LMU LUDWIG-MAXIMILIANS-UNIVERSITY MUNICH
DEPARTMENT INSTITUTE FOR INFORMATICS
DATABASE SYSTEMS GROUP

# Kapitel 4: Räumliche Indexstrukturen

Skript zur Vorlesung

Geo-Informationssysteme

Wintersemester 2014/15

Ludwig-Maximilians-Universität München



# 4. Räumliche Indexstrukturen

# 4.1 Einführung (I)



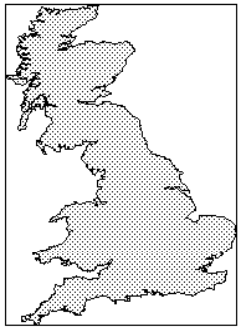
## Grundlegende Ideen

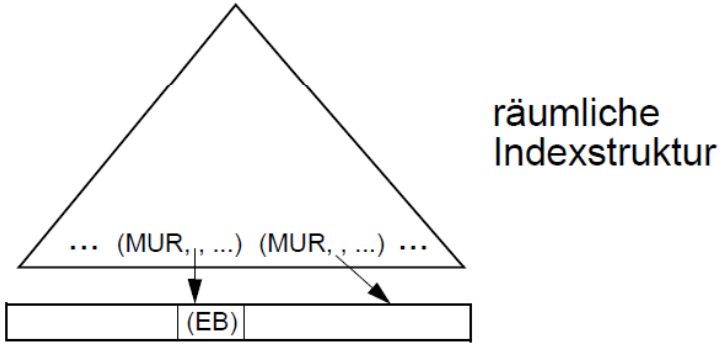
- konventionelle Zugriffsstrukturen sind für die Verwaltung von geometrischen Daten schlecht geeignet

  ⇒ *räumliche Indexstrukturen*, die die Ordnung des multidimensionalen Datenraums auf dem Sekundärspeicher möglichst gut erhalten
- Geo-Objekte variieren stark in ihrer Größe, sind aber auf Seiten fester Größe abzuspeichern

  ⇒ Organisation von vereinfachten Geo-Objekten *(Approximationen)*, z.B. *minimal umgebende Rechtecke (MUR)*

  ⇒ Verweis auf die exakte Beschreibung (EB) des Geo-Objektes



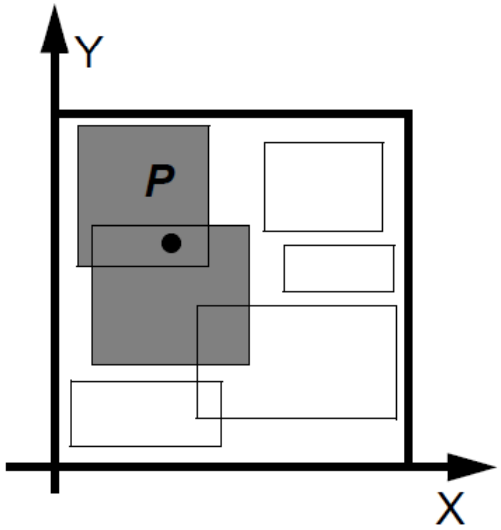


# 4.1 Einführung (II)

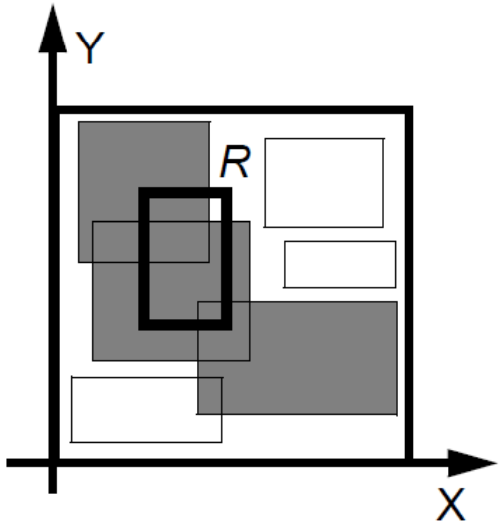


## Zu unterstützende Anfragen

- Gegeben ein Anfragepunkt *P* bzw. *ein Anfragerechteck R*
  - *Punkt-Anfrage:* Finde die Geo-Objekte Obj mit P ∈ Obj.MUR.
  - *Fenster-Anfrage:* Finde die Geo-Objekte Obj mit R ∩ Obj.MUR ≠ ∅.




*Punkt-Anfrage*




*Fenster-Anfrage*

## Zu unterstützende Anfragen (Forts.)

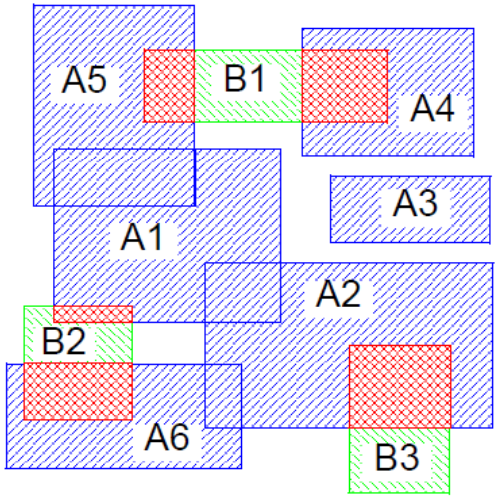
- Gegeben zwei Mengen minimal umgebender Rechtecke

  $M1 = \{MUR_{1,1}, MUR_{1,2}, \ldots, MUR_{1,m}\}$

  $M_2 = \{MUR_{2,1}, MUR_{2,2}, \ldots, MUR_{2,n}\}$
- *Spatial-Join: Berechne die Menge*

  $\{ (MUR_1, MUR_2) \mid MUR_1 \in M_1, MUR_2 \in M_2 \text{ und } MUR_1 \cap MUR_2 \neq \emptyset \}$.

  ⇒ auch andere räumliche Prädikate möglich




*Spatial-Join*

Antwortmenge:

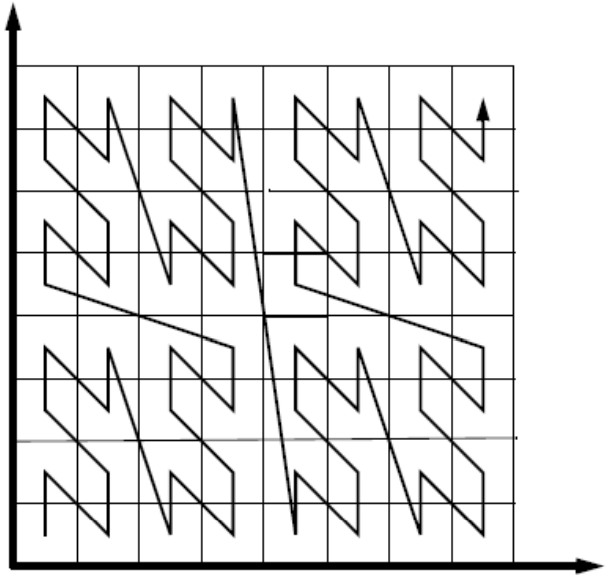
(A5, B1)
(A4, B1)
(A1, B2)
(A6, B2)
(A2, B3)

# 4.1 Prinzipielle Techniken (I)



## Einbettung in eindimensionalen Raum

- vollständige Zerlegung des Datenraums in gleichförmige disjunkte Zellen
- Definition einer linearen Ordnung auf diesen Zellen (z.B. Z-Ordnung)
- Organisation der Zellen über eine konventionelle (eindimensionale) Indexstruktur (z.B. B-Baum)

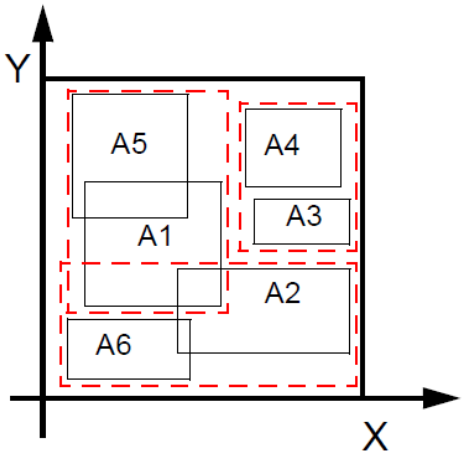


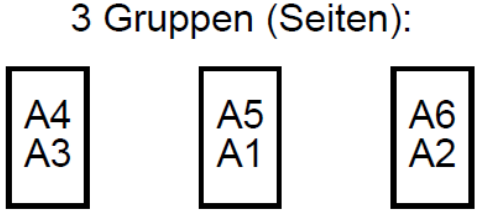
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 21 | 23 | 29 | 31 | 53 | 55 | 61 | 63 |
| 20 | 22 | 28 | 30 | 52 | 54 | 60 | 62 |
| 17 | 19 | 25 | 27 | 49 | 51 | 57 | 59 |
| 16 | 18 | 24 | 26 | 48 | 50 | 56 | 58 |
| 5 | 7 | 13 | 15 | 37 | 39 | 45 | 47 |
| 4 | 6 | 12 | 14 | 36 | 38 | 44 | 46 |
| 1 | 3 | 9 | 11 | 33 | 35 | 41 | 43 |
| 0 | 2 | 8 | 10 | 32 | 34 | 40 | 42 |

**Z-Ordnung**

## Organisation des 2D-Raums (Gruppierung und Speicherung in MURs)

- Aufteilung der Rechtecke (MURs) in disjunkte Gruppen, wobei jede Gruppe exklusiv in einer Seite abgespeichert wird

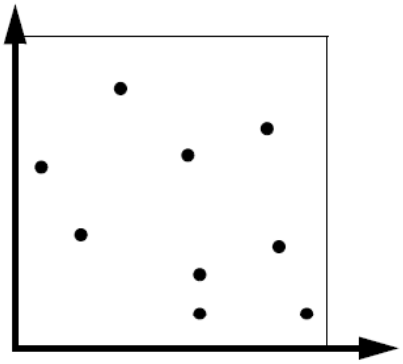


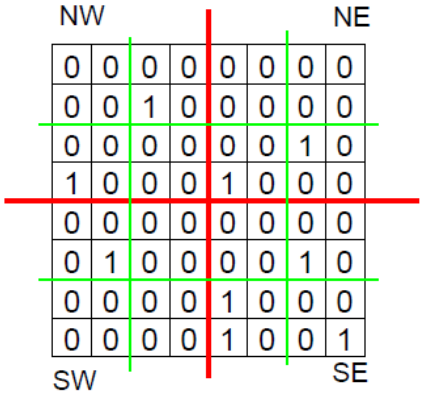




- Im Directory MURs der im Teilbaum enthaltenen MURs abspeichern.
- Split des Datenraums je nach abgespeicherten Daten.
  ⇒ R-Baum

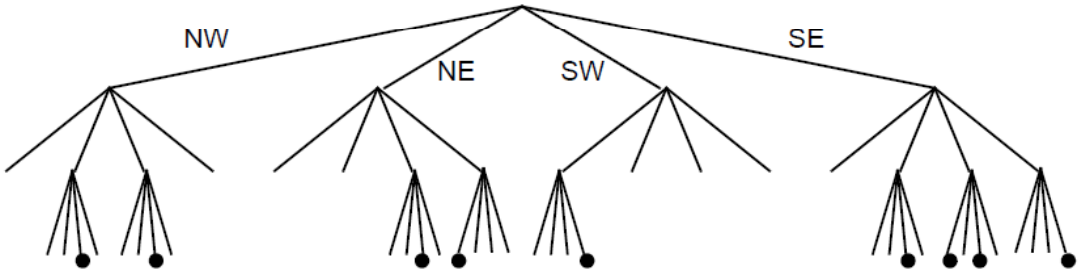
## Organisation des 2D-Raums (hierarchische Zerlegung des Raums)

- Split des Datenraums immer in der Mitte einer Dimension
- Punkte = Pixel in dem gerasterten Datenraum





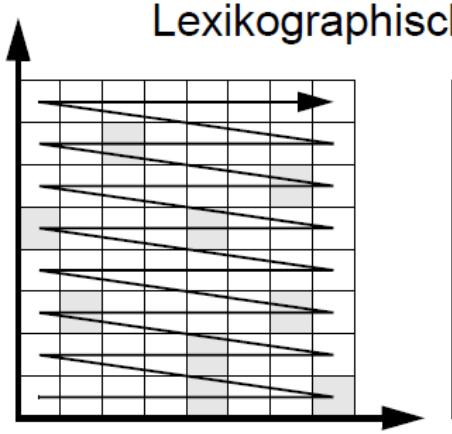

⇒ Quadtrees



# 4.2 Raumfüllende Kurven

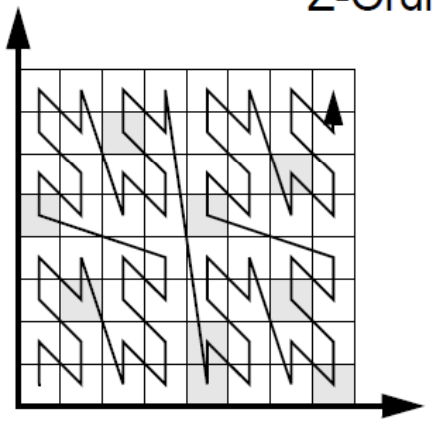


## Lexikographische Ordnung



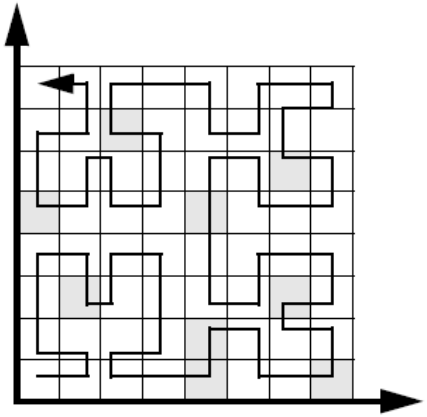
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

## Z-Ordnung



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 21 | 23 | 29 | 31 | 53 | 55 | 61 | 63 |
| 20 | 22 | 28 | 30 | 52 | 54 | 60 | 62 |
| 17 | 19 | 25 | 27 | 49 | 51 | 57 | 59 |
| 16 | 18 | 24 | 26 | 48 | 50 | 56 | 58 |
| 5 | 7 | 13 | 15 | 37 | 39 | 45 | 47 |
| 4 | 6 | 12 | 14 | 36 | 38 | 44 | 46 |
| 1 | 3 | 9 | 11 | 33 | 35 | 41 | 43 |
| 0 | 2 | 8 | 10 | 32 | 34 | 40 | 42 |

## Hilbert-Kurve

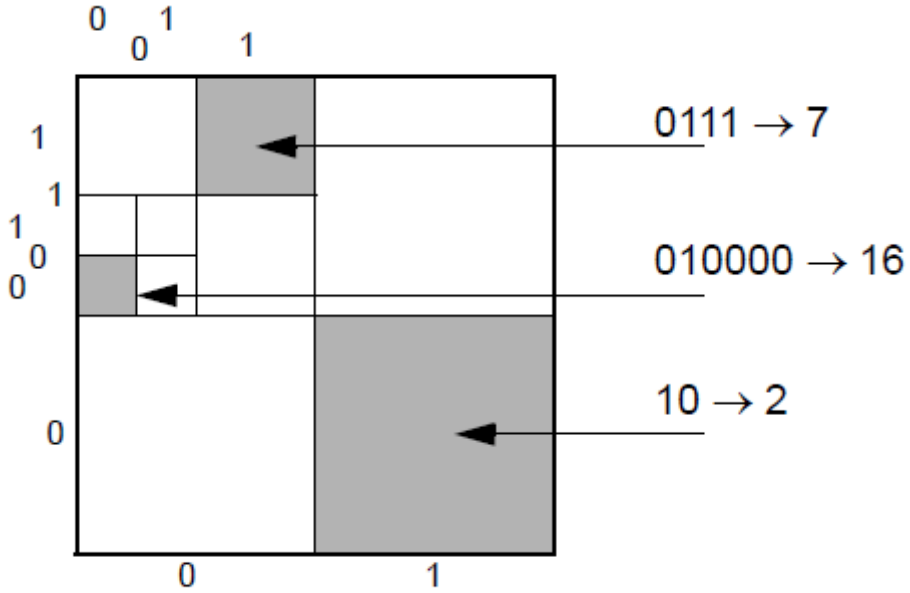


| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 63 | 62 | 49 | 48 | 47 | 44 | 43 | 42 |
| 60 | 61 | 50 | 51 | 46 | 45 | 40 | 41 |
| 59 | 56 | 55 | 52 | 33 | 34 | 39 | 38 |
| 58 | 57 | 54 | 53 | 32 | 35 | 36 | 37 |
| 5 | 6 | 9 | 10 | 31 | 28 | 27 | 26 |
| 4 | 7 | 8 | 11 | 30 | 29 | 24 | 25 |
| 3 | 2 | 13 | 12 | 17 | 18 | 23 | 22 |
| 0 | 1 | 14 | 15 | 16 | 19 | 20 | 21 |

- Z-Ordnung erhält die räumliche Nähe relativ gut
- Z-Ordnung ist effizient berechenbar

# 4.2 Z-Ordnung (I)

## Codierung von Zellen




abwechselnd rechts/links und oben/unten partitionieren

## Z-Werte

- *Level* eines Codes = Anzahl der Bits
- *Z-Wert* = (Dezimalwert des Codes, Level)

# 4.2 Z-Ordnung (II)



## Codierung von Rechtecken (MURs)

- durch die minimal umgebende Zelle

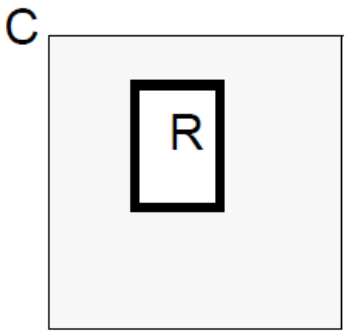
  ⇒ Probleme mit Rechtecken, die auf den Schnittlinien liegen

- durch mehrere Zellen

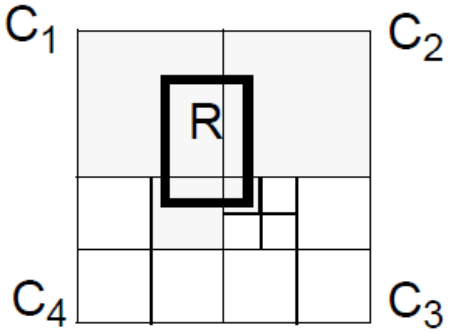
  ⇒ bessere Approximation des Rechtecks

  ⇒ redundante Abspeicherung

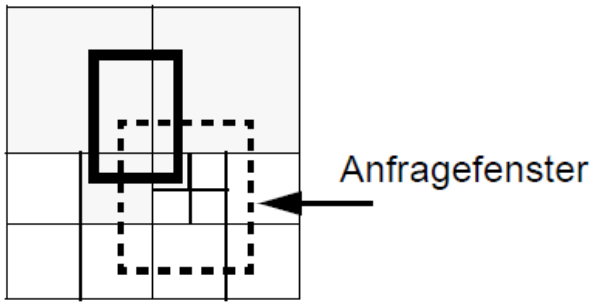
  ⇒ redundante Antworten




Codierung von R durch eine Zelle




Codierung von R durch mehrere Zellen




Anfrage liefert mehrmals die gleiche Antwort

# 4.2 Z-Ordnung (III)



## Abbildung auf B$^{+-}$Baum

- Lineare Ordnung zur Verwaltung im B$^+$-Baum
  - Seien $(c_1, l_1)$ und $(c_2, l_2)$ zwei Z-Werte und sei $l = \min \{l_1, l_2\}$.
  - Dann ist die Ordnungsrelation $\leq_Z$ wie folgt definiert:

    $(c_1, l_1) \leq (c_2, l_2)$ falls $c_1 \text{div } 2^{l1-l}$ $c_2 \text{div } 2^{l2-l}$

  - Beispiele:

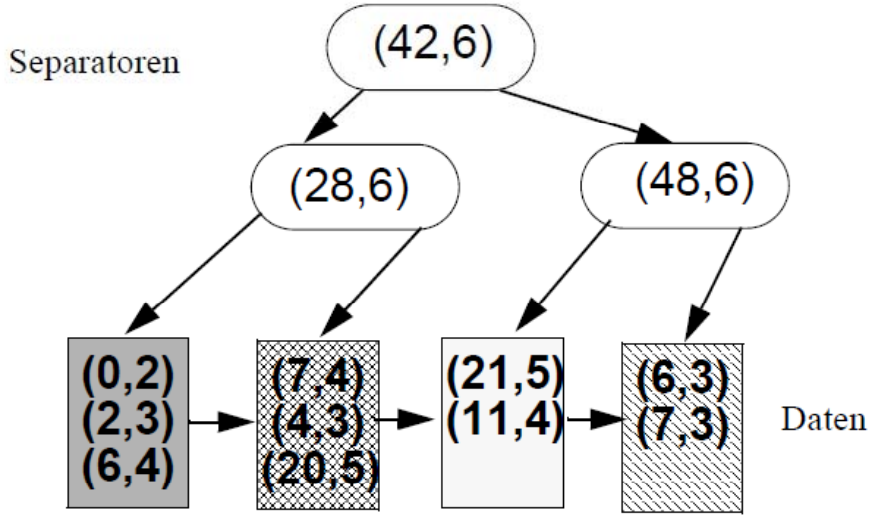
    $(1,2) \leq_Z (3,2)$, $(3,4) \leq_Z (3,2)$, $(1,2) \leq_Z (10,4)$

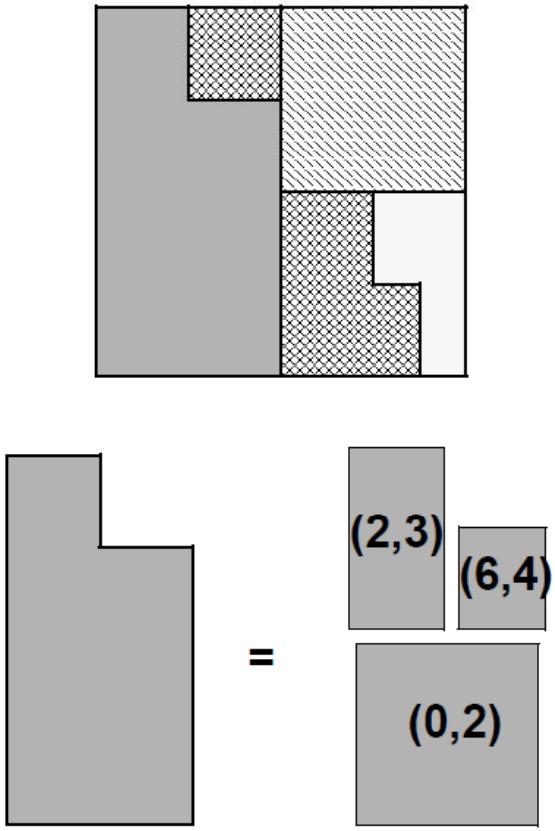
- Wenn ein Blatt des B$^+$-Baums überläuft, dann Split der Seite in 2 Seiten gemäß B$^+$-Baum Strategie.

# 4.2 Z-Ordnung (IV)



## Beispiel




$(0,2) \leq (28,6) \leq (7,4) \leq (42,6)$

$(2,3) \leq (28,6) \leq (4,3) \leq (42,6)$

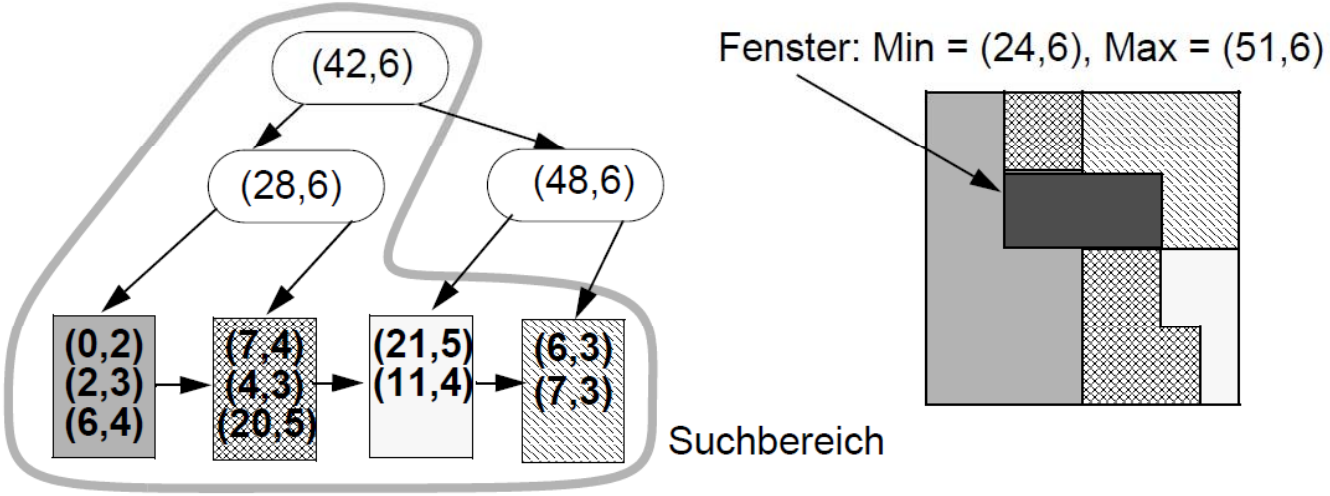
$(6,4) \leq (28,6) \leq (20,5) \leq (42,6)$

# 4.2 Z-Ordnung (V)



## Fenster-Anfrage (1. Ansatz)

- Benutze den "gewöhnlichen" Algorithmus für Bereichsanfragen im B+-Baum:
  - Suche für den kleinsten Z-Wert des Windows (entspricht dem linken unteren Eckpunkt) das zugehörige Blatt im B+-Baum
  - Durchlaufe sequentiell die Blätter bis ein Z-Wert größer als der größte Z Wert im Suchrechteck gefunden wurde



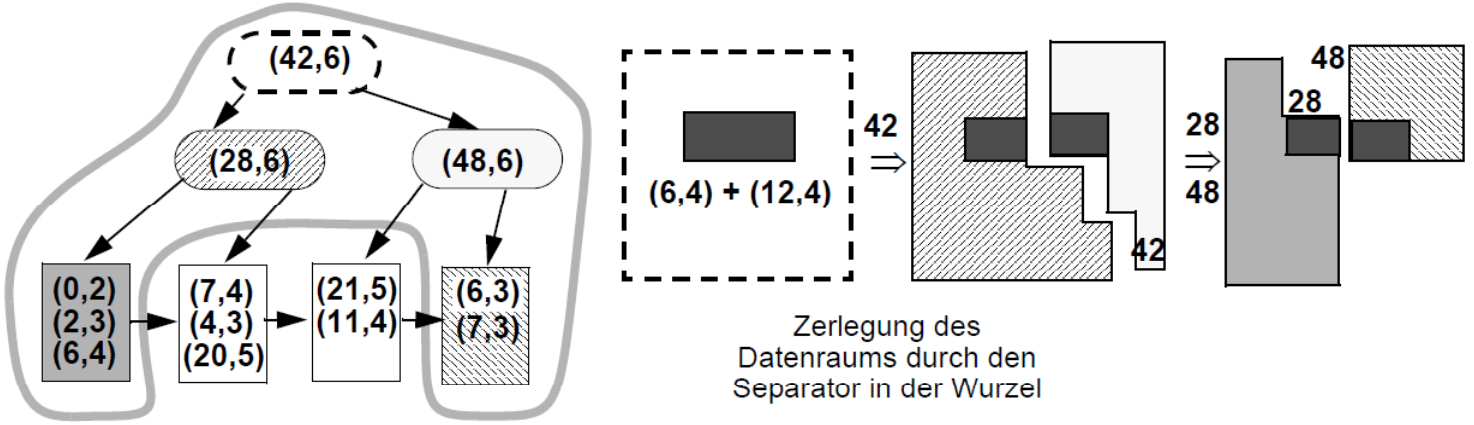

  - ineffizient, da der Suchbereich verglichen mit dem Fenster sehr groß ist

# 4.2 Z-Ordnung (VI)



## Fenster-Anfrage (2. Ansatz)

- Jeder Knoten des B⁺-Baums repräsentiert einen Bereich des Datenraums, durch die Separatoren wird der zugehörige Bereich jedes Knotens in Teilbereiche zerlegt
- *Idee:* Verwende zur Beantwortung der Anfrage in einem Teilbaum nur den Teil des Windows, der den Bereich des Teilbaums schneidet




Zerlegung des Datenraums durch den Separator in der Wurzel

- Mehraufwand für das Durchlaufen der Indexseiten im B⁺-Baum (Separatoren)
- Teilbereiche sind nicht notwendigerweise Rechtecke
+ Zugriff nur auf die tatsächlich relevanten Datenseiten