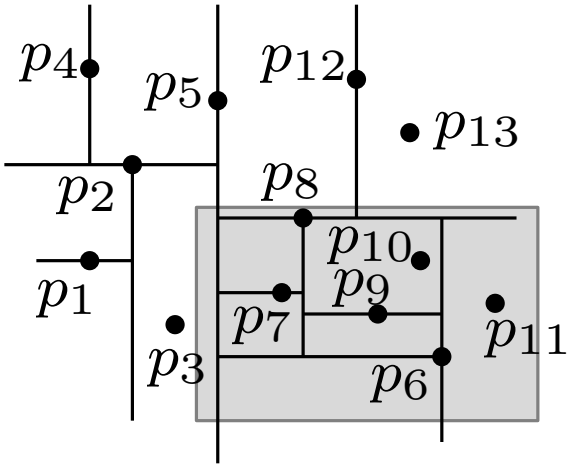
ReportSubtree in $\mathcal{O}(k)$

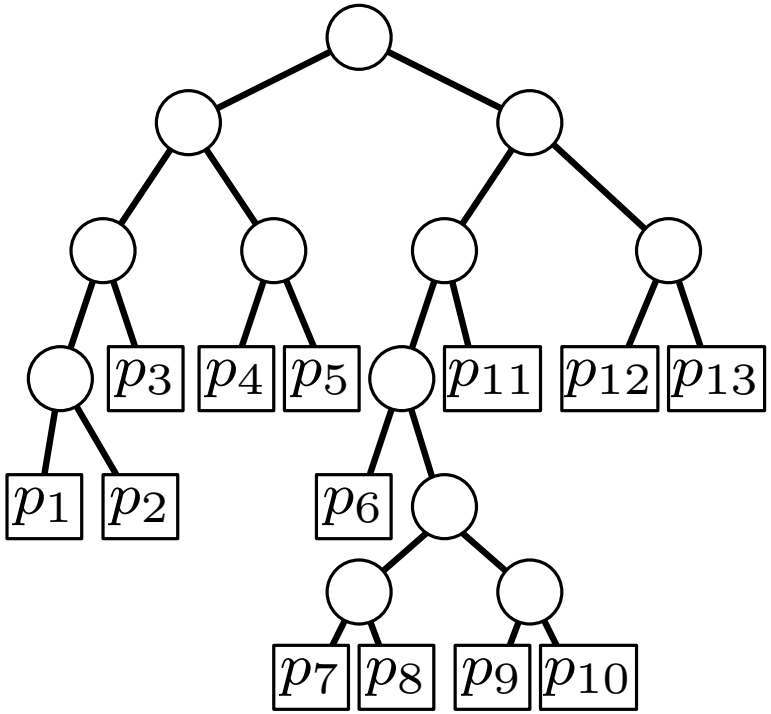


How many nodes do we consider?
Each node corresponds with one recursion step.

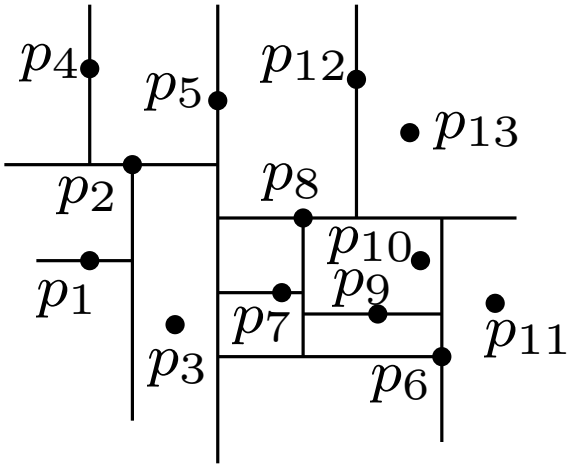
each node $\triangleq$ intersected region

Range Queries in kd-Trees



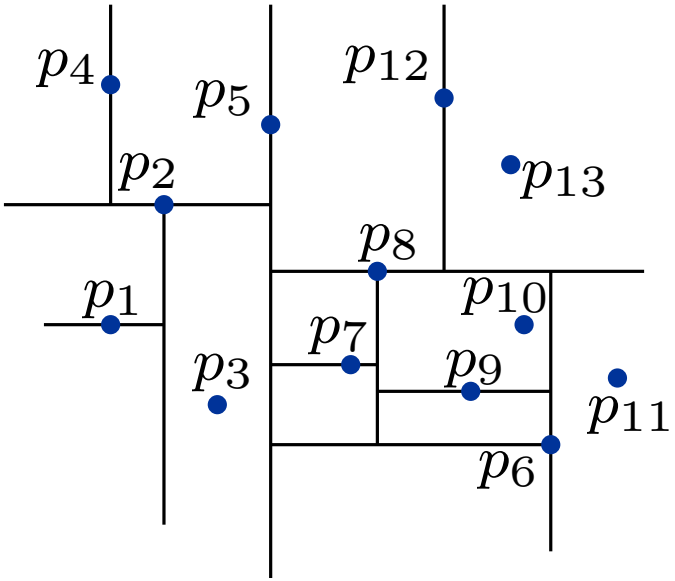


Range Queries in kd-Trees

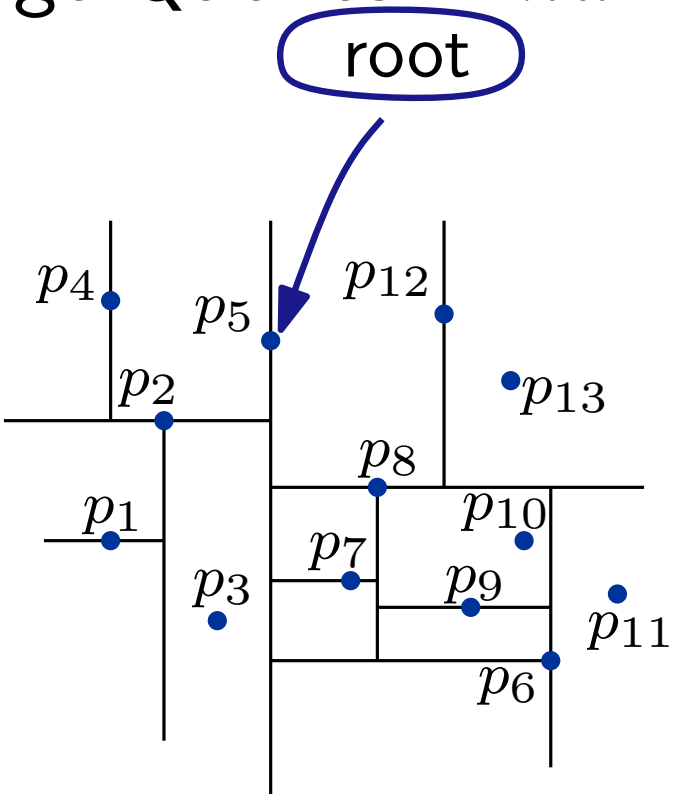


How many regions are intersected by a vertical line?

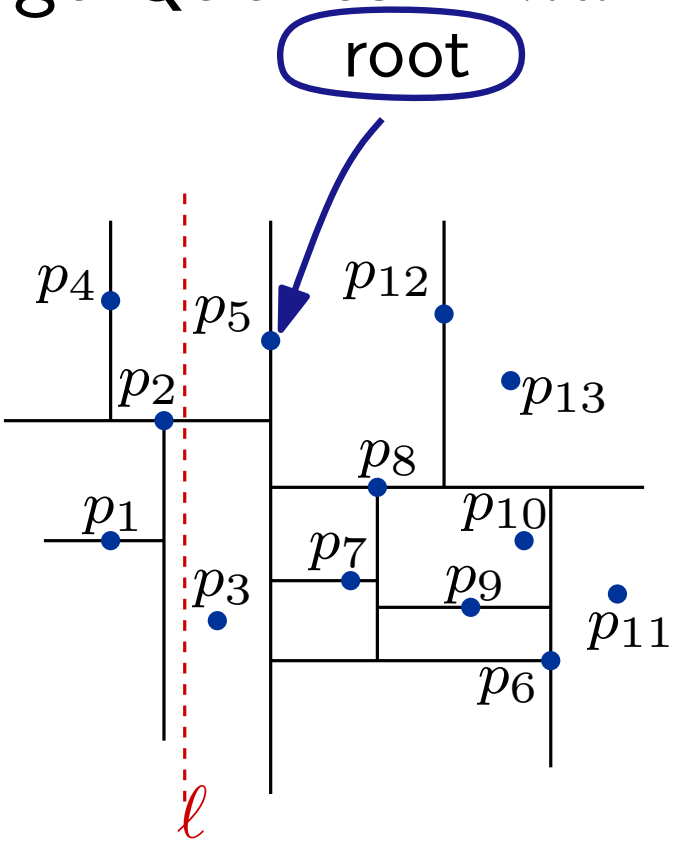
Range Queries in kd-Trees



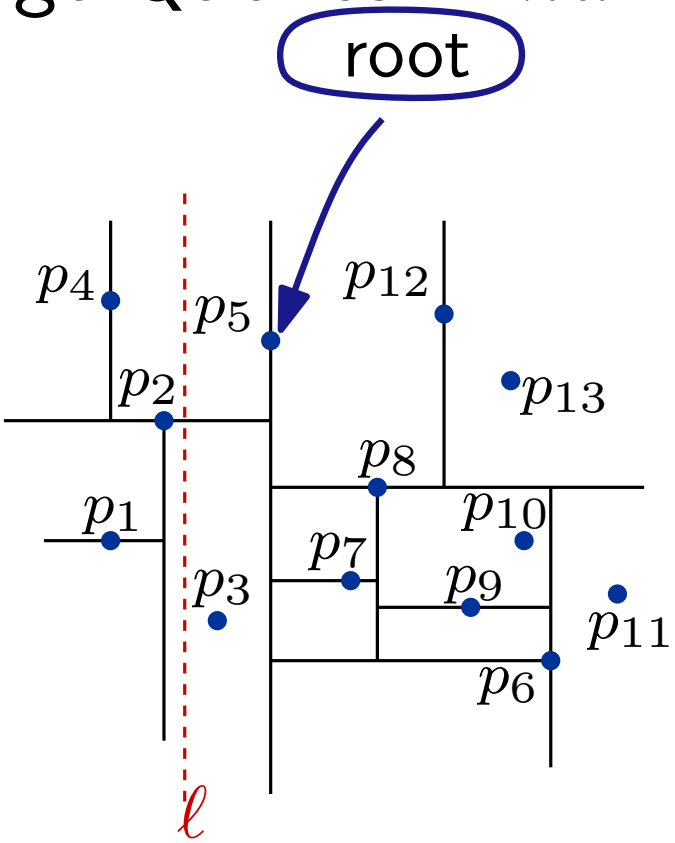
Range Queries in kd-Trees



Range Queries in kd-Trees



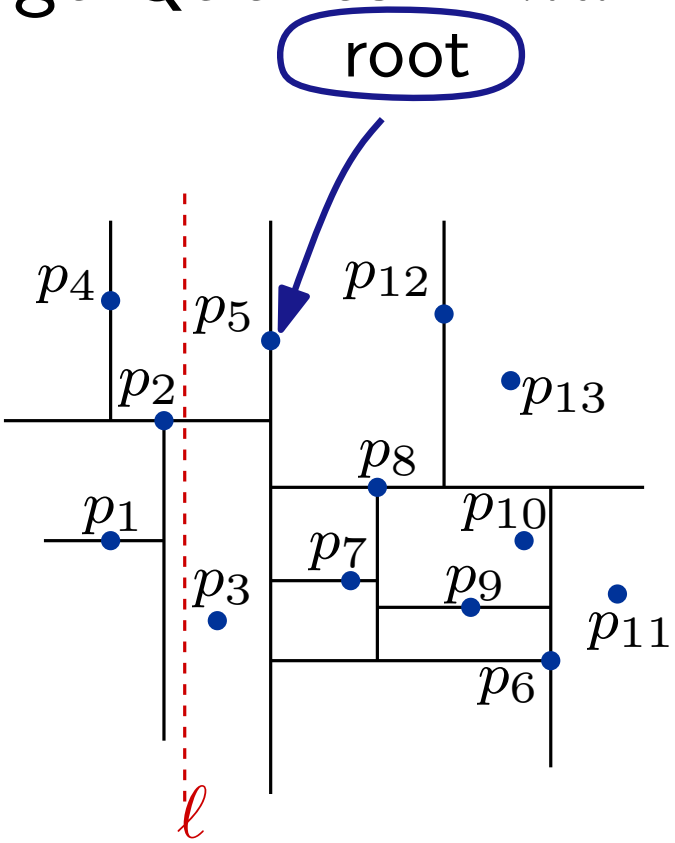
Range Queries in kd-Trees



Conjecture:

$$Q(n) = 1 + Q(n/2)$$

Range Queries in kd-Trees



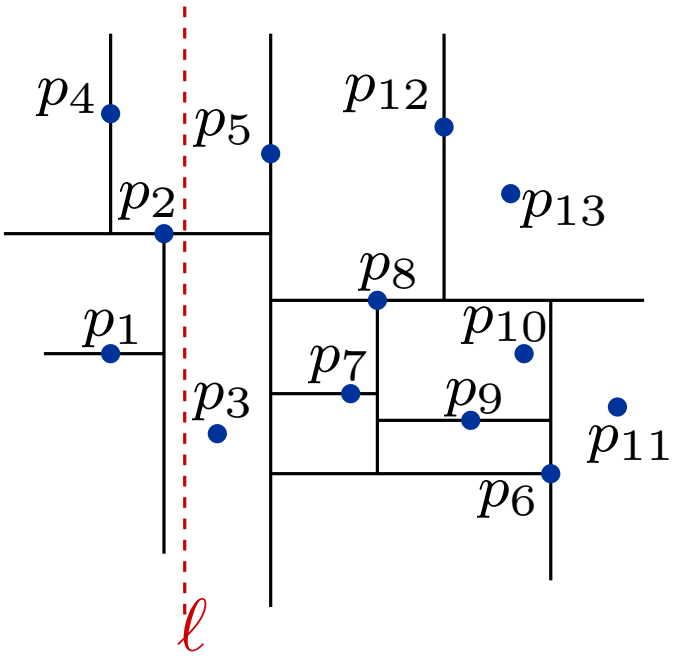
Conjecture:

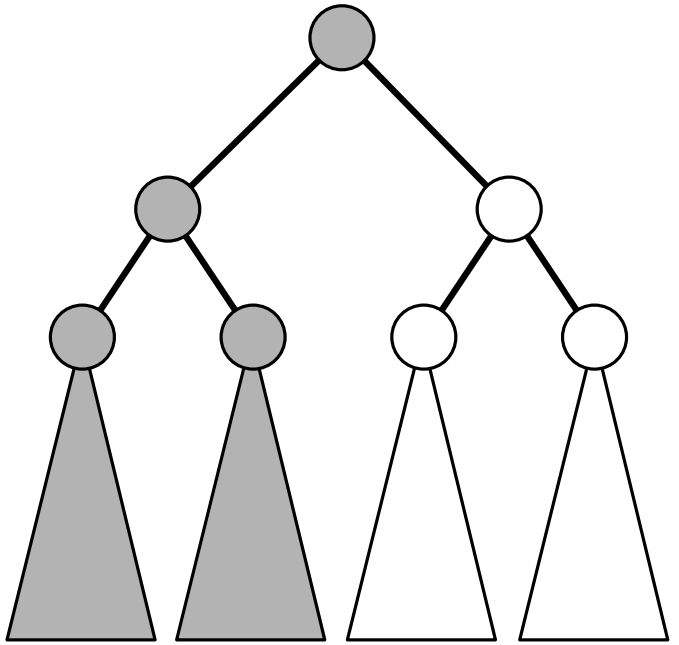
$Q(n) = 1 + Q(n/2)$

Problem?

ℓ intersects both children of left child of root

Range Queries in kd-Trees





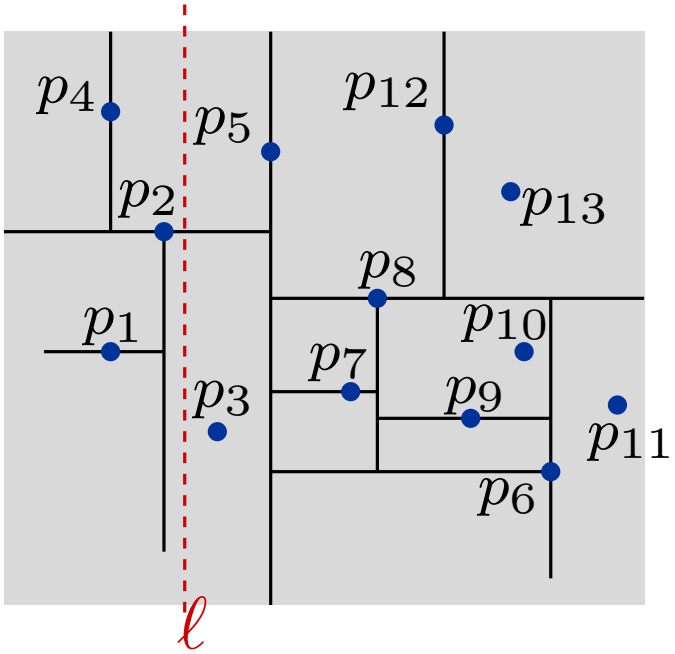
Conjecture:

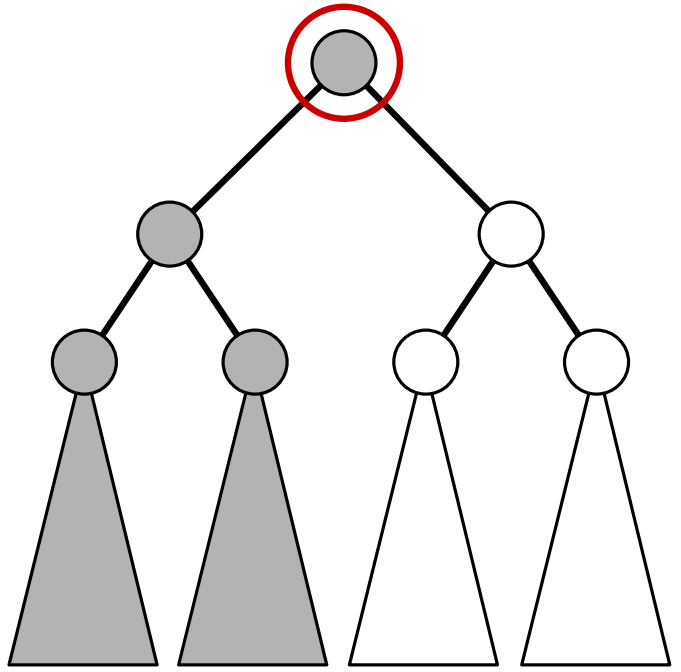
$$Q(n) = 1 + Q(n/2)$$

Problem?

ℓ intersects both children of left child of root

Range Queries in kd-Trees





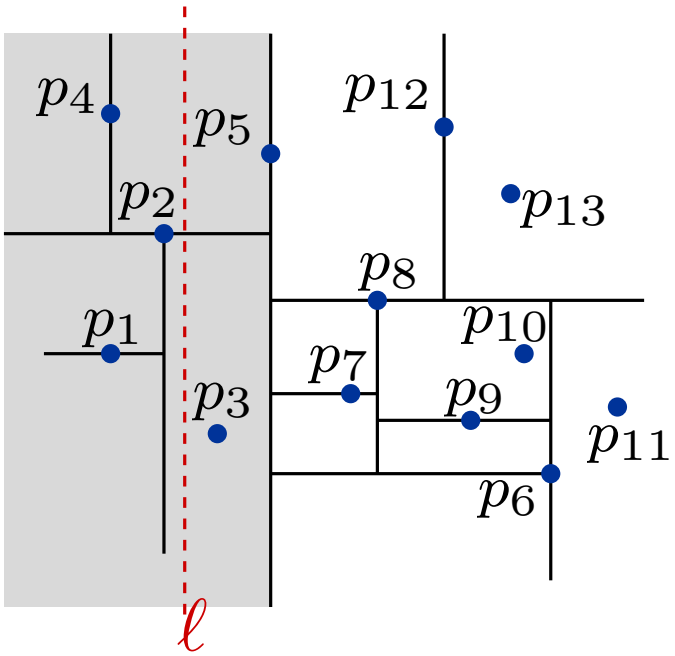
Conjecture:

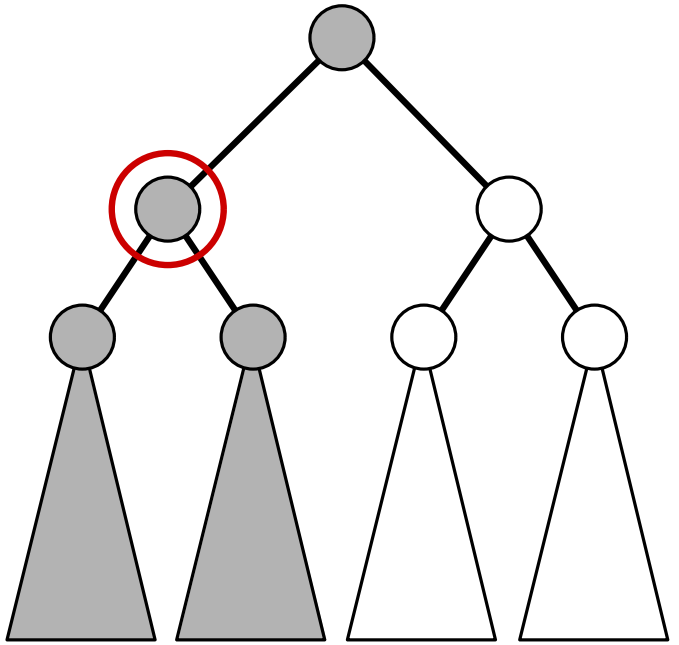
$Q(n) = 1 + Q(n/2)$

Problem?

ℓ intersects both children of left child of root

Range Queries in kd-Trees





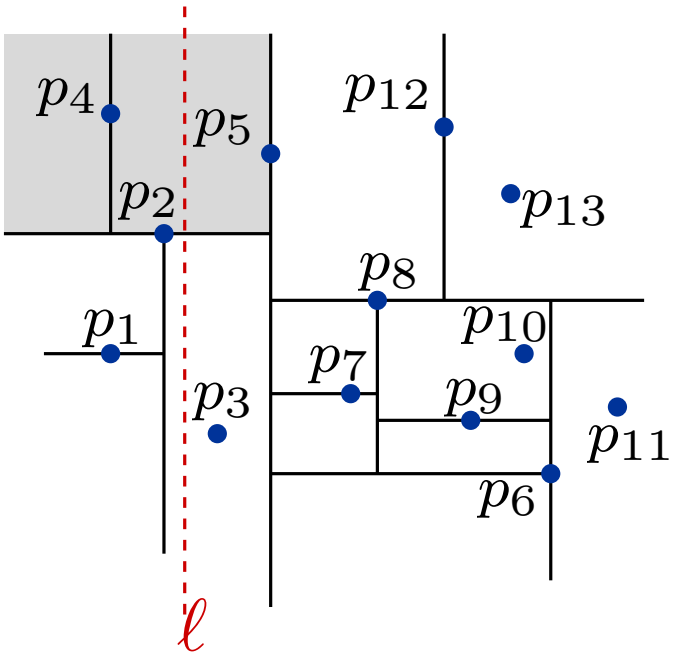
Conjecture:

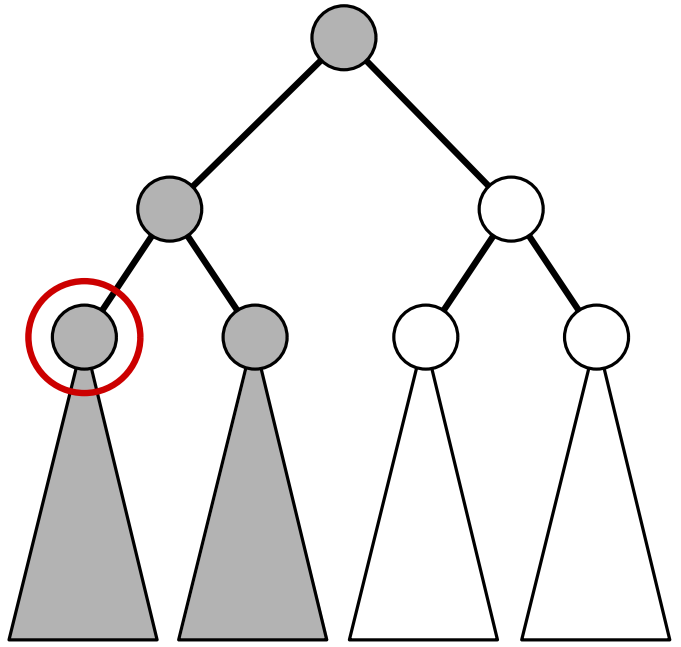
$Q(n) = 1 + Q(n/2)$

Problem?

ℓ intersects both children of left child of root

Range Queries in kd-Trees





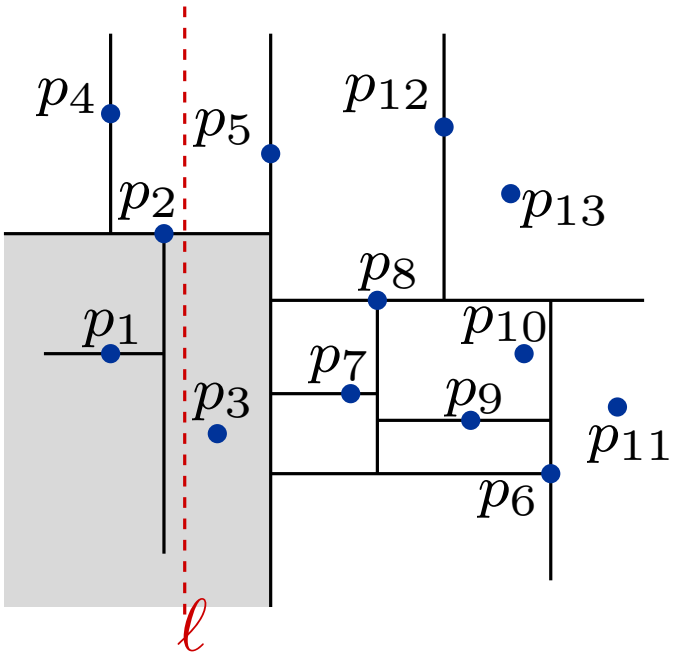
Conjecture:

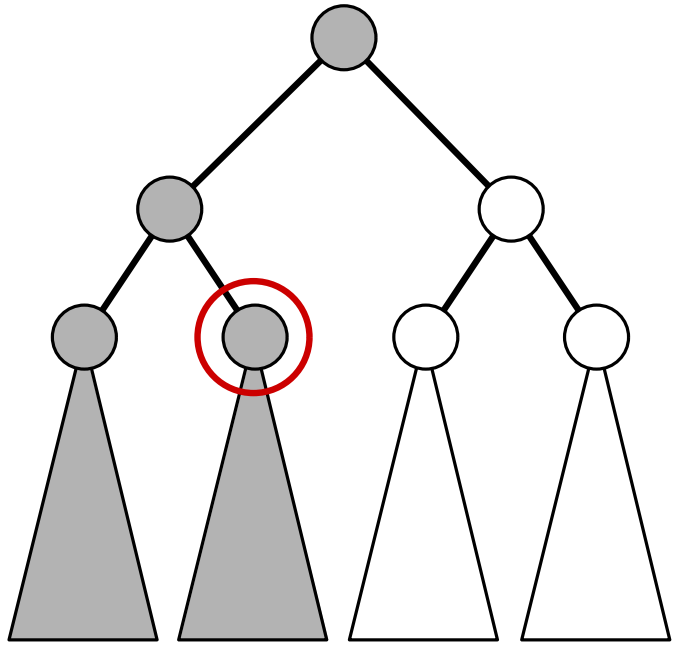
$Q(n) = 1 + Q(n/2)$

Problem?

ℓ intersects both children of left child of root

Range Queries in kd-Trees





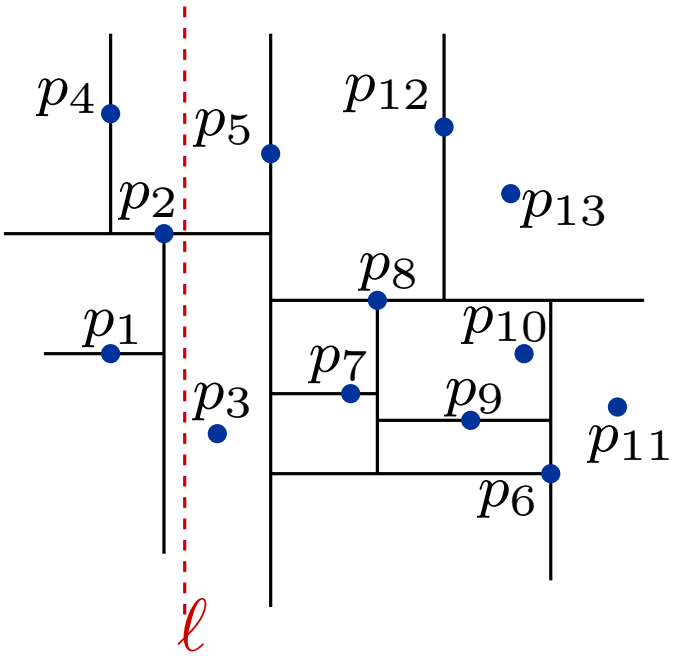
Conjecture:

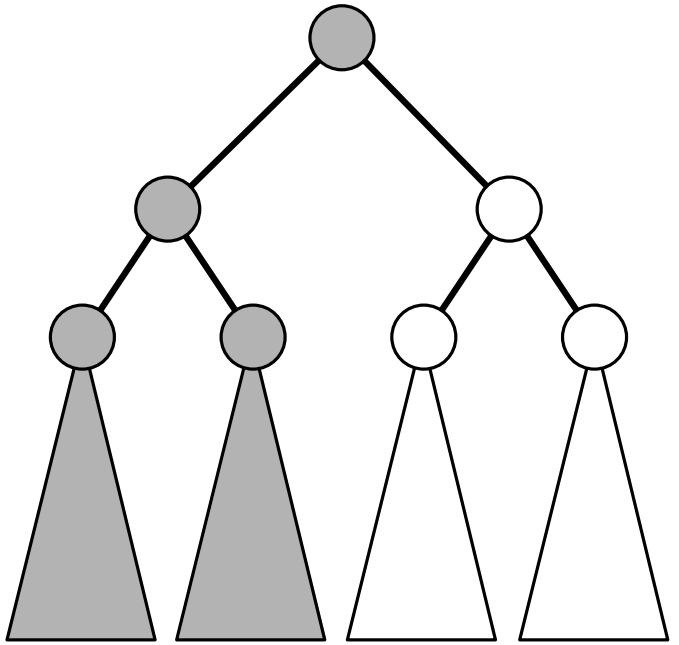
$Q(n) = 1 + Q(n/2)$

Problem?

ℓ intersects both children of left child of root

Range Queries in kd-Trees





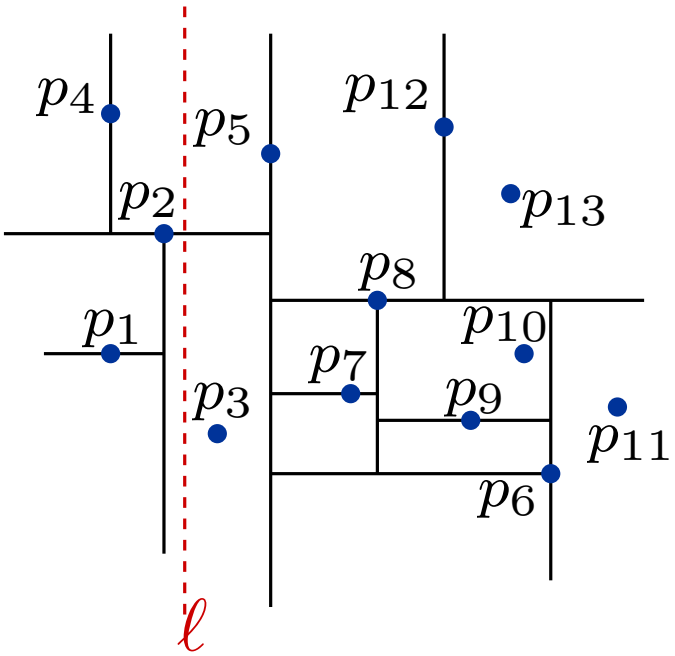
Conjecture:

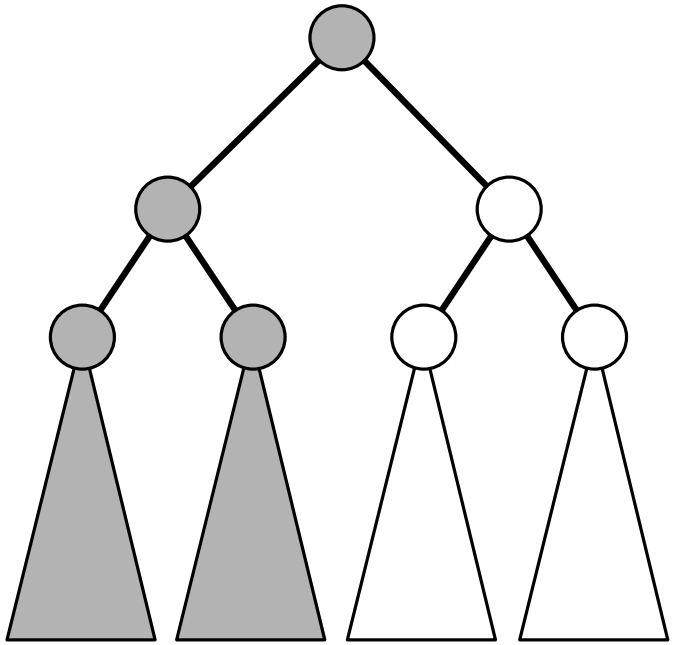
$$Q(n) = 1 + Q(n/2)$$

Problem?

ℓ intersects both children of left child of root

Range Queries in kd-Trees





Conjecture:

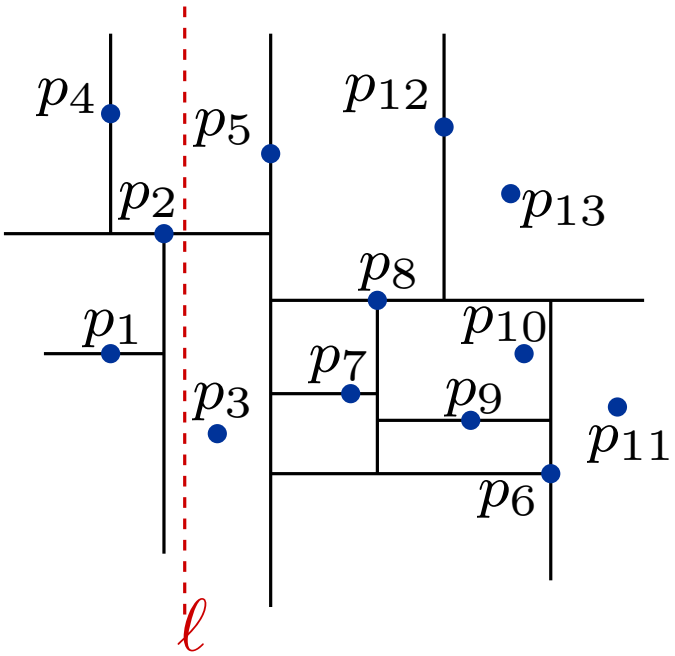
$Q(n) = 1 + Q(n/2)$

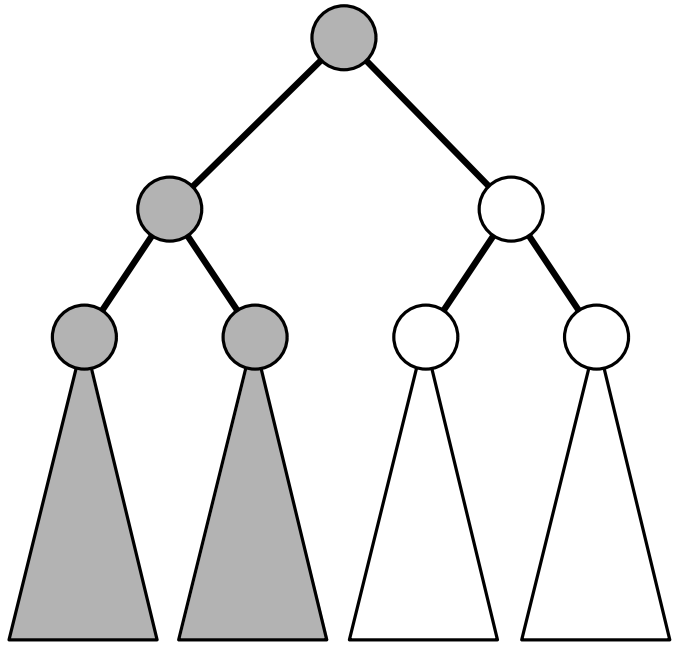
Problem?

ℓ intersects both children of left child of root

Enforce same situation!

Range Queries in kd-Trees

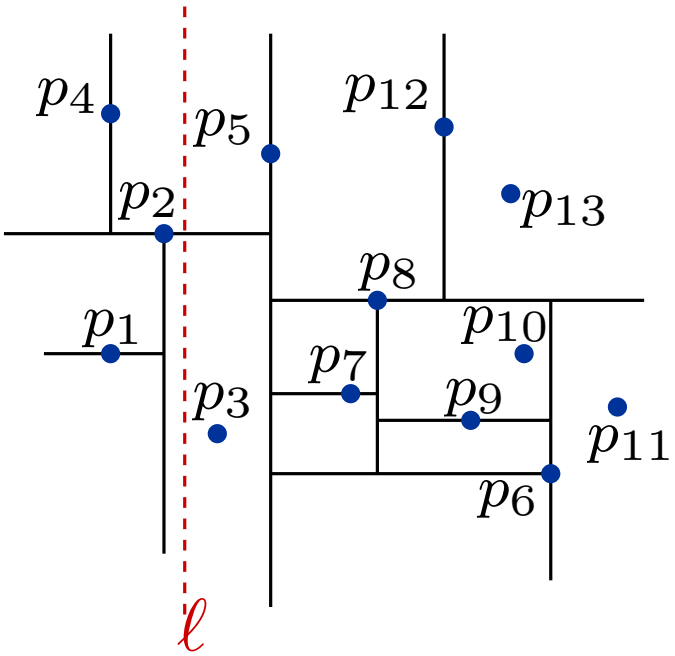


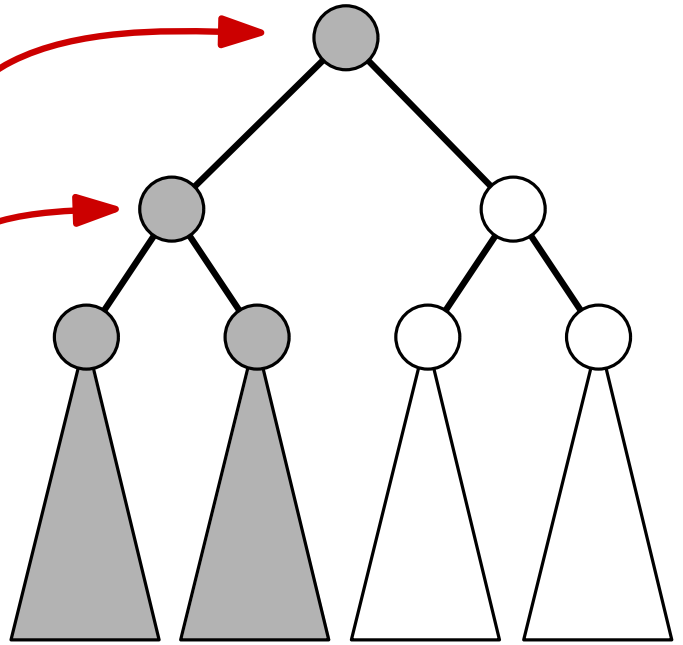


$$Q(n) = \begin{cases} \mathcal{O}(1) & , \text{ for } n = 1 \\ 2 + 2Q(n/4) & , \text{ for } n > 1 \end{cases}$$

$Q(n) =$ number of intersected regions of a kd-tree whose root contains a vertical splitting line.

Range Queries in kd-Trees

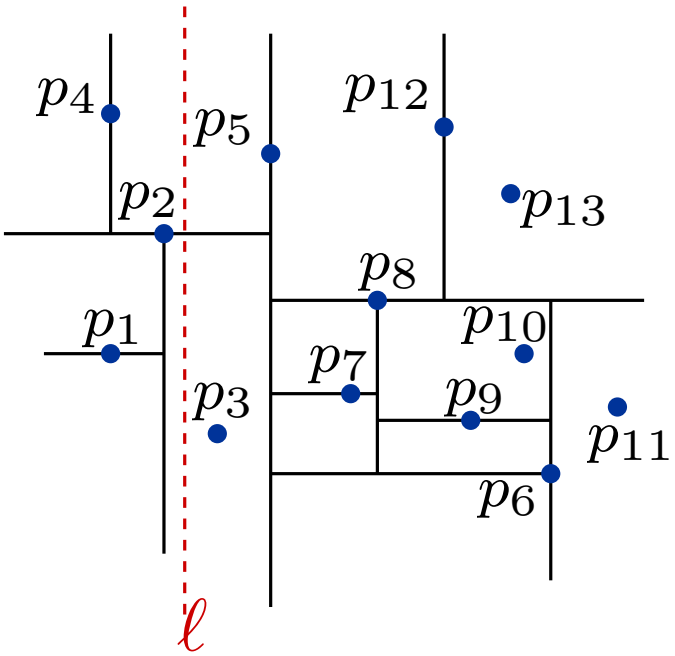


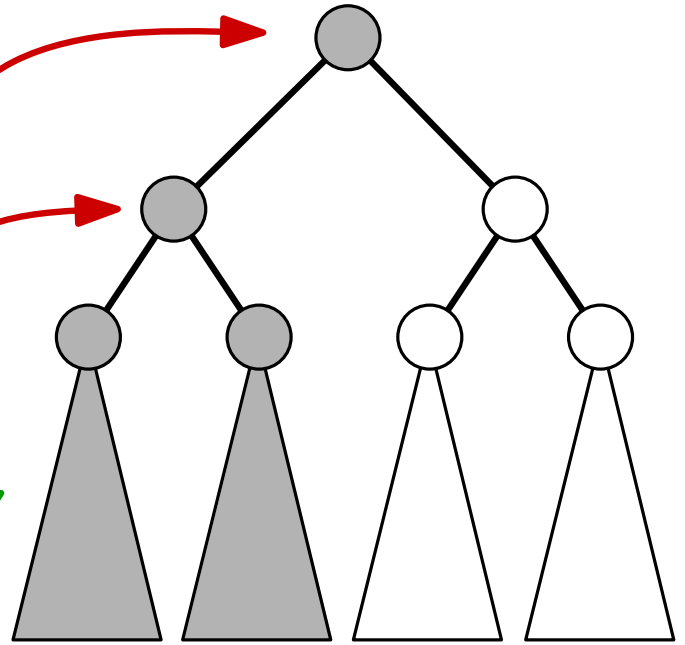


$$Q(n) = \begin{cases} \mathcal{O}(1) & , \text{ for } n = 1 \\ 2 + 2Q(n/4) & , \text{ for } n > 1 \end{cases}$$

$Q(n) =$ number of intersected regions of a kd-tree whose root contains a vertical splitting line.

Range Queries in kd-Trees





$$Q(n) = \begin{cases} \mathcal{O}(1) & , \text{ for } n = 1 \\ 2 + 2Q(n/4) & , \text{ for } n > 1 \end{cases}$$

$Q(n) =$ number of intersected regions of a kd-tree whose root contains a vertical splitting line.

Exercise 1

Claim: Queries in $O(\sqrt{n} + k)$ time

$Q(n) =$ Number of checked regions.

a) Show the following recurrence.

$$Q(n) = \begin{cases} \mathcal{O}(1) & , \text{ for } n = 1 \\ 2 + 2Q(n/4) & , \text{ for } n > 1 \end{cases}$$

Exercise 1

Claim: Queries in $O(\sqrt{n} + k)$ time

$Q(n) =$ Number of checked regions.

a) Show the following recurrence.

$$Q(n) = \begin{cases} \mathcal{O}(1) & , \text{ for } n = 1 \\ 2 + 2Q(n/4) & , \text{ for } n > 1 \end{cases}$$

b) Resolve recurrence: $Q(n) = \mathcal{O}(\sqrt{n})$.

Exercise 1

Claim: Queries in $O(\sqrt{n} + k)$ time

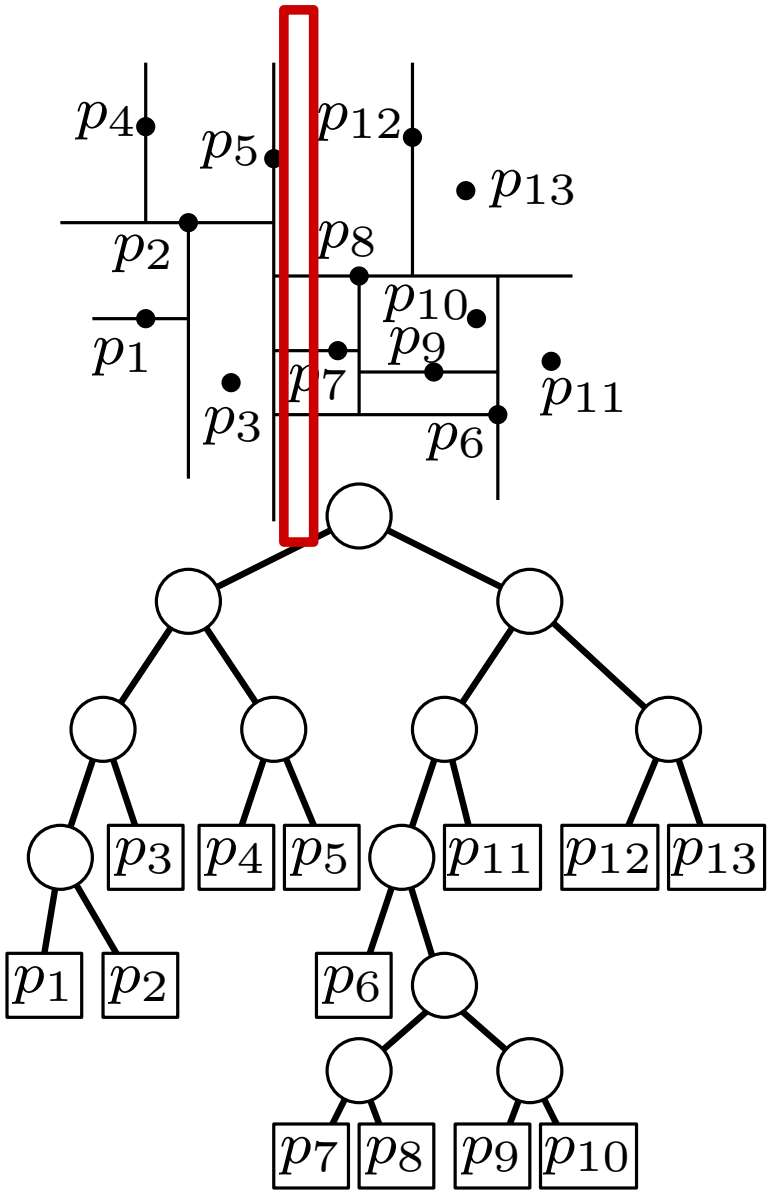
$Q(n) =$ Number of checked regions.

a) Show the following recurrence.

$$Q(n) = \begin{cases} \mathcal{O}(1) & , \text{ for } n = 1 \\ 2 + 2Q(n/4) & , \text{ for } n > 1 \end{cases}$$

b) Resolve recurrence: $Q(n) = \mathcal{O}(\sqrt{n})$.

c) $\Omega(\sqrt{n})$ lower bound for range queries in kd-trees



SearchKdTree(v, R)

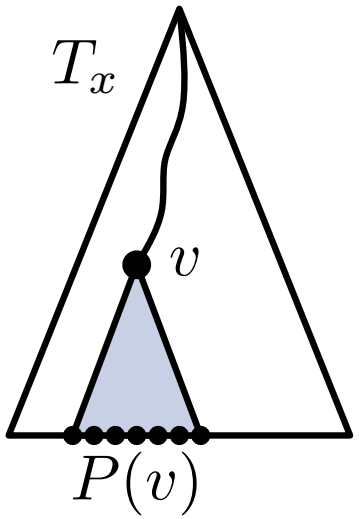
- **if** v leaf **then**
  - report point p in v when $p \in R$
- **else**
  - **if** region(lc(v)) $\subseteq R$ **then**
    - ReportSubtree(lc(v))
  - **else**
    - **if** region(lc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(lc(v), R)
  - **if** region(rc(v)) $\subseteq R$ **then**
    - ReportSubtree(rc(v))
  - **else**
    - **if** region(rc(v))$\cap R \neq \emptyset$ **then**
      - SearchKdTree(rc(v), R)

c) $\Omega(\sqrt{n})$ lower bound for range queries kd-trees

Range Trees

Idea: Use 1-dimensional search trees on two levels:

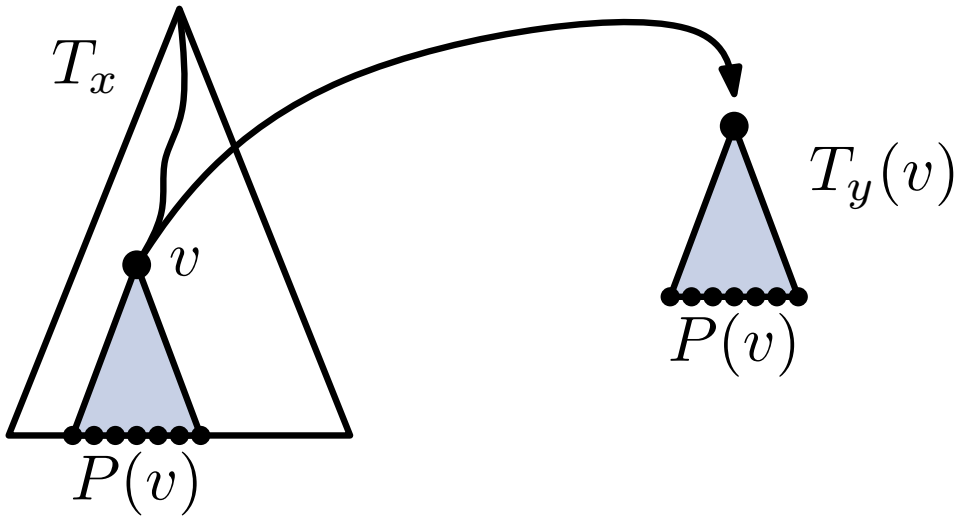
- a 1d search tree T_x on x-coordinates



Range Trees

Idea: Use 1-dimensional search trees on two levels:

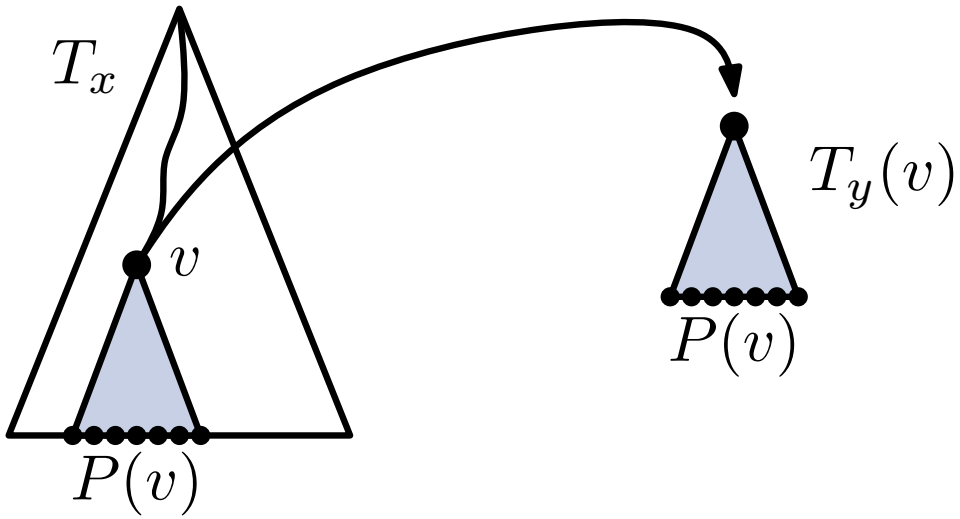
- a 1d search tree T_x on x-coordinates
- in each node v of T_x a 1d search tree $T_y(v)$ stores the canonical subset $P(v)$ on y-coordinates



Range Trees

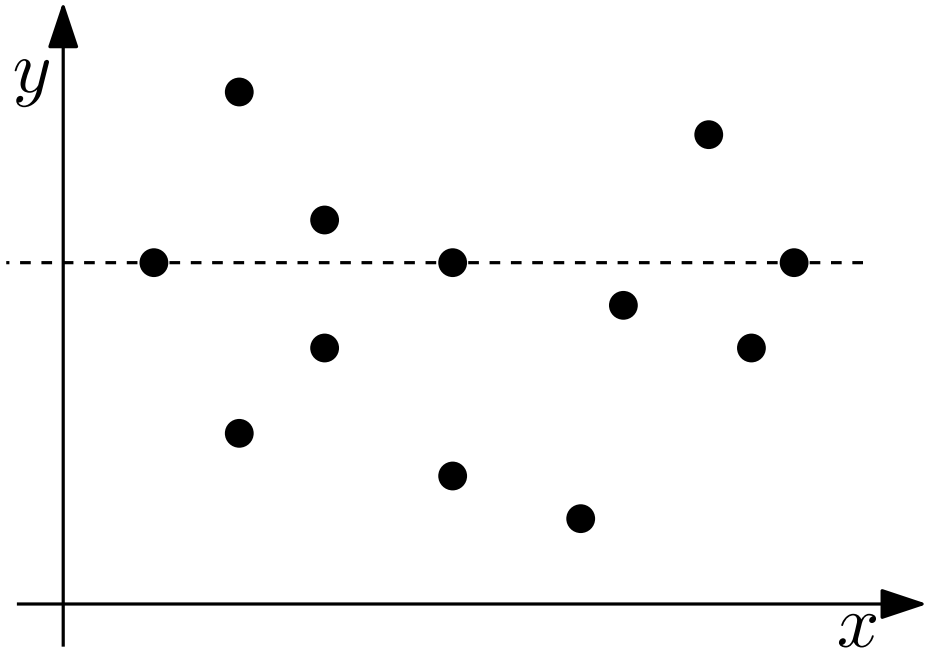
Idea: Use 1-dimensional search trees on two levels:

- a 1d search tree T_x on x-coordinates
- in each node v of T_x a 1d search tree $T_y(v)$ stores the canonical subset $P(v)$ on y-coordinates
- Compute the points by x-query in T_x and subsequent y-queries in the auxiliary structures T_y for the subtrees in T_x



Exercise 2

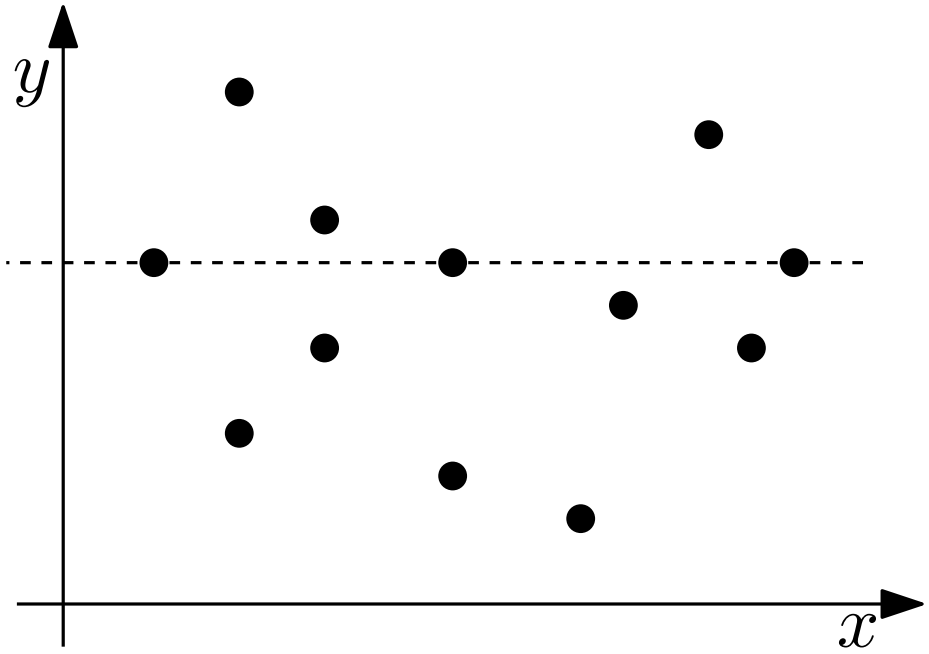
'partial match queries'



Report all points with $y = 7$.

Exercise 2

'partial match queries'

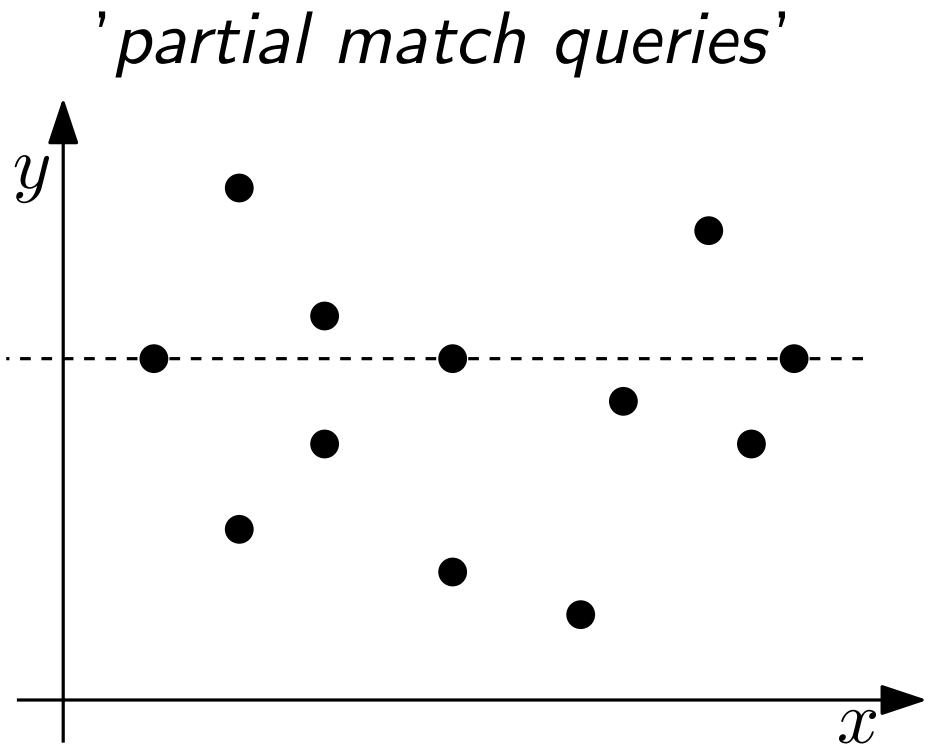


Report all points with $y = 7$.

---

a) How to apply kd-trees?

Exercise 2



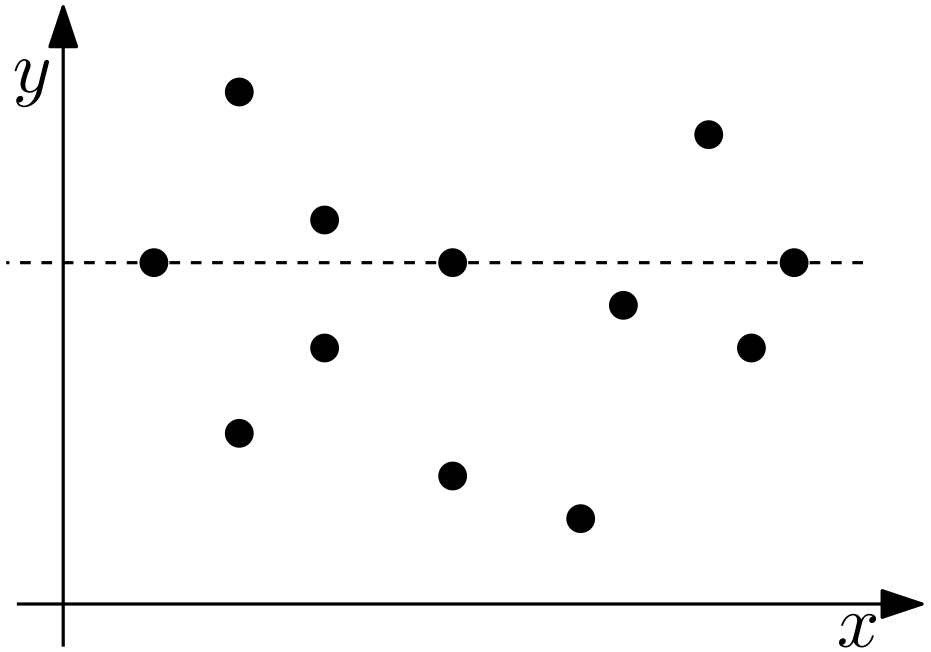
Report all points with $y = 7$.

---

b) How to apply range-trees?

Exercise 2

'partial match queries'



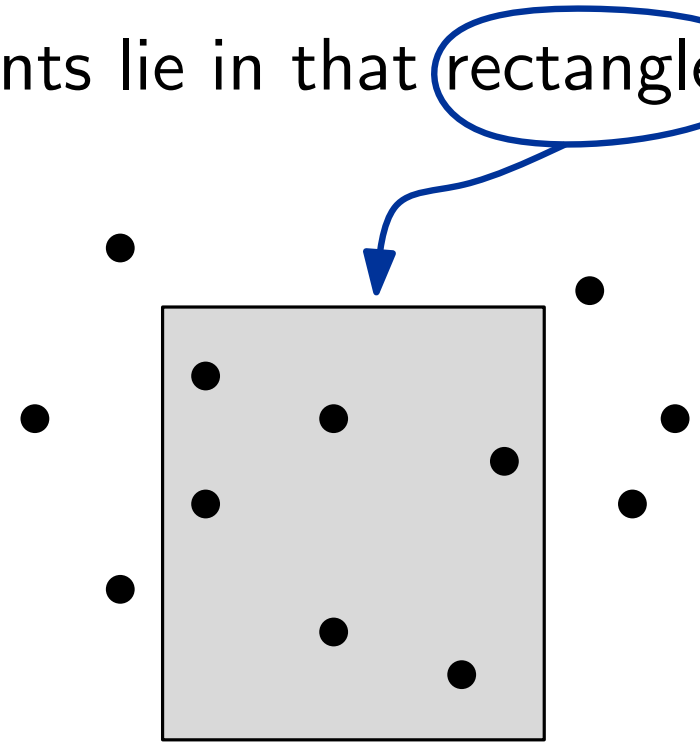
Report all points with $y = 7$.

---

c) Find: Datatstructure that solves problem in $\mathcal{O}(\log n + k)$ time and $\mathcal{O}(n)$ storage.

Exercise 3

How many points lie in that rectangle?

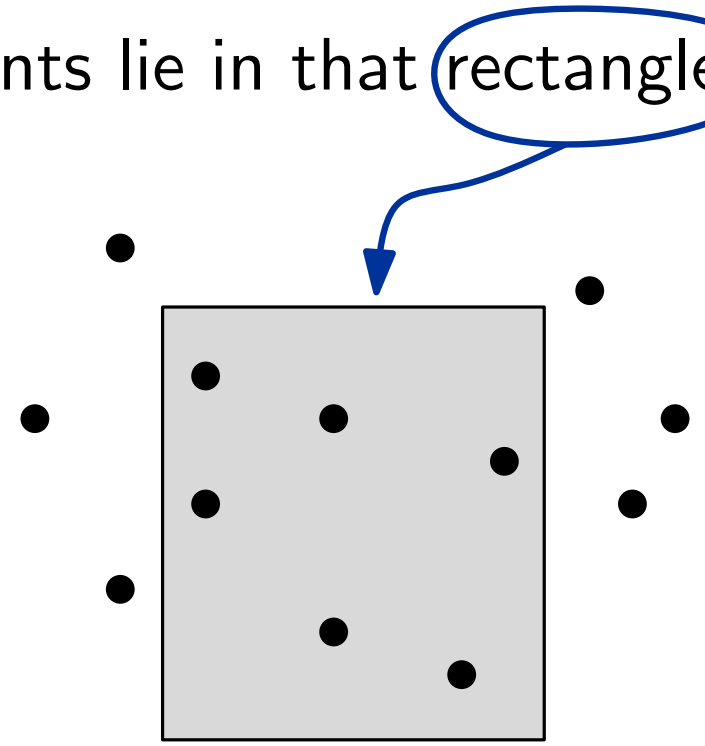


range counting query

Requirement: Without additive constant $\mathcal{O}(k)$ in running time.

Exercise 3

How many points lie in that rectangle?



range counting query

Requirement: Without additive constant $\mathcal{O}(k)$ in running time.

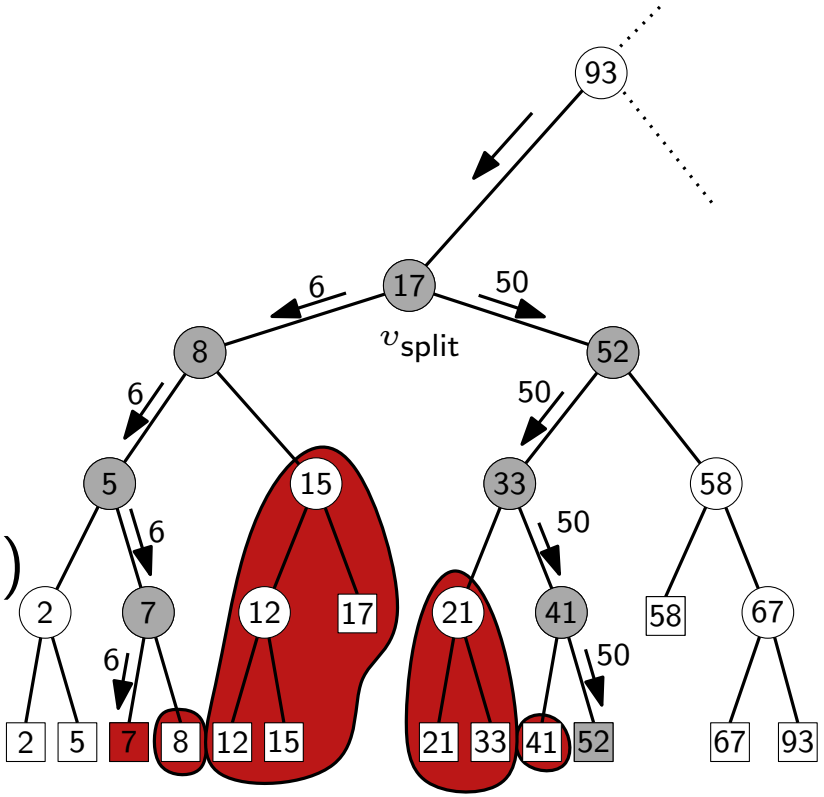
a) Adapt 1-dim range-tree for range counting queries in $\mathcal{O}(\log n)$.

Exercise 3

1dRangeQuery(T, x, x')

$v_{\text{split}} \leftarrow$ FindSplitNode(T, x, x')
if v_{split} ist Blatt **then** prüfe v_{split}

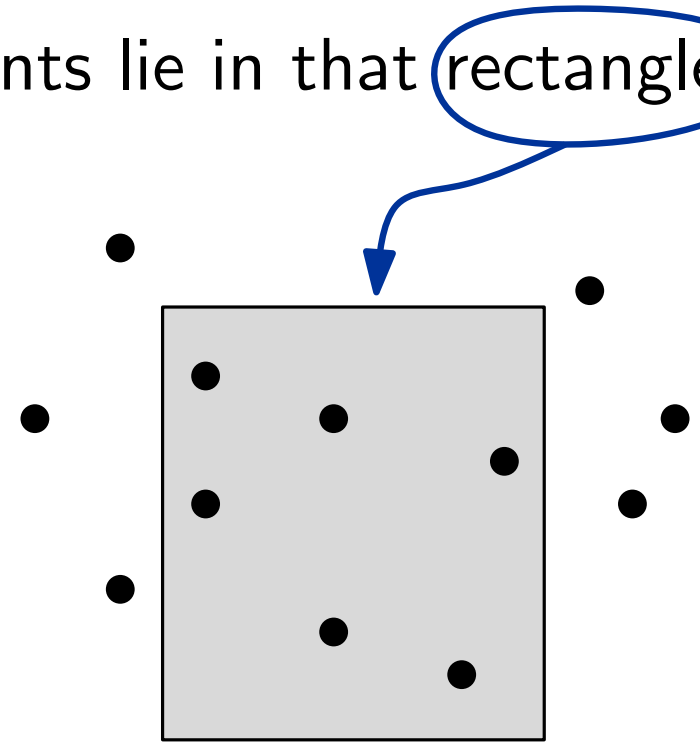
else
  $v \leftarrow \text{lc}(v_{\text{split}})$
  while v kein Blatt **do**
    if $x \leq x_v$ **then**
      ReportSubtree(rc(v)); $v \leftarrow \text{lc}(v)$
    else $v \leftarrow \text{rc}(v)$
  prüfe v
  // analog für x' und rc(v_{split})



a) Adapt 1-dim range-tree for range counting queries in $\mathcal{O}(\log n)$.

Exercise 3

How many points lie in that rectangle?



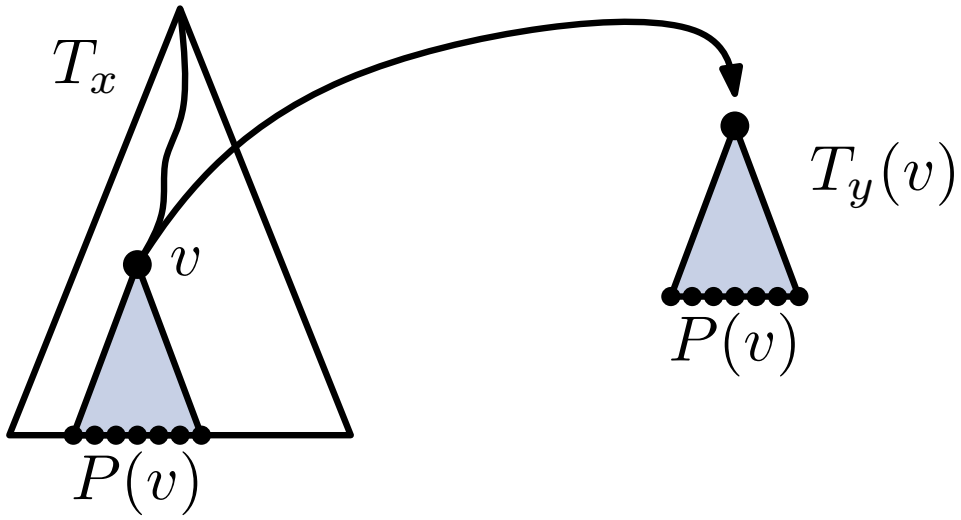
range counting query

Requirement: Without additive constant $\mathcal{O}(k)$ in running time.

b) How to solve the d-dimensional problem.

Exercise 3

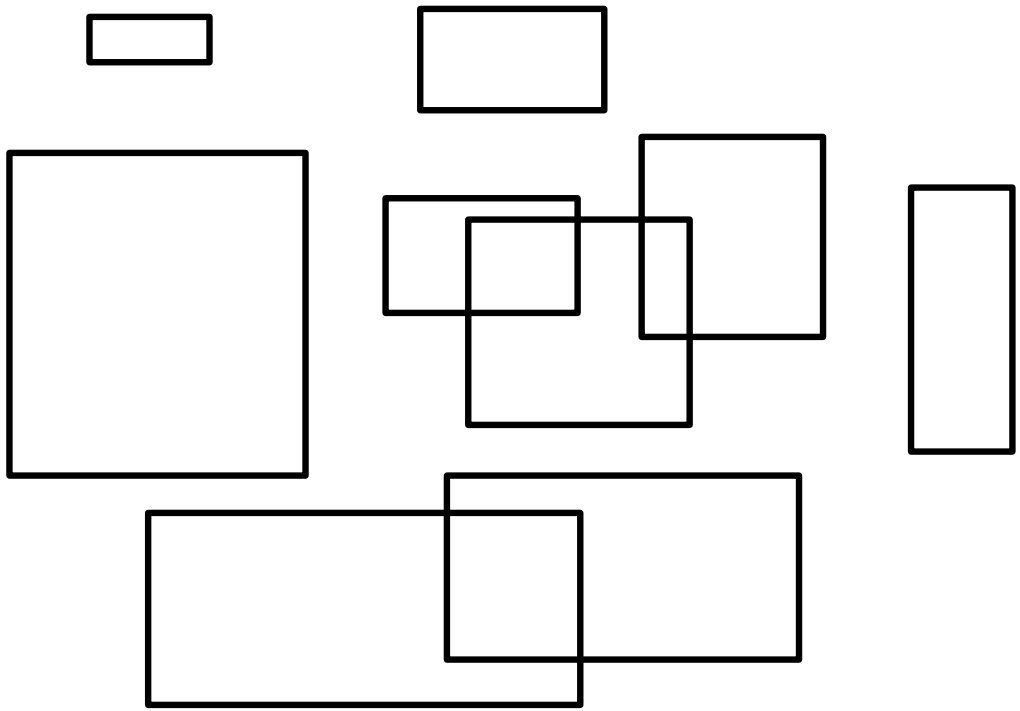
2dim Range-Trees:



Requirement: Without additive constant $\mathcal{O}(k)$ in running time.

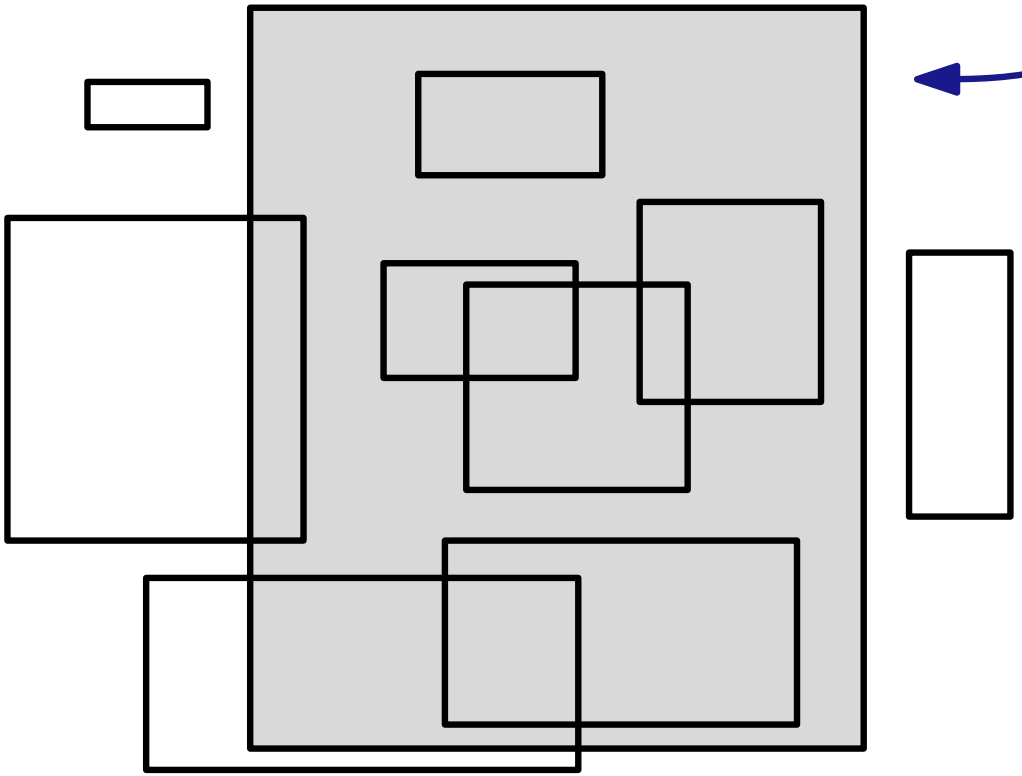
b) How to solve the d-dimensional problem.

Exercise 4



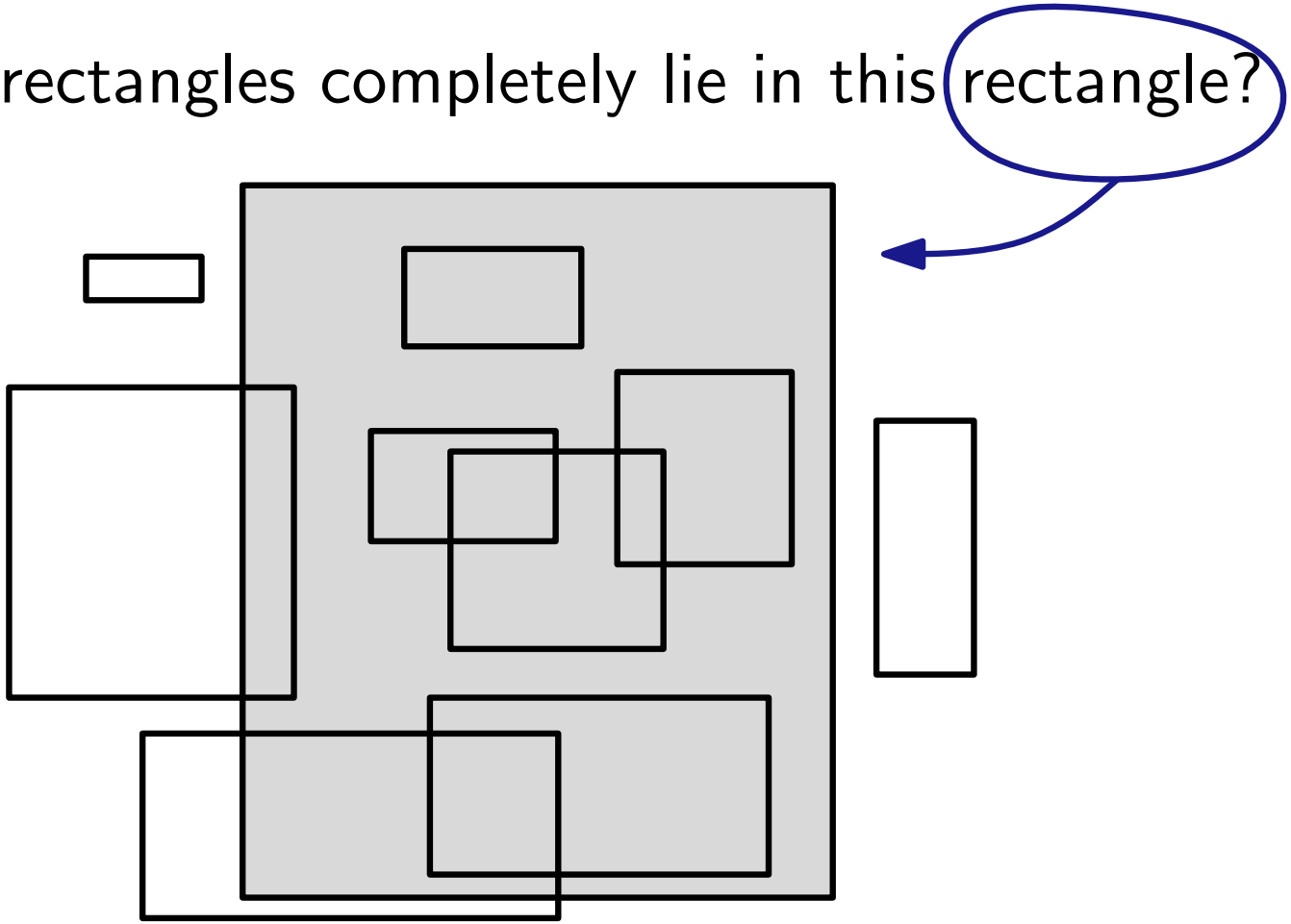
Exercise 4

Which rectangles completely lie in this rectangle?



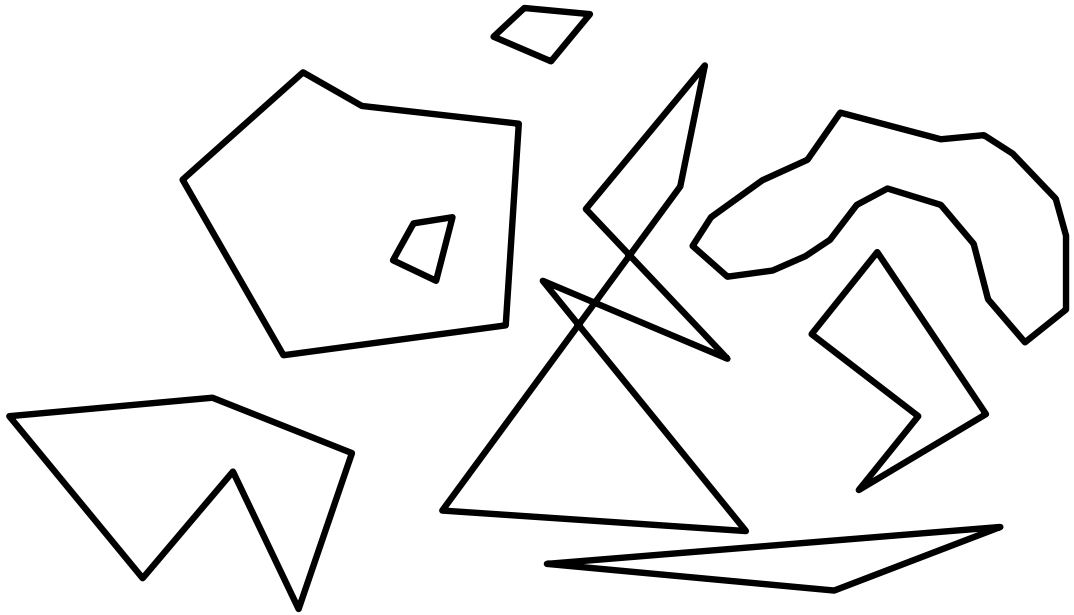
Exercise 4

Which rectangles completely lie in this rectangle?



Data structure with $\mathcal{O}(n \log^3 n)$ storage and $\mathcal{O}(\log^4 n + k)$ query time.

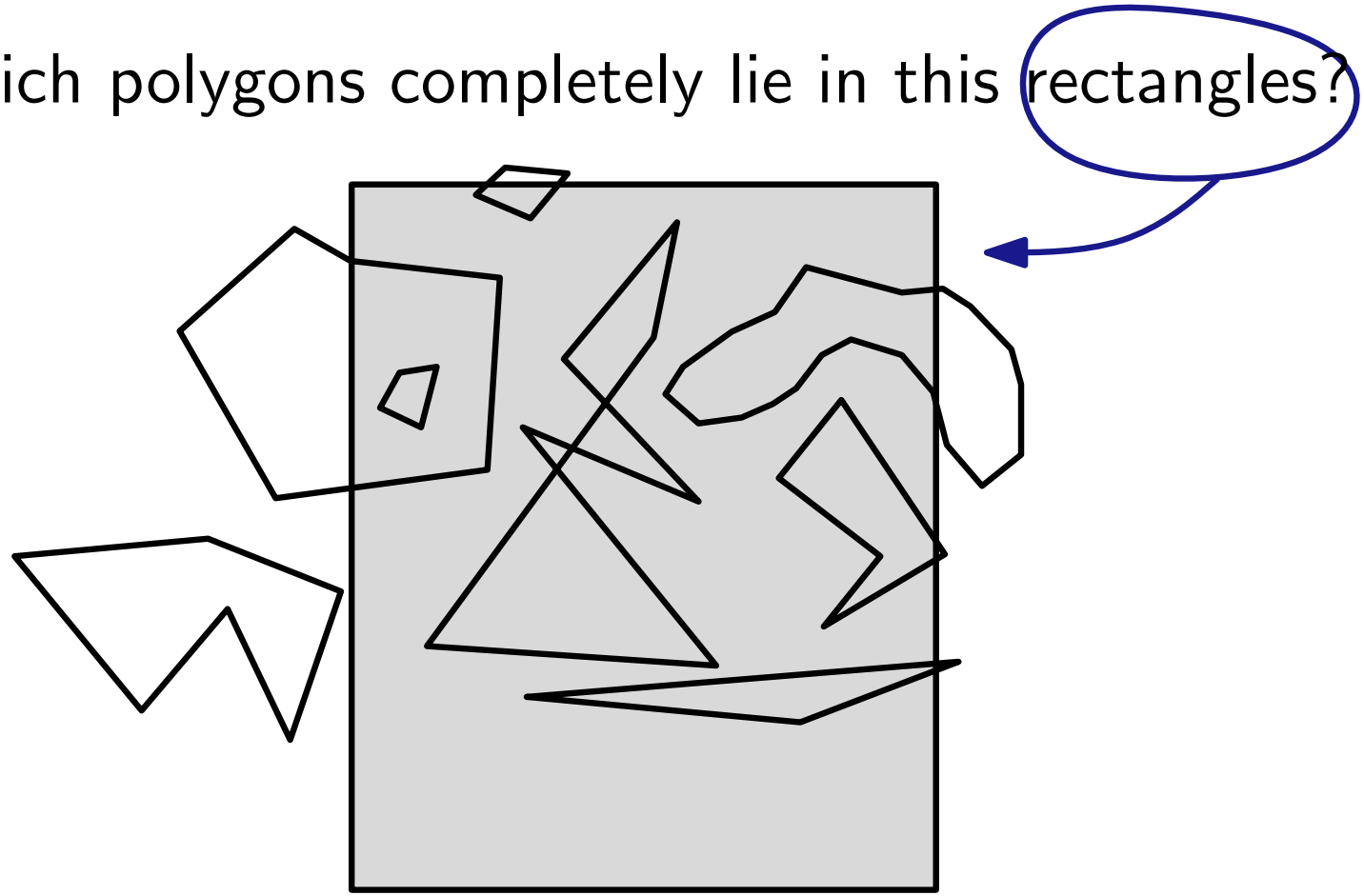
Exercise 4



Data structure with $\mathcal{O}(n \log^3 n)$ storage and $\mathcal{O}(\log^4 n + k)$ query time.

Exercise 4

Which polygons completely lie in this rectangles?



Data structure with $\mathcal{O}(n \log^3 n)$ storage and $\mathcal{O}(\log^4 n + k)$ query time.