Algorithmen für geographische Informationssysteme

5. Vorlesung

Sommersemester 2023

Alexander Wolff

Schriftplazierung

Europe seen by the Germans

http://bigthink.com/blogs/strange-maps

Schriftplazierung

Europe seen by the French

http://bigthink.com/blogs/strange-maps

http://www.worldofmaps.net/

nach Imhof (1975)

Beschriftung (Namen) für

- **Flächen**
- **Linien**
- **Punkte**

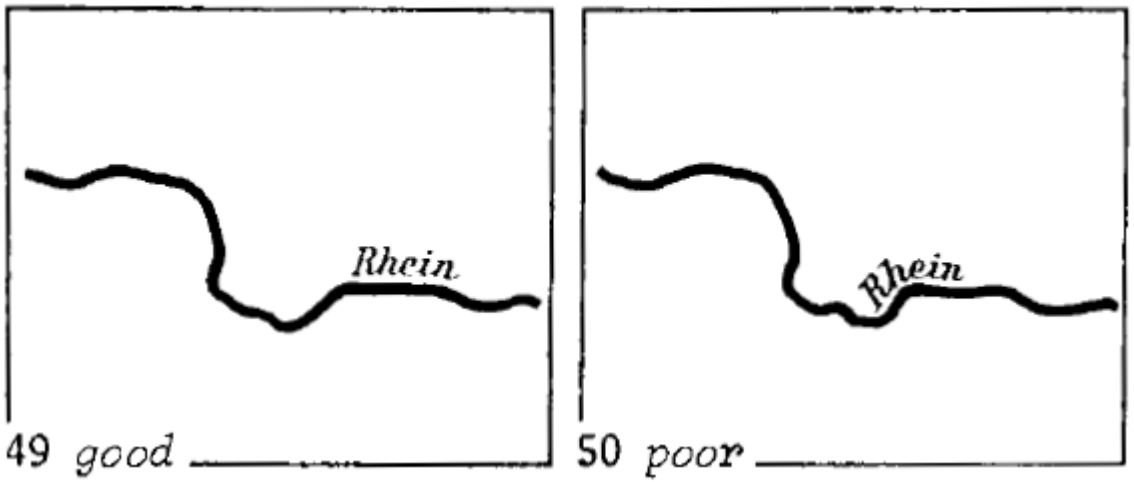
abhängig vom Maßstab!
(Berlin: Fläche oder Punkt)

http://www.worldofmaps.net/

nach Imhof (1975)

Kriterien für gute Beschriftung

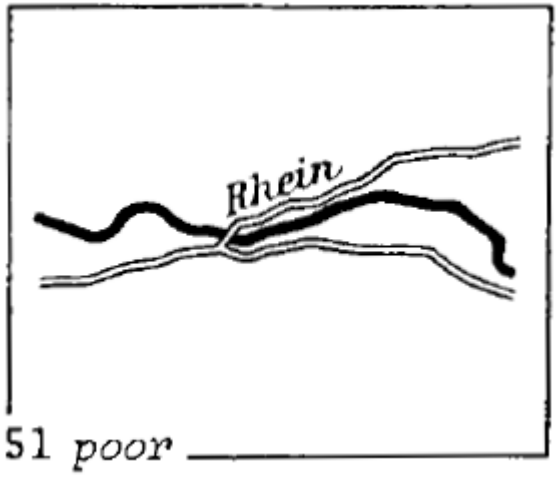
1. Lesbarkeit



nach Imhof (1975)

Kriterien für gute Beschriftung

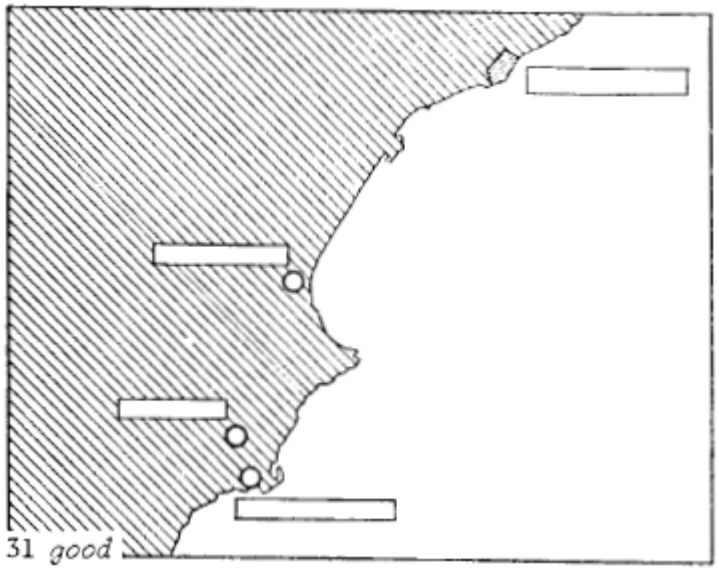
1. Lesbarkeit
2. klare Zuordnung von Namen zu Objekten

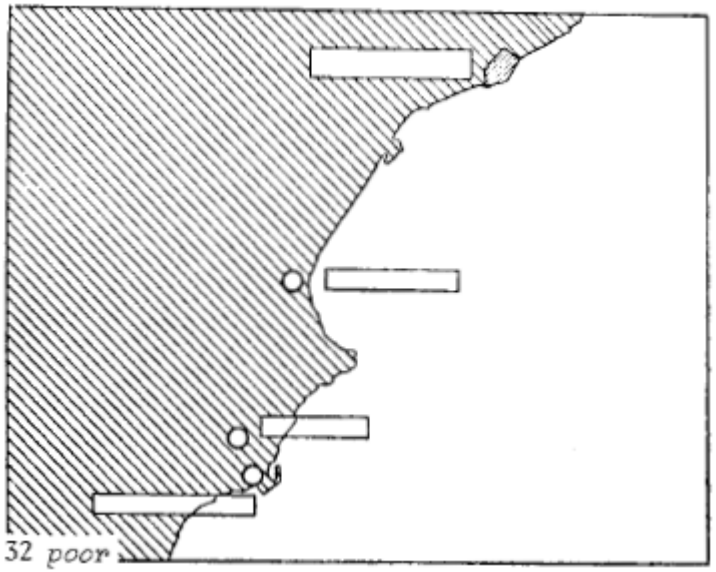


nach Imhof (1975)

Kriterien für gute Beschriftung

1. Lesbarkeit
2. klare Zuordnung von Namen zu Objekten
3. Namen sollen anderen Karteninhalt wenig stören (keine Verdeckung von Objekten)

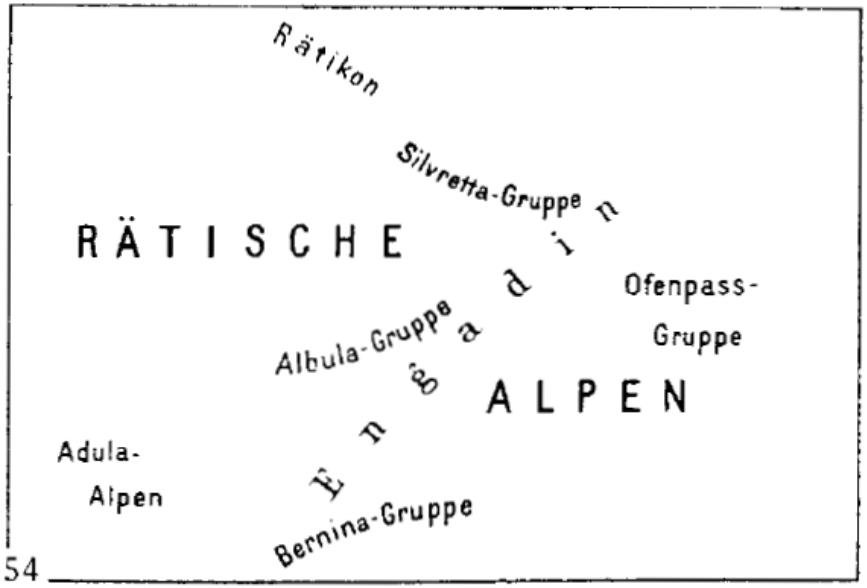

31 *good*


32 *poor*

nach Imhof (1975)

Kriterien für gute Beschriftung

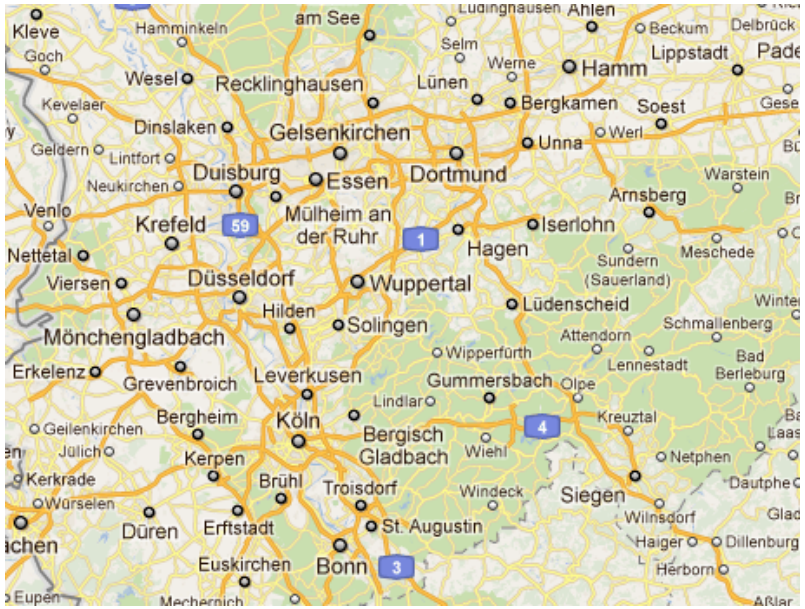
1. Lesbarkeit
2. klare Zuordnung von Namen zu Objekten
3. Namen sollen anderen Karteninhalt wenig stören (keine Verdeckung von Objekten)
4. Namen sollen Verständnis von Kartenobjekten erleichtern.
5. Schrifttyp und -größe sollen Klassen und Hierarchien wiedergeben.



nach Imhof (1975)

Kriterien für gute Beschriftung

1. Lesbarkeit
2. klare Zuordnung von Namen zu Objekten
3. Namen sollen anderen Karteninhalt wenig stören (keine Verdeckung von Objekten)
4. Namen sollen Verständnis von Kartenobjekten erleichtern.
5. Schrifttyp und -größe sollen Klassen und Hierarchien wiedergeben.
6. Dichte der Namen soll angemessen variieren.



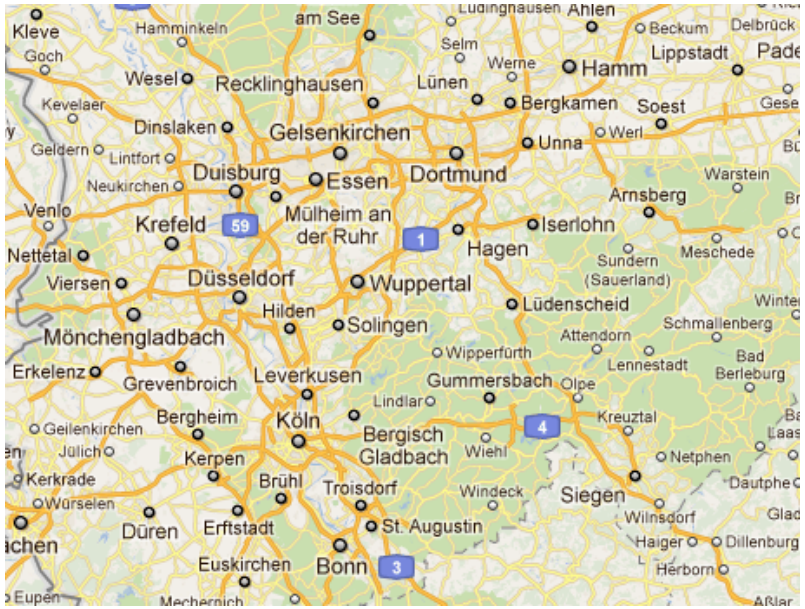
maps.google.com

Dichte der Namen soll mit Siedlungsdichte variieren, in Ballungszentren aber nicht zu hoch sein.

nach Imhof (1975)

Kriterien für gute Beschriftung

1. Lesbarkeit
2. klare Zuordnung von Namen zu Objekten
3. Namen sollen anderen Karteninhalt wenig stören (keine Verdeckung von Objekten)
4. Namen sollen Verständnis von Kartenobjekten erleichtern.
5. Schrifttyp und -größe sollen Klassen und Hierarchien wiedergeben.
6. Dichte der Namen soll angemessen variieren.

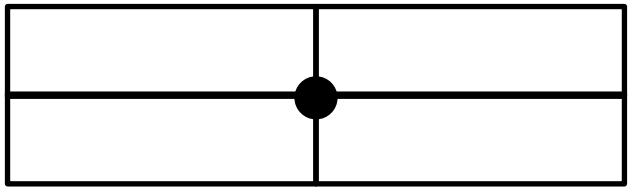


maps.google.com

Textplazierung:

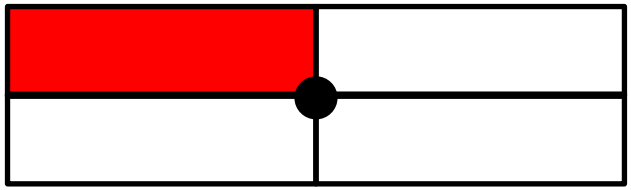
1. **Auswahl von Namen**
2. **Anordnung der Schrift**

Beschriftung von Punkten



four-position model

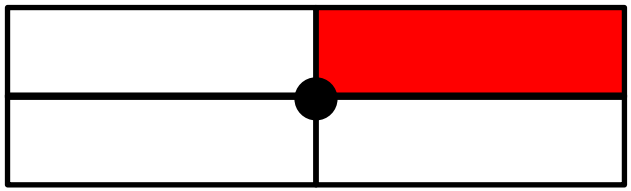
Beschriftung von Punkten



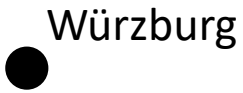
four-position model

Würzburg

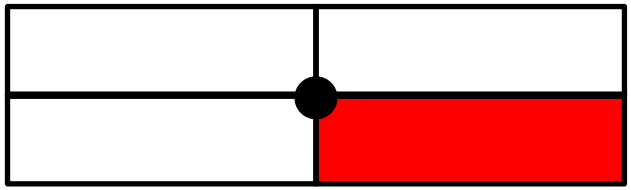
Beschriftung von Punkten



four-position model



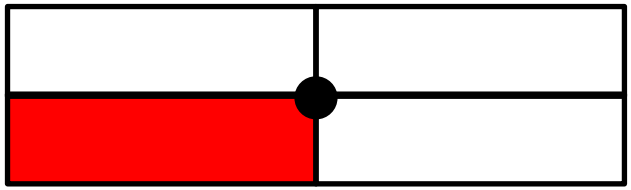
Beschriftung von Punkten



four-position model

Würzburg

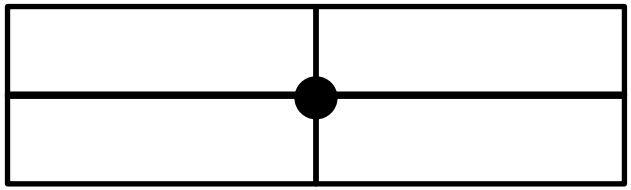
Beschriftung von Punkten



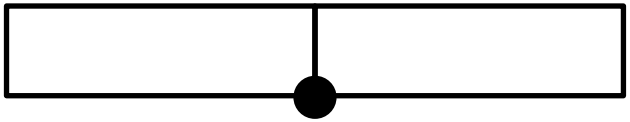
four-position model

Würzburg

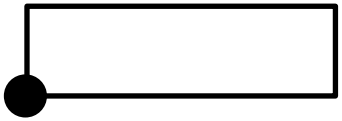
Beschriftung von Punkten



four-position model



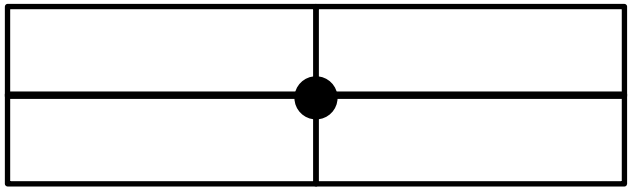
two-position model



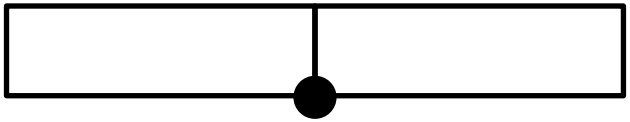
one-position model

Beschriftung von Punkten

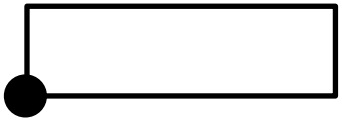
fixed-position model



four-position model

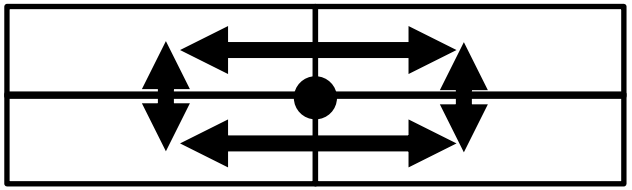


two-position model

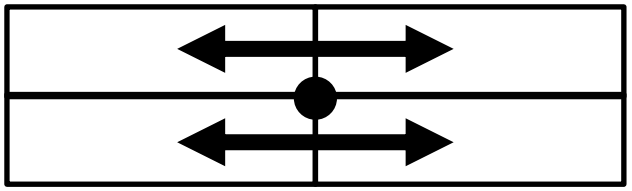


one-position model

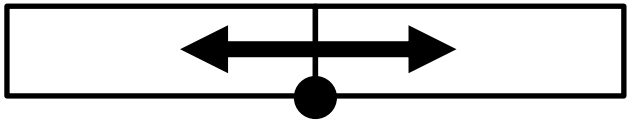
slider model



four-slider model



two-slider model

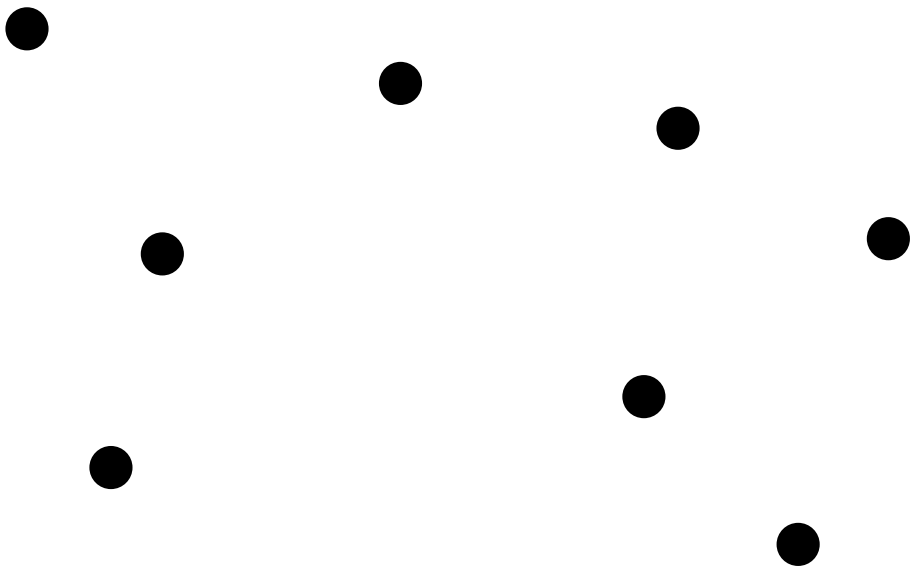


one-slider model

Würzburg

Würzburg

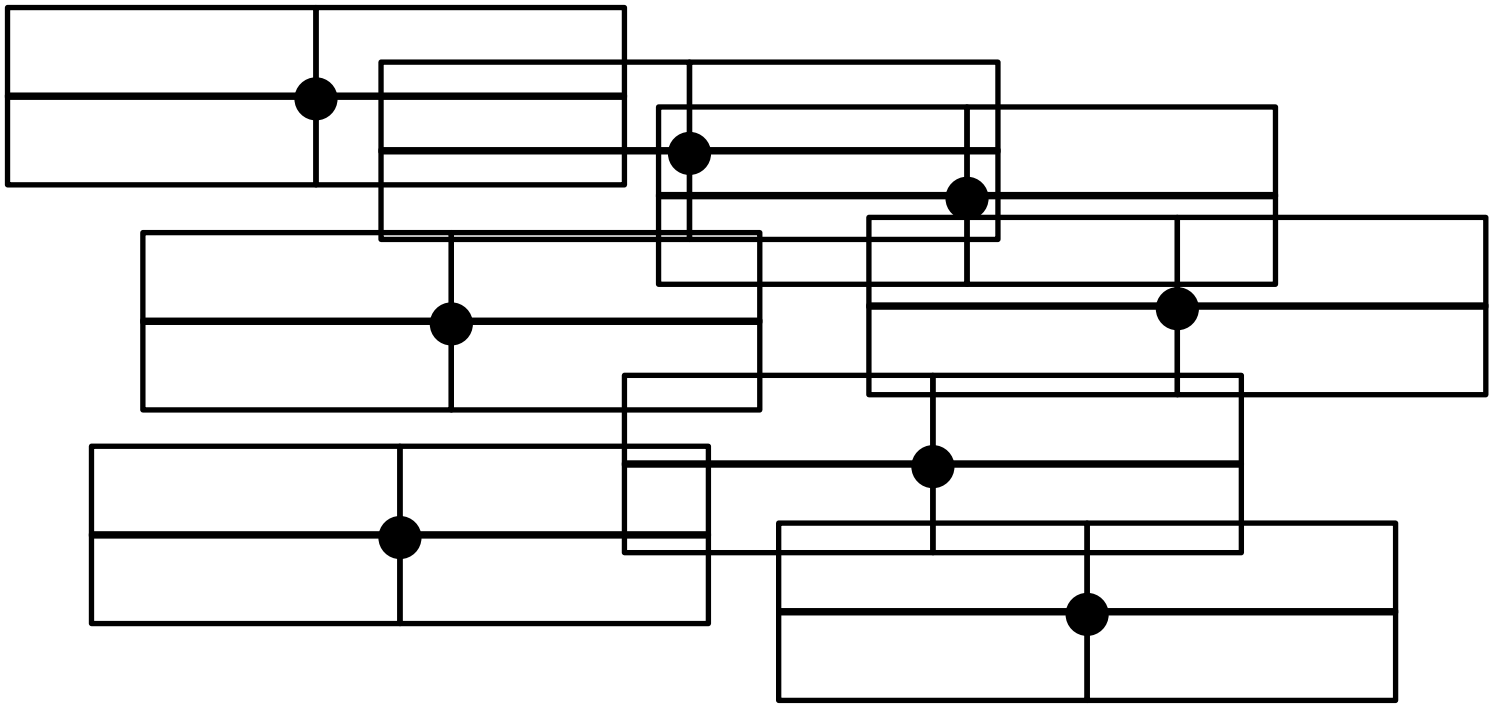
Beschriftung von Punkten



typisches Problem:

- Punktmenge gegeben

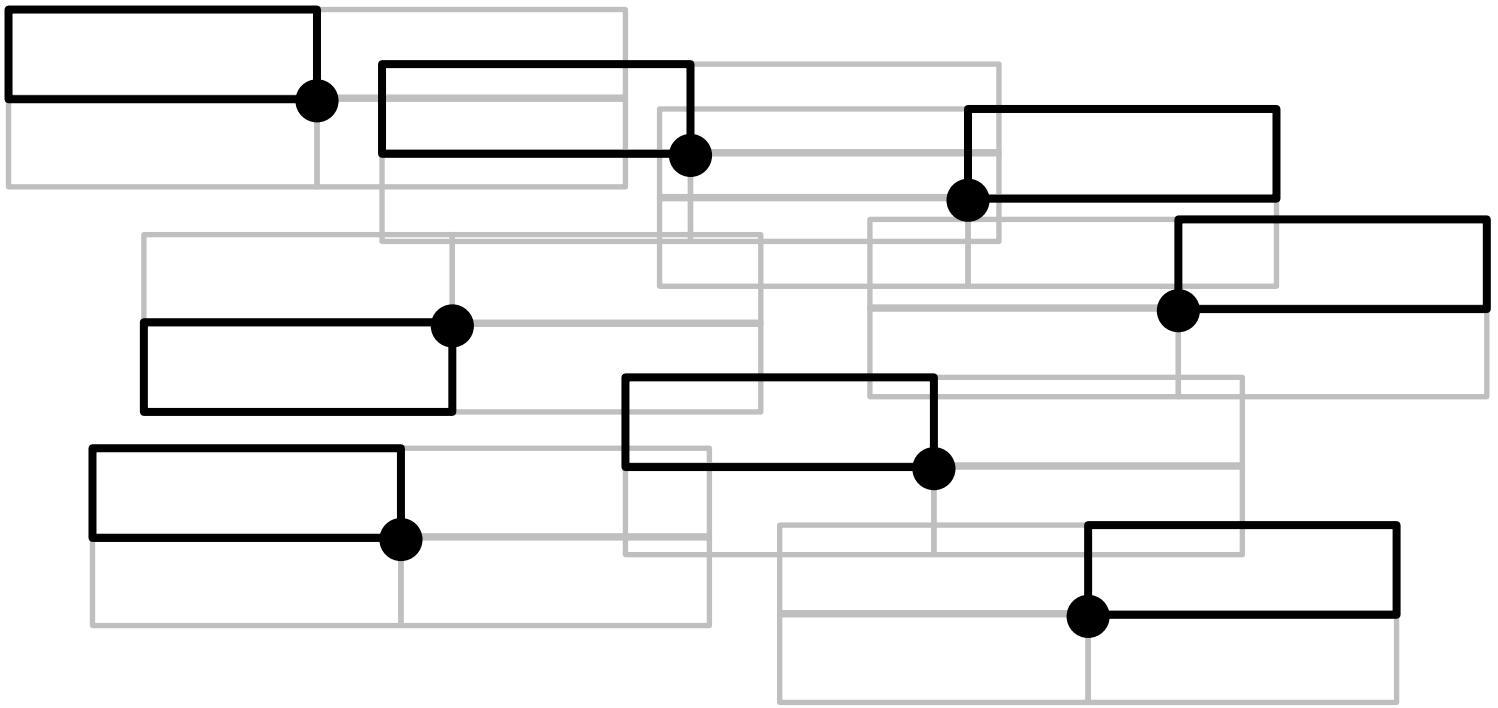
Beschriftung von Punkten



typisches Problem:

- Punktmenge gegeben
- Modell gewählt

Beschriftung von Punkten



typisches Problem:

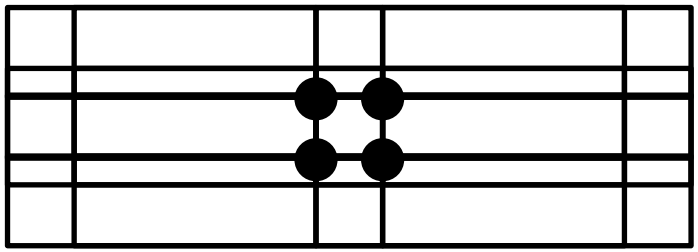
- Punktmenge gegeben
- Modell gewählt
- beschrifte **möglichst alle Punkte**, so dass sich Label nicht schneiden

Beschriftung von Punkten

typisches Problem:

- Punktmenge gegeben
- Modell gewählt
- beschrifte **möglichst alle Punkte**, so dass sich Label nicht schneiden

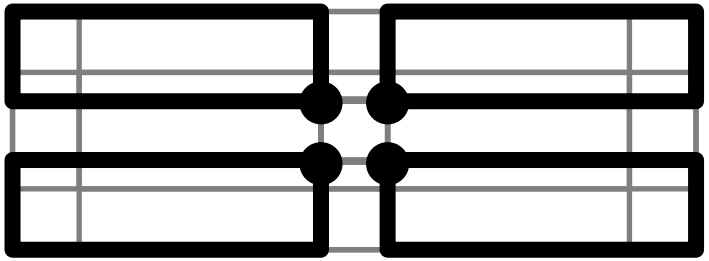
Beschriftung von Punkten



typisches Problem:

- Punktmenge gegeben
- Modell gewählt
- beschrifte **möglichst alle Punkte**, so dass sich Label nicht schneiden

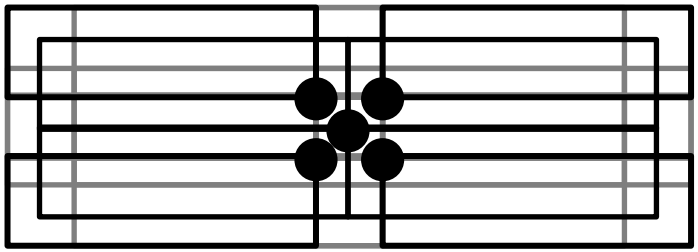
Beschriftung von Punkten



typisches Problem:

- Punktmenge gegeben
- Modell gewählt
- beschrifte **möglichst alle Punkte**, so dass sich Label nicht schneiden

Beschriftung von Punkten

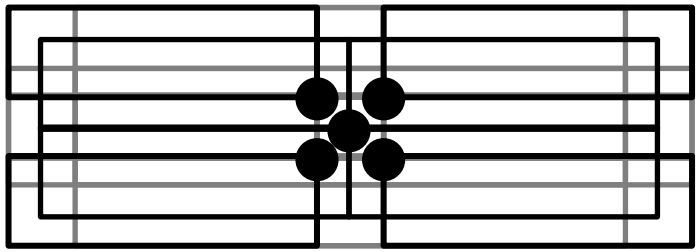


Bei fünf Punkten gibt es im four-position model möglicherweise keine Lösung, in der alle Punkte beschriftet werden!

typisches Problem:

- Punktmenge gegeben
- Modell gewählt
- beschrifte **möglichst alle Punkte**, so dass sich Label nicht schneiden

Beschriftung von Punkten



Bei fünf Punkten gibt es im four-position model möglicherweise keine Lösung, in der alle Punkte beschriftet werden!

typisches Problem:

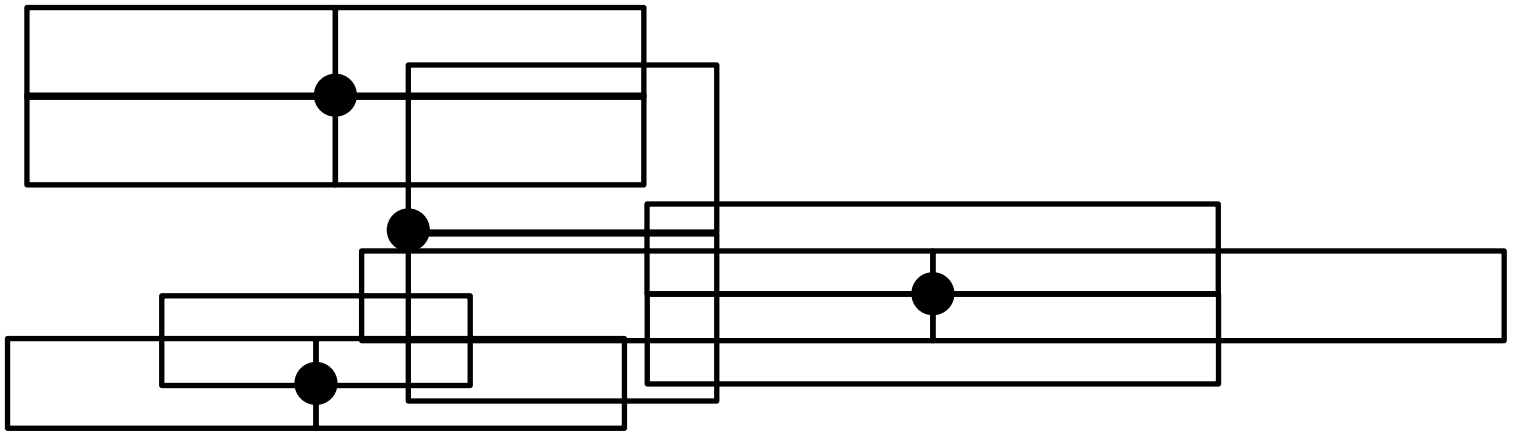
- Punktmenge gegeben
- Modell gewählt
- beschrifte **möglichst *viele* Punkte**, so dass sich Label nicht schneiden

Beschriftung von Punkten

Allgemeiner Ansatz für fixed-position models:

Agarwal et al. (1998): Label placement by maximum independent set in rectangles

- Für Punkt p_i mit $i = 1, \ldots, m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.

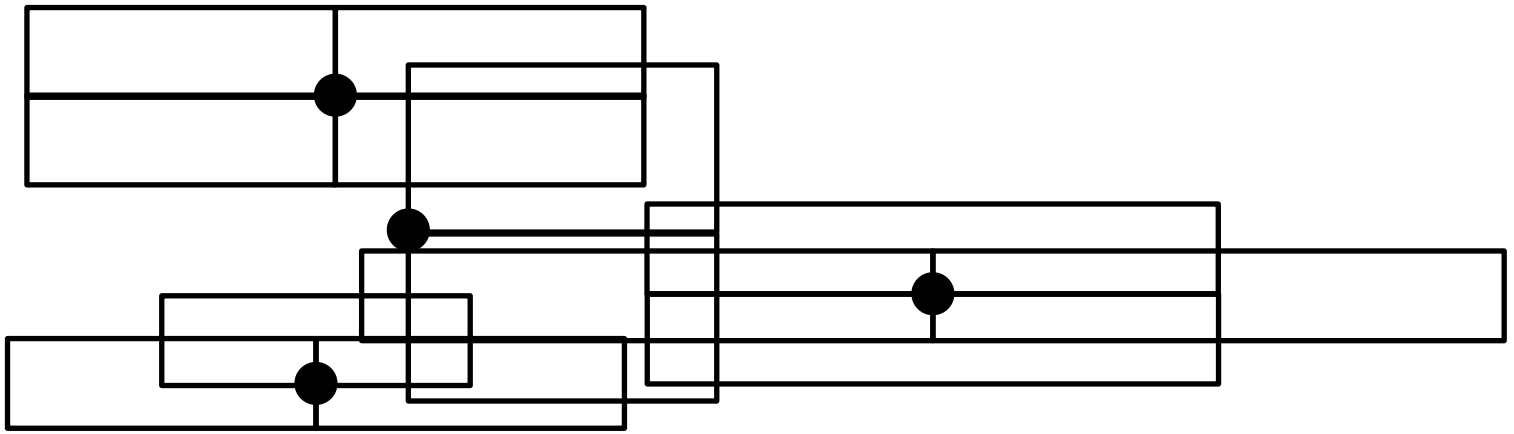


Beschriftung von Punkten

Allgemeiner Ansatz für fixed-position models:

Agarwal et al. (1998): Label placement by maximum independent set in rectangles

- Für Punkt p_i mit $i = 1, \ldots, m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.



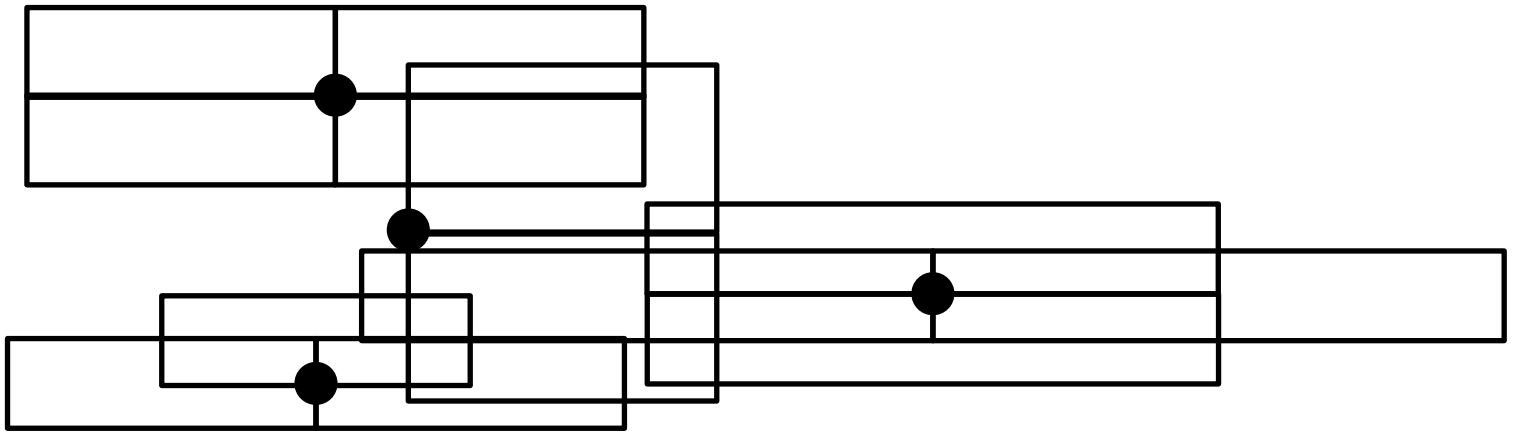
- gegeben Menge $R = R_1 \cup \cdots \cup R_m$
- wähle **möglichst viele Rechtecke** aus R, die sich nicht schneiden

Beschriftung von Punkten

Allgemeiner Ansatz für fixed-position models:

Agarwal et al. (1998): Label placement by maximum independent set in rectangles

- Für Punkt p_i mit $i = 1, \ldots, m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.



- gegeben Menge $R = R_1 \cup \cdots \cup R_m$
- wähle **möglichst viele Rechtecke** aus R, die sich nicht schneiden

(Es werden nie zwei Rechtecke r_1, r_2 für denselben Punkt p gewählt, da p in r_1 und r_2 liegt; r_1 und r_2 schneiden sich also.)

Beschriftung von Punkten

Problem:

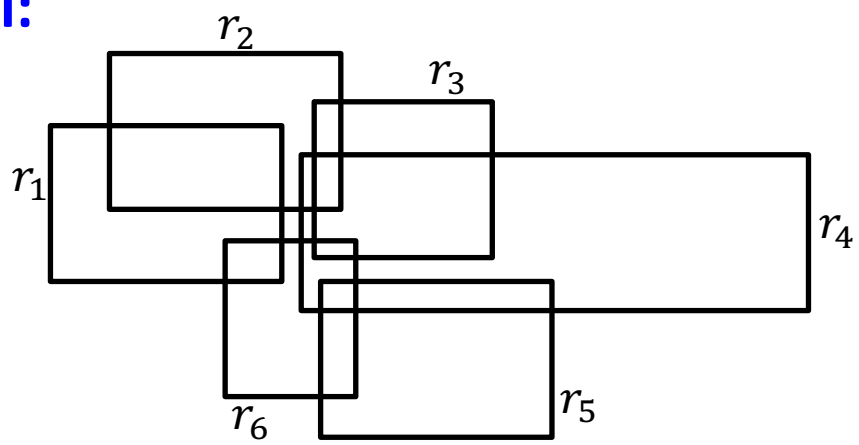
- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Allgemeinfall:

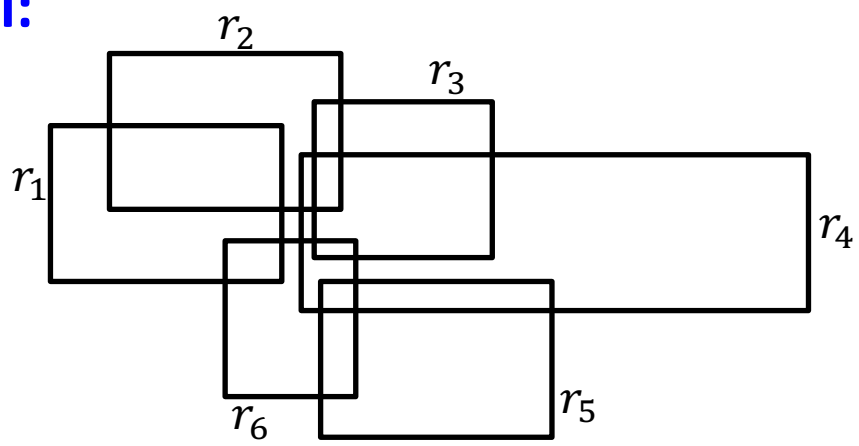


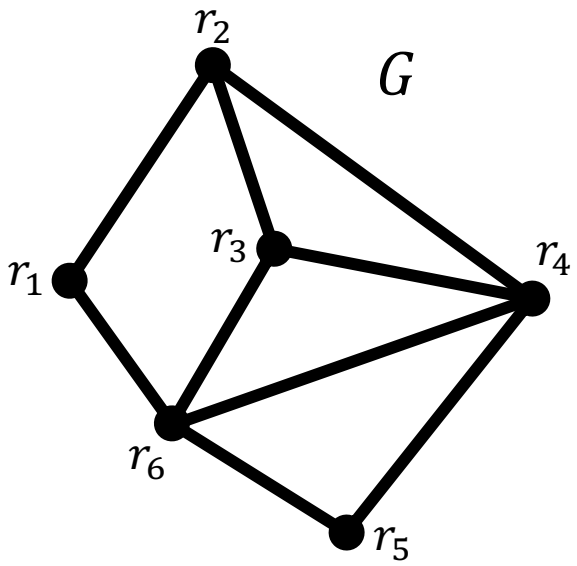
Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Allgemeinfall:





Definition:

Der Schnittgraph $G = (R, E)$

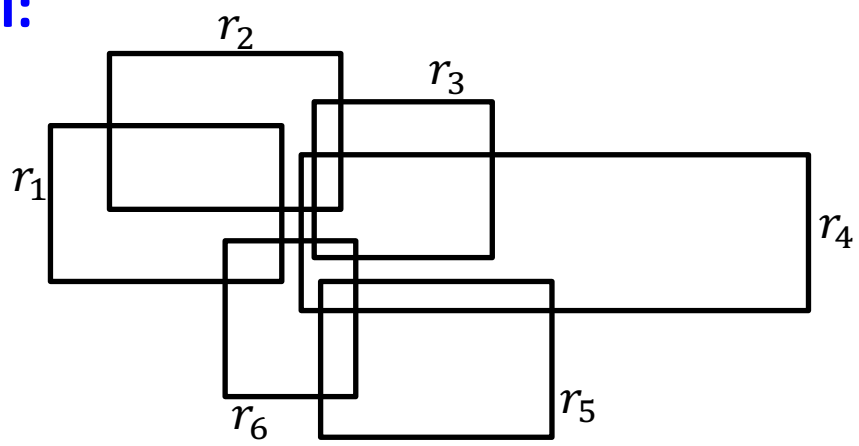
- enthält einen Knoten für jedes Rechteck und
- eine Kante $\{u, v\}$ wenn sich die Rechtecke u und v schneiden.

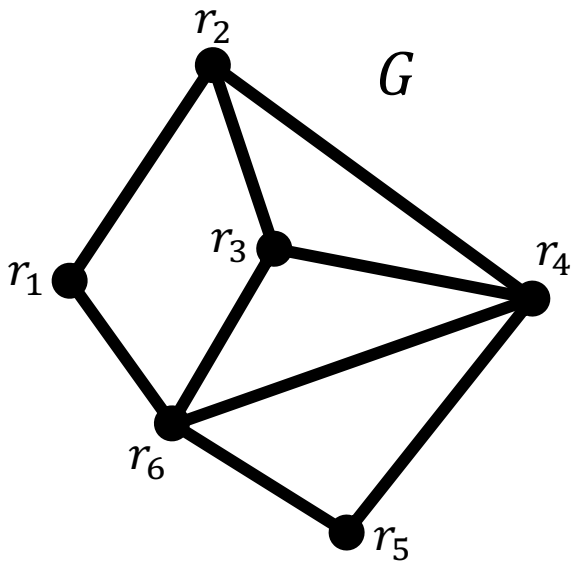
Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Allgemeinfall:





Definition:

Der Schnittgraph $G = (R, E)$

- enthält einen Knoten für jedes Rechteck und
- eine Kante $\{u, v\}$ wenn sich die Rechtecke u und v schneiden.

Lösung:

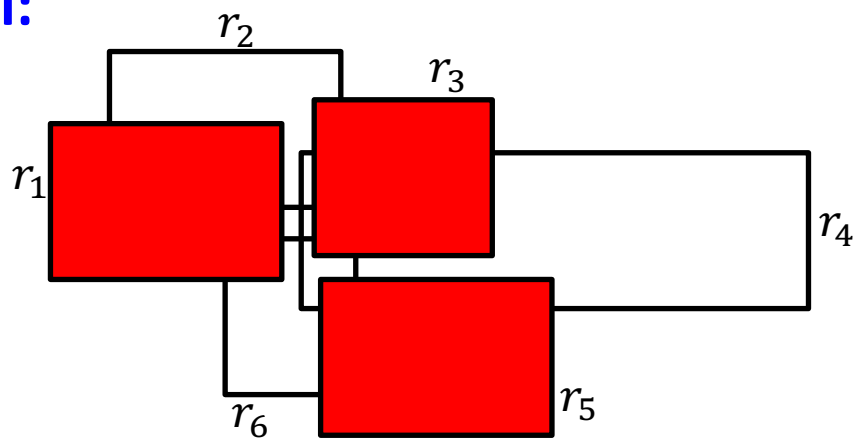
- Finde ***größte unabhängige Menge*** in G

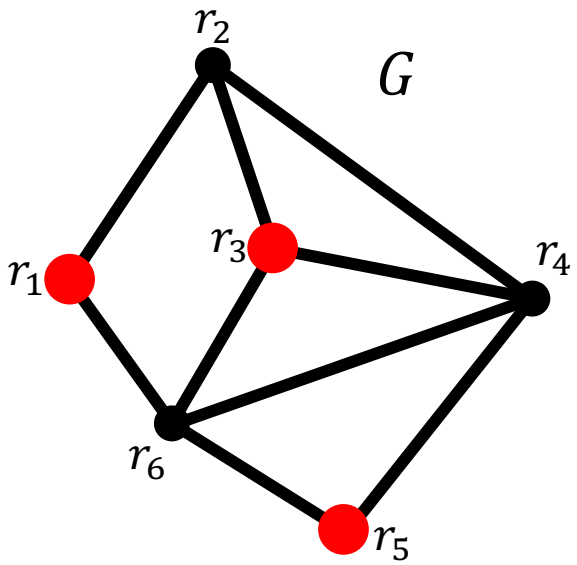
Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Allgemeinfall:





Definition:

Der Schnittgraph $G = (R, E)$

- enthält einen Knoten für jedes Rechteck und
- eine Kante $\{u, v\}$ wenn sich die Rechtecke u und v schneiden.

Lösung:

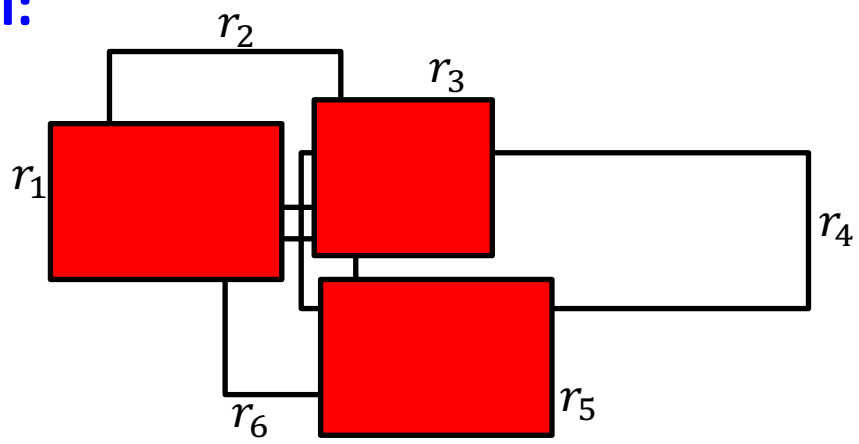
- Finde ***größte unabhängige Menge*** in G

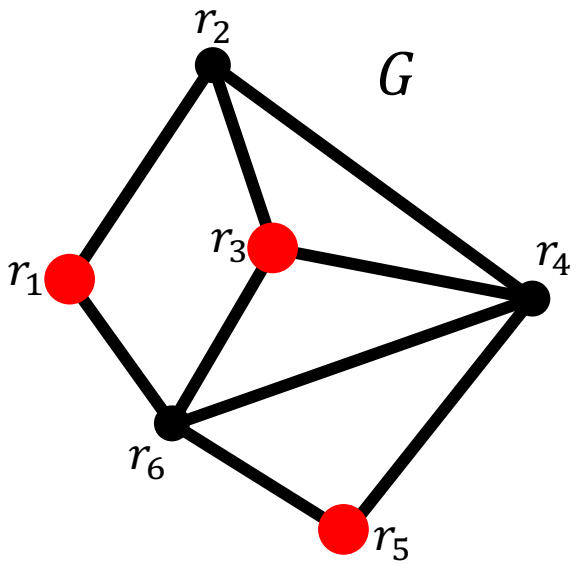
Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Allgemeinfall:





Definition:

Der Schnittgraph $G = (R, E)$

- enthält einen Knoten für jedes Rechteck und
- eine Kante $\{u, v\}$ wenn sich die Rechtecke u und v schneiden.

Lösung:

- Finde ***größte unabhängige Menge*** in G

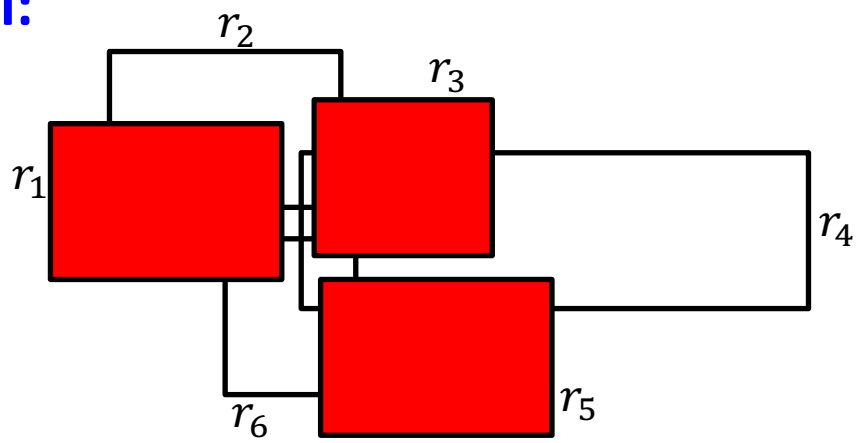
  für allgemeine Graphen NP-schwer!

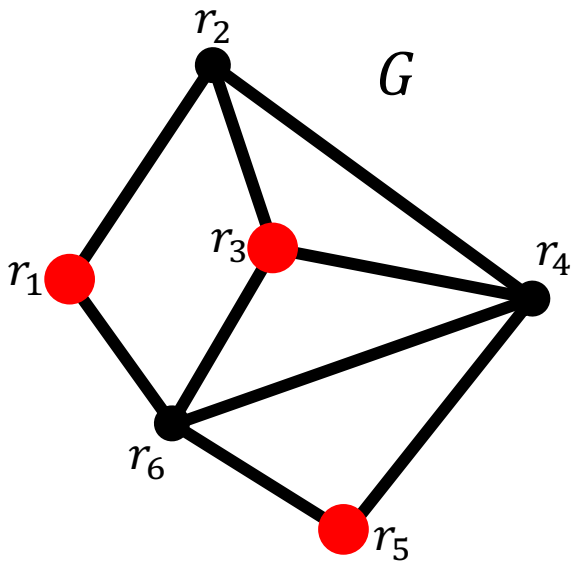
Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Allgemeinfall:





Definition:

Der Schnittgraph $G = (R, E)$

- enthält einen Knoten für jedes Rechteck und
- eine Kante $\{u, v\}$ wenn sich die Rechtecke u und v schneiden.

Lösung:

- Finde ***größte unabhängige Menge*** in G

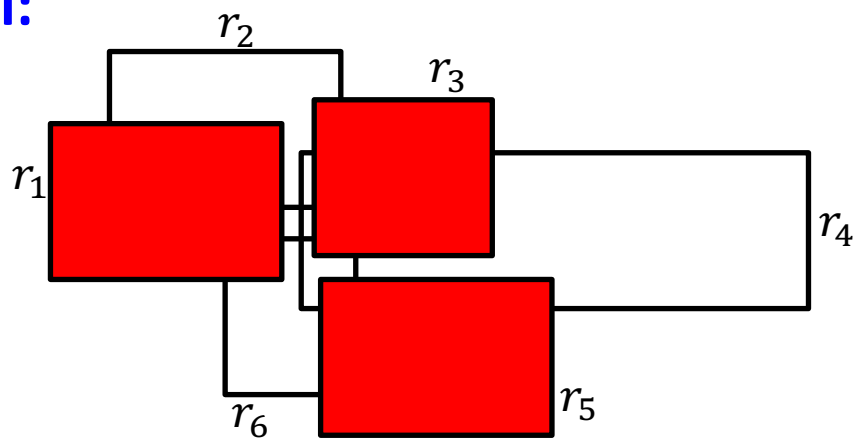
  für *Schnittgraphen von gleich großen Quadraten* NP-schwer!
  (Imai & Asano, 1983)

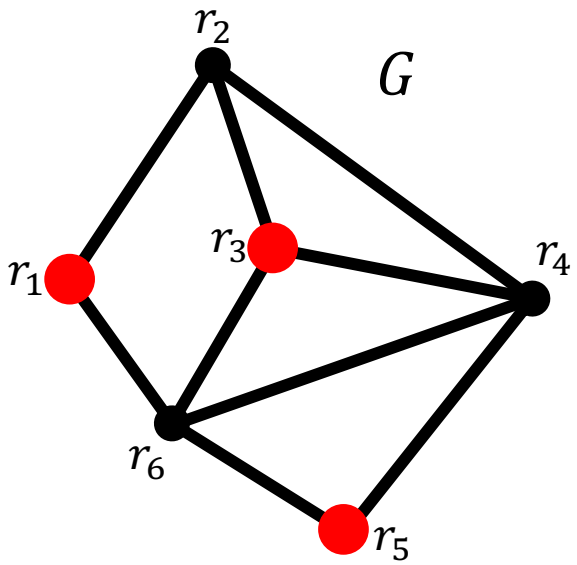
Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Allgemeinfall:





Definition:

Der Schnittgraph $G = (R, E)$

- enthält einen Knoten für jedes Rechteck und
- eine Kante $\{u, v\}$ wenn sich die Rechtecke u und v schneiden.

Ziel:

Approximationsalgorithmus

Ansatz:

Verwende Algorithmus für einfachen Spezialfall

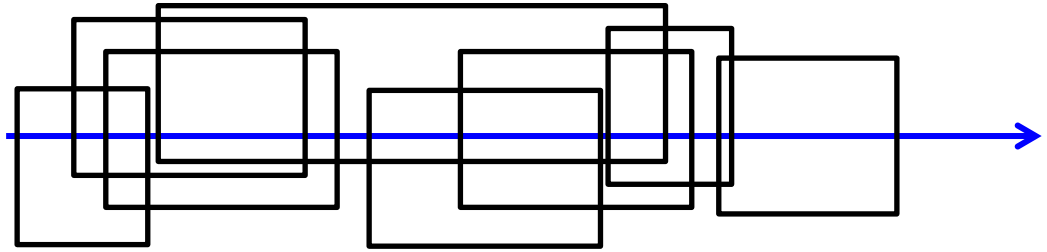
Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Spezialfall:

- Es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.



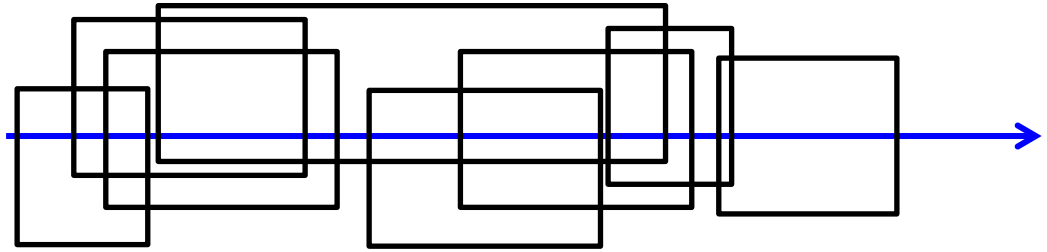
Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Spezialfall:

- Es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.



Algorithmus:

Greedy!

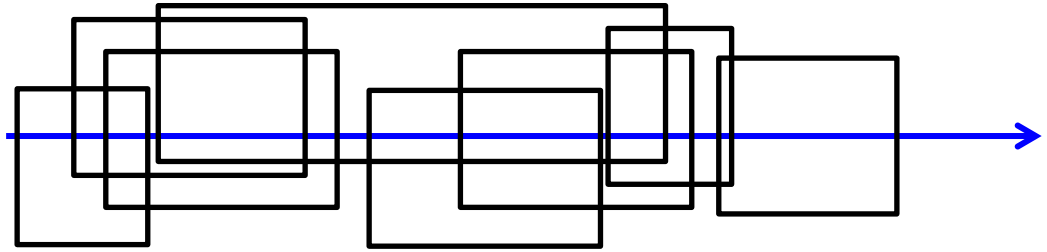
Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Spezialfall:

- Es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.



Algorithmus:

sortiere Rechtecke nach x-Koordinate ihres rechten Randes in Liste L

$S = \emptyset, r = L.head$

while $r \neq nil$ **do**

  $S = S \cup \{r\}$

  $r' = r.next$

  while $r' \neq nil$ **and** $r'.xMin \leq r.xMax$ **do** $r' = r'.next$

  $r = r'$

return S

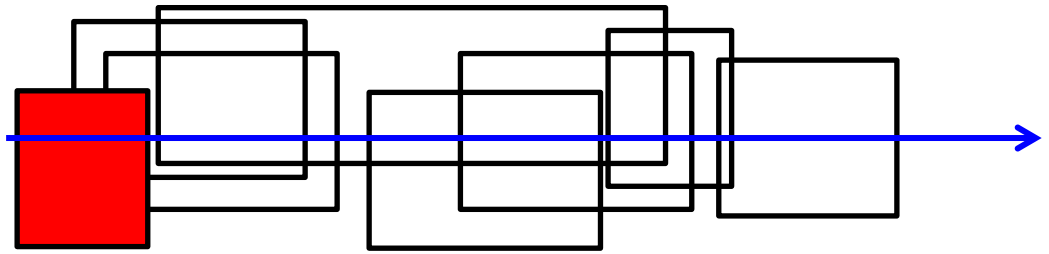
Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Spezialfall:

- Es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.



Algorithmus:

sortiere Rechtecke nach x-Koordinate ihres rechten Randes in Liste L

$S = \emptyset, r = L.head$

while $r \neq nil$ **do**

  $S = S \cup \{r\}$

  $r' = r.next$

  while $r' \neq nil$ **and** $r'.xMin \leq r.xMax$ **do** $r' = r'.next$

  $r = r'$

return S

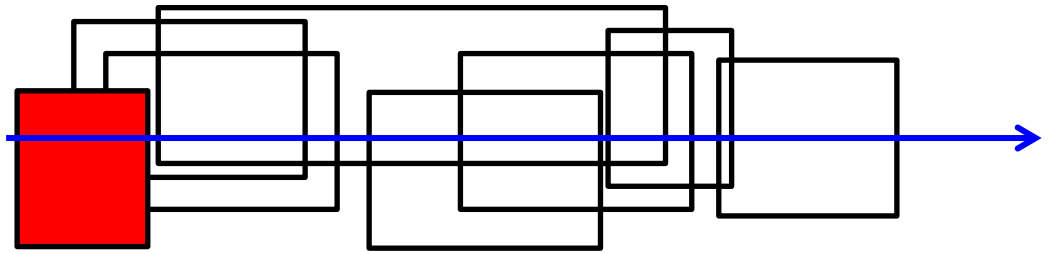
Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Spezialfall:

- Es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.



Algorithmus:

sortiere Rechtecke nach x-Koordinate ihres rechten Randes in Liste L
$S = \emptyset, r = L.head$
while $r \neq nil$ **do**
  $S = S \cup \{r\}$
  $r' = r.next$
  while $r' \neq nil$ **and** $r'.xMin \leq r.xMax$ **do** $r' = r'.next$
  $r = r'$
return S

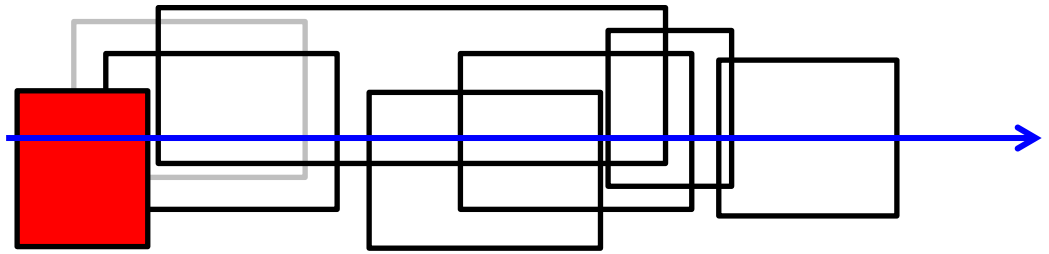
Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Spezialfall:

- Es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.



Algorithmus:

sortiere Rechtecke nach x-Koordinate ihres rechten Randes in Liste L

$S = \emptyset, r = L.head$

while $r \neq nil$ **do**

  $S = S \cup \{r\}$

  $r' = r.next$

  while $r' \neq nil$ **and** $r'.xMin \leq r.xMax$ **do** $r' = r'.next$

  $r = r'$

return S

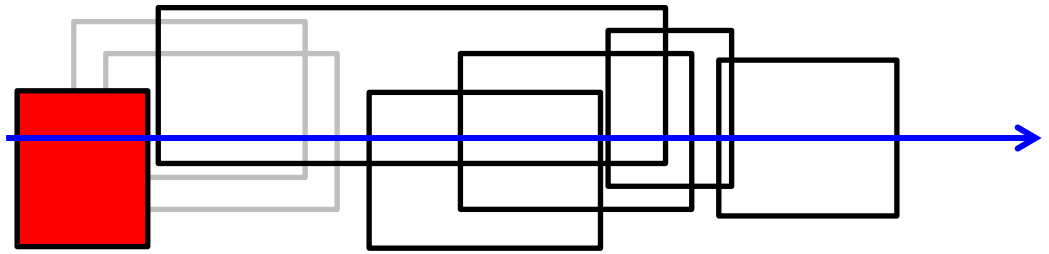
Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Spezialfall:

- Es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.



Algorithmus:

sortiere Rechtecke nach x-Koordinate ihres rechten Randes in Liste L
$S = \emptyset, r = L.head$
while $r \neq nil$ **do**
  $S = S \cup \{r\}$
  $r' = r.next$
  while $r' \neq nil$ **and** $r'.xMin \leq r.xMax$ **do** $r' = r'.next$
  $r = r'$
return S

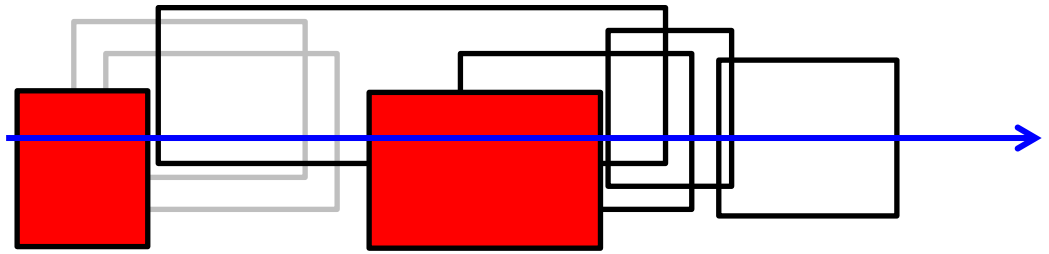
Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Spezialfall:

- Es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.



Algorithmus:

sortiere Rechtecke nach x-Koordinate ihres rechten Randes in Liste L

$S = \emptyset, r = L.head$

while $r \neq nil$ **do**

  $S = S \cup \{r\}$

  $r' = r.next$

  while $r' \neq nil$ **and** $r'.xMin \leq r.xMax$ **do** $r' = r'.next$

  $r = r'$

return S

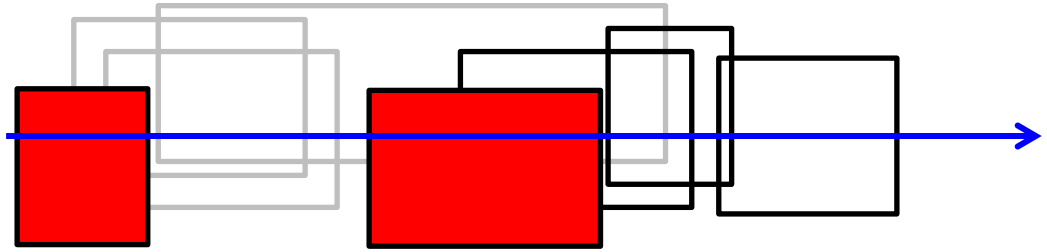
Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Spezialfall:

- Es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.



Algorithmus:

sortiere Rechtecke nach x-Koordinate ihres rechten Randes in Liste L

$S = \emptyset, r = L.head$

while $r \neq nil$ **do**

  $S = S \cup \{r\}$

  $r' = r.next$

  while $r' \neq nil$ **and** $r'.xMin \leq r.xMax$ **do** $r' = r'.next$

  $r = r'$

return S

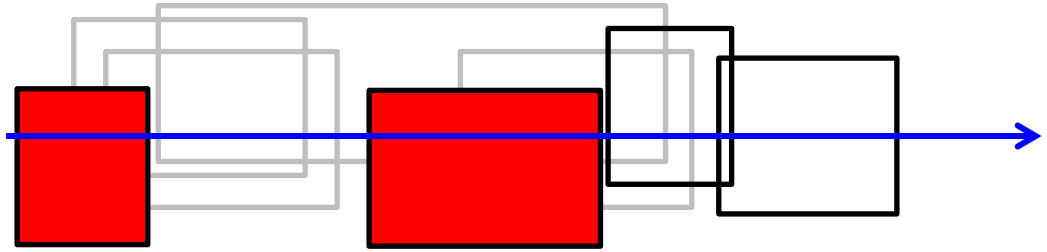
Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Spezialfall:

- Es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.



Algorithmus:

sortiere Rechtecke nach x-Koordinate ihres rechten Randes in Liste L

$S = \emptyset, r = L.head$

while $r \neq nil$ **do**

  $S = S \cup \{r\}$

  $r' = r.next$

  while $r' \neq nil$ **and** $r'.xMin \leq r.xMax$ **do** $r' = r'.next$

  $r = r'$

return S

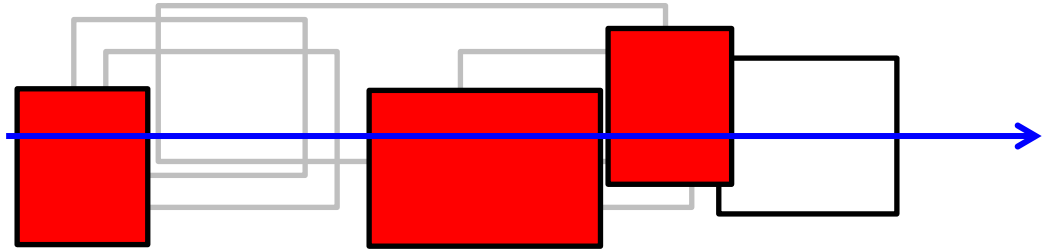
Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Spezialfall:

- Es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.



Algorithmus:

sortiere Rechtecke nach x-Koordinate ihres rechten Randes in Liste L

$S = \emptyset, r = L.head$

while $r \neq nil$ **do**

  $S = S \cup \{r\}$

  $r' = r.next$

  while $r' \neq nil$ **and** $r'.xMin \leq r.xMax$ **do** $r' = r'.next$

  $r = r'$

return S

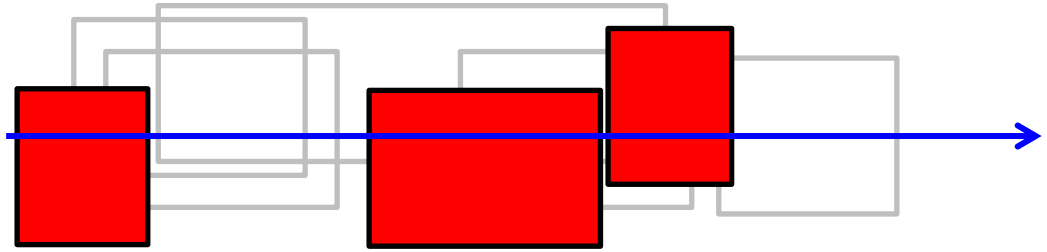
Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Spezialfall:

- Es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.



Algorithmus:

sortiere Rechtecke nach x-Koordinate ihres rechten Randes in Liste L

$S = \emptyset, r = L.head$

while $r \neq nil$ **do**

  $S = S \cup \{r\}$

  $r' = r.next$

  while $r' \neq nil$ **and** $r'.xMin \leq r.xMax$ **do** $r' = r'.next$

  $r = r'$

return S

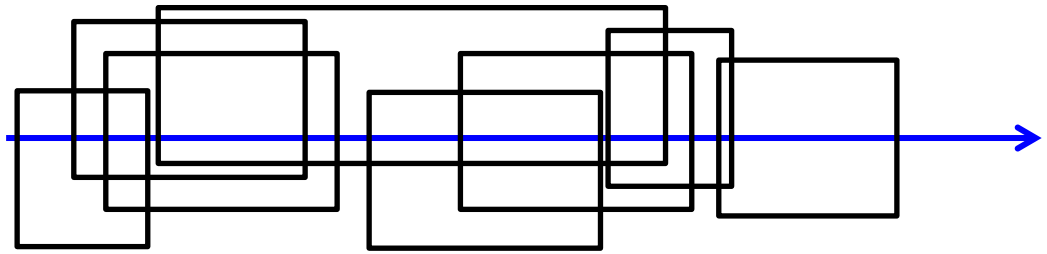
Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Spezialfall:

- Es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.



Algorithmus:

Greedy!

Laufzeit:

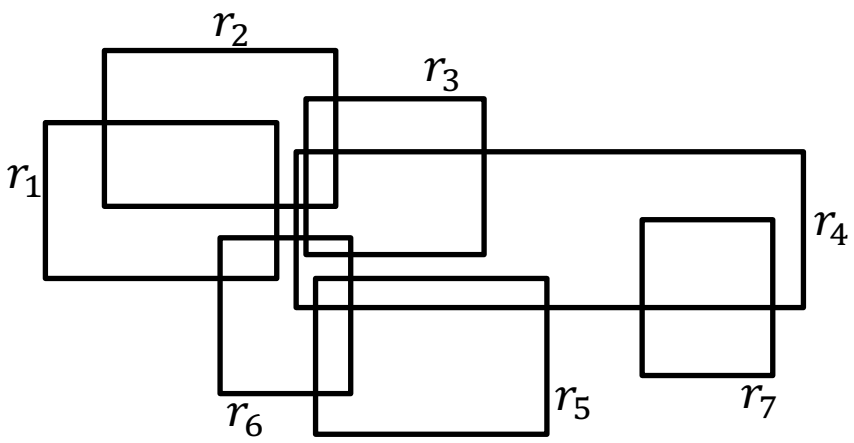
- $O(n \log n)$
- bzw. $O(n)$, wenn Eingabe bereits geordnet

Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Allgemeinfall:

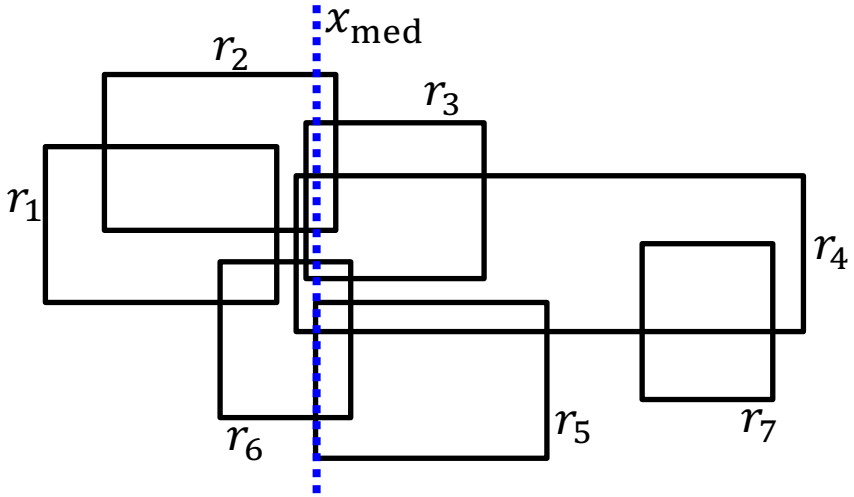


Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Allgemeinfall:



Definitionen:

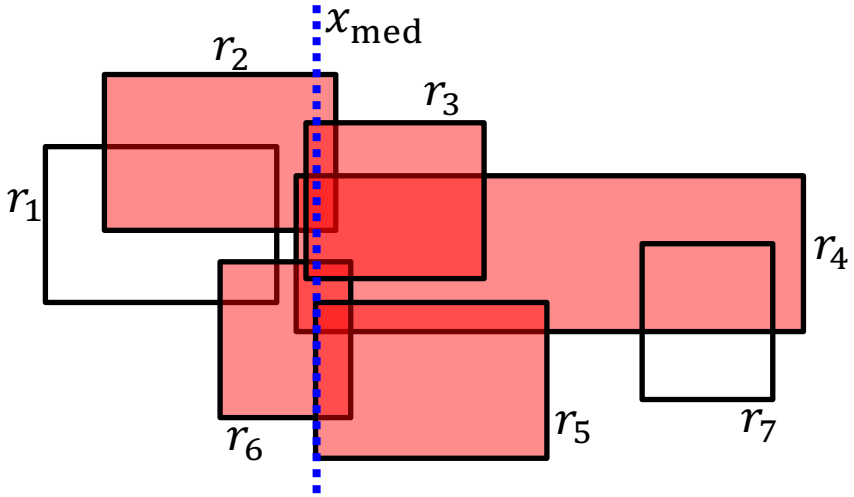
- Sei x_{med} der Median von $\{r.xMin, r.xMax \mid r \in R\}$.

Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Allgemeinfall:



Definitionen:

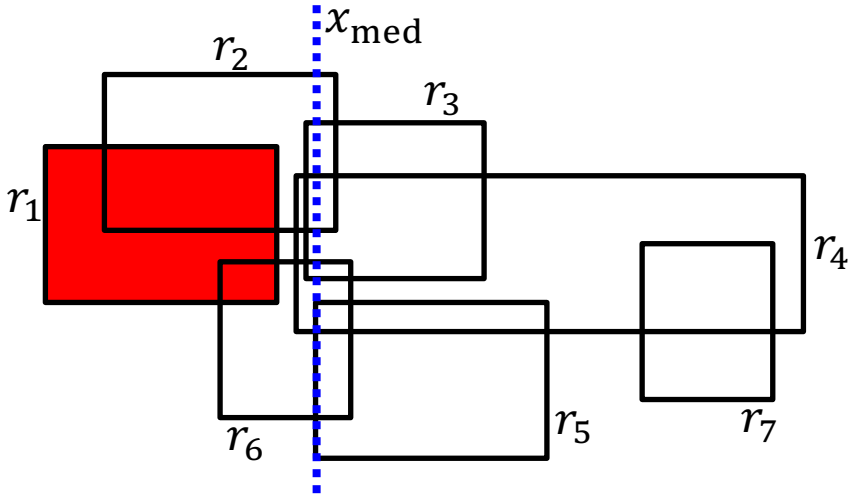
- Sei x_{med} der Median von $\{r.xMin, r.xMax \mid r \in R\}$.
- Sei $R_{12} \subseteq R$ die Menge der Rechtecke auf der Vertikalen durch x_{med}.

Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Allgemeinfall:



Definitionen:

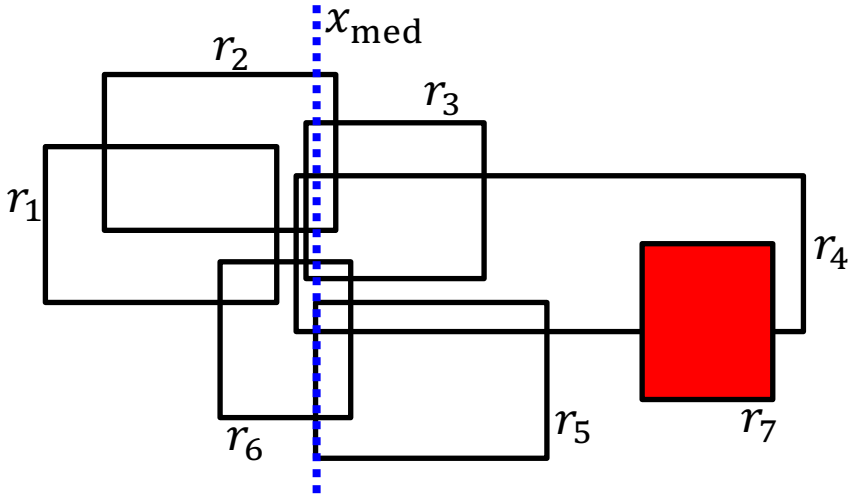
- Sei x_{med} der Median von $\{r.xMin, r.xMax \mid r \in R\}$.
- Sei $R_{12} \subseteq R$ die Menge der Rechtecke auf der Vertikalen durch x_{med}.
- Sei $R_1 \subseteq R$ die Menge der Rechtecke links der Vertikalen durch x_{med}.

Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Allgemeinfall:



Definitionen:

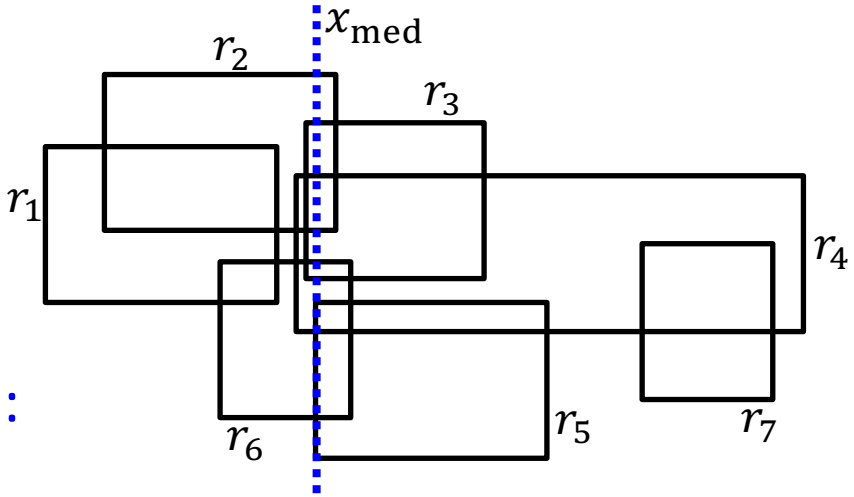
- Sei x_{med} der Median von $\{r.xMin, r.xMax \mid r \in R\}$.
- Sei $R_{12} \subseteq R$ die Menge der Rechtecke auf der Vertikalen durch x_{med}.
- Sei $R_1 \subseteq R$ die Menge der Rechtecke links der Vertikalen durch x_{med}.
- Sei $R_2 \subseteq R$ die Menge der Rechtecke rechts der Vertikalen durch x_{med}

Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Allgemeinfall:



Algorithmus (für $n \leq 2$):

if $n = 0$ **then return** $\emptyset$

if $n = 1$ **or** $r_1 \cap r_2 \neq \emptyset$ **then return** $\{r_1\}$

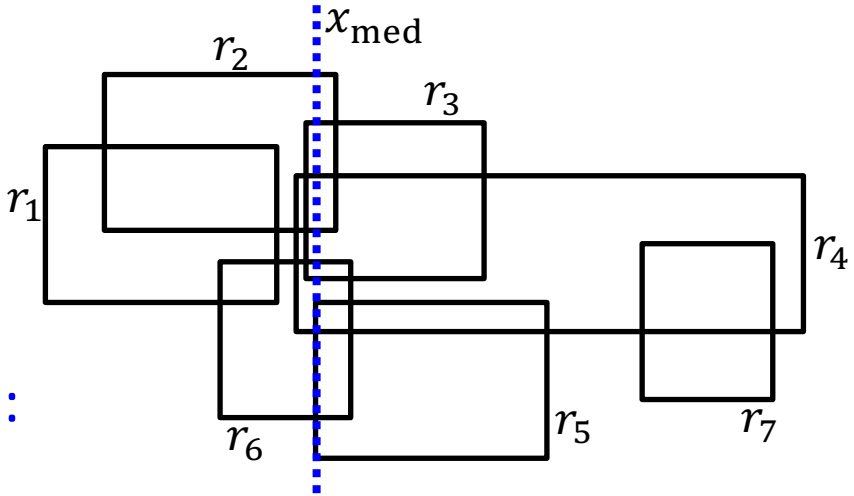
return $\{r_1, r_2\}$

Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Allgemeinfall:



Algorithmus (für $n > 2$):

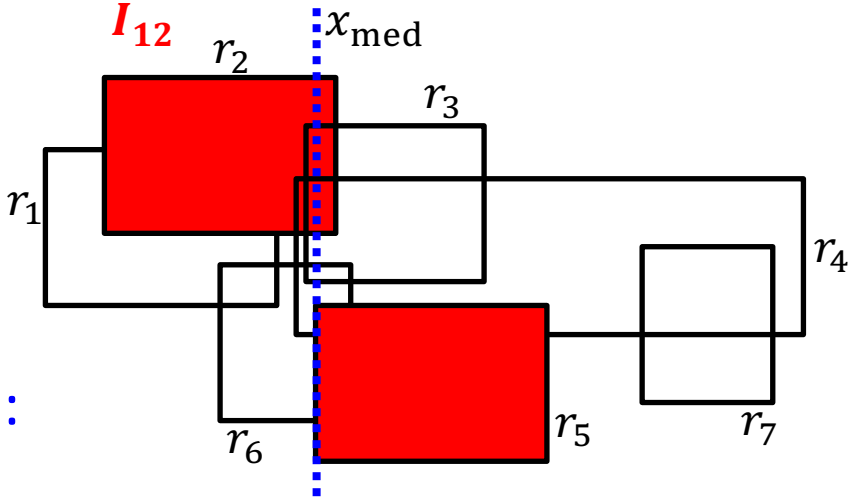
Berechne R_{12}, R_1, R_2
Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$
Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$
Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$
if $|I_{12}| \geq |I_1| + |I_2|$ then
  return I_{12}
else
  return $I_1 \cup I_2$

Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Allgemeinfall:



Algorithmus (für $n > 2$):

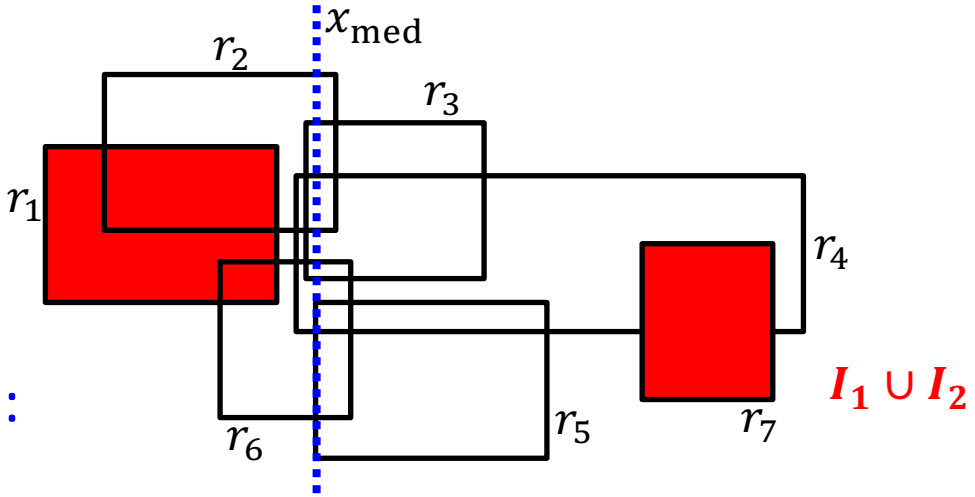
Berechne R_{12}, R_1, R_2
Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$
Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$
Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$
if $|I_{12}| \geq |I_1| + |I_2|$ then
  return I_{12}
else
  return $I_1 \cup I_2$

Beschriftung von Punkten

Problem:

- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R, die sich nicht schneiden

Allgemeinfall:



Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2
Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$
Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$
Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$
if $|I_{12}| \geq |I_1| + |I_2|$ then
  return I_{12}
else
  return $I_1 \cup I_2$

Beschriftung von Punkten

Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2

Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$

Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$

Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$

if $|I_{12}| \geq |I_1| + |I_2|$ then

  return I_{12}

else

  return $I_1 \cup I_2$

Beschriftung von Punkten

Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2
Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$
Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$
Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$
if $|I_{12}| \geq |I_1| + |I_2|$ then
  return I_{12}
else
  return $I_1 \cup I_2$

Behauptung: Der Algorithmus liefert eine Lösung I mit $|I| \geq |I^*|/\log |R|$,
wobei I^* eine optimale Lösung ist.

Beschriftung von Punkten

Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2

Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$

Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$

Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$

if $|I_{12}| \geq |I_1| + |I_2|$ then

  return I_{12}

else

  return $I_1 \cup I_2$

Behauptung: Der Algorithmus liefert eine Lösung I mit $|I| \geq |I^*|/\log |R|$, wobei I^* eine optimale Lösung ist.

Beweis durch Induktion:

Behauptung stimmt für $|R| \leq 2$

Zu zeigen:

Wenn Behauptung für $|R| < n$ stimmt, dann auch für $|R| = n$

Beschriftung von Punkten

Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2

Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$

Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$

Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$

if $|I_{12}| \geq |I_1| + |I_2|$ then

  return I_{12}

else

  return $I_1 \cup I_2$

Beweis durch Induktion:

I_{12}^*, I_1^*, I_2^* seien optimale Lösungen für R_{12}, R_1, R_2

$\left|I_{12}\right| = |I_{12}^*| \geq |I^* \cap R_{12}|$

Beschriftung von Punkten

Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2

Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$

Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$

Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$

if $|I_{12}| \geq |I_1| + |I_2|$ then

  return I_{12}

else

  return $I_1 \cup I_2$

Beweis durch Induktion:

I_{12}^*, I_1^*, I_2^* seien optimale Lösungen für R_{12}, R_1, R_2

$|I_{12}| = |I_{12}^*| \geq |I^* \cap R_{12}|$

$|I_1| \geq |I_1^*|/\log(n/2) \geq |I^* \cap R_1|/(\log n - 1)$

wegen $|R_1| \leq n/2$ und Induktionsannahme

Beschriftung von Punkten

Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2

Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$

Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$

Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$

if $|I_{12}| \geq |I_1| + |I_2|$ then

  return I_{12}

else

  return $I_1 \cup I_2$

Beweis durch Induktion:

I_{12}^*, I_1^*, I_2^* seien optimale Lösungen für R_{12}, R_1, R_2

$|I_{12}| = |I_{12}^*| \geq |I^* \cap R_{12}|$

$|I_1| \geq |I^* \cap R_1|/(\log n - 1)$

Beschriftung von Punkten

Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2

Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$

Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$

Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$

if $|I_{12}| \geq |I_1| + |I_2|$ then

  return I_{12}

else

  return $I_1 \cup I_2$

Beweis durch Induktion:

I_{12}^*, I_1^*, I_2^* seien optimale Lösungen für R_{12}, R_1, R_2

$|I_{12}| = |I_{12}^*| \geq |I^* \cap R_{12}|$

$|I_1| \geq |I^* \cap R_1|/(\log n - 1)\,, \;\; |I_2| \geq |I^* \cap R_2|/(\log n - 1)$

Beschriftung von Punkten

Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2

Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$

Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$

Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$

if $|I_{12}| \geq |I_1| + |I_2|$ then

  return I_{12}

else

  return $I_1 \cup I_2$

Beweis durch Induktion:

I_{12}^*, I_1^*, I_2^* seien optimale Lösungen für R_{12}, R_1, R_2

$|I_{12}| = |I_{12}^*| \geq |I^* \cap R_{12}|$

$|I_1| \geq |I^* \cap R_1|/(\log n - 1)\,, \;\; |I_2| \geq |I^* \cap R_2|/(\log n - 1)$

$|I| = \max\{|I_{12}|, |I_1| + |I_2|\}$

Beschriftung von Punkten

Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2
Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$
Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$
Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$
if $|I_{12}| \geq |I_1| + |I_2|$ then
  return I_{12}
else
  return $I_1 \cup I_2$

Beweis durch Induktion:

I_{12}^*, I_1^*, I_2^* seien optimale Lösungen für R_{12}, R_1, R_2

$|I_{12}| = |I_{12}^*| \geq |I^* \cap R_{12}|$

$|I_1| \geq |I^* \cap R_1|/(\log n - 1)\,, \;\; |I_2| \geq |I^* \cap R_2|/(\log n - 1)$

$$|I| = \max\{|I_{12}|, |I_1| + |I_2|\} \geq \max\left\{|I^* \cap R_{12}|, \frac{|I^* \cap R_1| + |I^* \cap R_2|}{\log n - 1}\right\}$$

Beschriftung von Punkten

Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2

Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$

Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$

Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$

if $|I_{12}| \geq |I_1| + |I_2|$ then

  return I_{12}

else

  return $I_1 \cup I_2$

Beweis durch Induktion:

I_{12}^*, I_1^*, I_2^* seien optimale Lösungen für R_{12}, R_1, R_2

$|I_{12}| = |I_{12}^*| \geq |I^* \cap R_{12}|$

$|I_1| \geq |I^* \cap R_1|/(\log n - 1)\,, \;\; |I_2| \geq |I^* \cap R_2|/(\log n - 1)$

$$|I| = \max\{|I_{12}|, |I_1| + |I_2|\} \geq \max\left\{|I^* \cap R_{12}|, \frac{|I^* \cap R_1| + |I^* \cap R_2|}{\log n - 1}\right\}$$

$|I^*| - |I^* \cap R_{12}|$

Beschriftung von Punkten

Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2
Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$
Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$
Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$
if $|I_{12}| \geq |I_1| + |I_2|$ then
  return I_{12}
else
  return $I_1 \cup I_2$

Beweis durch Induktion:

I_{12}^*, I_1^*, I_2^* seien optimale Lösungen für R_{12}, R_1, R_2

$|I_{12}| = |I_{12}^*| \geq |I^* \cap R_{12}|$

$|I_1| \geq |I^* \cap R_1|/(\log n - 1)\,, \;\; |I_2| \geq |I^* \cap R_2|/(\log n - 1)$

$$|I| = \max\{|I_{12}|, |I_1| + |I_2|\} \geq \max\left\{|I^* \cap R_{12}|, \frac{|I^*| - |I^* \cap R_{12}|}{\log n - 1}\right\}$$

Beschriftung von Punkten

Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2
Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$
Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$
Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$
if $|I_{12}| \geq |I_1| + |I_2|$ then
  return I_{12}
else
  return $I_1 \cup I_2$

Beweis durch Induktion:

I_{12}^*, I_1^*, I_2^* seien optimale Lösungen für R_{12}, R_1, R_2

$$|I| = \max\{|I_{12}|, |I_1| + |I_2|\} \geq \max\left\{|I^* \cap R_{12}|, \frac{|I^*| - |I^* \cap R_{12}|}{\log n - 1}\right\}$$

Beschriftung von Punkten

Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2

Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$

Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$

Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$

if $|I_{12}| \geq |I_1| + |I_2|$ then

  return I_{12}

else

  return $I_1 \cup I_2$

Beweis durch Induktion:

I_{12}^*, I_1^*, I_2^* seien optimale Lösungen für R_{12}, R_1, R_2

$$|\boldsymbol{I}| = \max\{|I_{12}|, |I_1| + |I_2|\} \geq \max\left\{|\boldsymbol{I^* \cap R_{12}}|, \frac{|I^*| - |I^* \cap R_{12}|}{\log n - 1}\right\}$$

Fall $|\boldsymbol{I^* \cap R_{12}}| \geq |\boldsymbol{I^*}|/\boldsymbol{\log n}$:

$$|I| \geq |I^*|/\log n$$

Beschriftung von Punkten

Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2

Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$

Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$

Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$

if $|I_{12}| \geq |I_1| + |I_2|$ then

  return I_{12}

else

  return $I_1 \cup I_2$

Beweis durch Induktion:

I_{12}^*, I_1^*, I_2^* seien optimale Lösungen für R_{12}, R_1, R_2

$$|\boldsymbol{I}| = \max\{|I_{12}|, |I_1| + |I_2|\} \geq \max\left\{|I^* \cap R_{12}|, \frac{|\boldsymbol{I^*}| - |\boldsymbol{I^*} \cap \boldsymbol{R_{12}}|}{\boldsymbol{\log n - 1}}\right\}$$

Fall $|\boldsymbol{I^*} \cap \boldsymbol{R_{12}}| < |\boldsymbol{I^*}|/\boldsymbol{\log n}$:

$|I| \geq$

Beschriftung von Punkten

Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2
Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$
Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$
Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$
if $|I_{12}| \geq |I_1| + |I_2|$ then
  return I_{12}
else
  return $I_1 \cup I_2$

Beweis durch Induktion:

I_{12}^*, I_1^*, I_2^* seien optimale Lösungen für R_{12}, R_1, R_2

$$|\boldsymbol{I}| = \max\{|I_{12}|, |I_1| + |I_2|\} \geq \max\left\{|I^* \cap R_{12}|, \frac{|\boldsymbol{I^*}| - |\boldsymbol{I^*} \cap \boldsymbol{R_{12}}|}{\boldsymbol{\log n - 1}}\right\}$$

Fall $|\boldsymbol{I^*} \cap \boldsymbol{R_{12}}| < |\boldsymbol{I^*}|/\boldsymbol{\log n}$:

$$|I| \geq \frac{|I^*| - |I^*|/\log n}{\log n - 1} =$$

Beschriftung von Punkten

Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2
Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$
Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$
Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$
if $|I_{12}| \geq |I_1| + |I_2|$ then
  return I_{12}
else
  return $I_1 \cup I_2$

Beweis durch Induktion:

I_{12}^*, I_1^*, I_2^* seien optimale Lösungen für R_{12}, R_1, R_2

$$|\boldsymbol{I}| = \max\{|I_{12}|, |I_1| + |I_2|\} \geq \max\left\{|I^* \cap R_{12}|, \frac{|\boldsymbol{I^*}| - |\boldsymbol{I^*} \cap \boldsymbol{R_{12}}|}{\boldsymbol{\log n - 1}}\right\}$$

Fall $|\boldsymbol{I^*} \cap \boldsymbol{R_{12}}| < |\boldsymbol{I^*}|/\boldsymbol{\log n}$:

$$|I| \geq \frac{|I^*| - |I^*|/\log n}{\log n - 1} = \frac{|I^*|}{\log n}, \text{ denn } \log n\left(1 - \frac{1}{\log n}\right) = \log n - 1$$

Beschriftung von Punkten

Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2
Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$
Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$
Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$
if $|I_{12}| \geq |I_1| + |I_2|$ then
  return I_{12}
else
  return $I_1 \cup I_2$

Beweis durch Induktion:

I_{12}^*, I_1^*, I_2^* seien optimale Lösungen für R_{12}, R_1, R_2

Also gilt in beiden Fällen: $|I| \geq \frac{|I^*|}{\log n}$ ☺

Beschriftung von Punkten

Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2
Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$
Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$
Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$
if $|I_{12}| \geq |I_1| + |I_2|$ then
  return I_{12}
else
  return $I_1 \cup I_2$

Laufzeit:

$$T(n) = 2T(n/2) + O(n)$$

↑ Rekursionen

Beschriftung von Punkten

Algorithmus (für $n > 2$):

Berechne R_{12}, R_1, R_2
Löse Problem für R_{12} **optimal** mit Greedy-Algorithmus $\rightarrow I_{12}$
Löse Problem für R_1 durch rekursiven Aufruf des Algorithmus $\rightarrow I_1$
Löse Problem für R_2 durch rekursiven Aufruf des Algorithmus $\rightarrow I_2$
if $|I_{12}| \geq |I_1| + |I_2|$ then
  return I_{12}
else
  return $I_1 \cup I_2$

Laufzeit:

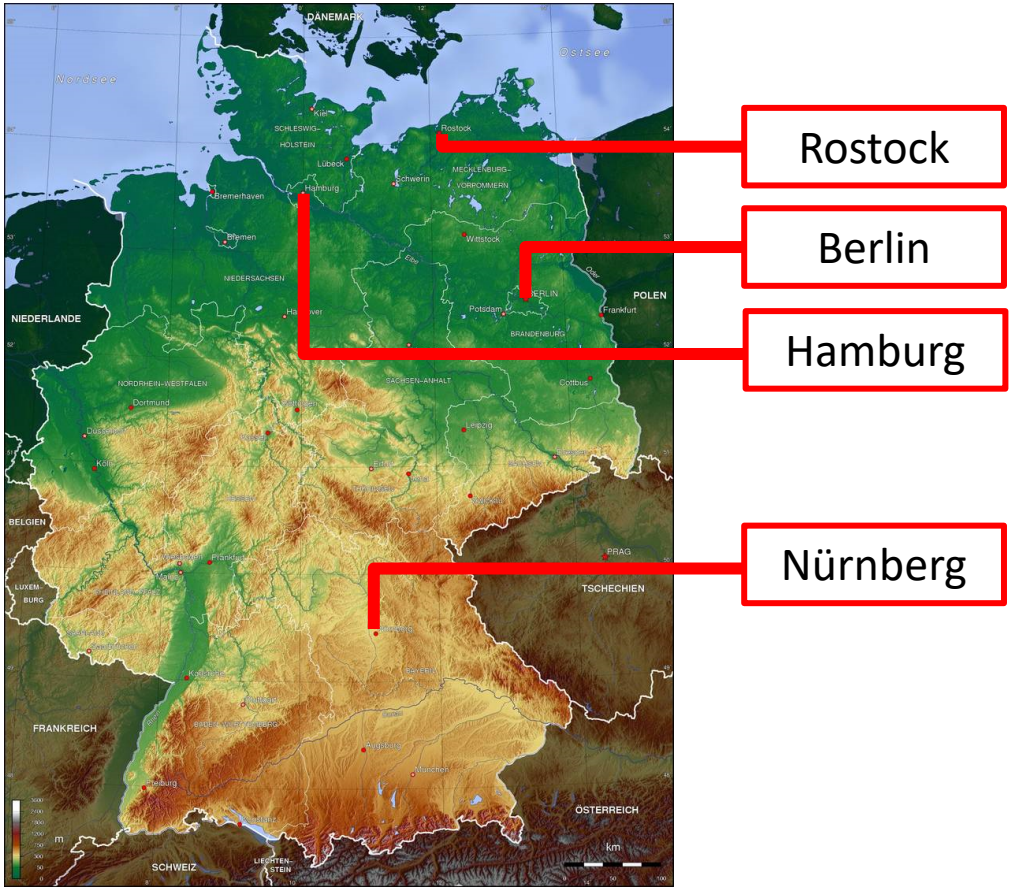
$$T(n) = 2T(n/2) + O(n) = O(n \log n)$$

Berechnung von R_{12}, R_1, R_2
Greedy-Algorithmus (Sortierung nur am Anfang erforderlich)

Boundary Labeling

po-Leader

(**p**arallel + **o**rthogonal)



Rostock

Berlin

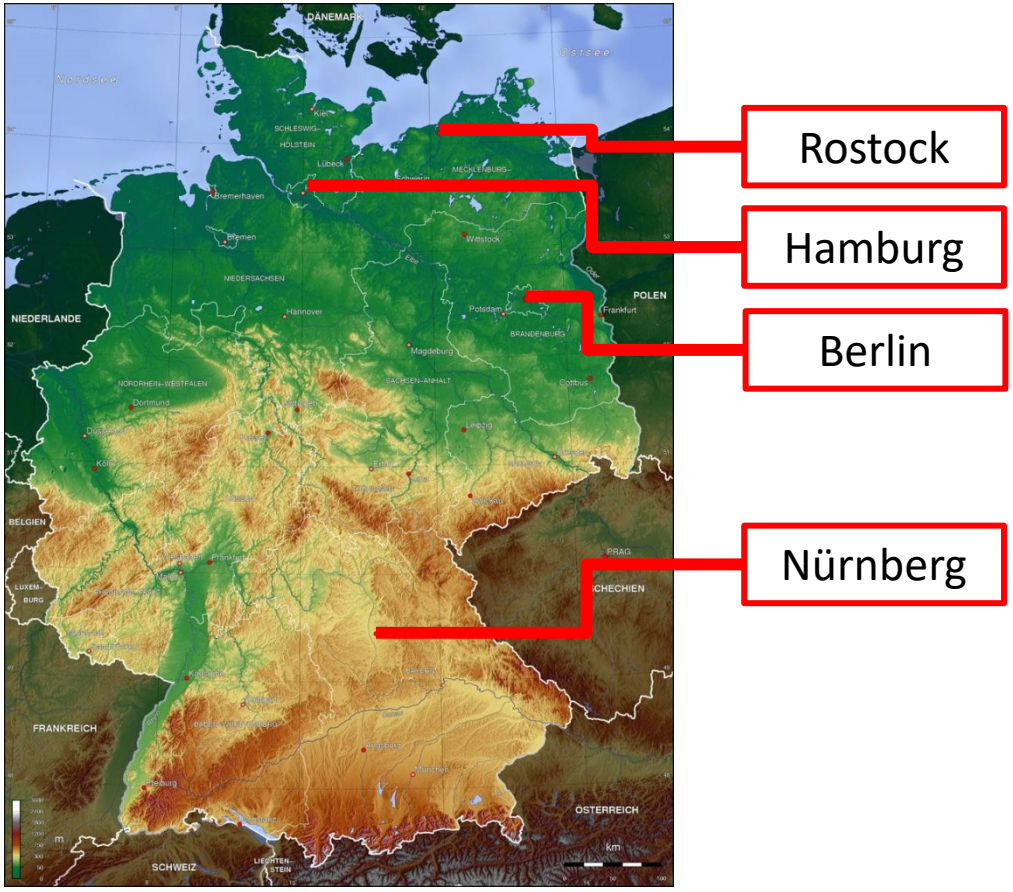
Hamburg

Nürnberg

Boundary Labeling

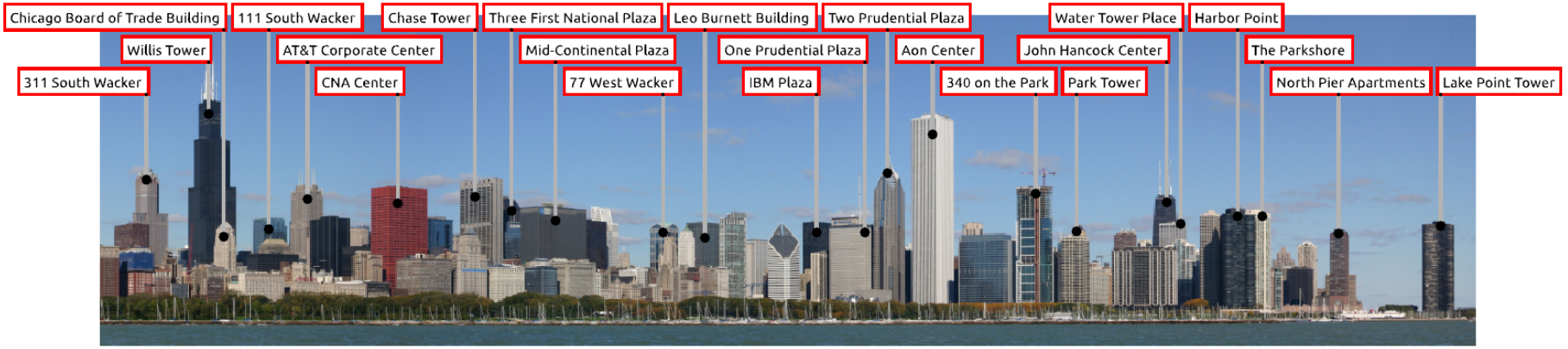
opo-Leader

(**o**rthogonal + **p**arallel + **o**rthogonal)

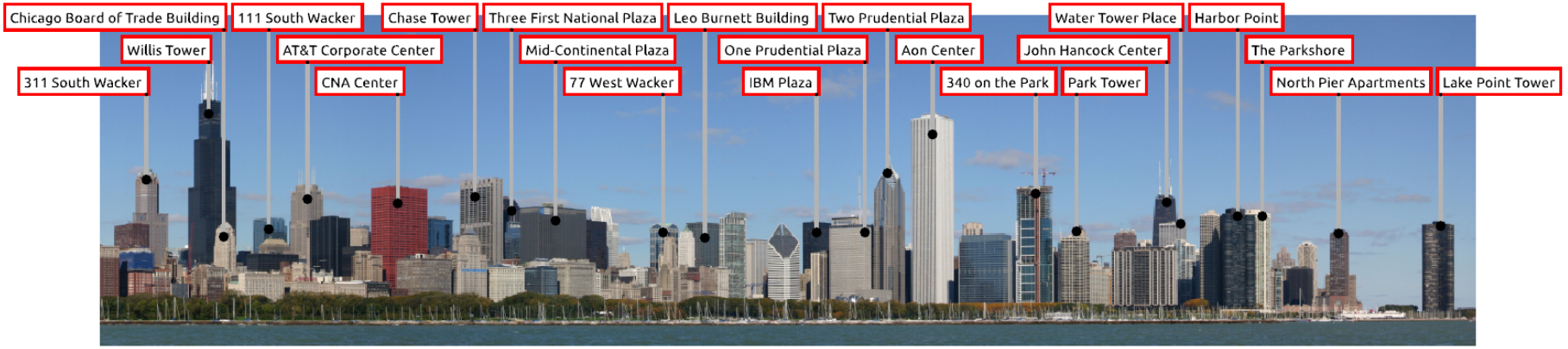


Boundary Labeling mit *o*-Leadern

Jan-Henrik Haunert, Martin Nöllenburg und Andreas Gemsa
ACM Trans. Spatial Algorithms & Syst., 1(1):1–30, 2015



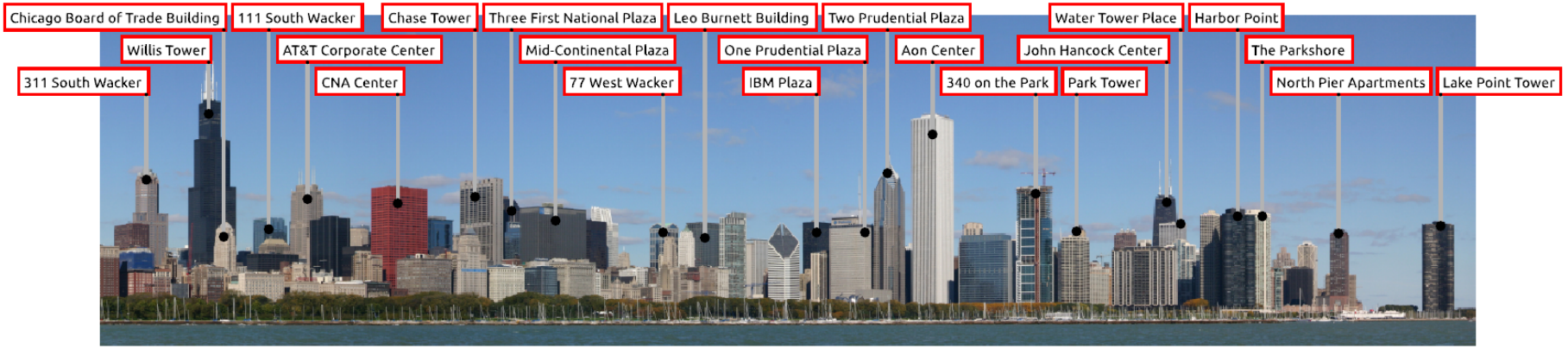
Boundary Labeling mit *o*-Leadern



gegeben:

- eine Menge von Punkten mit x-Koordinaten $x_1, \ldots, x_n$.
- ein Label L_i für jeden Punkt x_i mit Höhe 1 und Breite W_i.

Boundary Labeling mit *o*-Leadern



gegeben:

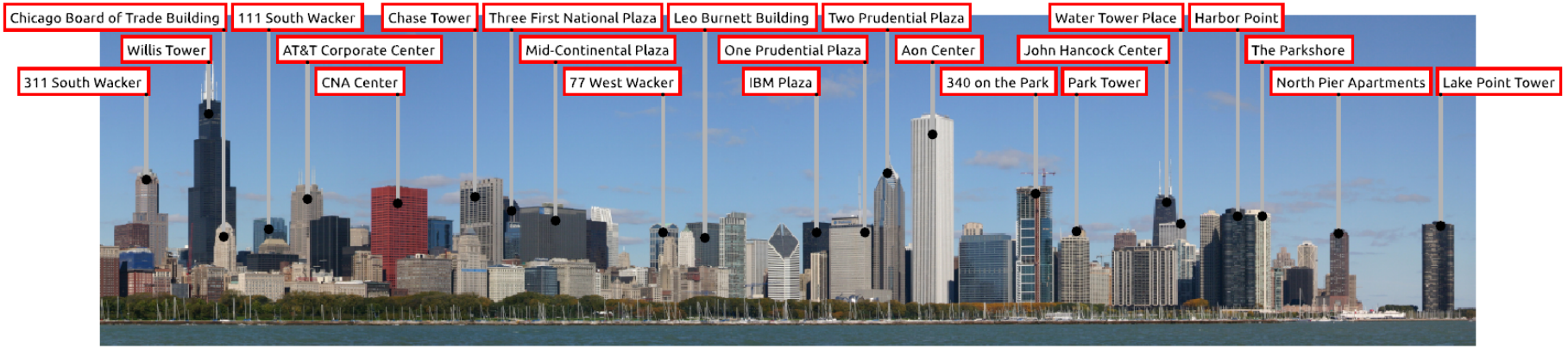
- eine Menge von Punkten mit x-Koordinaten $x_1, \ldots, x_n$.
- ein Label L_i für jeden Punkt x_i mit Höhe 1 und Breite W_i.

gesucht:

- eine x-Koordinate $X_i \in \mathbb{R}$ für jedes Label (bzw. dessen rechten Rand).
- eine y-Koordinate $Y_i \in \mathbb{Z}_0^+$ für jedes Label (bzw. die Zeile des Labels).

Label müssen durch vertikale Leader mit Punkten verbunden werden.
Schnitte von Labeln bzw. zwischen Labeln und Leadern sind verboten!

Boundary Labeling mit *o*-Leadern



gegeben:

- eine Menge von Punkten mit x-Koordinaten $x_1, \ldots, x_n$.
- ein Label L_i für jeden Punkt x_i mit Höhe 1 und Breite W_i.

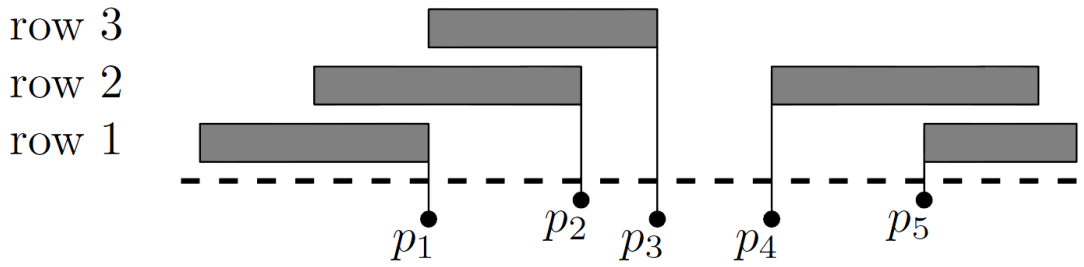
gesucht:

- eine x-Koordinate $X_i \in \mathbb{R}$ für jedes Label (bzw. dessen rechten Rand).
- eine y-Koordinate $Y_i \in \mathbb{Z}_0^+$ für jedes Label (bzw. die Zeile des Labels).

Label müssen durch vertikale Leader mit Punkten verbunden werden.
Schnitte von Labeln bzw. zwischen Labeln und Leadern sind verboten!

Ziel: minimiere die Anzahl der Zeilen

Boundary Labeling mit *o*-Leadern



Es gibt immer eine Lösung mit $\lceil n/2 \rceil$ Zeilen.

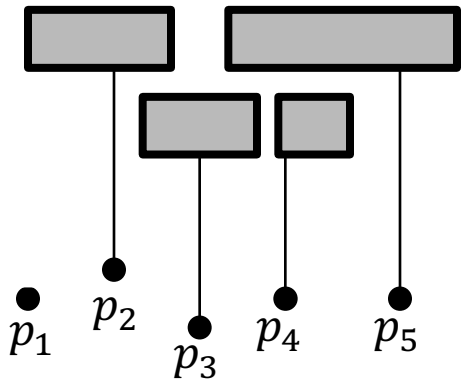
Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Definition:

Ein (i, j, k)-Labeling ist eine Lösung für die Label $L_i, \ldots, L_j$ so dass

- Label L_i und L_j in Zeile k sind und
- Label $L_{i+1}, \ldots, L_{j-1}$ in Zeile k oder darunter sind



Beispiel: ein $(2,5,2)$-Labeling

Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

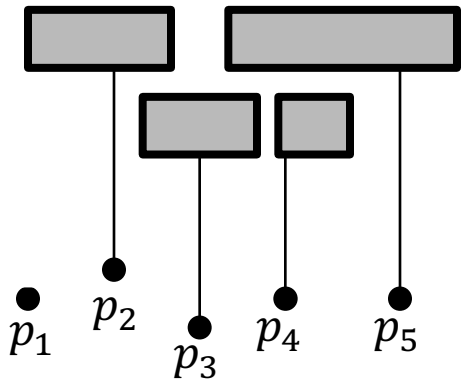
Definition:

Ein (i, j, k)-Labeling ist eine Lösung für die Label $L_i, \ldots, L_j$ so dass

- Label L_i und L_j in Zeile k sind und
- Label $L_{i+1}, \ldots, L_{j-1}$ in Zeile k oder darunter sind

Definition:

Ein (i, j, k)-Labeling ist ***kompakt***, wenn es kein (i, j, k)-Labeling gibt, in dem X_j kleiner ist



Beispiel: ein $(2,5,2)$-Labeling

Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

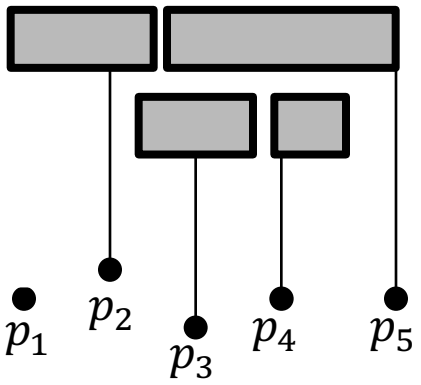
Definition:

Ein (i, j, k)-Labeling ist eine Lösung für die Label $L_i, \ldots, L_j$ so dass

- Label L_i und L_j in Zeile k sind und
- Label $L_{i+1}, \ldots, L_{j-1}$ in Zeile k oder darunter sind

Definition:

Ein (i, j, k)-Labeling ist ***kompakt***, wenn es kein (i, j, k)-Labeling gibt, in dem X_j kleiner ist



Beispiel: ein ***kompaktes*** $(2,5,2)$-Labeling

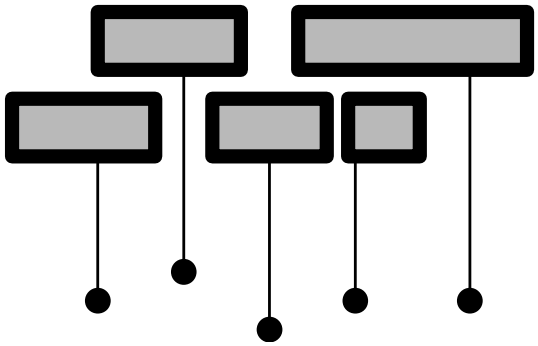
Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Ansatz:

1. Definiere zwei Dummy-Label
   - L_0 mit $x_0 = x_1 - M$ und $W_0 = 0$
   - L_{n+1} mit $x_{n+1} = x_{n+1} + M$ und $W_{n+1} = 0$

   ($M = \max_{i=1,\dots,n} \{W_i\}$, bzw. große Konstante)



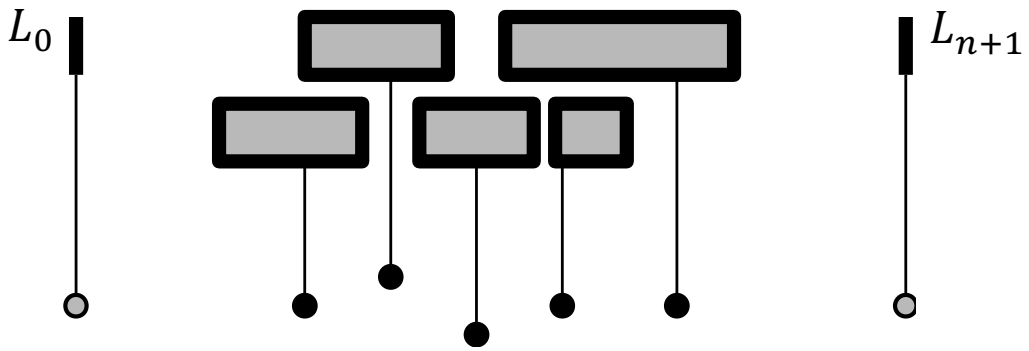
Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Ansatz:

1. Definiere zwei Dummy-Label
   - L_0 mit $x_0 = x_1 - M$ und $W_0 = 0$
   - L_{n+1} mit $x_{n+1} = x_{n+1} + M$ und $W_{n+1} = 0$

   ($M = \max_{i=1,\dots,n} \{W_i\}$, bzw. große Konstante)



Boundary Labeling mit *o*-Leadern

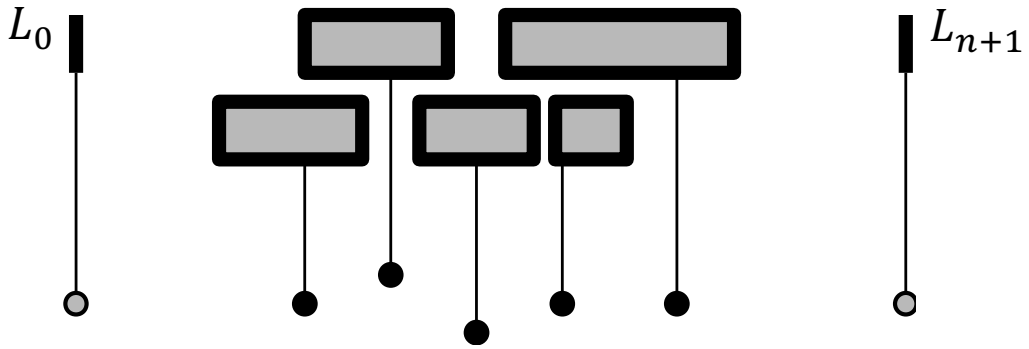
Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Ansatz:

1. Definiere zwei Dummy-Label
   - L_0 mit $x_0 = x_1 - M$ und $W_0 = 0$
   - L_{n+1} mit $x_{n+1} = x_{n+1} + M$ und $W_{n+1} = 0$

   ($M = \max_{i=1,\dots,n} \{W_i\}$, bzw. große Konstante)

2. Für $k = 1, \dots, \lceil n/2 \rceil$ teste, ob es ein $(0, n+1, k)$-Labeling gibt



Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Es gibt ein (i, j, k)-Labeling

- Einfach zu entscheiden, wenn $k = 1$.
- Immer wahr, wenn $i = j$.

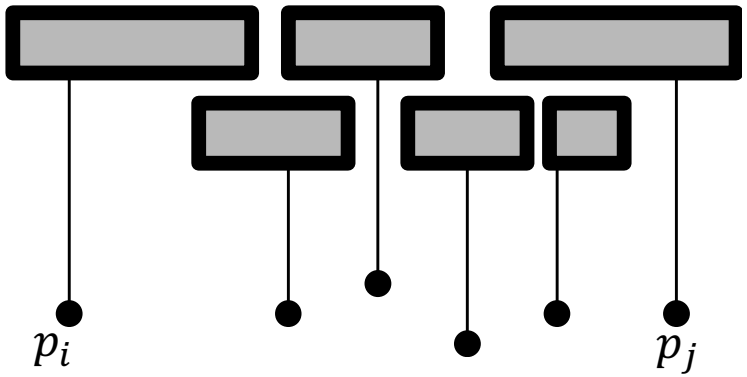
Problem:

- Entscheide für $k > 1$ und $i < j$.

Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Es gibt ein (i, j, k)-Labeling $\Leftrightarrow$



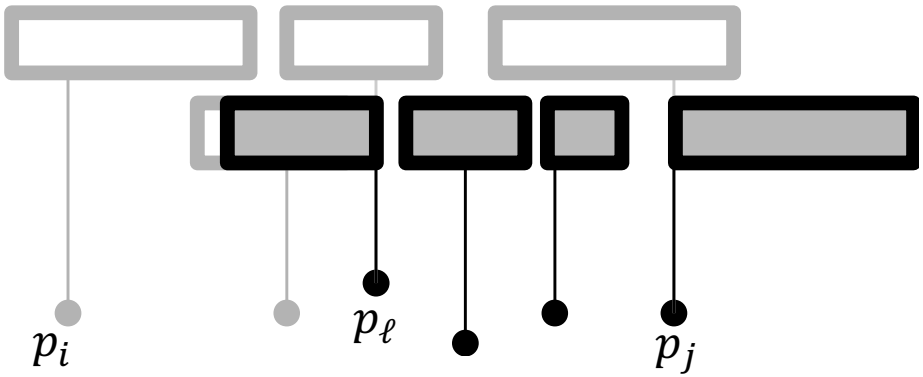
Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Es gibt ein (i, j, k)-Labeling $\Leftrightarrow$

Es gibt ein $i \leq \ell < j$ so dass

- es ein $(\ell, j, k-1)$-Labeling gibt



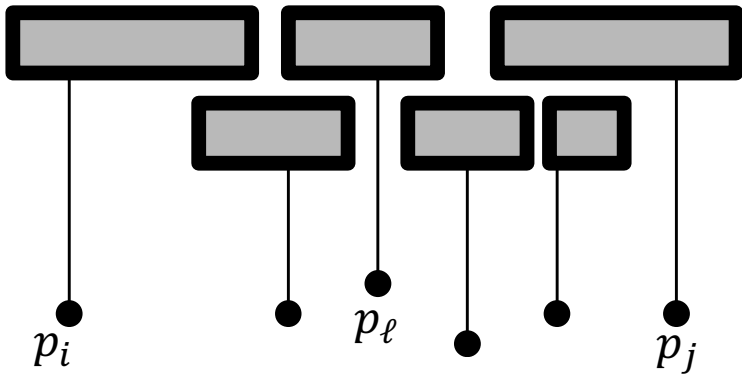
Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Es gibt ein (i, j, k)-Labeling $\Leftrightarrow$

Es gibt ein $i \leq \ell < j$ so dass

- es ein $(\ell, j, k-1)$-Labeling gibt



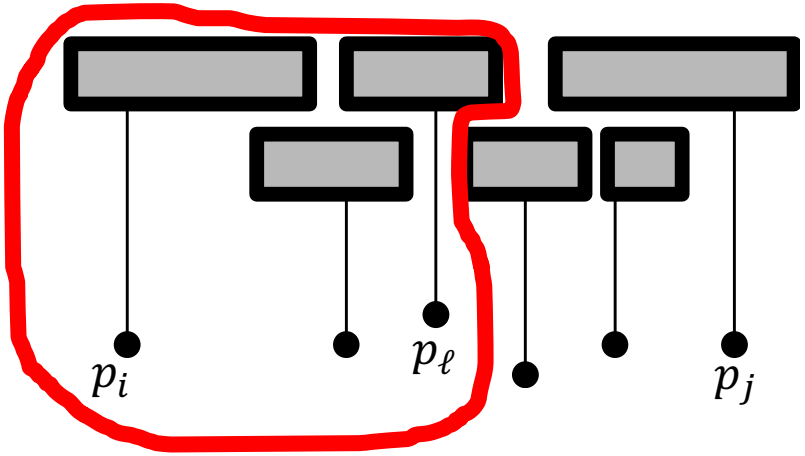
Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Es gibt ein (i, j, k)-Labeling $\Leftrightarrow$

Es gibt ein $i \leq \ell < j$ so dass

- es ein $(\ell, j, k-1)$-Labeling gibt und
- es ein (i, ℓ, k)-Labeling gibt



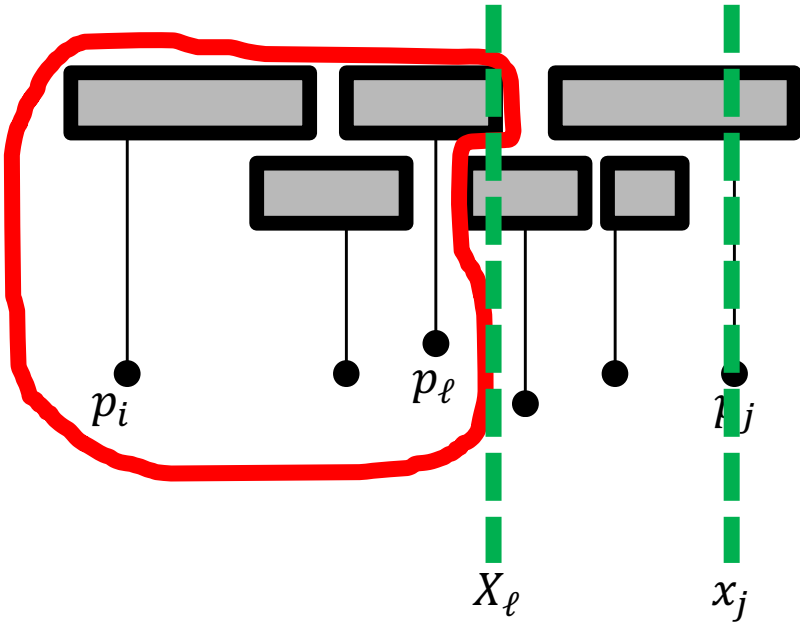
Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Es gibt ein (i, j, k)-Labeling $\Leftrightarrow$

Es gibt ein $i \leq \ell < j$ so dass

- es ein $(\ell, j, k-1)$-Labeling gibt und
- es ein (i, ℓ, k)-Labeling gibt, in dem $X_\ell \leq x_j$



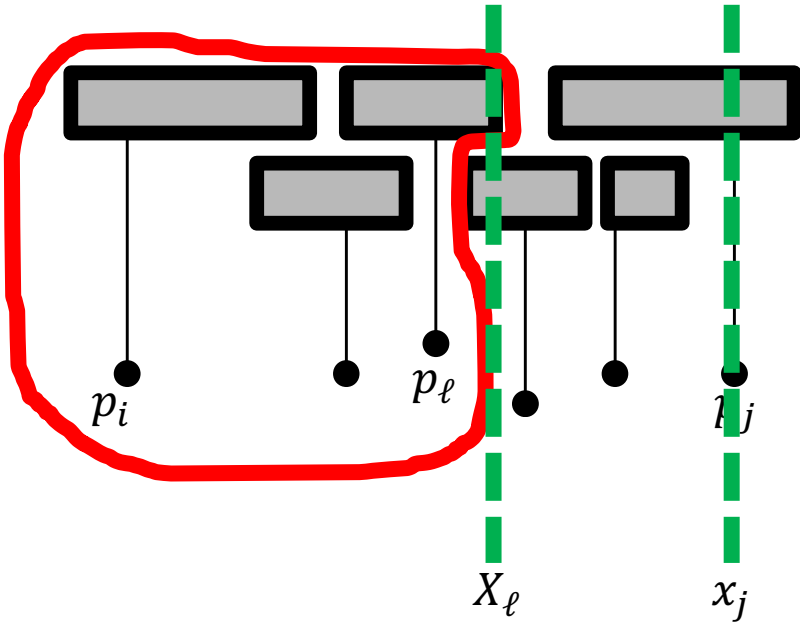
Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Es gibt ein (i, j, k)-Labeling $\Leftrightarrow$

Es gibt ein $i \leq \ell < j$ so dass

- es ein $(\ell, j, k-1)$-Labeling gibt und
- es ein **kompaktes** (i, ℓ, k)-Labeling gibt, in dem $X_\ell \leq x_j$



Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Es gibt ein (i, j, k)-Labeling $\Leftrightarrow$

Es gibt ein $i \leq \ell < j$ so dass

- es ein $(\ell, j, k-1)$-Labeling gibt und
- es ein **kompaktes** (i, ℓ, k)-Labeling gibt, in dem $X_\ell \leq x_j$

Zu berechnen:

$\mathcal{T}[i, j, k] = X_j$ in einem kompakten (i, j, k)-Labeling (∞ wenn es dies nicht gibt).

Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Es gibt ein (i, j, k)-Labeling $\Leftrightarrow$

Es gibt ein $i \leq \ell < j$ so dass

- es ein $(\ell, j, k-1)$-Labeling gibt und
- es ein **kompaktes** (i, ℓ, k)-Labeling gibt, in dem $X_\ell \leq x_j$

Zu berechnen:

$\mathcal{T}[i, j, k] = X_j$ in einem kompakten (i, j, k)-Labeling (∞, wenn es das nicht gibt).

Annahme: Vorgänger L_ℓ von L_j in kompaktem (i, j, k)-Labeling bekannt.

$$\mathcal{T}[i, j, k] = \max\{x_j, \mathcal{T}[i, \ell, k] + W_j\}$$

Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Es gibt ein (i, j, k)-Labeling $\Leftrightarrow$

Es gibt ein $i \leq \ell < j$ so dass

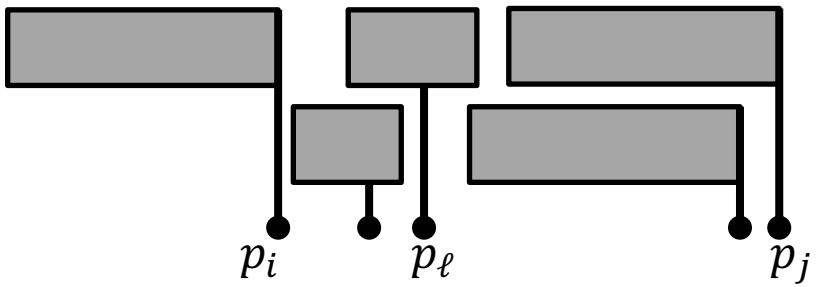
- es ein $(\ell, j, k-1)$-Labeling gibt und
- es ein **kompaktes** (i, ℓ, k)-Labeling gibt, in dem $X_\ell \leq x_j$

Zu berechnen:

$\mathcal{T}[i, j, k] = X_j$ in einem kompakten (i, j, k)-Labeling (∞ wenn es dies nicht gibt).

Annahme: Vorgänger L_ℓ von L_j in kompaktem (i, j, k)-Labeling bekannt.

$$\mathcal{T}[i, j, k] = \max\{x_j, \mathcal{T}[i, \ell, k] + W_j\}$$



$$\mathcal{T}[i, j, k] = x_j$$

Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Es gibt ein (i, j, k)-Labeling $\Leftrightarrow$

Es gibt ein $i \leq \ell < j$ so dass

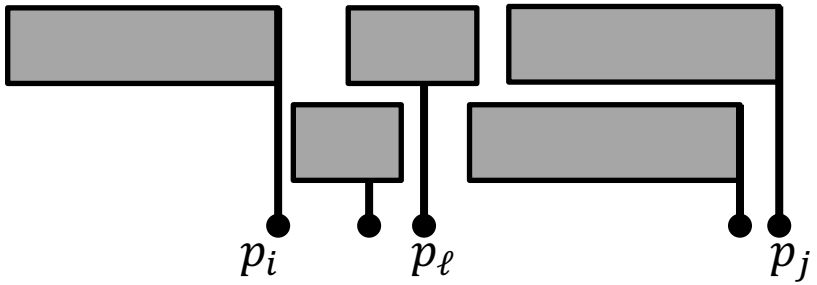
- es ein $(\ell, j, k-1)$-Labeling gibt und
- es ein **kompaktes** (i, ℓ, k)-Labeling gibt, in dem $X_\ell \leq x_j$

Zu berechnen:

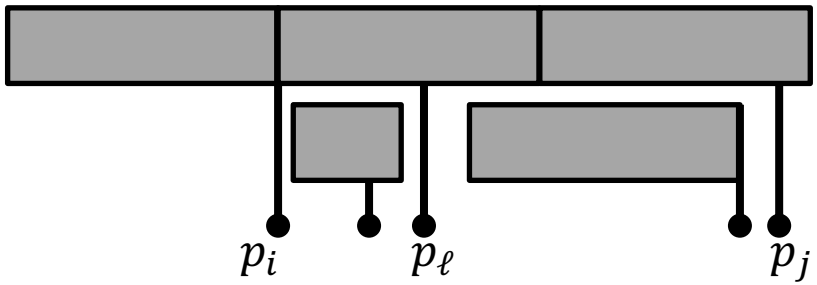
$\mathcal{T}[i, j, k] = X_j$ in einem kompakten (i, j, k)-Labeling (∞ wenn es dies nicht gibt).

Annahme: Vorgänger L_ℓ von L_j in kompaktem (i, j, k)-Labeling bekannt.

$$\mathcal{T}[i, j, k] = \max\{x_j, \mathcal{T}[i, \ell, k] + W_j\}$$



$\mathcal{T}[i, j, k] = x_j$



$\mathcal{T}[i, j, k] = \mathcal{T}[i, \ell, k] + W_j$

Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Es gibt ein (i, j, k)-Labeling $\Leftrightarrow$

Es gibt ein $i \leq \ell < j$ so dass

- es ein $(\ell, j, k-1)$-Labeling gibt und
- es ein **kompaktes** (i, ℓ, k)-Labeling gibt, in dem $X_\ell \leq x_j$

Zu berechnen:

$\mathcal{T}[i, j, k] = X_j$ in einem kompakten (i, j, k)-Labeling (∞ wenn es dies nicht gibt).

Annahme: Vorgänger L_ℓ von L_j in kompaktem (i, j, k)-Labeling bekannt.

$$\mathcal{T}[i, j, k] = \max\{x_j, \mathcal{T}[i, \ell, k] + W_j\}$$

Sonst:

$$\mathcal{T}[i, j, k] = \min\left\{ \max\{x_j, \mathcal{T}[i, \ell, k] + W_j\} \;\middle|\; \begin{array}{c} i \leq \ell < j \\ \mathcal{T}[i, \ell, k] \leq x_j \\ \mathcal{T}[\ell, j, k-1] < \infty \end{array} \right\}$$

Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Es gibt ein (i, j, k)-Labeling $\Leftrightarrow$

Es gibt ein $i \leq \ell < j$ so dass

- es ein $(\ell, j, k-1)$-Labeling gibt und
- es ein **kompaktes** (i, ℓ, k)-Labeling gibt, in dem $X_\ell \leq x_j$

Zu berechnen:

$\mathcal{T}[i, j, k] = X_j$ in einem kompakten (i, j, k)-Labeling (∞ wenn es dies nicht gibt).

Annahme: Vorgänger L_ℓ von L_j in kompaktem (i, j, k)-Labeling bekannt.

$$\mathcal{T}[i, j, k] = \max\{x_j, \mathcal{T}[i, \ell, k] + W_j\}$$

Sonst:

$$\mathcal{T}[i, j, k] = \min\left\{ \max\{x_j, \mathcal{T}[i, \ell, k] + W_j\} \;\middle|\; \begin{array}{c} i \leq \ell < j \\ \mathcal{T}[i, \ell, k] \leq x_j \\ \mathcal{T}[\ell, j, k-1] < \infty \end{array} \right\}$$

Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Es gibt ein (i, j, k)-Labeling $\Leftrightarrow$

Es gibt ein $i \leq \ell < j$ so dass

- es ein $(\ell, j, k-1)$-Labeling gibt und
- es ein **kompaktes** (i, ℓ, k)-Labeling gibt, in dem $X_\ell \leq x_j$

Zu berechnen:

$\mathcal{T}[i, j, k] = X_j$ in einem kompakten (i, j, k)-Labeling (∞ wenn es dies nicht gibt).

Annahme: Vorgänger L_ℓ von L_j in kompaktem (i, j, k)-Labeling bekannt.

$$\mathcal{T}[i, j, k] = \max\{x_j, \mathcal{T}[i, \ell, k] + W_j\}$$

Sonst:

$$\mathcal{T}[i, j, k] = \min\left\{ \max\{x_j, \mathcal{T}[i, \ell, k] + W_j\} \;\middle|\; \begin{array}{c} i \leq \ell < j \\ \mathcal{T}[i, \ell, k] \leq x_j \\ \mathcal{T}[\ell, j, k-1] < \infty \end{array} \right\}$$

Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Es gibt ein (i, j, k)-Labeling $\Leftrightarrow$

Es gibt ein $i \leq \ell < j$ so dass

- es ein $(\ell, j, k-1)$-Labeling gibt und
- es ein **kompaktes** (i, ℓ, k)-Labeling gibt, in dem $X_\ell \leq x_j$

Zu berechnen:

$\mathcal{T}[i, j, k] = X_j$ in einem kompakten (i, j, k)-Labeling (∞ wenn es dies nicht gibt).

Annahme: Vorgänger L_ℓ von L_j in kompaktem (i, j, k)-Labeling bekannt.

$$\mathcal{T}[i, j, k] = \max\{x_j, \mathcal{T}[i, \ell, k] + W_j\}$$

Sonst:

$$\mathcal{T}[i, j, k] = \min\left\{ \max\{x_j, \mathcal{T}[i, \ell, k] + W_j\} \;\middle|\; \begin{array}{c} i \leq \ell < j \\ \mathcal{T}[i, \ell, k] \leq x_j \\ \mathcal{T}[\ell, j, k-1] < \infty \end{array} \right\}$$

Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Algorithmus:

Berechne alle $\mathcal{T}[i,j,1]$ (mit einem einfachen Greedy-Algorithmus)
for $k = 2$ **to** $\lceil n/2 \rceil$ **do**
  for $j = 1$ **to** $n+1$ **do**
    for $i = 0$ **to** $j-1$ **do**
      berechne $\mathcal{T}[i,j,k]$ nach (1)
    $\mathcal{T}[j,j,k] = x_j$
  if $\mathcal{T}[0,n+1,k] < \infty$ **then return** k

$$\mathcal{T}[i,j,k] = \min\left\{ \max\{x_j, \mathcal{T}[i,\ell,k] + W_j\} \;\middle|\; \begin{array}{c} i \le \ell < j \\ \mathcal{T}[i,\ell,k] \le x_j \\ \mathcal{T}[\ell,j,k-1] < \infty \end{array} \right\} \quad (1)$$

Boundary Labeling mit *o*-Leadern

Lösung mit minimaler Zeilenanzahl durch dynamische Programmierung

Algorithmus:

Berechne alle $\mathcal{T}[i,j,1]$ (mit einem einfachen Greedy-Algorithmus)
for $k = 2$ **to** $\lceil n/2 \rceil$ **do**
  for $j = 1$ **to** $n+1$ **do**
    for $i = 0$ **to** $j-1$ **do**
      berechne $\mathcal{T}[i,j,k]$ nach (1)
    $\mathcal{T}[j,j,k] = x_j$
  if $\mathcal{T}[0,n+1,k] < \infty$ **then return** k

Laufzeit: $O(kn^3)$

$$\mathcal{T}[i,j,k] = \min\left\{ \max\{x_j, \mathcal{T}[i,\ell,k] + W_j\} \;\middle|\; \begin{array}{c} i \le \ell < j \\ \mathcal{T}[i,\ell,k] \le x_j \\ \mathcal{T}[\ell,j,k-1] < \infty \end{array} \right\} \quad (1)$$