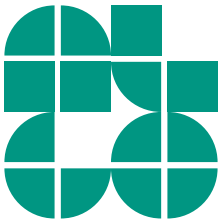
Karlsruhe Institute of Technology

Übung Algorithmische Geometrie
Bereichsanfragen

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

Andreas Gemsa
26.05.2011



Aufgabe 1

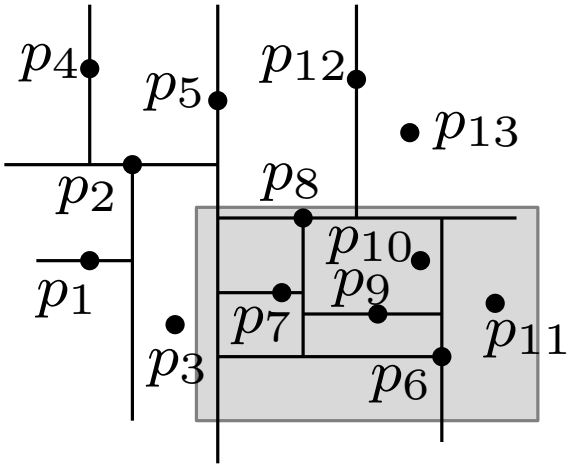
kd-Trees – w-c Laufzeit

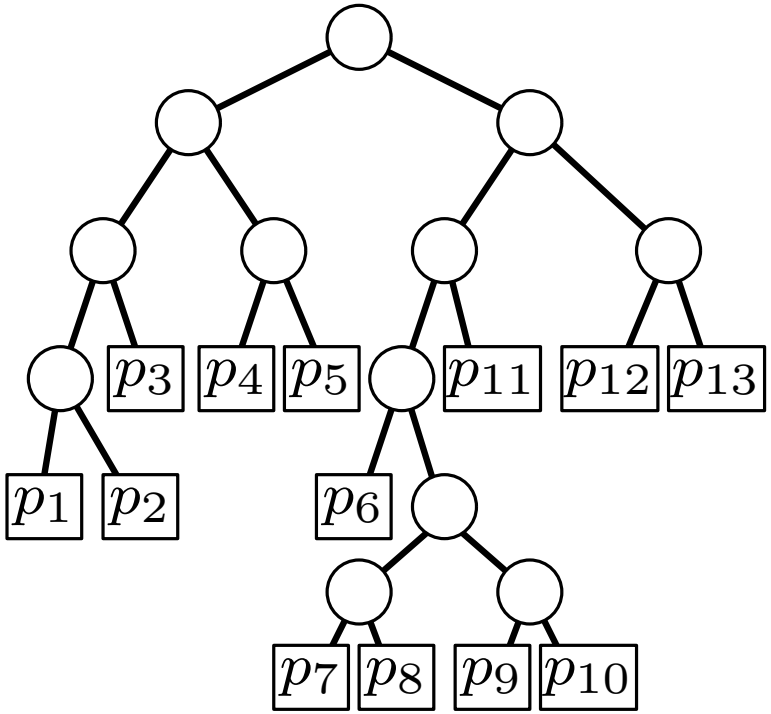
Anfragen

a) warum?

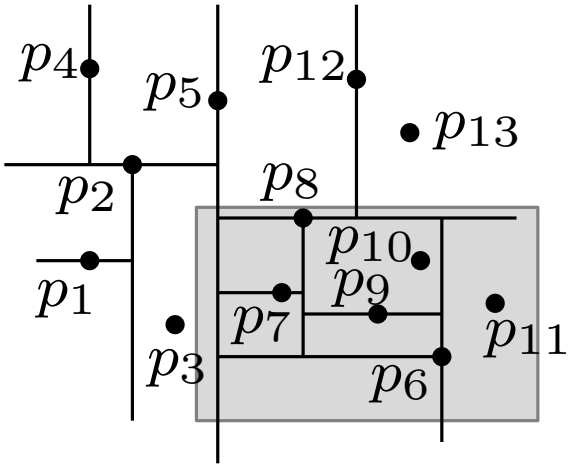
$$Q(n) = \begin{cases} \mathcal{O}(1) & , \text{ für } n = 1 \\ 2 + 2Q(n/4) & , \text{ für } n > 1 \end{cases}$$

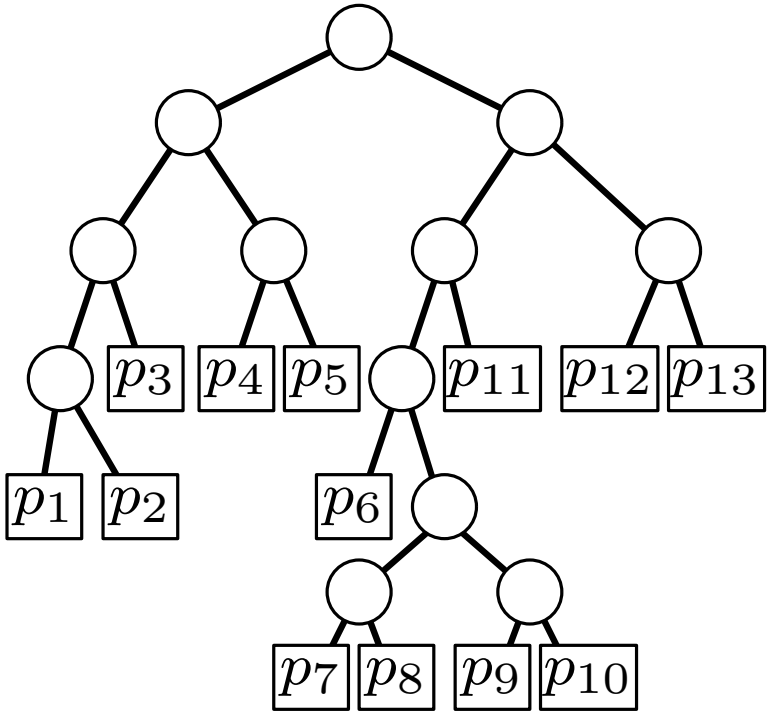
Bereichsabfrage in einem kd-Tree





Bereichsabfrage in einem kd-Tree

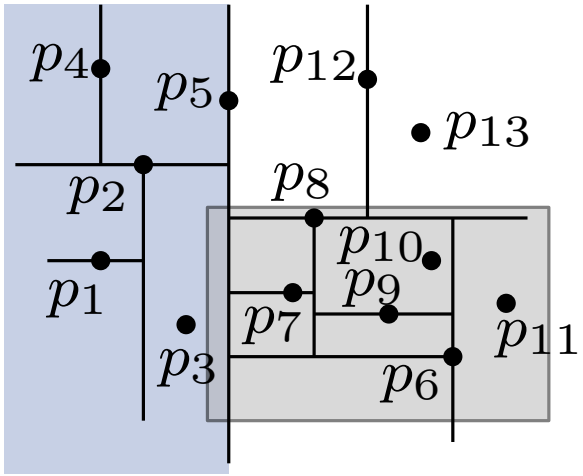


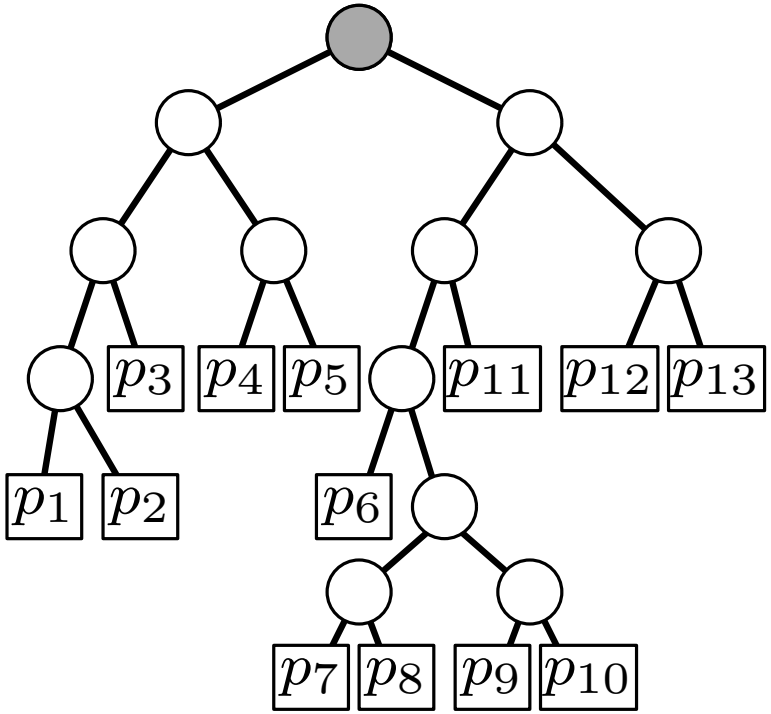


SearchKdTree(v, R)

- **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

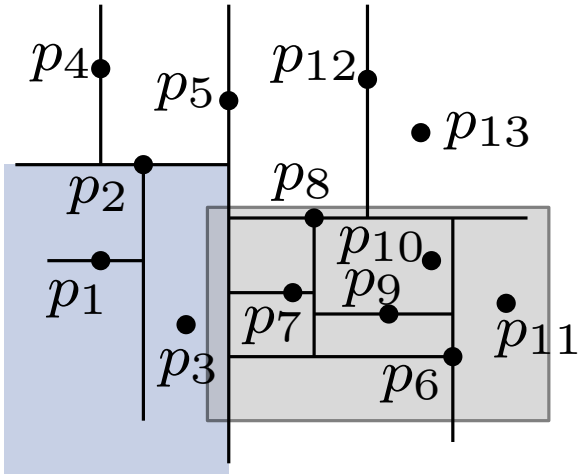


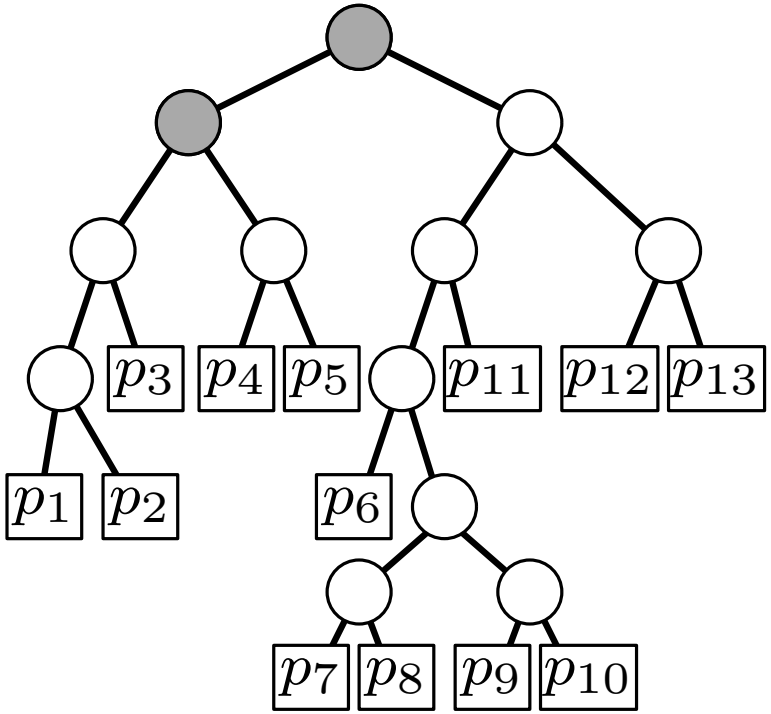


SearchKdTree(v, R)

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

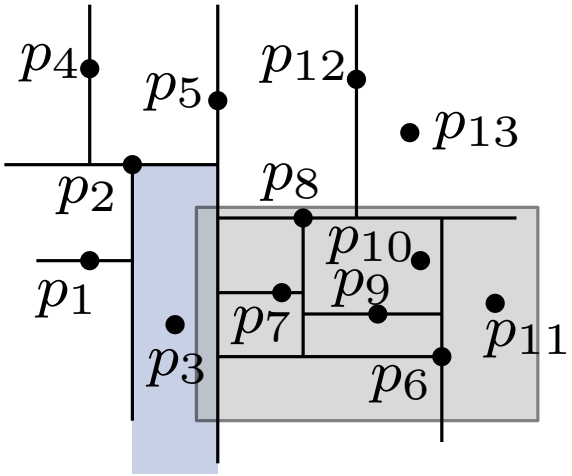


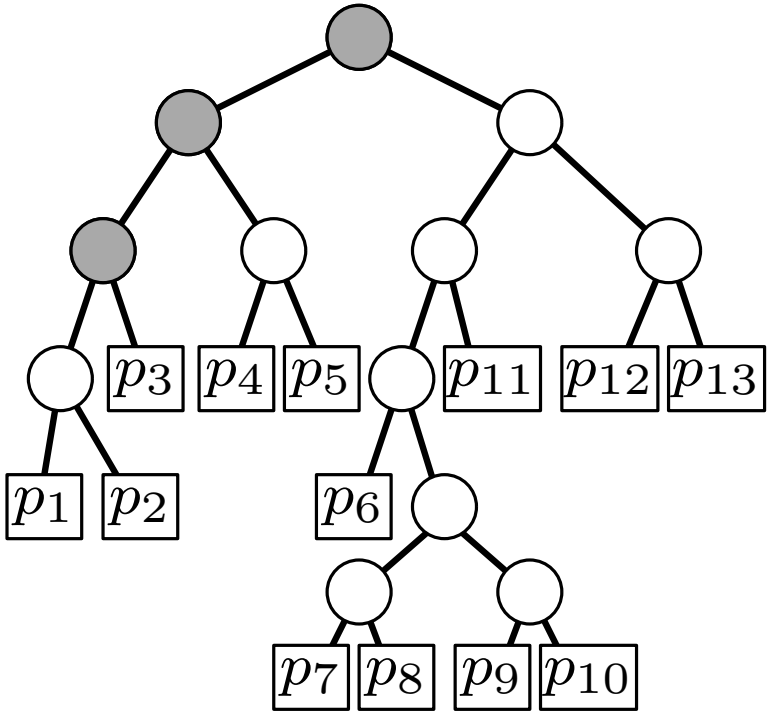


SearchKdTree(v, R)

- **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

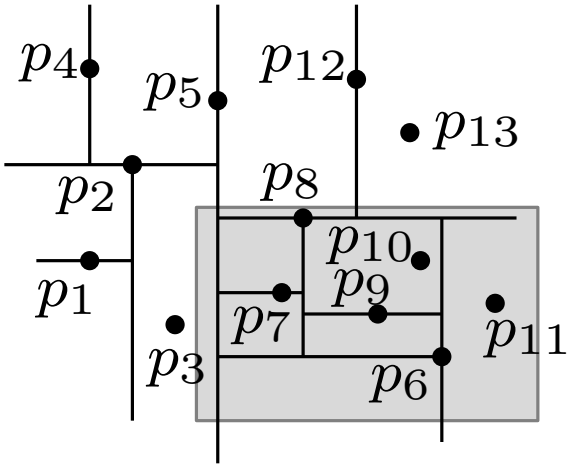


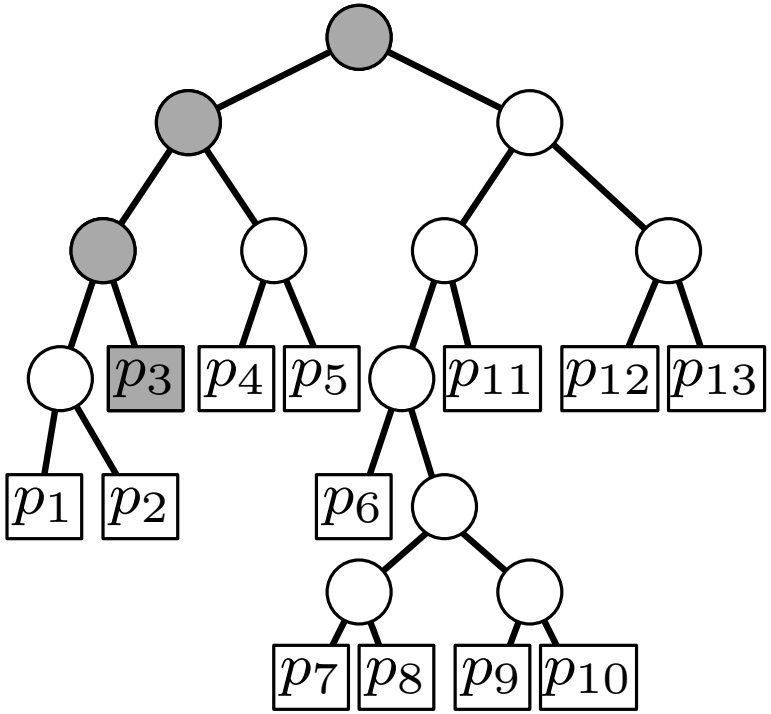


SearchKdTree(v, R)

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

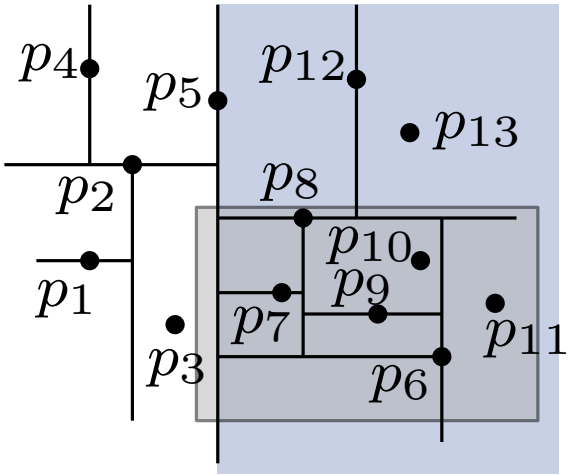


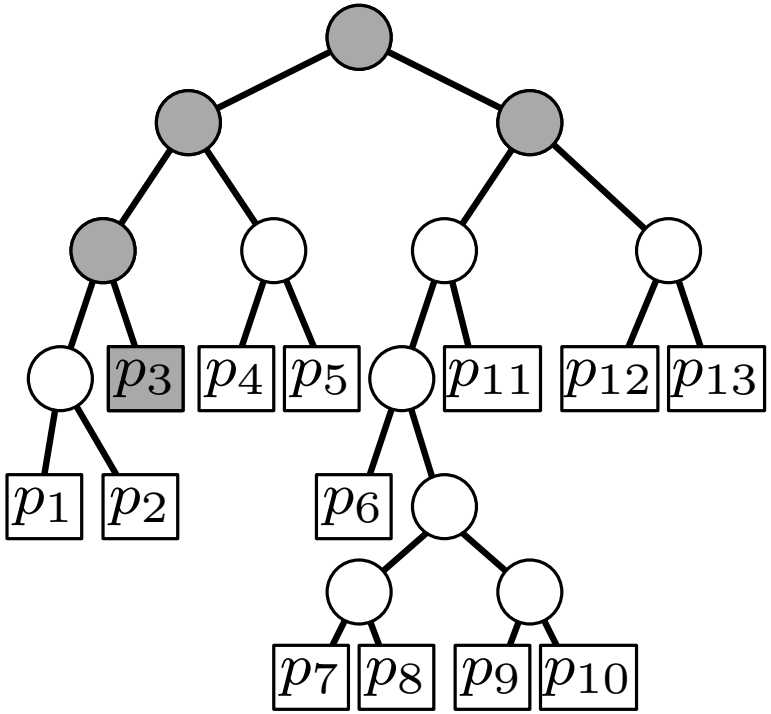


SearchKdTree(v, R)

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

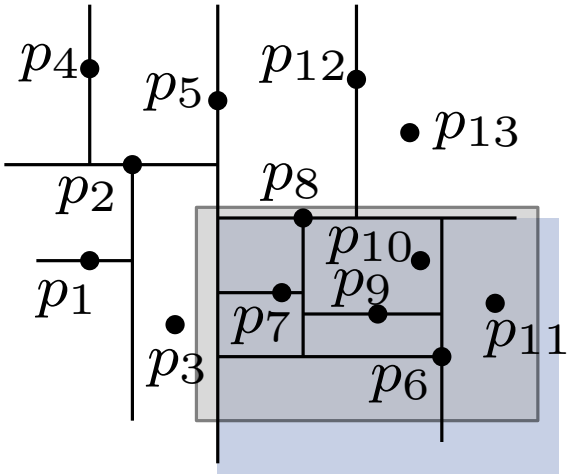


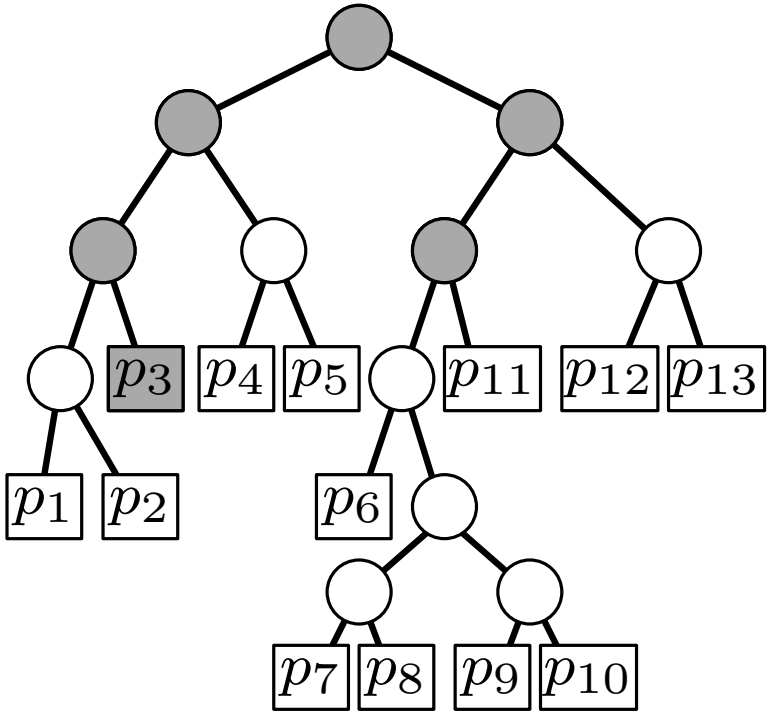


SearchKdTree(v, R)

- **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

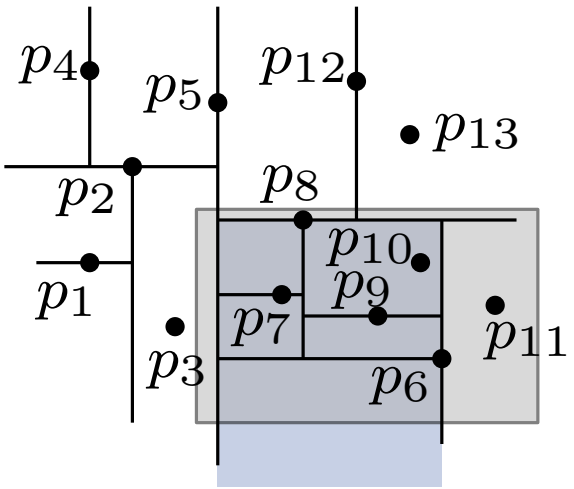


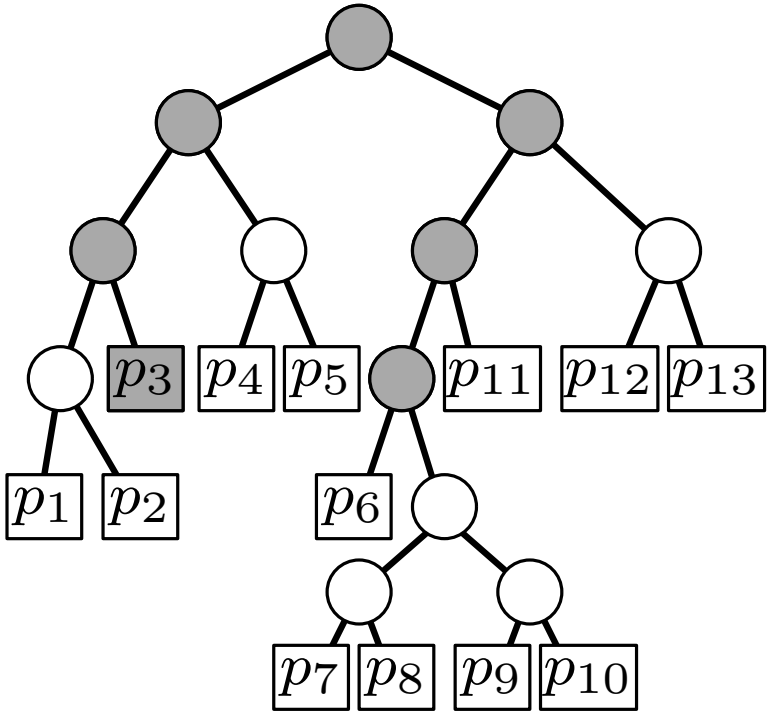


SearchKdTree(v, R)

- **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

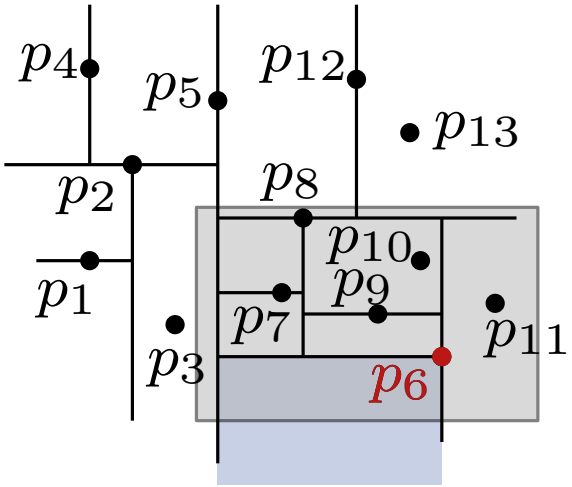


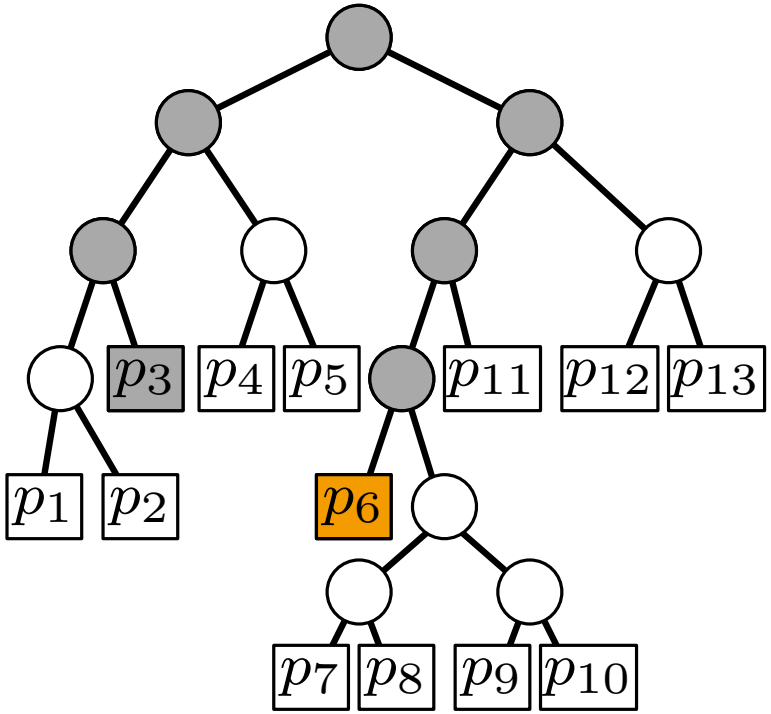


SearchKdTree(v, R)

- **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

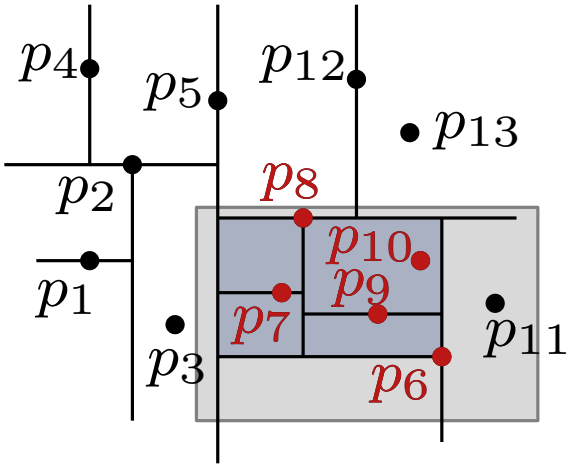


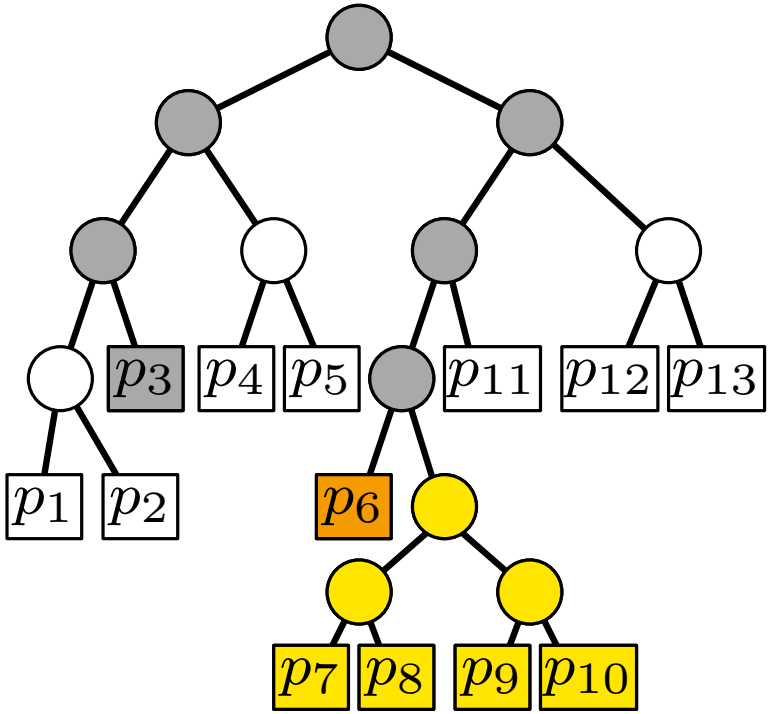


SearchKdTree(v, R)

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

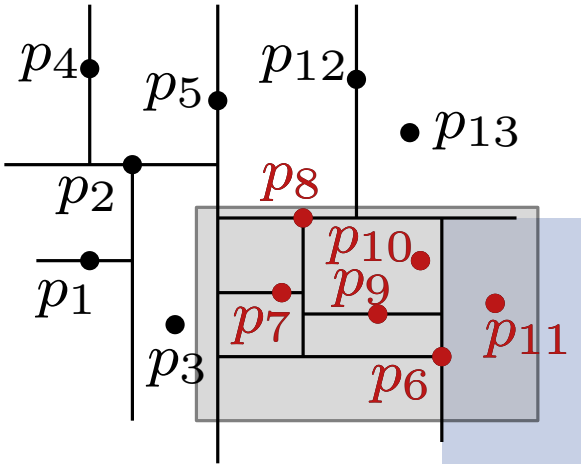


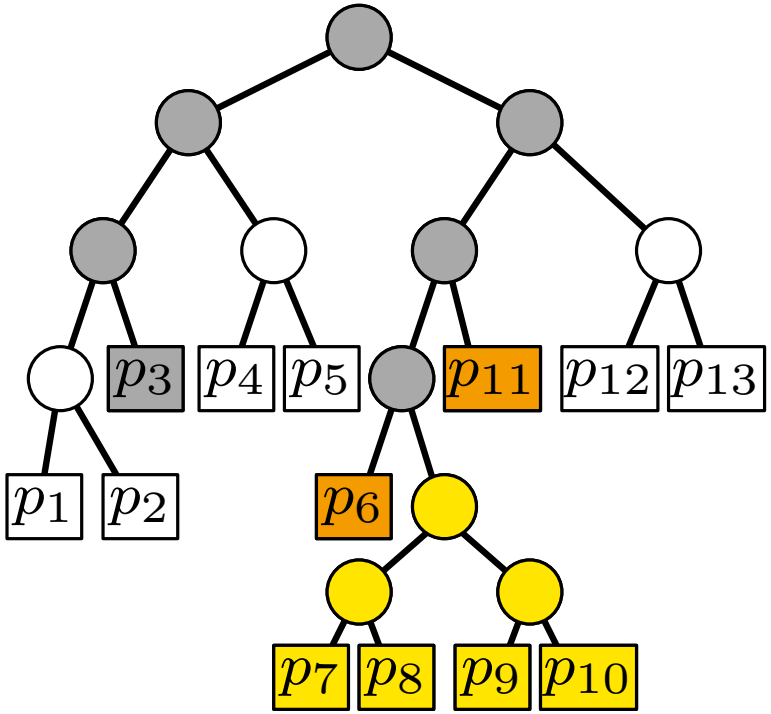


SearchKdTree(v, R)

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

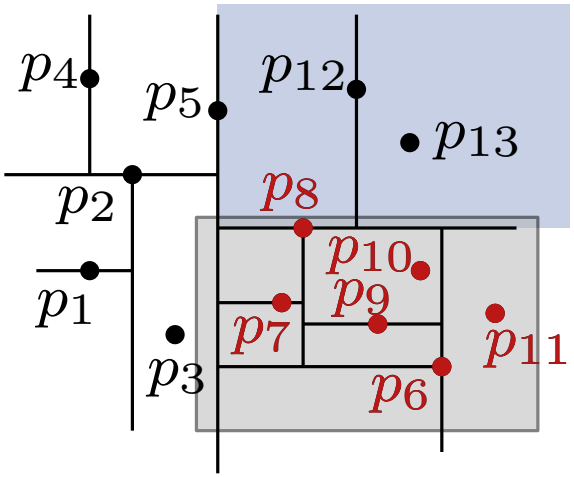


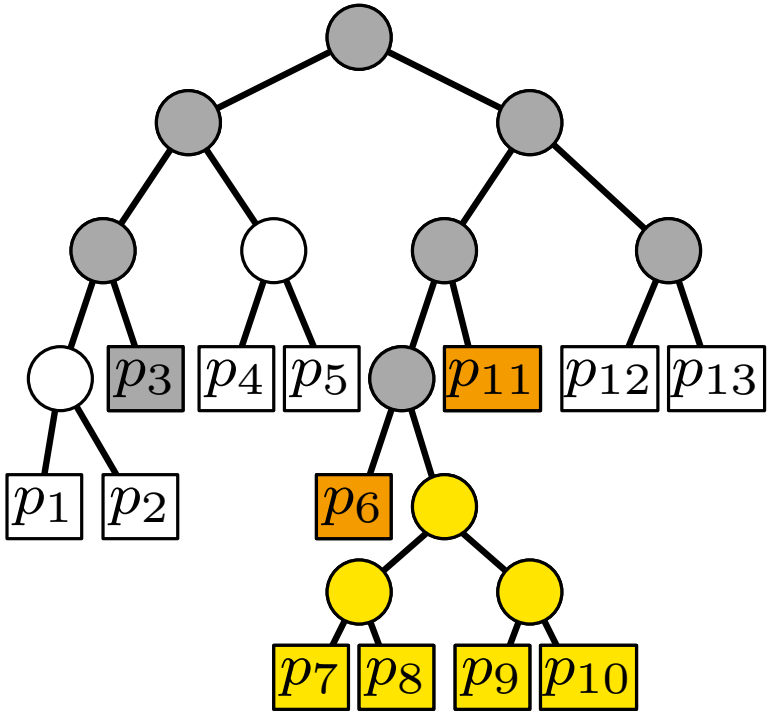


SearchKdTree(v, R)

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)

Bereichsabfrage in einem kd-Tree

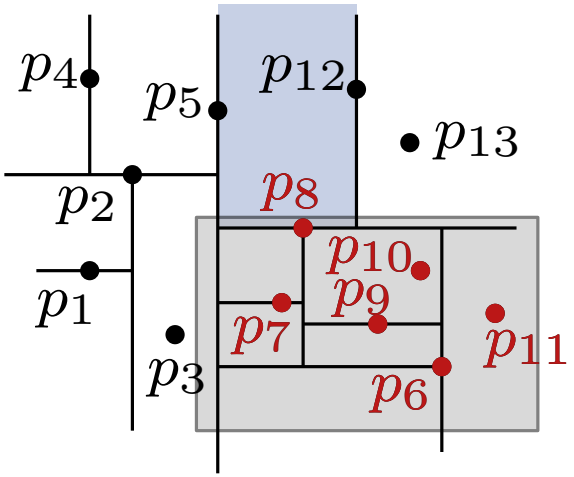


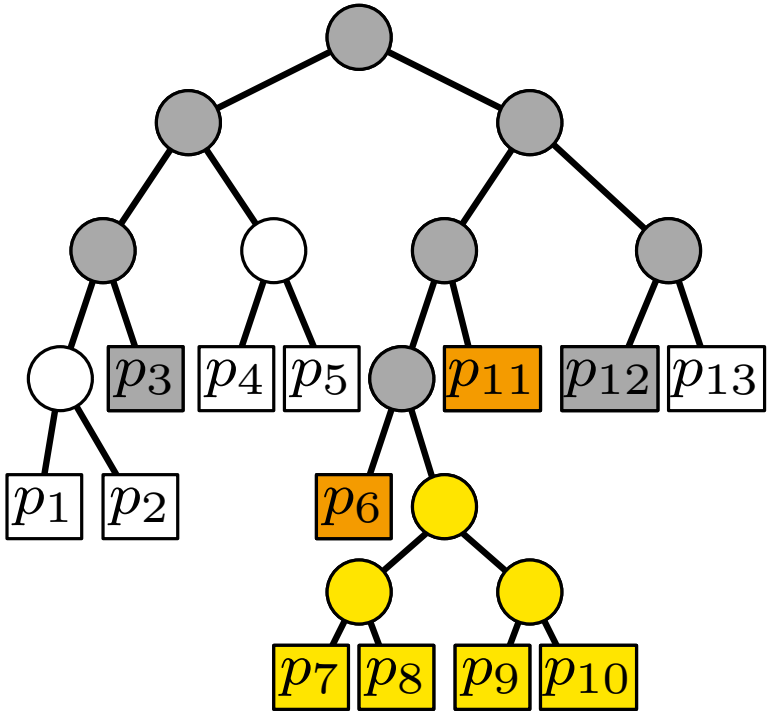


SearchKdTree(v, R)

- **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

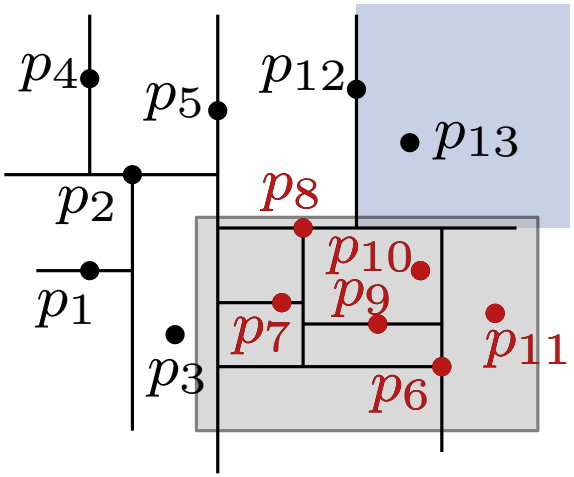


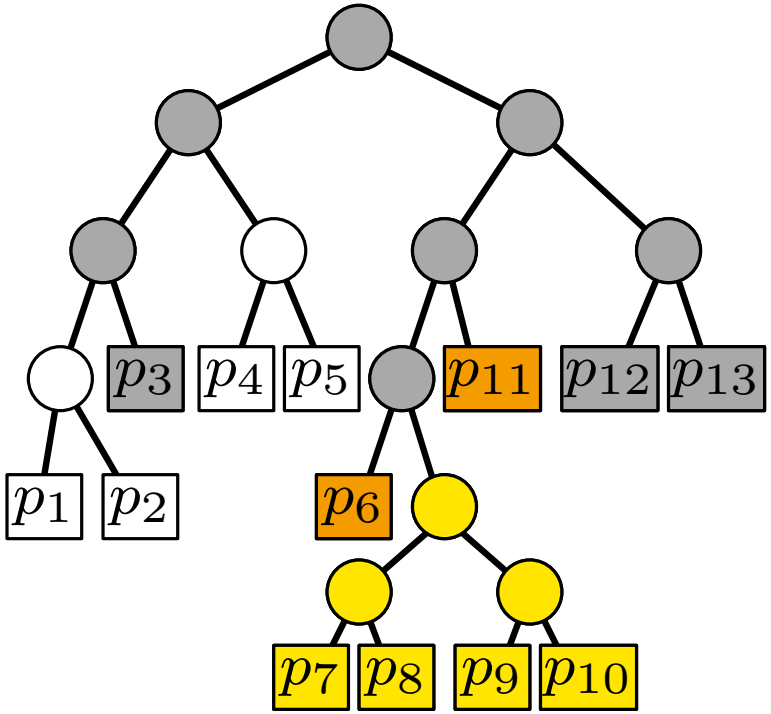


SearchKdTree(v, R)

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

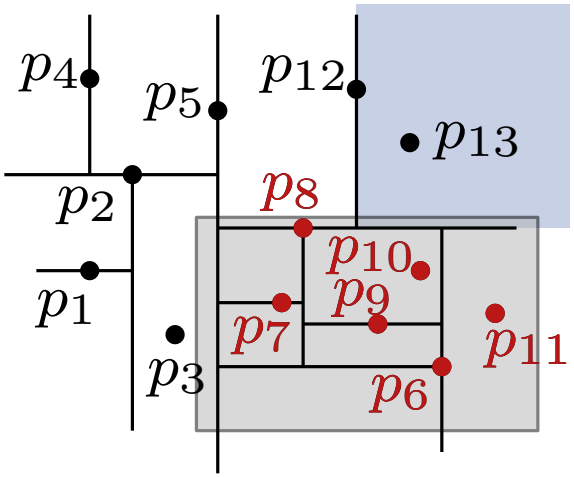


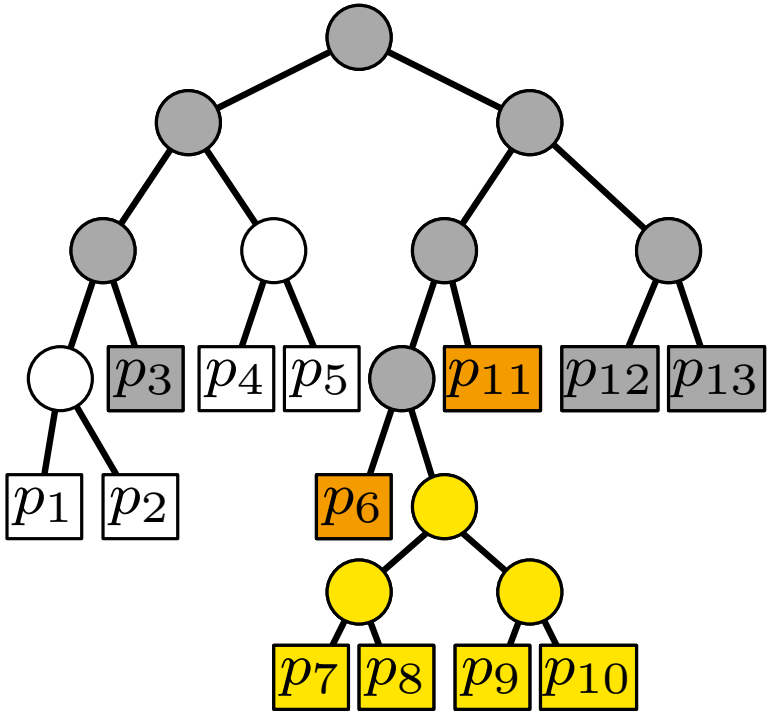


SearchKdTree(v, R)

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

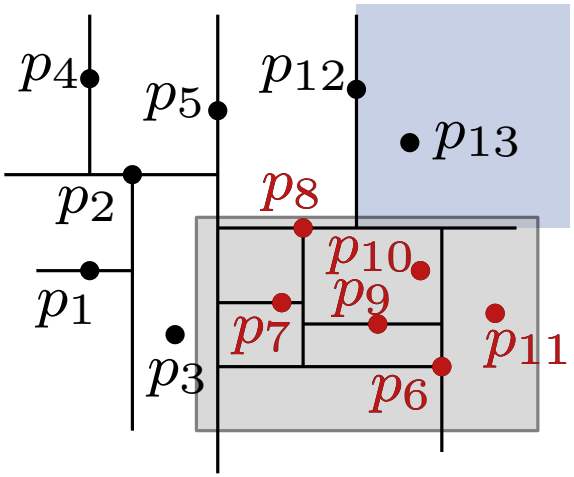




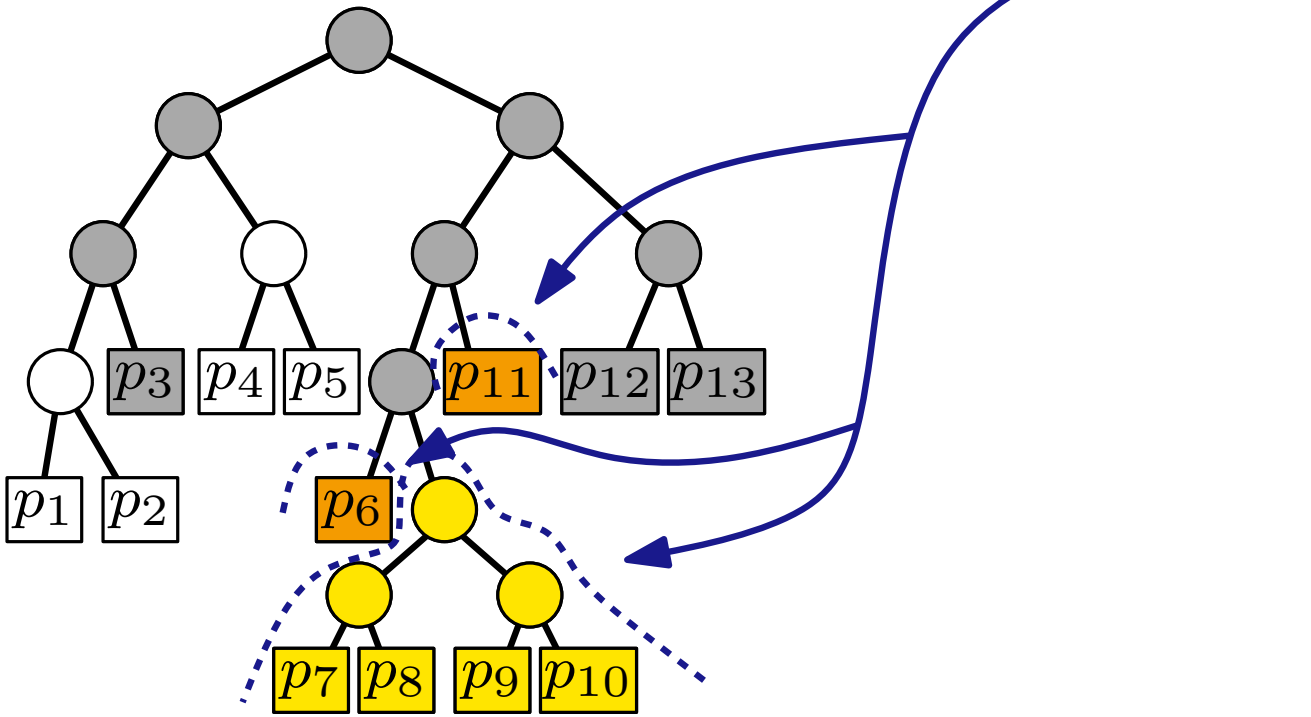
SearchKdTree(v, R)

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)

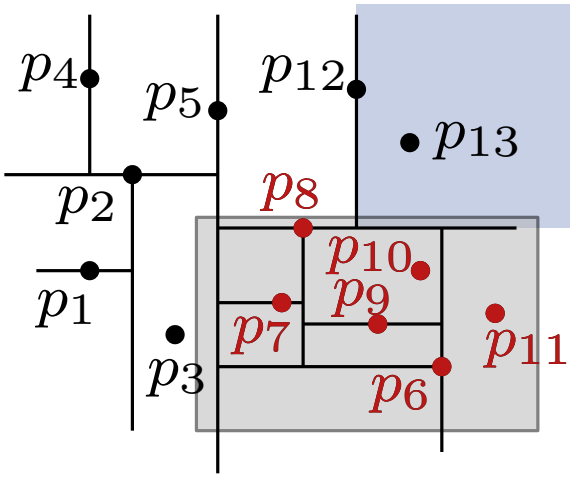
Bereichsabfrage in einem kd-Tree



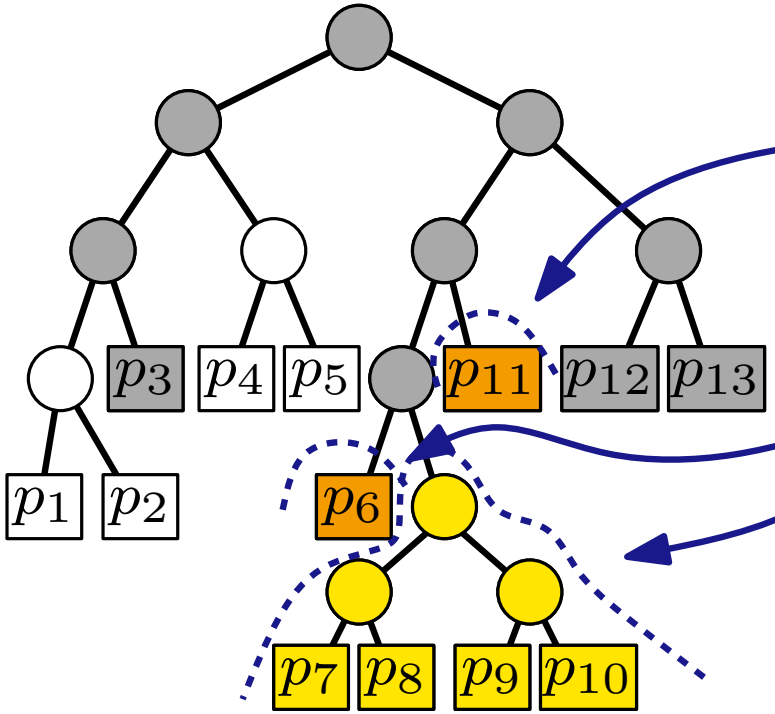
ReportSubtree in $\mathcal{O}(k)$



Bereichsabfrage in einem kd-Tree

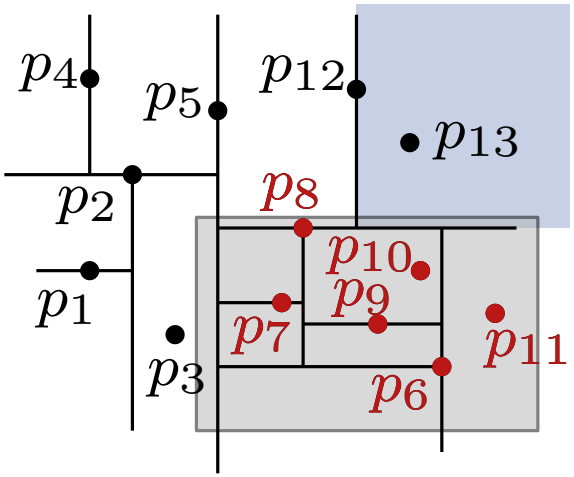


ReportSubtree in $\mathcal{O}(k)$

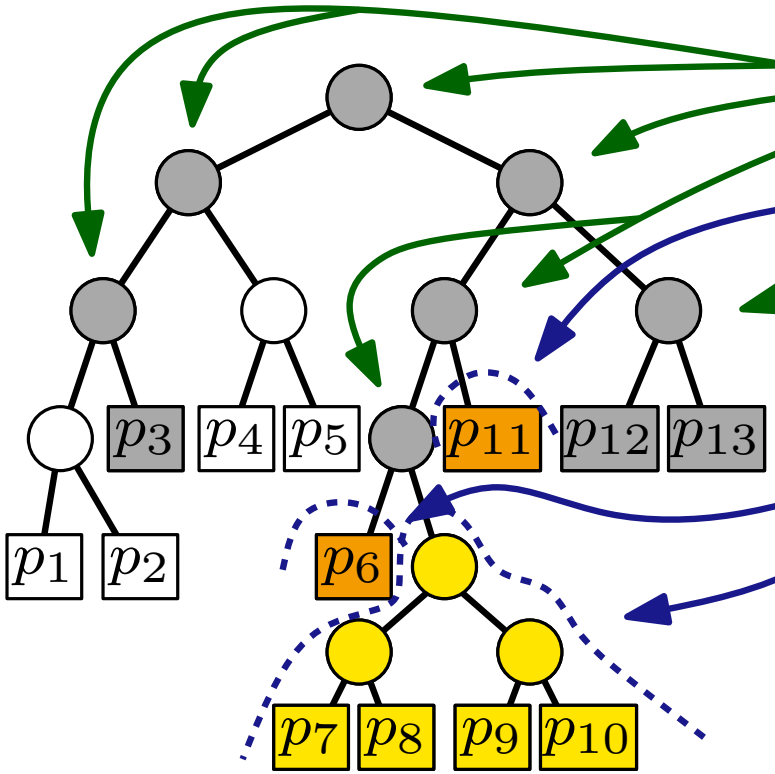


Wie viele Knoten betrachten wir?

Bereichsabfrage in einem kd-Tree

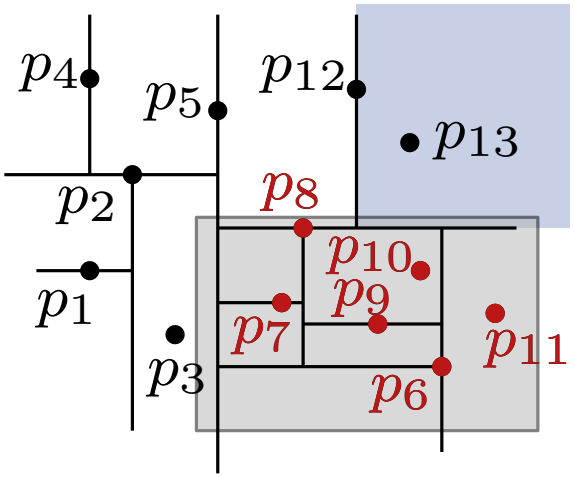


ReportSubtree in $\mathcal{O}(k)$

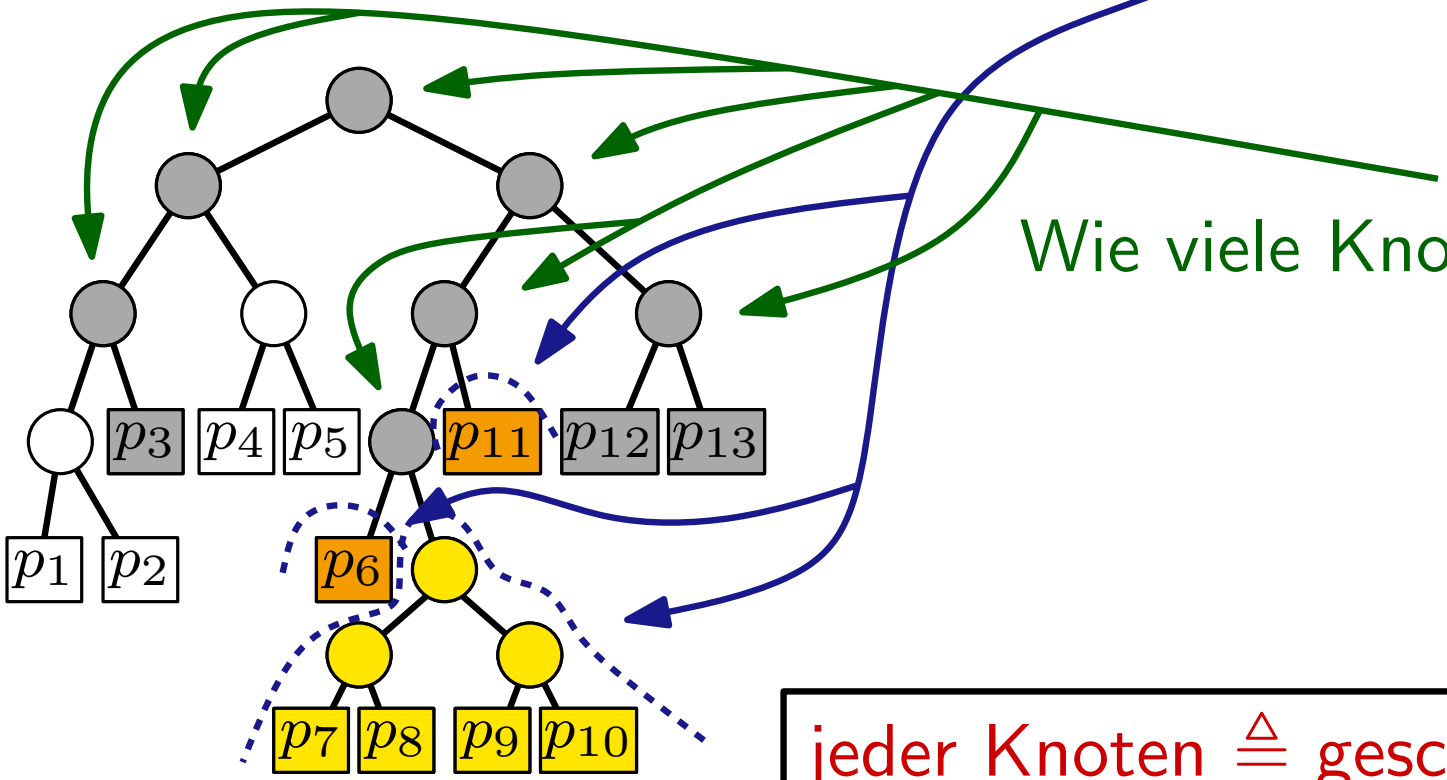


Wie viele Knoten betrachten wir?

Bereichsabfrage in einem kd-Tree



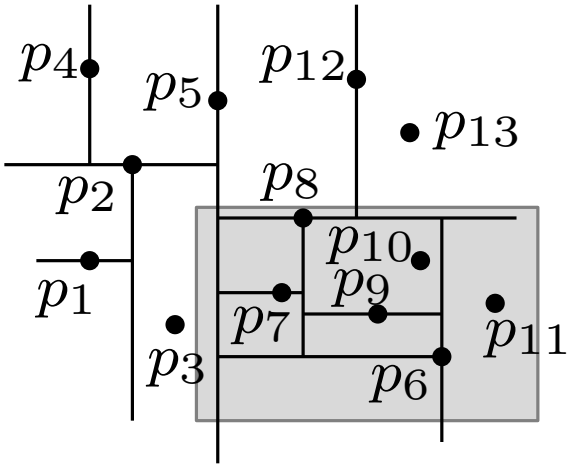
ReportSubtree in $\mathcal{O}(k)$

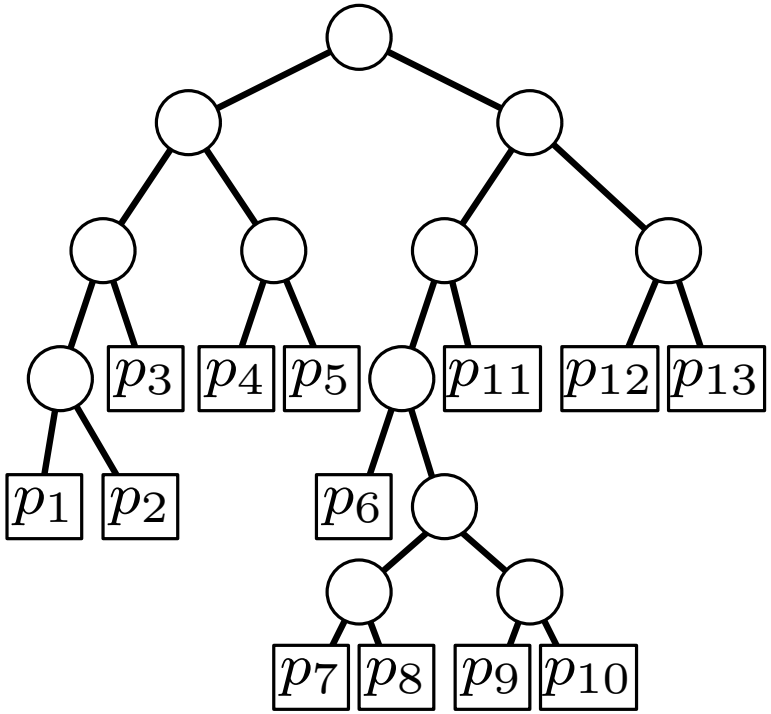


Wie viele Knoten betrachten wir?

jeder Knoten $\triangleq$ geschnittene Region

Bereichsabfrage in einem kd-Tree

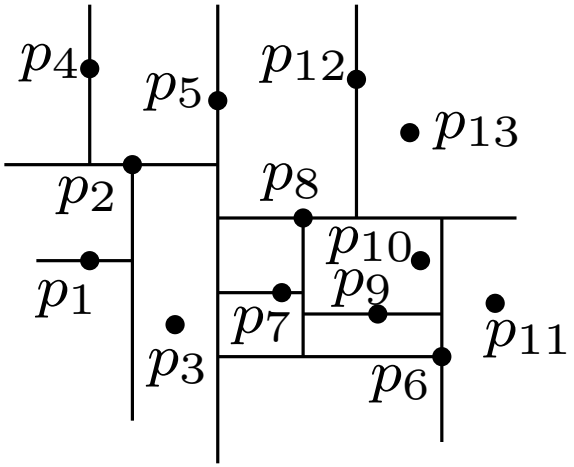




SearchKdTree(v, R)

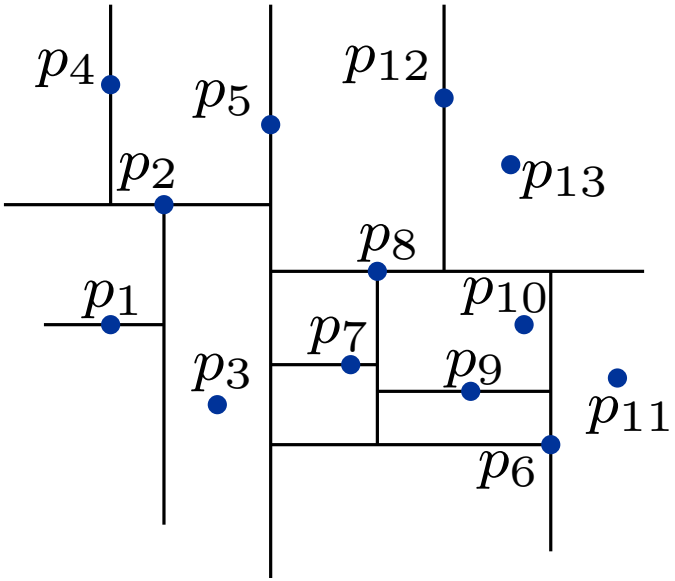
- **if** v Blatt **then**
  - prüfe Punkt p in v auf $p \in R$
- **else**
  - **if** region(lc(v)) $\subseteq R$ **then**
    - ReportSubtree(lc(v))
  - **else**
    - **if** region(lc(v)) $\cap R \neq \emptyset$ **then**
      - SearchKdTree(lc(v), R)
  - **if** region(rc(v)) $\subseteq R$ **then**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

Bereichsabfrage in einem kd-Tree

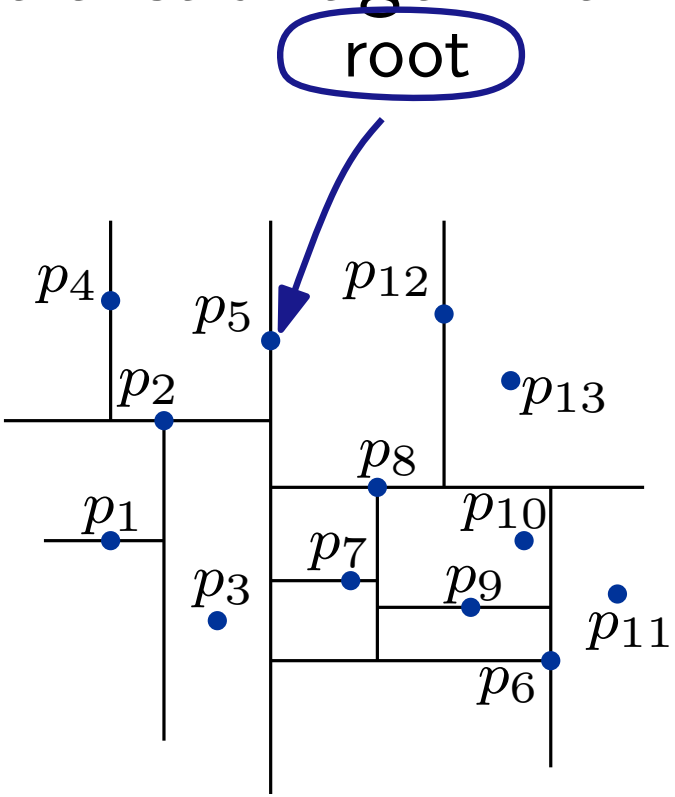


Wie viele Regionen schneidet eine vertikale Linie?

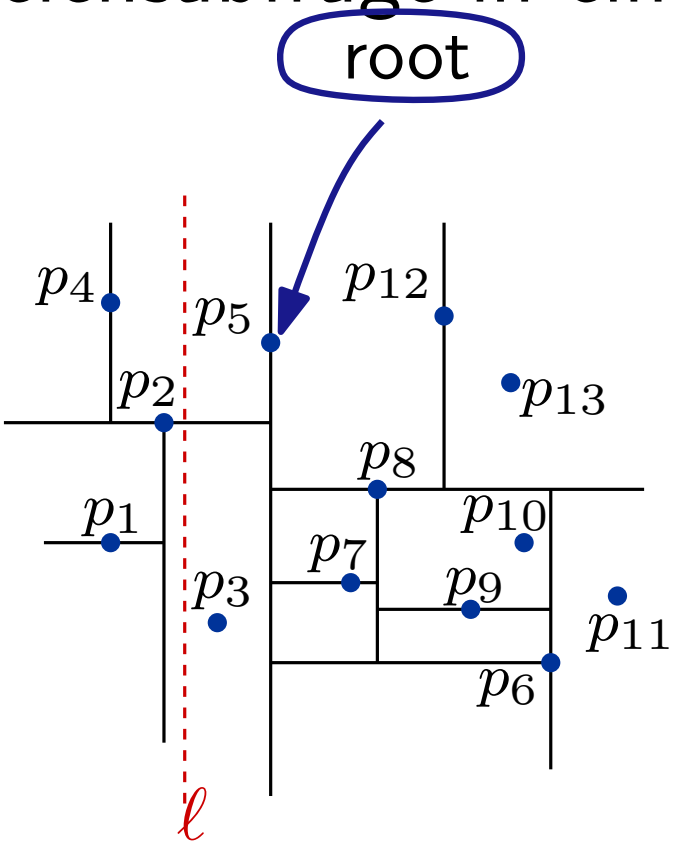
Bereichsabfrage in einem kd-Tree



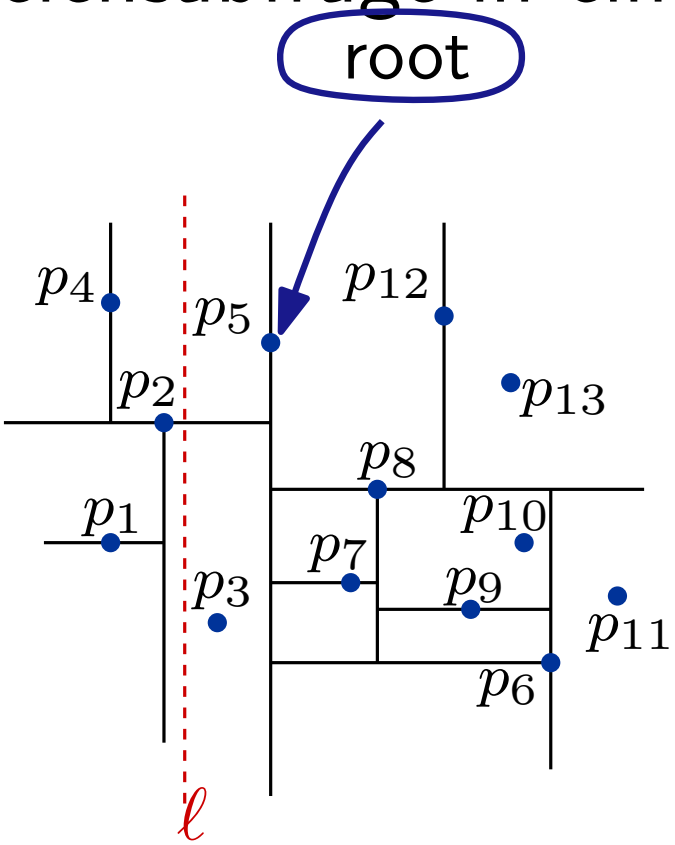
Bereichsabfrage in einem kd-Tree



Bereichsabfrage in einem kd-Tree



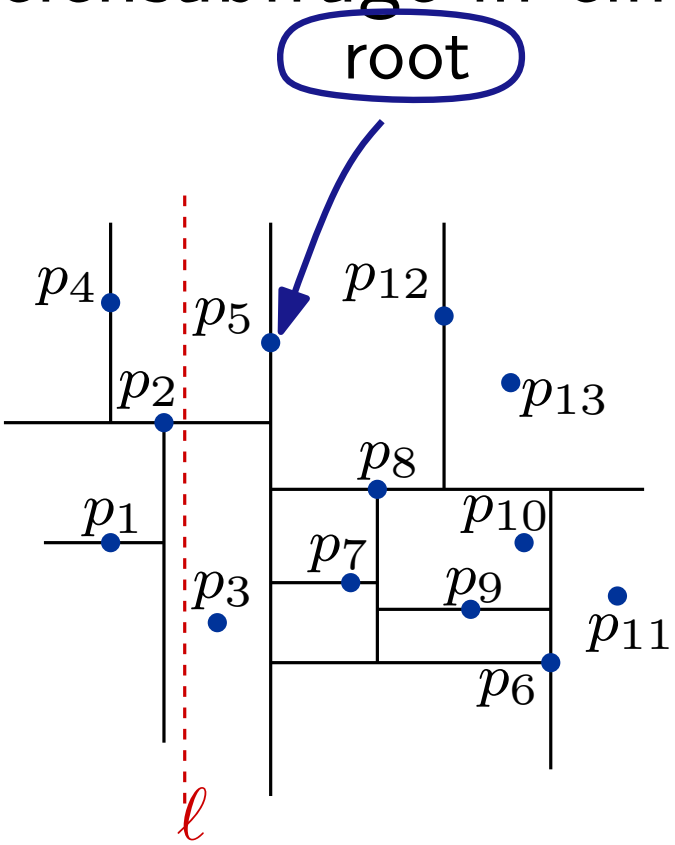
Bereichsabfrage in einem kd-Tree



Vermutung:

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

Bereichsabfrage in einem kd-Tree



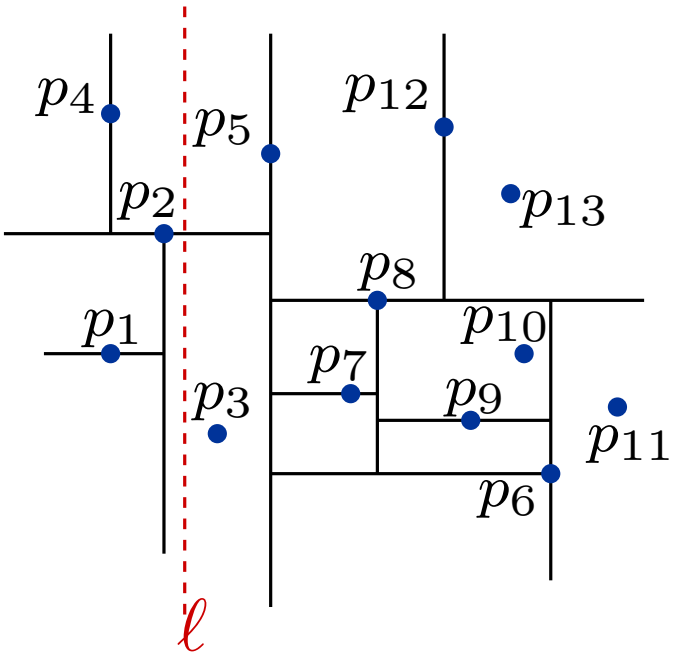
Vermutung:

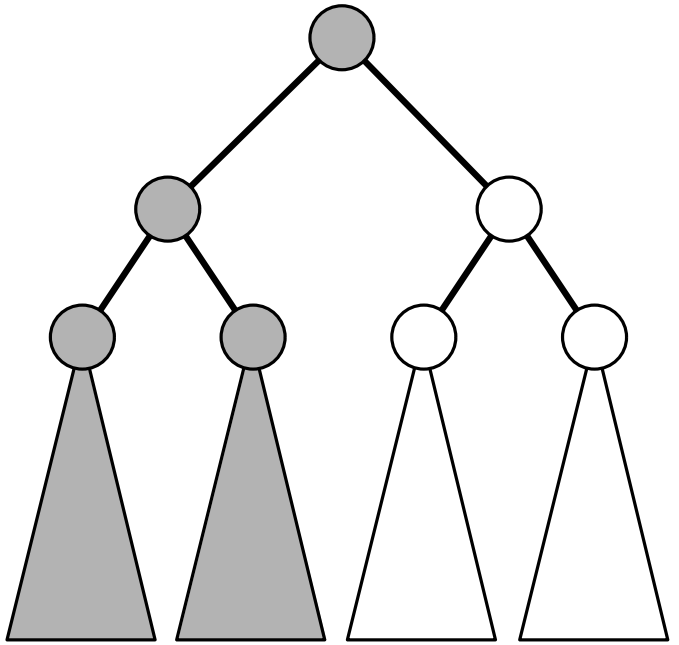
$Q(n) = 1 + Q(n/2)$

Problem?

ℓ schneidet beide Kinder des linken Kindes von root

Bereichsabfrage in einem kd-Tree





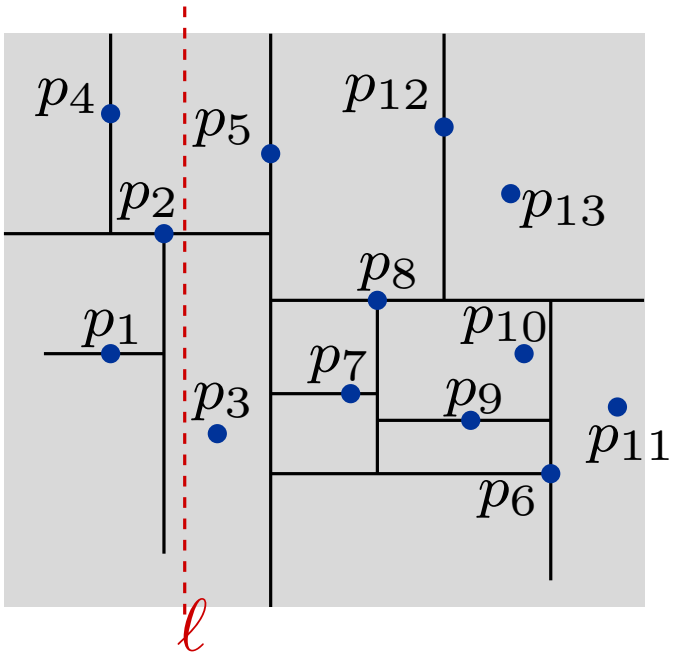
Vermutung:

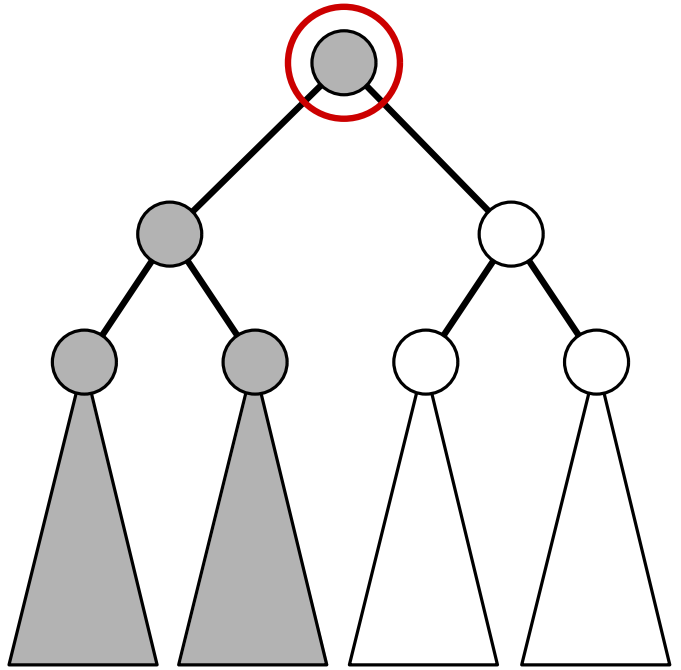
$Q(n) = 1 + Q(n/2)$

Problem?

ℓ schneidet beide Kinder des linken Kindes von root

Bereichsabfrage in einem kd-Tree





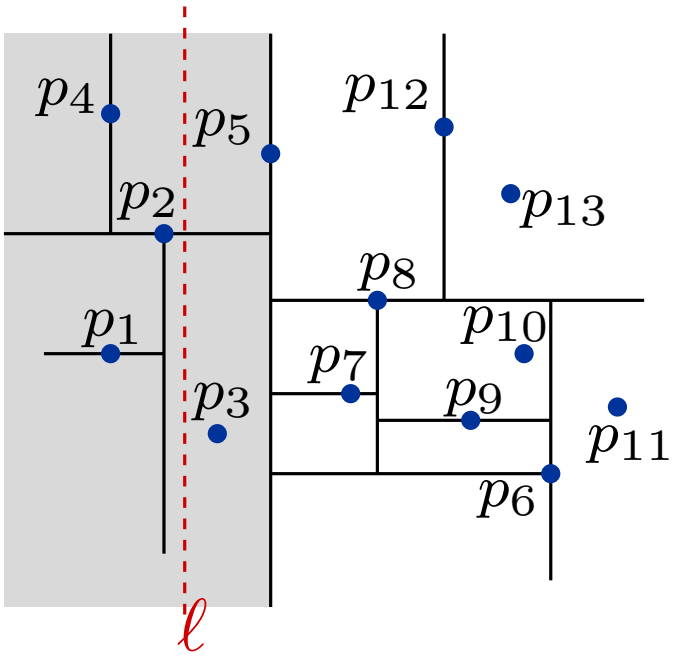
Vermutung:

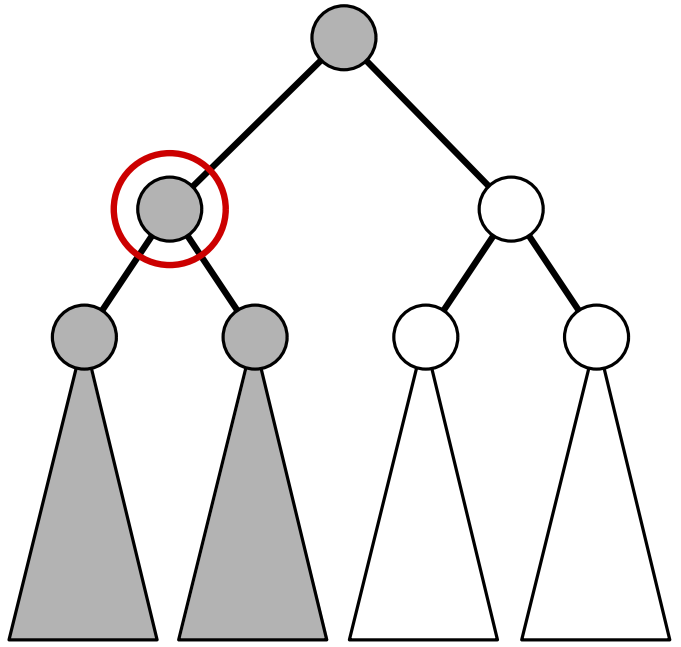
$Q(n) = 1 + Q(n/2)$

Problem?

ℓ schneidet beide Kinder des linken Kindes von root

Bereichsabfrage in einem kd-Tree





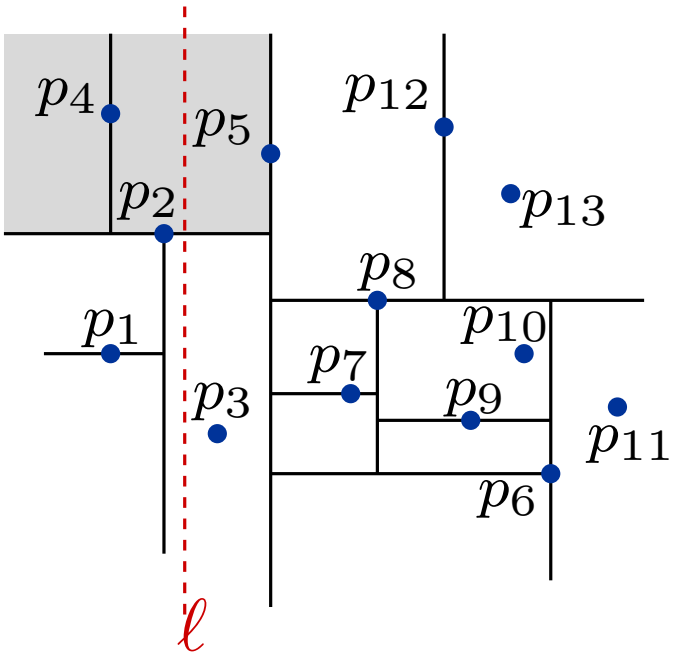
Vermutung:

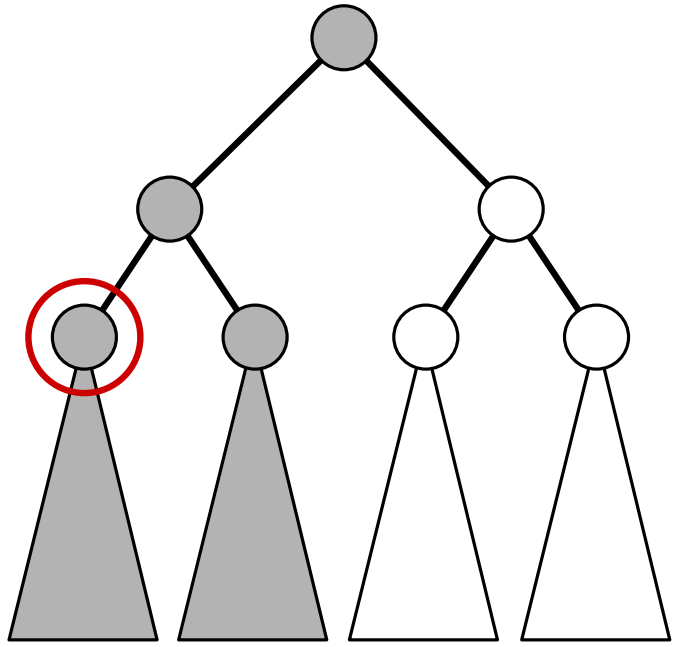
$Q(n) = 1 + Q(n/2)$

Problem?

ℓ schneidet beide Kinder des linken Kindes von root

Bereichsabfrage in einem kd-Tree





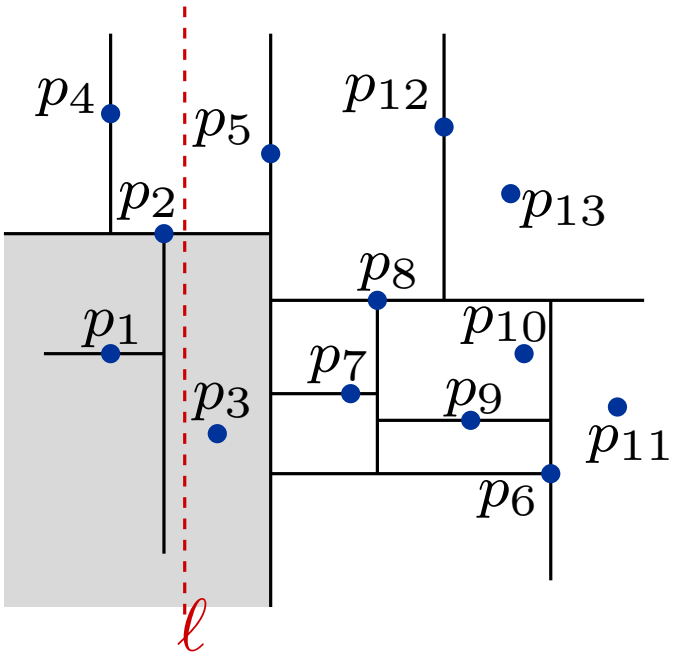
Vermutung:

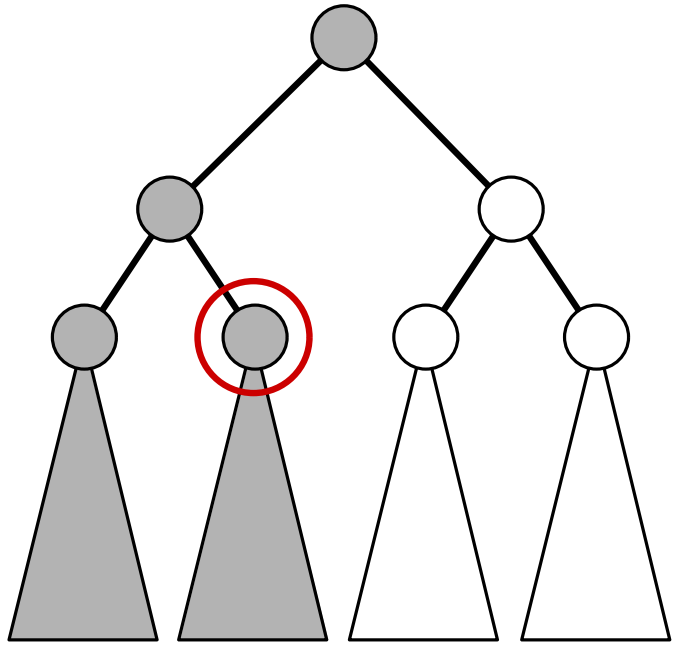
$Q(n) = 1 + Q(n/2)$

Problem?

ℓ schneidet beide Kinder des linken Kindes von root

Bereichsabfrage in einem kd-Tree





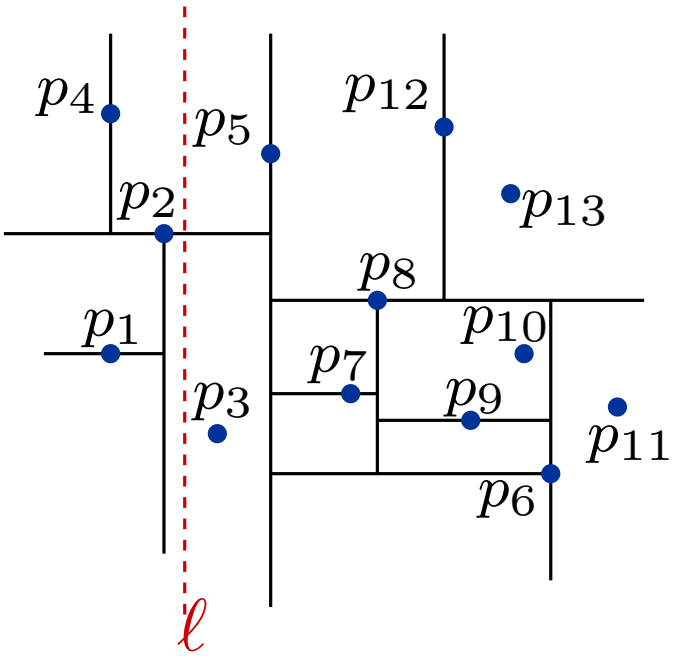
Vermutung:

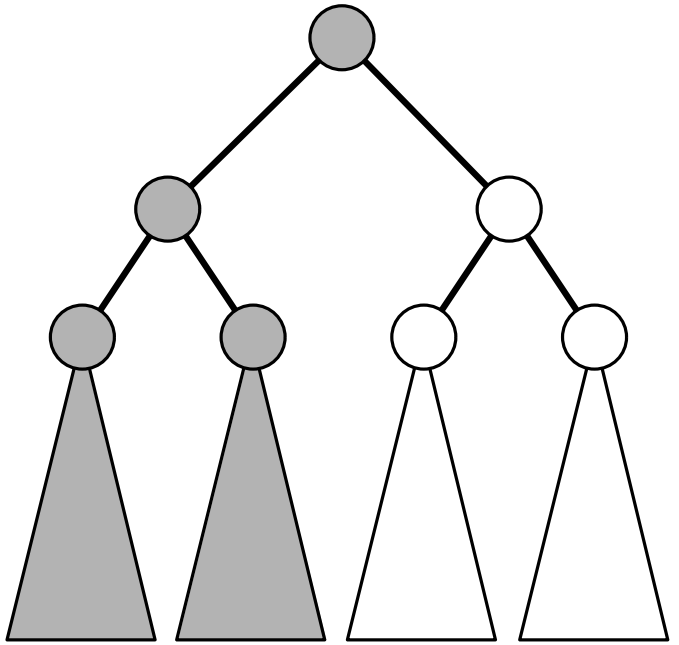
$Q(n) = 1 + Q(n/2)$

Problem?

ℓ schneidet beide Kinder des linken Kindes von root

Bereichsabfrage in einem kd-Tree





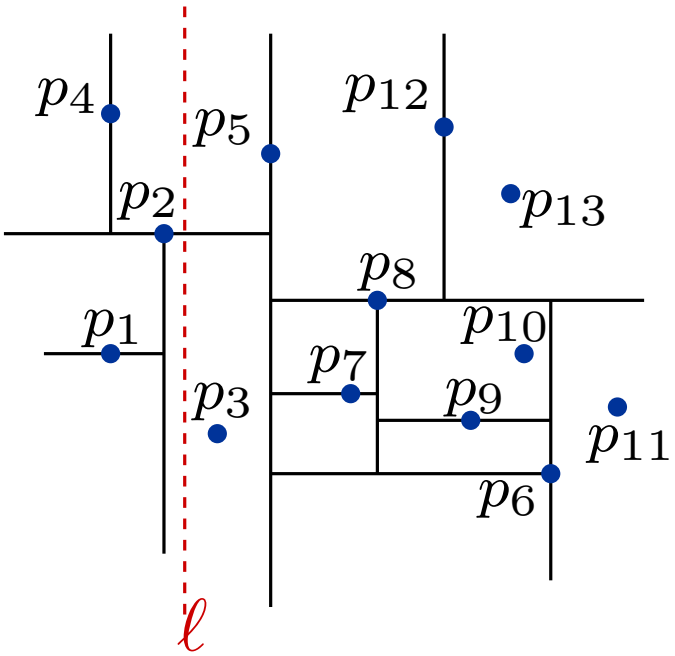
Vermutung:

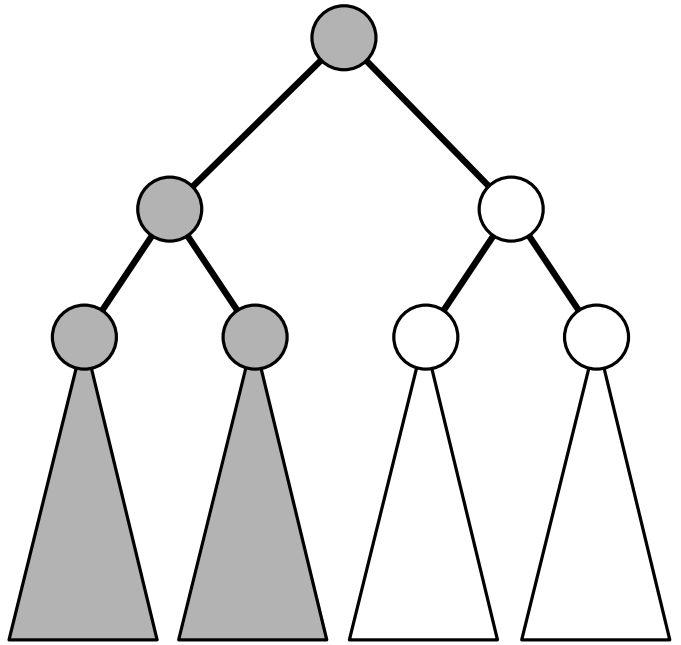
$Q(n) = 1 + Q(n/2)$

Problem?

ℓ schneidet beide Kinder des linken Kindes von root

Bereichsabfrage in einem kd-Tree





Vermutung:

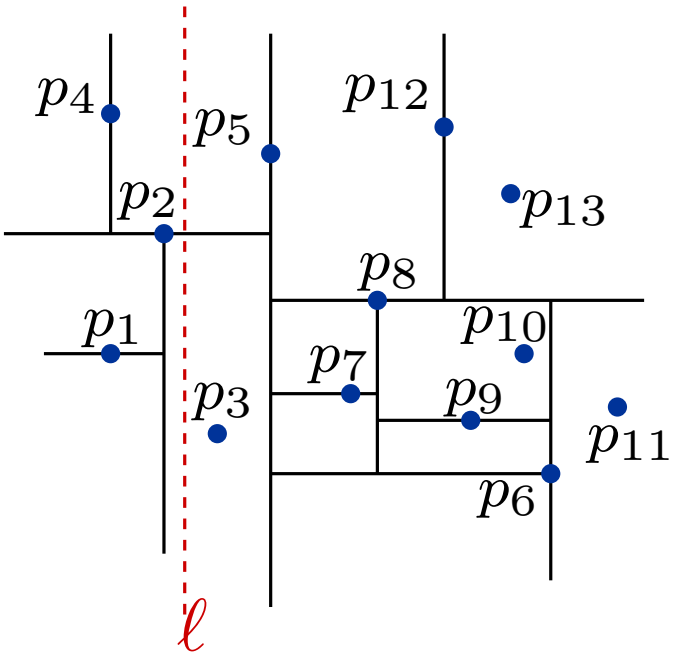
$Q(n) = 1 + Q(n/2)$

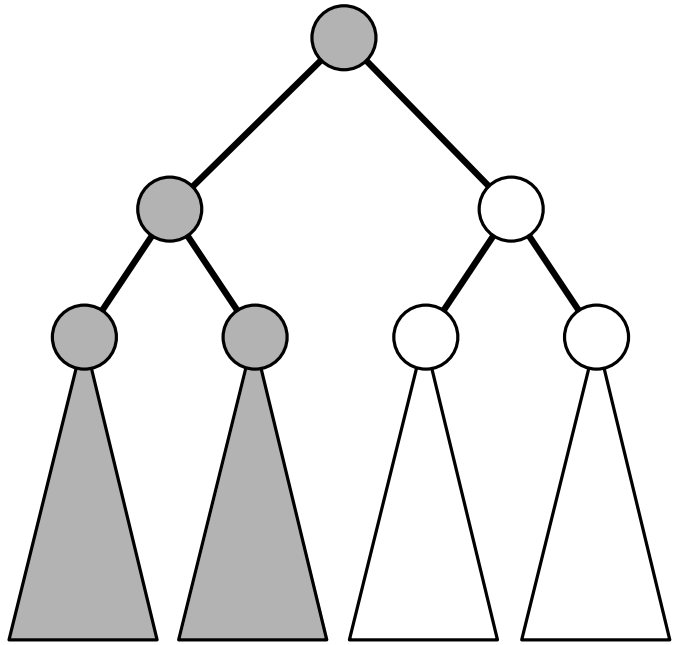
Problem?

ℓ schneidet beide Kinder des linken Kindes von root

Gleiche Rekursionssituation

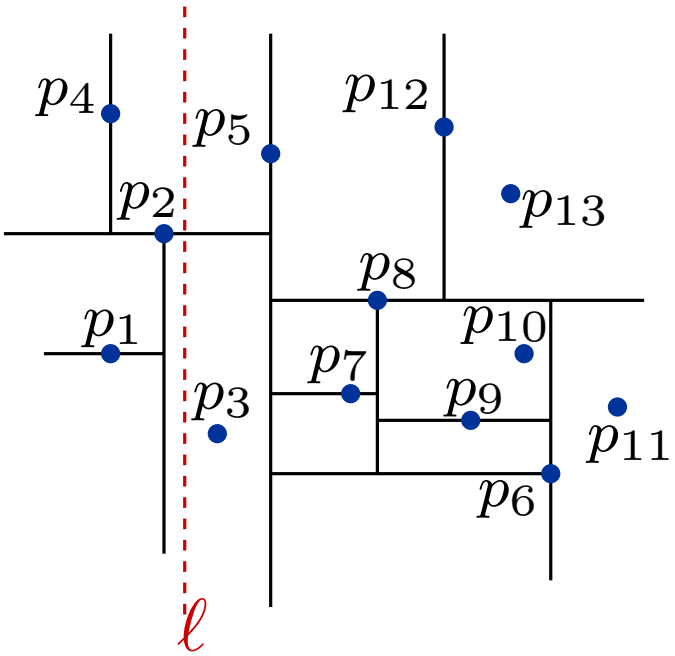
Bereichsabfrage in einem kd-Tree

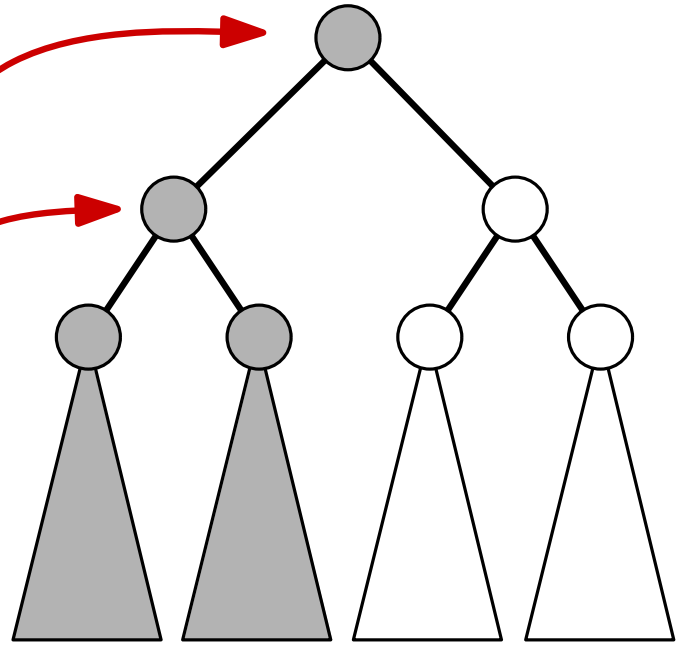




$$Q(n) = \begin{cases} \mathcal{O}(1) & , \text{ für } n = 1 \\ 2 + 2Q(n/4) & , \text{ für } n > 1 \end{cases}$$

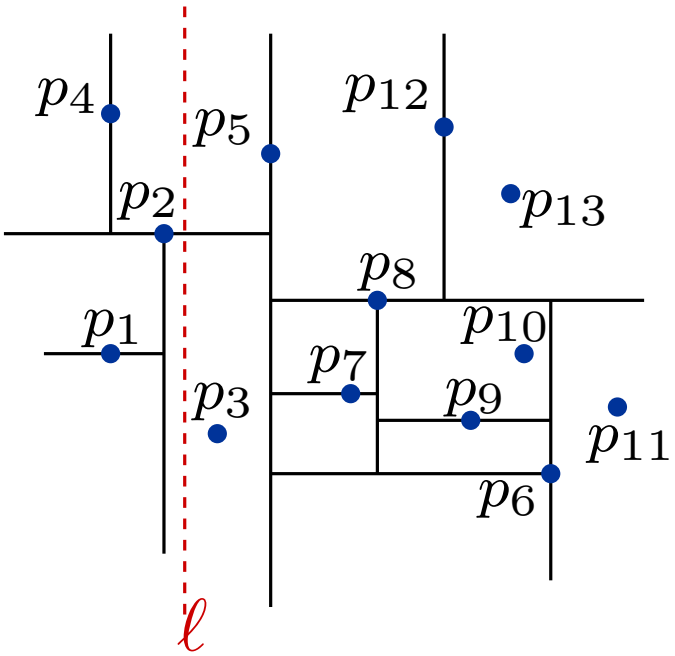
Bereichsabfrage in einem kd-Tree

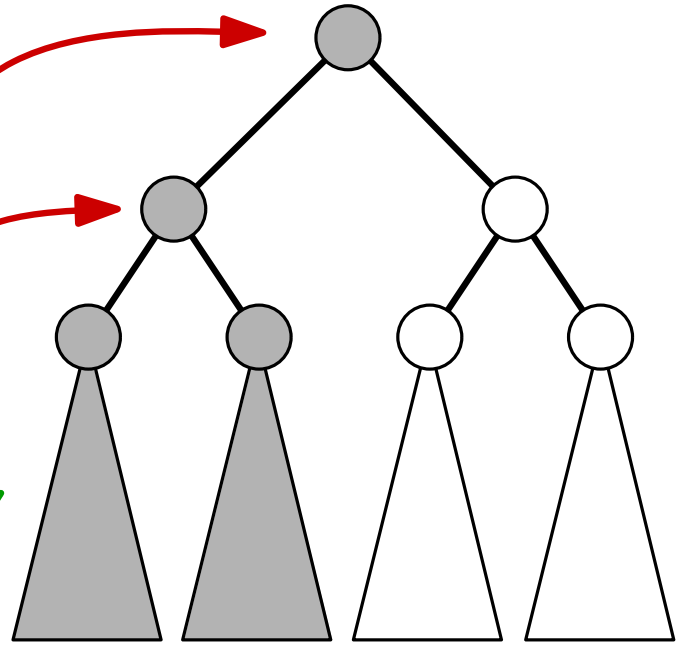




$$Q(n) = \begin{cases} \mathcal{O}(1) & , \text{ für } n = 1 \\ 2 + 2Q(n/4) & , \text{ für } n > 1 \end{cases}$$

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

kd-Trees – w-c Laufzeit

Anfragen

a) warum?

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

Aufgabe 1

Karlsruhe Institute of Technology

kd-Trees – w-c Laufzeit

Anfragen

a) warum?

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

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

Aufgabe 1

kd-Trees – w-c Laufzeit

Anfragen

a) warum?

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

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

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

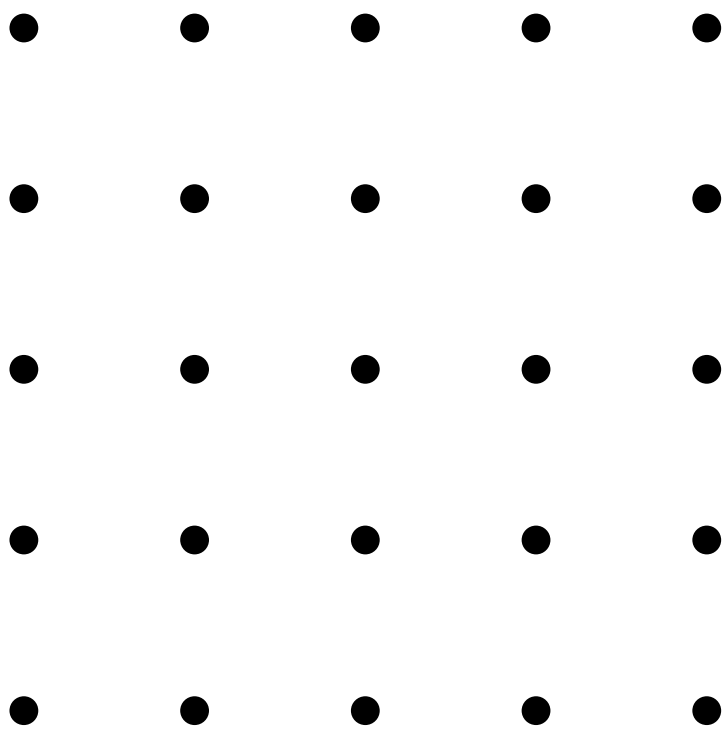
Aufgabe 1

Karlsruhe Institute of Technology

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

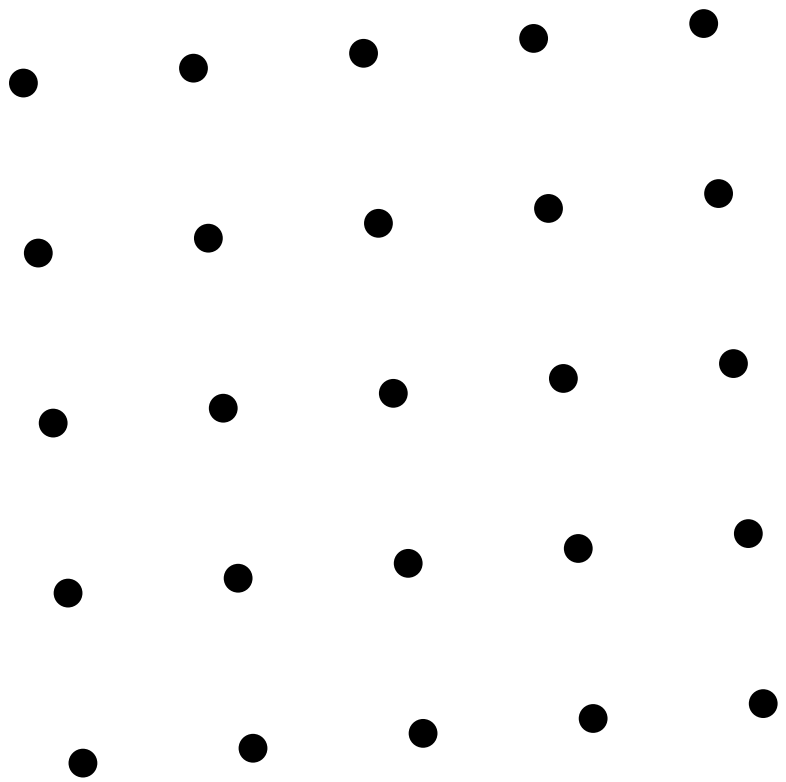
Aufgabe 1

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



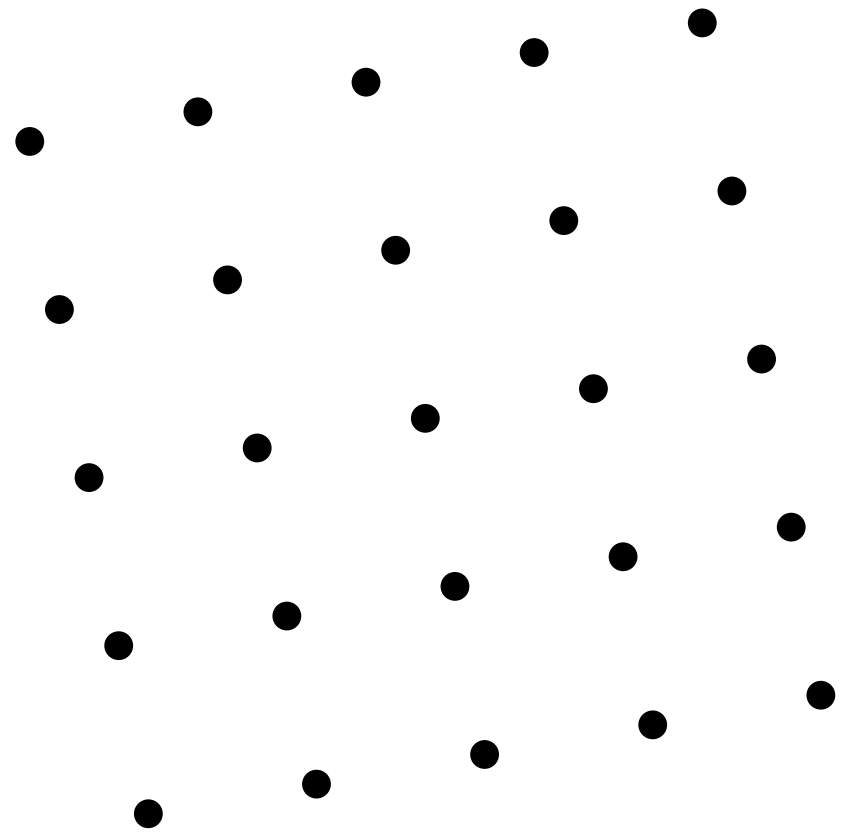
Aufgabe 1

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



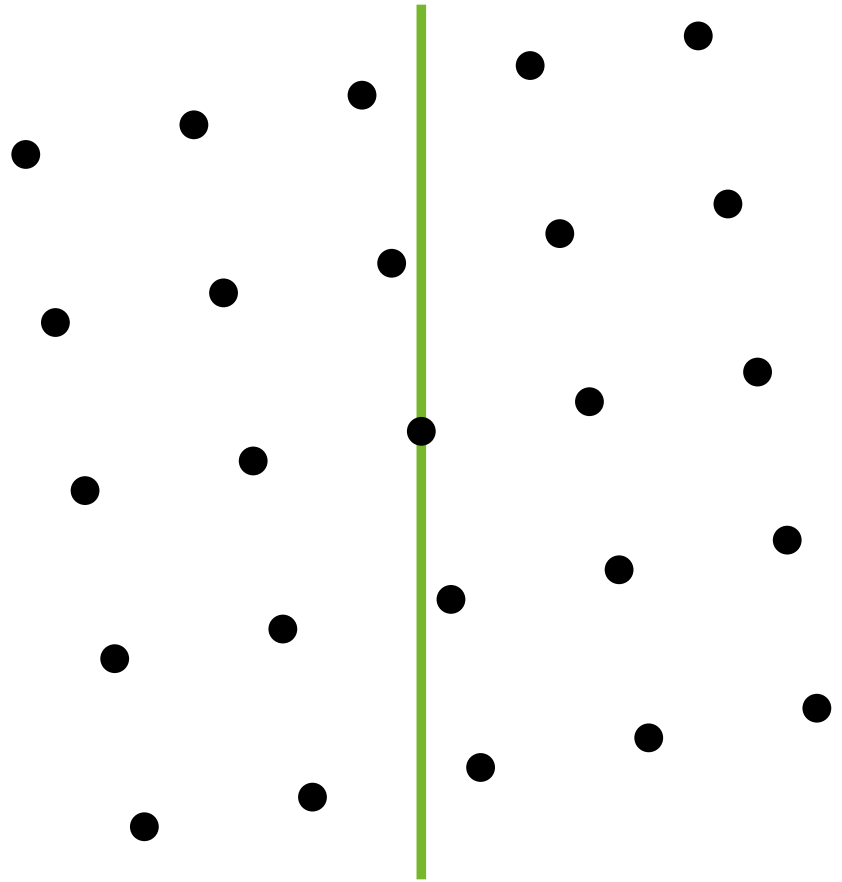
Aufgabe 1

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



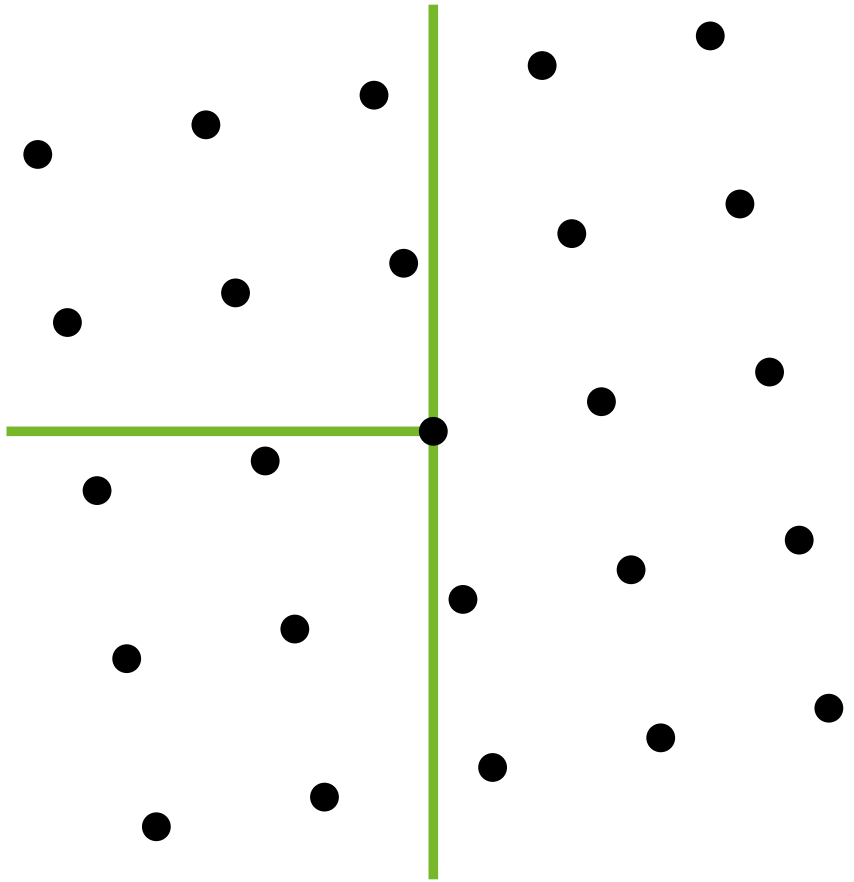
Aufgabe 1

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



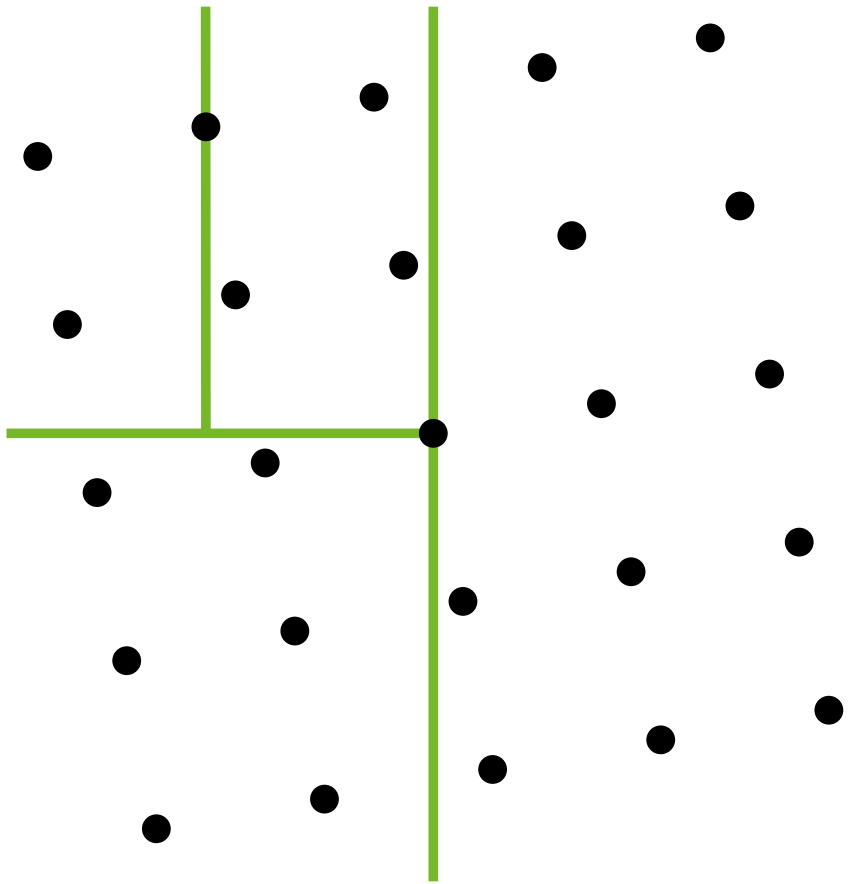
Aufgabe 1

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



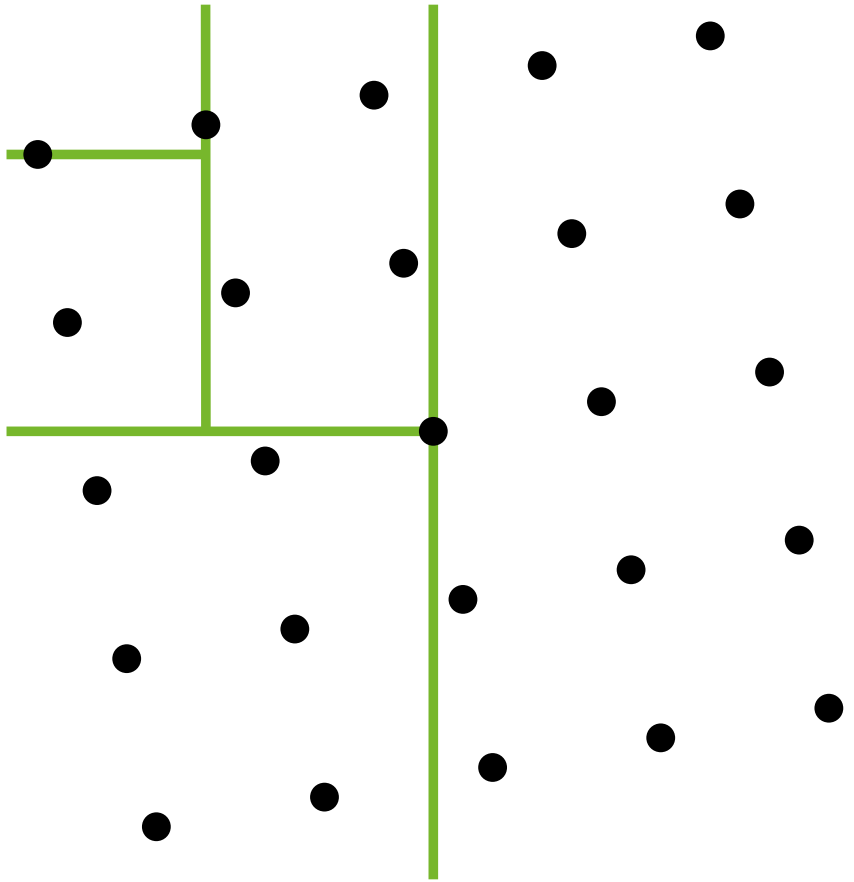
Aufgabe 1

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



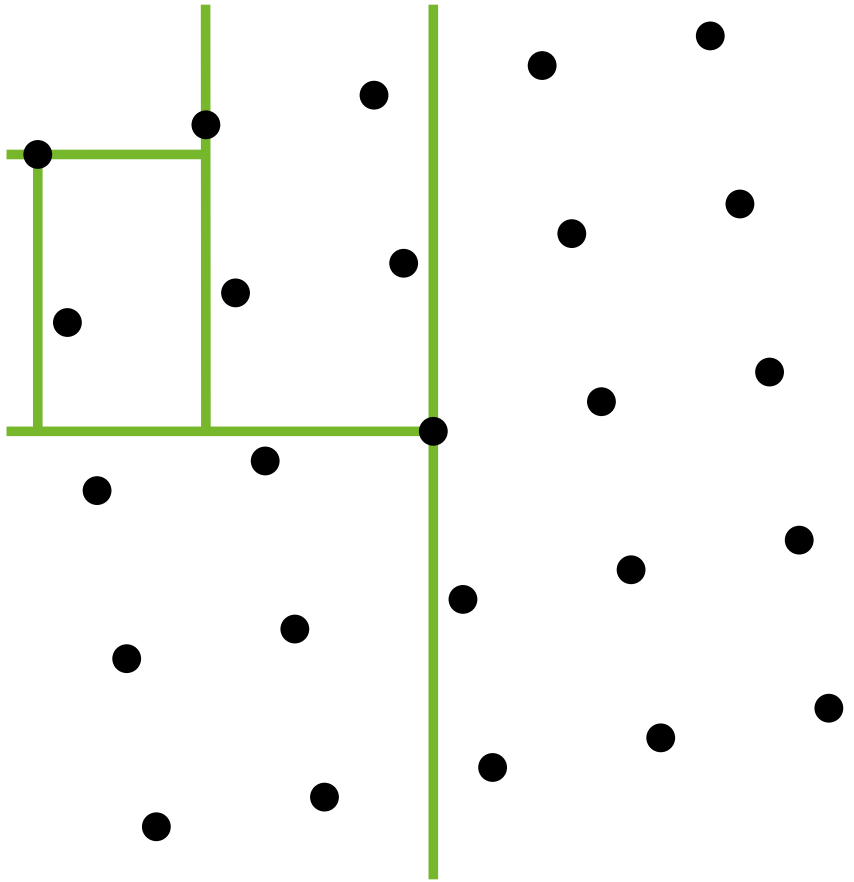
Aufgabe 1

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



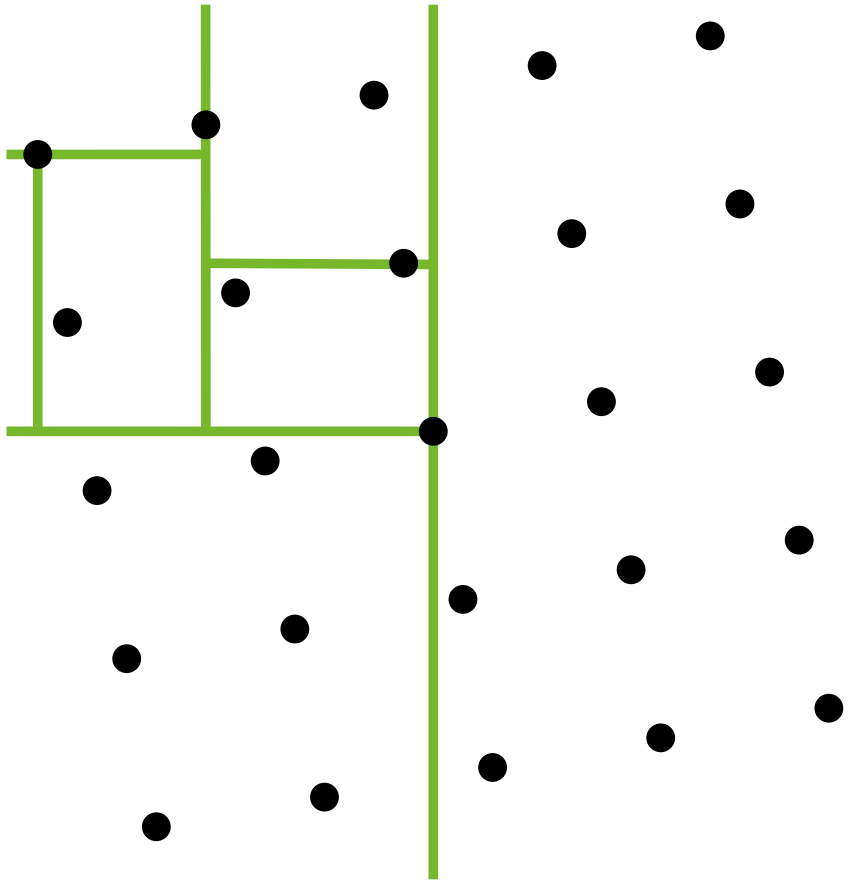
Aufgabe 1

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



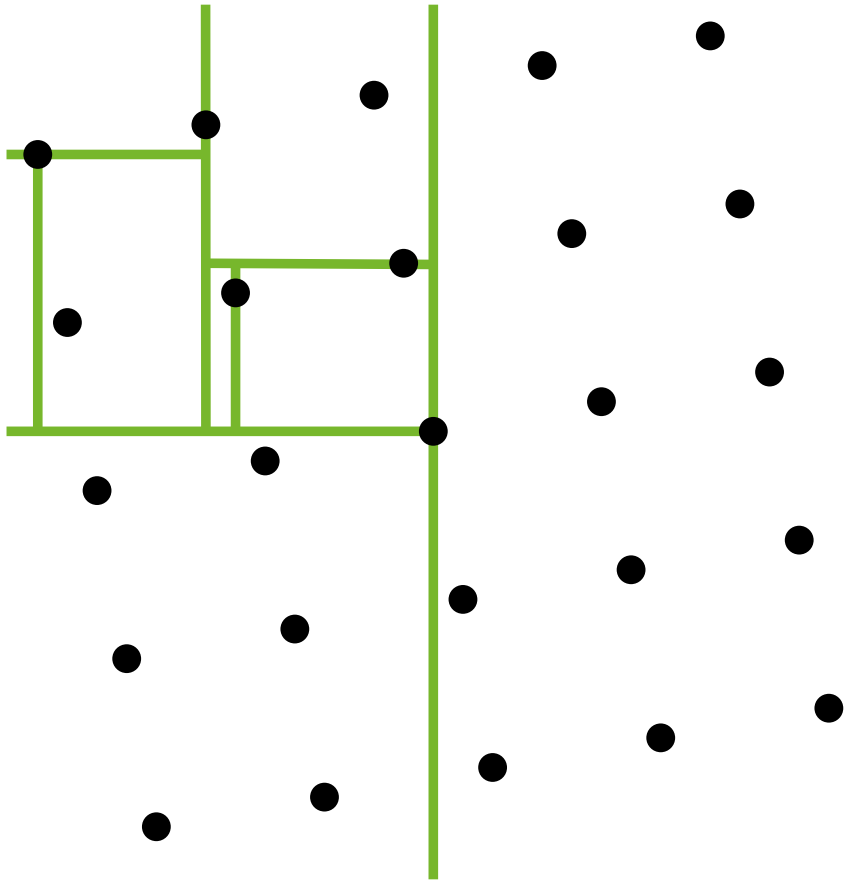
Aufgabe 1

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



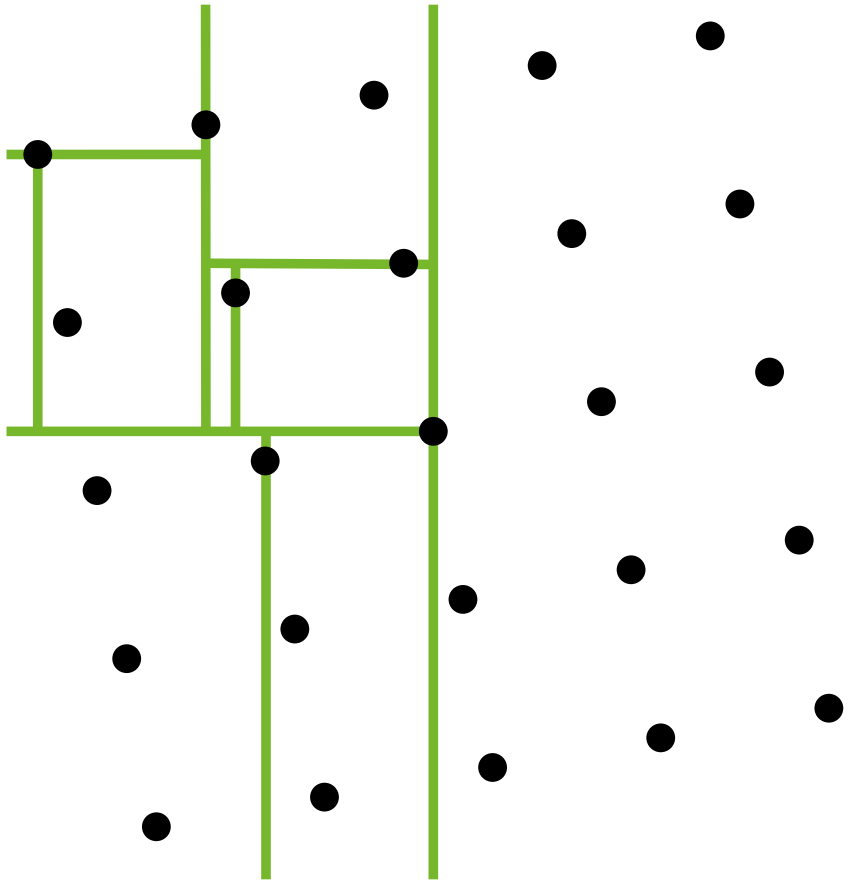
Aufgabe 1

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



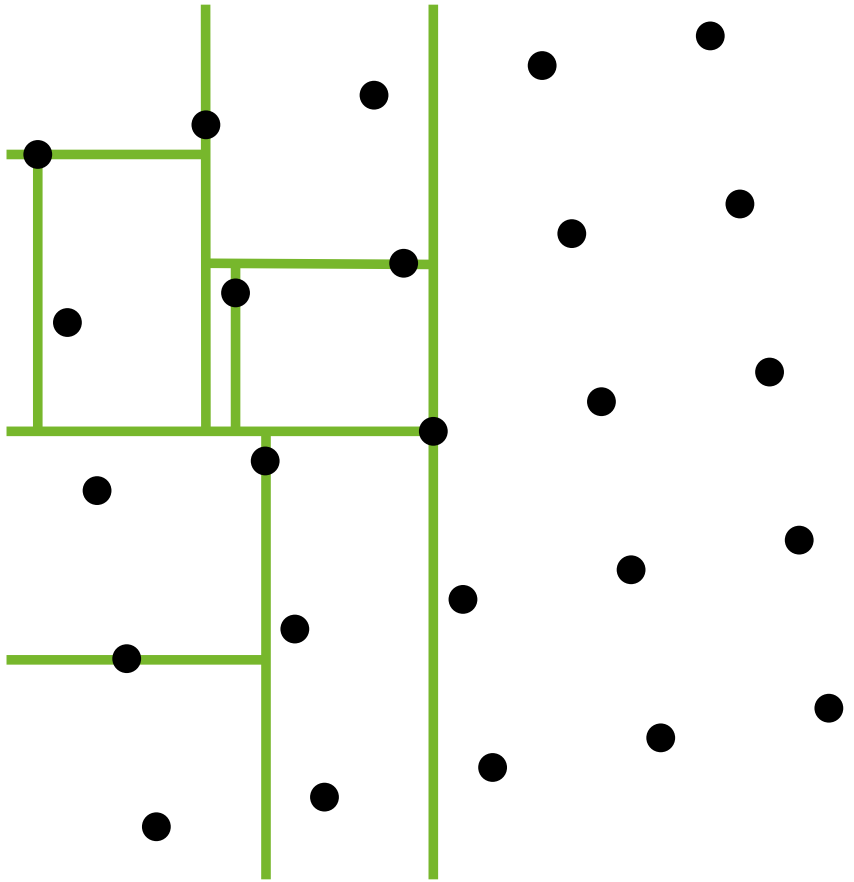
Aufgabe 1

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



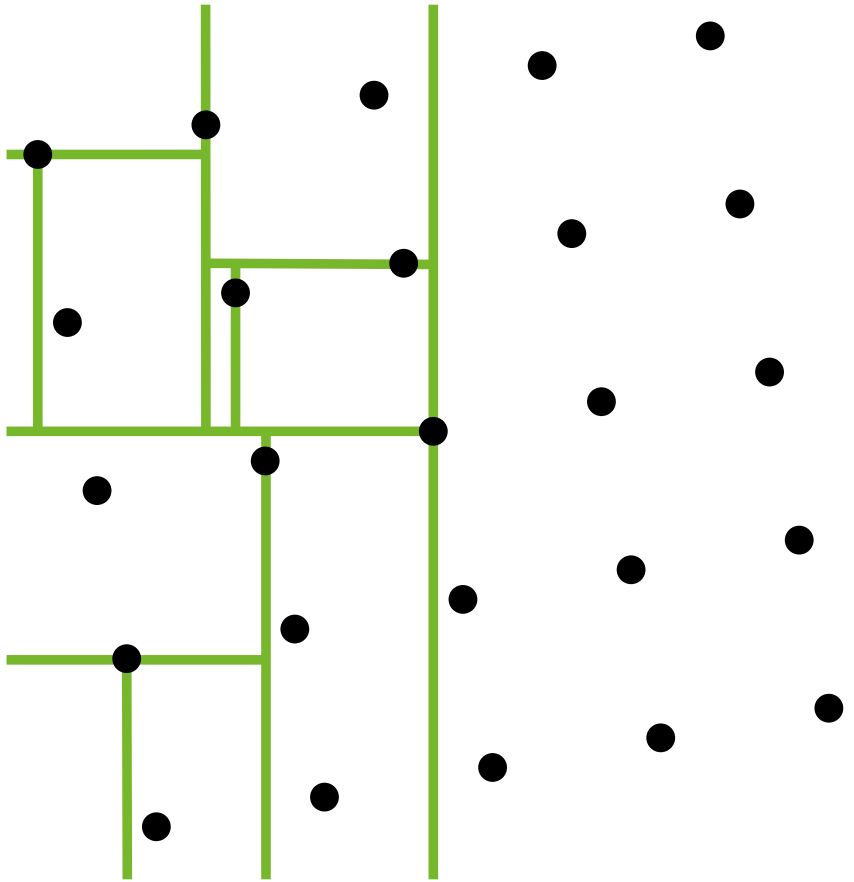
Aufgabe 1

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



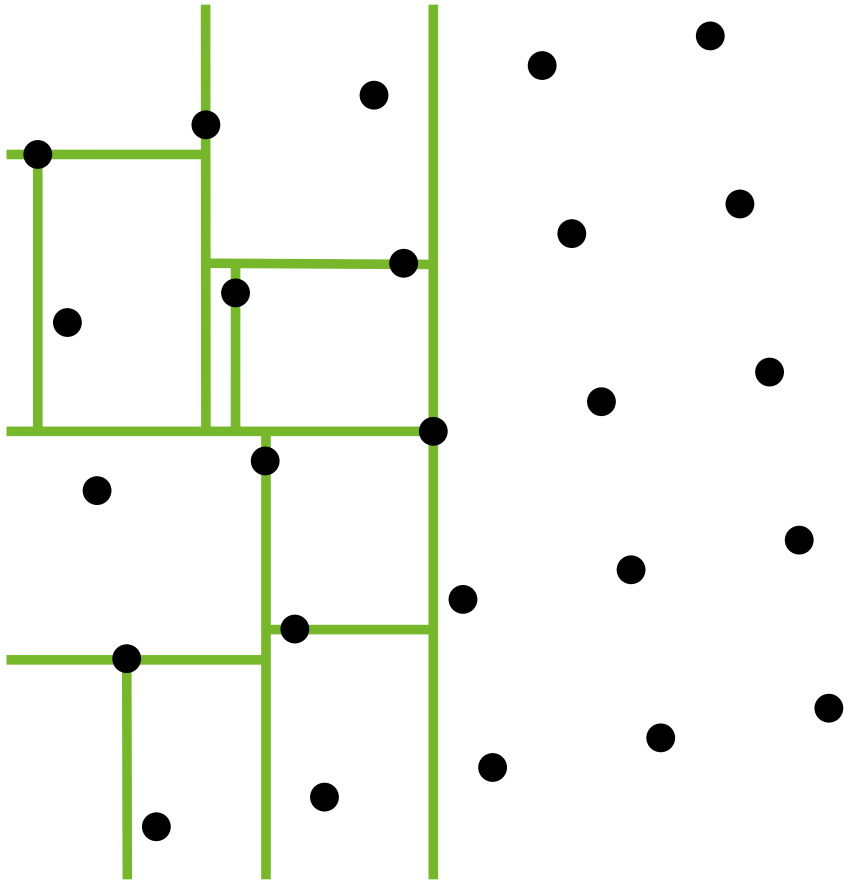
Aufgabe 1

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



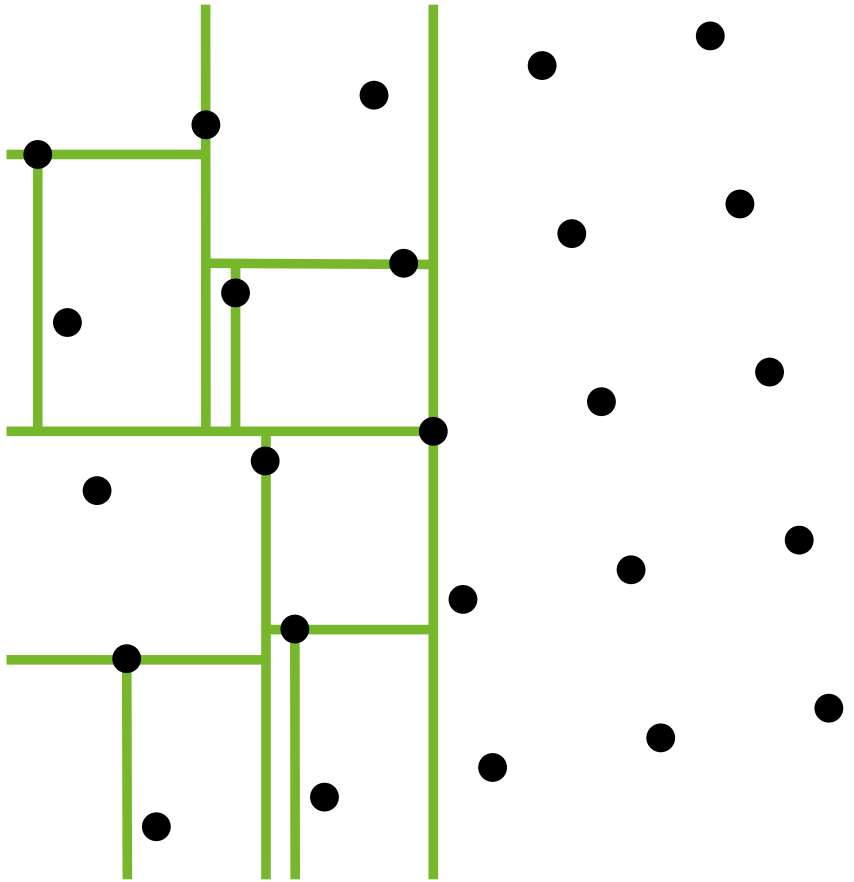
Aufgabe 1

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



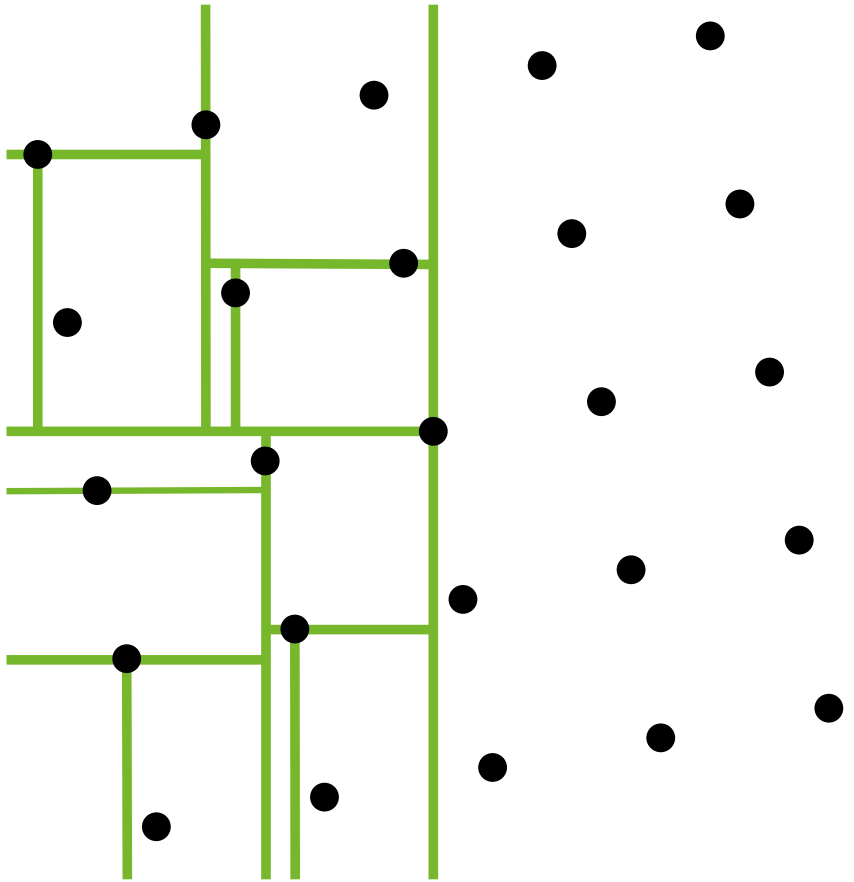
Aufgabe 1

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



Aufgabe 1

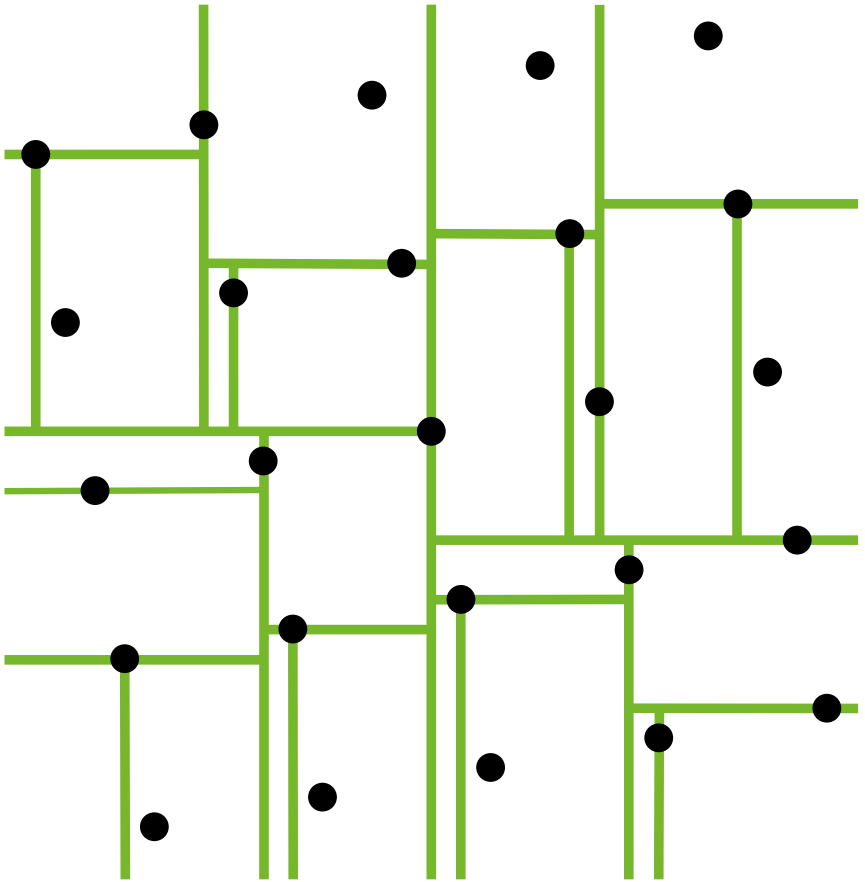
c) $\Omega(\sqrt{n})$ untere Schranke für Bereichsanfragen in kd-Trees



Aufgabe 1

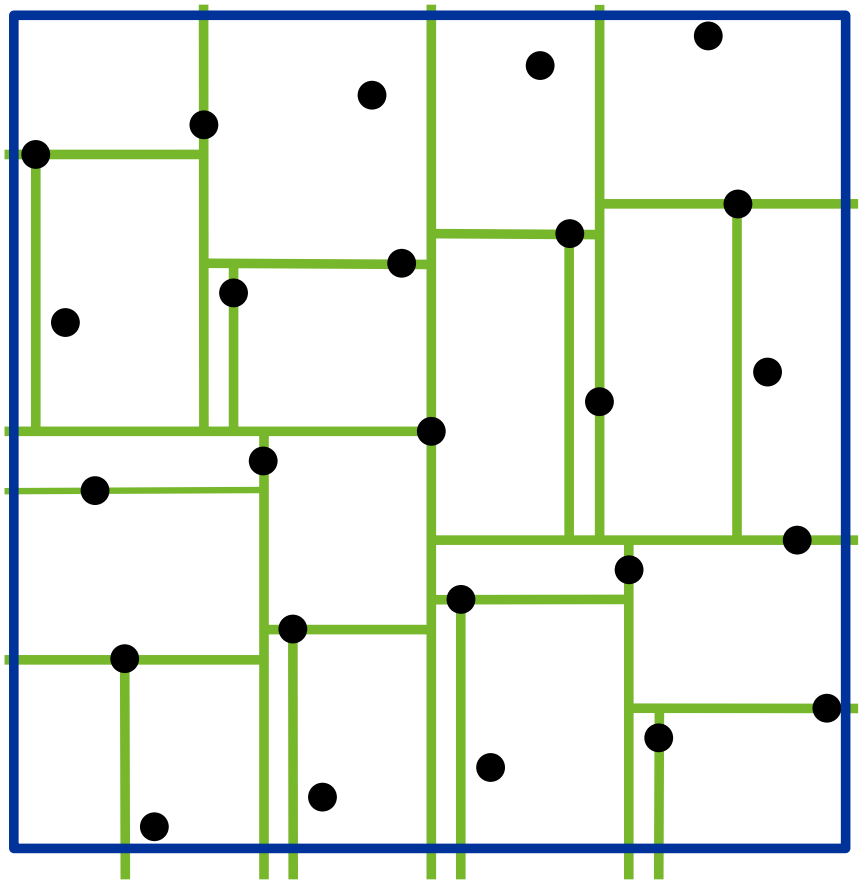
Karlsruhe Institute of Technology

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



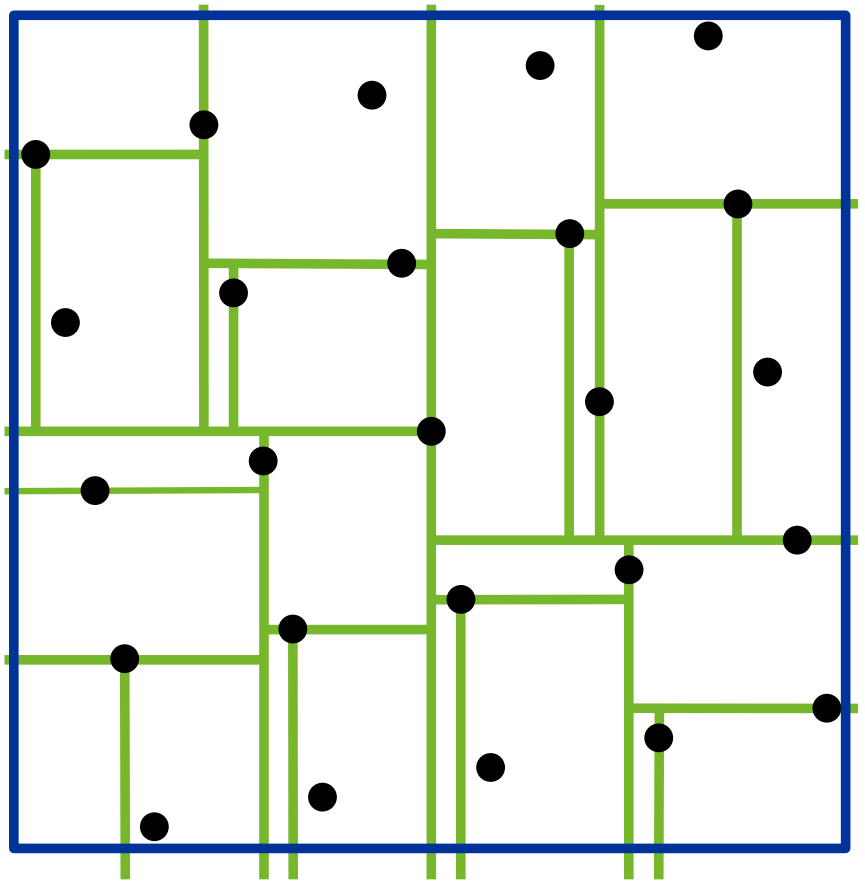
Aufgabe 1

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



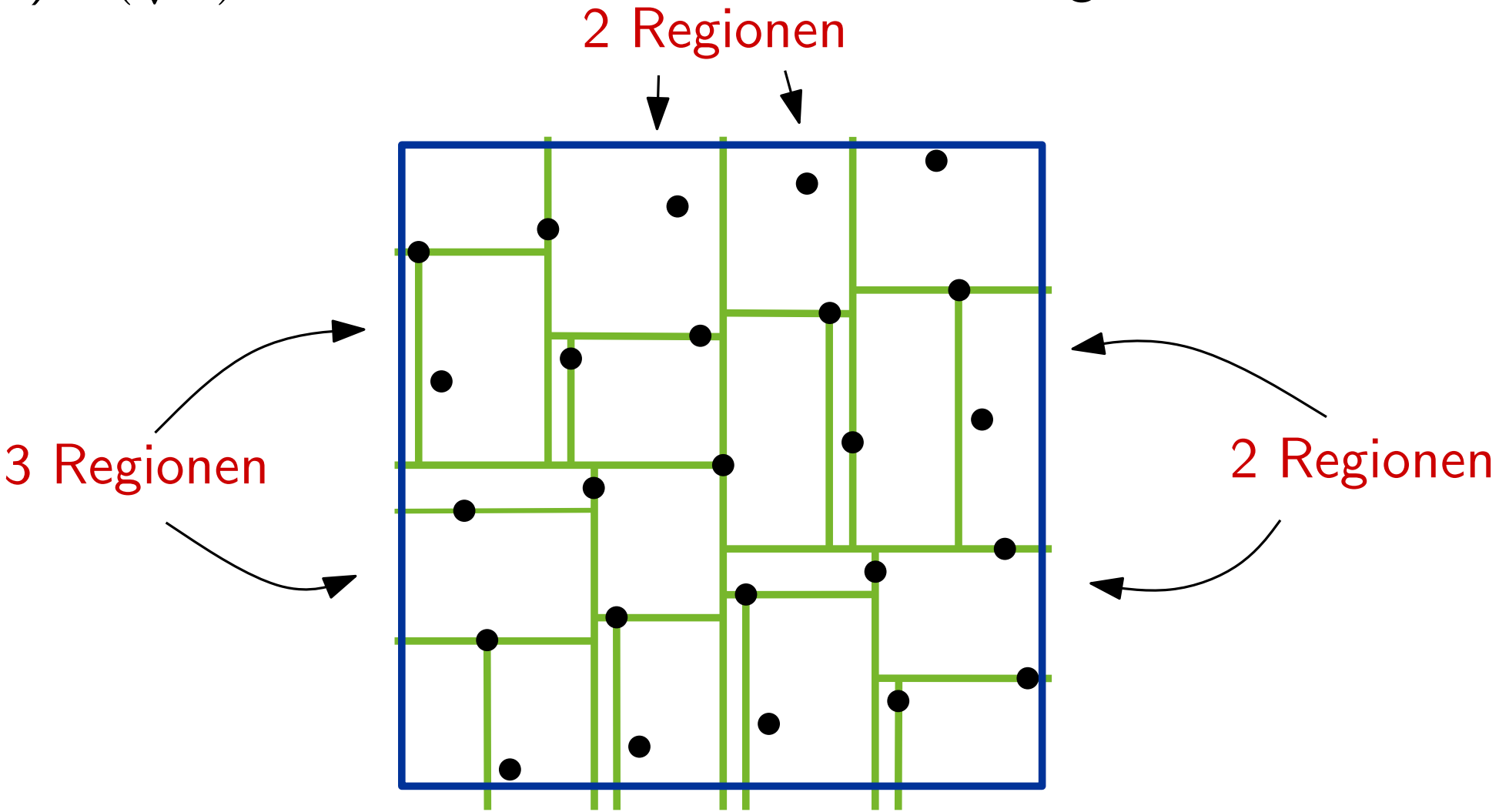
Aufgabe 1

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



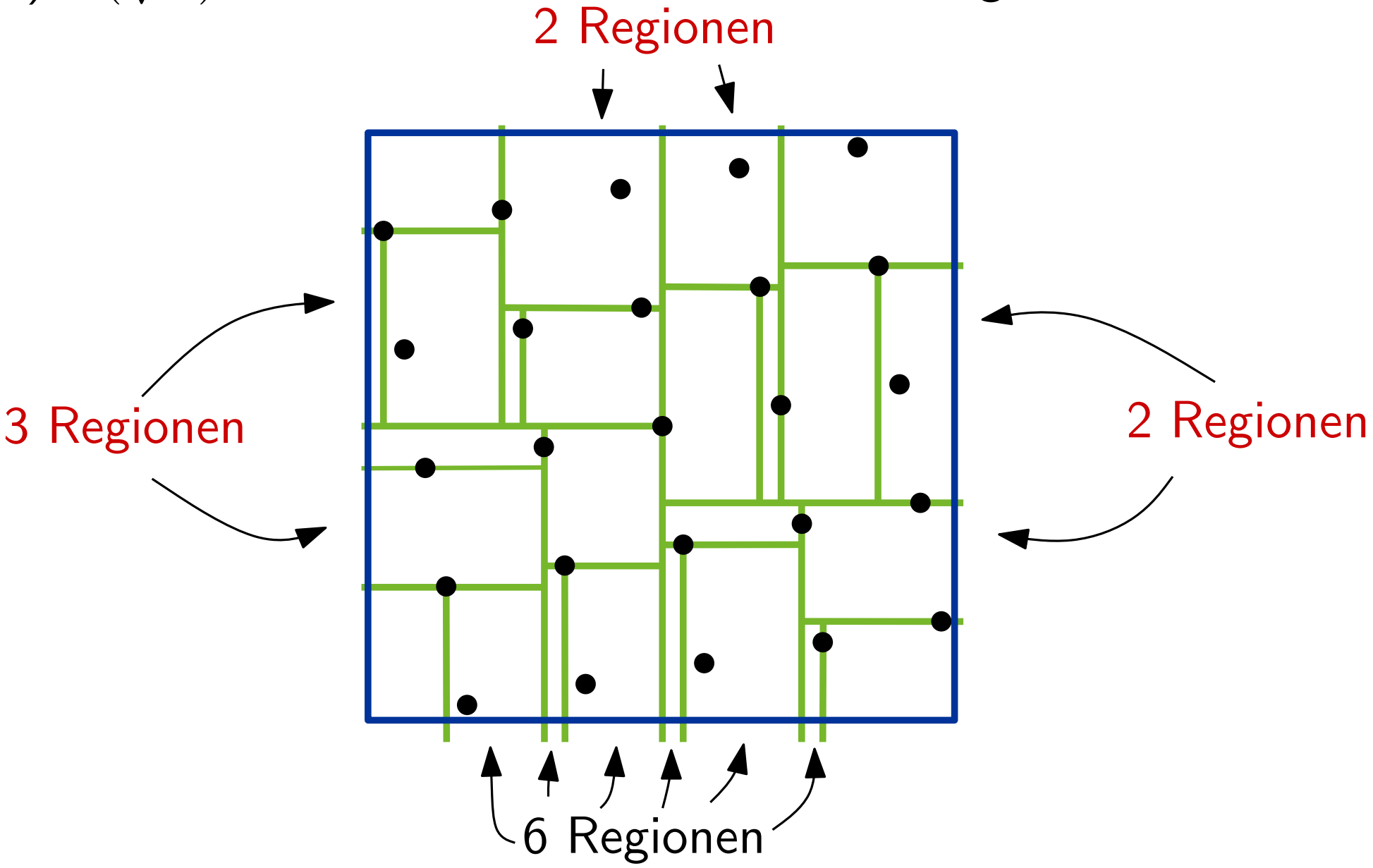
Aufgabe 1

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



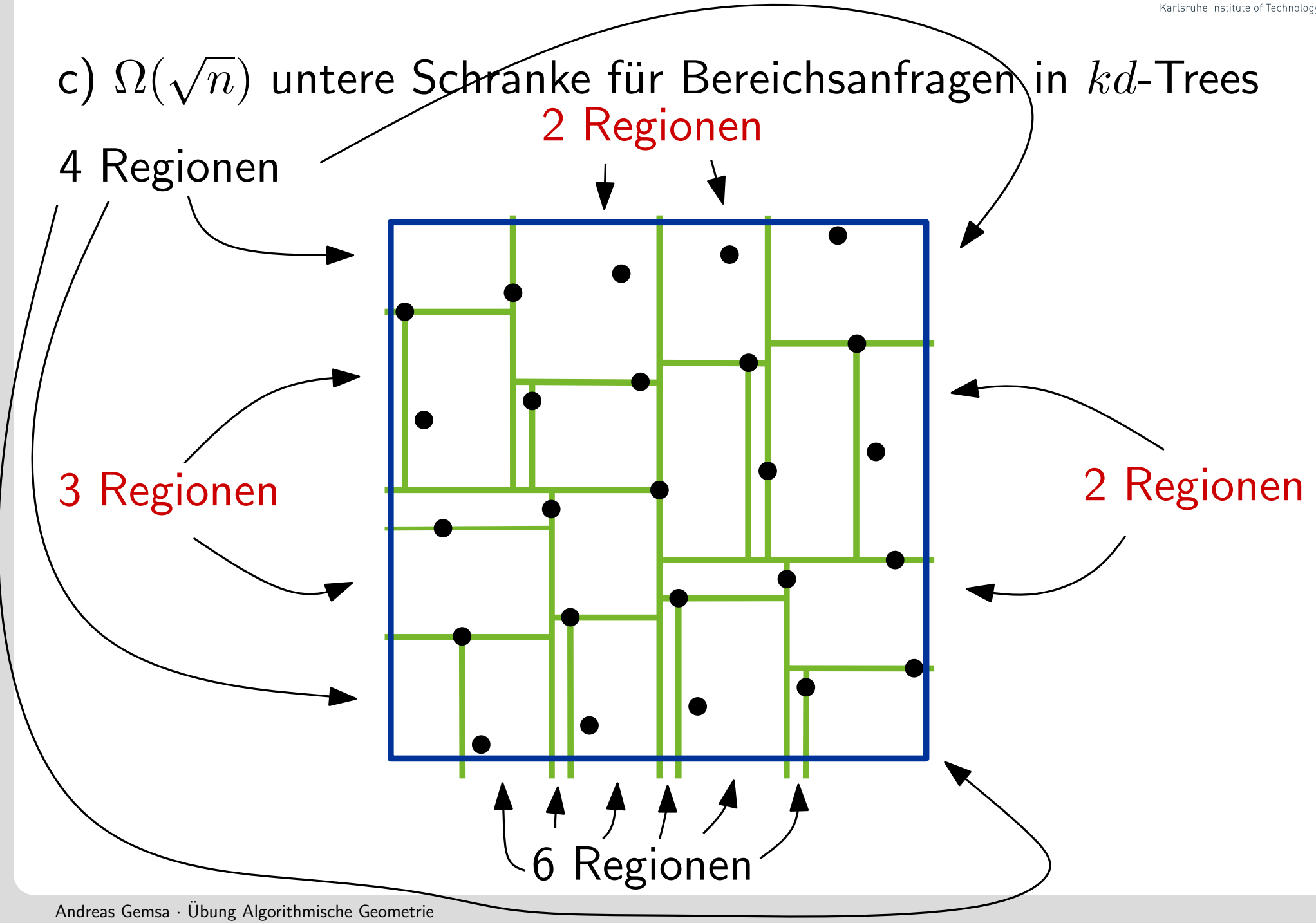
Aufgabe 1

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

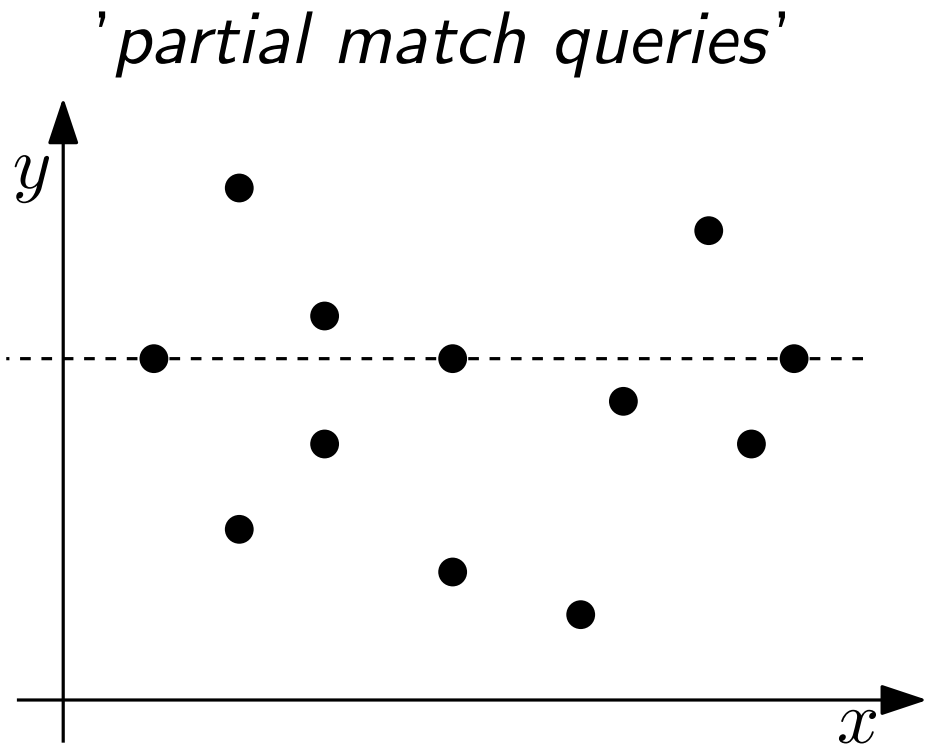


Aufgabe 1

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



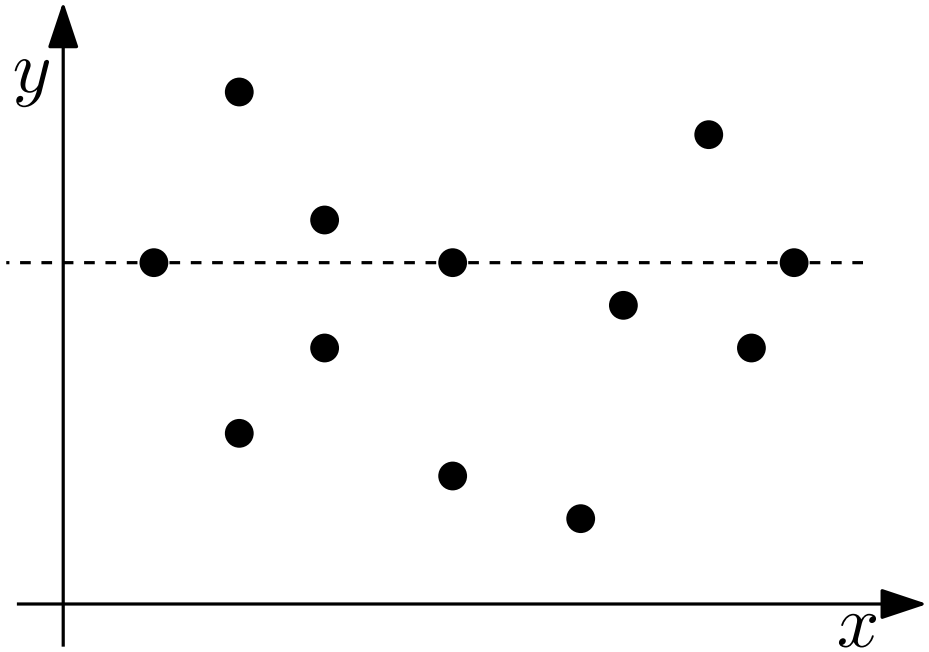
Aufgabe 2



Gebe mir alle Punkte mit $y = 7$.

Aufgabe 2

'partial match queries'

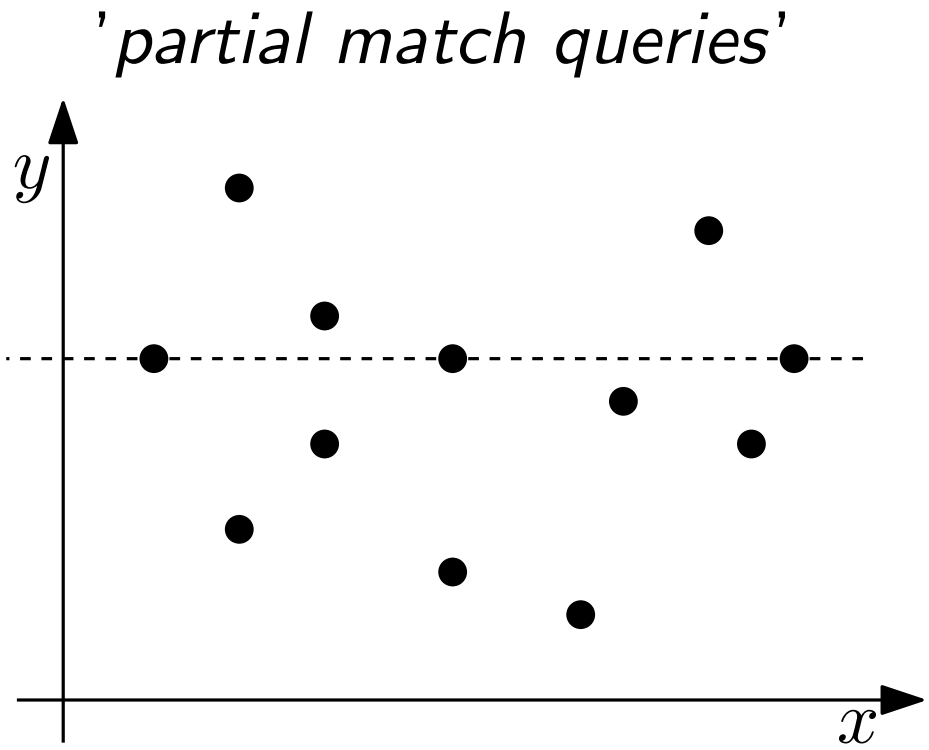


Gebe mir alle Punkte mit $y = 7$.

---

a) Wie kann man kd-Trees dafür nutzen?

Aufgabe 2

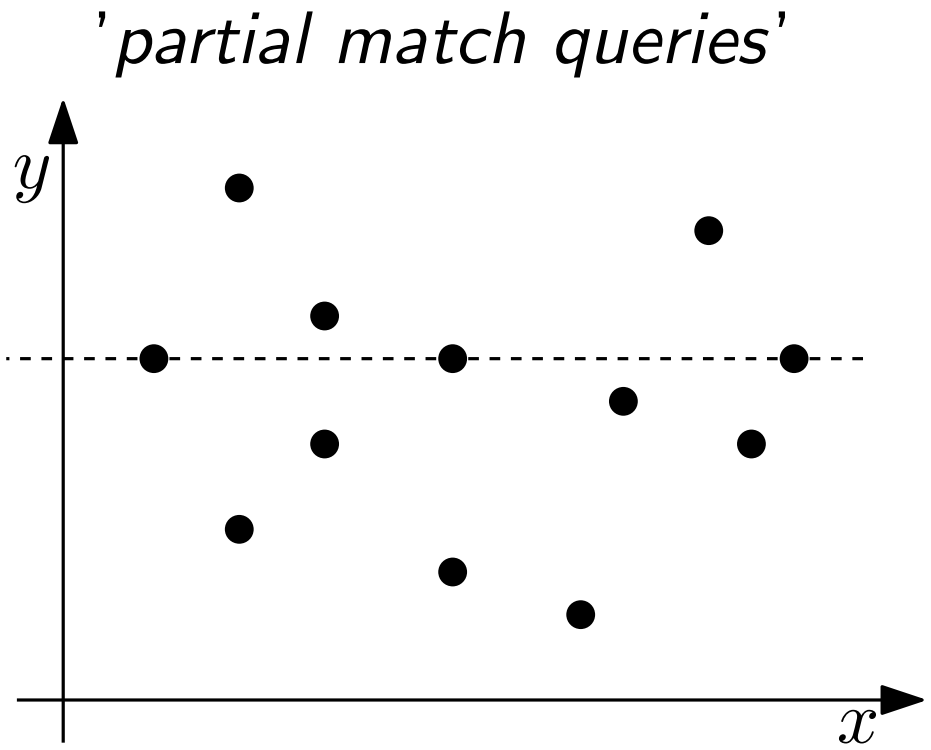


Gebe mir alle Punkte mit $y = 7$.

---

b) Wie kann man range-Trees dafür nutzen?

Aufgabe 2



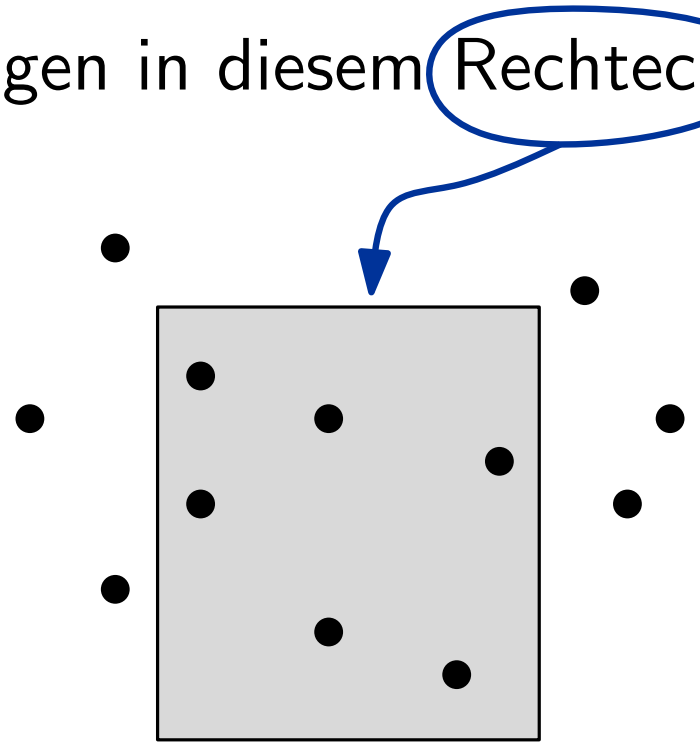
Gebe mir alle Punkte mit $y = 7$.

---

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?

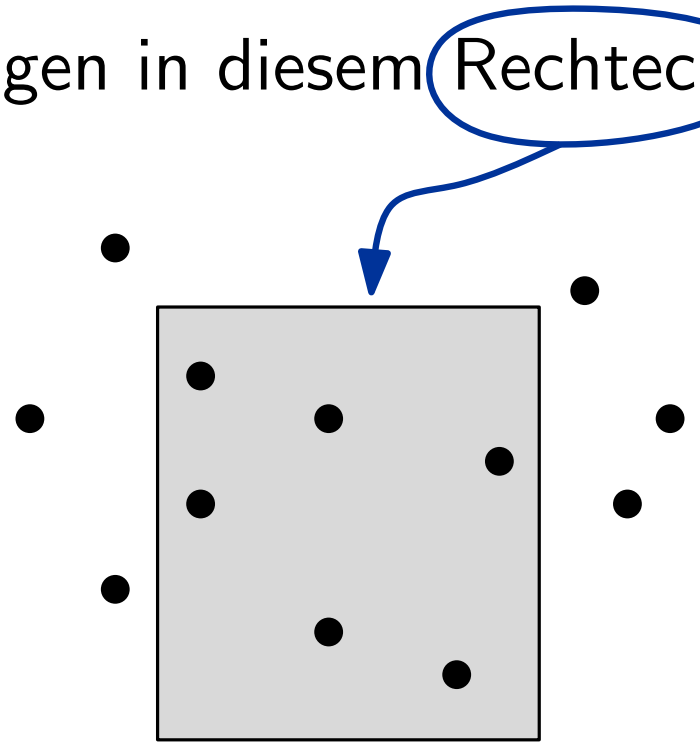


range counting query

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

Aufgabe 3

Wie viele Punkte liegen in diesem Rechteck?



range counting query

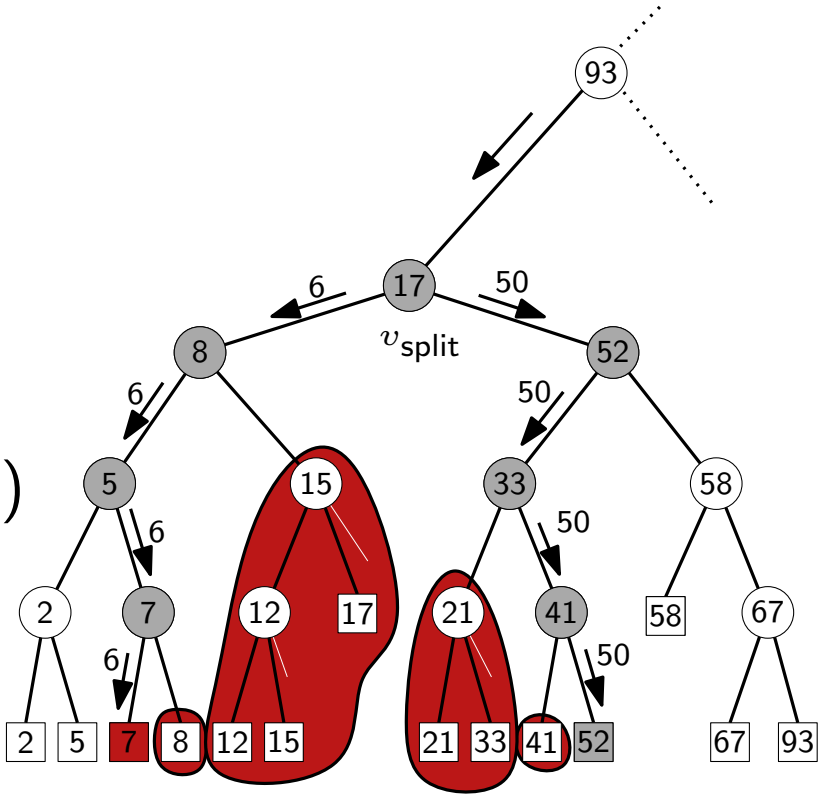
Anforderung: Additive Konstante $\mathcal{O}(k)$ in Laufzeit vermeiden

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

Aufgabe 3

1dRangeQuery(T, x, x')

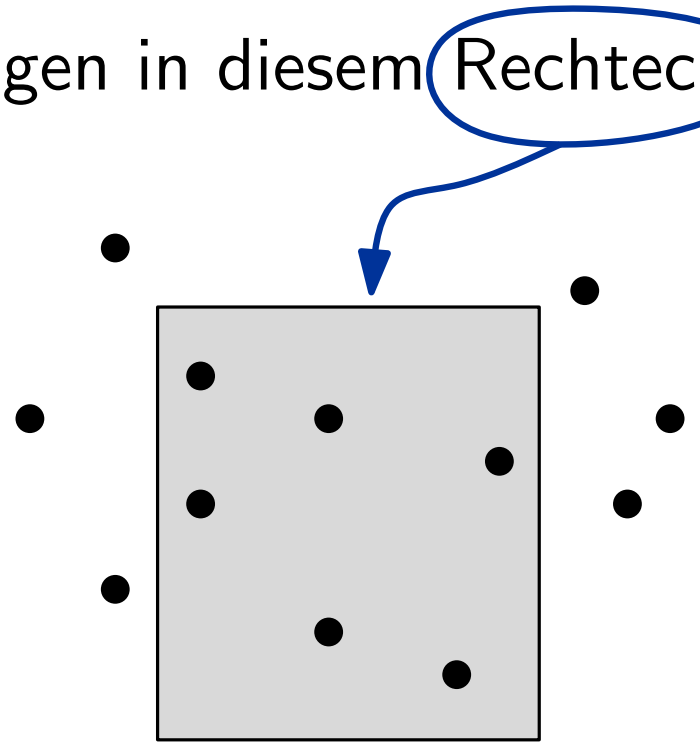
$v_{\text{split}} \leftarrow$ FindSplitNode(T, x, x')
if v_{split} ist Blatt **then** prüfe v_{split}
else
  $v \leftarrow \text{lc}(v_{\text{split}})$
  while v kein Blatt **do**
    if $x \leq x_v$ **then**
      ReportSubtree(rc(v)); $v \leftarrow \text{lc}(v)$
    else $v \leftarrow \text{rc}(v)$
  prüfe v
  // analog für x' und $\texttt{rc}(v_{\text{split}})$



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?



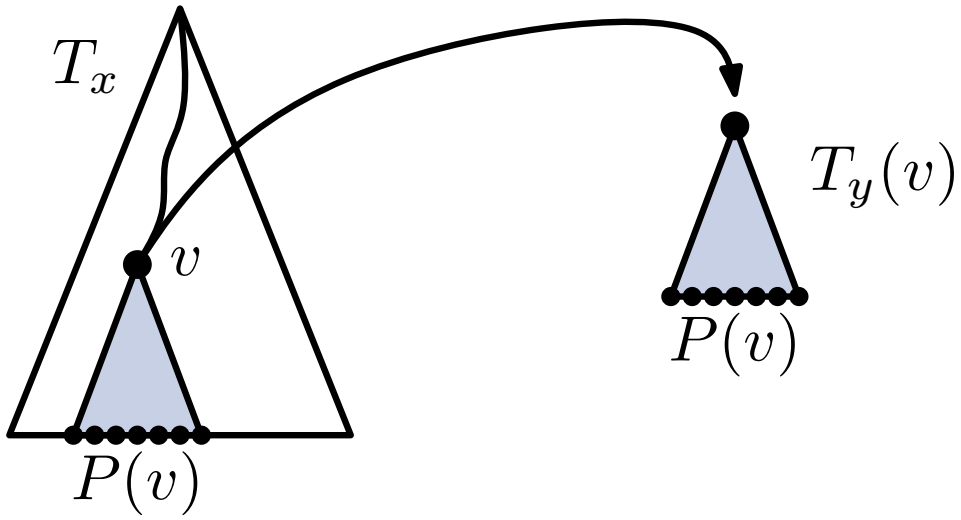
range counting query

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

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

Aufgabe 3

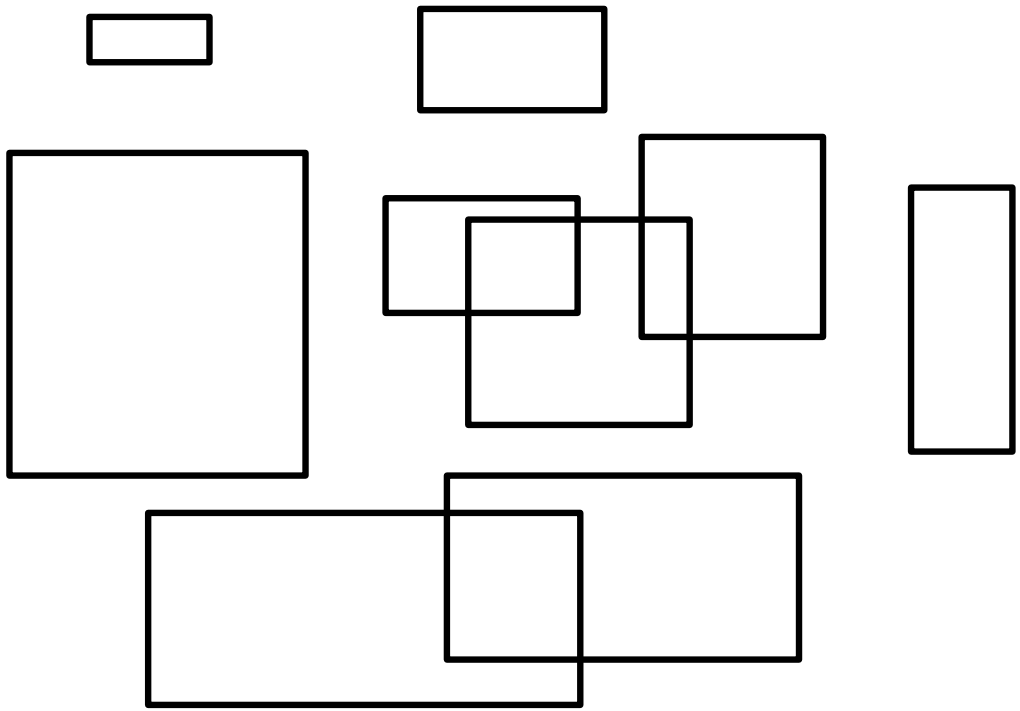
2dim Range-Trees:



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

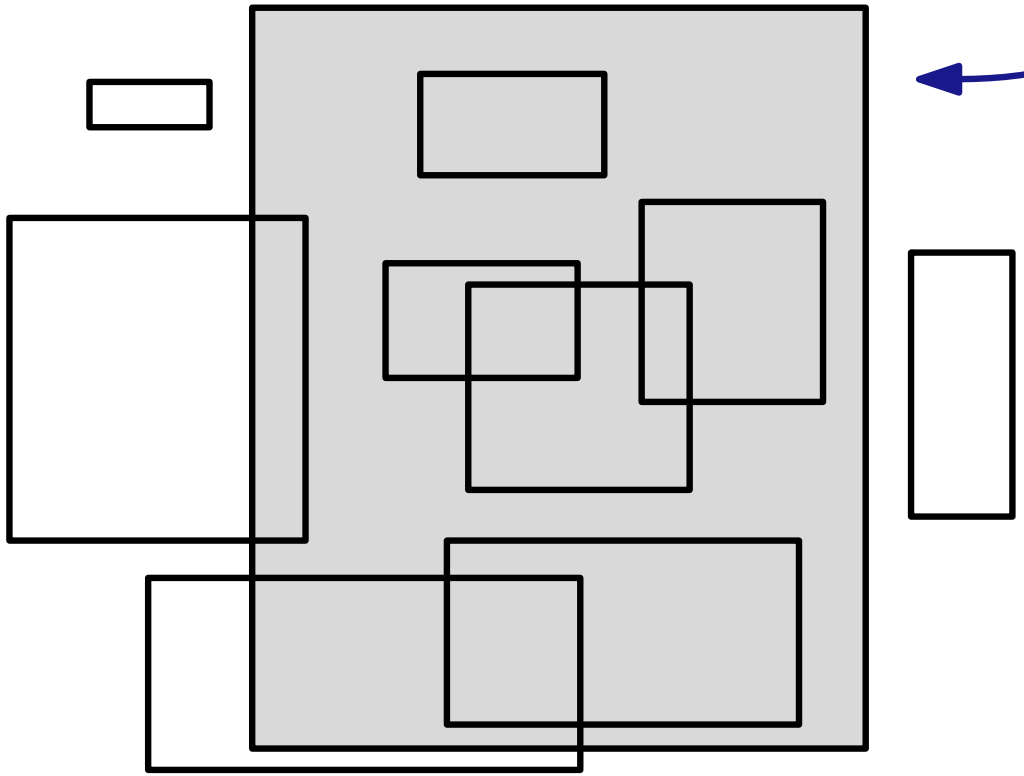
b) Benutze Lösung aus a) um das d-dimensionale Problem zu lösen

Aufgabe 4



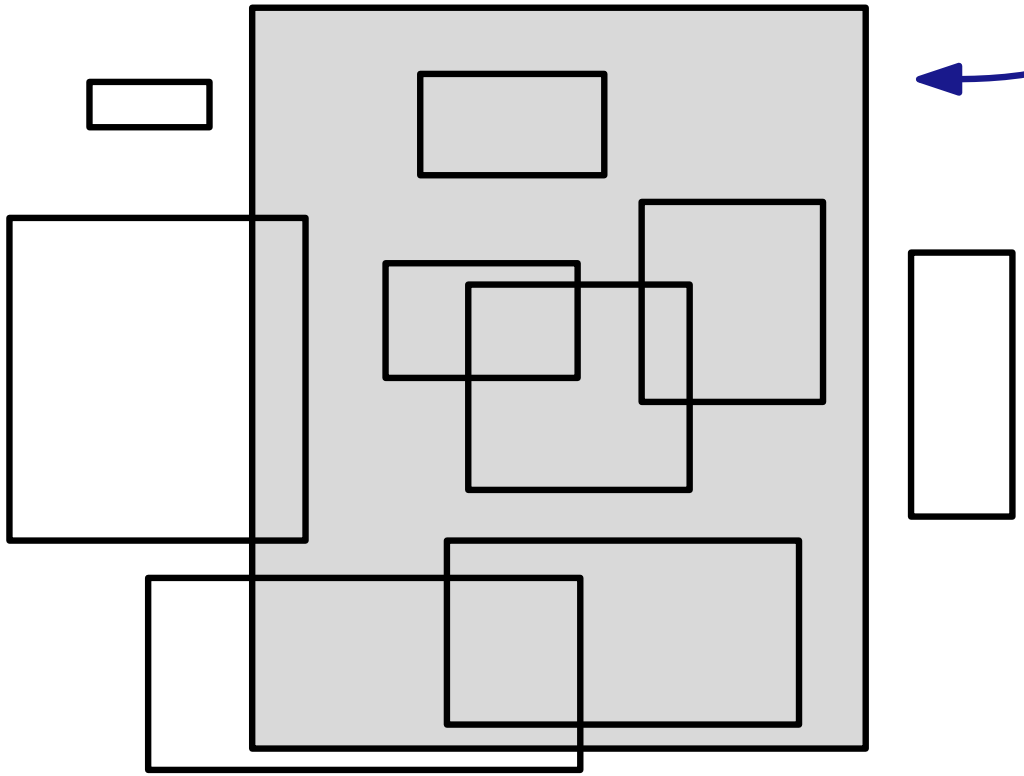
Aufgabe 4

Welche Rechtecke liegen vollständig in diesem Rechteck?



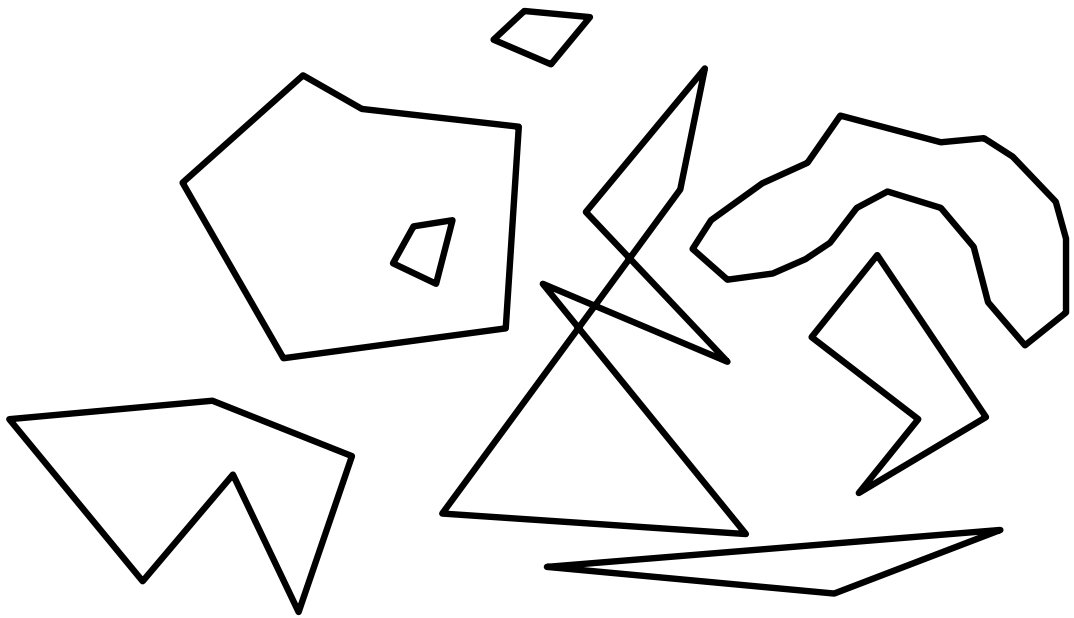
Aufgabe 4

Welche Rechtecke liegen vollständig in diesem Rechteck?



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

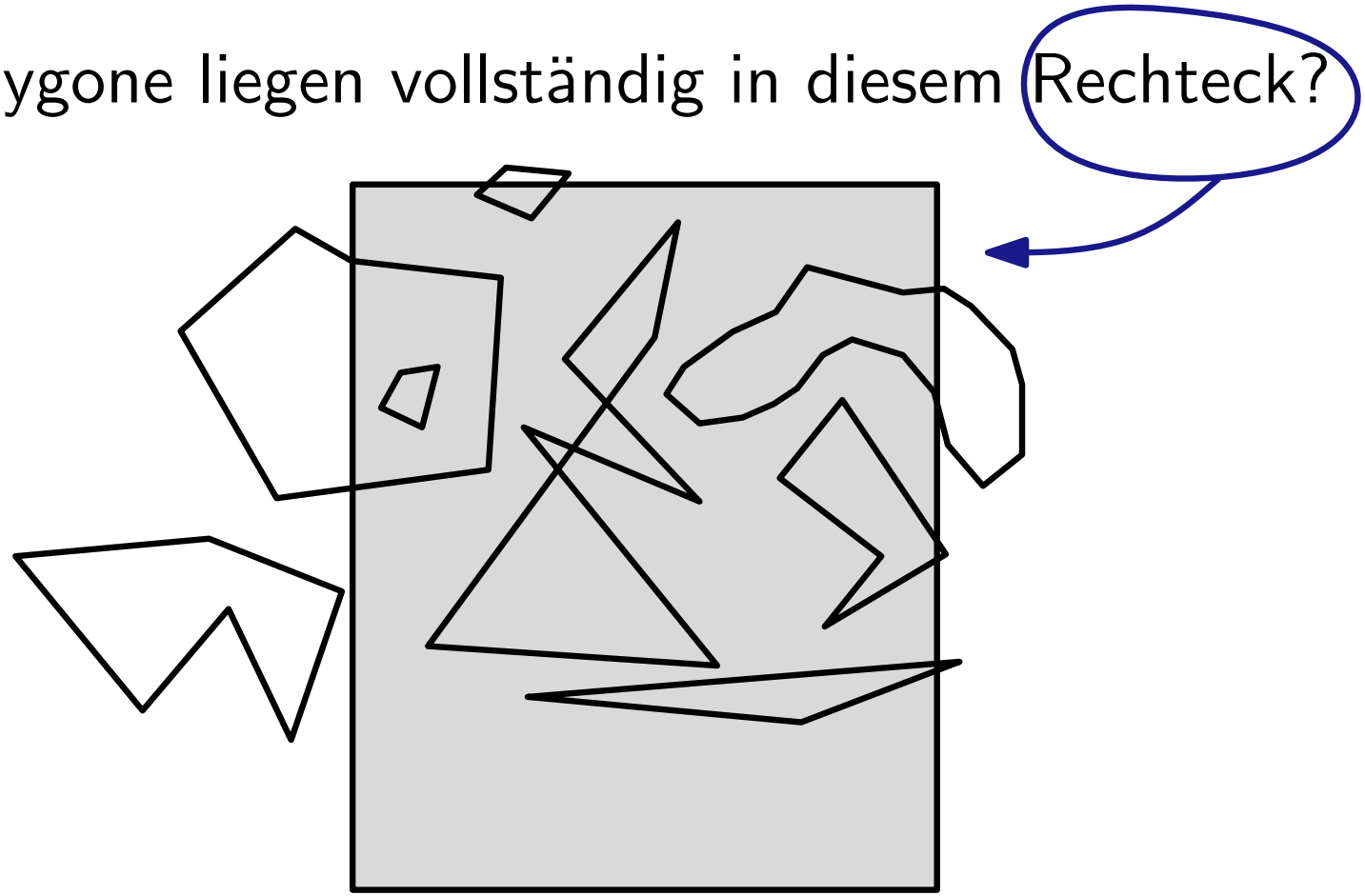
Aufgabe 4



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.

Das war's!

Karlsruhe Institute of Technology

Achtung!

Nächster Termin:

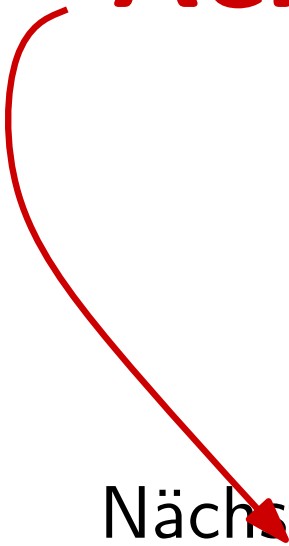
Donnertag, 09.06, 09:45 Uhr

Raum 131, Gebäude 50.34

Das war's!

Karlsruhe Institute of Technology

Achtung!



Nächster Termin:

Donnertag, 09.06. 09:45 Uhr

Raum 131, Gebäude 50.34

Das war's!

Karlsruhe Institute of Technology

Nächster Termin:

Donnertag, 09.06. 09:45 Uhr

Raum 131, Gebäude 50.34

Das war's!

Karlsruhe Institute of Technology

Nächster Termin:

Donnertag, 09.06. 09:45 Uhr

Raum 131, Gebäude 50.34