

Übung Algorithmische Geometrie

Bereichsabfragen

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

Benjamin Niedermann
21.05.2014



Übungsblatt 5

Orthogonale Bereichsabfragen für $d = 2$

Geg: Menge P von n Punkten in \mathbb{R}^2

Ziel: Datenstruktur zur effizienten Beantwortung von Bereichsabfragen der Form $R = [x, x'] \times [y, y']$

Orthogonale Bereichsabfragen für $d = 2$

Geg: Menge P von n Punkten in \mathbb{R}^2

Ziel: Datenstruktur zur effizienten Beantwortung von Bereichsabfragen der Form $R = [x, x'] \times [y, y']$

Lösungsansätze:

- ein Suchbaum, der abwechselnd nach x - und y -Koordinaten trennt
→ **kd -Tree**
- *primärer* Suchbaum für x -Koordinaten, mehrere *sekundäre* Suchbäume für y -Koordinaten
→ **Range-Tree**

Orthogonale Bereichsabfragen für $d = 2$

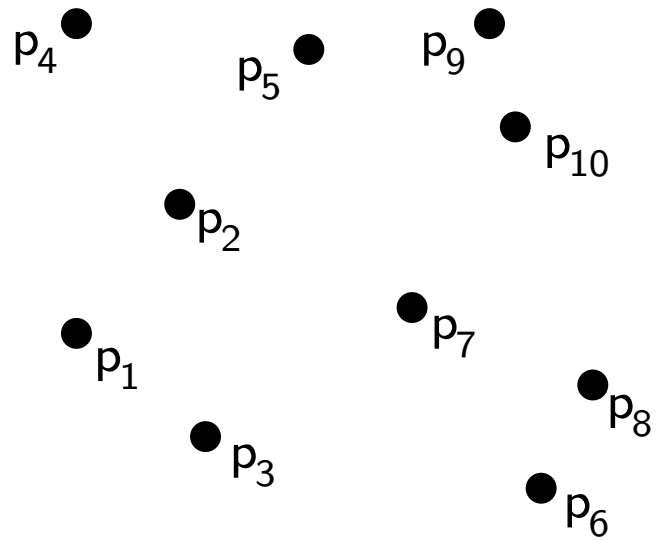
Geg: Menge P von n Punkten in \mathbb{R}^2

Ziel: Datenstruktur zur effizienten Beantwortung von Bereichsabfragen der Form $R = [x, x'] \times [y, y']$

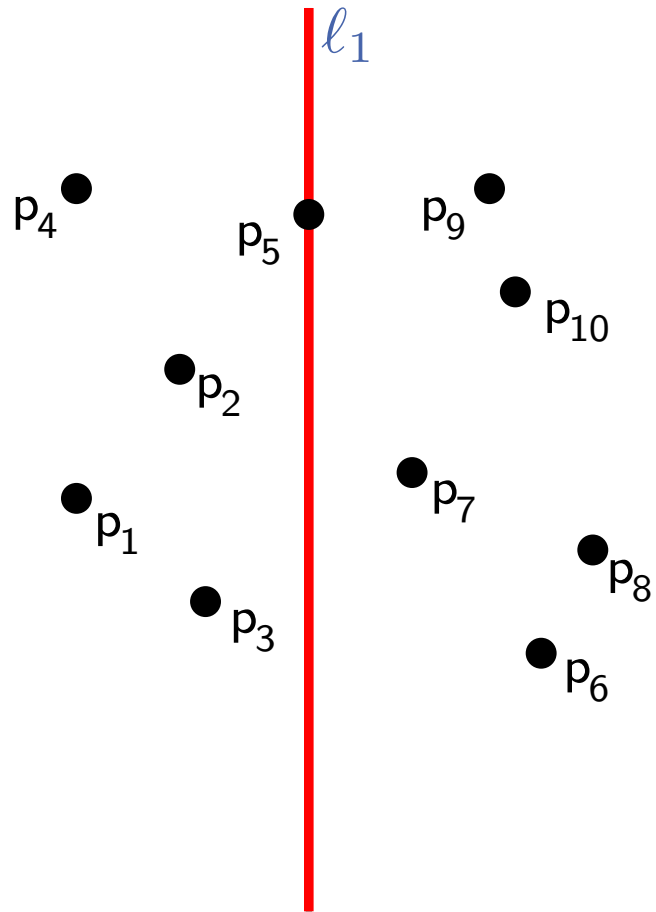
Lösungsansätze:

- ein Suchbaum, der abwechselnd nach x - und y -Koordinaten trennt \rightarrow ***kd-Tree***
- *primärer* Suchbaum für x -Koordinaten, mehrere *sekundäre* Suchbäume für y -Koordinaten \rightarrow **Range-Tree**

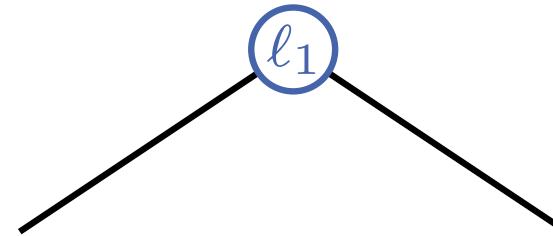
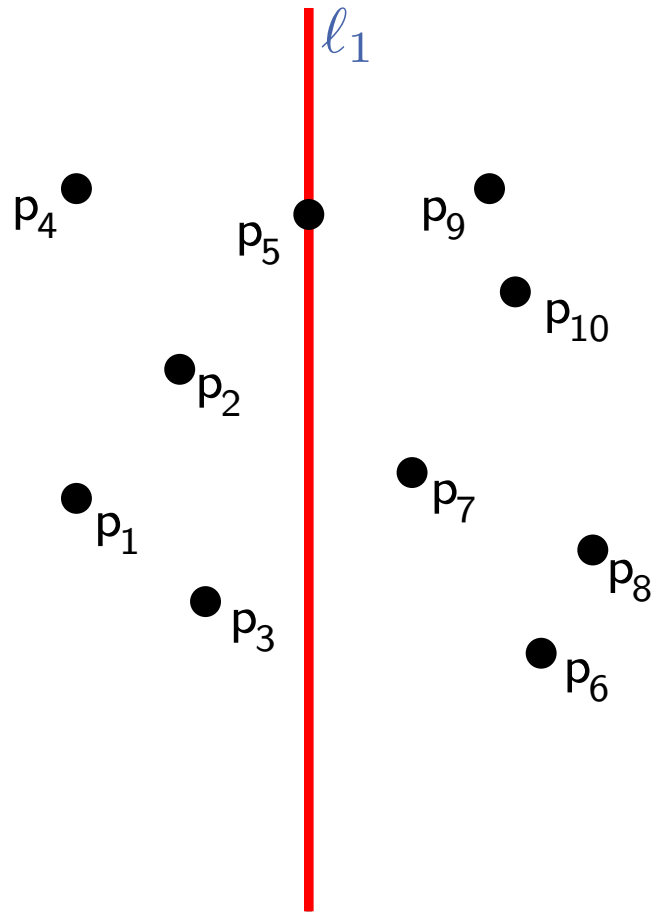
kd-Trees: Beispiel



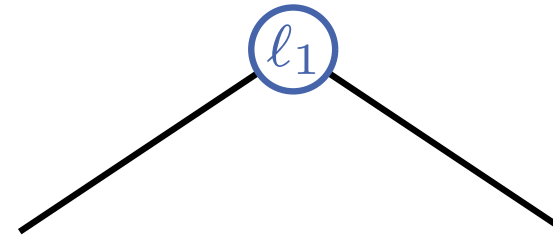
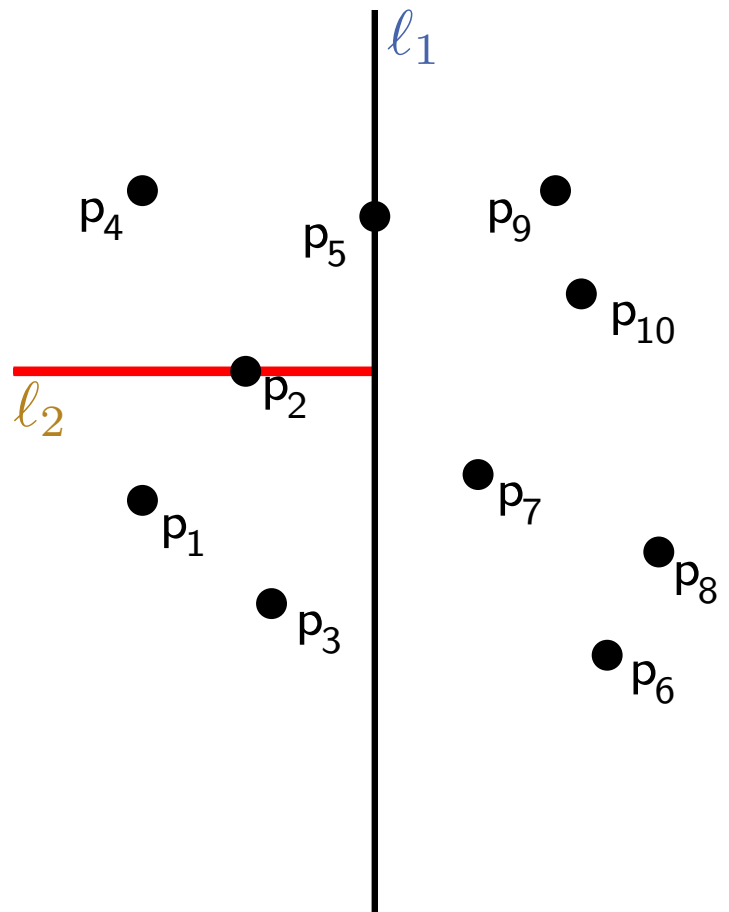
kd-Trees: Beispiel



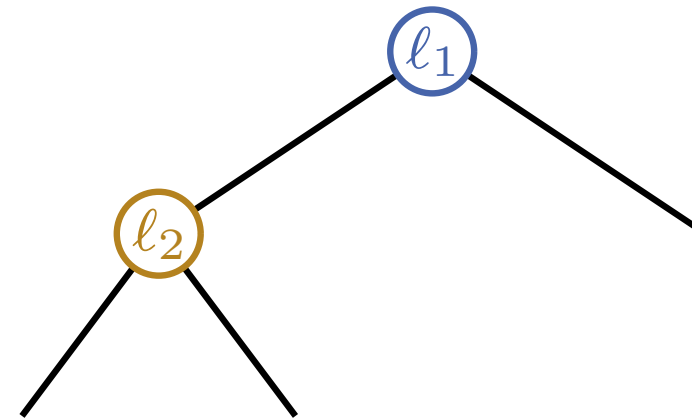
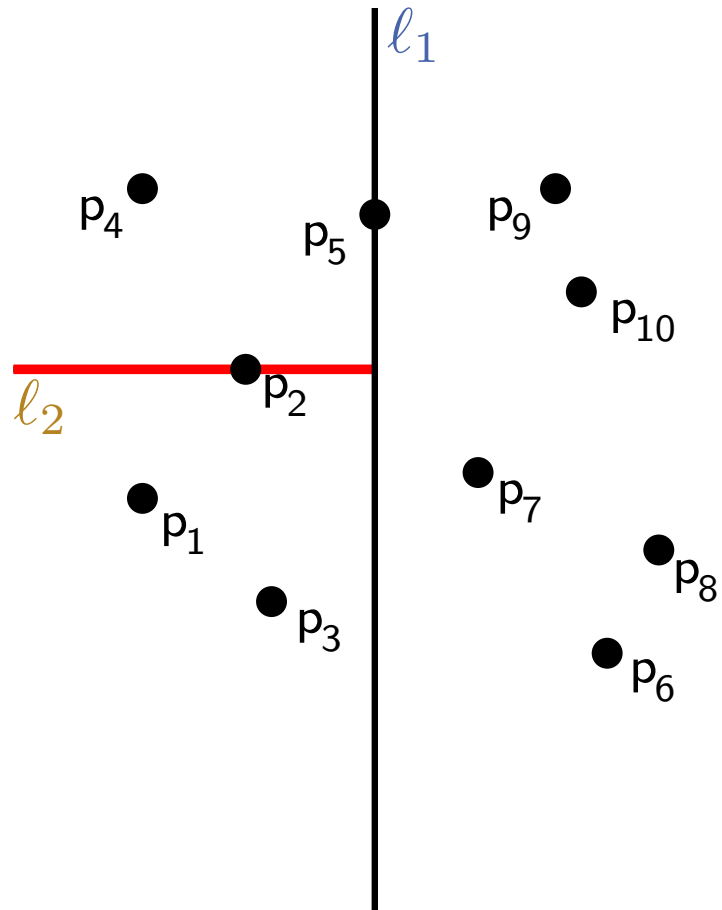
kd-Trees: Beispiel



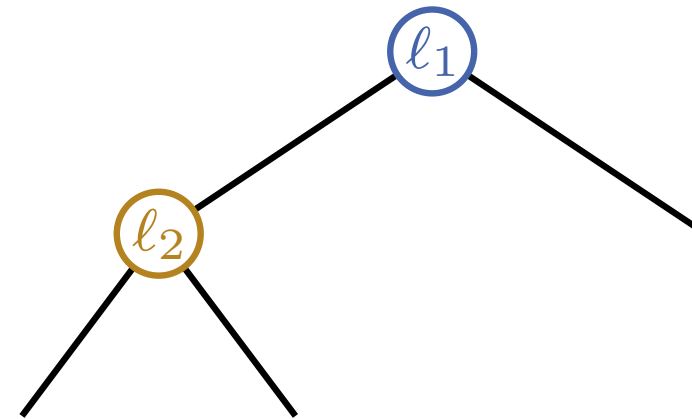
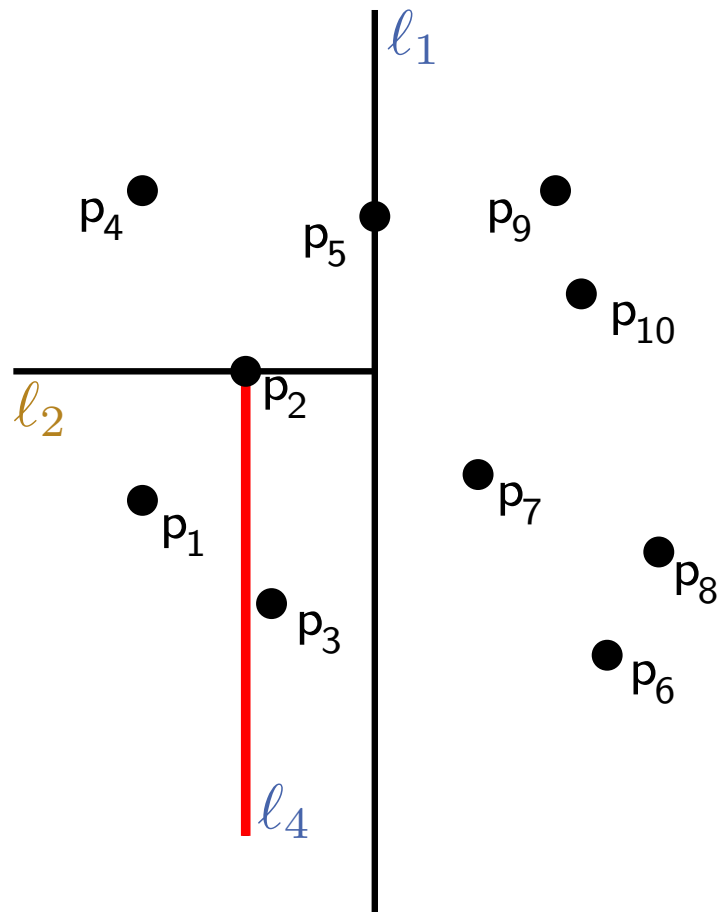
kd-Trees: Beispiel



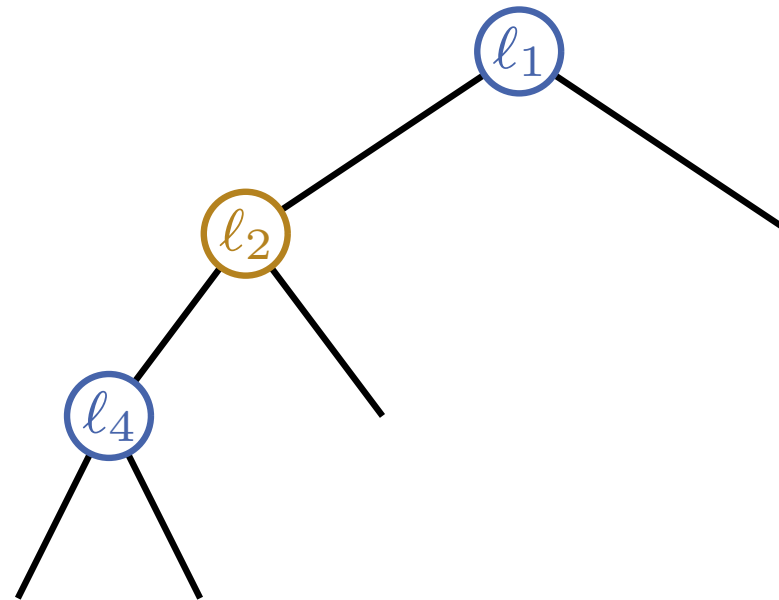
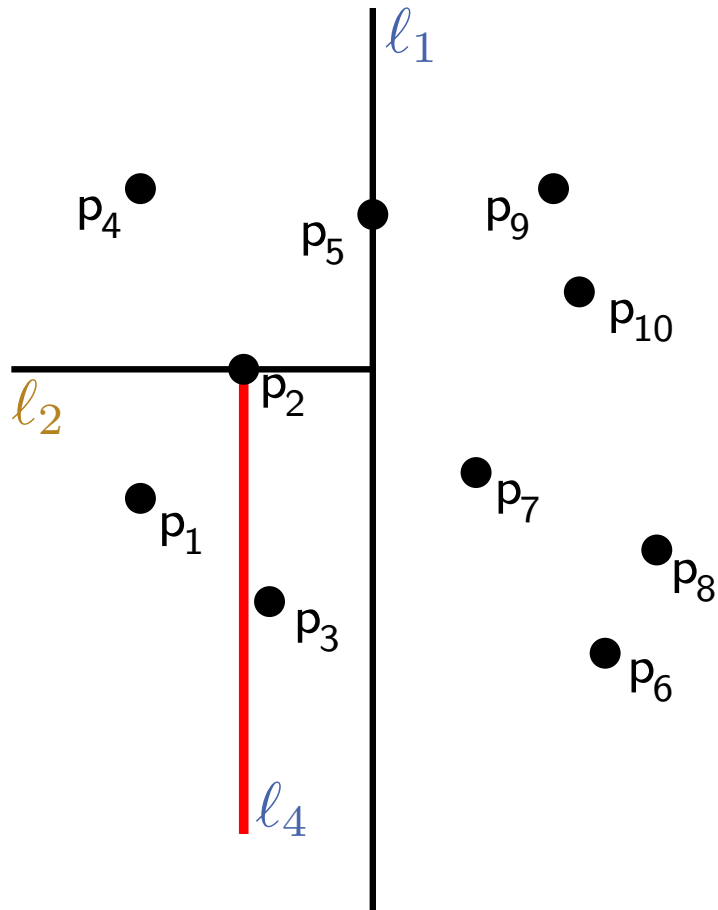
kd-Trees: Beispiel



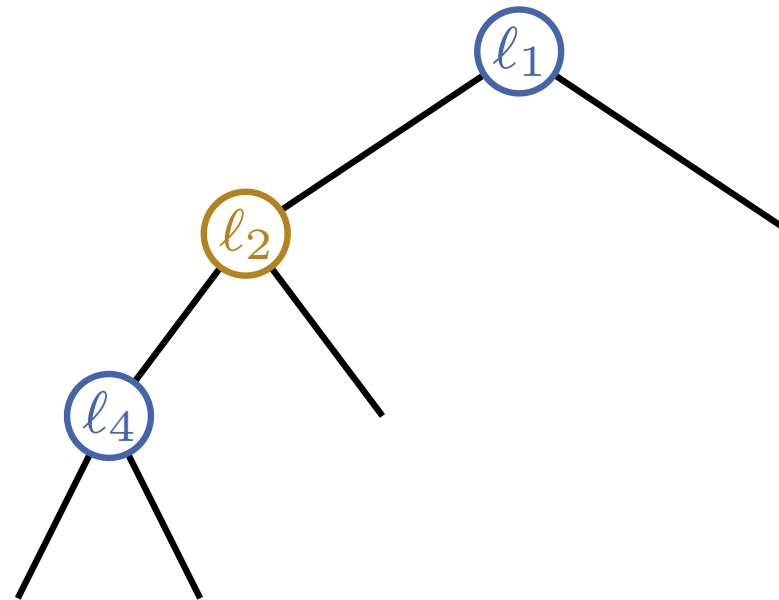
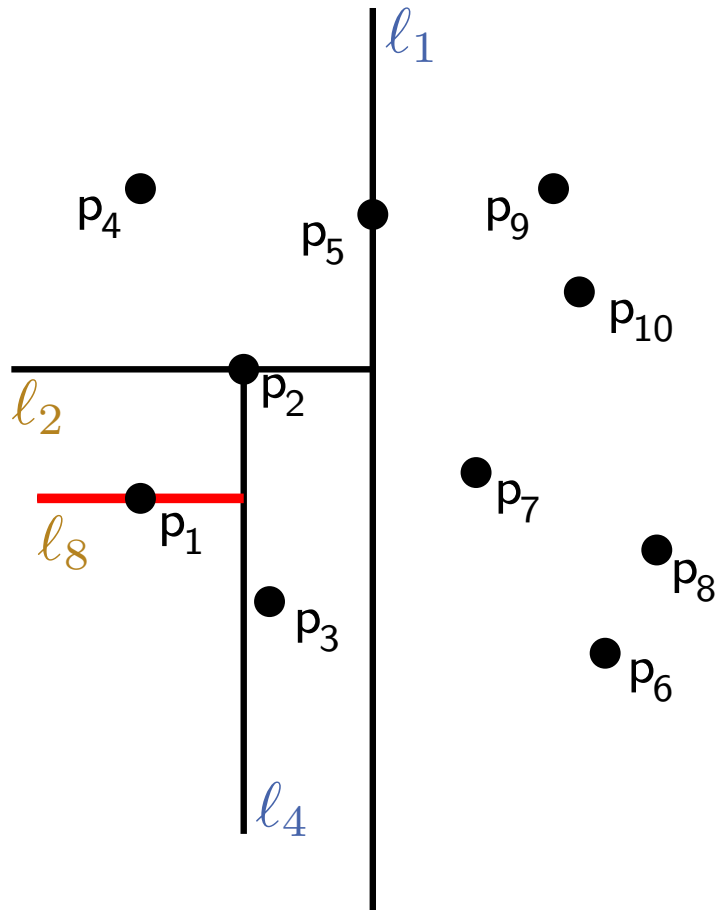
kd-Trees: Beispiel



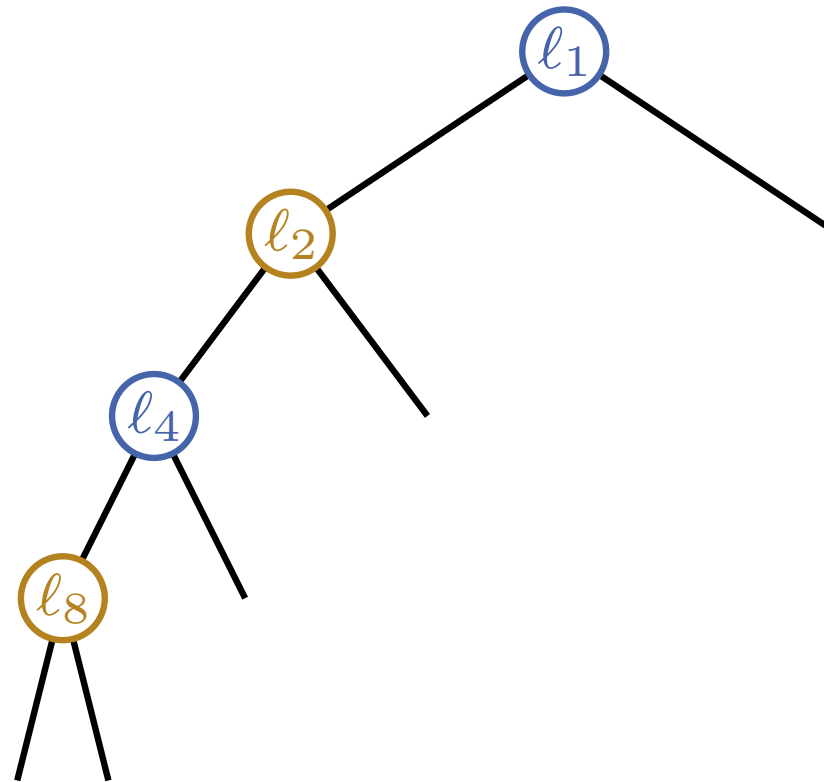
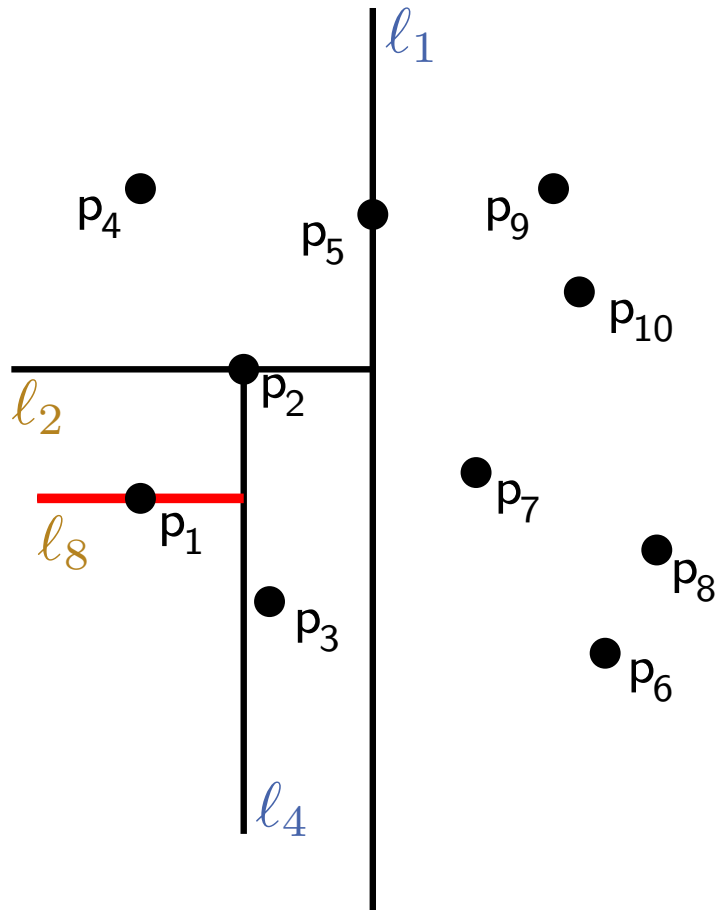
kd-Trees: Beispiel



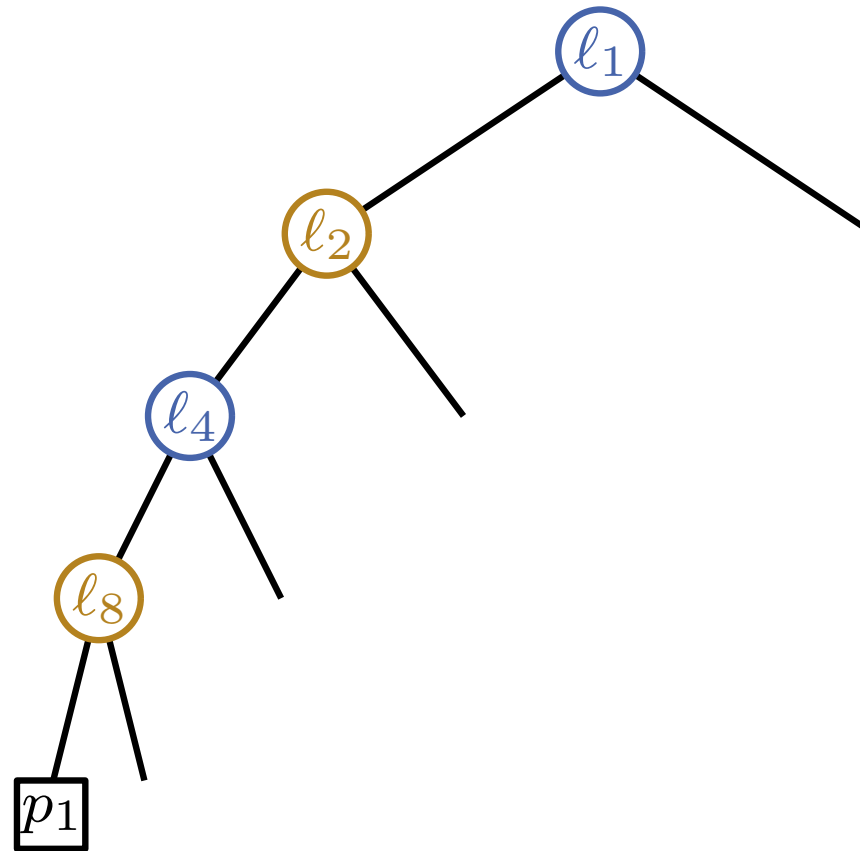
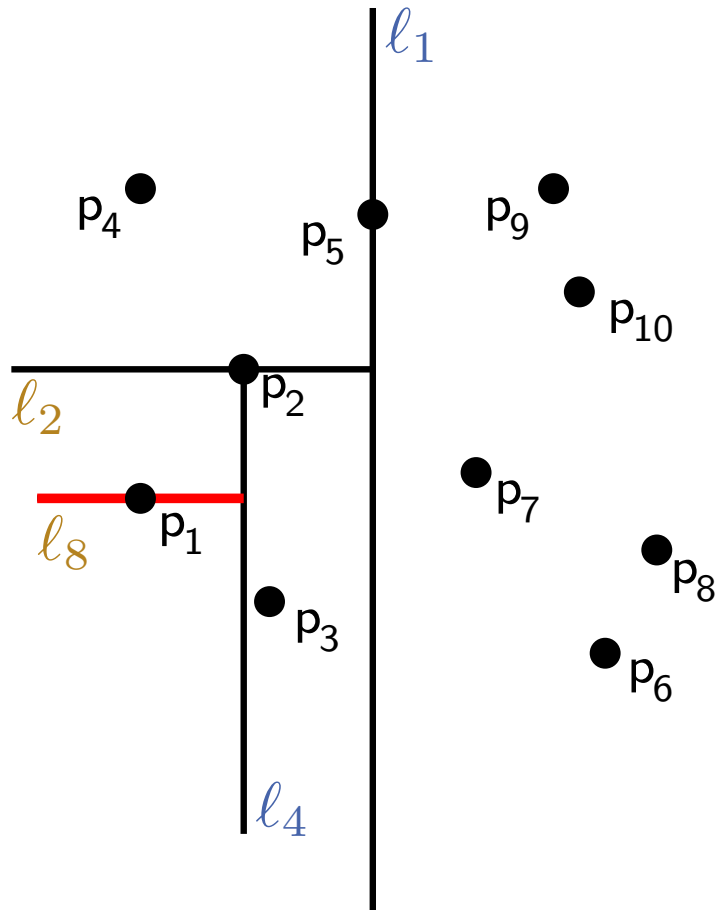
kd-Trees: Beispiel



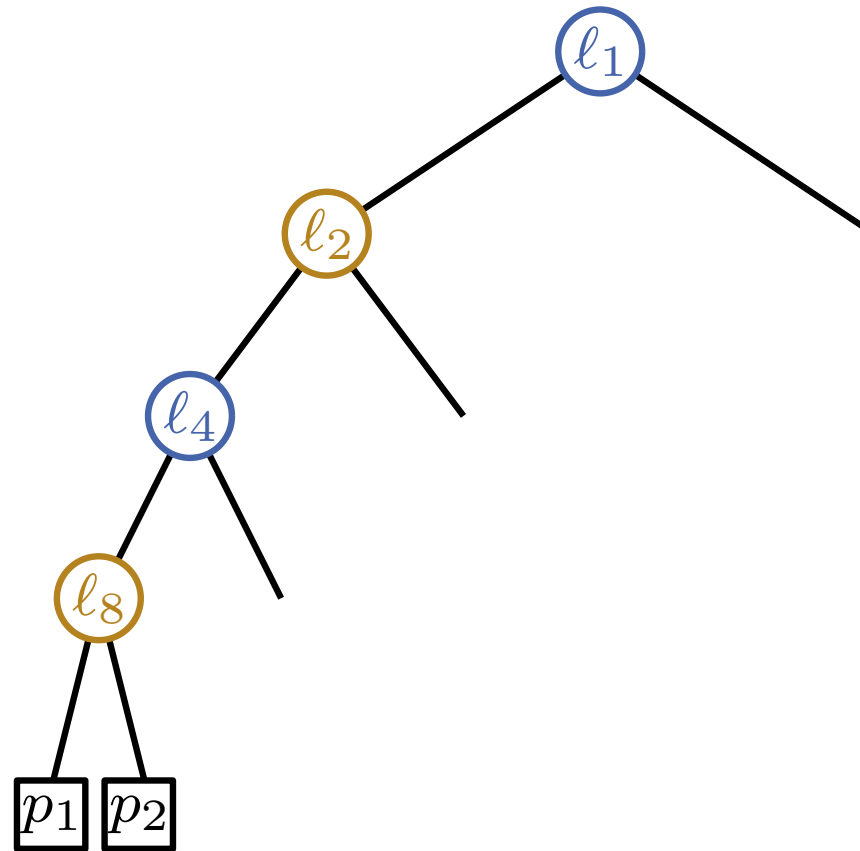
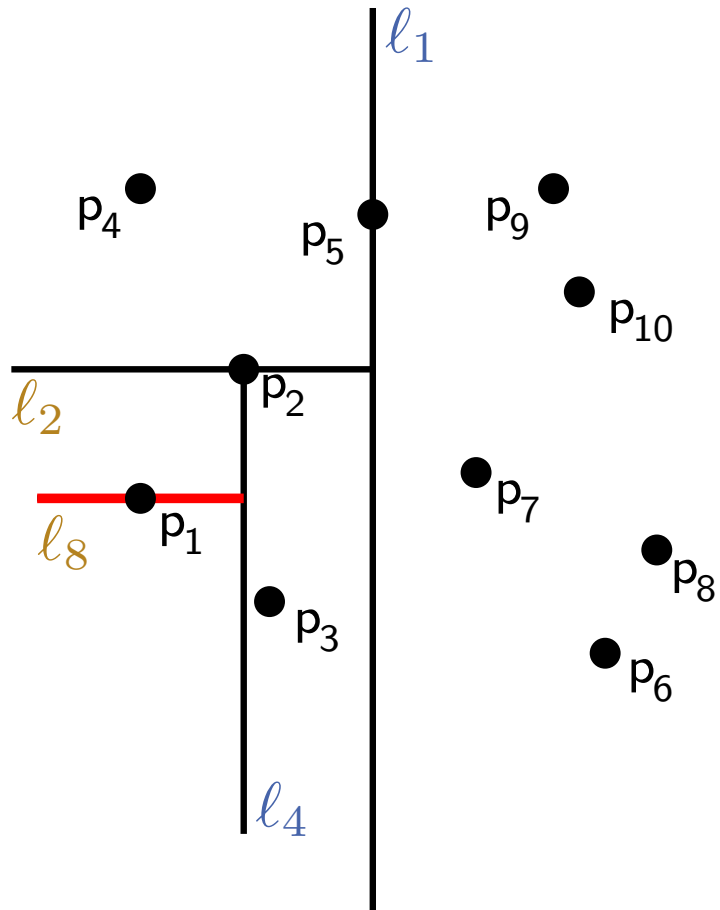
kd-Trees: Beispiel



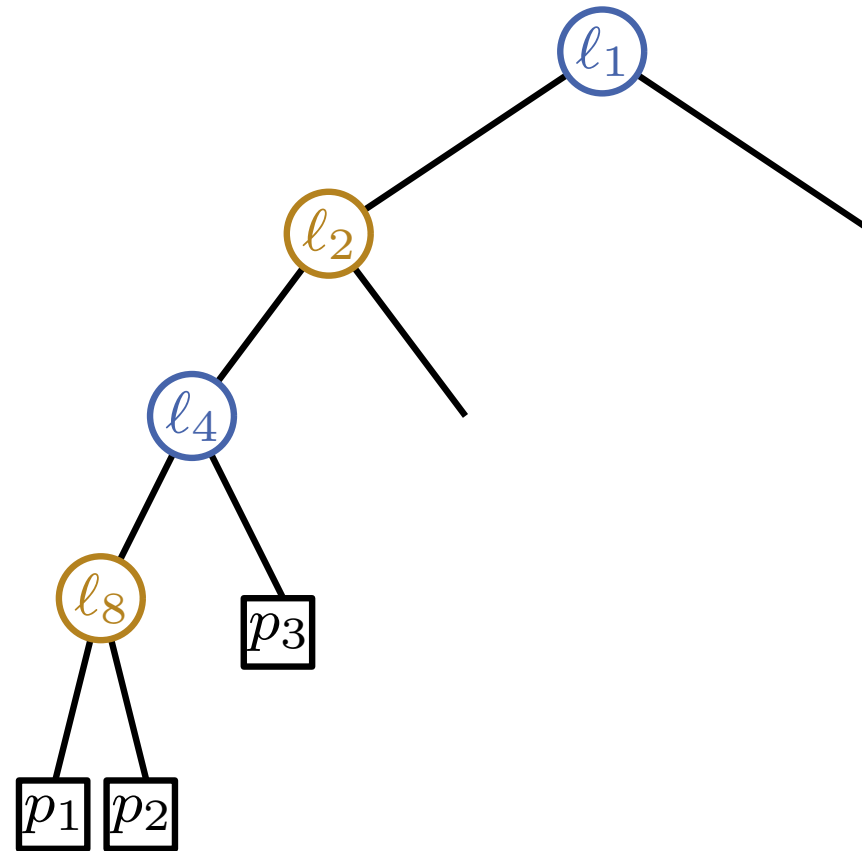
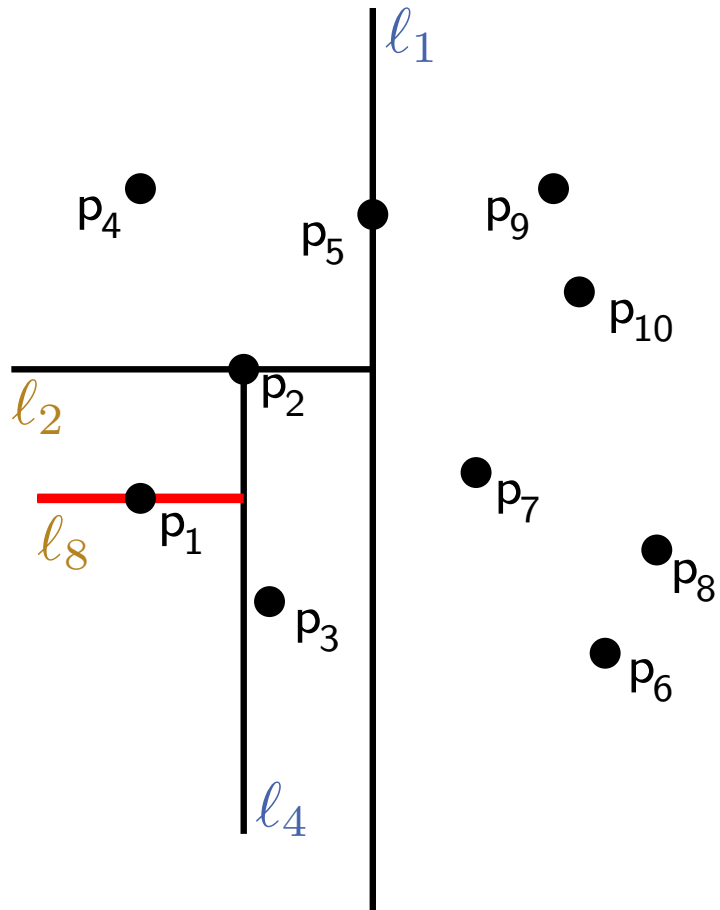
kd-Trees: Beispiel



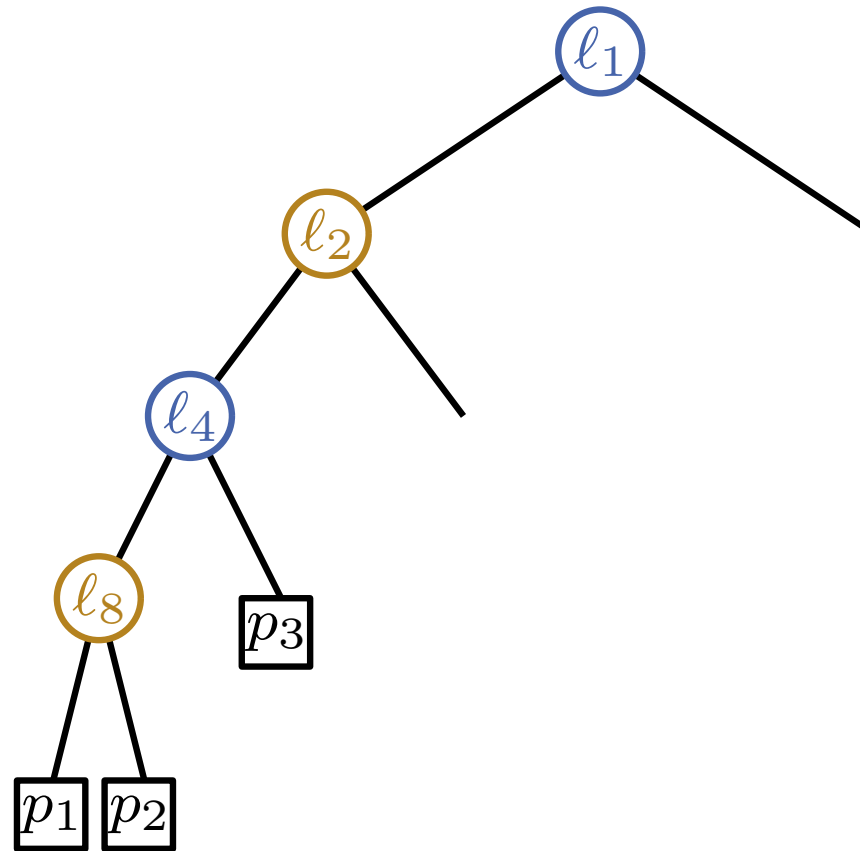
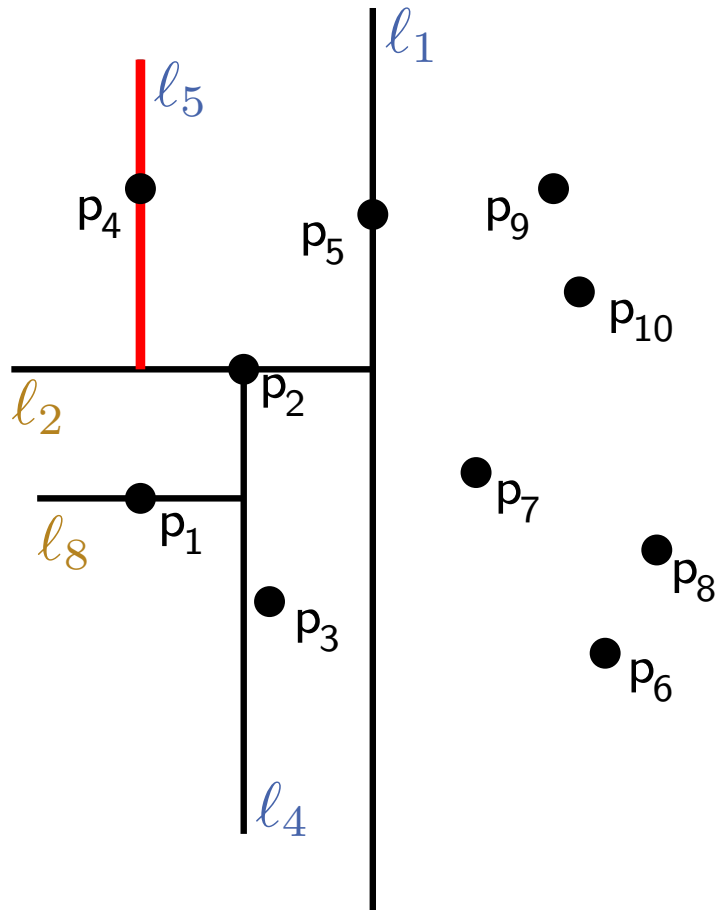
kd-Trees: Beispiel



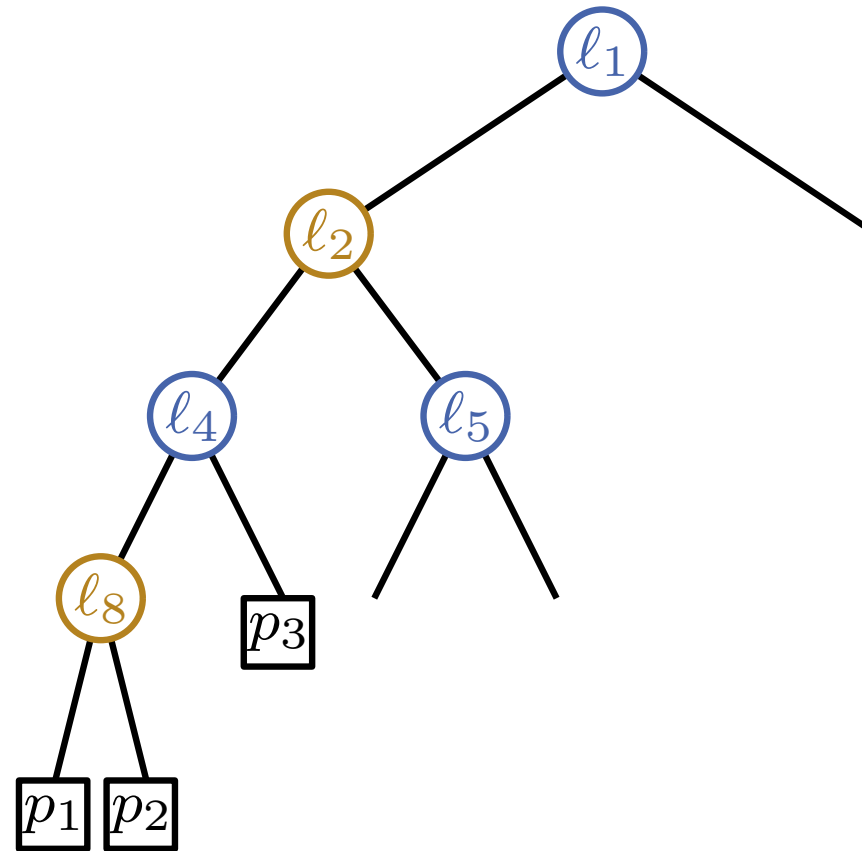
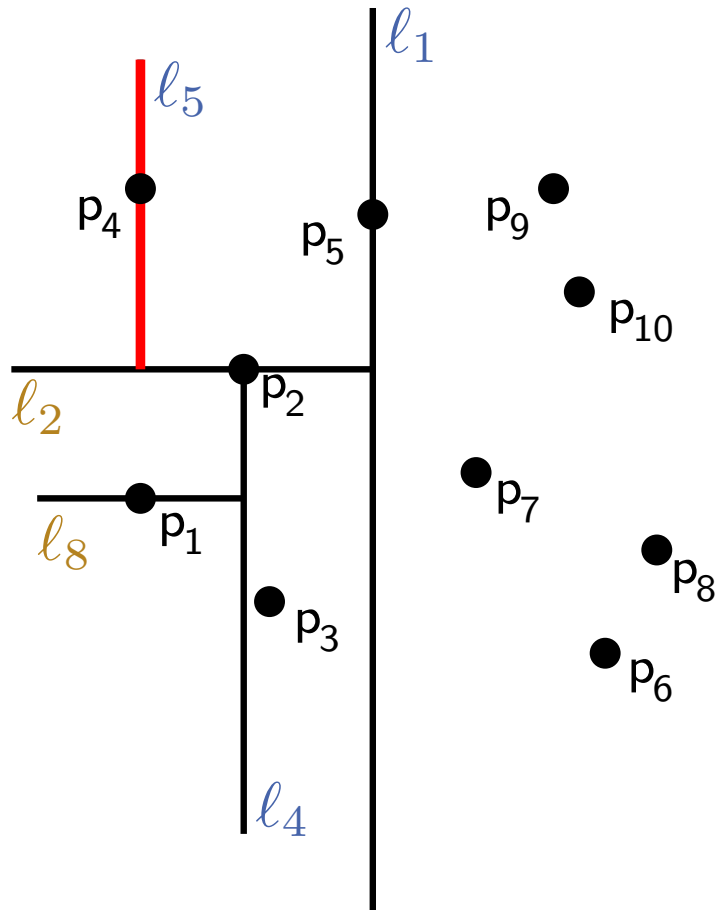
kd-Trees: Beispiel



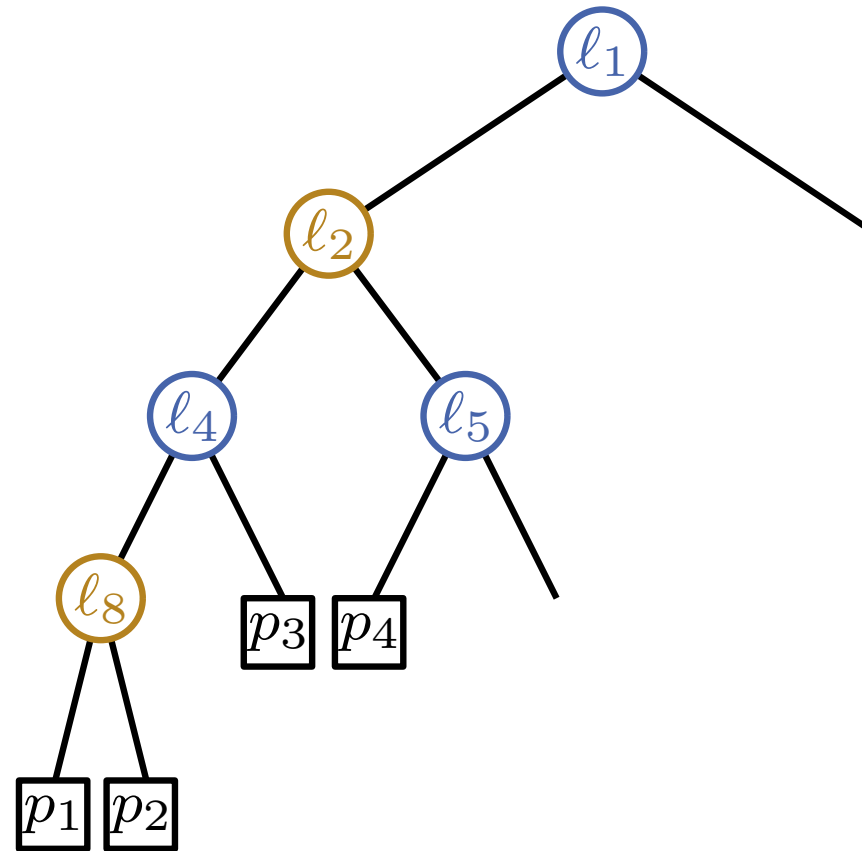
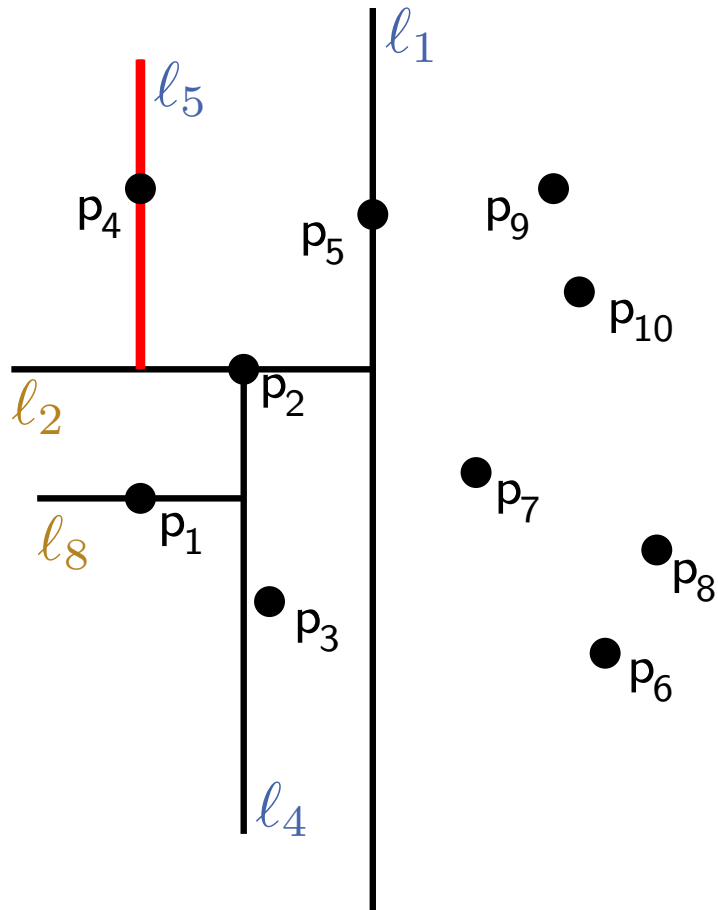
kd-Trees: Beispiel



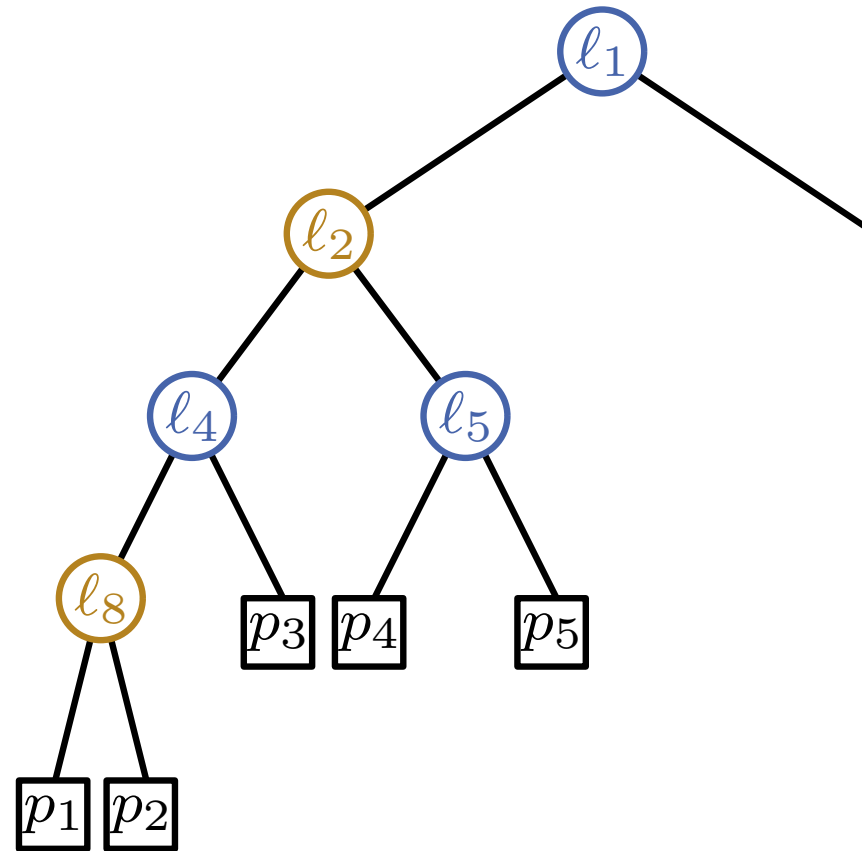
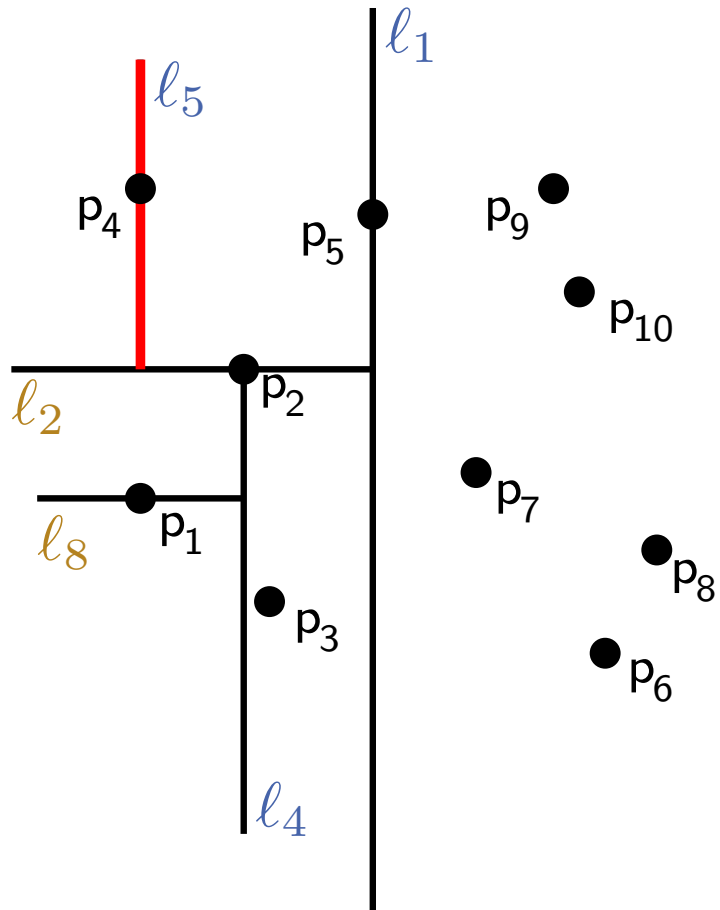
kd-Trees: Beispiel



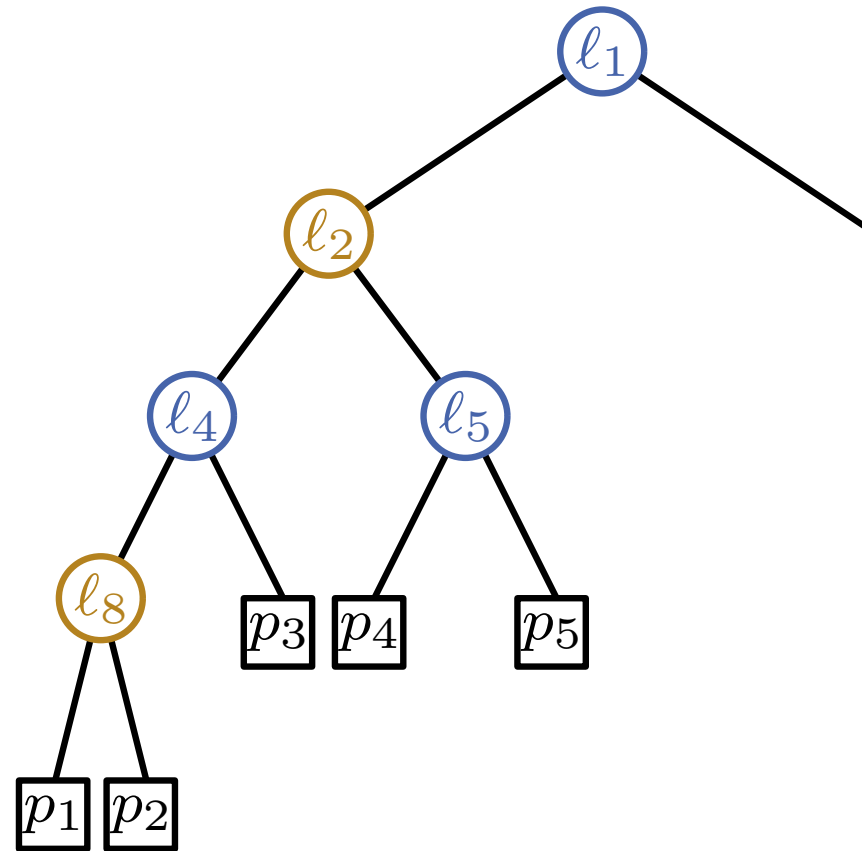
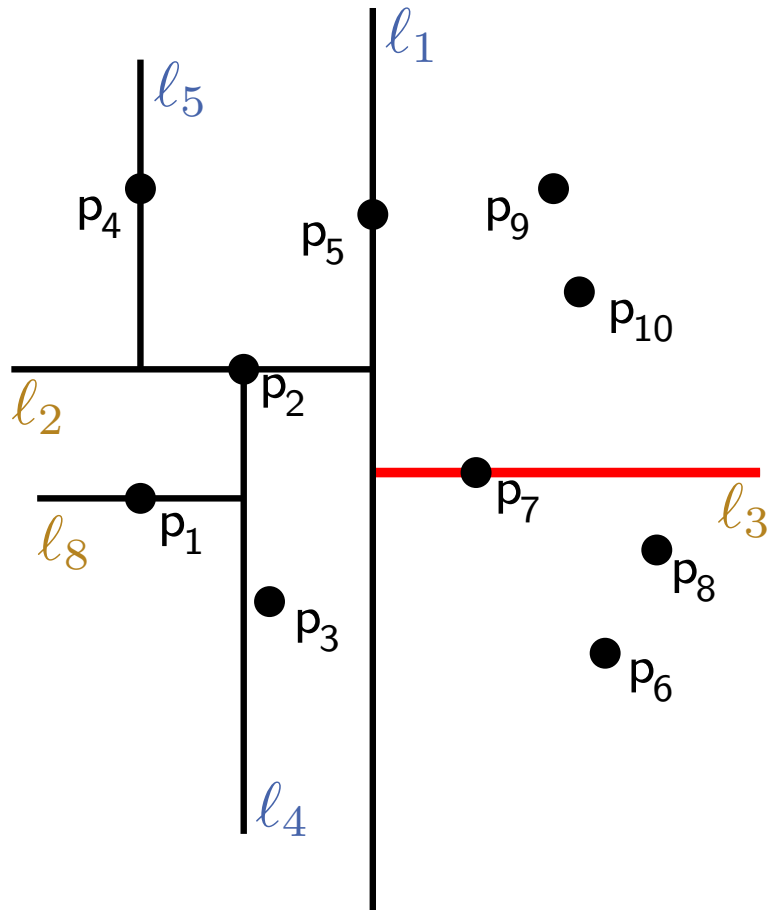
kd-Trees: Beispiel



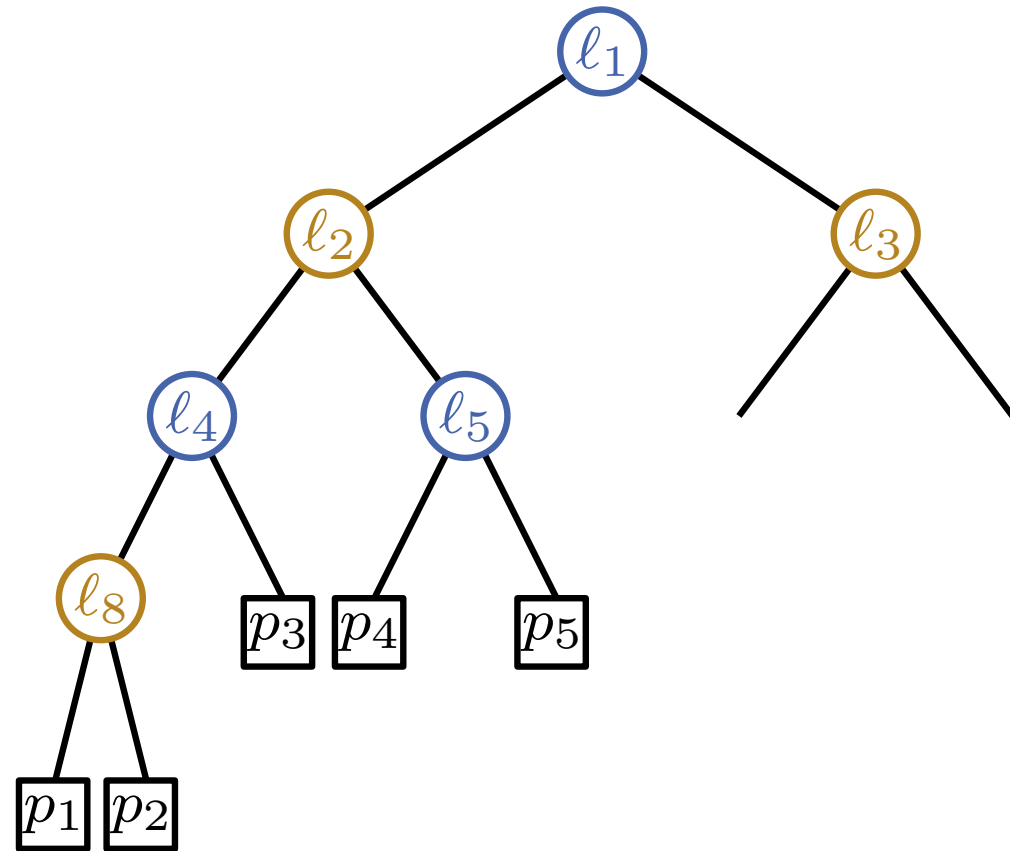
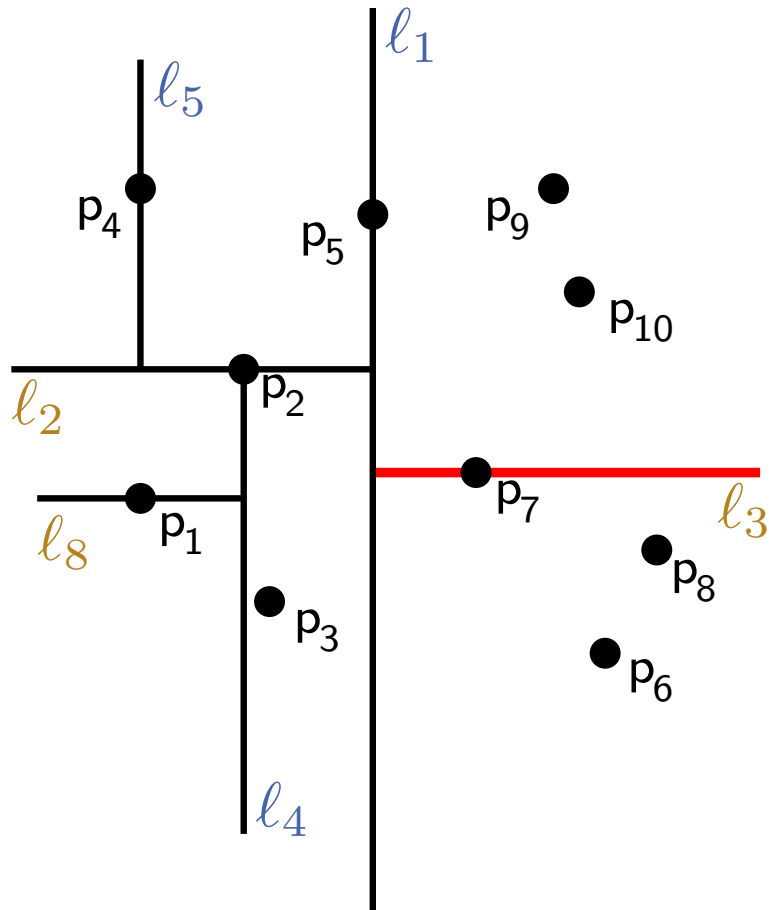
kd-Trees: Beispiel



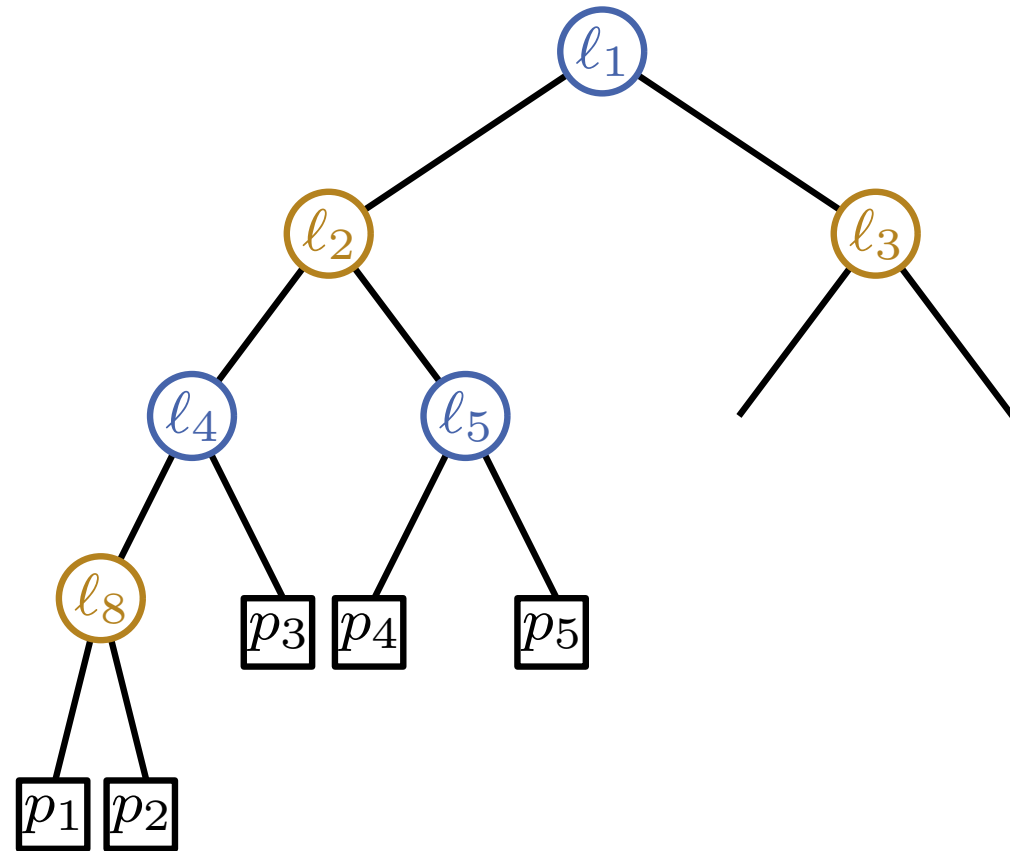
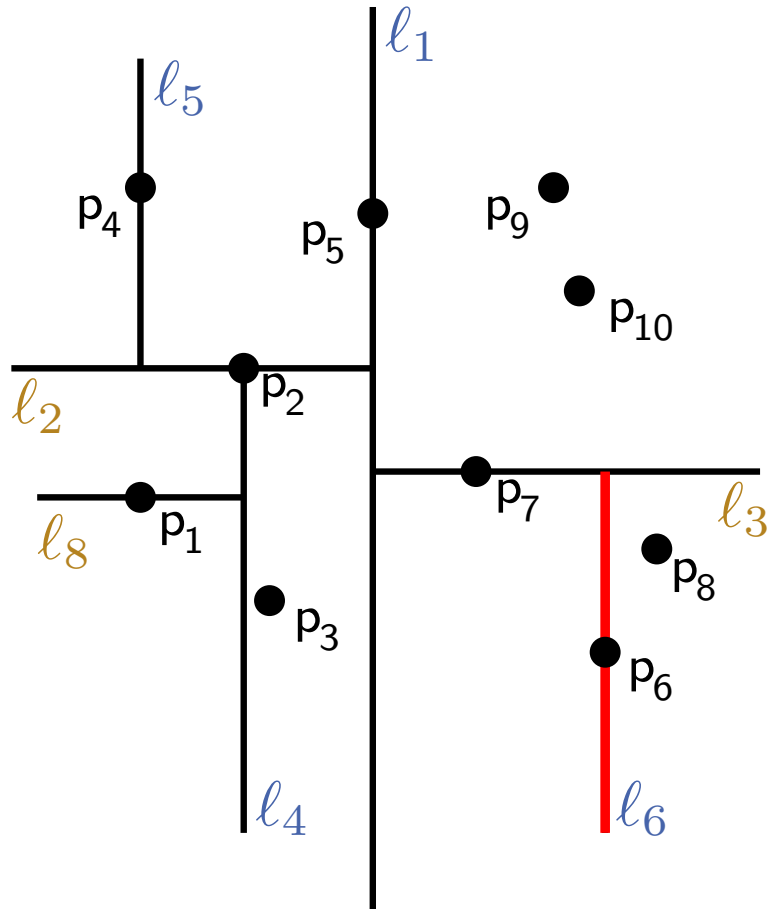
kd-Trees: Beispiel



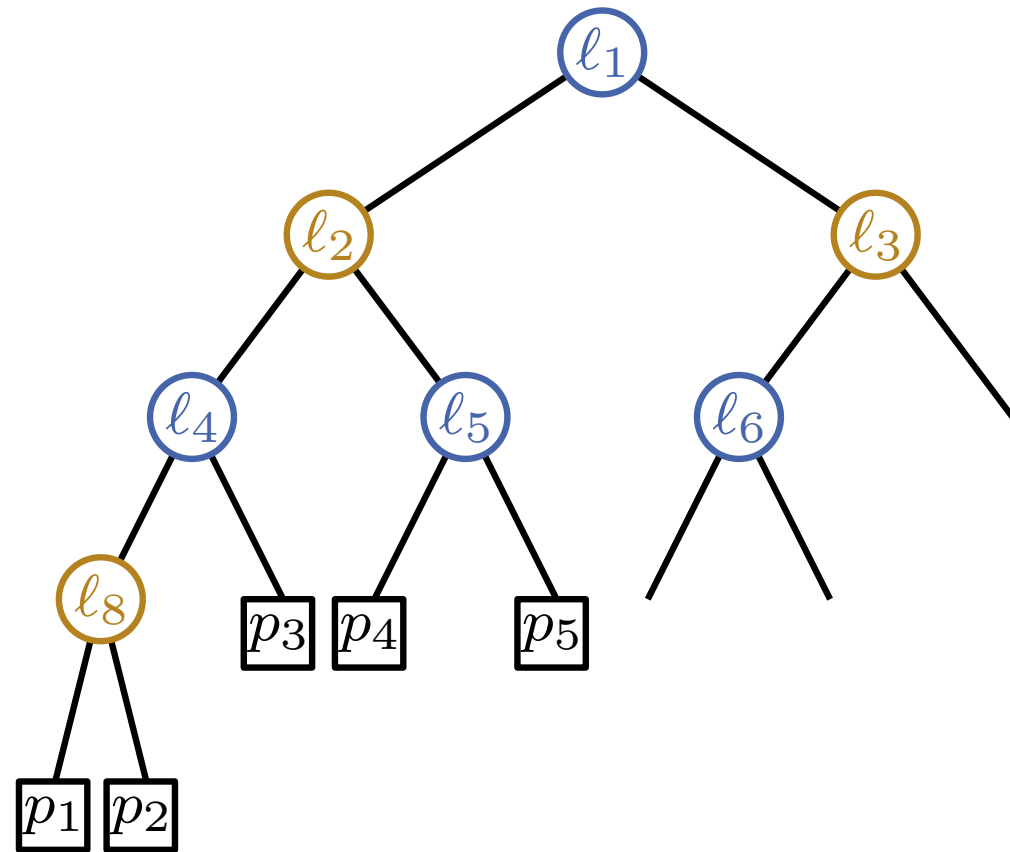
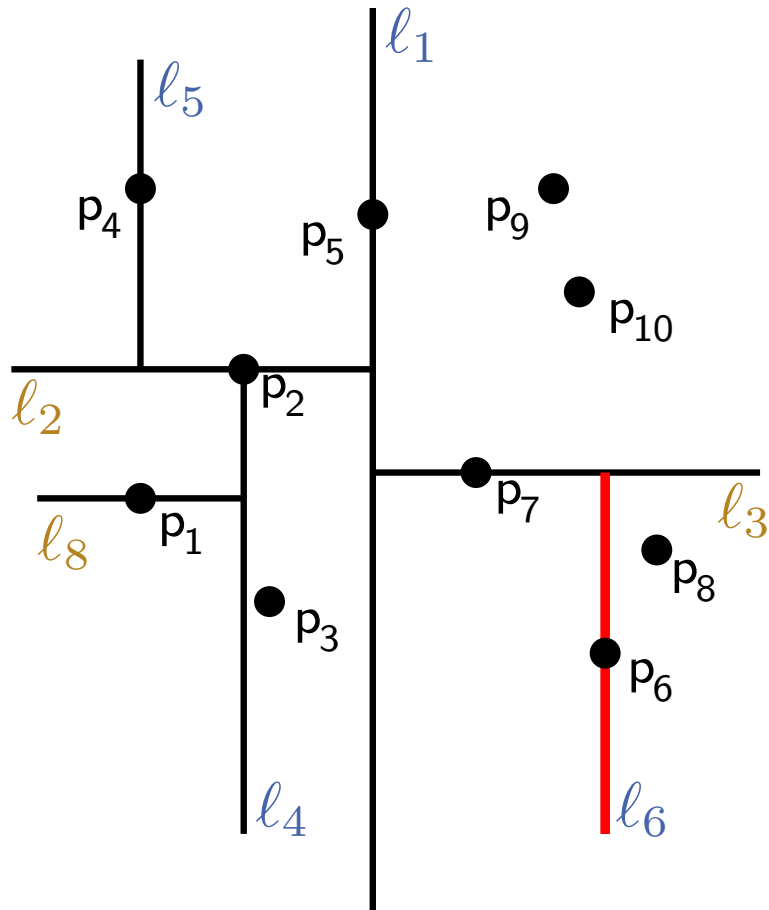
kd-Trees: Beispiel



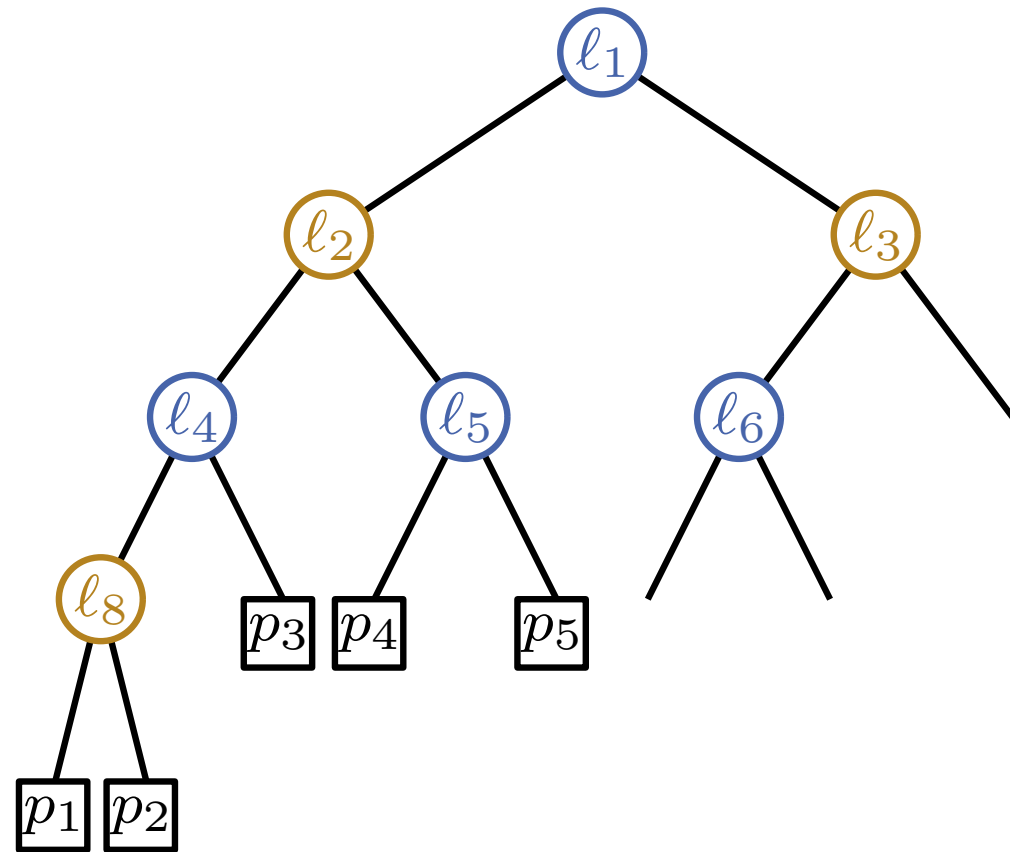
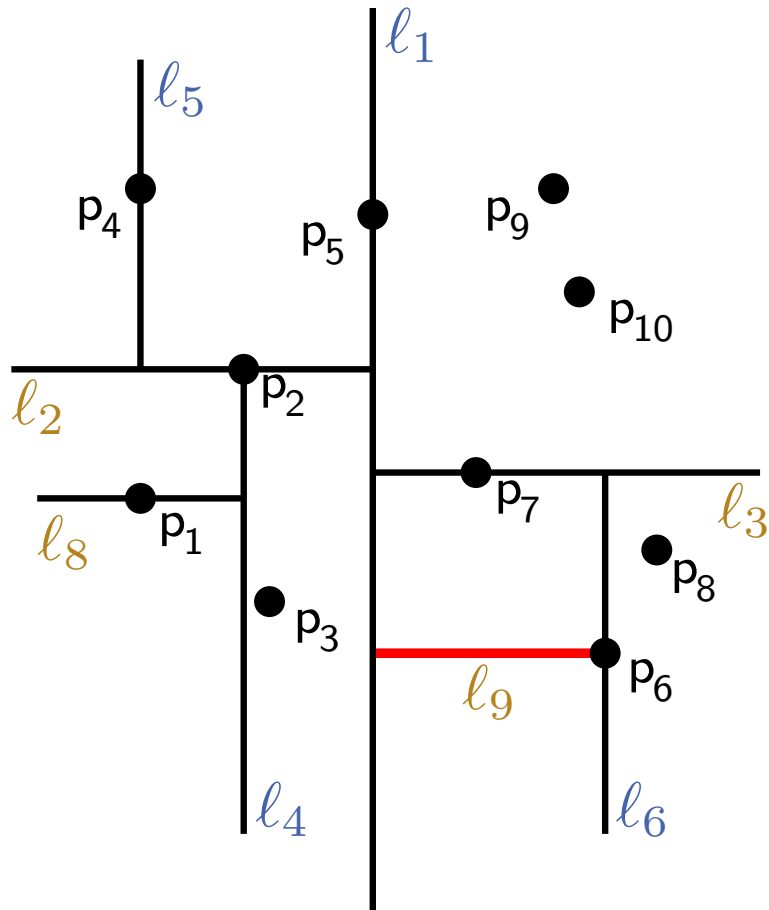
kd-Trees: Beispiel



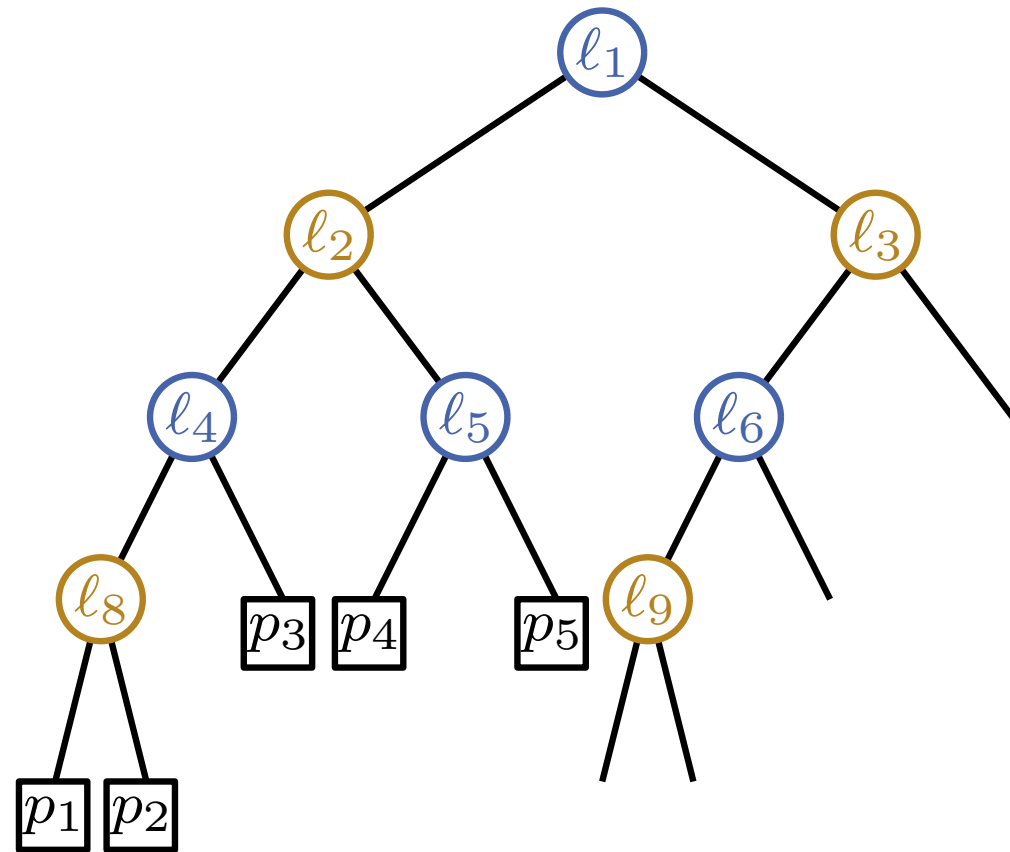
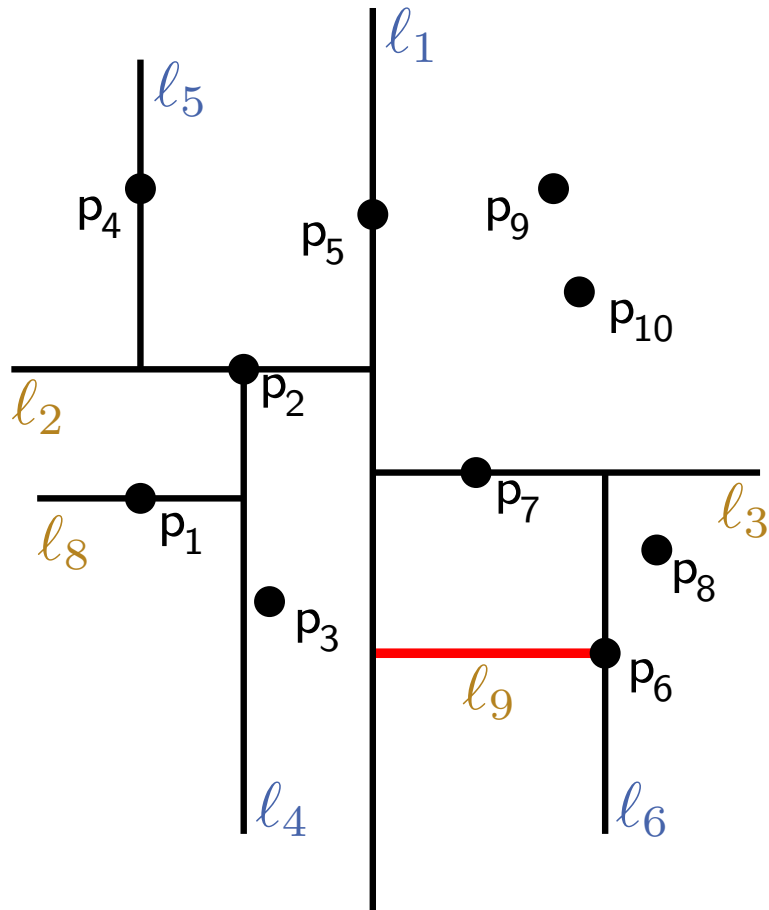
kd-Trees: Beispiel



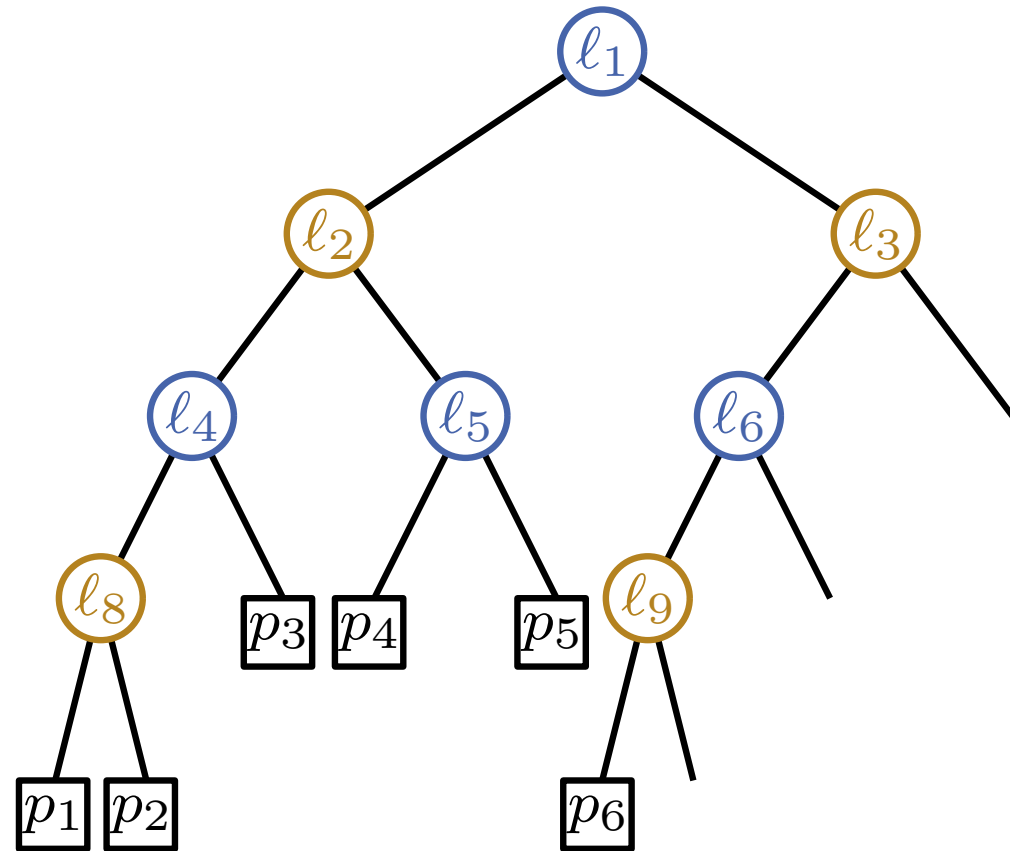
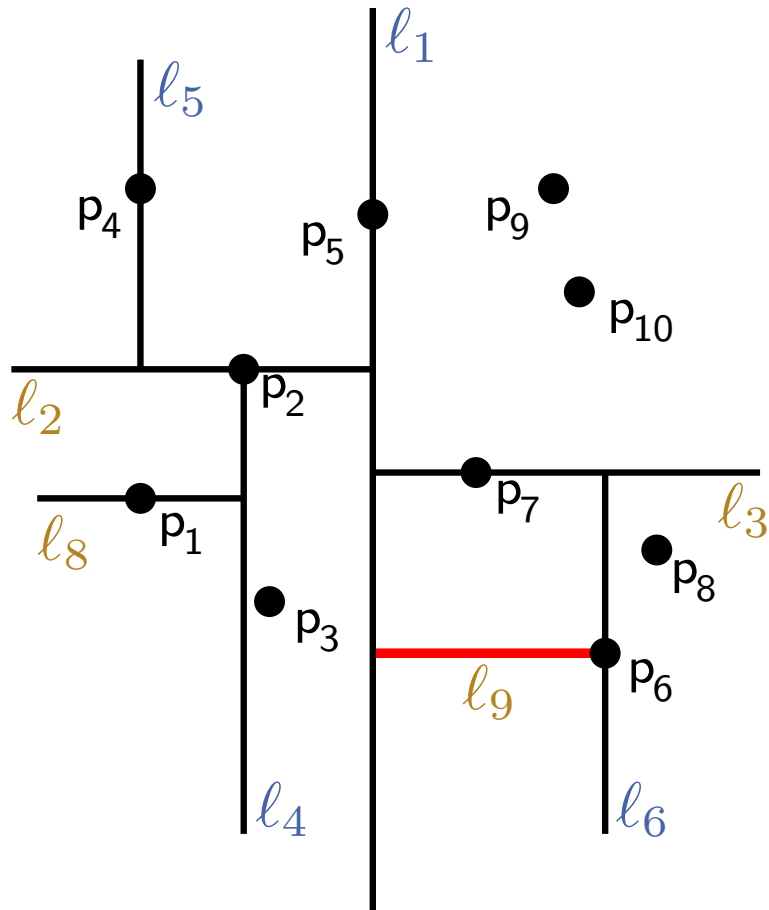
kd-Trees: Beispiel



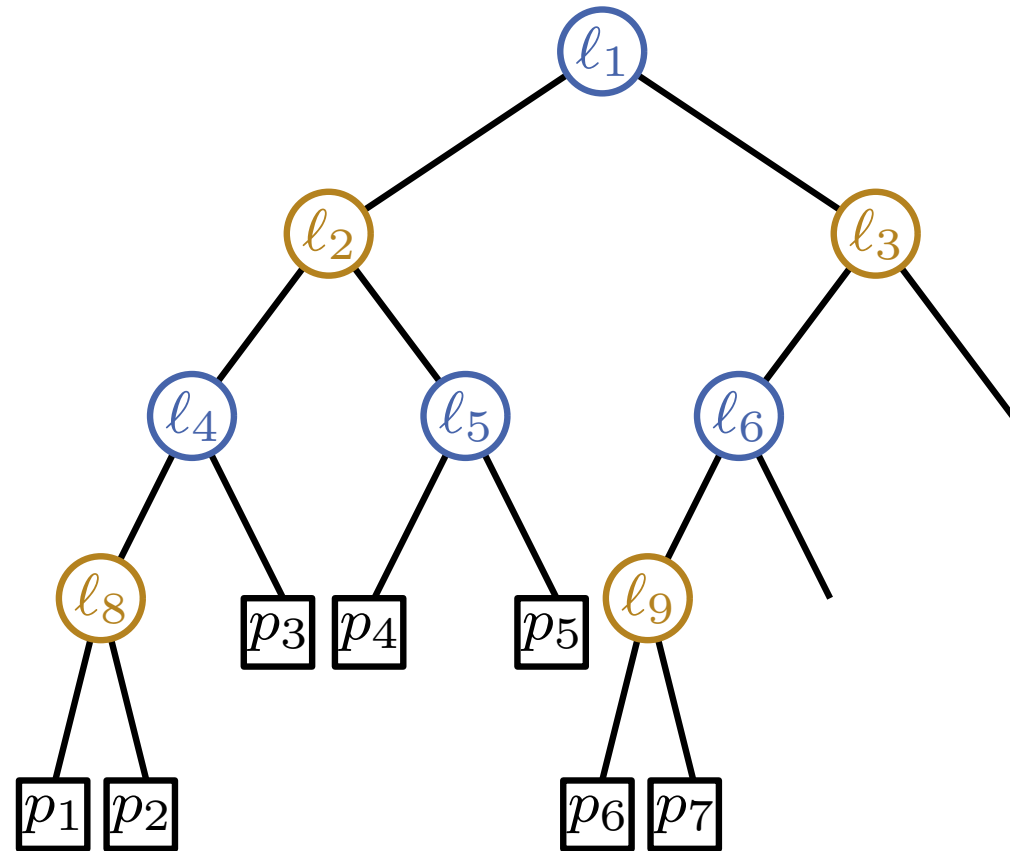
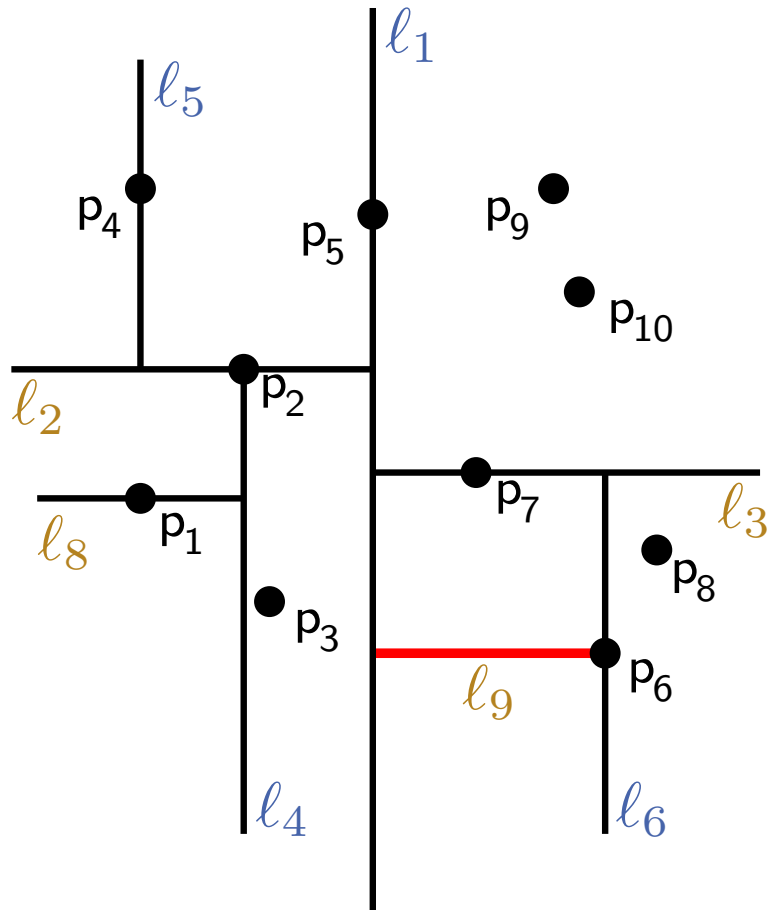
kd-Trees: Beispiel



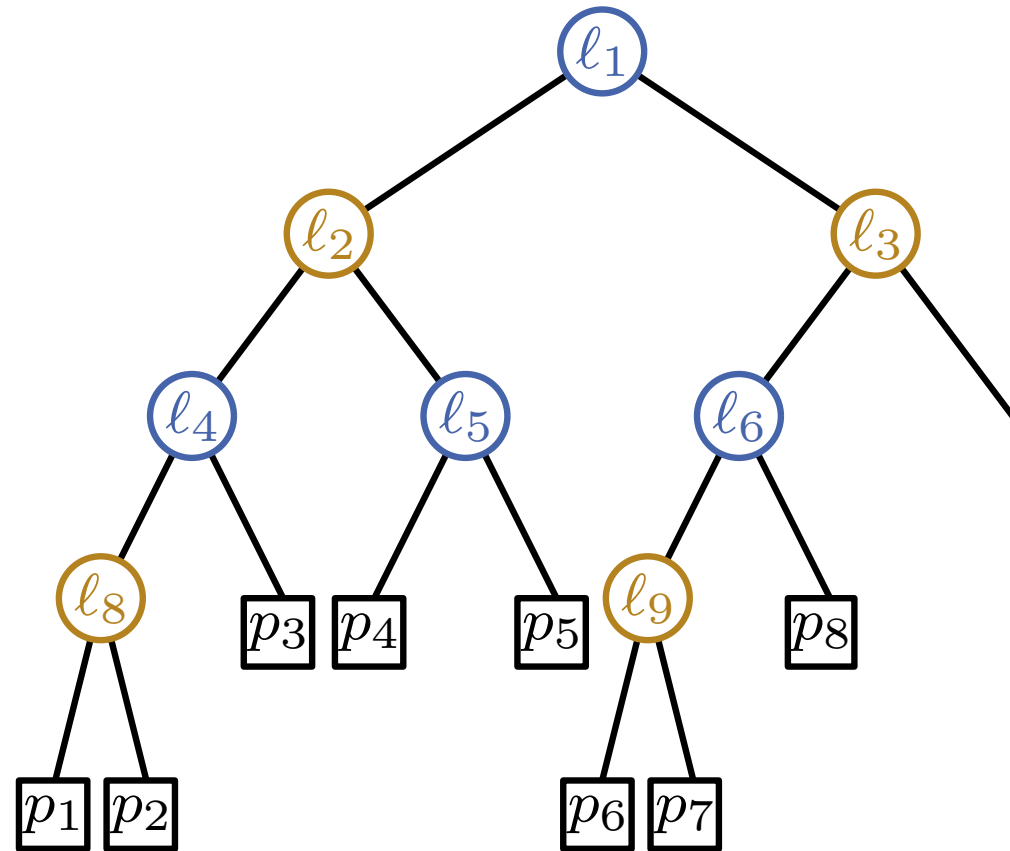
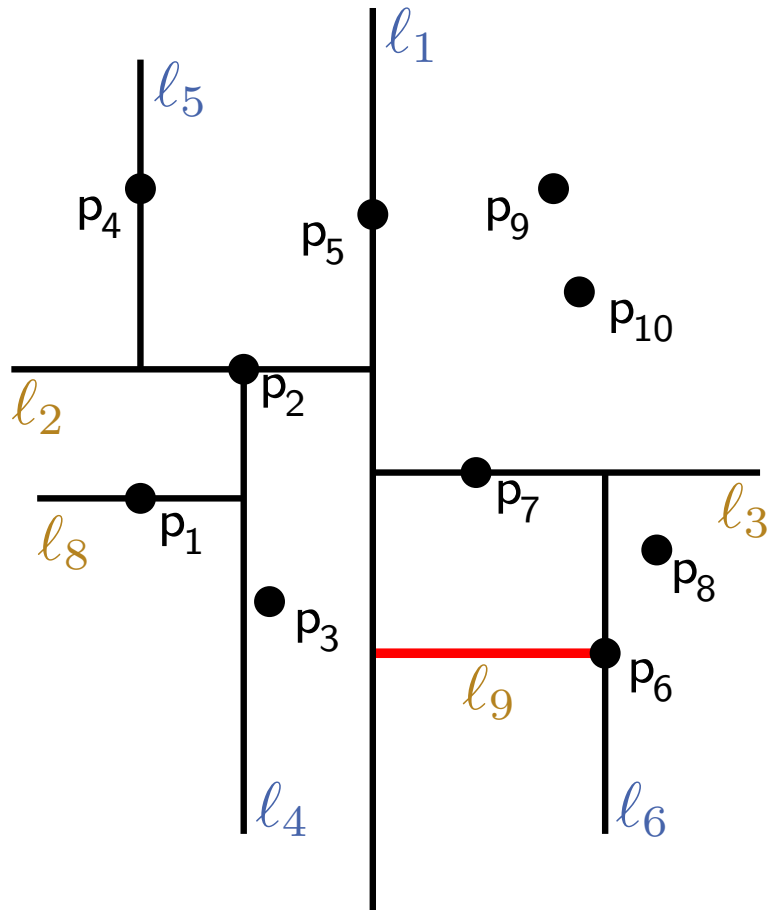
kd-Trees: Beispiel



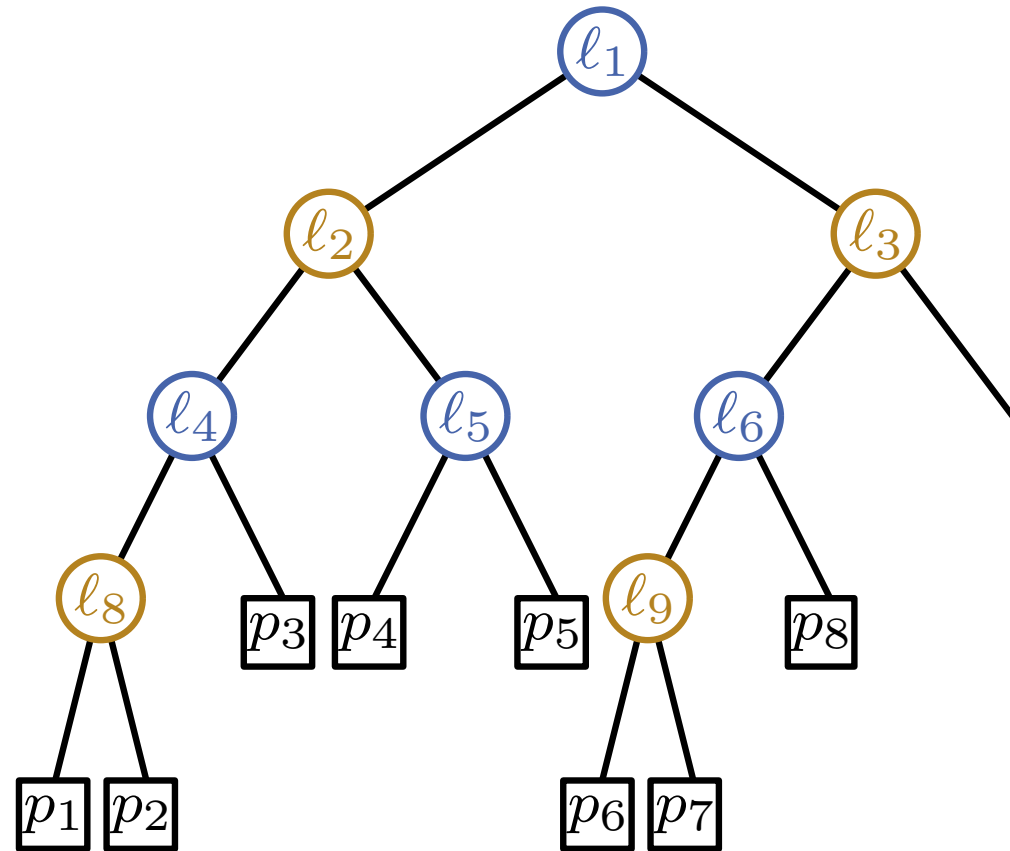
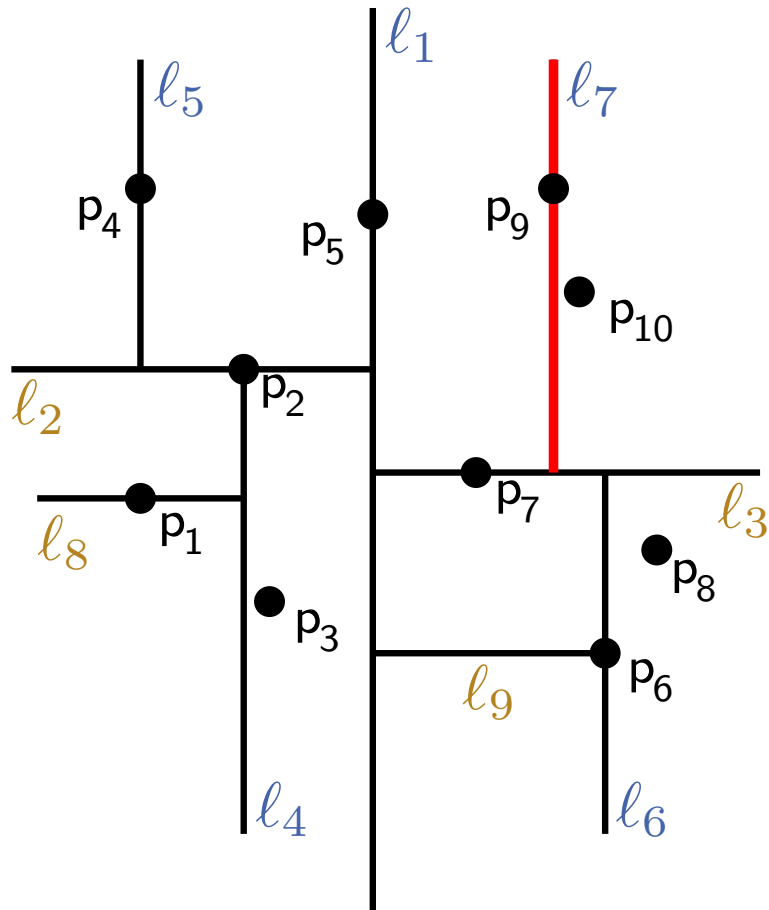
kd-Trees: Beispiel



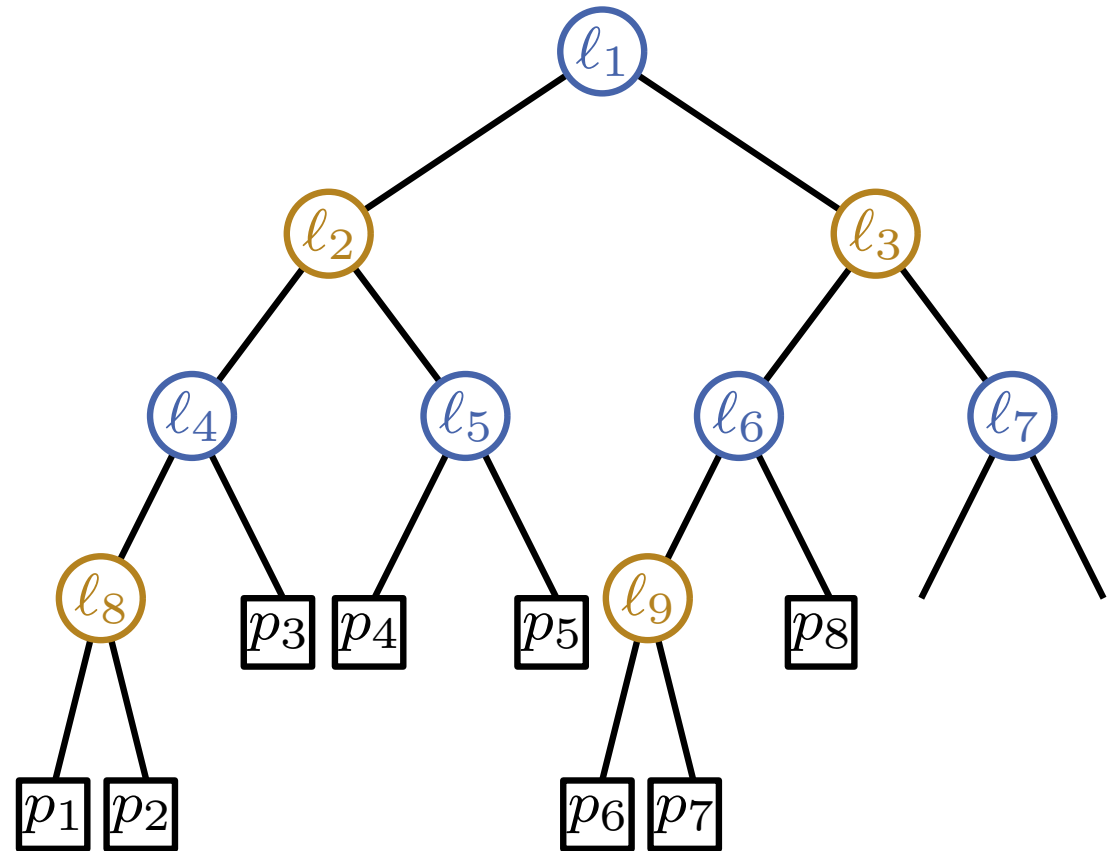
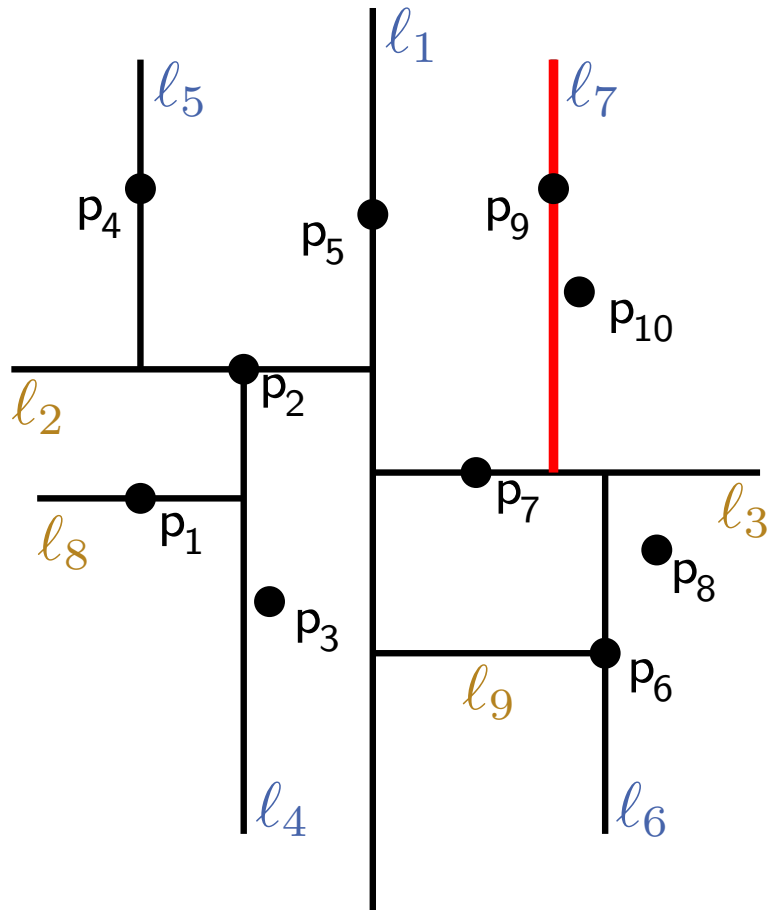
kd-Trees: Beispiel



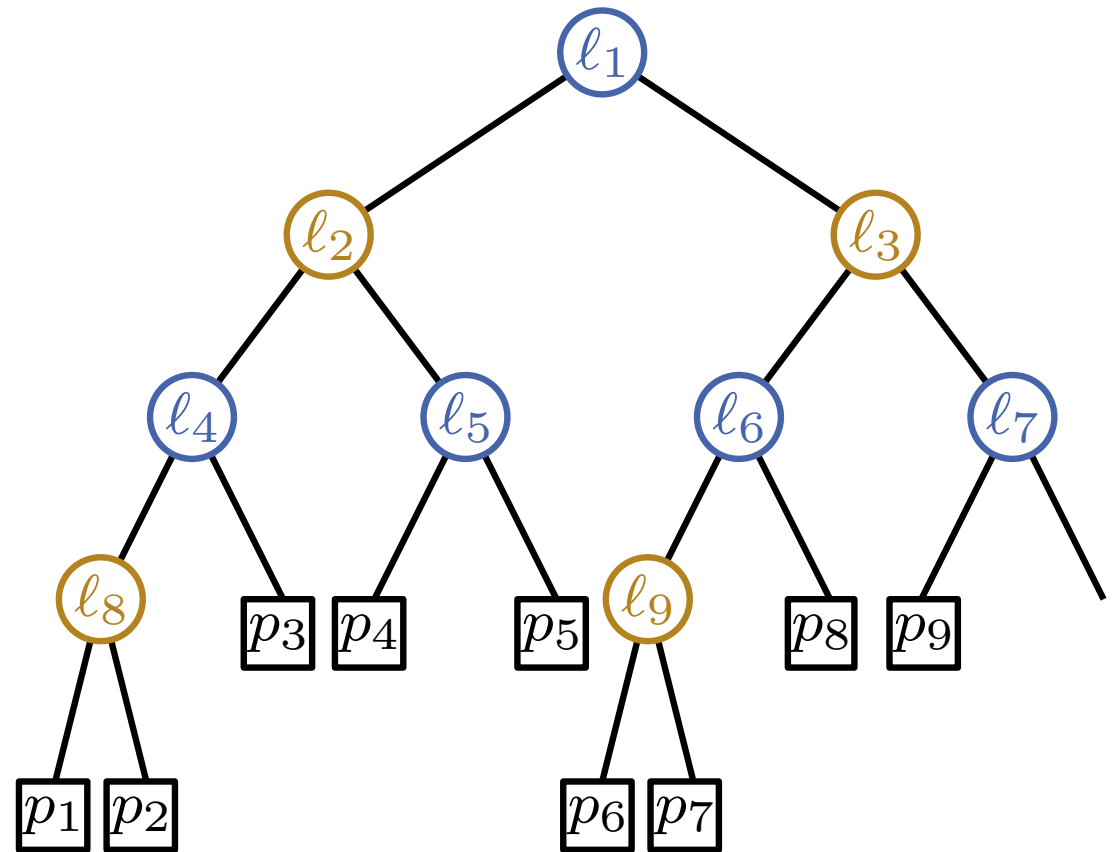
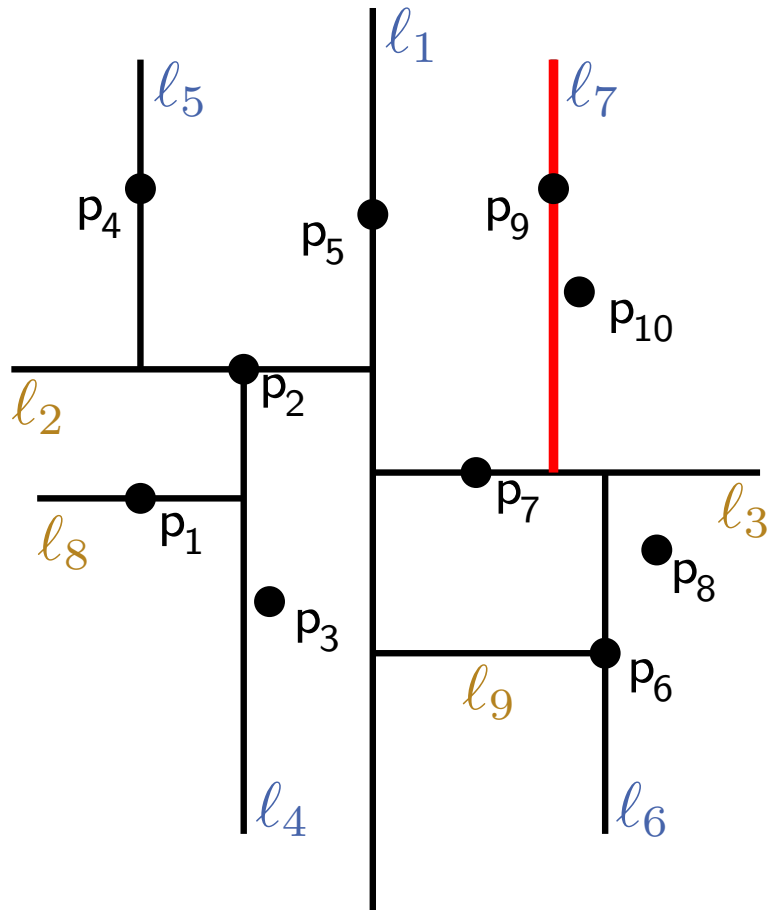
kd-Trees: Beispiel



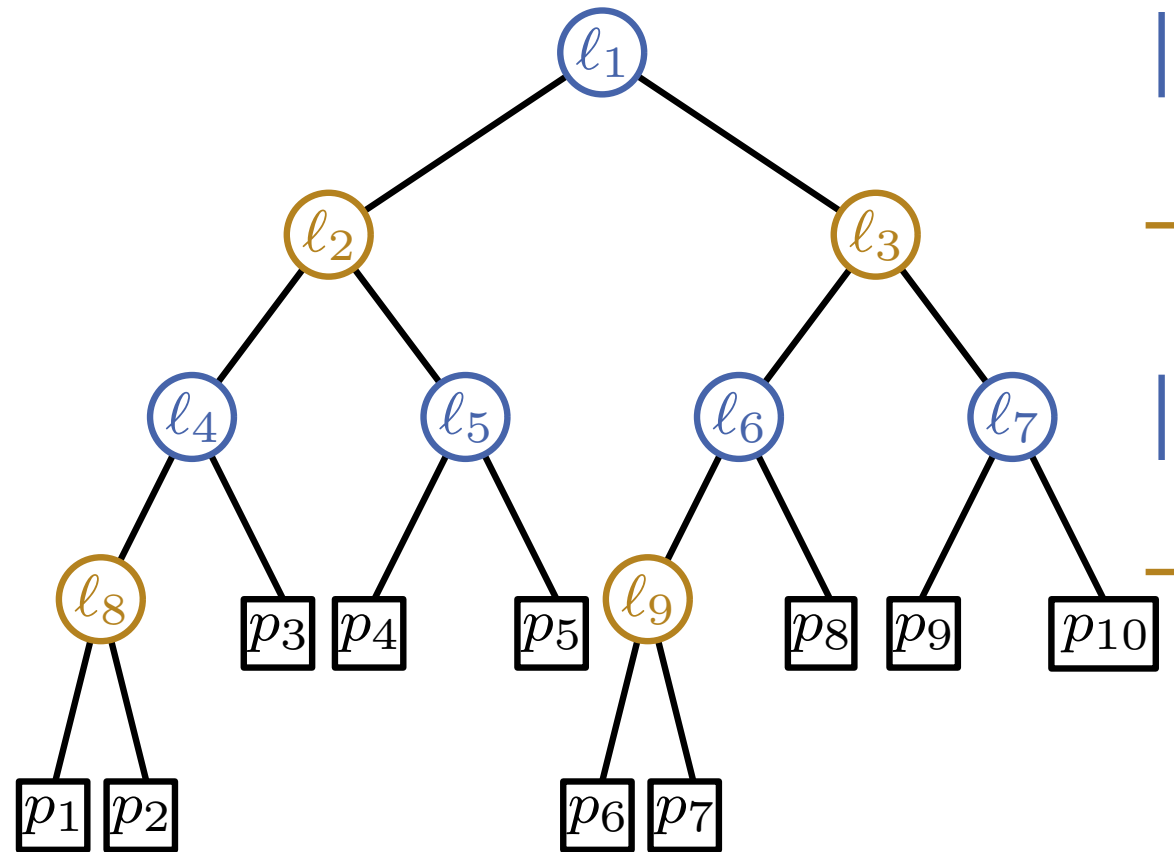
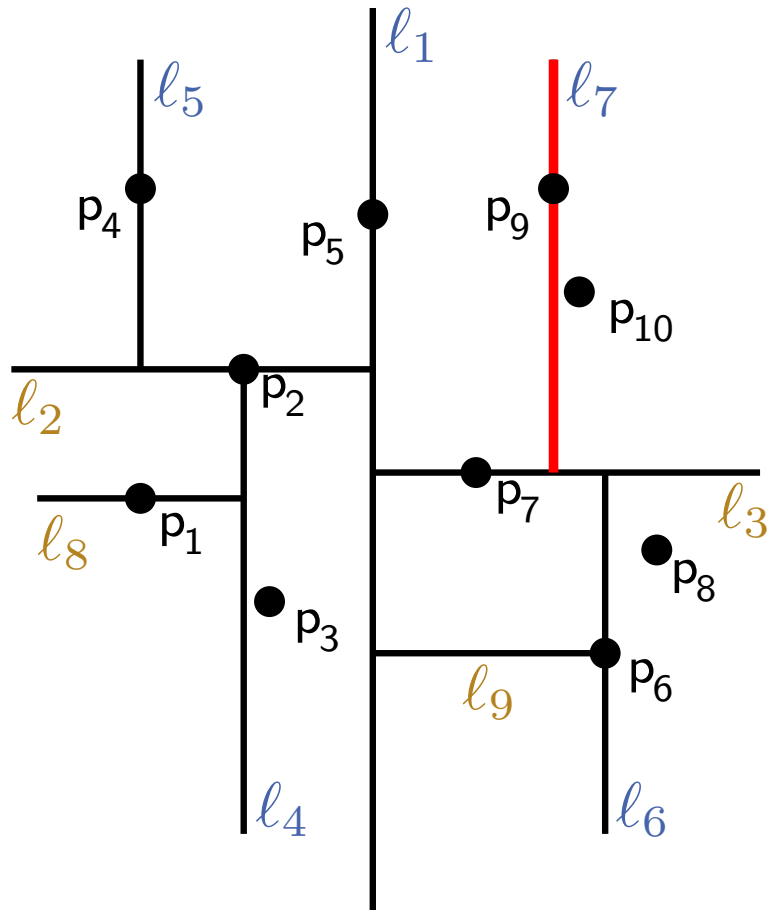
kd-Trees: Beispiel



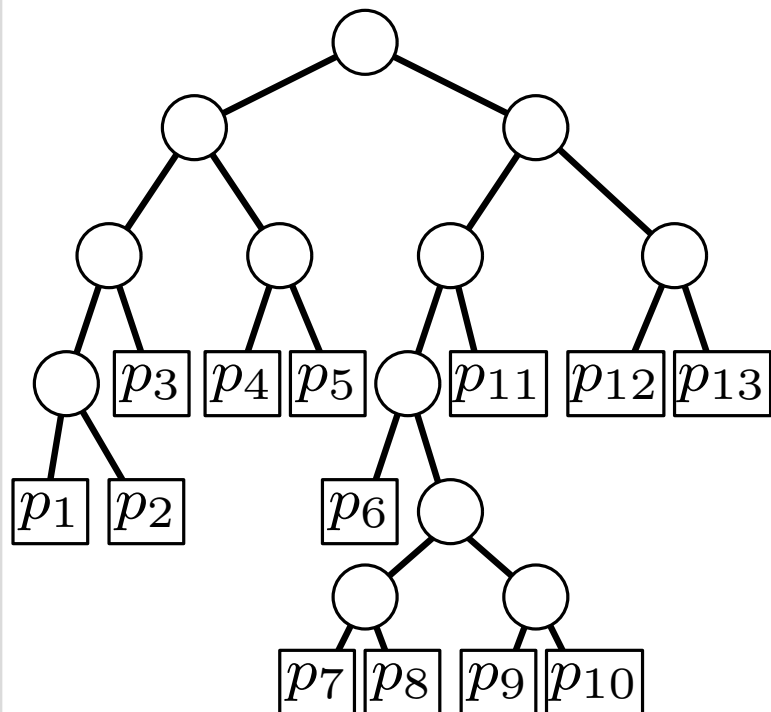
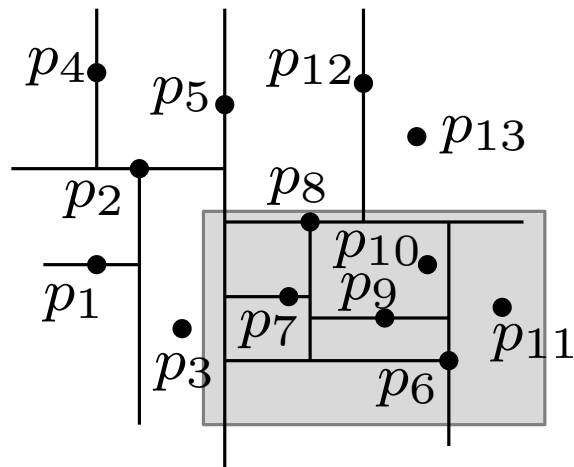
kd-Trees: Beispiel



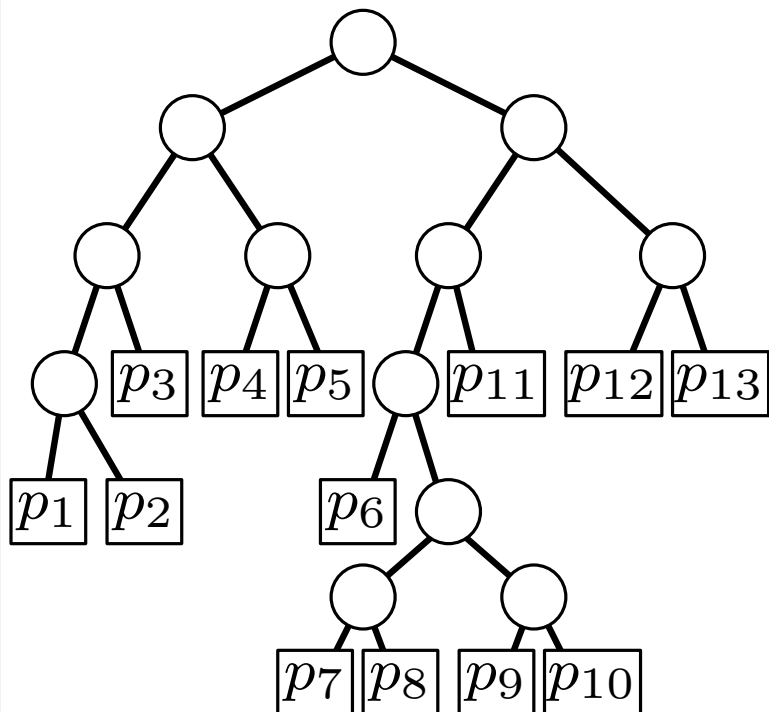
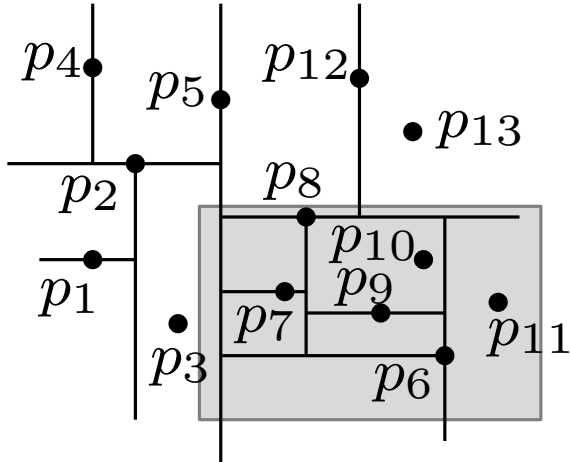
kd-Trees: Beispiel



Bereichsabfrage in einem kd -Tree



Bereichsabfrage in einem kd -Tree



SearchKdTree(v, R)

if v Blatt **then**

| prüfe Punkt p in v auf $p \in R$

else

if $\text{region}(\text{lc}(v)) \subseteq R$ **then**

| ReportSubtree($\text{lc}(v)$)

else

if $\text{region}(\text{lc}(v)) \cap R \neq \emptyset$ **then**

| SearchKdTree($\text{lc}(v), R$)

if $\text{region}(\text{rc}(v)) \subseteq R$ **then**

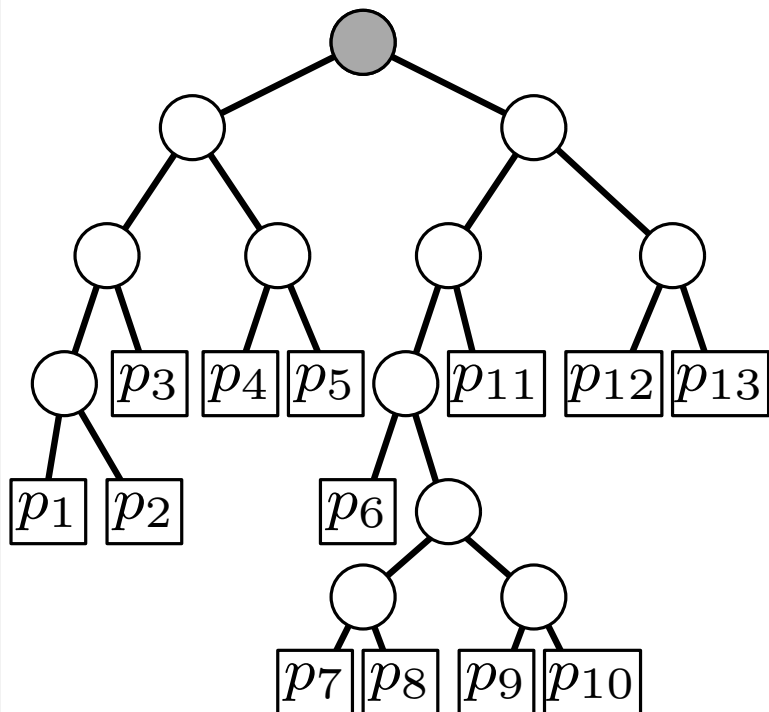
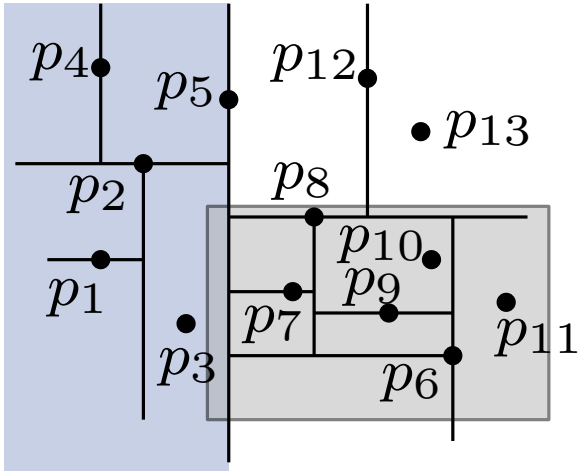
| ReportSubtree($\text{rc}(v)$)

else

if $\text{region}(\text{rc}(v)) \cap R \neq \emptyset$ **then**

| SearchKdTree($\text{rc}(v), R$)

Bereichsabfrage in einem kd -Tree

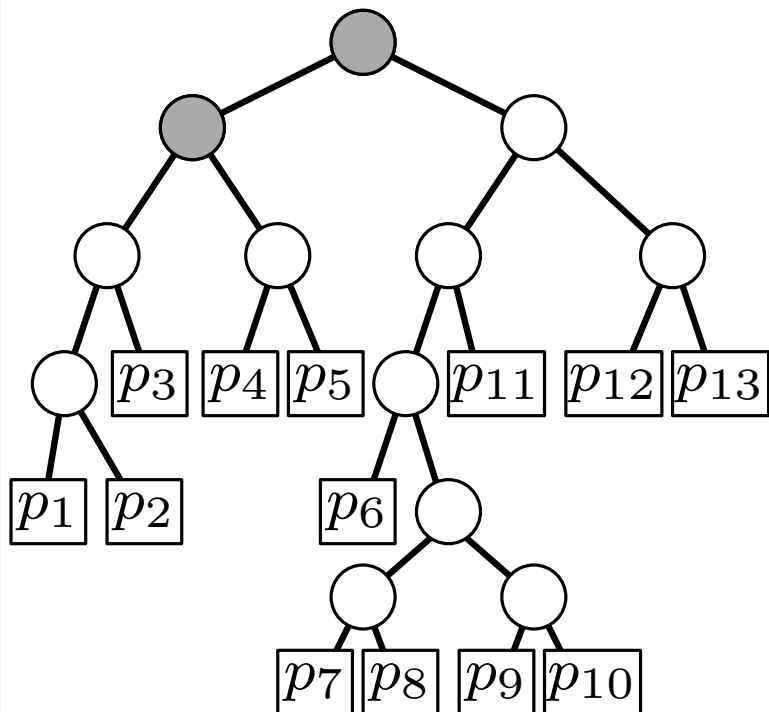
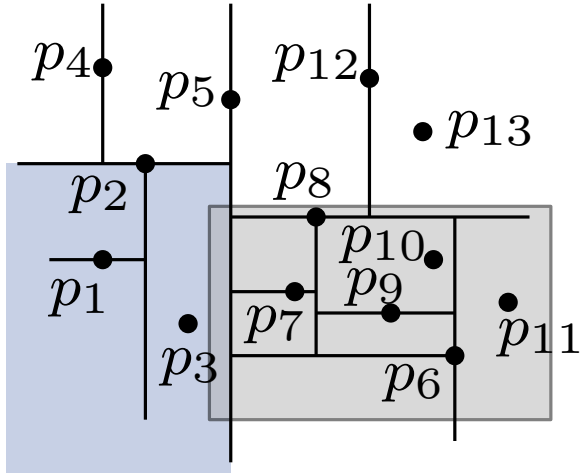


SearchKdTree(v, R)

```

if  $v$  Blatt then
    prüfe Punkt  $p$  in  $v$  auf  $p \in R$ 
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( $v$ ),  $R$ )
    
```

Bereichsabfrage in einem kd -Tree

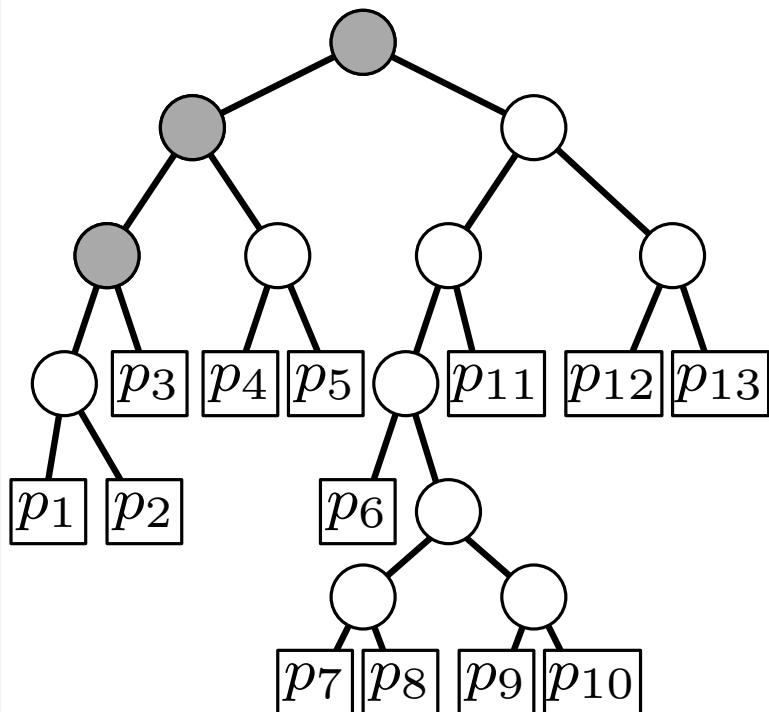
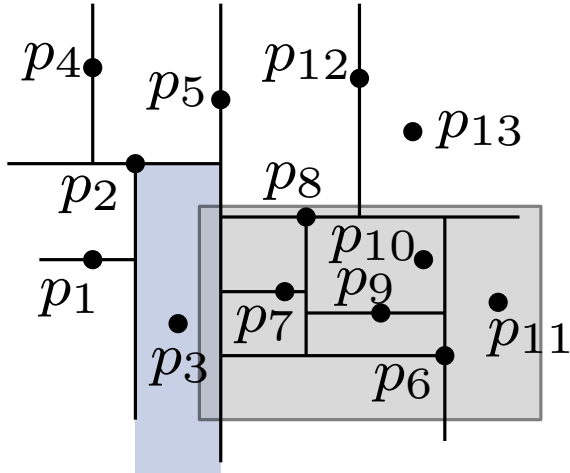


SearchKdTree(v, R)

```

if  $v$  Blatt then
    | prüfe Punkt  $p$  in  $v$  auf  $p \in R$ 
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( $v$ ),  $R$ )
    
```

Bereichsabfrage in einem kd -Tree

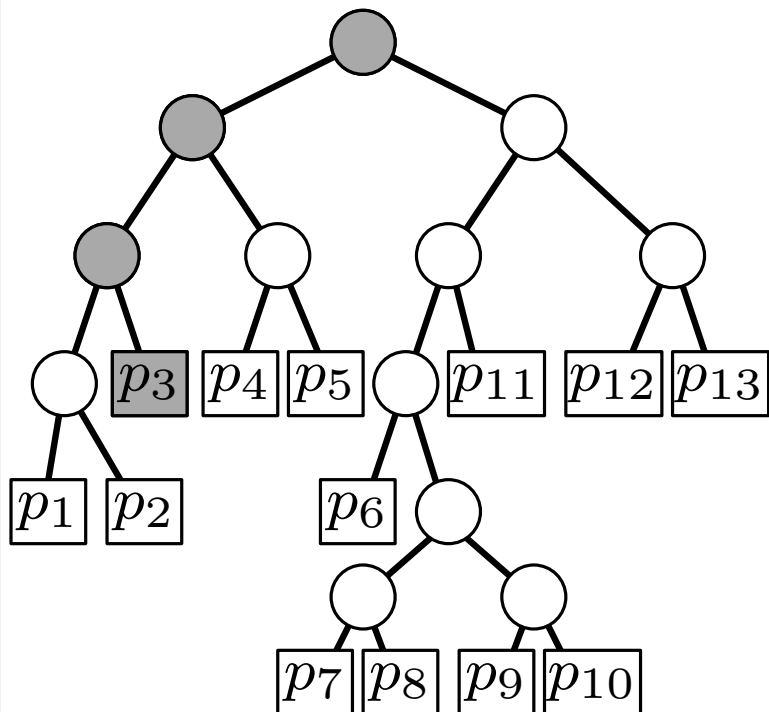
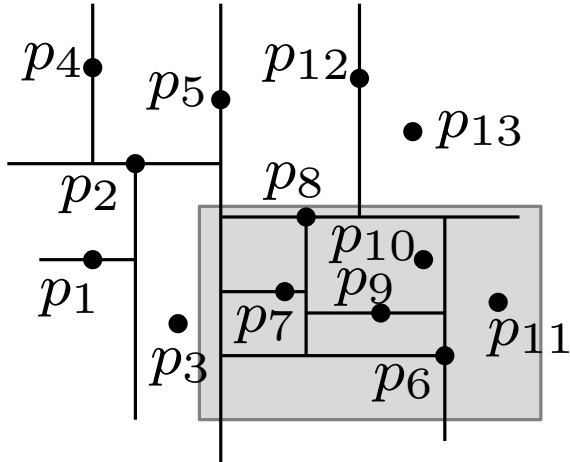


SearchKdTree(v, R)

```

if  $v$  Blatt then
    prüfe Punkt  $p$  in  $v$  auf  $p \in R$ 
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( $v$ ),  $R$ )
    
```

Bereichsabfrage in einem kd -Tree

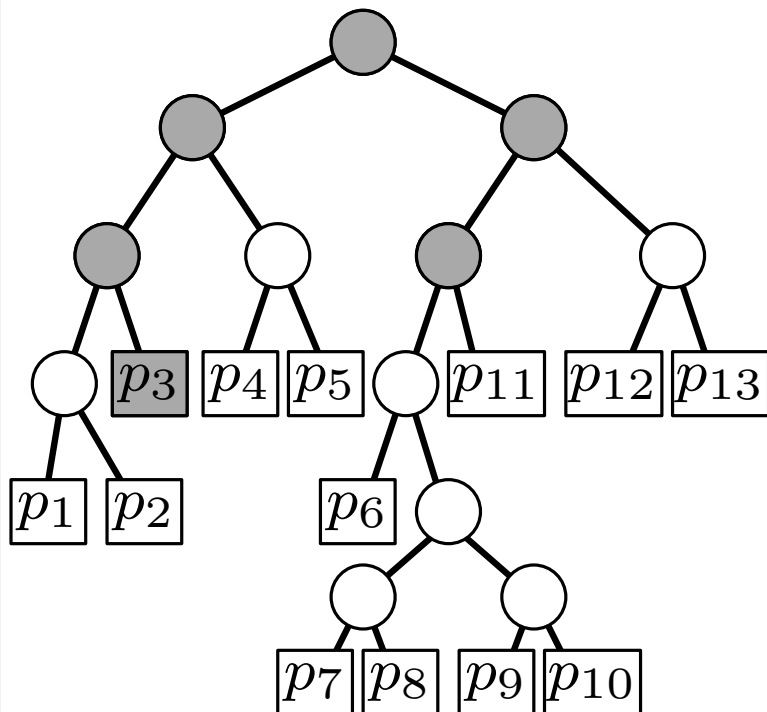
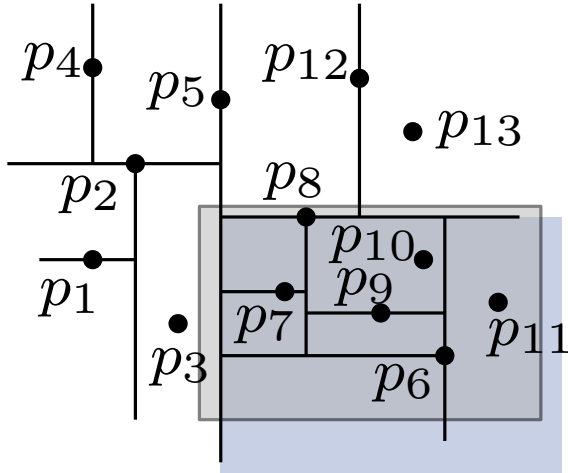


SearchKdTree(v, R)

```

if  $v$  Blatt then
    prüfe Punkt  $p$  in  $v$  auf  $p \in R$ 
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( $v$ ),  $R$ )
    
```


Bereichsabfrage in einem kd -Tree

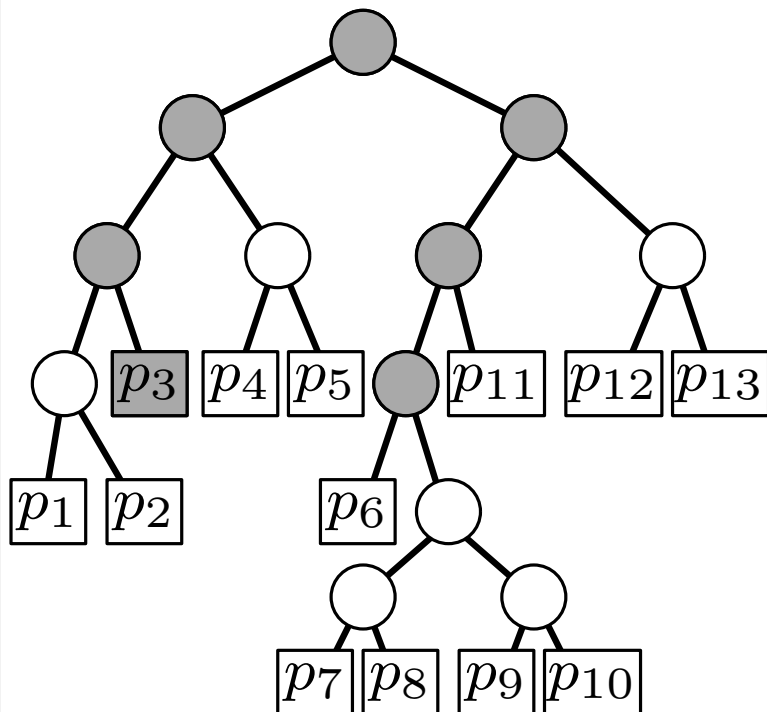
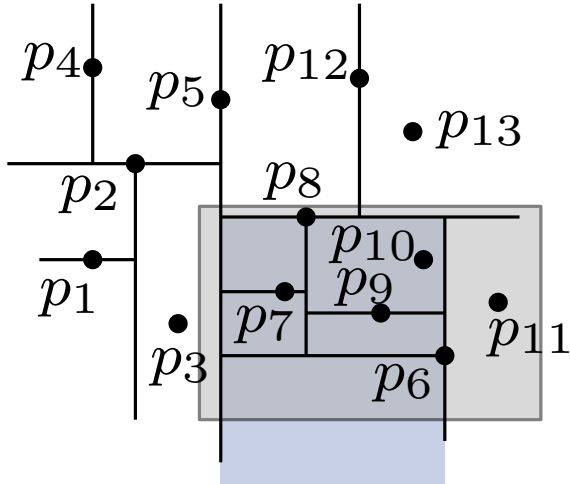


SearchKdTree(v, R)

```

if  $v$  Blatt then
    | prüfe Punkt  $p$  in  $v$  auf  $p \in R$ 
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( $v$ ),  $R$ )
    
```

Bereichsabfrage in einem kd -Tree

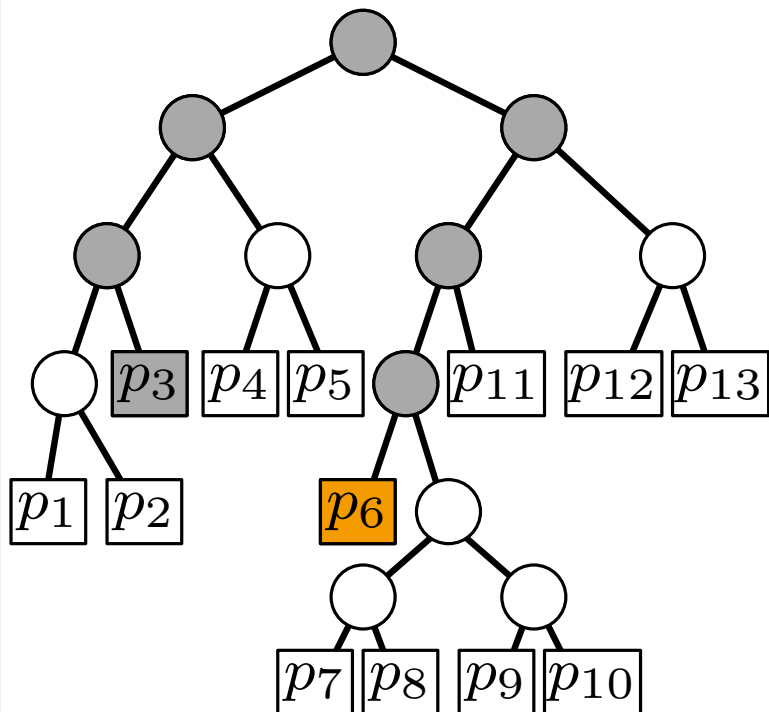
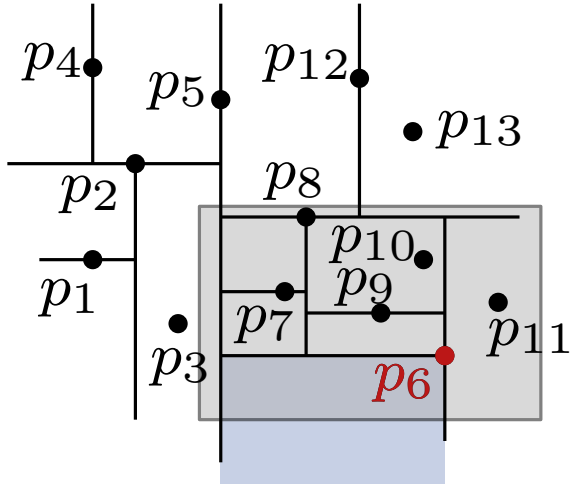


SearchKdTree(v, R)

```

if  $v$  Blatt then
    prüfe Punkt  $p$  in  $v$  auf  $p \in R$ 
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( $v$ ),  $R$ )
    
```

Bereichsabfrage in einem kd -Tree

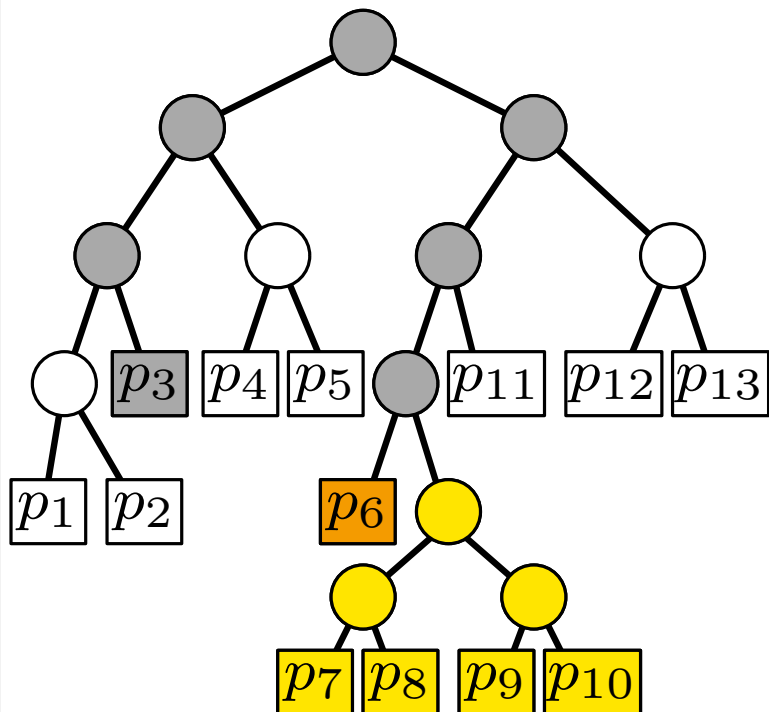
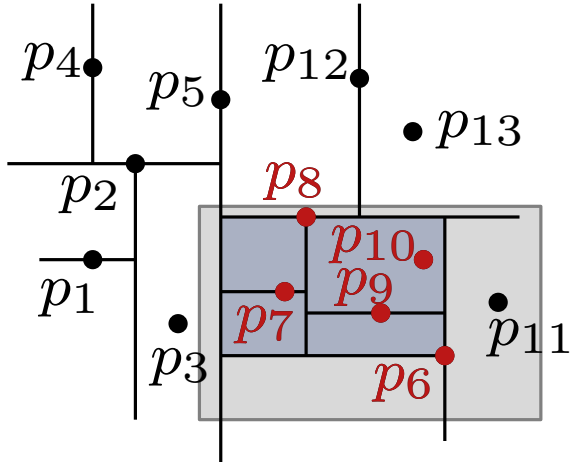


SearchKdTree(v, R)

```

if  $v$  Blatt then
    prüfe Punkt  $p$  in  $v$  auf  $p \in R$ 
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( $v$ ),  $R$ )
    
```

Bereichsabfrage in einem kd -Tree

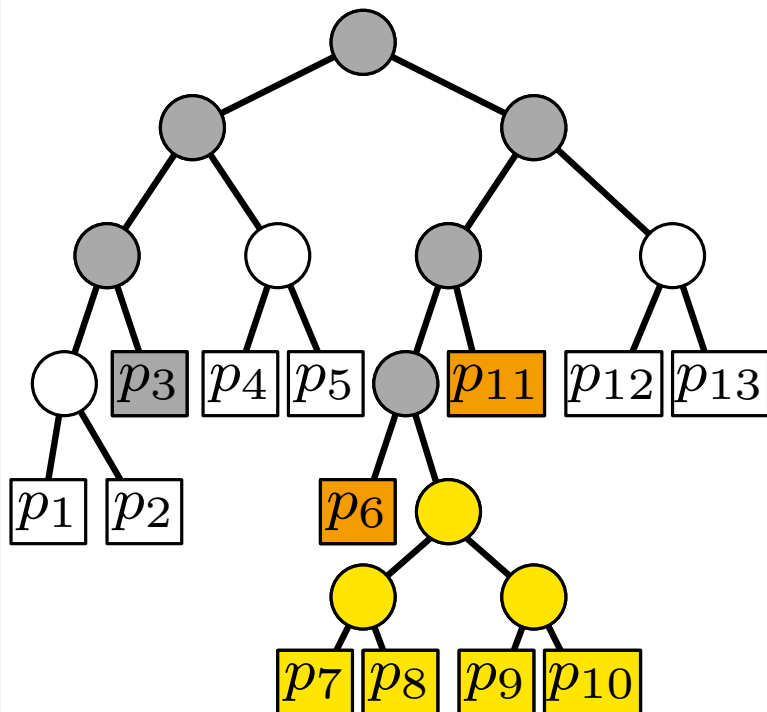
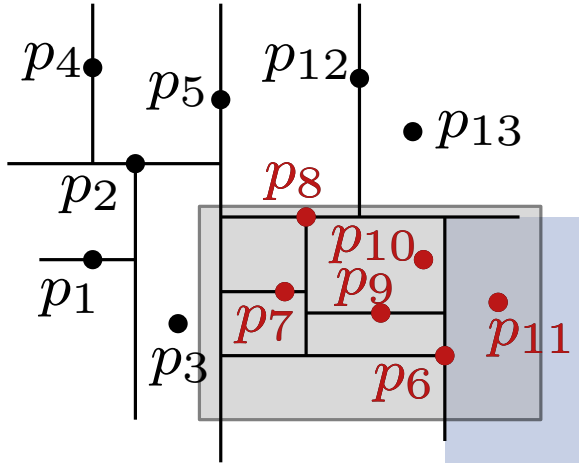


SearchKdTree(v, R)

```

if  $v$  Blatt then
    prüfe Punkt  $p$  in  $v$  auf  $p \in R$ 
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( $v$ ),  $R$ )
    
```

Bereichsabfrage in einem kd -Tree

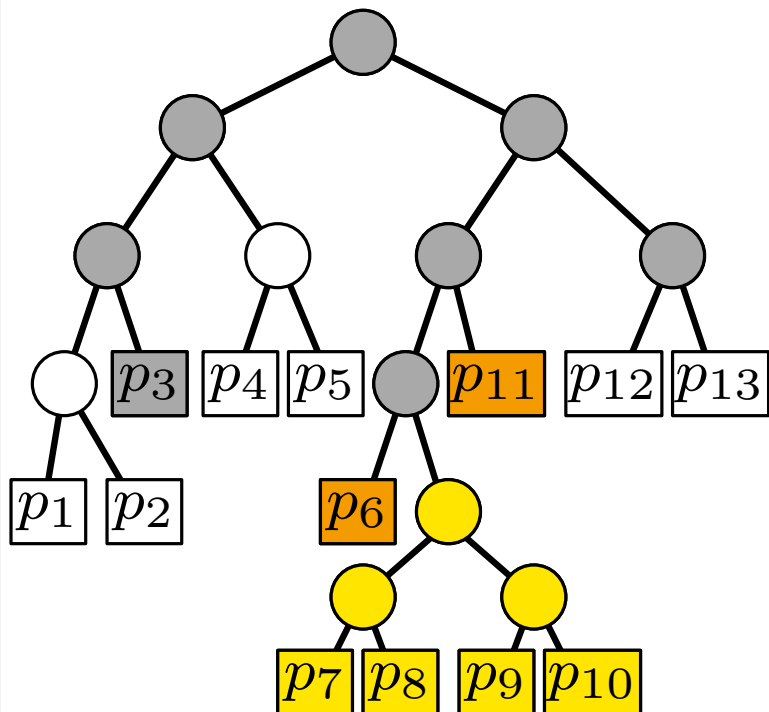
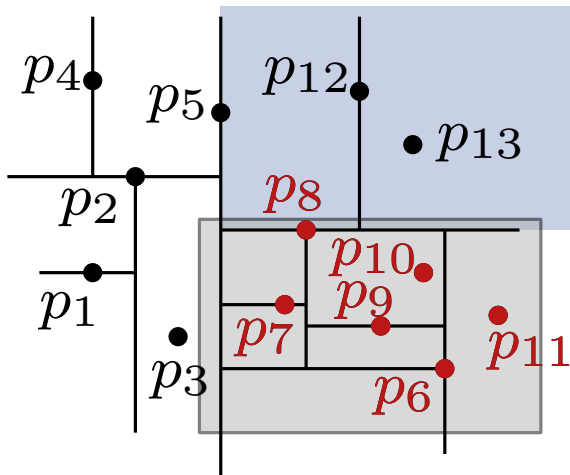


SearchKdTree(v, R)

```

if  $v$  Blatt then
    prüfe Punkt  $p$  in  $v$  auf  $p \in R$ 
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( $v$ ),  $R$ )
    
```

Bereichsabfrage in einem kd -Tree

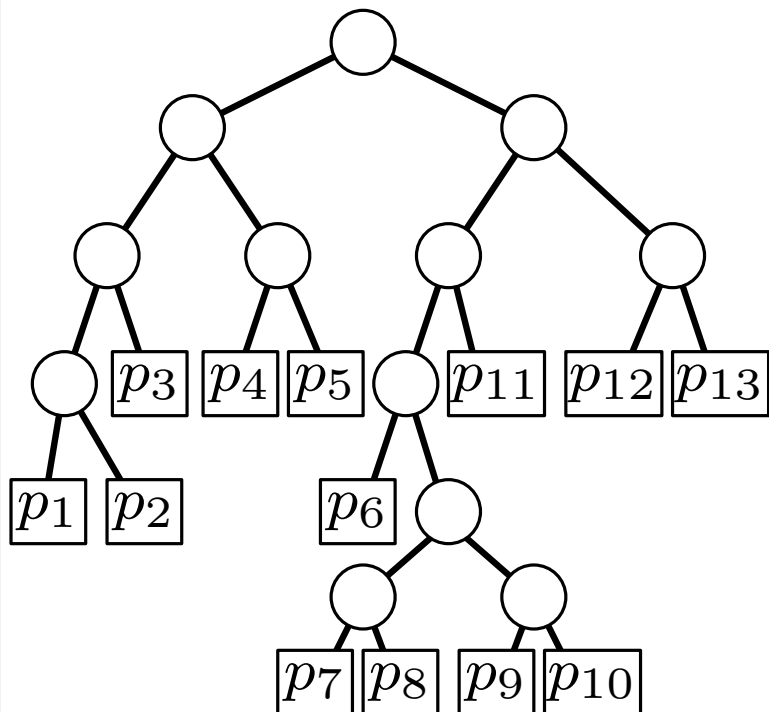
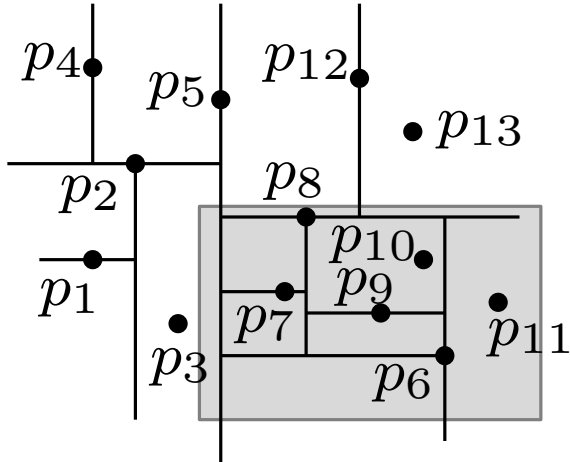


SearchKdTree(v, R)

```

if  $v$  Blatt then
    prüfe Punkt  $p$  in  $v$  auf  $p \in R$ 
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( $v$ ),  $R$ )
    
```


Bereichsabfrage in einem kd -Tree



SearchKdTree(v, R)

```

if  $v$  Blatt then
    prüfe Punkt  $p$  in  $v$  auf  $p \in R$ 
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( $v$ ),  $R$ )
    
```

Aufgabe 1

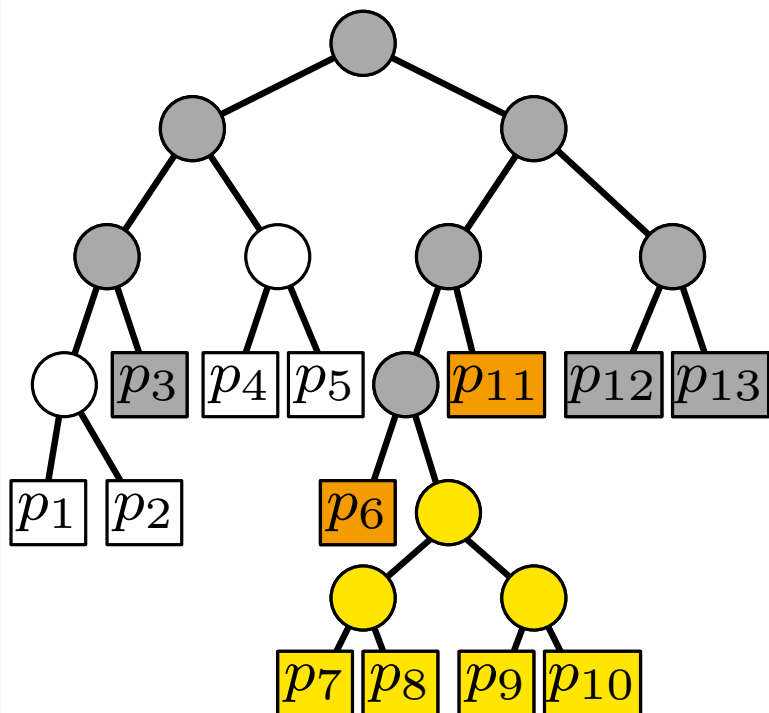
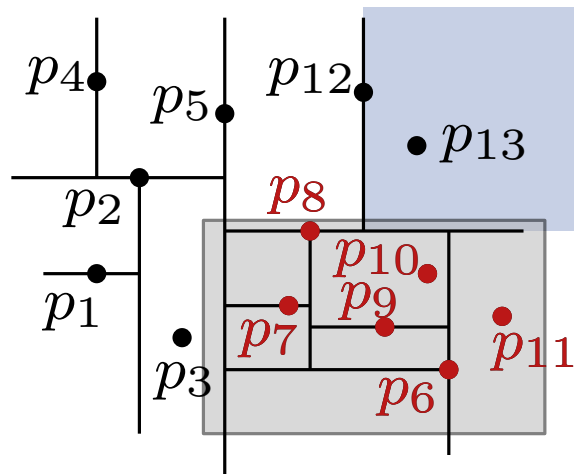
Behauptung: Anfrage liegt in $O(\sqrt{n} + k)$

$Q(n)$ = Anzahl der überprüften Regionen

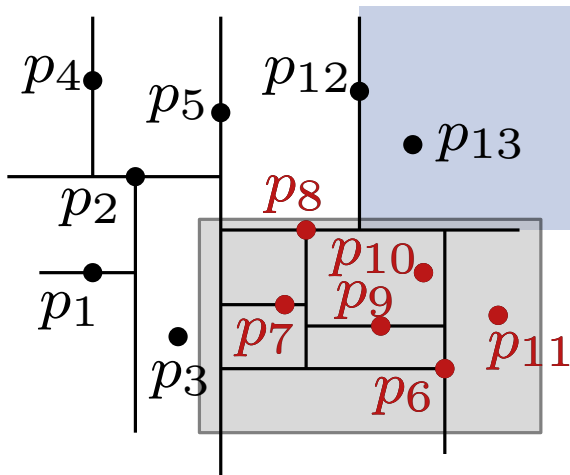
a) Schätze Anzahl Rekursionsaufrufe ab.

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

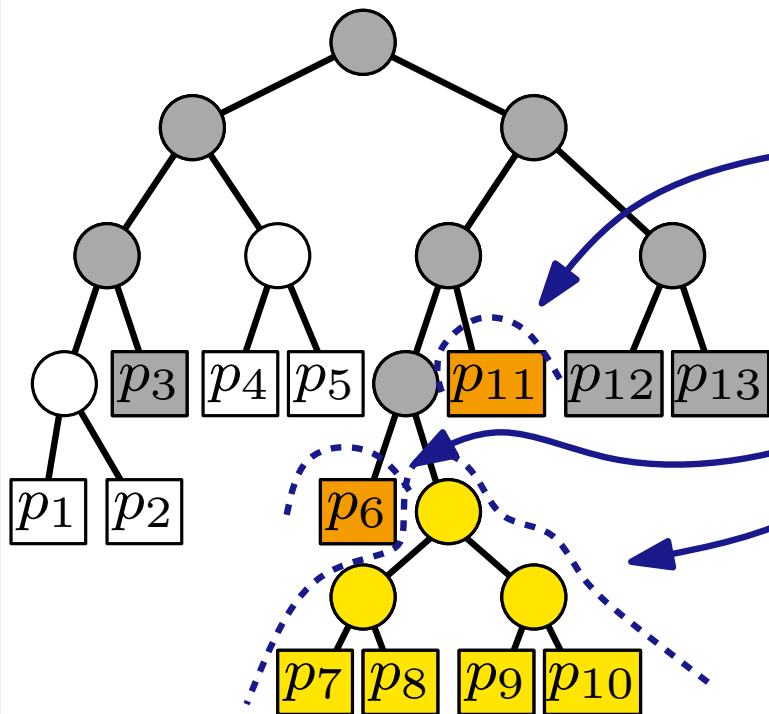
Bereichsabfrage in einem kd -Tree



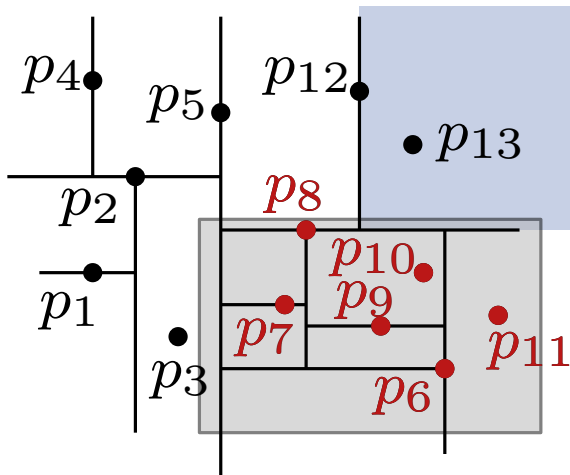
Bereichsabfrage in einem kd -Tree



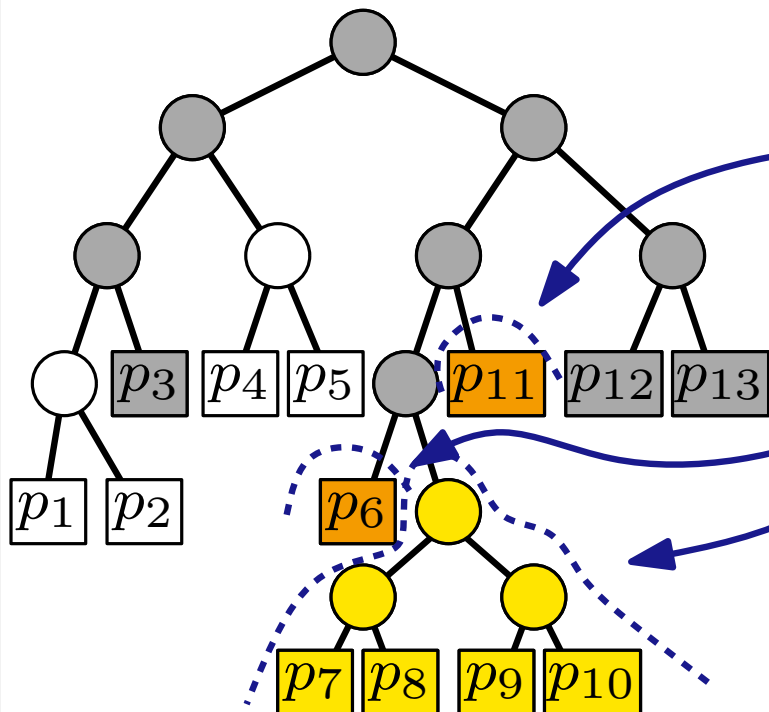
ReportSubtree in $\mathcal{O}(k)$



Bereichsabfrage in einem kd -Tree

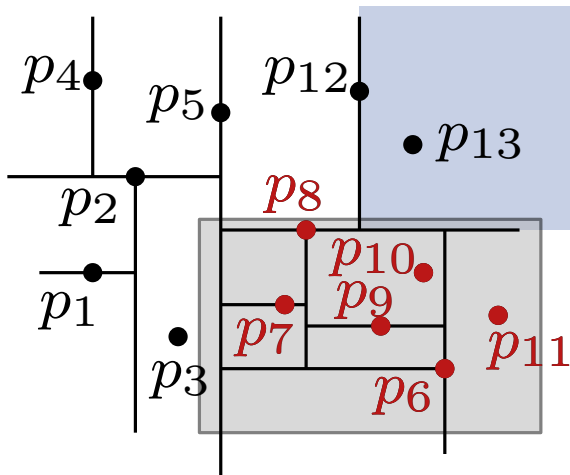


ReportSubtree in $\mathcal{O}(k)$

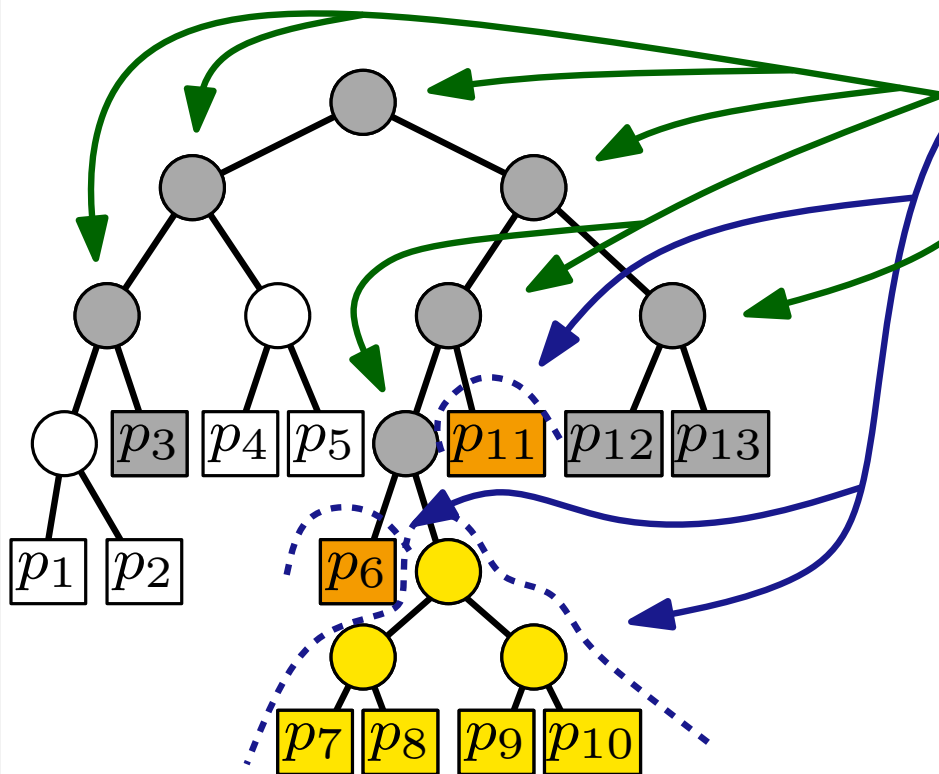


Wie viele Knoten betrachten wir?
Jeder Knoten entspricht einem
Rekursionsaufruf.

Bereichsabfrage in einem kd -Tree

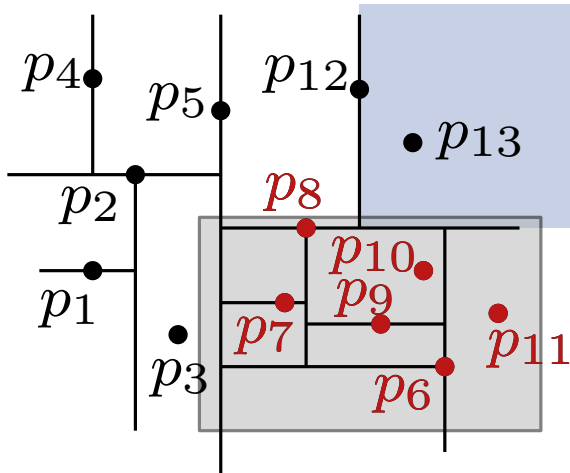


ReportSubtree in $\mathcal{O}(k)$

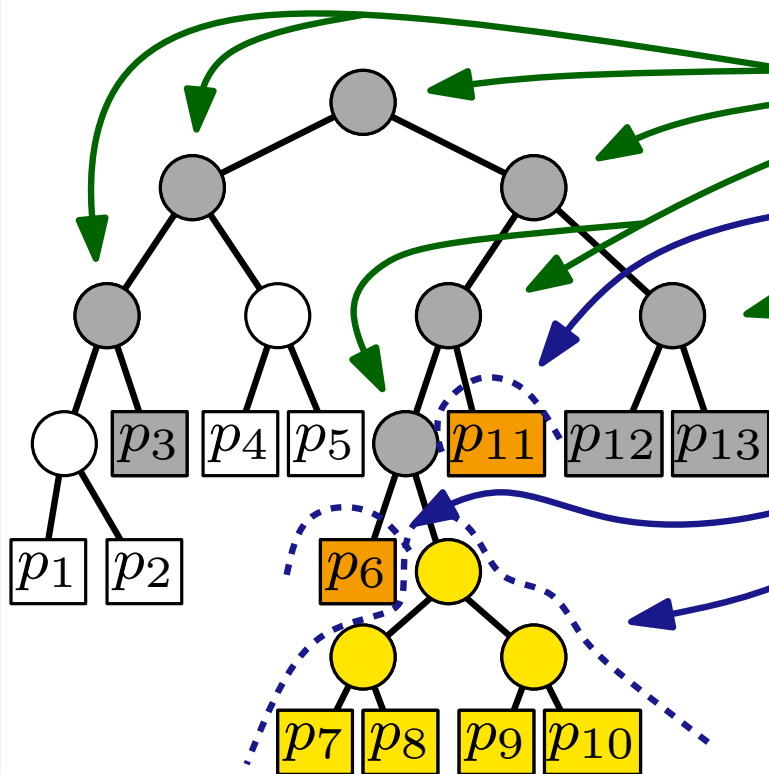


Wie viele Knoten betrachten wir?
Jeder Knoten entspricht einem
Rekursionsaufruf.

Bereichsabfrage in einem kd -Tree



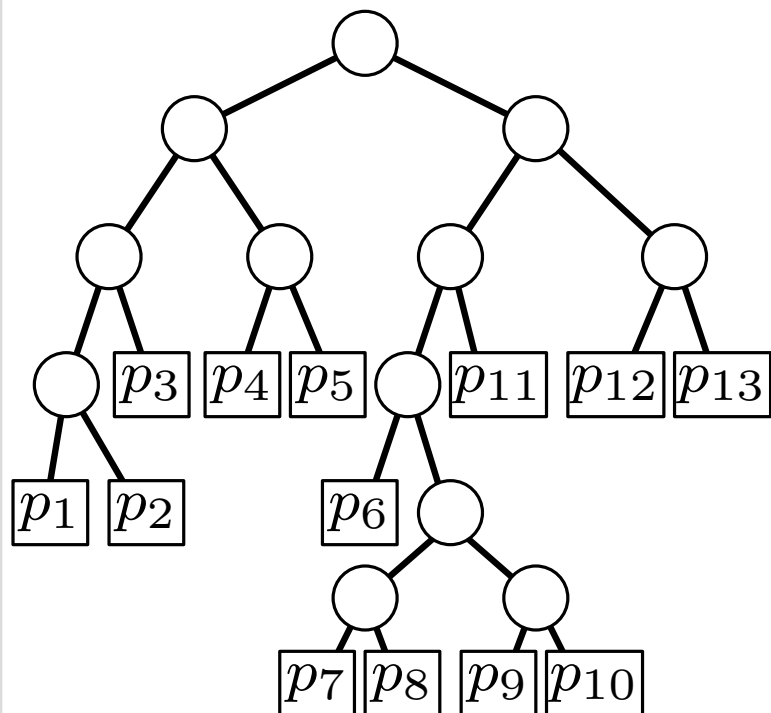
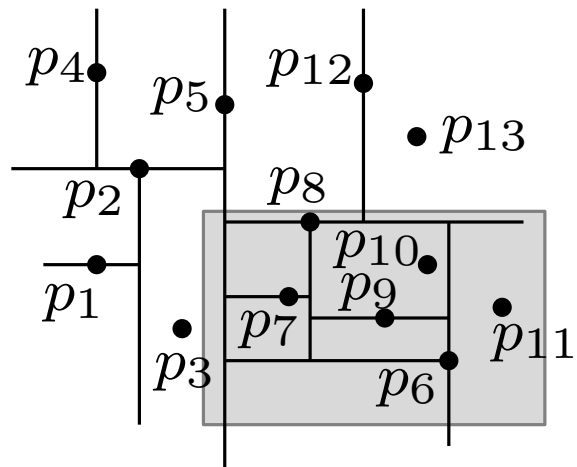
ReportSubtree in $\mathcal{O}(k)$



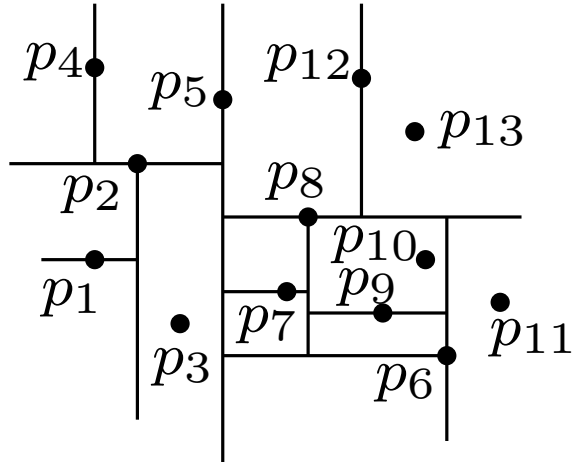
Wie viele Knoten betrachten wir?
Jeder Knoten entspricht einem
Rekursionsaufruf.

jeder Knoten \triangleq geschnittene Region

Bereichsabfrage in einem kd -Tree

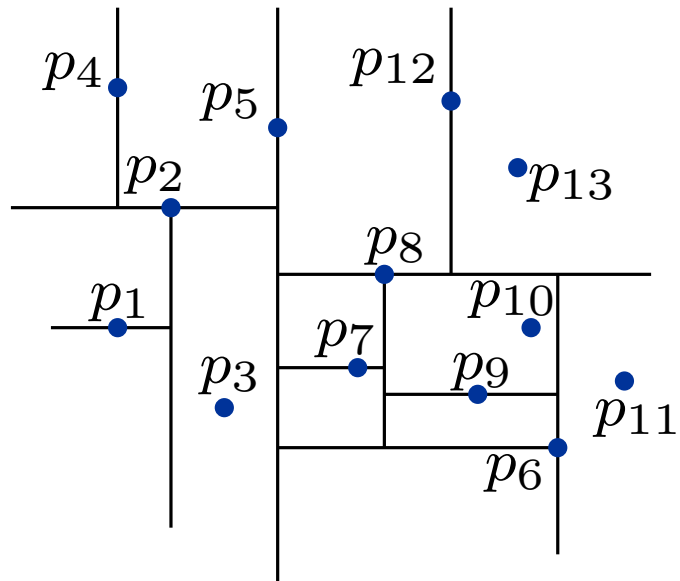


Bereichsabfrage in einem kd -Tree

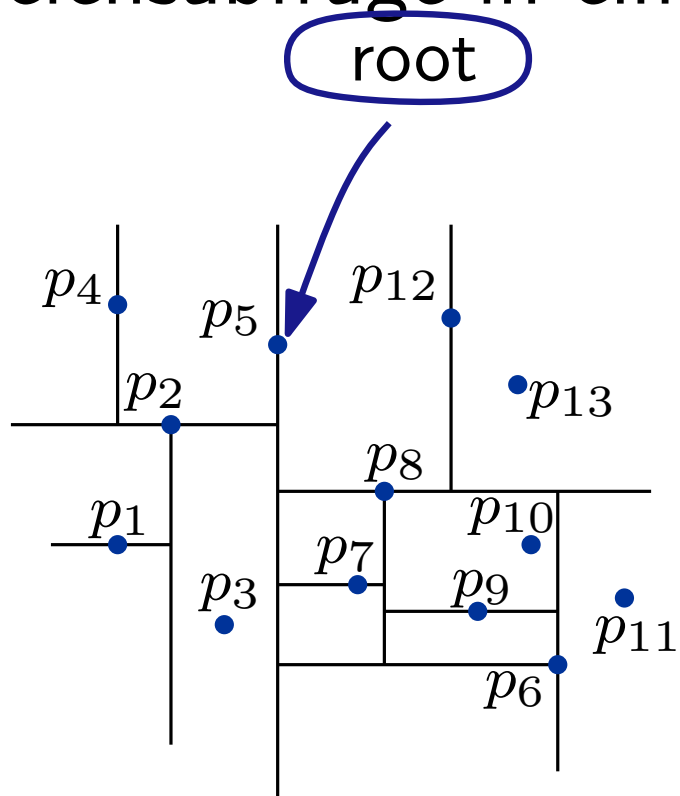


Wie viele Regionen schneidet eine vertikale Linie?

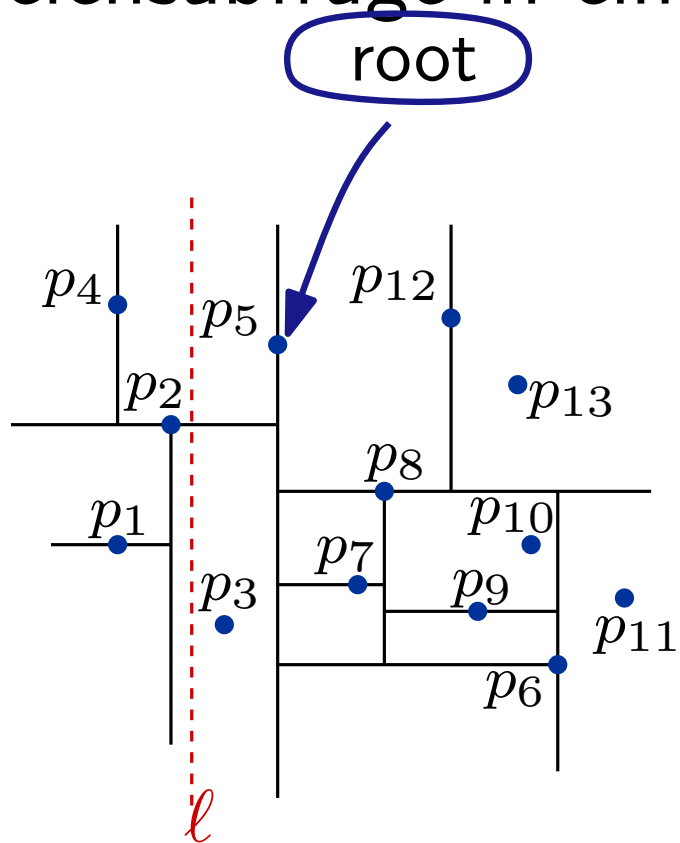
Bereichsabfrage in einem kd -Tree



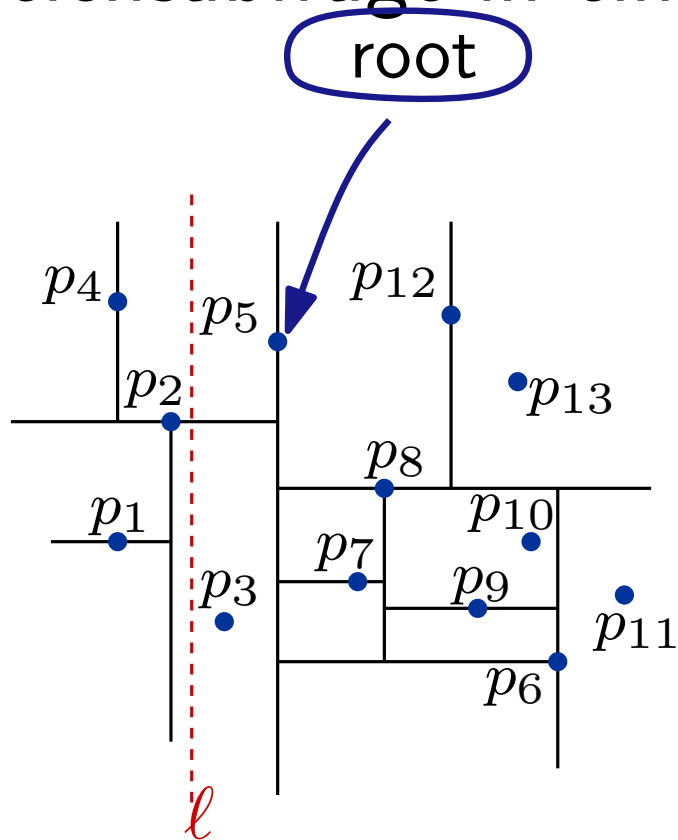
Bereichsabfrage in einem kd -Tree



Bereichsabfrage in einem kd -Tree



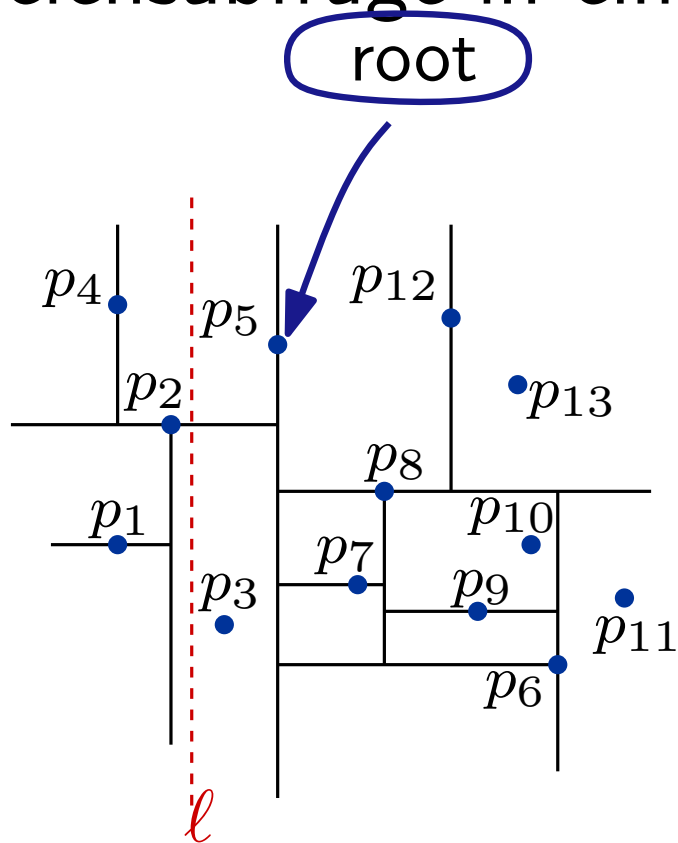
Bereichsabfrage in einem kd -Tree



Vermutung:

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

Bereichsabfrage in einem kd -Tree



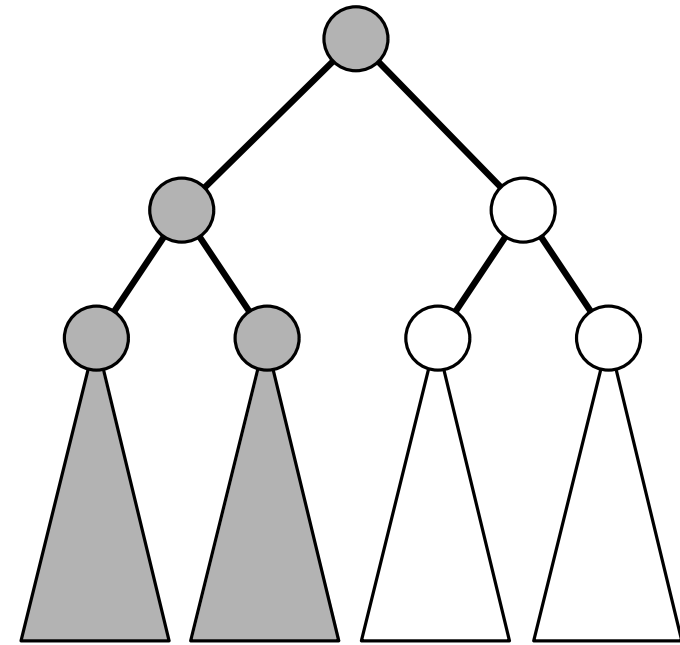
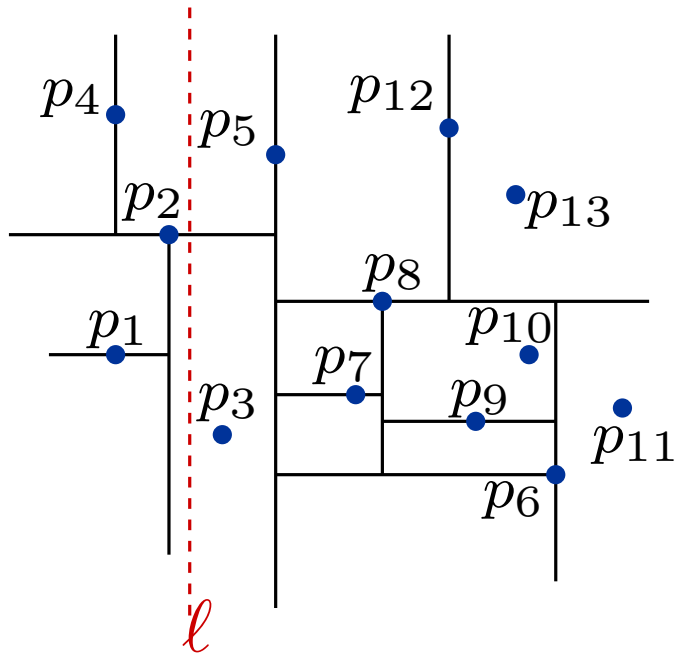
Vermutung:

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

Problem?

l schneidet beide Kinder des linken Kindes von root

Bereichsabfrage in einem kd -Tree



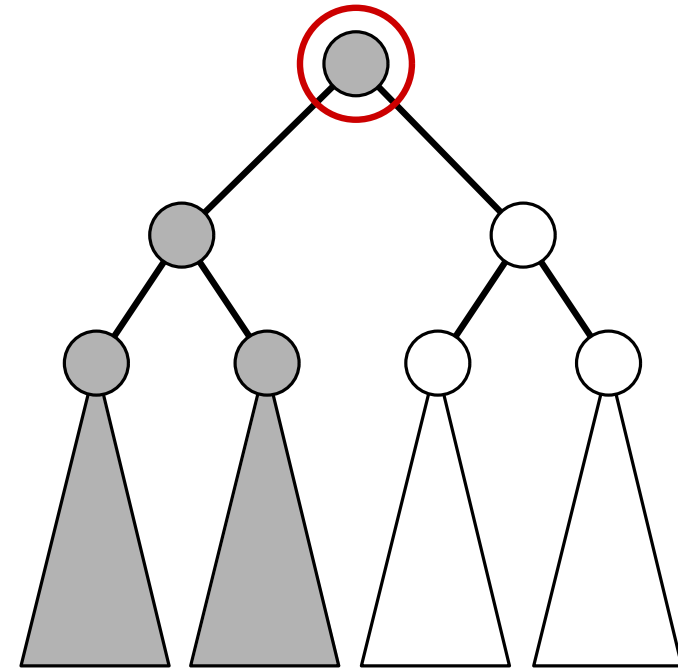
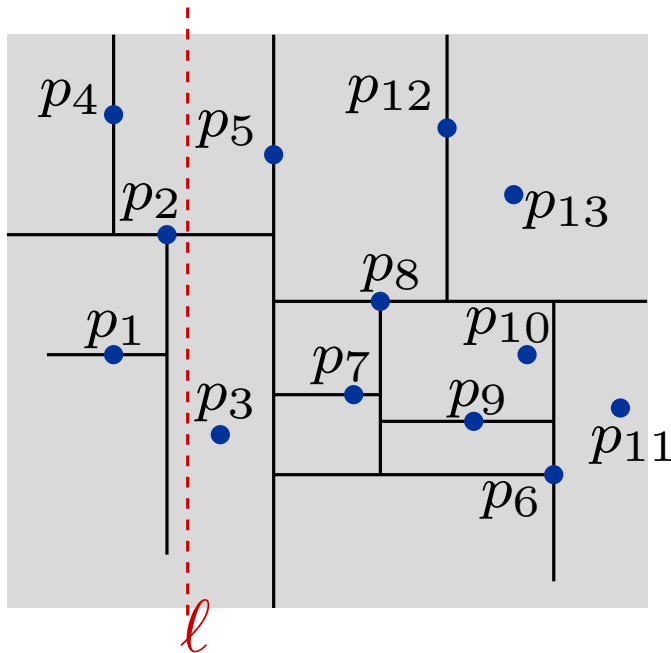
Vermutung:

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

Problem?

l schneidet beide Kinder des linken Kindes von root

Bereichsabfrage in einem kd -Tree



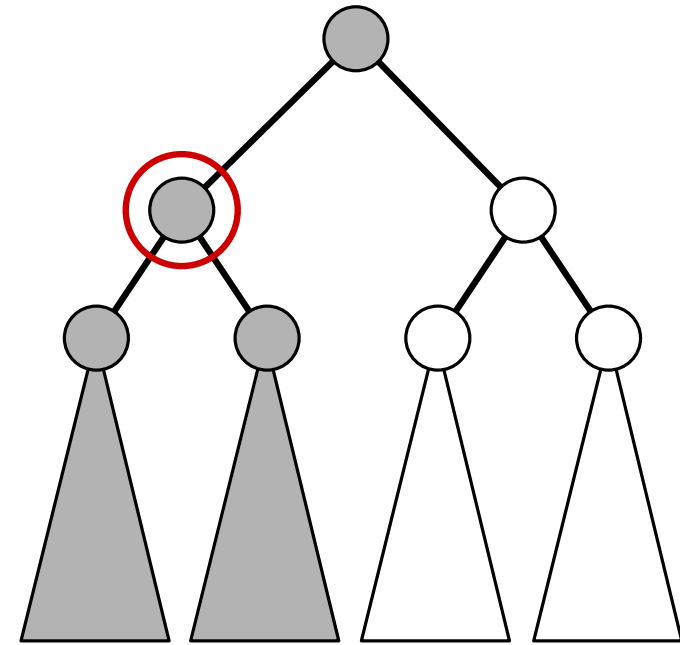
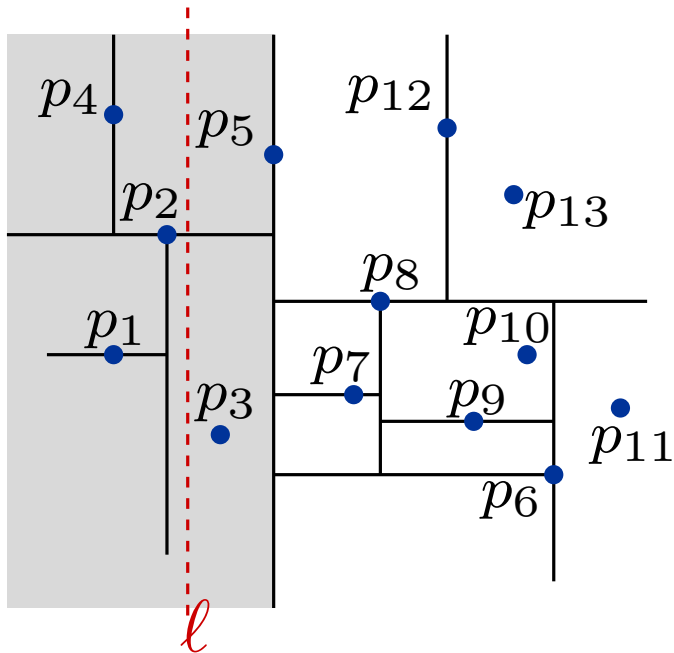
Vermutung:

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

Problem?

ℓ schneidet beide Kinder des linken Kindes von root

Bereichsabfrage in einem kd -Tree



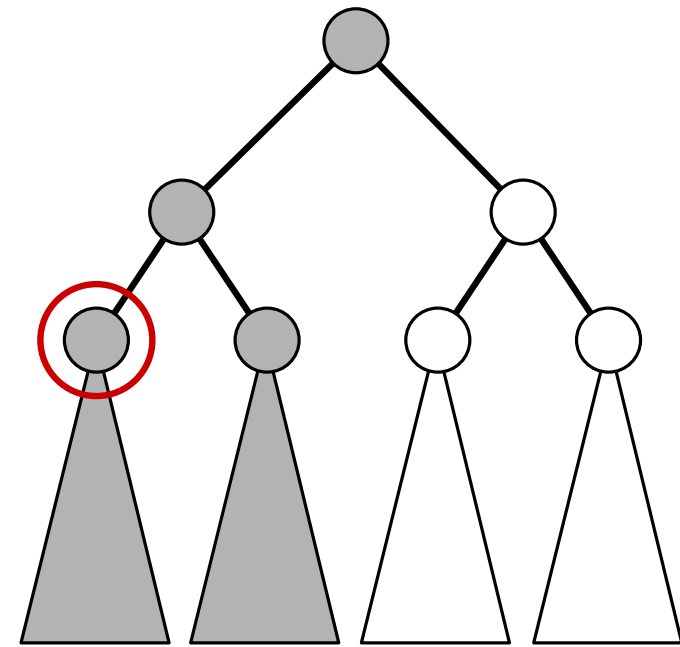
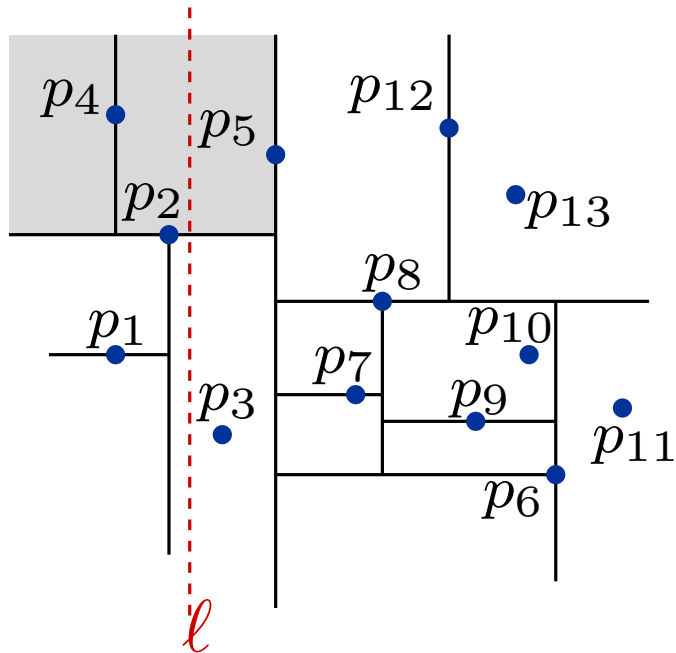
Vermutung:

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

Problem?

ℓ schneidet beide Kinder des linken Kindes von root

Bereichsabfrage in einem kd -Tree



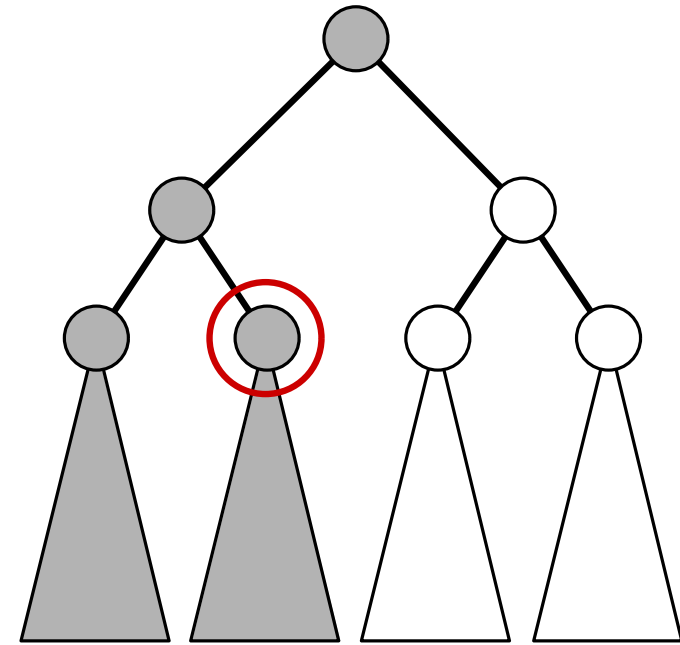
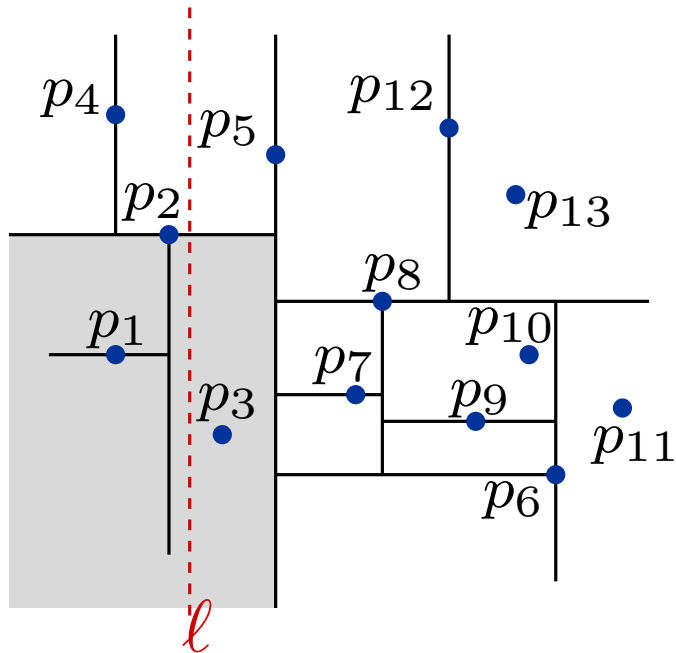
Vermutung:

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

Problem?

l schneidet beide Kinder des linken Kindes von root

Bereichsabfrage in einem kd -Tree



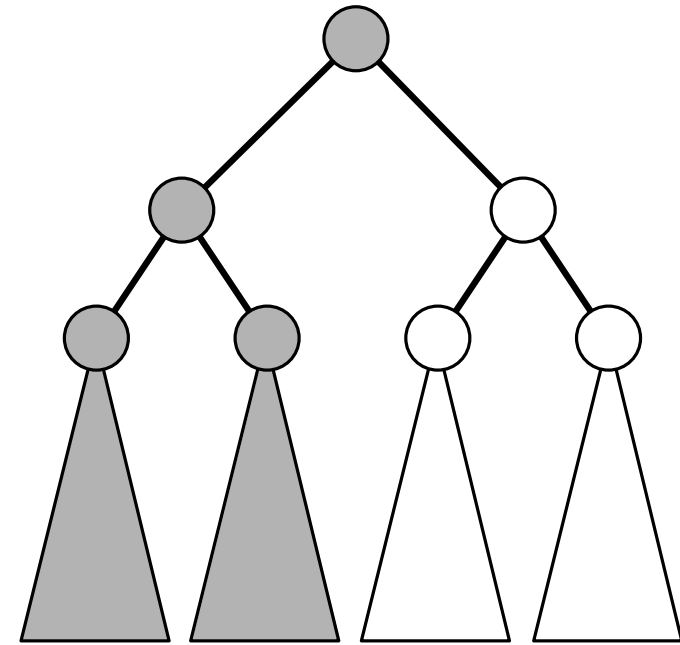
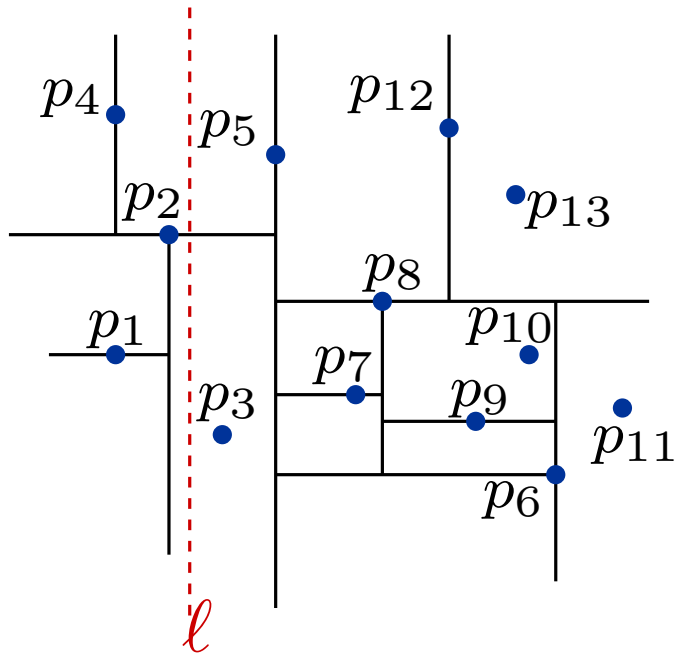
Vermutung:

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

Problem?

l schneidet beide Kinder des linken Kindes von root

Bereichsabfrage in einem kd -Tree



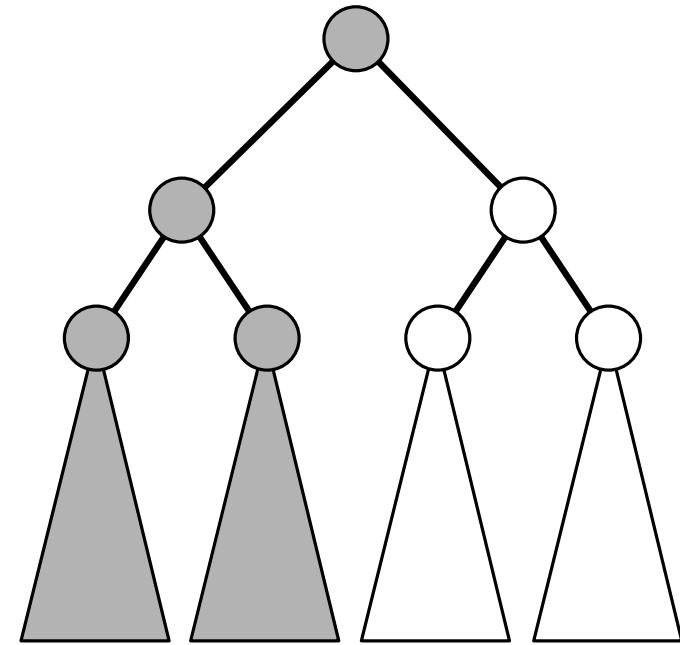
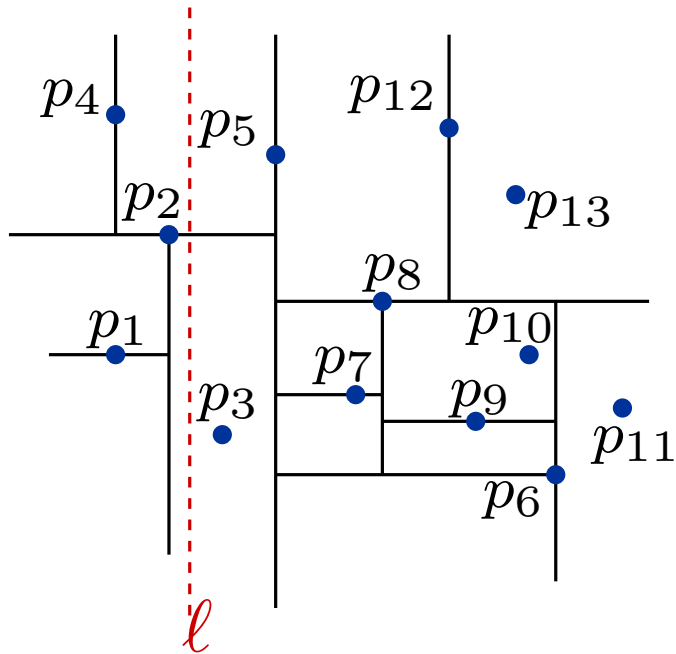
Vermutung:

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

Problem?

l schneidet beide Kinder des linken Kindes von root

Bereichsabfrage in einem kd -Tree



Vermutung:

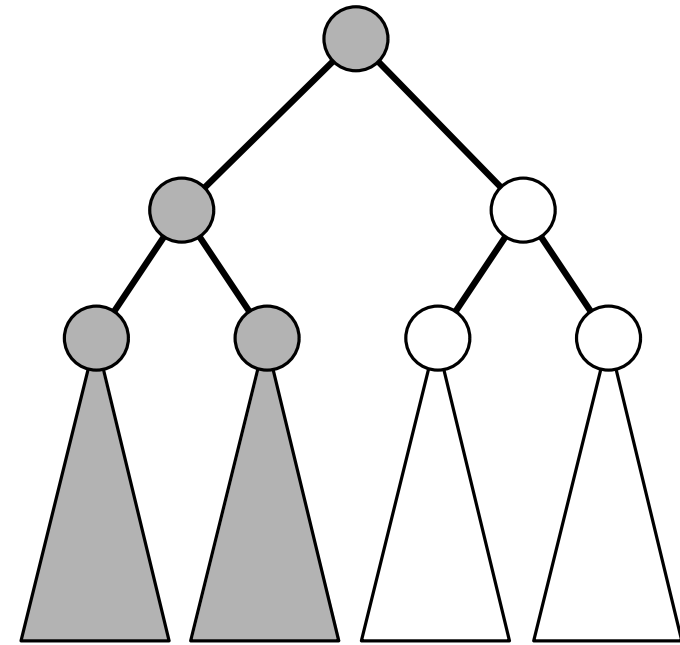
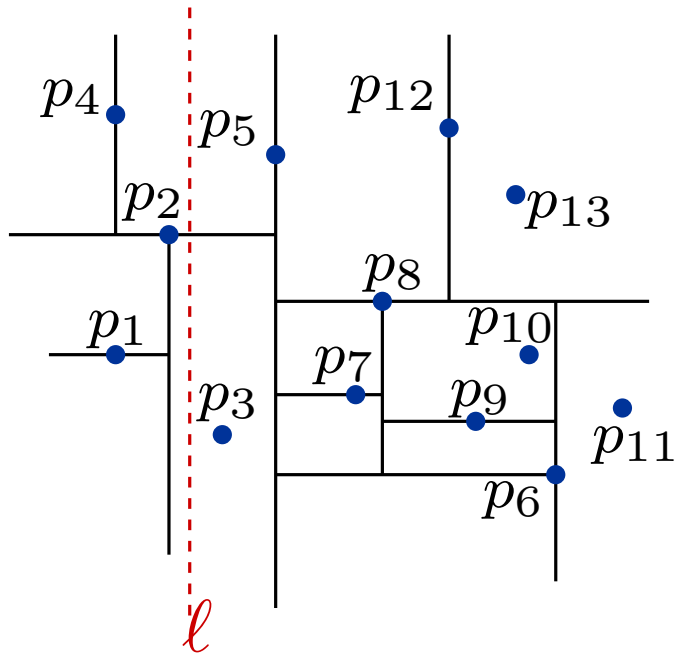
$$Q(n) = 1 + Q(n/2)$$

Problem?

l schneidet beide Kinder des linken Kindes von root

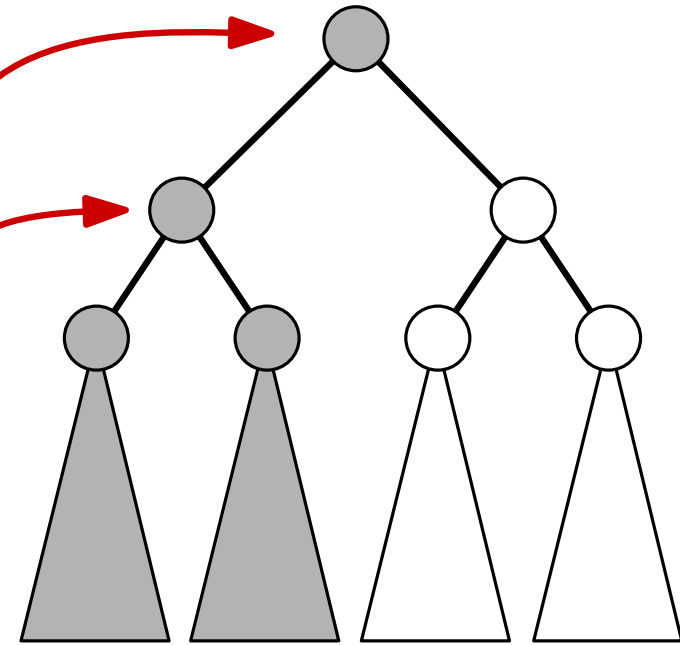
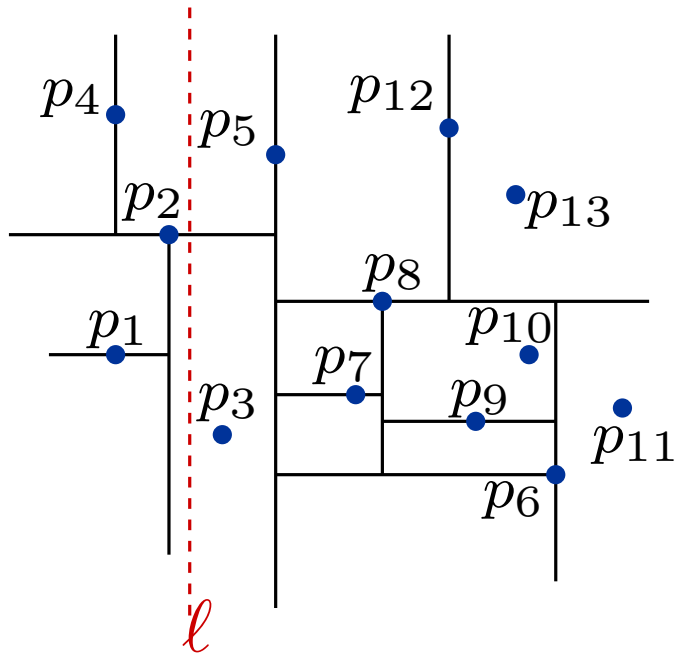
Gleiche Rekursionsituation

Bereichsabfrage in einem kd -Tree



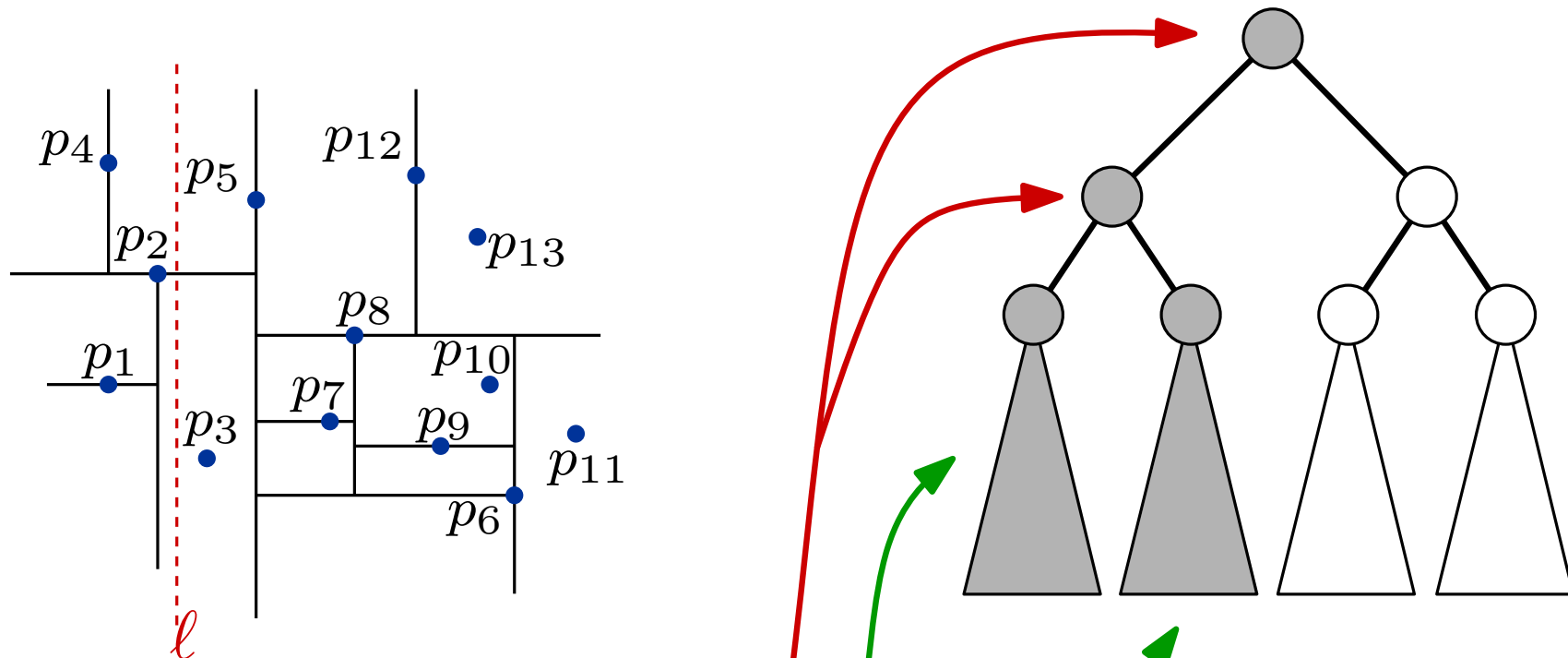
$$Q(n) = \begin{cases} \mathcal{O}(1) & , \text{ für } n = 1 \\ 2 + 2Q(n/4) & , \text{ für } n > 1 \end{cases}$$

Bereichsabfrage in einem kd -Tree



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

Bereichsabfrage in einem kd -Tree



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

Aufgabe 1

Behauptung: Anfrage liegt in $O(\sqrt{n} + k)$

$Q(n)$ = Anzahl der überprüften Regionen

a) Zeige:

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

Aufgabe 1

Behauptung: Anfrage liegt in $O(\sqrt{n} + k)$

$Q(n)$ = Anzahl der überprüften Regionen

a) Zeige:

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

b) Rekurrenz auflösen: $Q(n) = \mathcal{O}(\sqrt{n})$.

Aufgabe 1

Behauptung: Anfrage liegt in $O(\sqrt{n} + k)$

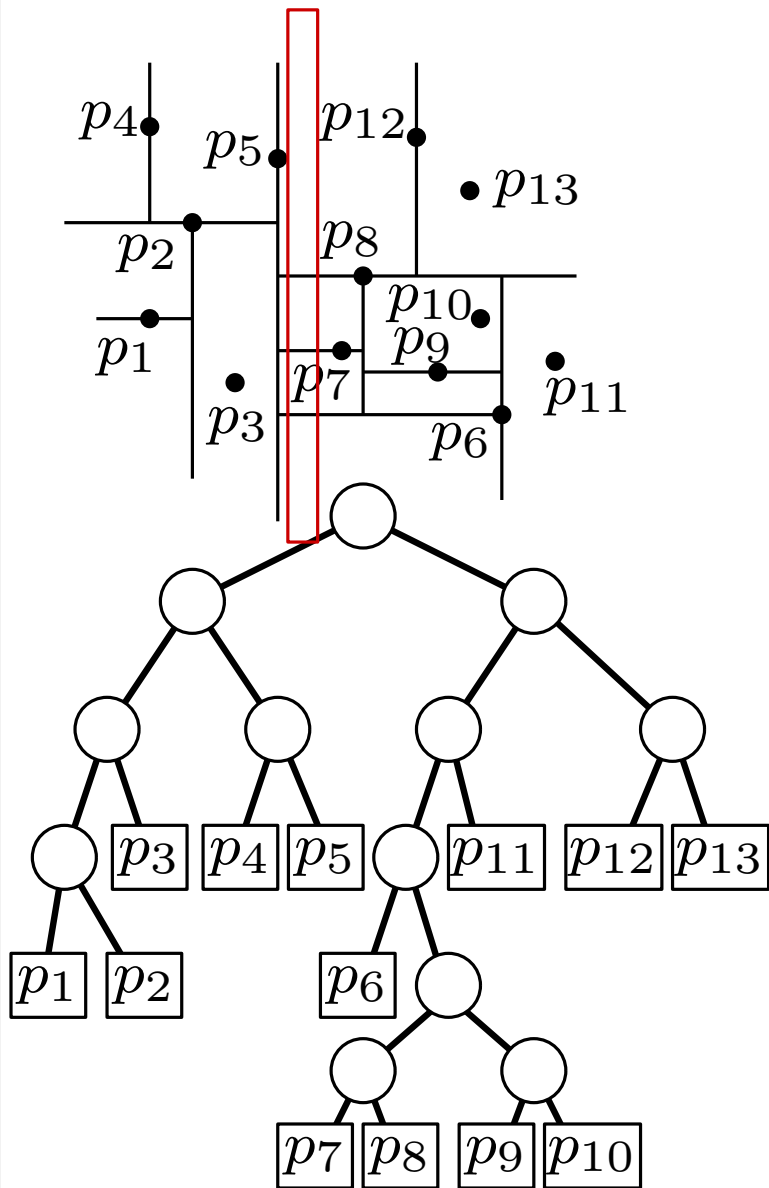
$Q(n)$ = Anzahl der überprüften Regionen

a) Zeige:

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

b) Rekurrenz auflösen: $Q(n) = \mathcal{O}(\sqrt{n})$.

c) $\Omega(\sqrt{n})$ untere Schranke für Bereichsanfragen in kd -Trees



SearchKdTree(v, R)

```

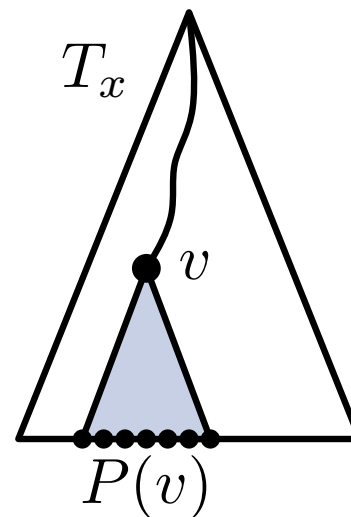
if  $v$  Blatt then
  | prüfe Punkt  $p$  in  $v$  auf  $p \in R$ 
else
  | if region(lc( $v$ ))  $\subseteq R$  then
  | | ReportSubtree(lc( $v$ ))
  | else
  | | if region(lc( $v$ ))  $\cap R \neq \emptyset$  then
  | | | SearchKdTree(lc( $v$ ),  $R$ )
  | if region(rc( $v$ ))  $\subseteq R$  then
  | | ReportSubtree(rc( $v$ ))
  | else
  | | if region(rc( $v$ ))  $\cap R \neq \emptyset$  then
  | | | SearchKdTree(rc( $v$ ),  $R$ )
  
```

c) $\Omega(\sqrt{n})$ untere Schranke für Bereichsanfragen in kd -Trees

Range Trees

Idee: Nutze eindimensionale binäre Suchbäume auf zwei Ebenen:

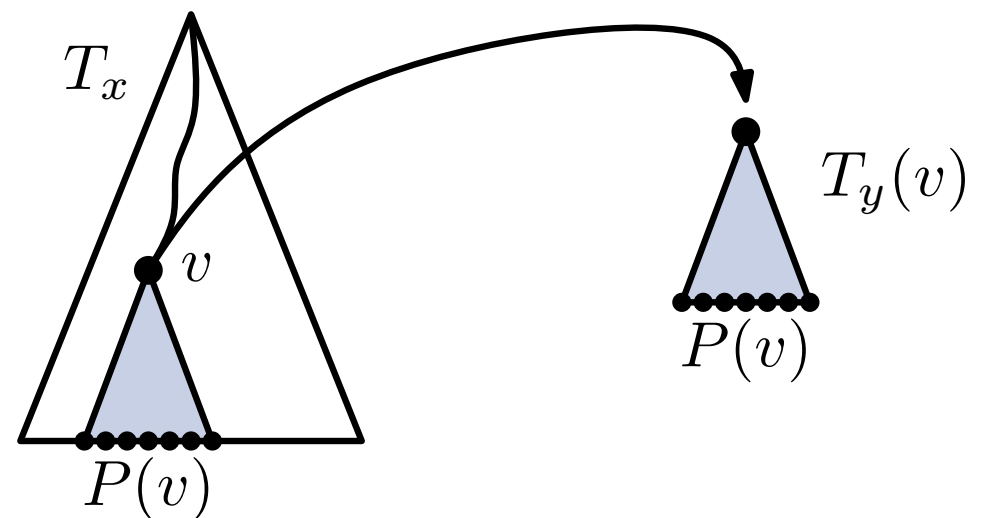
- ein 1d Suchbaum T_x bzgl. x -Koordinaten



Range Trees

Idee: Nutze eindimensionale binäre Suchbäume auf zwei Ebenen:

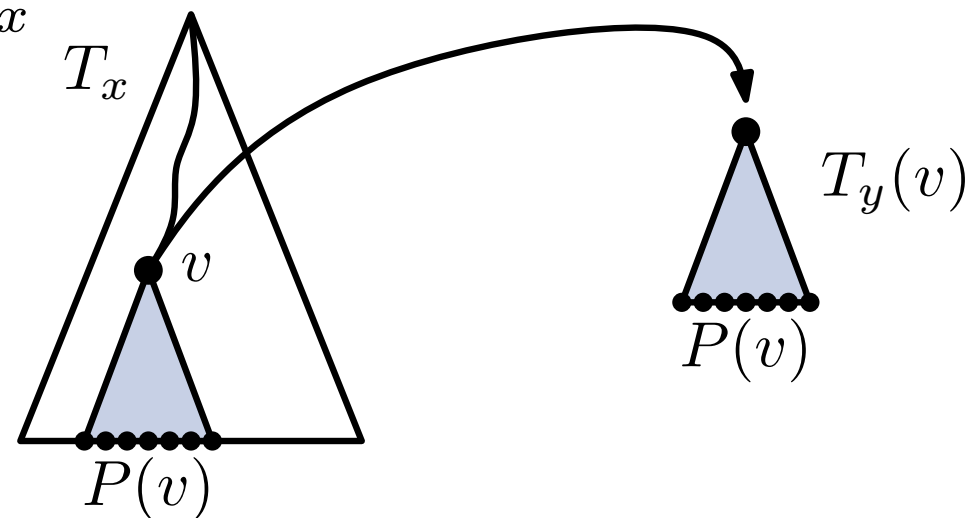
- ein 1d Suchbaum T_x bzgl. x -Koordinaten
- in jedem Knoten v von T_x einen 1d Suchbaum $T_y(v)$ zum Speichern der kanonischen Blattmenge $P(v)$ bzgl. y -Koordinaten



Range Trees

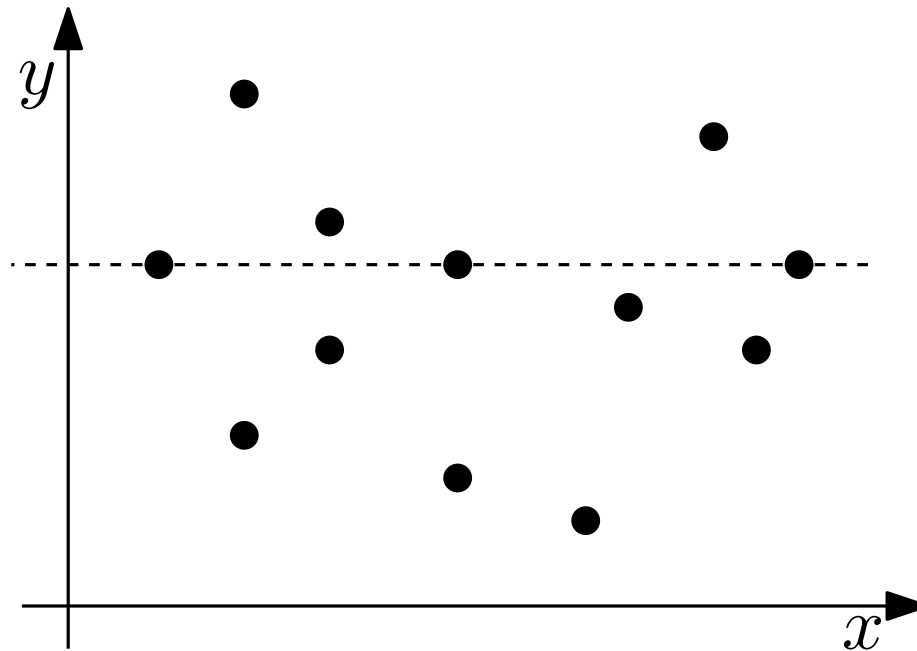
Idee: Nutze eindimensionale binäre Suchbäume auf zwei Ebenen:

- ein 1d Suchbaum T_x bzgl. x -Koordinaten
- in jedem Knoten v von T_x einen 1d Suchbaum $T_y(v)$ zum Speichern der kanonischen Blattmenge $P(v)$ bzgl. y -Koordinaten
- bestimme Lösungsmenge durch x -Abfrage in T_x und anschließender y -Abfrage in den Hilfsstrukturen T_y der Teilbäume in T_x



Aufgabe 2

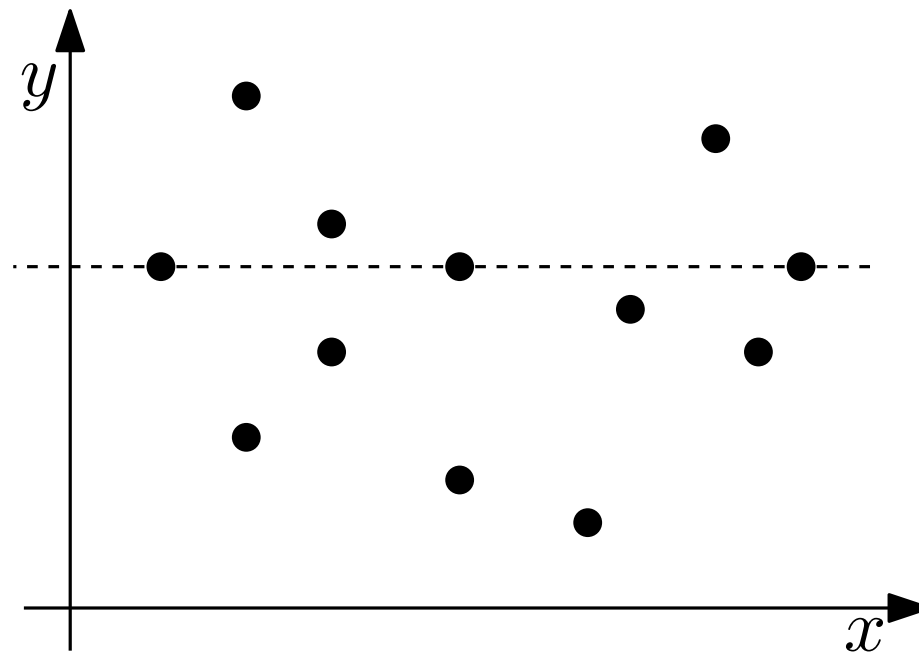
'partial match queries'



Gebe mir alle Punkte mit $y = 7$.

Aufgabe 2

'partial match queries'

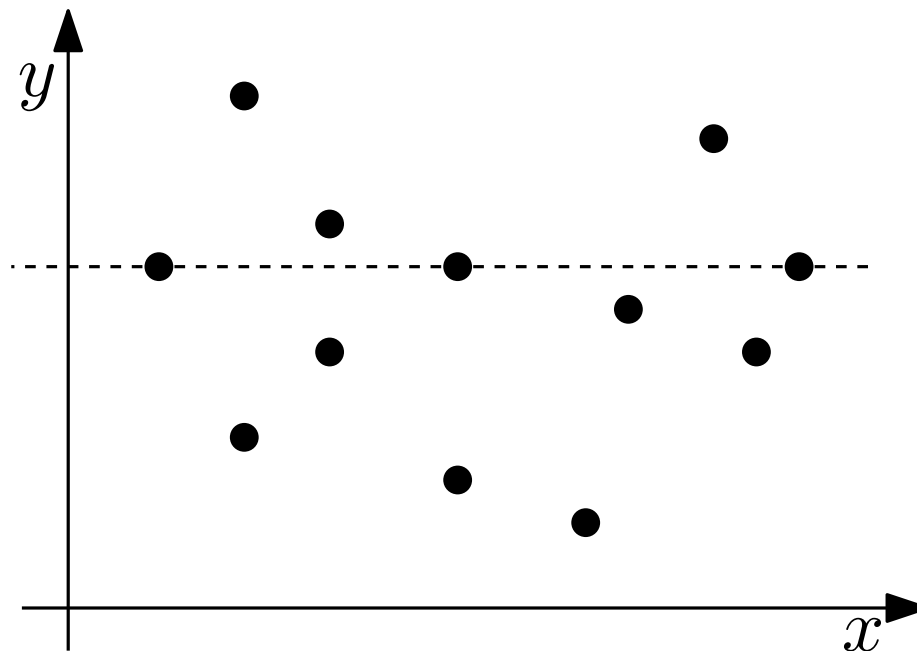


Gebe mir alle Punkte mit $y = 7$.

a) Wie kann man *kd*-Trees dafür nutzen?

Aufgabe 2

'partial match queries'

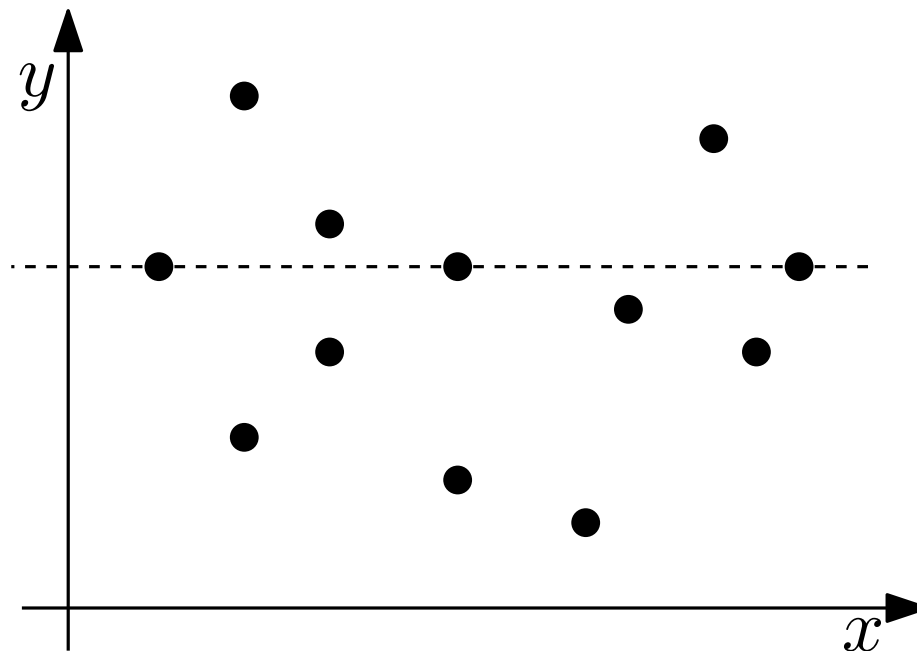


Gebe mir alle Punkte mit $y = 7$.

b) Wie kann man range-Trees dafür nutzen?

Aufgabe 2

'partial match queries'

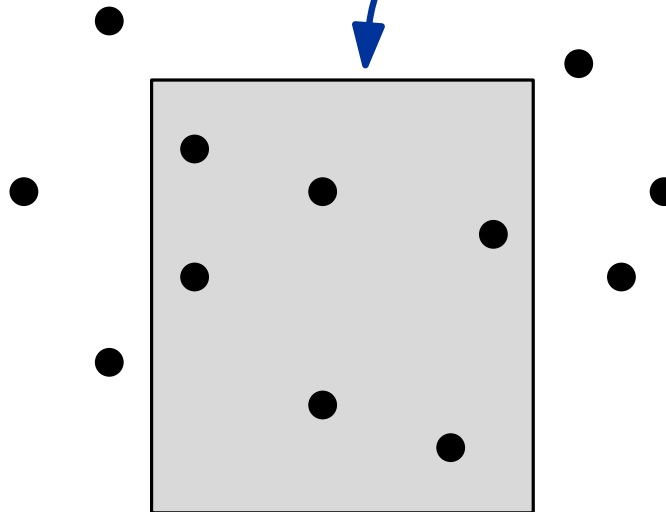


Gebe mir alle Punkte mit $y = 7$.

c) Gesucht: Datenstruktur, die das Problem in $\mathcal{O}(\log n + k)$ Zeit und $\mathcal{O}(n)$ Speicher löst.

Aufgabe 3

Wie viele Punkte liegen in diesem Rechteck?

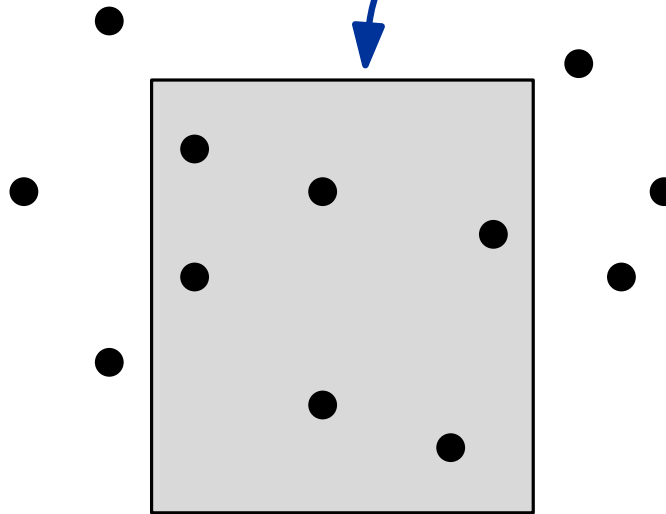


range counting query

Anforderung: Additive Konstante $\mathcal{O}(k)$ in Laufzeit vermeiden

Aufgabe 3

Wie viele Punkte liegen in diesem Rechteck?



range counting query

Anforderung: Additive Konstante $\mathcal{O}(k)$ in Laufzeit vermeiden

a) Adaptiere 1-dim Range-Tree um range counting queries in $\mathcal{O}(\log n)$ machbar ist

Aufgabe 3

1dRangeQuery(T, x, x')

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

else

$v \leftarrow \text{lc}(v_{\text{split}})$

while v kein Blatt **do**

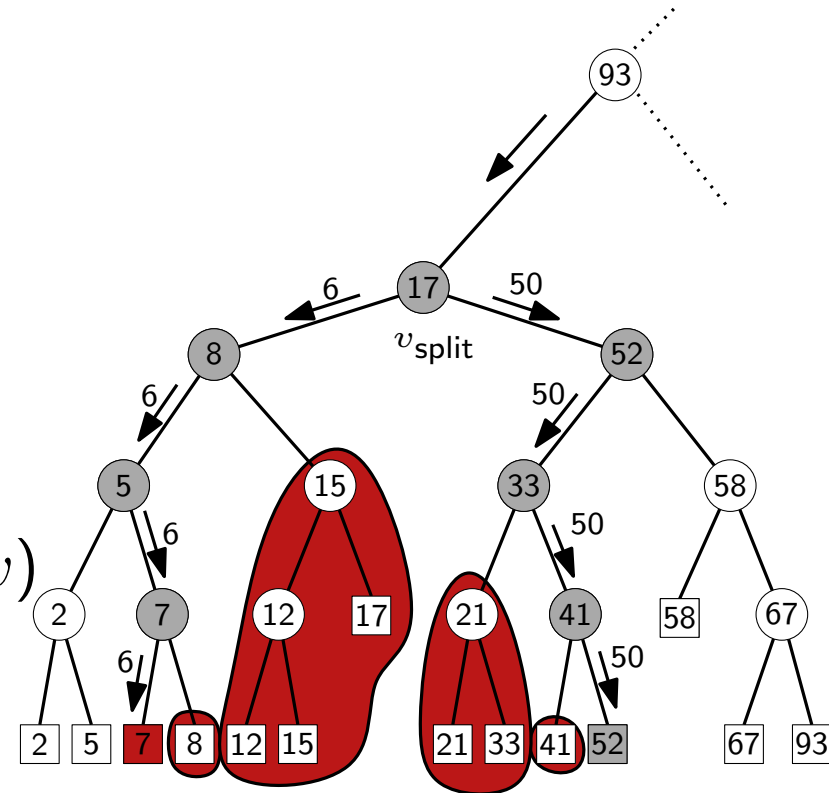
if $x \leq x_v$ **then**

 | ReportSubtree($\text{rc}(v)$); $v \leftarrow \text{lc}(v)$

else $v \leftarrow \text{rc}(v)$

 prüfe v

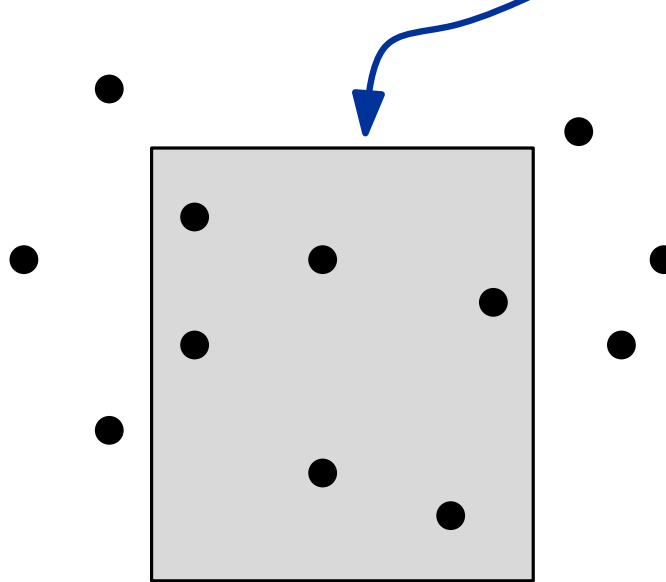
 // analog für x' und $\text{rc}(v_{\text{split}})$



a) Adaptiere 1-dim Range-Tree um range counting queries in $\mathcal{O}(\log n)$ machbar ist

Aufgabe 3

Wie viele Punkte liegen in diesem Rechteck?

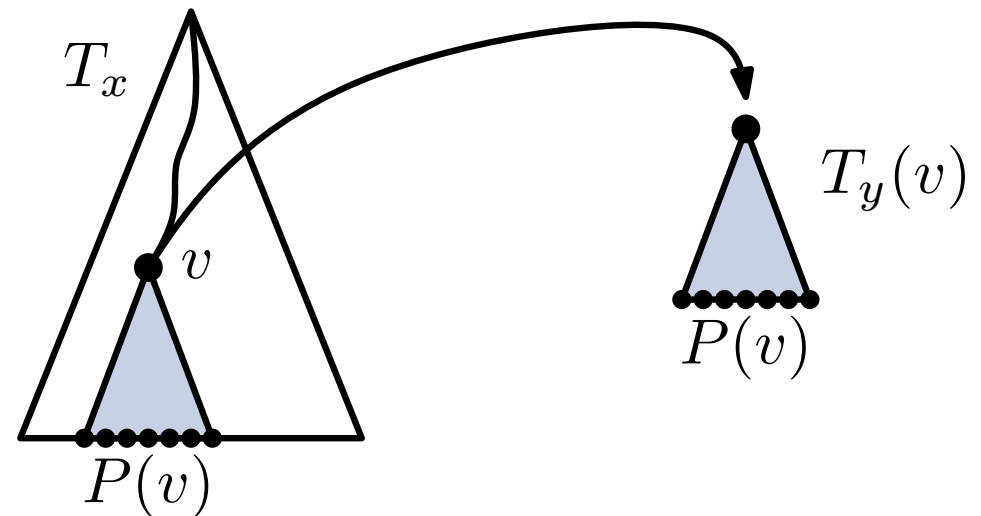


range counting query

Anforderung: Additive Konstante $\mathcal{O}(k)$ in Laufzeit vermeiden

b) Benutze Lösung aus a) um das d -dimensionale Problem zu lösen

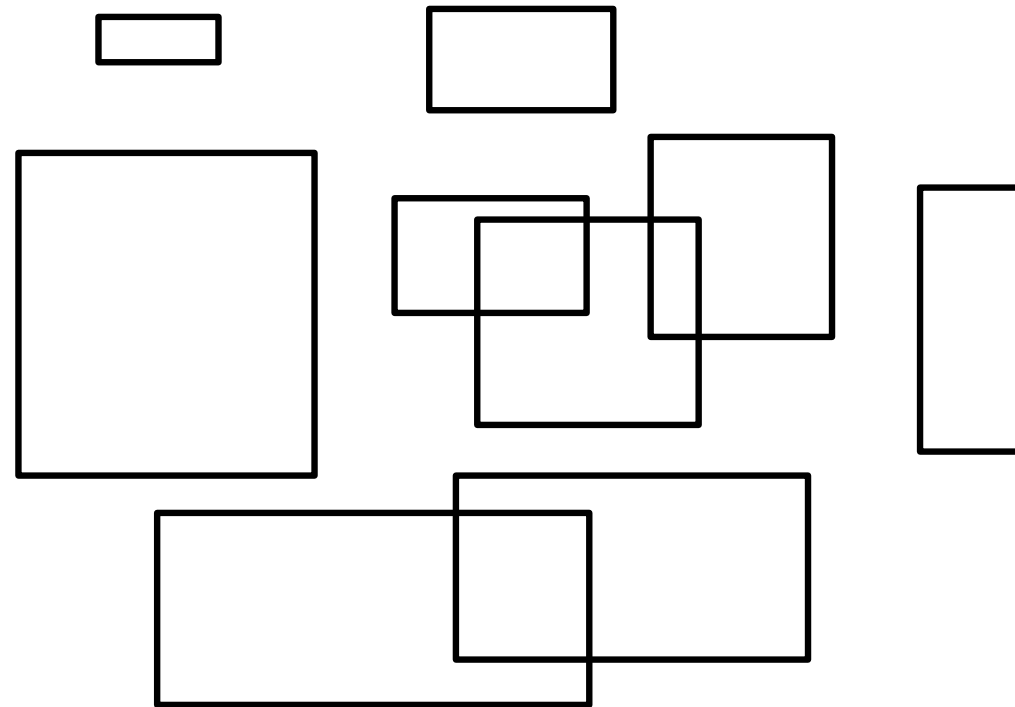
2dim Range-Trees:



Anforderung: Additive Konstante $\mathcal{O}(k)$ in Laufzeit vermeiden

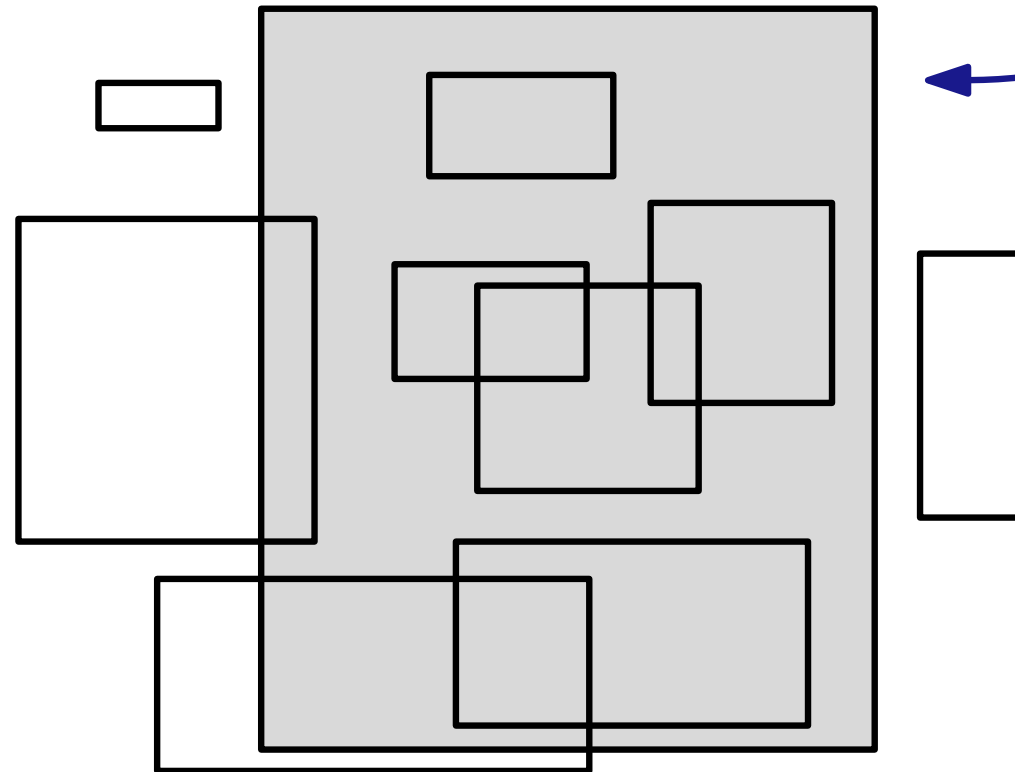
b) Benutze Lösung aus a) um das d -dimensionale Problem zu lösen

Aufgabe 4



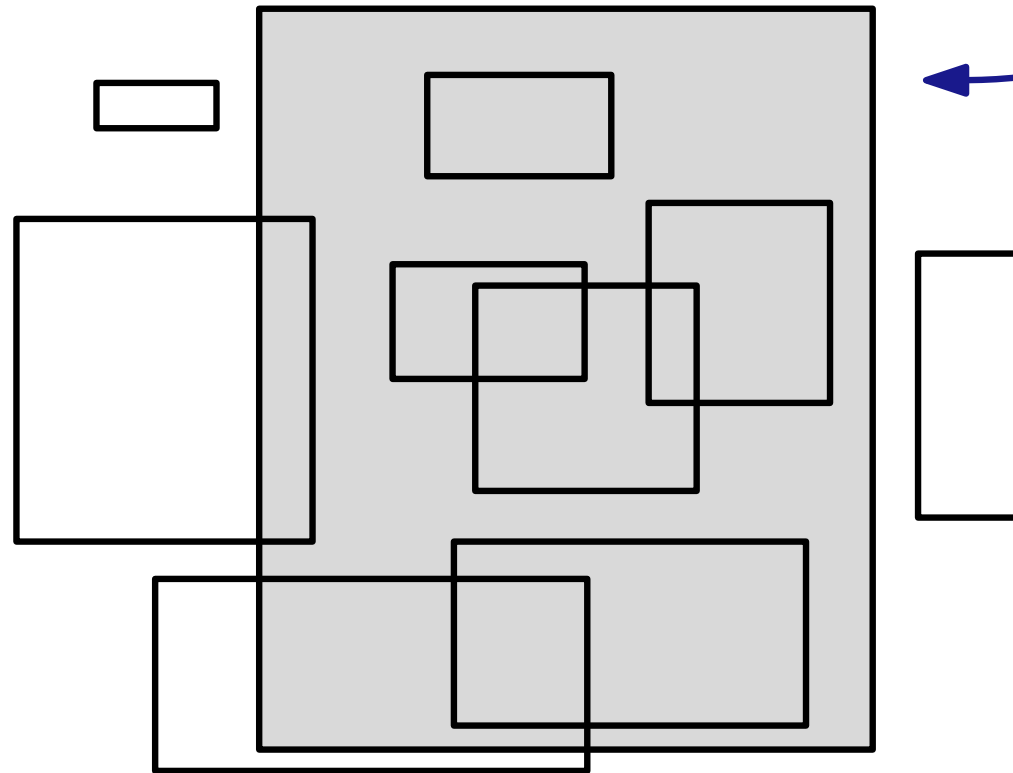
Aufgabe 4

Welche Rechtecke liegen vollständig in diesem Rechteck?



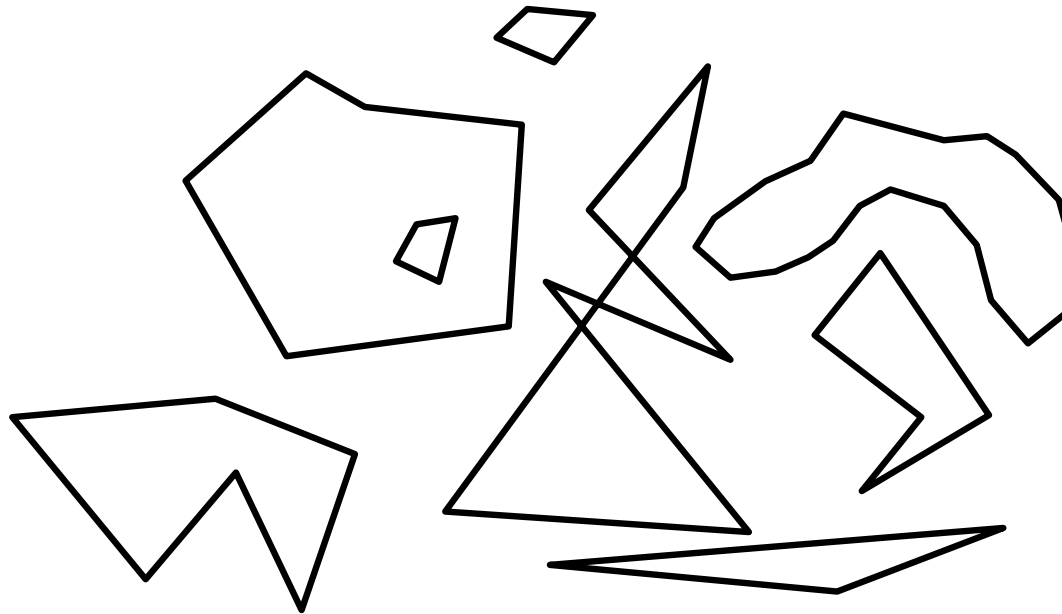
Aufgabe 4

Welche Rechtecke liegen vollständig in diesem Rechteck?



Datenstruktur mit $\mathcal{O}(n \log^3 n)$ Speicher und $\mathcal{O}(\log^4 n + k)$ Anfragezeit.

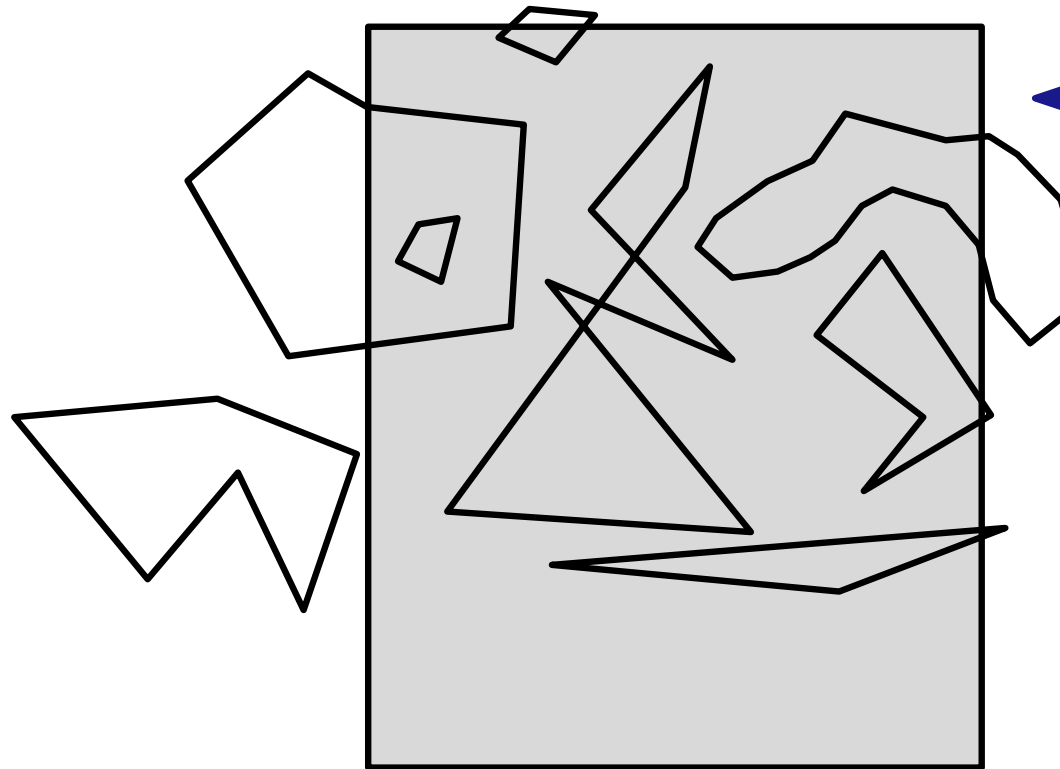
Aufgabe 4



Datenstruktur mit $\mathcal{O}(n \log^3 n)$ Speicher und $\mathcal{O}(\log^4 n + k)$ Anfragezeit.

Aufgabe 4

Welche Polygone liegen vollständig in diesem Rechteck?



Datenstruktur mit $\mathcal{O}(n \log^3 n)$ Speicher und $\mathcal{O}(\log^4 n + k)$ Anfragezeit.