

Schriftplatzierung

Europe seen by the Germans



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

Schriftplatzierung

Europe seen by the French



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

BUNDESREPUBLIK DEUTSCHLAND

2007

MAINZ = Landeshauptstadt
Fürth = Sonstige Städte
● Städte über 500.000 Einwohner



nach Imhof (1975)

Beschriftung (Namen)
für

- Flächen
- Linien
- Punkte

abhängig vom Maßstab!
(Berlin: Fläche oder Punkt)

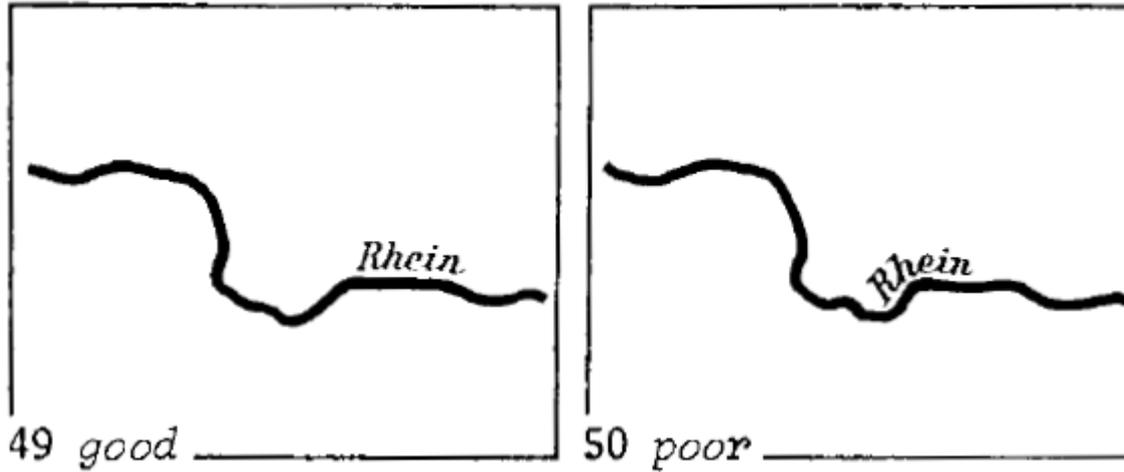
<http://www.worldofmaps.net/>



nach Imhof (1975)

Kriterien für gute Beschriftung

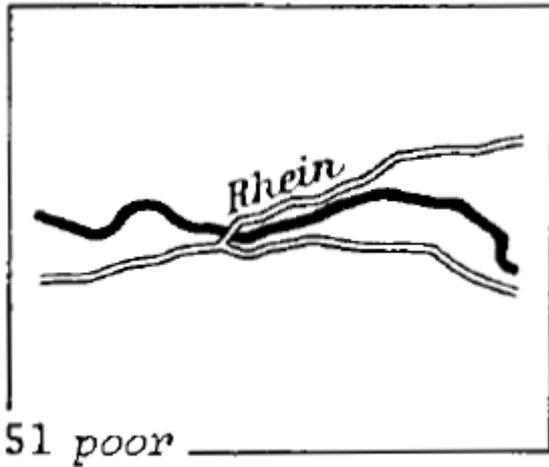
1. Lesbarkeit



nach Imhof (1975)

Kriterien für gute Beschriftung

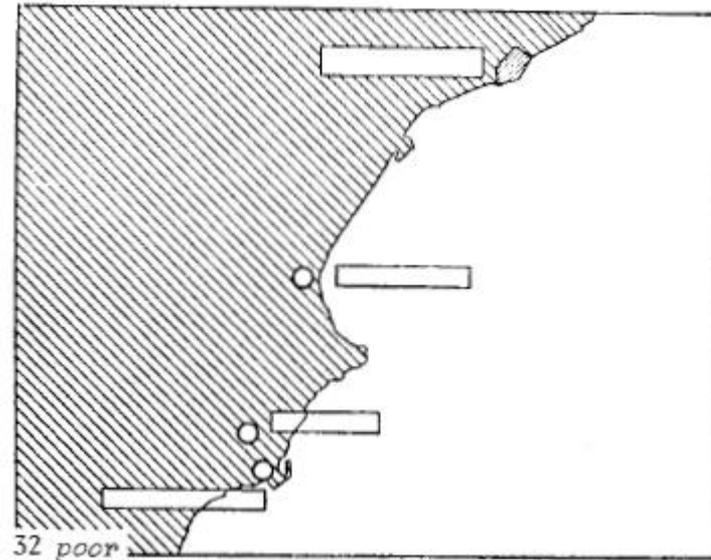
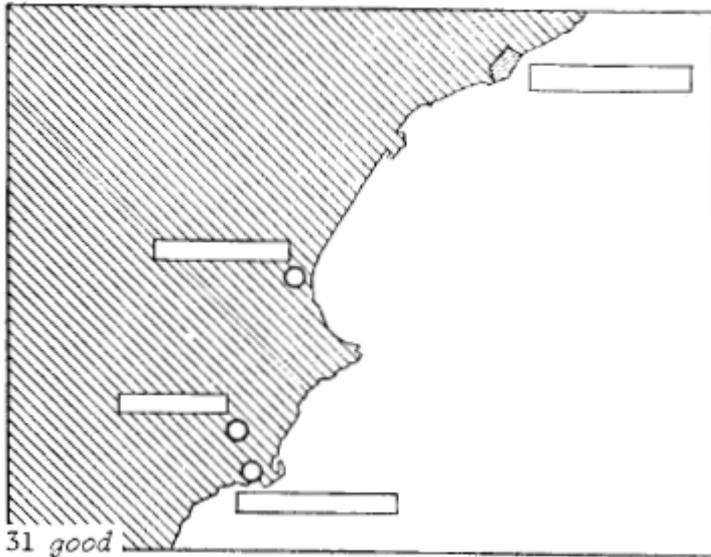
1. Lesbarkeit
2. klare Zuordnung von Namen zu Objekten



nach Imhof (1975)

Kriterien für gute Beschriftung

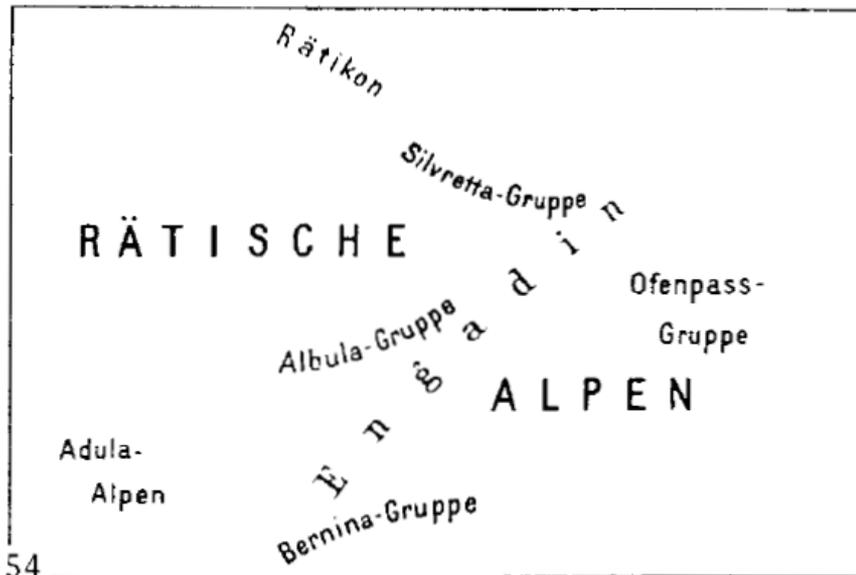
1. Lesbarkeit
2. klare Zuordnung von Namen zu Objekten
3. Namen sollen anderen Karteninhalt wenig stören (keine Verdeckung von Objekten)



nach Imhof (1975)

Kriterien für gute Beschriftung

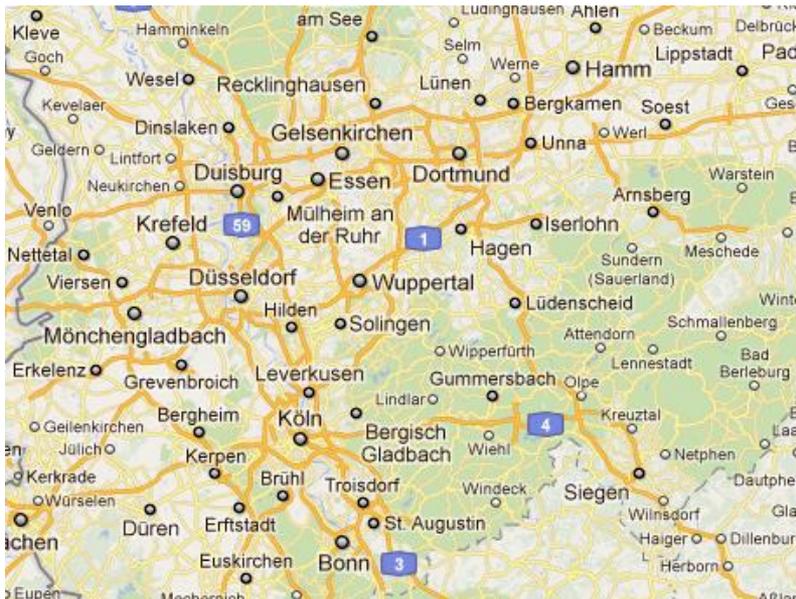
1. Lesbarkeit
2. klare Zuordnung von Namen zu Objekten
3. Namen sollen anderen Karteninhalt wenig stören (keine Verdeckung von Objekten)
4. Namen sollen Verständnis von Kartenobjekten erleichtern.
5. Schrifttyp und -größe sollen Klassen und Hierarchien wiedergeben.



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.

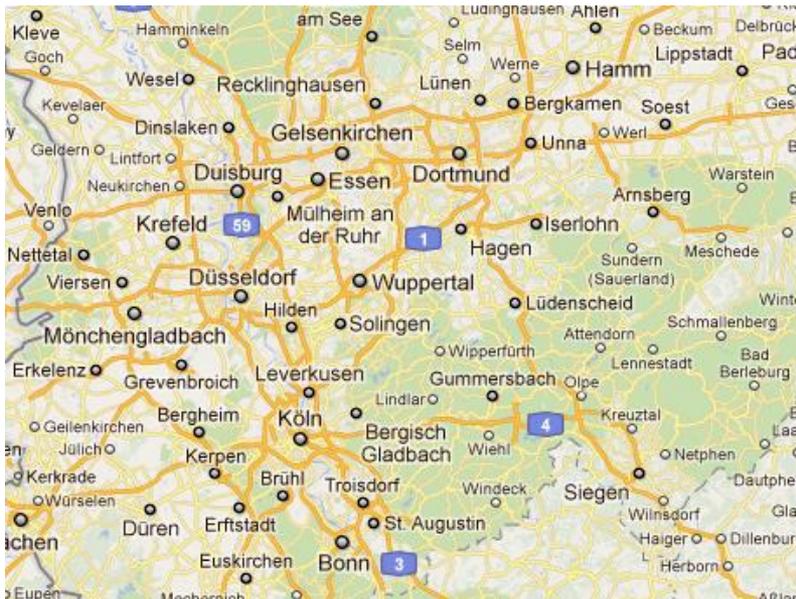


Dichte der Namen soll mit Siedlungsdichte variieren, in Ballungszentren aber nicht zu hoch sein.

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.

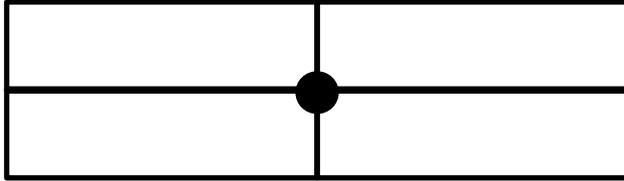


maps.google.com

Textplatzierung:

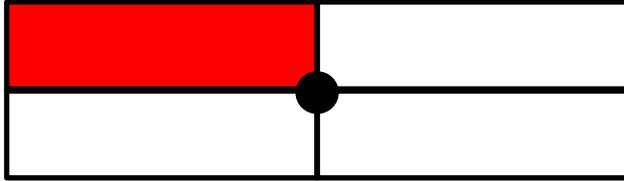
1. Auswahl von Namen
2. Anordnung der Schrift

Beschriftung von Punkten



four-position model

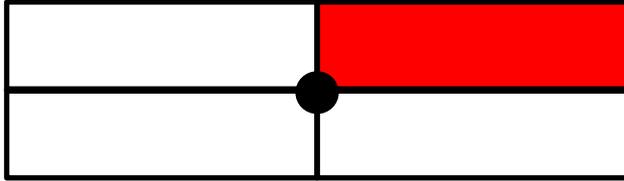
Beschriftung von Punkten



four-position model

Würzburg ●

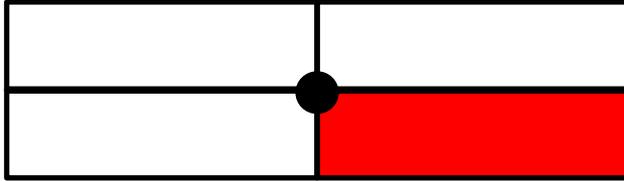
Beschriftung von Punkten



four-position model

Würzburg

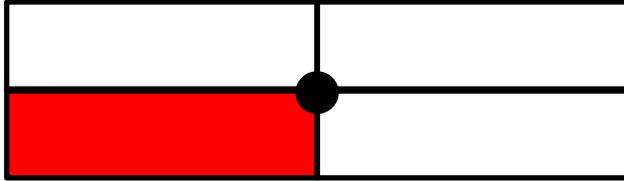
Beschriftung von Punkten



four-position model

● Würzburg

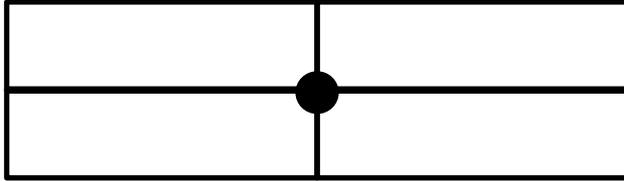
Beschriftung von Punkten



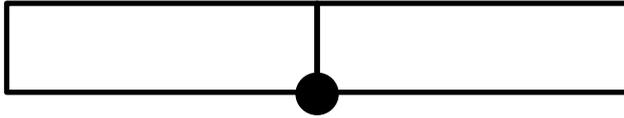
four-position model

Würzburg ●

Beschriftung von Punkten



four-position model



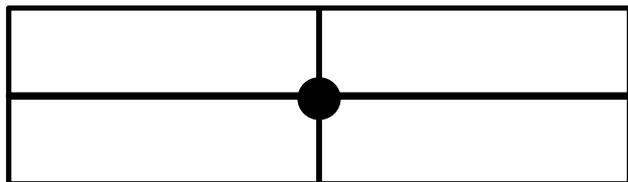
two-position model



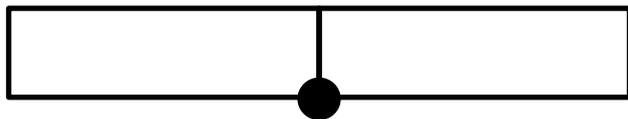
one-position model

Beschriftung von Punkten

fixed-position model



four-position model

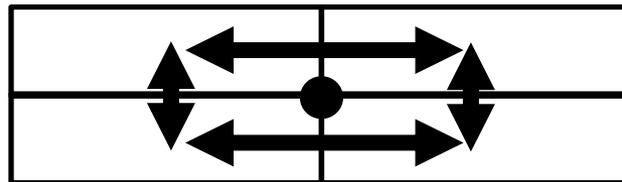


two-position model

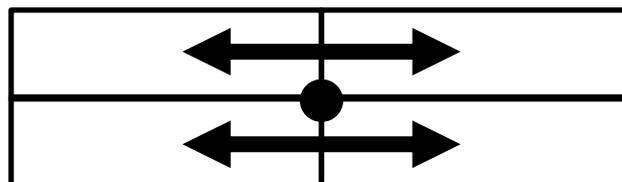


one-position model

slider model



four-slider model



two-slider model



one-slider model

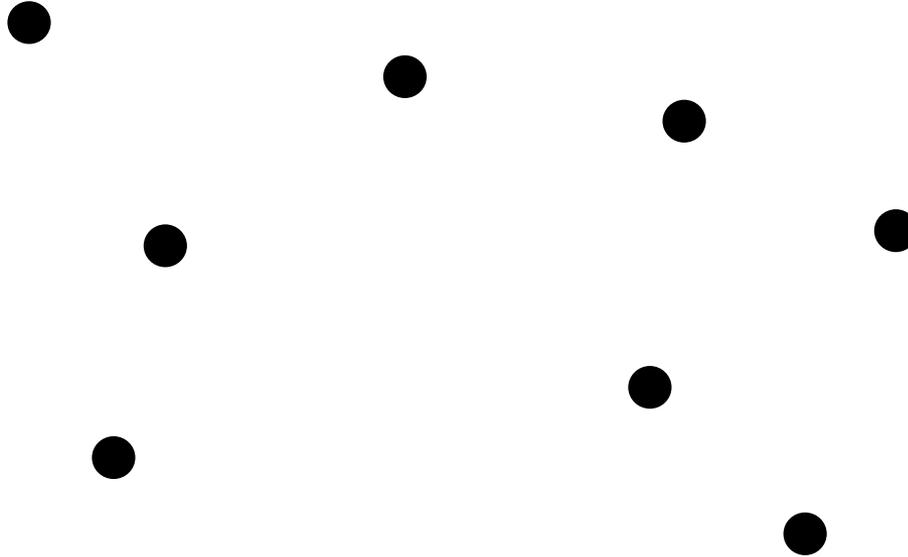
Würzburg



Würzburg



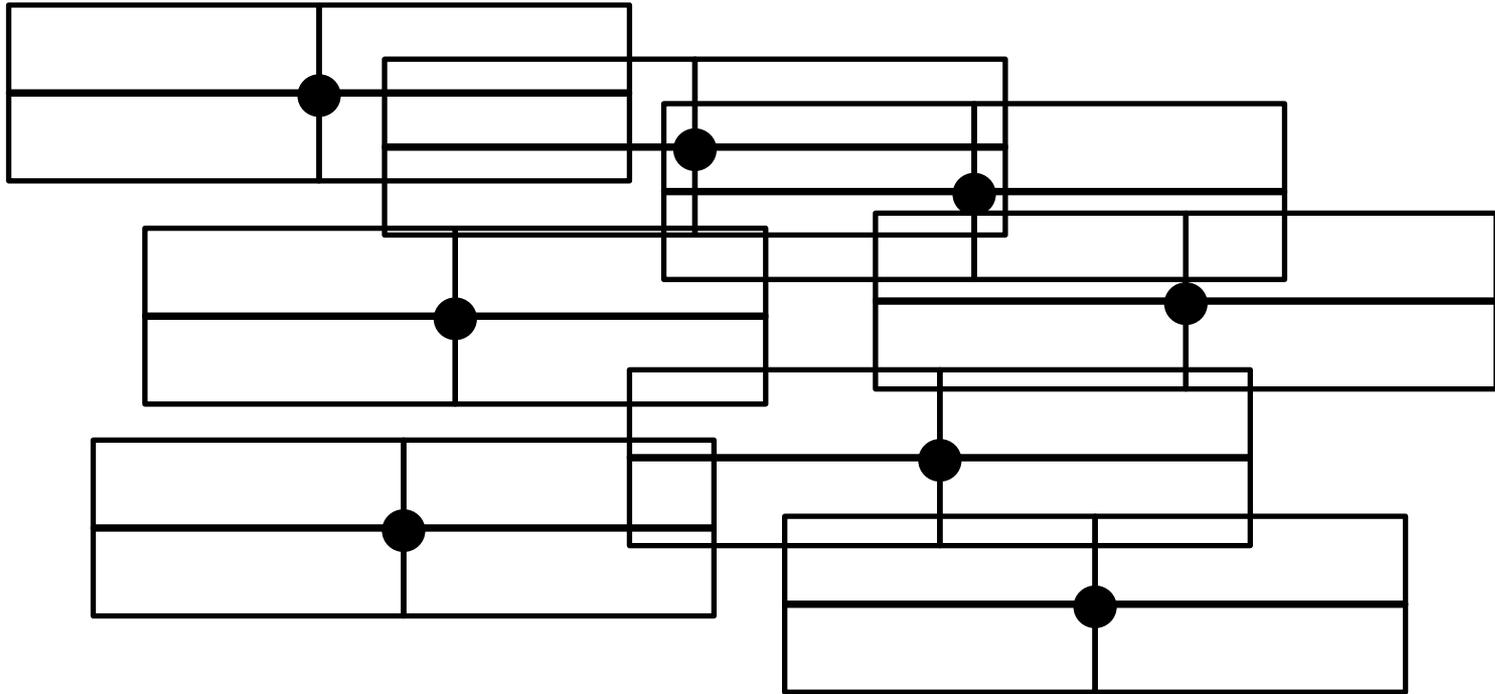
Beschriftung von Punkten



typisches Problem:

- Punktmenge gegeben

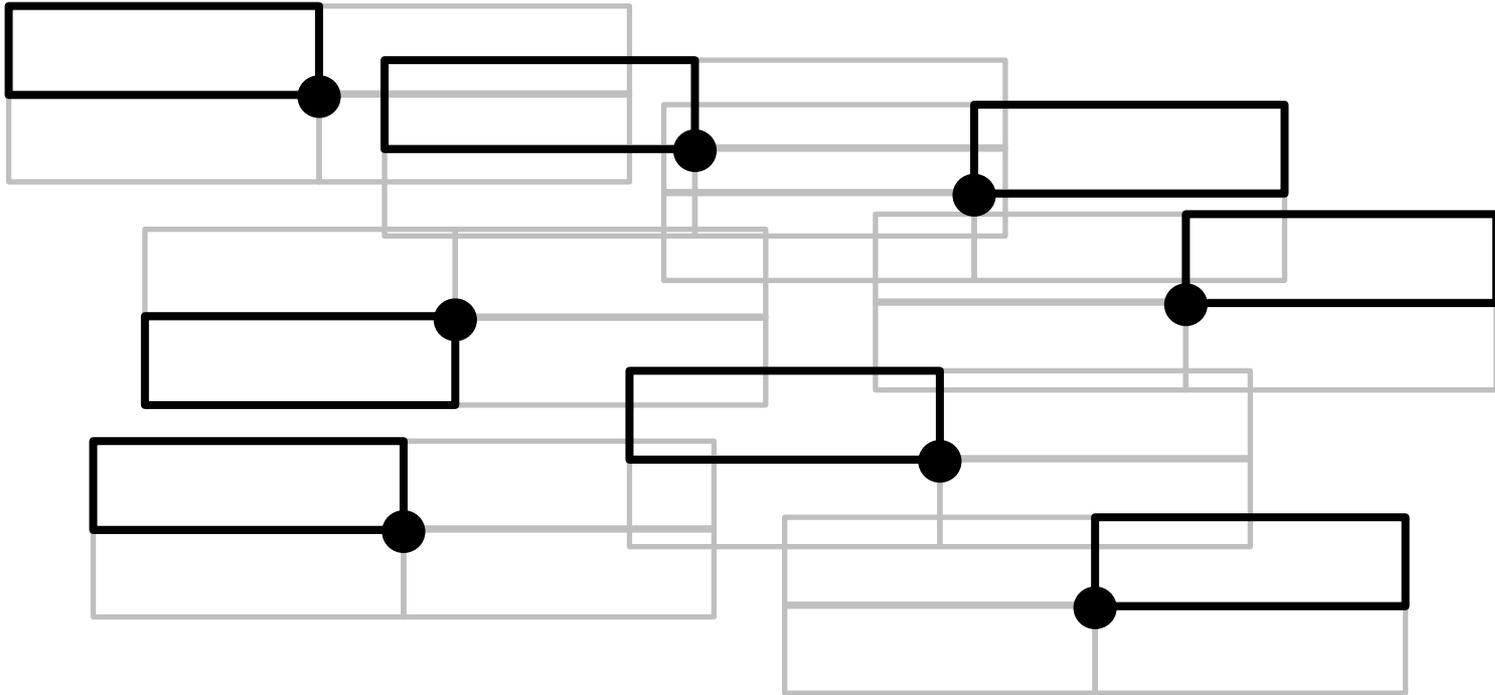
Beschriftung von Punkten



typisches Problem:

- Punktmenge gegeben
- Modell gewählt

Beschriftung von Punkten



typisches Problem:

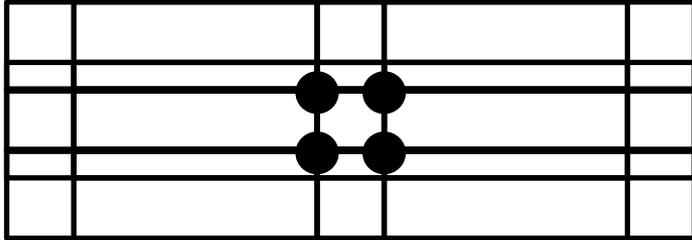
- Punktmenge gegeben
- Modell gewählt
- beschrifte **möglichst alle Punkte**, so dass sich Label nicht schneiden

Beschriftung von Punkten

typisches Problem:

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

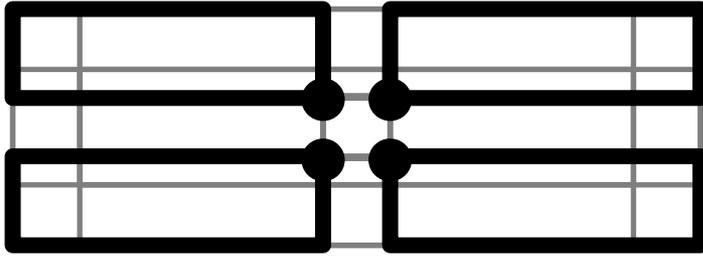
Beschriftung von Punkten



typisches Problem:

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

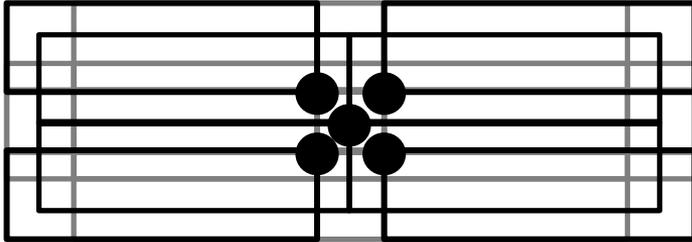
Beschriftung von Punkten



typisches Problem:

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

Beschriftung von Punkten

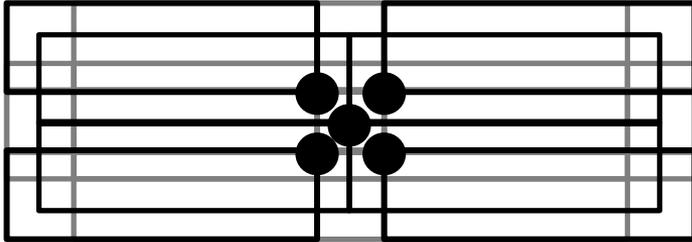


Bei fünf Punkten gibt es im four-position model
möglicherweise keine Lösung, in der alle Punkte
beschriftet werden!

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:

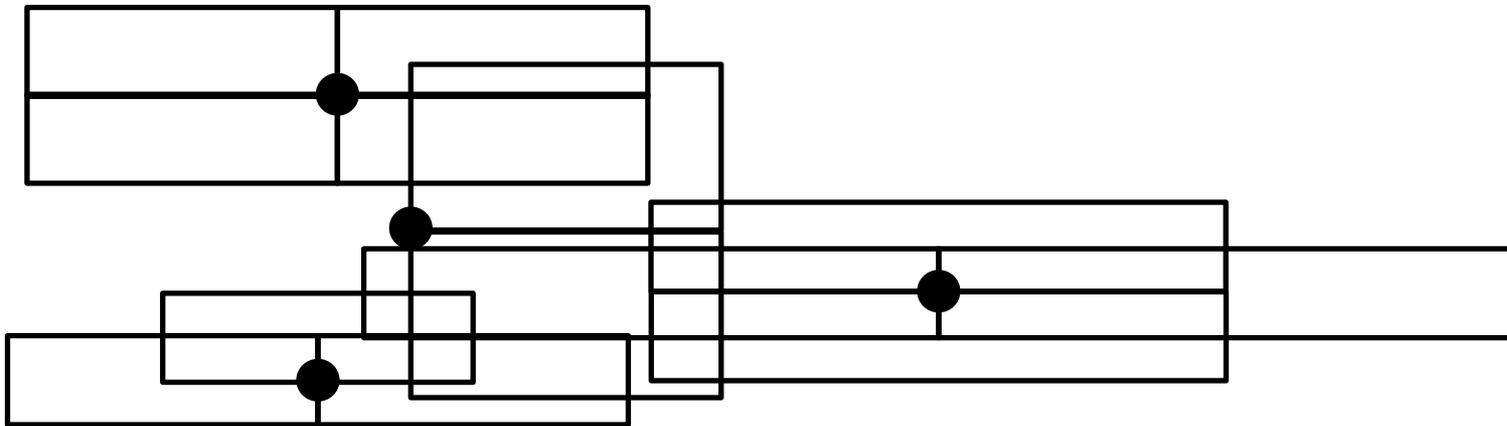
- Punktmenge gegeben
- Modell gewählt
- beschrifte **möglichst viele Punkte**, so dass sich Label nicht schneiden

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

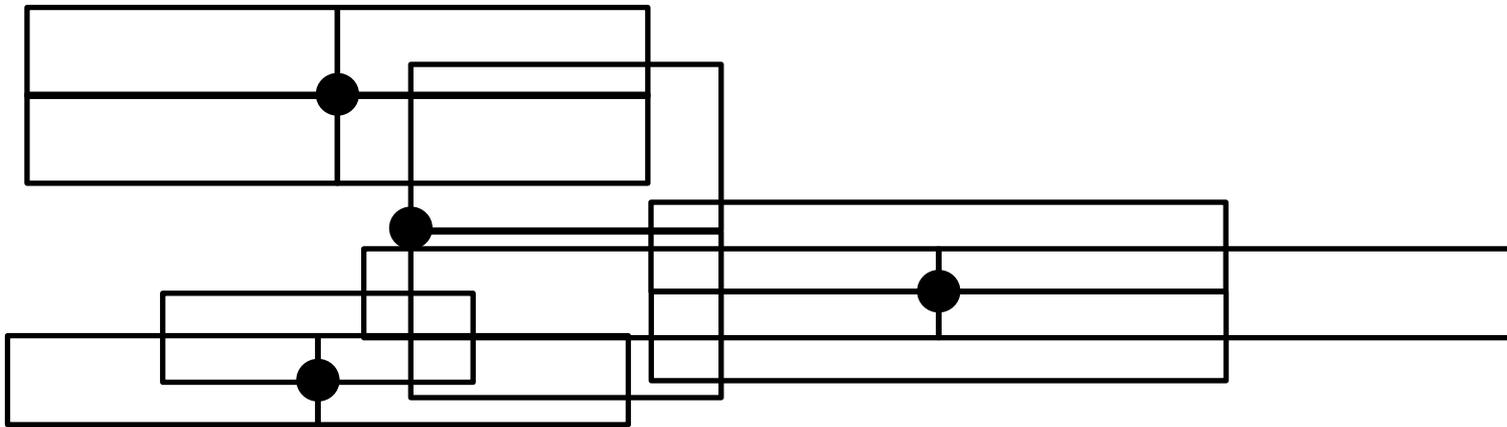


Beschriftung von Punkten

Allgemeiner Ansatz für fixed-position models:

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

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



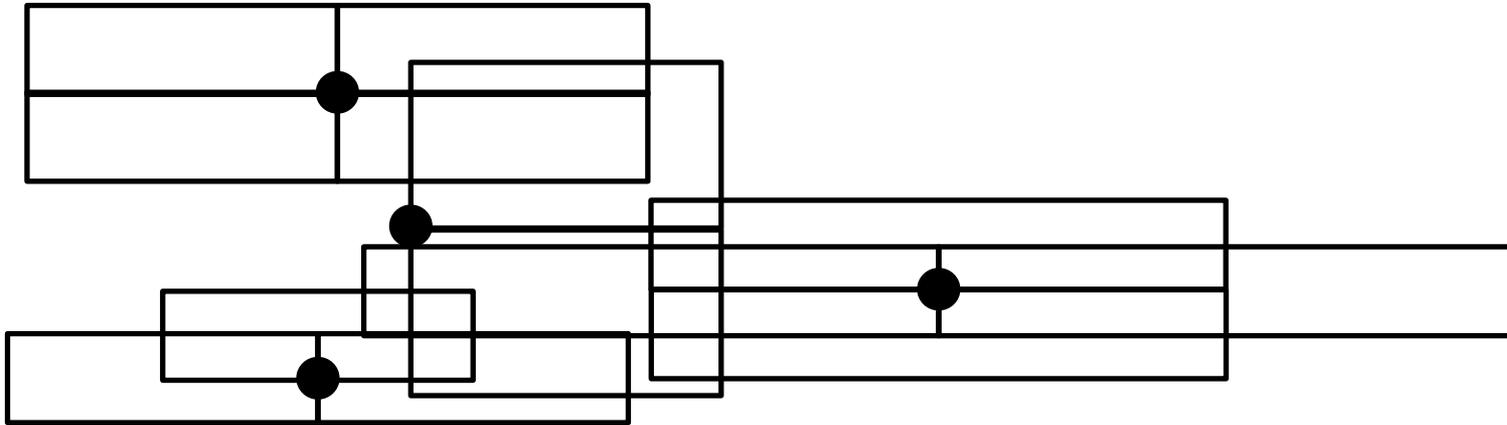
- gegeben Menge $R = R_1 \cup \dots \cup R_m$
- wähle **möglichst viele Rechtecke** aus R , die sich nicht schneiden

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 $i = 1, \dots, m$ 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 $R = R_1 \cup \dots \cup R_m$
- 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:

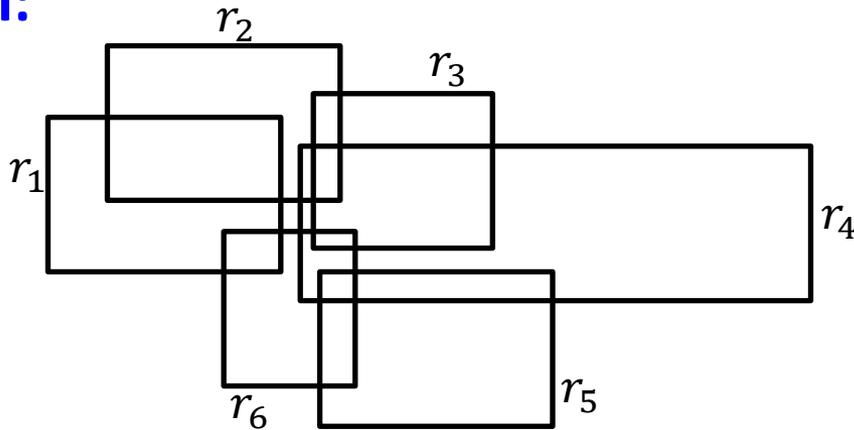
- gegeben Menge R von n achsparallelen Rechtecken
- wähle ***möglichst viele Rechtecke*** aus R , die sich nicht schneiden

Beschriftung von Punkten

Problem:

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

Allgemeinfall:

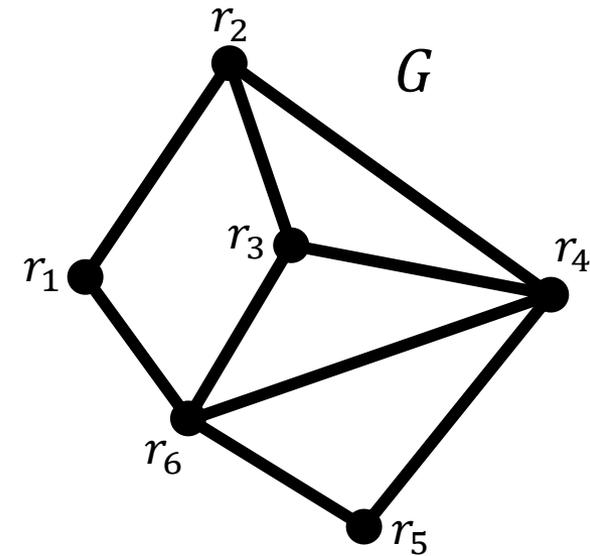
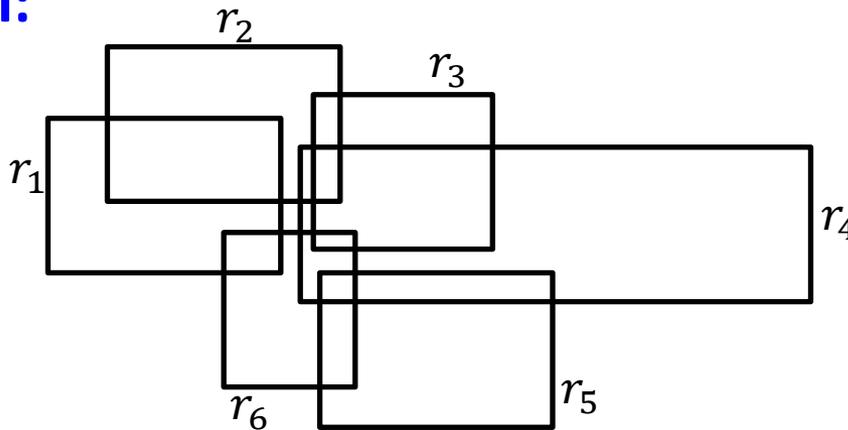


Beschriftung von Punkten

Problem:

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

Allgemeinfall:



Definition:

Der Schnittgraph $G = (R, E)$

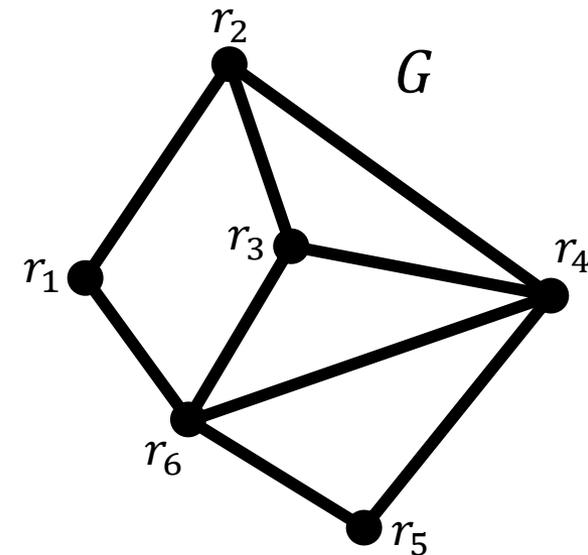
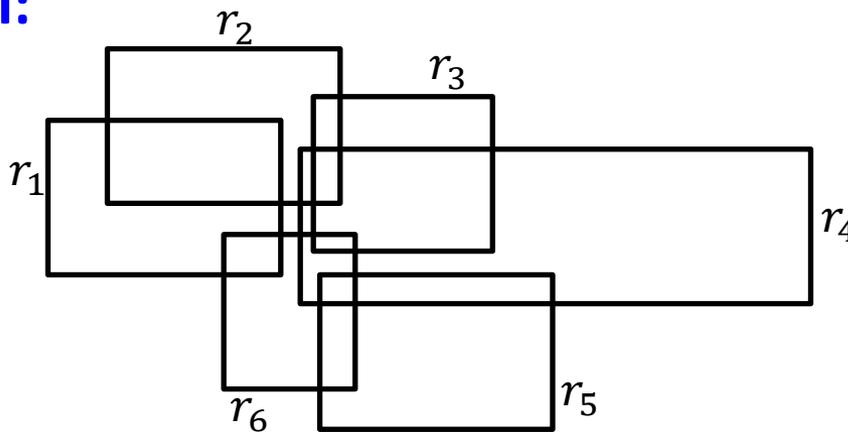
- enthält einen Knoten für jedes Rechteck und
- eine Kante $\{u, v\}$ wenn sich die Rechtecke u und v schneiden.

Beschriftung von Punkten

Problem:

- gegebene 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:

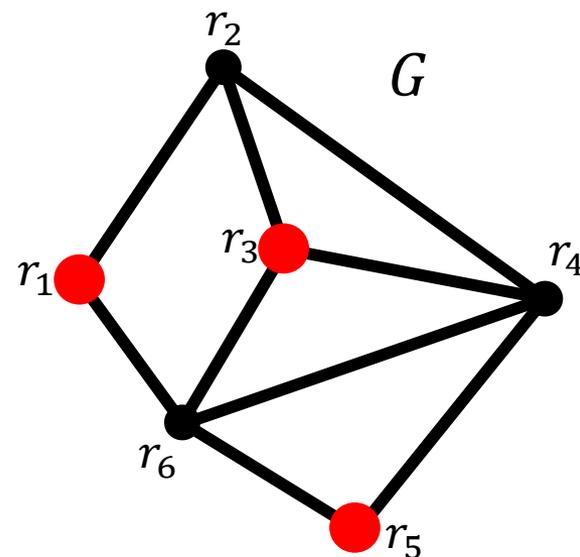
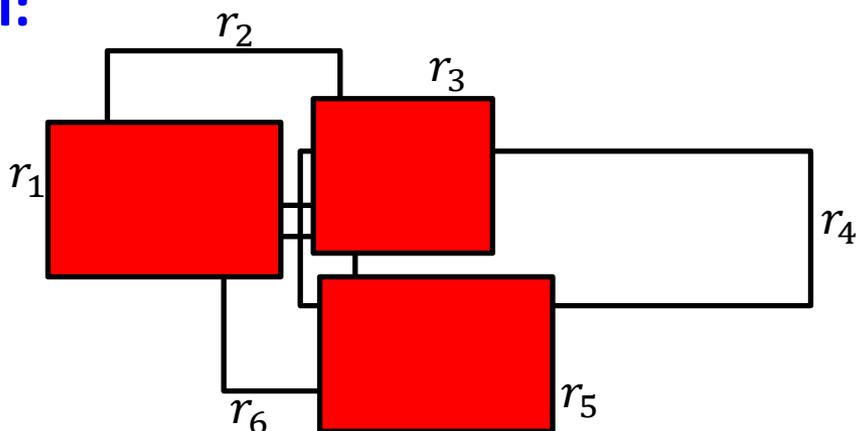
- Finde **größte unabhängige Menge** in G

Beschriftung von Punkten

Problem:

- gegebene 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:

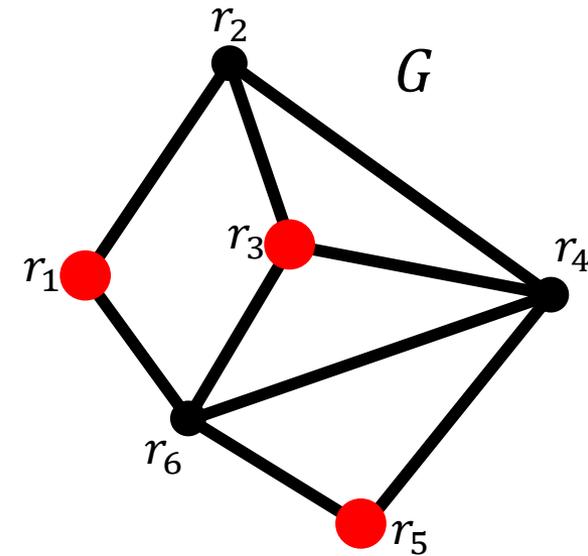
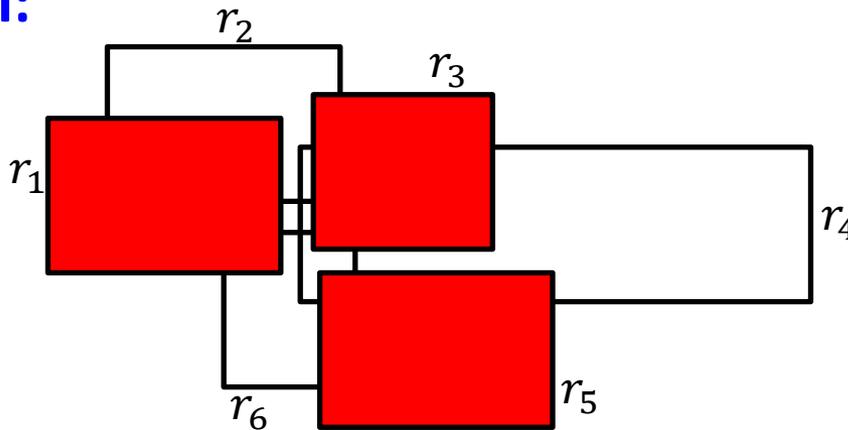
- Finde **größte unabhängige Menge** in G

Beschriftung von Punkten

Problem:

- gegebene 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:

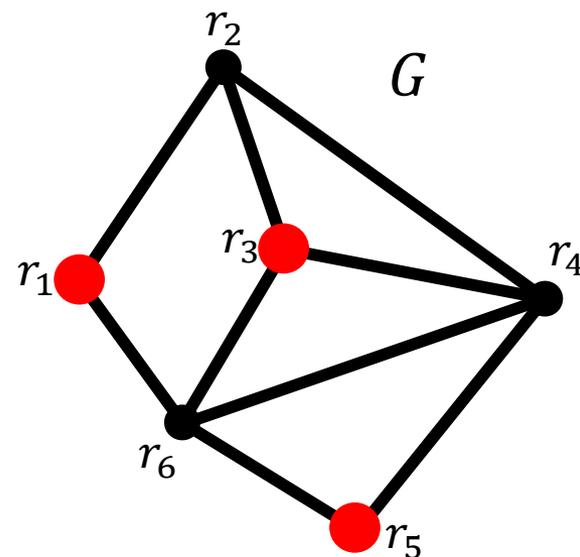
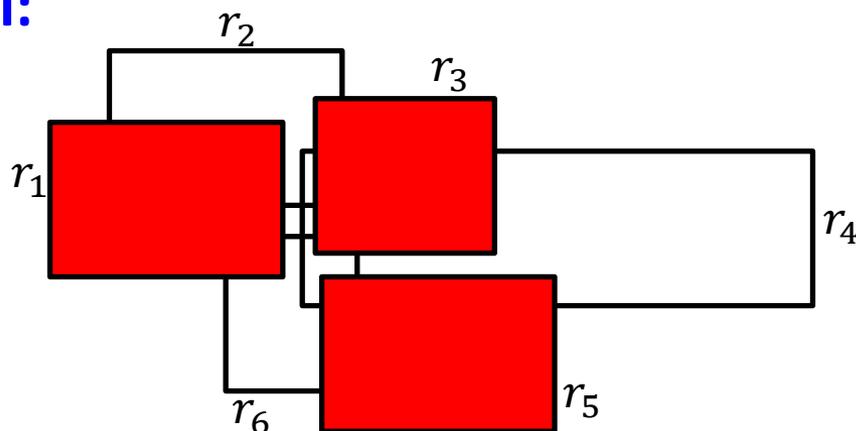
- Finde **größte unabhängige Menge** in G
für allgemeine Graphen NP-schwer!

Beschriftung von Punkten

Problem:

- gegebene 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

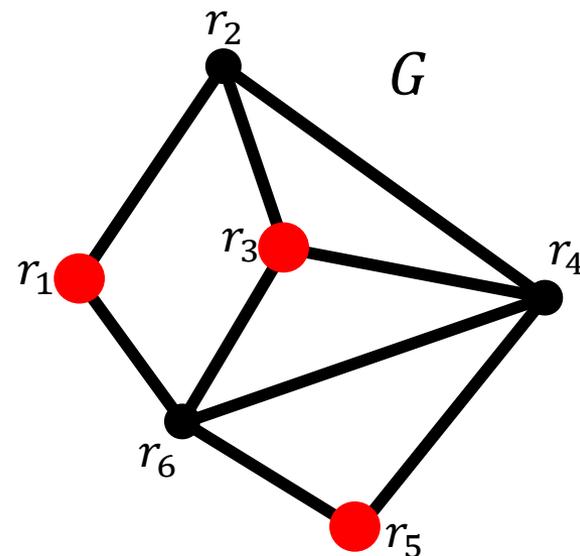
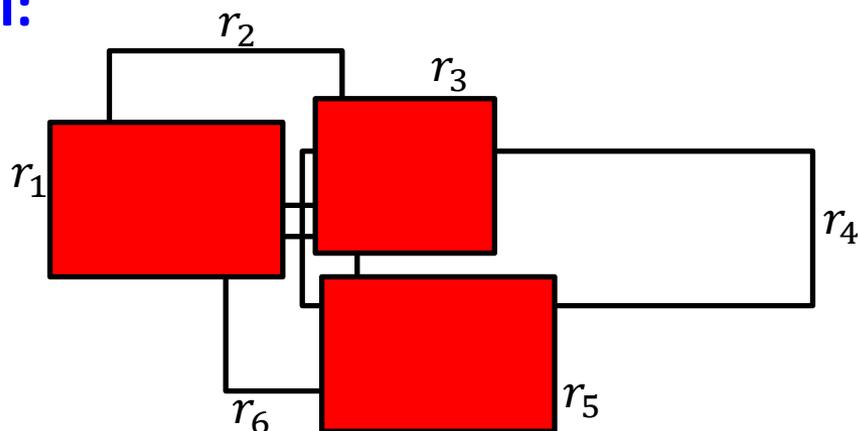
für Schnittgraphen von gleich großen Quadraten NP-schwer!
(Imai & Asano, 1983)

Beschriftung von Punkten

Problem:

- gegebene 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

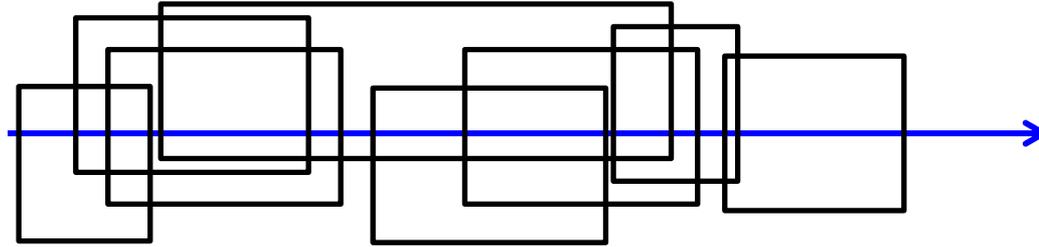
Beschriftung von Punkten

Problem:

- gegebene 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.



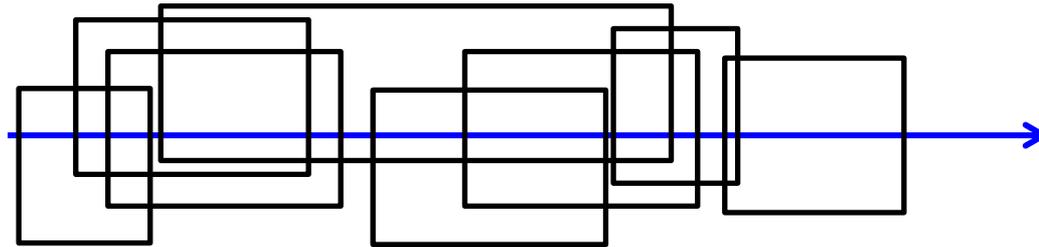
Beschriftung von Punkten

Problem:

- gegebene 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!

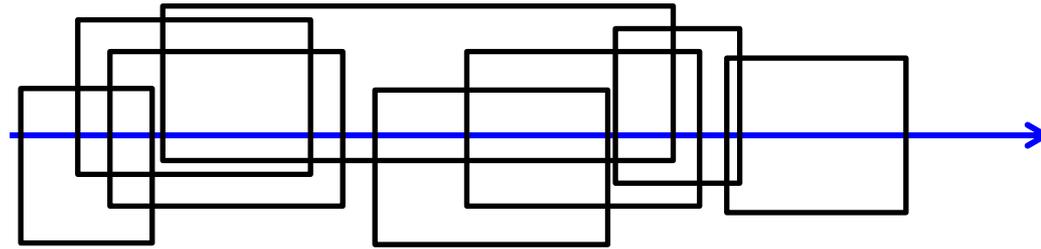
Beschriftung von Punkten

Problem:

- gegebene 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

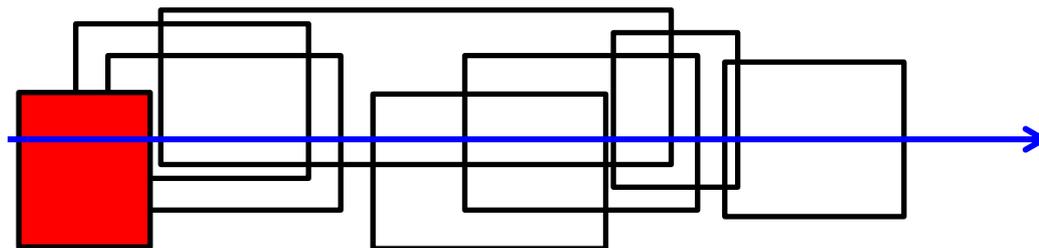
Beschriftung von Punkten

Problem:

- gegebene 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

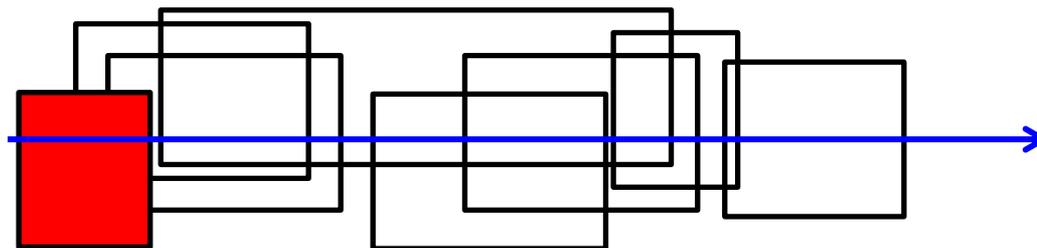
Beschriftung von Punkten

Problem:

- gegebene 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

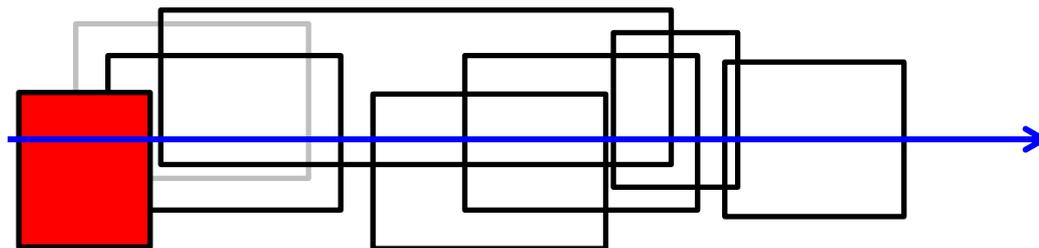
Beschriftung von Punkten

Problem:

- gegebene 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

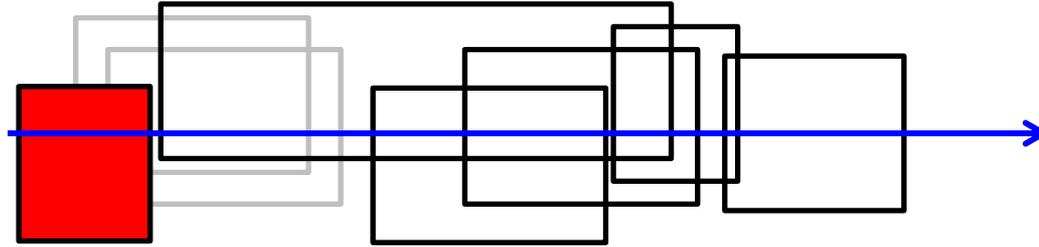
Beschriftung von Punkten

Problem:

- gegebene 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

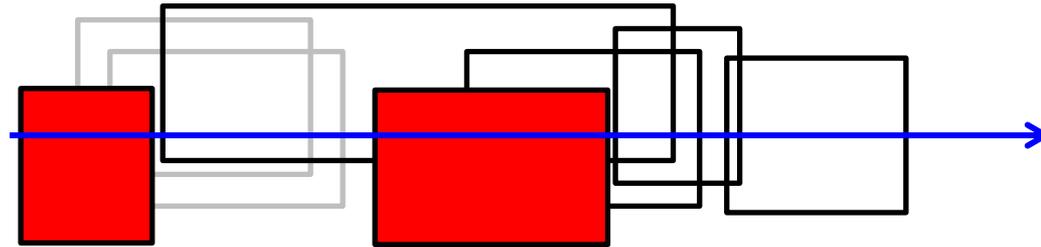
Beschriftung von Punkten

Problem:

- gegebene 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

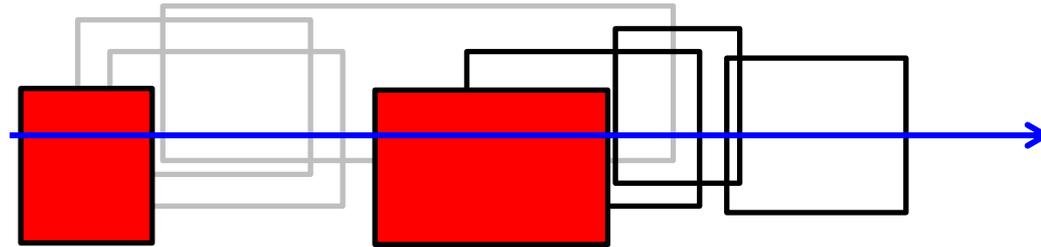
Beschriftung von Punkten

Problem:

- gegebene 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

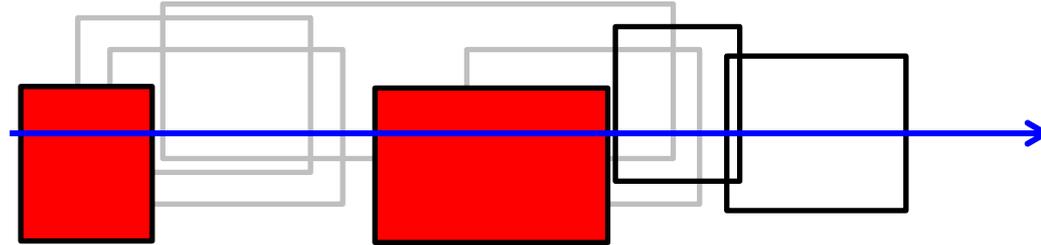
Beschriftung von Punkten

Problem:

- gegebene 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

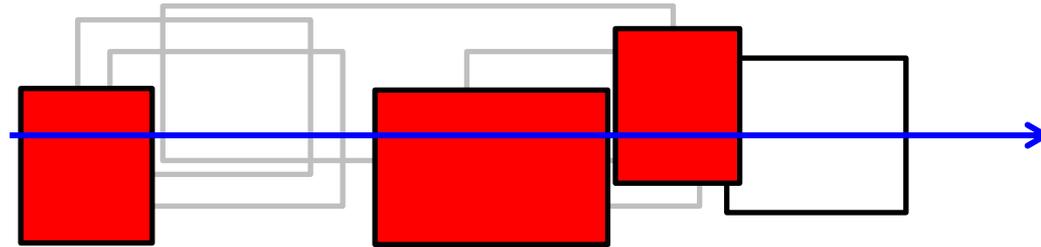
Beschriftung von Punkten

Problem:

- gegebene 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

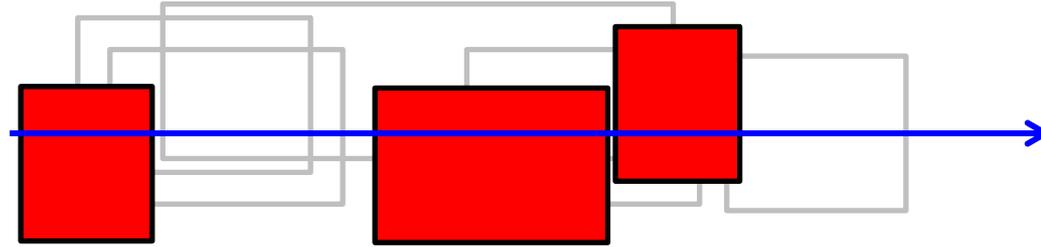
Beschriftung von Punkten

Problem:

- gegebene 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

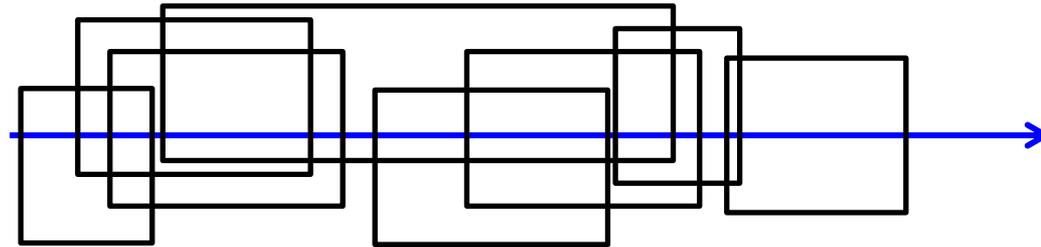
Beschriftung von Punkten

Problem:

- gegebene 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:

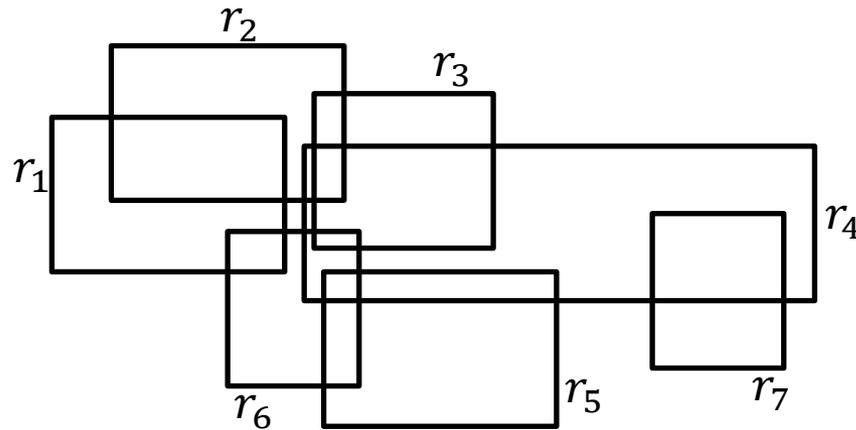
- $O(n \log n)$
- bzw. $O(n)$, wenn Eingabe bereits geordnet

Beschriftung von Punkten

Problem:

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

Allgemeinfall:

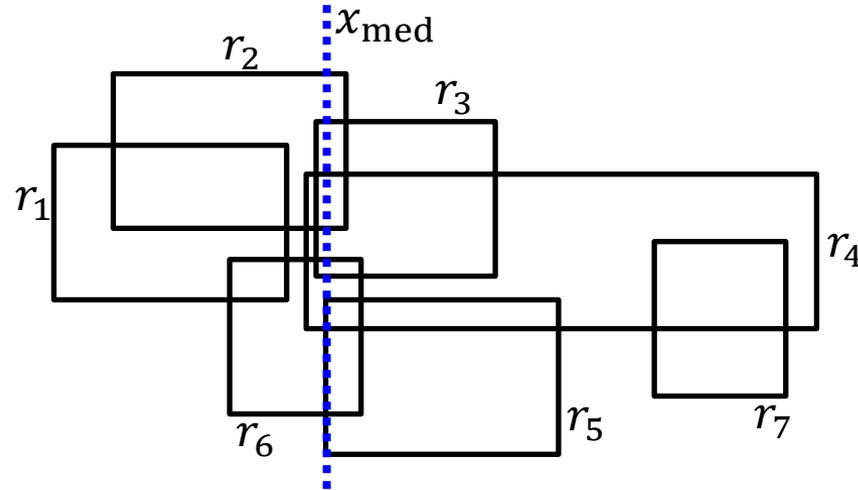


Beschriftung von Punkten

Problem:

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

Allgemeinfall:



Definitionen:

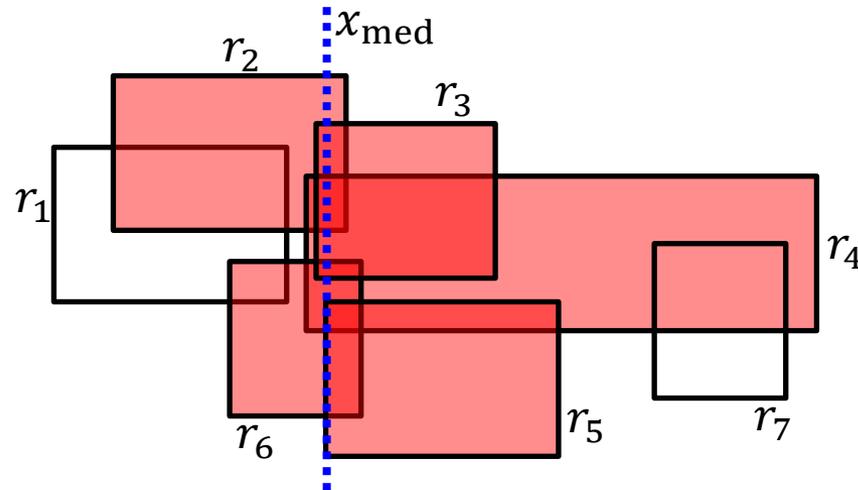
- Sei x_{med} der Median von $\{r.xMin, r.xMax \mid r \in R\}$.

Beschriftung von Punkten

Problem:

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

Allgemeinfall:



Definitionen:

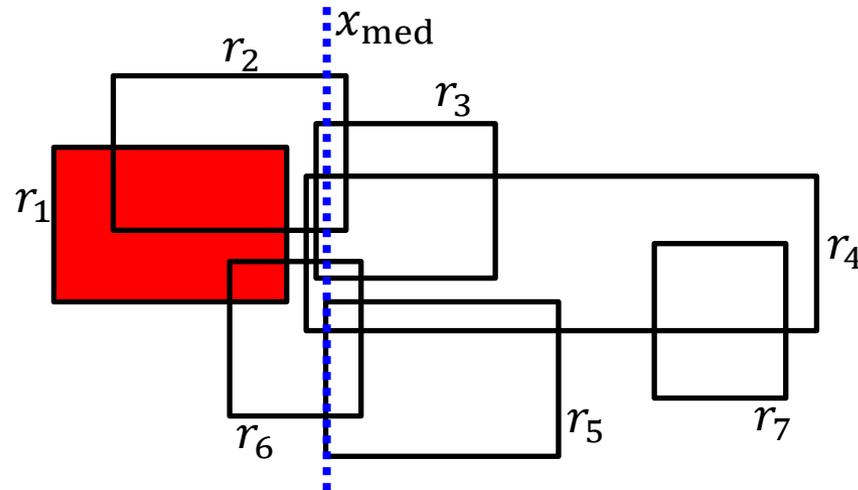
- Sei x_{med} der Median von $\{r.xMin, r.xMax \mid r \in R\}$.
- Sei $R_{12} \subseteq R$ die Menge der Rechtecke auf der Vertikalen durch x_{med} .

Beschriftung von Punkten

Problem:

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

Allgemeinfall:



Definitionen:

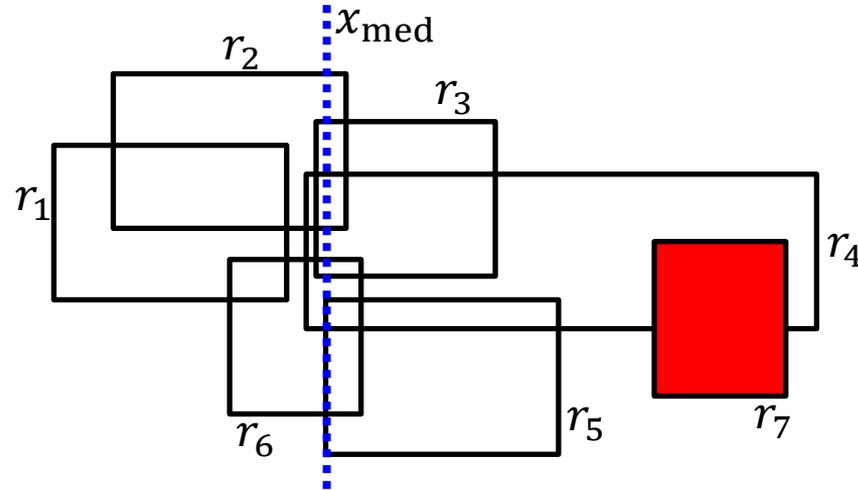
- Sei x_{med} der Median von $\{r.xMin, r.xMax \mid r \in R\}$.
- Sei $R_{12} \subseteq R$ die Menge der Rechtecke auf der Vertikalen durch x_{med} .
- Sei $R_1 \subseteq R$ die Menge der Rechtecke links der Vertikalen durch x_{med} .

Beschriftung von Punkten

Problem:

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

Allgemeinfall:



Definitionen:

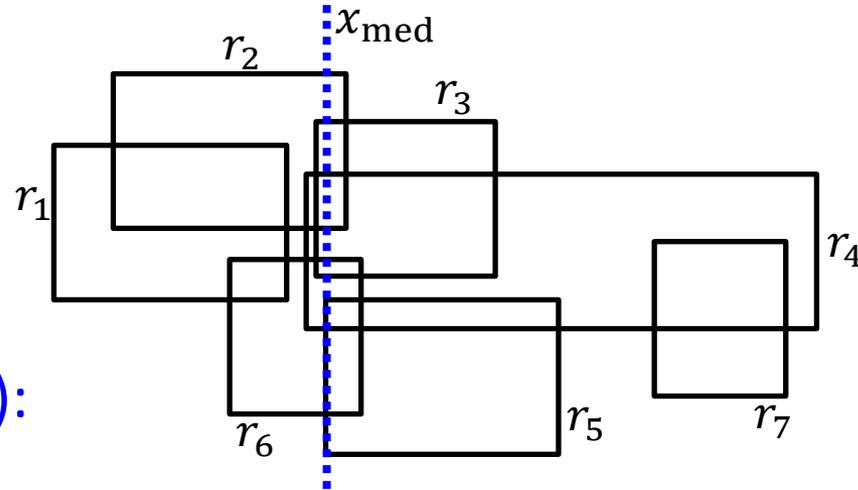
- Sei x_{med} der Median von $\{r.xMin, r.xMax \mid r \in R\}$.
- Sei $R_{12} \subseteq R$ die Menge der Rechtecke auf der Vertikalen durch x_{med} .
- Sei $R_1 \subseteq R$ die Menge der Rechtecke links der Vertikalen durch x_{med} .
- Sei $R_2 \subseteq R$ die Menge der Rechtecke rechts der Vertikalen 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 $\{r_1\}$

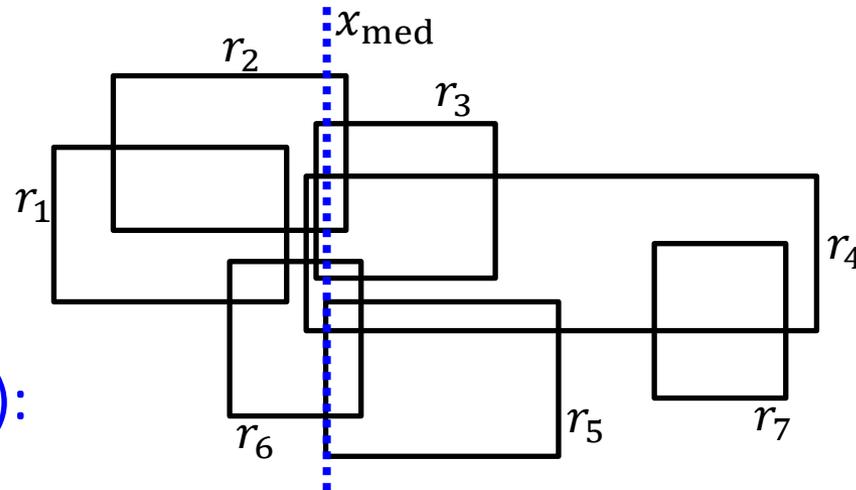
return $\{r_1, r_2\}$

Beschriftung von Punkten

Problem:

- gegebene 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

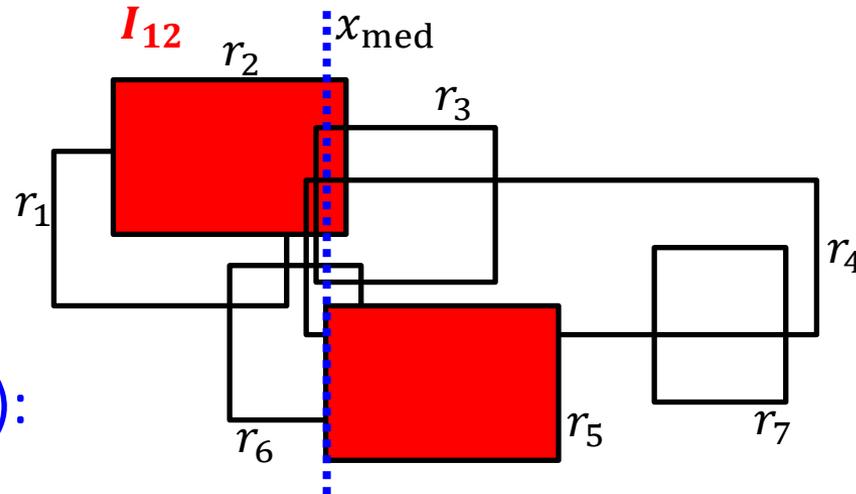
 | return $I_1 \cup I_2$

Beschriftung von Punkten

Problem:

- gegebene 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

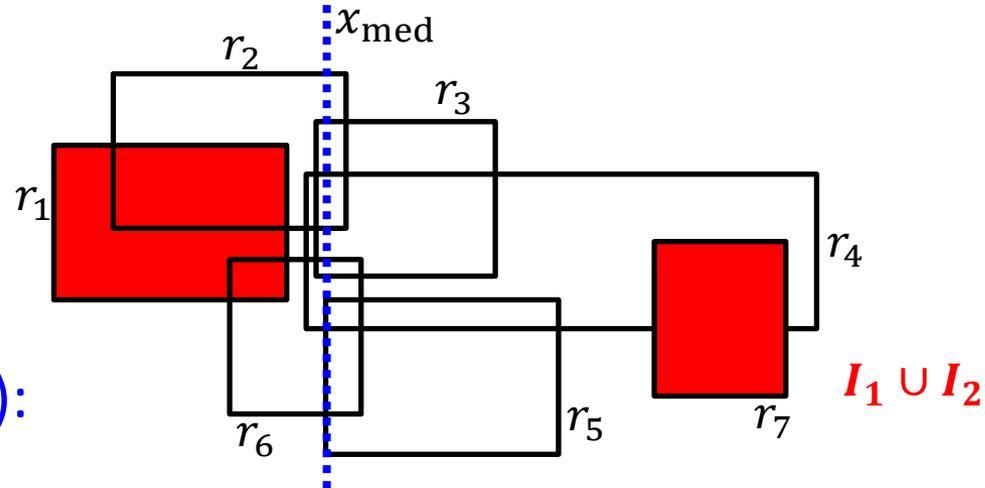
 | return $I_1 \cup I_2$

Beschriftung von Punkten

Problem:

- gegebene 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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$|I| \geq \max\left\{|I^* \cap R_{12}|, \frac{|I^*| - |I^*|/\log n}{\log n - 1}\right\}$$

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

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

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

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

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

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

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

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

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 $I_1 \cup I_2$

Beweis durch Induktion:

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

Also $|I| \geq \frac{|I^*|}{\log n}$



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:

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



Berechnung von R_{12}, R_1, R_2

Greedy-Algorithmus (Sortierung nur am Anfang erforderlich)

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:

$$T(n) = O(n \log n)$$